PHOTO EDITING ANDROID APPLICATION

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Mrs. Sanjana Singh

By

Ankit Mehta (111254)

To

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that project report entitled "**PHOTO EDITING ANDROID APPLICATION**", submitted by "**Ankit Mehta**" in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: 8/05/2015 **Sanjana Singh**

Assistant Professor

ACKNOWLEDGEMENT

It gives me immense pleasure in presenting project report on the topic **Photo Editing Android Application.** Apart from the efforts of me, the success of my project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I would like to show my greatest appreciation to my project in-charge, **Mrs. Sanjana Singh.** I can't say thank you enough for the tremendous support and help. I feel motivated and encouraged every time I attend her meeting. Without her encouragement and guidance this project work would not have materialized.

Date: 08/05/2015 Ankit Mehta

(111254)

TABLE OF CONTENTS

LIST OF FIGURES

S. No. **Title** **Page No**

LIST OF TABLES

LIST OF SYMBOLS & ACRONYMS

GUI – Graphical User Interface

ADT - Android Development Tools

IDE - Integrated Development Environment

AVD – Android Virtual Device

OEM – Original Equipment Manufacturer

DDMS - Dalvik Debug Monitor Server

ADB - Android Debug Bridge

SDK – Software Development Kit

ABSTRACT

Photo editing can be a challenging task, and it becomes even more difficult on the small, portable screens of mobile devices that are now frequently used to capture and edit images. To address this problem I present Photo Editor, a photo editing interface for direct manipulation.

Through this application user can easily and quickly edit there pictures with the help of the features provided in the application.

Some of the features of the application are: - One tap Auto Enhance, Ability to Crop, rotate and straighten your photo, Adjust brightness, contrast and saturation, adding effects like blur, snowy, emboss, engrave, etc.

All the coding has been done in JAVA language using a plugin of Eclipse IDE i.e. Android Development Tools (ADT).

ADT is designed to provide an integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let developers set up new Android projects, create an application UI, add packages based on the Android Framework API, debug their applications using the Android SDK tools, and export signed (or unsigned) .apk files in order to distribute their applications. It is a freeware available to download.

Through this software we can run the code either directly in our android device or by using AVD (Android virtual device) manager to create an AVD.

CHAPTER 1 INTRODUCTION

1.1. General Introduction

The project named "**PHOTO EDITING ANDROID APPLICATION**" is developed using Eclipse IDE and Android SDK manager using JAVA language. This project has been developed in partial fulfillment of Requirements for the degree of B.TECH.(CSE) from JUIT, Waknaghat.

Developed for busy customers, Photo Editor is an easy to use Android application through which customers can easily edit photos using various features provided in the application. Its graphical user interface is designed in a manner to attract wide variety of people varying from age group of 18 years to 55 years.

The editing methods are optimized in such a way so that user can quickly and easily edit there photographs without wasting much of their time. Hence it's quick, easy and efficient.

1.2. Problem statement

To develop a Photo editing Android application with a quick and easy way to edit photos having both basic and advance level features to edit your photographs.

CHAPTER 2 WORKING ENVIRONMENT

2.1. About Android

Android is an open source operating system, created by Google, and available to all kinds of developers with various expertise levels, ranging from rookie to professional.

From a developer's perspective, Android is a Linux-based operating system for smartphones and tablets. It includes a touch screen user interface, widgets, camera, network data monitoring and all the other features that enable a cell phone to be called a smartphone. Android is a platform that supports various applications, available through the Android Play Store. The Android platform also allows end users to develop, install and use their own applications on top of the Android framework. The Android framework is licensed under the Apache License, with Android application developers holding the right to distribute their applications under their customized license.

2.2. Understanding android

To begin development on Android even at the application level, it is paramount to understand the basic internal architecture. Knowing how things are arranged inside helps us understand the application framework better, so we can design the application in a better way.

Android is an OS based on Linux. Hence, deep inside, Android is pretty similar to Linux. To understand Android internals, let us look at an architectural diagram.

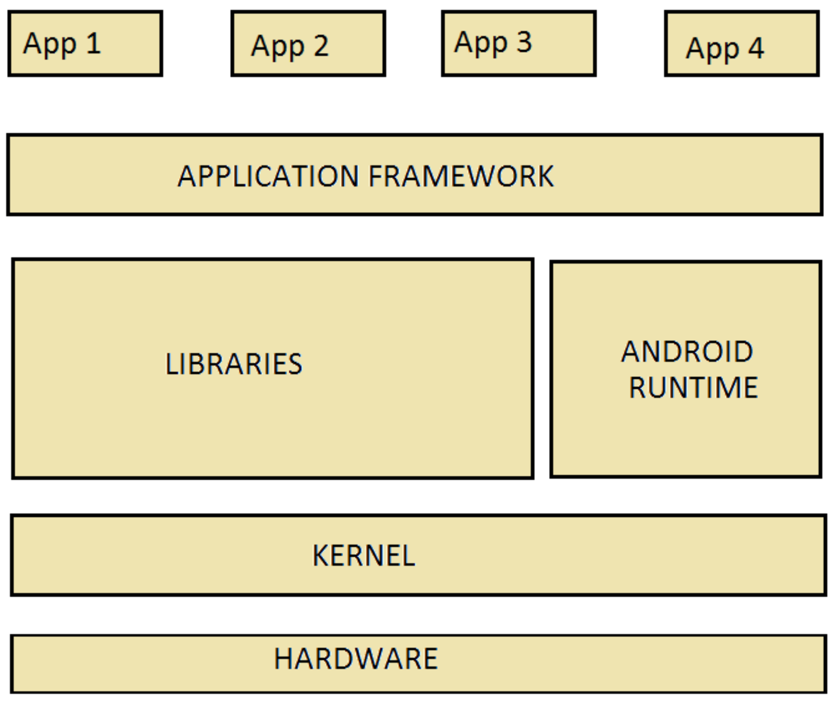


Fig.1.1

Source:http://www.cprogramming.com/android/android_getting_started.html

The above diagram illustrates the Android architecture. As you can see, it is a software stack above the hardware that is provided by the OEMs. Let's start with the topmost layer, i.e., the applications.

2.2.1 Applications

The diagram shows four basic apps (App 1, App 2, App 3 and App 4), just to give the idea that there can be multiple apps sitting on top of Android. These apps are like any user interface you use on Android; for example, when you use a music player, the GUI on which there are buttons to play, pause, seek, etc is an application. Similarly, is an app for making calls, a camera app, and so on. All these apps are not necessarily from Google. Anyone can develop an app and make it available to everyone through Google Play Store. These apps are developed in Java, and are installed directly, without the need to integrate with Android OS.

2.2.2 Application Framework

Scratching further below the applications, we reach the application framework, which application developers can leverage in developing Android applications. The framework offers a huge set of APIs used by developers for various standard purposes, so that they don't have to code every basic task. The framework consists of certain entities; major ones are:

1. Activity Manager

This manages the activities that govern the application life cycle and has several states. An application may have multiple activities, which have their own life cycles. However, there is one main activity that starts when the application is launched. Generally, each activity in an application is given a window that has its own layout and user interface. An activity is stopped when another starts, and gets back to the window that initiated it through an activity callback.

2. Notification Manager

This manager enables the applications to create customized alerts

3. Views

Views are used to create layouts, including components such as grids, lists, buttons, etc.

4. Resource Managers

Applications do require external resources, such as graphics, external strings, etc. All these resources are managed by the resource manager, which makes them available in a standardized way.

5. Content Provider

Applications also share data. From time to time, one application may need some data from another application. For example, an international calling application will need to access the user's address book. This access to another application's data is enabled by the content providers.

2.2.3 Libraries

This layer holds the Android native libraries. These libraries are written in C/C++ and offer capabilities similar to the above layer, while sitting on top of the kernel. A few of the major native libraries include

- Surface Manager: manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.
- System C Libraries: Standard C library like libc targeted for ARM or embedded devices.
- OpenGL ES Libraries : These are the graphics libraries for rendering 2D and 3D graphics.
- SQLite : A database engine for Android.

2.2.4 Android Runtime

The Android runtime consists of the Dalvik Virtual Machine. It is basically a virtual machine for embedded devices, which like any other virtual machine, is a bytecode interpreter. When we say it is for embedded devices, it means it is low on memory, comparatively slower and runs on battery power. Besides the Dalvik Virtual Machine, it also consists of the core libraries, which are Java libraries and are available for all devices.

2.2.5 Kernel

The Android OS is derived from Linux Kernel 2.6 and is actually created from Linux source, compiled for mobile devices. The memory management, process management etc. are mostly similar. The kernel acts as a Hardware Abstraction Layer between hardware and the Android software stack.

CHAPTER 3 LITERATURE SURVEY

3.1. Summary of Papers

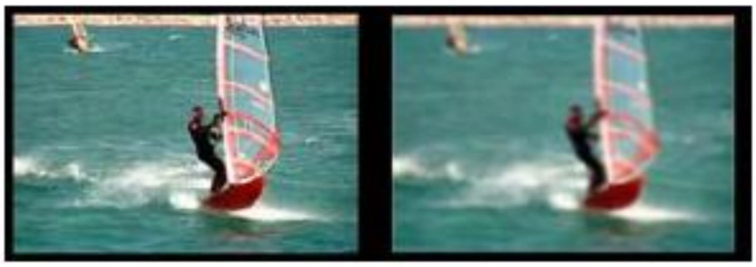
Title of Paper	Image Convolution
Author	Jamie Ludwig
Year of Publication	2013
Publishing Details	Satellite Digital Image Analysis, 581 Portland State University
Summary	This paper focuses on Image Convolution method in editing photographs. Through Image Convolution various editing effects like Smooth, Sharpen, Intensify and Enhancement of image etc is achieved. Convolution is a general purpose filter effect for images. It works by determining the value of a central pixel by adding the weighted values of all its neighbors together. The output is a new modified filtered image. A convolution is done by multiplying a pixel's and its neighboring pixels color value by a matrix. A kernel matrix is used for that purpose. A **kernel** is a (usually) small matrix of numbers that is used in image convolutions. Differently sized kernels containing different patterns of numbers produce different results under convolution. The size of a kernel is arbitrary but 3x3 is often used.
Web Link	http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Ludwig_ImageConvolution.pdf

1	1	1
1	1	1
1	1	1

Unweighted 3x3 smoothing kernel

0	1	0
1	4	1
0	1	0

Weighted 3x3 smoothing kernel with Gaussian blur



Gaussian Blur

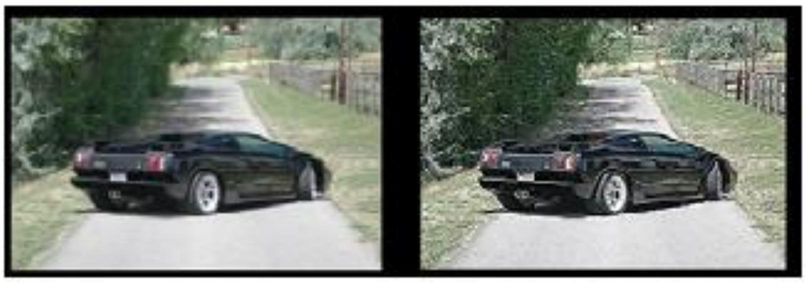
Fig.1.2

0	-1	0
-1	5	-1
0	-1	0

Kernel to make image sharper

-1	-1	-1
-1	9	-1
-1	-1	-1

Intensified sharper image



Sharpened image

(a) Fig.1.2 (b)

Title of Paper	Shift-Map Image Editing
Authors	Yael Pritch, Eitam Kav-Venaki and Shmuel Peleg.
Year of Publication	2009
Publishing Details	IEEE 12th International Conference on Computer Vision (ICCV)
Summary	This paper focuses on Shift-Map Image editing method which is done using geometric rearrangement. Geometric rearrangement of images includes operations such as image retargeting, inpainting, or object rearrangement. Each such operation can be characterized by a shift map: The relative shift of every pixel in the output image from its source in an input image. They described a new representation of these operations as an optimal graph labeling, where the shift-map represents the selected label for each output pixel. Two terms were used in computing the optimal shift-map: (i) A data term which indicated constraints such as the change in image size, object rearrangement, a possible saliency map, etc. (ii) A smoothness term, which minimized the new discontinuities in the output image caused by discontinuities in the shift-map. This graph labeling problem could be solved using graph cuts. Efficient hierarchical solutions for graph-cuts were presented, and operations on 1M images could take only a few seconds.
Web Link	www.cs.huji.ac.il/~peleg/papers/iccv09-shiftmap.pdf

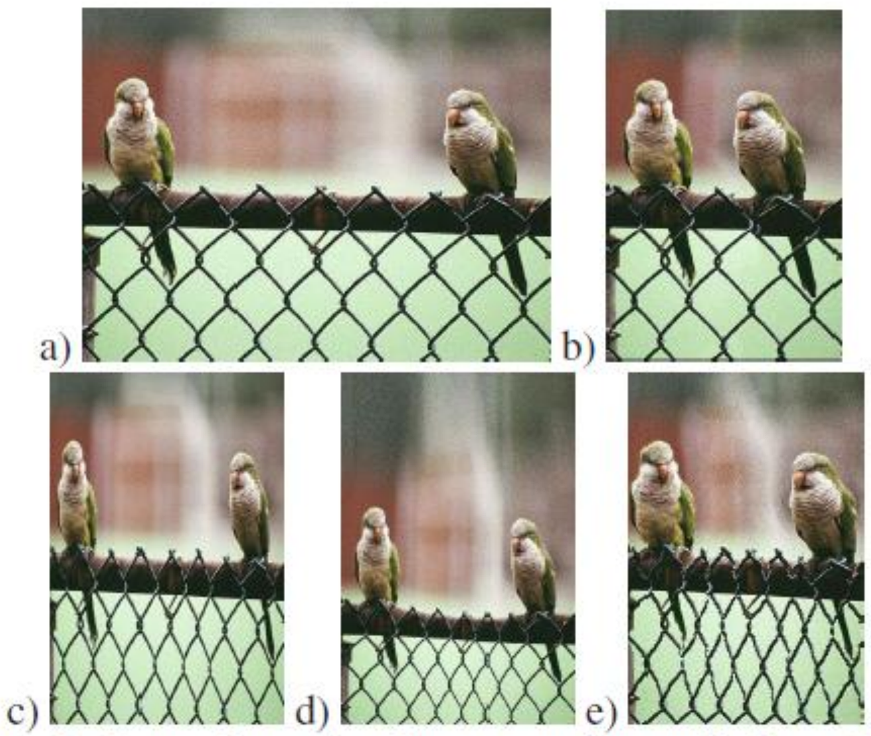


Figure 2. Comparison of a few retargeting methods, reducing width by half. (a) Original image. (b) **Our shift-map editing**. (c) Video-retargeting [19]; (d) Optimized scale-and-stretch [16]; (e) Improved Seam Carving [13];

Fig.1.3 Comparison of a few retargeting methods

Title of Paper	Photo editing algorithm changes weather, seasons automatically
Authors	Hays, Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, and Chao Qian.
Year of Publication	2014
Publishing Details	Kevin Stacey, Brown University, Providence, Rhode Island.
Summary	This paper focuses on a computer algorithm being developed by Brown University researchers, which enables the users to instantly change the weather, time of day, season, or other features in outdoor photos with simple text commands. All this is made possible with machine learning and a clever database. To start the project, Hays and his team defined a list of transient attributes that users might want to edit. They settled on 40 attributes that range from the simple (cloudy, sunny, snowy, rainy, or foggy) to the subjective (gloomy, bright, sentimental, mysterious, or calm). The next step was to teach the algorithm what these attributes look like. To do that, the researchers compiled a database consisting of thousands of photos taken by 101 stationary webcams around the world. The cameras took pictures of the same scenes in varying of conditions — different times of day, different seasons and in all kinds of weather. The researchers then asked workers on Mechanical Turk, to annotate more than 8,000 photos according to which of the 40 attributes are present in each. Those annotated photos were then fed through a machine learning algorithm. Armed with the knowledge of what each attribute looks like, the algorithm can apply that knowledge to new photos. It does so by making what Hays refers to as "local color transforms."

	It splits the picture into regions (clusters of pixels) and draws on the database to determine how colors in those regions should change with a given attribute. The participants preferred the new results around 70 percent of the time compared to the output of traditional approaches. The paper also mentioned the limitations to what the program can do at this point, however. It can't reproduce attributes that require new structures to be added to the photo.
Web Link	https://news.brown.edu/articles/2014/08/photo

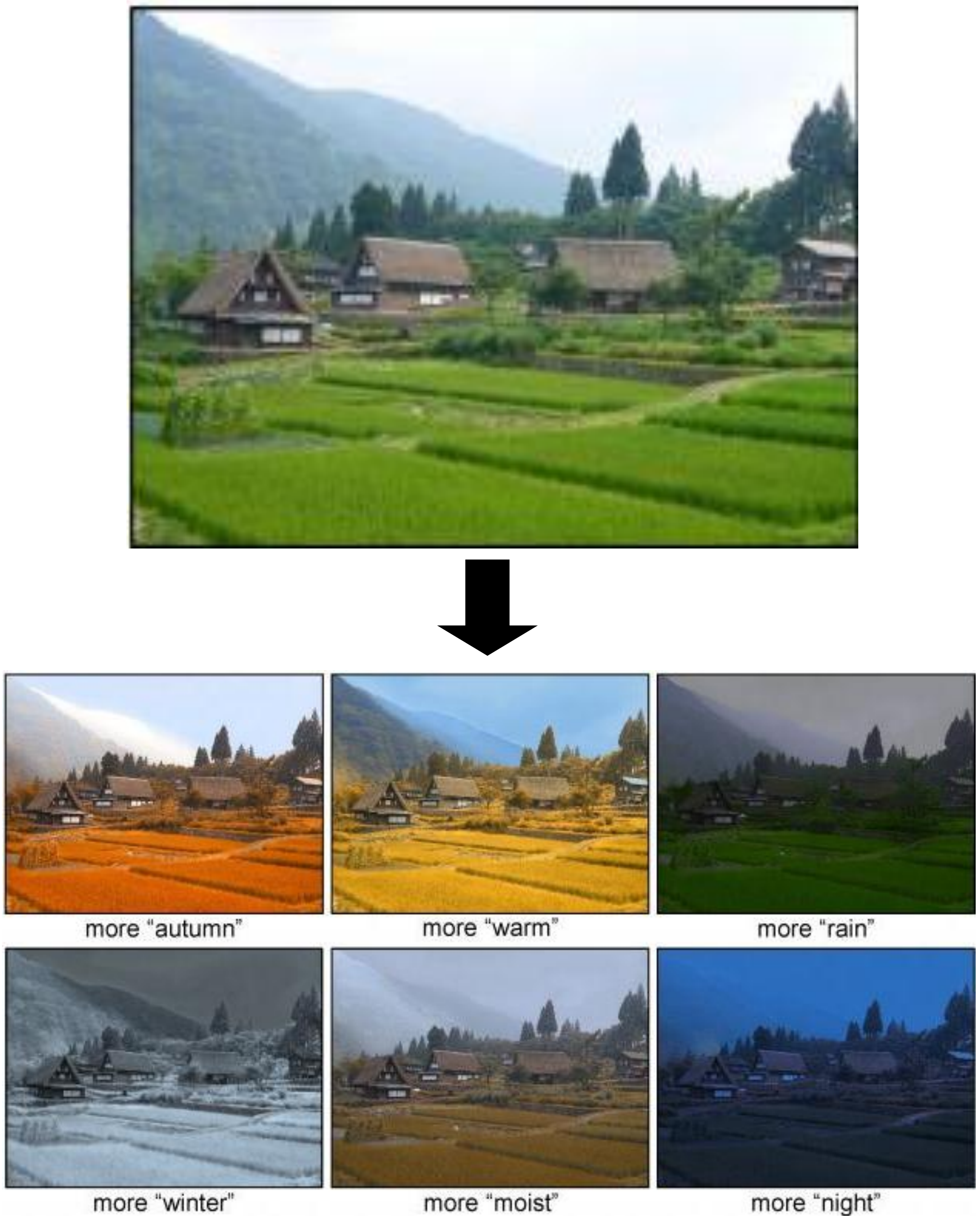


Fig.1.4 Changing weather

3.2. Terminology and Terms Used

The terminology related to the use of some terms in the above mentioned papers has been described briefly for the proper understanding of the basics and to get proper knowledge.

3.2.1 Image Editing as Graph Labeling

The relationship between an input image $I(x, y)$ and an output image $R(u, v)$ in image rearrangement and retargeting is defined by a shift-map $M(u, v) = (tx, ty)$. The output pixel $R(u, v)$ will be derived from the input pixel $I(u + tx, v + ty)$. The optimal shift-map is defined as a graph labeling, where the nodes are the pixels of the output image, and each output pixel can be labeled by a shift (tx, ty). The optimal shift-map M minimizes the following cost function:

$$E(M) = \alpha \sum_{p \in R} E_d(M(p)) + \sum_{(p,q) \in N} E_s(M(p), M(q))$$

where E_d is a data term providing external requirements, and E_s is a smoothness term defined over neighboring pixels N. α is a user defined weight balancing the two terms, and in all our examples we used $\alpha = 1$. Each term will now be defined in detail. Once the graph is given, the shift-map labeling is computed using multi-label graph cuts.

3.2.2 Image Retargeting

Image retargeting is the change of image size, which is typically done in only a single direction in order to change the image aspect ratio. We will assume that the change is in image width, but we could also address changing both image dimensions.

3.2.3 Image Rearrangement

Image rearrangement consists of moving an object to a new image location, or deleting part of the image, while keeping some of the content of the image unchanged. The user selects a region to move, and specifies the location at which the selected region will be placed. A new image is generated satisfying this constraint. This application was demonstrated and gave impressive results in many cases.

In image rearrangement pixels can be relocated by a large displacement, creating a possible computational complexity.

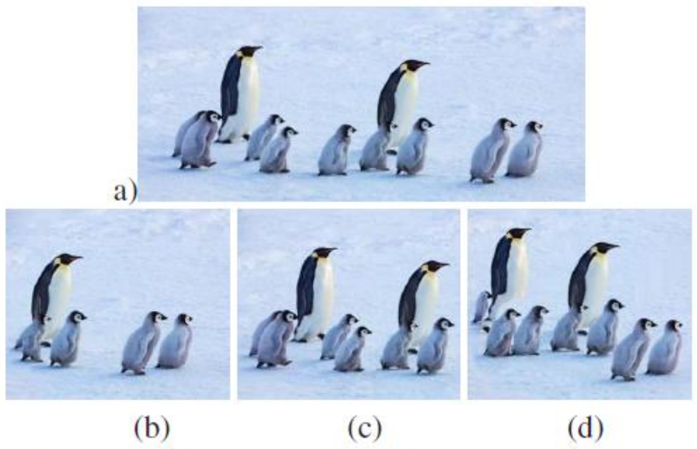


Figure 4. Controlling object removal by changing the number of steps. (a) Original image. (b) Resizing in a single step may cut out some of the objects. (c) Six smaller resizing steps remove fewer objects. (d) Ten even smaller steps remove even fewer objects. Note that in order to fit more objects in a smaller image, the result in (d) has vertical shifts introduced automatically by the optimization process.

Fig.1.5

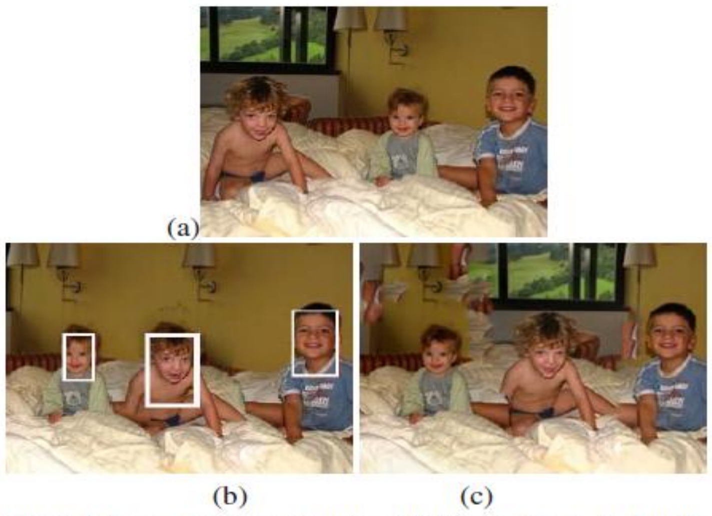


Figure 7. Image Rearrangement: (a) Original image. Kid on the left should move to the center, baby should move to the left, kid on the right should remain in place. (b) **Shift-map results** with user constraints marked on top. (c) Patch transform results on the same input.

Fig.1.6

Source: www.cs.huji.ac.il/~peleg/papers/iccv09-shiftmap.pdf

CHAPTER 4 ALGORITHMS IMPLEMENTED

Some algorithm(s) have been implemented to carry out photo editing and they are mentioned below:

4.1 Image Convolution

Convolution filtering is used to modify the spatial frequency characteristics of an image. It is a matrix applied to an image and a mathematical operation comprised of integers. It works by determining the value of a central pixel by adding the weighted values of all its neighbors together. The output is a new modified filtered image.

Through Image Convolution various effects like Smooth, Sharpen, Intensify and Enhancement of image etc is achieved.

4.1.1 Example

For the following image shown below:

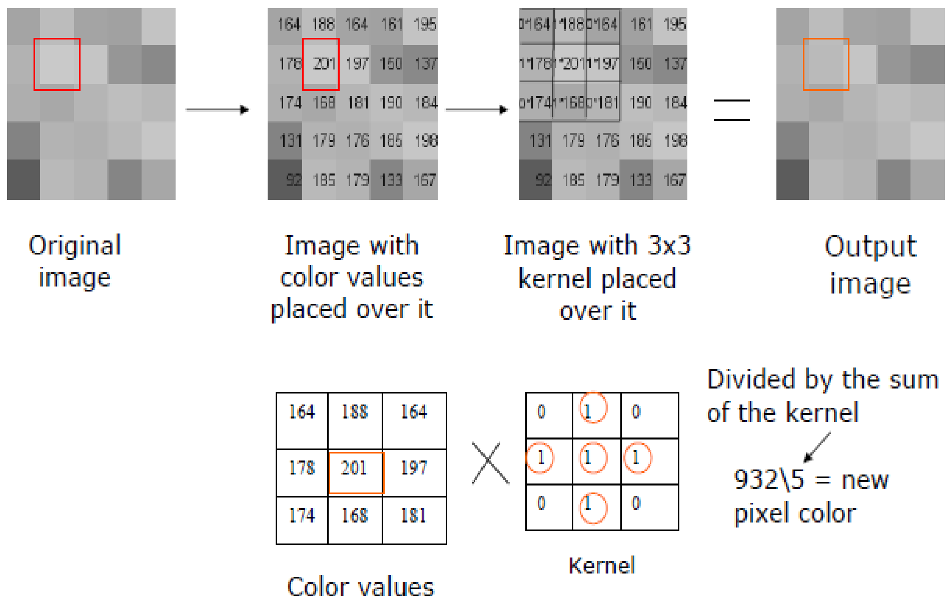

Original image
Image with color values placed over it
Image with 3x3 kernel placed over it
Output image
Divided by the sum of the kernel
932\5 = new pixel color
Color values
Kernel

Fig.1.7

Source:http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Lud wig_ImageConvolution.pdf

Convolution Formula:

$$V = \left| \frac{\sum_{i=1}^{q} \left(\sum_{j=1}^{q} f_{ij} d_{ij} \right)}{F} \right|$$

Where:

f_{ij} =the coefficient of convolution kernel at position i,j (in the kernel)

D_{ij} = the data value of the pixel that corresponds to f_{ij}.

Q = the dimension of the kernel, assuming a square kernel (if q=3, the kernel is 3X3).

F = either the sum of coefficients of the kernel, or 1 if the sum of coefficients is 0.

V = the output pixel value

In cases where V is less than 0, V is clipped to 0.

4.1.2 Pseudo-code

```
for each image row in output image:
    for each pixel in image row:

          set accumulator to zero

          for each kernel row in kernel:
              for each element in kernel row:

                  if element position  corresponding* to pixel position then
                     multiply element value  corresponding* to pixel value
                     add result to accumulator
                  endif

          set output image pixel to accumulator
```

*corresponding input image pixels are found relative to the kernel's origin.

CHAPTER 5 UML DIAGRAMS

5.1 Class Diagram

A **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

In the diagram, classes are represented with boxes which contain three parts:

- The top part contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle part contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom part contains the methods the class can execute. They are also left-aligned and the first letter is lowercase.

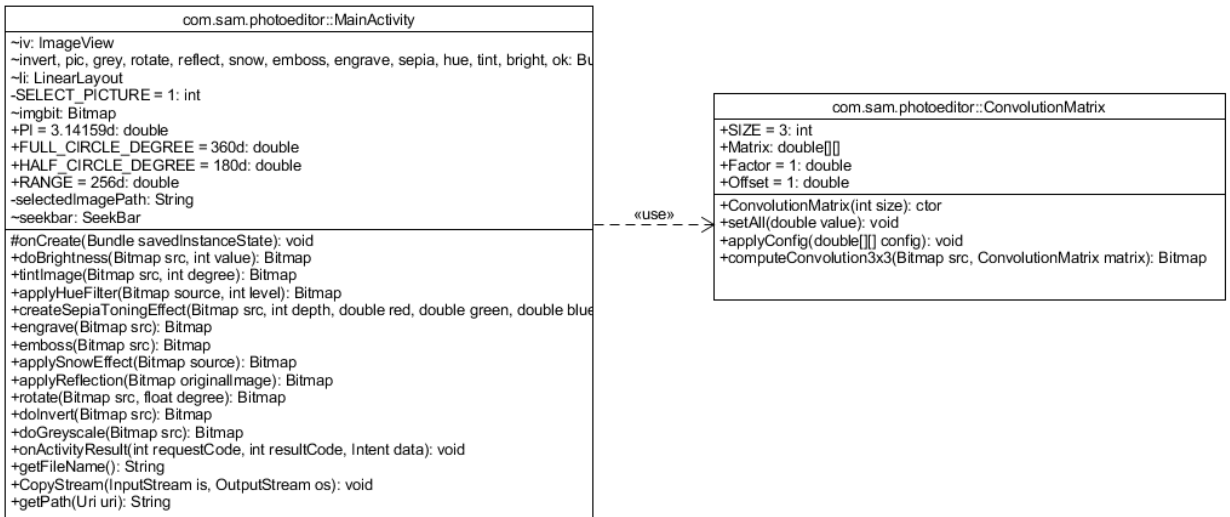

com.sam.photoeditor::MainActivity
~iv: ImageView
~invert, pic, grey, rotate, reflect, snow, emboss, engrave, sepia, hue, tint, bright, ok: Bu
~li: LinearLayout
-SELECT_PICTURE = 1: int
~imgbit: Bitmap
+PI = 3.14159d: double
+FULL_CIRCLE_DEGREE = 360d: double
+HALF_CIRCLE_DEGREE = 180d: double
+RANGE = 256d: double
-selectedImagePath: String
~seekbar: SeekBar
#onCreate(Bundle savedInstanceState): void
+doBrightness(Bitmap src, int value): Bitmap
+tintImage(Bitmap src, int degree): Bitmap
+applyHueFilter(Bitmap source, int level): Bitmap
+createSepiaToningEffect(Bitmap src, int depth, double red, double green, double blue
+engrave(Bitmap src): Bitmap
+emboss(Bitmap src): Bitmap
+applySnowEffect(Bitmap source): Bitmap
+applyReflection(Bitmap originalImage): Bitmap
+rotate(Bitmap src, float degree): Bitmap
+doInvert(Bitmap src): Bitmap
+doGreyscale(Bitmap src): Bitmap
+onActivityResult(int requestCode, int resultCode, Intent data): void
+getFileName(): String
+CopyStream(InputStream is, OutputStream os): void
+getPath(Uri uri): String
«use»
com.sam.photoeditor::ConvolutionMatrix
+SIZE = 3: int
+Matrix: double[][]
+Factor = 1: double
+Offset = 1: double
+ConvolutionMatrix(int size): ctor
+setAll(double value): void
+applyConfig(double[][] config): void
+computeConvolution3x3(Bitmap src, ConvolutionMatrix matrix): Bitmap

Fig.1.8

5.2 Use-Case Diagram

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the case and will often be accompanied by other types of diagrams as well.

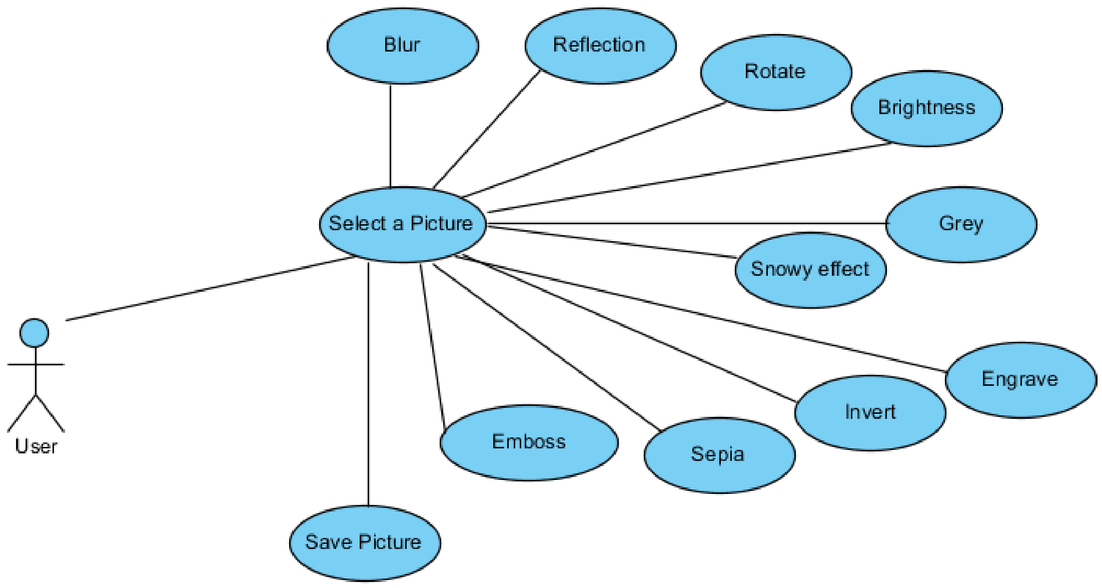
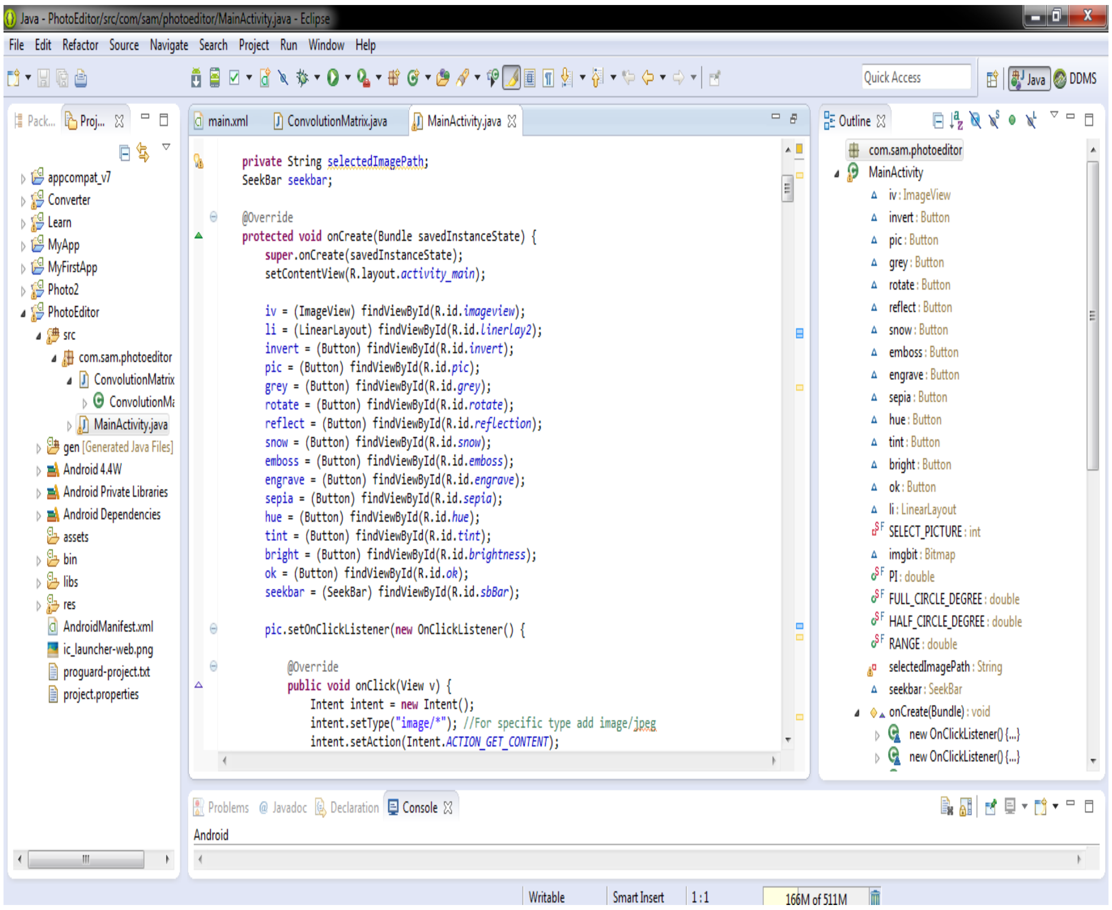


Fig.1.9

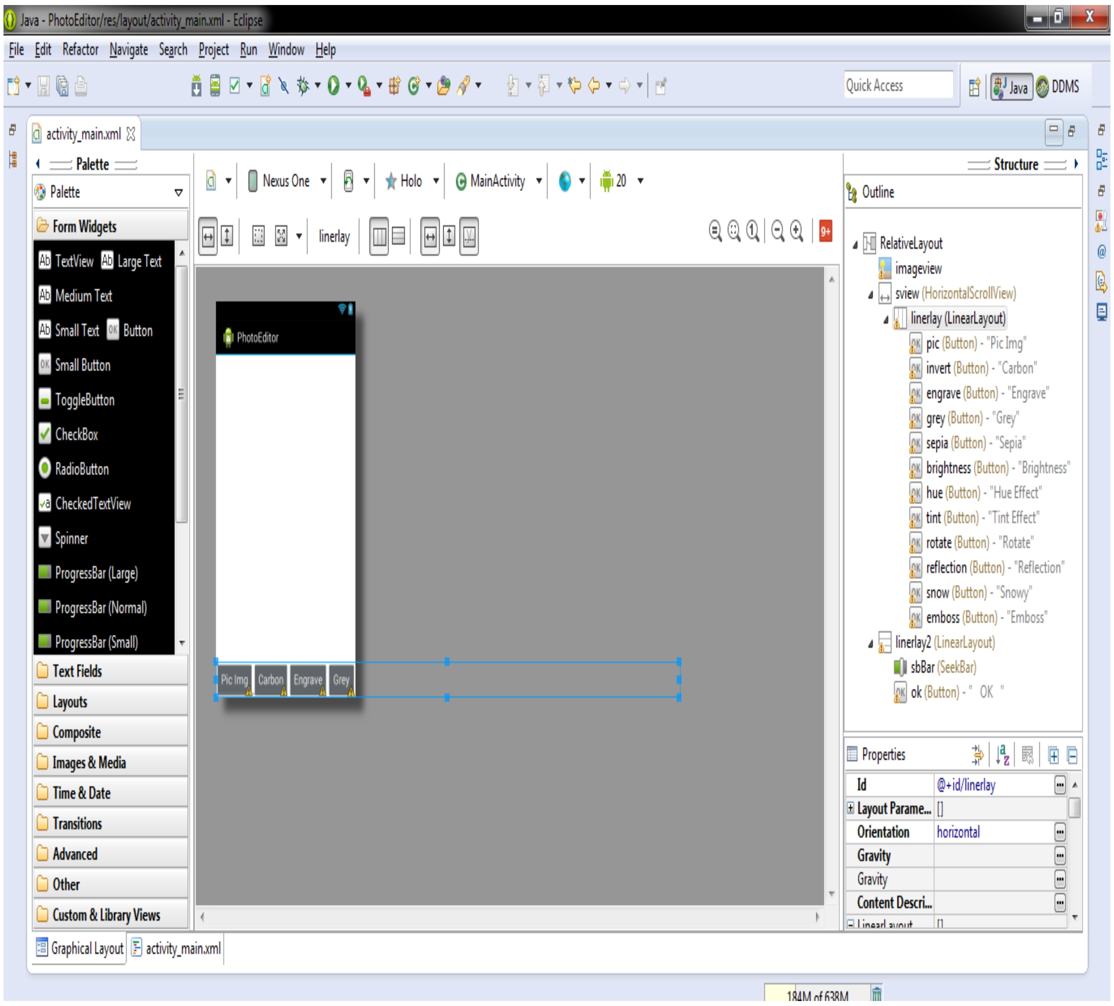
CHAPTER 6 IMPLEMENTATION

6.1 Coding & Design



```java
    private String selectedImagePath;
    SeekBar seekbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        iv = (ImageView) findViewById(R.id.imageview);
        li = (LinearLayout) findViewById(R.id.linerlay2);
        invert = (Button) findViewById(R.id.invert);
        pic = (Button) findViewById(R.id.pic);
        grey = (Button) findViewById(R.id.grey);
        rotate = (Button) findViewById(R.id.rotate);
        reflect = (Button) findViewById(R.id.reflection);
        snow = (Button) findViewById(R.id.snow);
        emboss = (Button) findViewById(R.id.emboss);
        engrave = (Button) findViewById(R.id.engrave);
        sepia = (Button) findViewById(R.id.sepia);
        hue = (Button) findViewById(R.id.hue);
        tint = (Button) findViewById(R.id.tint);
        bright = (Button) findViewById(R.id.brightness);
        ok = (Button) findViewById(R.id.ok);
        seekbar = (SeekBar) findViewById(R.id.sbBar);

        pic.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent intent = new Intent();
                intent.setType("image/*"); //For specific type add image/jpeg
                intent.setAction(Intent.ACTION_GET_CONTENT);
```

Fig.1.10 (a) MainActivity.java

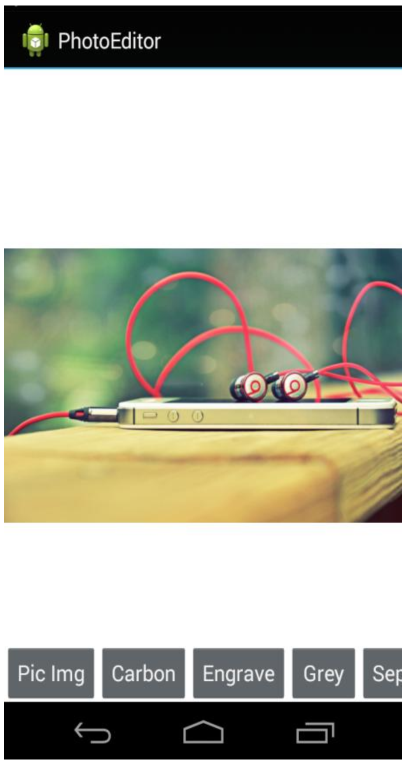


```
RelativeLayout
  imageview
  sview (HorizontalScrollView)
    linerlay (LinearLayout)
      pic (Button) - "Pic Img"
      invert (Button) - "Carbon"
      engrave (Button) - "Engrave"
      grey (Button) - "Grey"
      sepia (Button) - "Sepia"
      brightness (Button) - "Brightness"
      hue (Button) - "Hue Effect"
      tint (Button) - "Tint Effect"
      rotate (Button) - "Rotate"
      reflection (Button) - "Reflection"
      snow (Button) - "Snowy"
      emboss (Button) - "Emboss"
  linerlay2 (LinearLayout)
    sbBar (SeekBar)
    ok (Button) - " OK "
```

Id	@+id/linerlay
Layout Parame...	[]
Orientation	horizontal
Gravity	
Gravity	
Content Descri...	

Fig. 1.10 (b) Activity_main.xml

6.2 Screenshots



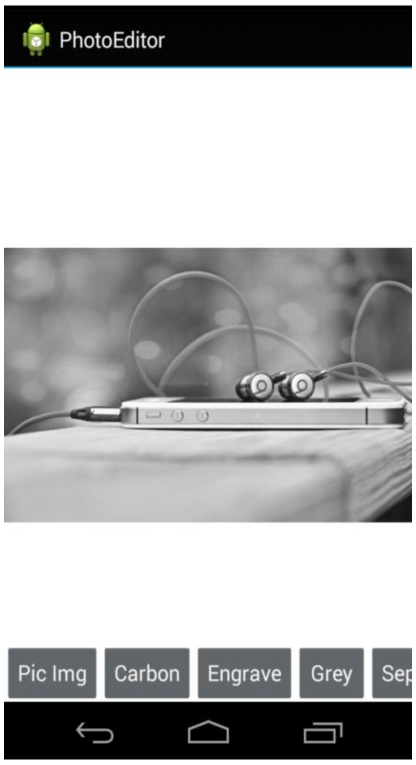
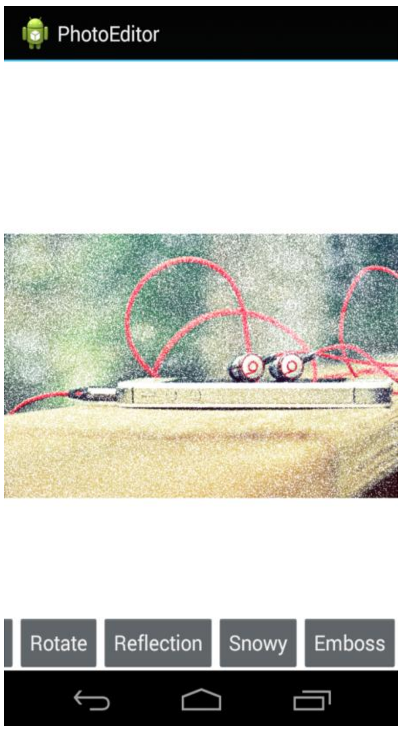


Fig. 1.11 (a) Pick Image Fig. 1.11 (b) Grey



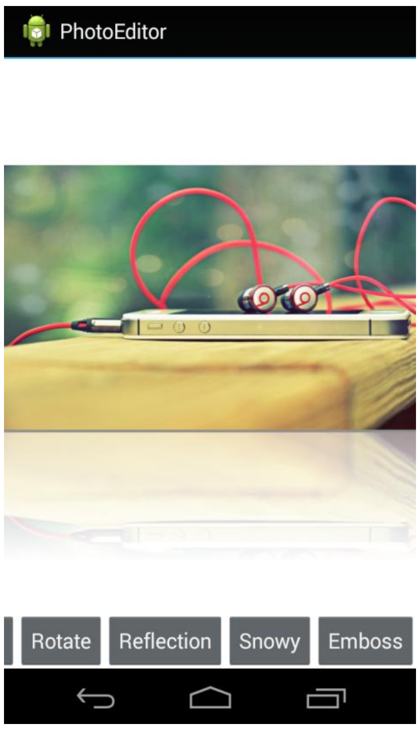
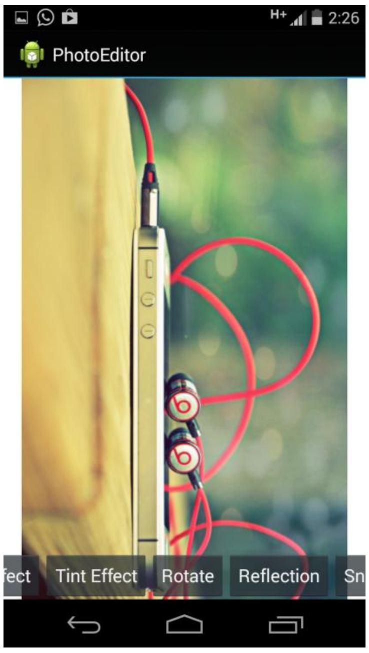


Fig. 1.11 (c) Snowy Fig. 1.11 (d) Reflection



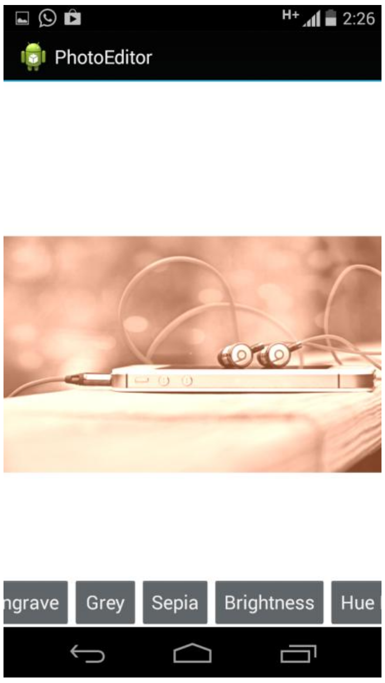


Fig. 1.11 (e) Rotate Fig. 1.11 (f) Sepia

CHAPTER 7 FINDINGS & CONCLUSION

7.1 Findings

From the literature survey and the books I have read, I have understood that making an android application may look simple but it involves a lot of testing and debugging. Also it needs to be tested on various android devices running different android versions so as to check if the application is compatible with those versions and aren't facing any difficulties or lag. Also proper updating of the application will be must in order for the application to survive in this huge android app market

Android is a huge market and everyday tons of apps are added into the play store but only some of them survive. The key to a successful app is its impressive GUI and smooth running without any crashes. Also it shouldn't take too much of space.

7.2 Limitations of Solution

Currently this application of mine does have some limitation. Which have been mentioned below:

i. Integration with social Networking websites like Facebook, twitter, instagram etc is not available.
ii. Limited editing options.
iii. Adding layers and editing different layers separately is not yet available.

7.3 Future Work

The Future Work for the Application will include the following:

1. Adding more photo editing packages.
2. Integration with social networking websites like Facebook, Twitter etc
3. Layers functionality like in that of Adobe Photoshop.

7.4 Conclusion

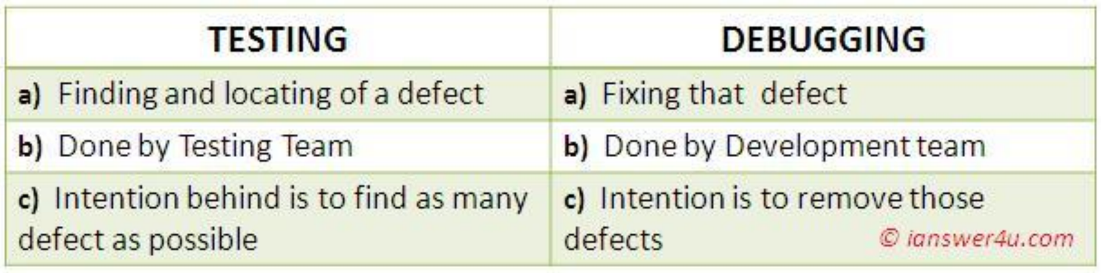
This app gives user the power to edit their picture easily and efficiently. Its an application which can be used by people of all ages who knows how to use a smartphone. The application uses minimum CPU memory and doesn’t compromise with its performance. The editing is fast and smooth and its GUI is easy to use. There are many features which can be added to the app and those features will be added time to time with its regular updates.

CHAPTER 8 TESTING & DEBUGGING

8.1 Introduction

Testing activity is carried down by a team of testers, in order to find the defect in the software. Test engineers run their tests on the piece of software and if they encounter any defect (i.e. actual results don't match expected results), they report it to the development team. Along with the nature of defect, testers also have to report at what point the defect occurred and what happened due the occurrence of that defect. All this information will be used by development team to DEBUG the defect.

Debugging is the activity which is carried out by the development team (or developer), after getting the test report from the testing team about defect(s) (you may note defects can also be reports by the client). The developer then tries to find the cause of the defect, in this quest he may need to go through lines of code and find which part of code in causing that defect. After finding out the bug, he tries to modify that portion of code and then he rechecks if the defect has been finally removed. After fixing the bug, developers send the software back to testers.



TESTING	DEBUGGING
a) Finding and locating of a defect	**a)** Fixing that defect
b) Done by Testing Team	**b)** Done by Development team
c) Intention behind is to find as many defect as possible	**c)** Intention is to remove those defects © ianswer4u.com

Fig 1.12

Source: **http://www.ianswer4u.com/2012/06/testing-and-debugging.html#axzz3ZTw4LMuu**

8.2 Mobile Device Testing

Mobile application testing is a process by which application software developed for hand held mobile devices is tested for its functionality, usability and consistency. Mobile application testing can be automated or manual type of testing. Mobile applications either come pre-installed or can be installed from mobile software distribution platforms. Mobile devices have witnessed a phenomenal growth in the past few years.

8.2.1 Key Challenges in Mobile Application Testing

1. **Variety of Mobile Devices**- Mobile devices differ in screen sizes, input methods (QWERTY, touch, normal) with different hardware capabilities.

2. **Diversity in Mobile Platforms/OS**- There are different Mobile Operating Systems in the market. The major ones are Android, IOS, Symbian, Windows Phone, and BlackBerry (RIM). Each operating system has its own limitations. Testing a single application across multiple devices running on the same platform and every platform poses a unique challenge for testers.

3. **Mobile network operators**- There are over 400 mobile network operators in the world; out of which some are CDMA, some GSM, whereas others use less common network standards like FOMA, and TD-SCDMA. Each network operator uses a different kind network infrastructure and this limits the flow of information.

4. **Scripting**- The variety of devices makes executing the test script (Scripting) a key challenge. As devices differ in keystrokes, input methods, menu structure and display properties single script does not function on every device.

8.2.2 Types of Mobile Application Testing

1. Functional Testing- Functional testing ensures that the application is working as per the requirements. Most of the test conducted for this is driven by the user interface and call flows.

2. Laboratory Testing- Laboratory testing, usually carried out by network carriers, is done by simulating the complete wireless network. This test is performed to find out any glitches when a mobile application uses voice and/or data connection to perform some functions.

3. Performance Testing- This testing process is undertaken to check the performance and behavior of the application under certain conditions such as low battery, bad network coverage, low available memory, simultaneous access to application's server by several users and other conditions. Performance of an application can be affected from two sides: application's server side and client's side. Performance testing is carried out to check both.

4. Memory Leakage Testing- Memory leakage happens when a computer program or application is unable to manage the memory it is allocated resulting in poor performance of the application and the overall slowdown of the system. As mobile devices have significant constraints of available memory, memory leakage testing is crucial for the proper functioning of an application

5. Interrupt Testing- An application while functioning may face several interruptions like incoming calls or network coverage outage and recovery. The different types of interruptions are:

i. Incoming and Outgoing SMS and MMS
ii. Incoming and Outgoing calls
iii. Incoming Notifications
iv. Battery Removal
v. Cable Insertion and Removal for data transfer
vi. Network outage and recovery
vii. Media Player on/off
viii. Device Power cycle

An application should be able to handle these interruptions by going into a suspended state and resuming afterwards.

6. Usability testing- Usability testing is carried out to verify if the application is achieving its goals and getting a favorable response from users. This is important as the usability of an application is its key to commercial success (it is nothing but user friendliness).

7. Certification Testing- To get a certificate of compliance, each mobile device needs to be tested against the guidelines set by different mobile platforms.

The Certified Mobile Application Tester popularly known as CMAT certification exam is offered by the Global Association for Quality Management (GAQM) via Pearson Vue Testing Center worldwide to benefit the Mobile Application Testing Community.

8. Installation testing- Certain mobile applications come pre-installed on the device whereas others have to be installed from the store. Installation testing verifies that the installation process goes smoothly without the user having to face any difficulty. This testing process covers installation, updating and uninstalling of an application.

8.2.3 Some Mobile Application Testing Tools

Some tools that are being used to test code quality in general for mobile applications are as follows:

- **Cross-Platform (Android and iOS)**

1. Appium - Mobile device automation for functional testing

Link : http://appium.io

2. Calabash - Mobile device automation for functional testing

Link : http://calaba.sh

3. Testdroid - Mobile App and Game test automation on real Android and iOS devices
Link : http://www.testdroid.com/

4. Perfecto Mobile - Mobile device automation for functional testing

Link : http://www.perfectomobile.com

5. SOASTA TouchTest - Mobile test automation for functional testing of native & hybrid apps

Link : http://www.soasta.com/products/touchtest/

6. Testin - This tool let you test your apps across 300+ devices. This cloud based solution comes with automated testing features such as automated compatibility, functionality, UI & performance testing.

Link : http://www.itestin.com/

7. Ubertesters - This is a freemium tool which helps you conduct more structured and well organized Mobile QA process. Some of the features of Ubertesters are In-app bug editing, marking, reporting and user feedback, Multi-platform support, Over-the-air (OTA) app distribution, Build management etc. Ubertesters also offers in-the-wild app testing services with its global community of professional testers.

Link : http://ubertesters.com/

8. Crashlytics - This is a free tool available for both- iOS and Android devices.

Link : http://try.crashlytics.com/

9. Ranorex - This is a cross device app testing tool through which you can record one test and run it on multiple devices and languages. You can test your iOS, Android and Windows 8 Apps with this tool.

Link : http://www.ranorex.com/mobile-automation-testing.html

10. Experitest - Mobile device automation for functional testing

Link : http://www.experitest.com

11. Remote TestKit - A device cloud for mobile application testing

Link : https://appkitbox.com/en/testkit/

12. Test Fairy - Mobile application testing with video recording

Link : https://www.testfairy.com/

13. EggPlant - Image based solution

Link : http://www.testplant.com/eggplant/testing-tools/

- **For Android**

1. Android Lint - This is integrated with Eclipse IDE for Android. This will point out potential bugs, performance problems

Link : http://developer.android.com/tools/help/lint.html

2. Find Bugs - This is an open source library for static analysis in Java code

Link : https://code.google.com/p/findbugs-for-android/

3. Maveryx - Maveryx for Android is an automated testing tool for functional, regression, GUI, and data-driven testing of Android mobile application

Link : http://www.maveryx.com

8.3 Android automated testing

8.3.1 Android test strategy

Automated testing of Android applications is especially important because of the huge variety of available devices. As it is not possible to test Android application on all possible device configurations, it is common practice to run Android test on typical device configurations.

Having reasonable test coverage for your Android application helps you to enhance and maintain the Android application.

8.3.2 How to test Android applications

Android testing is based on JUnit. Testing for Android can be classified into tests which require only the JVM and tests which require the Android system.

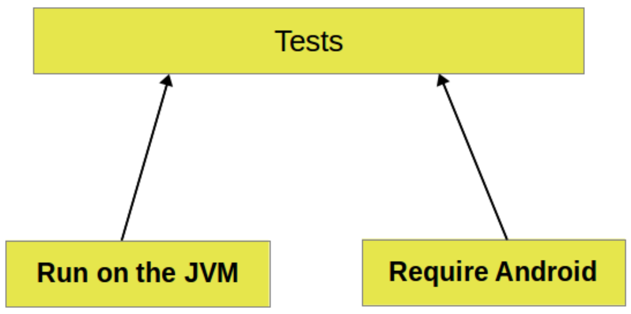


Fig. 1.13

Source: http://www.vogella.com/tutorials/AndroidTesting/article.html

If possible, you should prefer to run your unit tests directly on the JVM as the test execution is much faster compared to the time required to deploy and run the test on an Android device.

8.3.3 What to test on Android applications

The following tables list the important areas you should test in your Android applications.

(Table 1) Areas to test

Test Area	**Description**
Activity life cycle events	You should test if you activity handles the Android life cycle events correctly. You should also test if the configuration change events are handled well and if instance state of your user interface components is restored.
File system and database operations	Write and read access from and to the file system should be tested including the handling of databases.
Different device configurations	You should also test if your application behaves well on different device configurations.

8.4 Testing & Evaluation

8.4.1 Testing Plan

(Table 2)

Type Of Test	**Will Test be performed**	**Comments/ Explanation**
Requirement Testing	[✓ Yes] No	The test has been performed on various mobile devices like Moto G2, Moto X2, Nexus 5, Nexus 4 etc
Unit Testing	[✓ Yes] No	Unit testing will test individual units/code modules of our Project like testing of the individual modules.
Integration Testing	[✓ Yes] No	Integration Testing will be useful for testing the project as a whole. The integrated testing involves the combination of various unit modules testing.
Performance Testing	[✓ Yes] No	Performance testing will be done for checking the overall performance of the application. It will be analyzed through the performance measures, response time, how successfully the database handles the information of various users that use the application, and how well it is presented to the user with the help

		of an extensive GUI.
Stress Testing	[✓ Yes] No	Stress testing will be performed to make predictions about expected load levels. A high stress environment will be created in the application to test till what size the application can pick an image.
Security Testing	[✓ Yes] No	User will be able to do authorized functions only.
Volume Testing	[✓ Yes] No	It is testing the application with large volume of data in the database to find out memory leaks.

8.4.2 List of Test Cases

(Table 3)

Test Case ID	**Input**	**Expected Output**	**Status**
1.	Picking Image Size upto 1 MB size.	No Error	Pass
2.	Picking Image Size upto 5 MB size.	No Error	Pass
3.	Picking JPG,PNG,JPEG Image format	No Error	Pass
4.	Picking GIF image format	No Error	Pass
5.	Editing JPG,PNG,JPEG Image format	No Error	Pass
6.	Editing GIF image format	Error	Fail
7.	Editing Image Size Upto 5 MB size	No Error	Pass
8.	View the Image before Editing	No Error	Pass
9.	View the Image after Editing	No Error	Pass

(Table 4)

Module	**Input**	**Expected Output**	**Status**
Convolution Matrix algorithm	Connected	Connect	Pass
SeekBar Movement	Connected	Connect	Pass
Colour Manipulations	Connected	Connect	Pass
Apply Editing Filters	Connected	Connect	Pass
Pick Image	Connected	Connect	Pass

CHAPTER 9 REFERENCES

[1] Ryan Cohen and Tao Wang, *GUI Design for Android App*, 2014.

[2] Hays, Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, and Chao Qian, *Photo editing algorithm changes weather, seasons automatically*, Brown University, 2014.

[3] Jamie Ludwig, *Image Convolution*, Portland State University, 2013.

[4] Reto Meier, *Professional Android 4 Application Development*, 2012.

[5] Michael Burton and Donn Felker, *Android Application Development For Dummies, 2nd Edition*, 2012.

[6] Yael Pritch, Eitam Kav-Venaki and Shmuel Peleg, *Shift-Map Image Editing*, The Hebrew University of Jerusalem, 2009.

[7] Lei Zhang, Yanfeng Sun, Mingjing Li, Hongjiang Zhang, *automated red-eye detection and correction in digital photographs*, Microsoft Research Asia, 2007

APPENDIX A
Description of Tools

ECLIPSE

In computer programming, **Eclipse** is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications.

By means of various plug-ins, Eclipse may also be used to develop applications in other programming languages:

Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Lua, Natural, Perl , PHP, Prolog, Python, R, Ruby, Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Released under the terms of the Eclipse Public License, Eclipse SDK is free and open source software (although it is incompatible with the GNU General Public License). It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

Eclipse ADT (Android Development Tools)

Android Development Tools (ADT) is a Google-provided plugin for the Eclipse IDE that is designed to provide an integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let developers set up new Android projects, create an application UI, add packages based on the Android Framework API, debug their applications using the Android SDK tools, and export signed (or unsigned) .apk files in order to distribute their applications. It is free download. It was the official IDE for Android but was replaced by Android Studio (based on IntelliJ IDEA Community Edition).

Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools.[8] These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows XP or later. As of March 2015, the SDK is not available on Android itself, but the software development is possible by using specialized Android applications.

Until around the end of 2014, the officially supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, andNetBeans IDE also supports Android development via a plugin. As of 2015, Android Studio, made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

Android Debug Bridge

The Android Debug Bridge (ADB) is a toolkit included in the Android SDK package. It consists of both client and server-side programs that communicate with one another. The ADB is typically accessed through the command-line interface,[18] although numerous graphical user interfaces exist to control ADB.