piClone

By

Mian Basit Mahmood

Haider Mushtaq

Hussain Ali

Supervisor

Dr Naima Iltaf

Submitted to the Faculty of Computer Science

National University of Science and Technology, Rawalpindi

For the partial requirement of Bachelors of Engineering in Computer Software Engineering

July 2016

Version 2.0.0.0

ABSTRACT

piClone

This Application is aimed to provide a platform for the end user to edit images on an Android device.

It takes in two pictures from the user. One picture can be selected as the base picture or as the background. By identifying and recognizing objects, via manual cropping, the desired object can be extracted from the image and then be stitched on top of Background image of the user's choosing.

The contours would be recognized by the help of some of the following different well proven techniques, such as Edge and Corner Detection, The Canny Edge Detector, The Sobel operator, Harris Corner Detection and others like Hough transformations. To improve the primary or the secondary images, our system will provide the user the ability to apply different filters to the images, including basic filters such as Mean Blur, Gaussian Blur, Median Blur, Sharpener, Dilation, Erosion and different Threshold filters such as Binary Threshold and Threshold to Zero filters. To make the editing and cloning of a selection region seamless we will make use of Poisson partial differential equation with Dirichlet boundary conditions which specifies the Laplacian of an unknown function over the domain of interest, along with the unknown function values over the boundary of the domain. The extent of the changes ranges from slight distortions to complete replacement by novel content.

The purpose of this app is to give the end user easy access to various image editing and image extraction techniques with the help of an easy to use graphic user interface and minimal learning effort. Using this app the end user can create amazing breathtaking pictures which can then be saved to local storage or can then be shared among friends using the social media.

CERTIFICATE FOR CORRECTNESS AND APPROVAL

Certified that work contained in the thesis – piClone carried out by MianBasitMahmood, HaiderMushtaq and Hussain Aliunder supervision of Dr. NaimaIltaf for partial fulfilment of Degree of Bachelor of Software Engineering is correct and approved.

Approved by

Dr. NaimaIltaf

CSE DEPARTMENT MCS

DATED:

_____

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our parents, without whose unflinching support and unstinting cooperation, a work of this magnitude would not have been possible

To our supervisor Dr. NaimaIltaf who has given us great support and valuable suggestions throughout the implementation process.

And finally to our Friends and siblings for their encouragement.

ACKNOWLEDGEMENTS

There is no success without the will of ALLAH Almighty. We are grateful to ALLAH, who has given us guidance, strength and enabled us to accomplish this task. Whatever we have achieved, we owe it to Him, in totality. We are also grateful to our parents and family and well-wishers for their admirable support and their critical reviews. We would like to thank our supervisor. Dr. NaimaIltaf, for her continuous guidance and motivation throughout the course of our project. Without their help we would have not been able to accomplish anything.

Table of Contents

Chapter 1: Introduction

1.1 Overview

The Application takes in two pictures from the user. One picture can be selected as the base picture or as the background. By identifying and recognizing objects, via manual cropping, the desired object can be extracted from the image and then be stitched on top of Background image of the user's choosing.

The contours would be recognized by the help of some of the following different well proven techniques, such as Edge and Corner Detection, The Canny Edge Detector, The Sobel operator, Harris Corner Detection and others like Hough transformations. To improve the primary or the secondary images, our system will provide the user the ability to apply different filters to the images, including basic filters such as Mean Blur, Gaussian Blur, Median Blur, Sharpener, Dilation, Erosion and different Threshold filters such as Binary Threshold and Threshold to Zero filters. To make the editing and cloning of a selection region seamless we will make use of Poisson partial differential equation with Dirichlet boundary conditions which specifies the Laplacian of an unknown function over the domain of interest, along with the unknown function values over the boundary of the domain. The extent of the changes ranges from slight distortions to complete replacement by novel content.

1.2 Problem Statement

This Application is aimed to provide a platform for the end user to edit images on an Android device. The purpose of this app is to give the end user easy access to various image extraction and image merging techniques with the help of an easy to use graphic user interface and minimal learning effort. Using this app the end user can create amazing breathtaking pictures which can then be saved to local storage or can then be shared among friends using the social media.

1.3 Approach

We are working with Image Extraction from a selected Image and Merging of the extracted piece of selected image on the base Image. It will be our algorithm that will be able to merge (clone) these 2 images seamless and effective manner .We aim to develop an algorithm that can merge 2 images in such a way that the output image is as realistic as possible.

1.4 Scope

The project aims at exploring, tinkering and modifying the image editing capabilities and implementing the *Poisson Equations*.

The scope of work is limited to extracting an object from the input image and merging it to a background image meanwhile providing a variety of image enhancing functionalities.

1.5 Objectives

The Objectives of this project are following:

- Develop an effective solution for modifying images using existing system.
- Detect the desired object from selected image and extract it.
- Provide a layer by layer canvas platform for image modification.
- Upload the app on Android Market using Google Play Store.
- Create Facebook page for application support and feedback.
- Make it easy for the user to upload the created images on Social Media.
- Create a large fan following of the application.

1.6 Deliverables

Table 1

Sr.	Tasks	Deliverables
1	Literature Review	Literature Survey
2	Requirements Gathering	SRS Document
3	Application Design	Design Document (SDS)
4	Implementation	Implementation on computer with a live test to show the accuracy and ability of the project
5	Testing	Evaluation plan and test document
6	Deployment	Complete application along with necessary documentation

1.7 Overview of the document

This document shows the working of our application piClone. It starts of with the system architecture which highlights the modules of the software and represents the system in the form of component diagram, Use Case Diagram, Sequence Diagram and general design of the system. Then we move on to discuss the detailed Description of all the components involved. Further we discuss the dependencies of the system and its relationship with other products and the capacity of it to be reused. Then towards the end we shall discuss the Design Tradeoffs and the Pseudocode.

1.8 Purpose of the document:

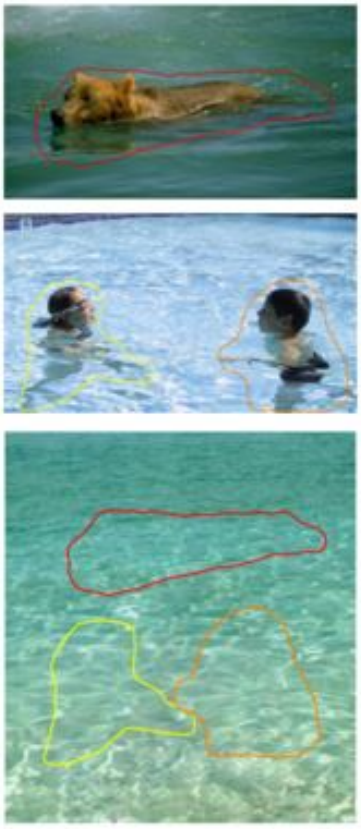
This document aims to elaborate the idea and design of the project that is piClone. This document will highlight all the specifications of our project i.e. how it will be used, what will be the scenario in which the project will be useful. This project is basically a baseline work for further research that is ultimately perfecting Image Cloning.

Chapter 2: Literature Review

There were a few projects that were based on the idea of Image Cloning following is a detailed description of projects previously carried out in this context.

Gradient Domain Cloning Research

The human visual system is quite sensitive to gradients: it tends to ignore small gradients and pick up large gradients. As discussed, researchers have developed a dual representation of images called the *gradient domain* which is just the gradient or *x* and *y* derivatives of the input image (in each color channel). One can modify gradients however one wishes, apply boundary conditions, and then "integrate" using a linear solver to go back to color domain. This allows for many applications such as the artistic depiction in Gradient Shop.



sources/destinations

naive cloning gradient domain cloning

Using generic interpolation machinery based on solving Poisson equations, a variety of novel tools are introduced for seamless editing of image regions. The first set of tools permits the seamless importation of both opaque and transparent source image regions into a destination region. The second set is based on similar mathematical ideas and allows the user to modify the appearance of the image seamlessly, within a selected region. These changes can be arranged to affect the texture, the illumination, and the color of objects lying in the region, or to make tile able a rectangular selection.

Image editing tasks concern either global changes (color/intensity corrections, filters, deformations) or local changes confined to a selection. Here we are interested in achieving local changes, ones that are restricted to a region manually selected, in a seamless and effortless manner. The extent of the changes ranges from slight distortions to complete replacement by novel content. Classic tools to achieve that include image filters confined to a selection, for slight changes, and interactive cut-and-paste with cloning tools for complete replacements. With these classic tools, changes in the selected regions result in visible seams, which can be only partly hidden, subsequently, by feathering along the border of the selected region. We propose here a generic machinery from which different tools for seamless editing and cloning of a selection region can be derived. The mathematical tool at the heart of the approach is the Poisson partial

differential equation with Dirichlet boundary conditions which specifies the Laplacian of an unknown function over the domain of interest, along with the unknown function values over the boundary of the domain. The motivation is twofold.

So, given methods for crafting the Laplacian of an unknown function over some domain, and its boundary conditions, the Poisson equation can be solved numerically to achieve seamless filling of that domain. This can be replicated independently in each of the channels of a color image. Solving the Poisson equation also has an alternative interpretation as a minimization problem: it computes the function whose gradient is the closest, in the L2-norm, to some prescribed vector field — the guidance vector field — under given boundary conditions. In that way, the reconstructed function interpolates the boundary conditions inwards, while following the spatial variations of the guidance field as closely as possible.

A number of possible choices for the guidance vector field. We show in particular that this interpolation machinery leverages classic cloning tools, both in terms of ease of use and capabilities. The resulting cloning allows the user to remove and add objects seamlessly. By mixing suitably the gradient of the source image with that of the destination image, it also becomes possible to add transparent objects convincingly. Furthermore, objects with complex outlines including holes can be added automatically without the need for painstaking cutting out.

Conclusion

Using the generic framework of guided interpolation, we have introduced a variety of tools to edit in a seamless and effortless manner the contents of an image selection. The extent of possible changes ranges from replacement by, or mixing with, another source image region, to alterations of some aspects of the original image inside the selection, such as texture, illumination, or color. An important common characteristic of all these tools is that there is no need for precise object delineation, in contrast with the classic tools that address similar tasks. This is a valuable feature, whether one is interested in small touch-up operations or in complex photomontages. It is clear that the cloning facilities described can be combined with the editing ones. It is for instance possible to insert an object while flattening its texture to make it match the style of a texture-free destination. Finally, it is worth noting that the range of editing facilities derived in this paper from the same generic framework could probably be extended further. Appearance changes could for instance also deal with the sharpness of objects of interest, thus allowing the user to make apparent changes of focus.

Chapter 3: Software Requirement Specification

3.1Introduction

The purpose of this part is to describe the project titled “piClone”. This part contains the functional and non-functional requirements of the project. It contains the guidelines for developers and examiners of the project.

The purpose of this application is to make it simple and easy for the end user to modify images in cool and exciting ways. The system would give the capability to choose any object from the image and would be able to extract it. This extracted object can then be merged on top of a base image that would be used as its background. The scope of this project is limited to the desired object extraction from selected image and merging it on top of desired background image.

3.2 Overall Description

3.2.1 Product Perspective

This product is aimed to provide a platform for the end user to edit images on an Android device.

It takes in two pictures from the user. One picture can be selected as the base picture or as the background. From the other picture, the desired object will be cloned on to the base picture. Gradients can be modified, boundary conditions can be applied, and then everything would be "integrated" using a linear solver to go back to color domain of the base image.

The plan to carry out this project consists of three main tasks:

1. Develop an algorithm for user to extract the desired object from the input image.
2. Develop an algorithm to merge the extracted object to a desired background image.
3. Interfacing Android App.

3.2.2 Product Functions

Following functions will be performed by piClone:

1. Capture image from Camera
2. Select image from gallery
3. Extract / modify image
4. Merge two or more images

3.2.3 User Classes and Characteristics

Following are the user classes and their brief description.

Researchers

Researchers will use this Project as a guide to understand the “Image Cloning”. They will use this as a base for upgrading and adding new features. They can also use this for developing a new project by using this a reference material.

Tester

Tester will also use this project to check for bug finding. They will also use this to check it is in accordance to the srs document.

Project Evaluator/Supervisor

Project supervisor/Evaluator will also use the product to evaluate. They will use this product to find the accuracy and error in the output.

3.2.4 Operating Environment

The operating environment required for this project is:

Android Studio

Software requirements:

OS: Android Operating System

3.2.5 Design and Implementation Constraints

Following are the constraints of design and implementation in our project

- Accuracy depends on the way user crops the desired Object from Selected Image.
- Overall performance of this project will depend on the quality of input Images.
- It will also depend on the position where the extracted object will be placed on the base image for merging it.

3.2.6 User Documentation

For the user documentation, a Guide Feature will be included in the system. It will include the details of the system's working. Help documents will also be a part of the system. The project report will also be available for the users which will highlight the system features, working and procedures.

3.2.7 Assumptions and Dependencies

1. Overall performance of this project will depend on the quality of input Images.
2. Accuracy depends on the way user crops the desired Object from Selected Image.
3. It will also depend on the position where the extracted object will be placed on the base image for merging it.

3.3 External Interface Requirements

3.3.1 User Interfaces

The System comprises of an android based application, using java and XML, which shall provide a graphical user interface for user friendly environment. The user would be asked to input images that need to be processed. Then different image transformation and processing would take place, ending in a new image as desired by the user.

The user interface for the android application of the System, shall be compatible to all android devise but for best user experience the following versions are preferable

- Jelly beans 4.1.2
- ICS 4.1.1

User Interface:

Figure 3.3.1 Interfaces

3.3.2 Software Interfaces

To visualize the brainwaves on the monitor Brainwave Visualizer software will be used.

- Android App will be installed on android device with android version ICS or later.
- The android app would be built using Android Studio.
- For this system, following API and external libraries will be used:
  - Google APIs
  - Android APIs

3.4 System Features

SF-1 Get Image from Camera

Description	The app will be able to access the device's camera using Android camera Api.
Priority	Medium
Pre-Conditions	Mobile should have a working camera.
Stimulus/Response	Capture image.
Post-Conditions	Saving images in png format to the gallery.
Risk	Medium

Functional Requirements:

REQ-1 The application must be properly installed in android mobile phone.

REQ-2 The android mobile phone should have a working camera.

REQ-3 The image taken should be saved in png format.

SF-2 Select image from Gallery

Description	Any image can be selected by the user from the phone's storage through its gallery. This can be used both as its primary image or its base image.
Priority	Medium
Pre-Conditions	Images should be present in phone memory.
Stimulus/Response	A new copy of the same image would be made and worked on so that the original image would not be damaged.
Post-Conditions	Images are available for further processing.
Risk	Low

Functional Requirements:

REQ-4 Mobile phone should contain image.

SF-3 Detection of desired object

Description	Image processing is applied to detect desired object. OpenCV library will be used for this purpose.
Priority	High
Pre-Conditions	The given image should be of png format.
Stimulus/Response	object in the given image will be detected.
Post-Conditions	The coordinates of the boundary of the object are obtained.
Risk	High

Functional Requirements:

REQ-5 The input image should have minimum 1 object.

REQ-6 The human figure should be clearly visible so that the coordinates of the object that will be obtained should be accurate.

REQ-7 If the application is not able to detect object from the input image, it will generate an error to the user.

SF-4 Extraction of Object

Description	The bitmap of the detected Object is extracted as a new image.
Priority	High
Pre-Conditions	Image with a detected Object.
Stimulus/Response	New png Image would be made.
Post-Conditions	The extracted human form is obtained as an image in png format.
Risk	HIGH

Functional Requirements:

REQ-8 The application should create a new image of the detected Object.

REQ-9 The new image of Object should be stored in phone's memory/SD card in png format.

SF-5 Layer by Layer image Canvas

Description	Images can be put on top of each other as layers. This is similar to cutting out or cropping a picture and then pasting on another complete picture.
Priority	Medium
Pre-Conditions	Multiple images should be available.
Stimulus/Response	User feedback can help make the image better.
Post-Conditions	Ready the images for merging
Risk	High

Functional Requirements:

REQ-10 The application will put the detected human form on the base image, which is a background image.

SF-6 Merging and Modifying images.

Description	After the images are set up on the canvas and ready, then all of them will be combined to form a single new Image.
Priority	High

Pre-Condition	Images set up on canvas.
Stimulus/Response	A new image would be made.
Post-Conditions	The image would be saved to gallery.
Risk	Medium

Functional Requirements:

REQ-11 The images, that are to be merged and modified, are set up on canvas.

REQ-12 Image processing algorithms will be used to merge those images.

SF-7 Creating new Image.

Description	In order not to damage any of the existing images used in our production, new image is made.
Priority	High
Pre-Condition	Image is set up and merged properly.
Stimulus/Response	New image with the desired output would be made.
Post-Conditions	Image saved to gallery
Risk	Low

Functional Requirements:

REQ-13 App shall create a new image with desired output and save it to gallery.

SF-8 Sharing images on Social media.

Description	The user would be allowed to share the image processed by the program on Social media by using Android API's
Priority	Low
Pre-Condition	Image ready and stored on Device.
Normal Course	The user would select the image and press the share button which would open a prompt asking the user which social media site he wants to share the image on.

Post-Condition	A message would be displayed that the image has been uploaded and published.
Alternate Course	The user can also upload the image from his gallery as the image would be made available there. This would be just like uploading any regular photo to the internet.
Post-Condition	The user would get informed if the image has been uploaded successfully.
Risk	Low

Functional Requirements:

REQ-14 Android phone must be connected with the internet.

SF-9 How to/ Help guide.

Description	In order to help the user get familiar with the app, a small help guide would be added to the app as an activity which the user can view to find out about the different features of the application.
Priority	Low
Pre-Condition	All the features should be well documented. This feature can only be added after at least a working prototype has been made.
Stimulus/Response	
Post-Condition	
Risk	Low

REQ-15 Application features must be well documented in order to create Help guide.

3.5 Other Nonfunctional Requirements

3.5.1 Performance Requirements

piClone performance will be based upon the quality of the image produced by merging the human form and the base image. The real the produced image will look, the higher will be the performance of piClone. The output image shall take no more than 1 minute to be produced.

3.5.2 Safety Requirements

The use of piClone has no harm whatsoever. If the application crashes during any phase, the input images will stay safe in phone's memory/SD card.

3.5.3 Security Requirements

The security is not much of an issue for piClone as the input images and the output image will be saved in phone's memory/SD card. As long as the phone's memory/SD card does not get corrupt, security will be no problem.

3.5.4 Software Quality Attributes

Usability

The application will be easy to operate for any user. The graphical user interface will be designed, organized and presented in a manner that is both visually appealing and easy to use.

Accuracy

To ensure accuracy and correctness of the output image, the image processing algorithms will be written with no tolerance for error.

Availability

The application will available to the user until the phone is in working state and the application is installed and configured properly.

Flexibility

The design and architecture of the application will be flexible enough for catering any new requirements, if any at some later stage or for the application enhancement.

Data Integrity

If the application crashes during any phase, the input images will stay safe in phone's memory/SD card.

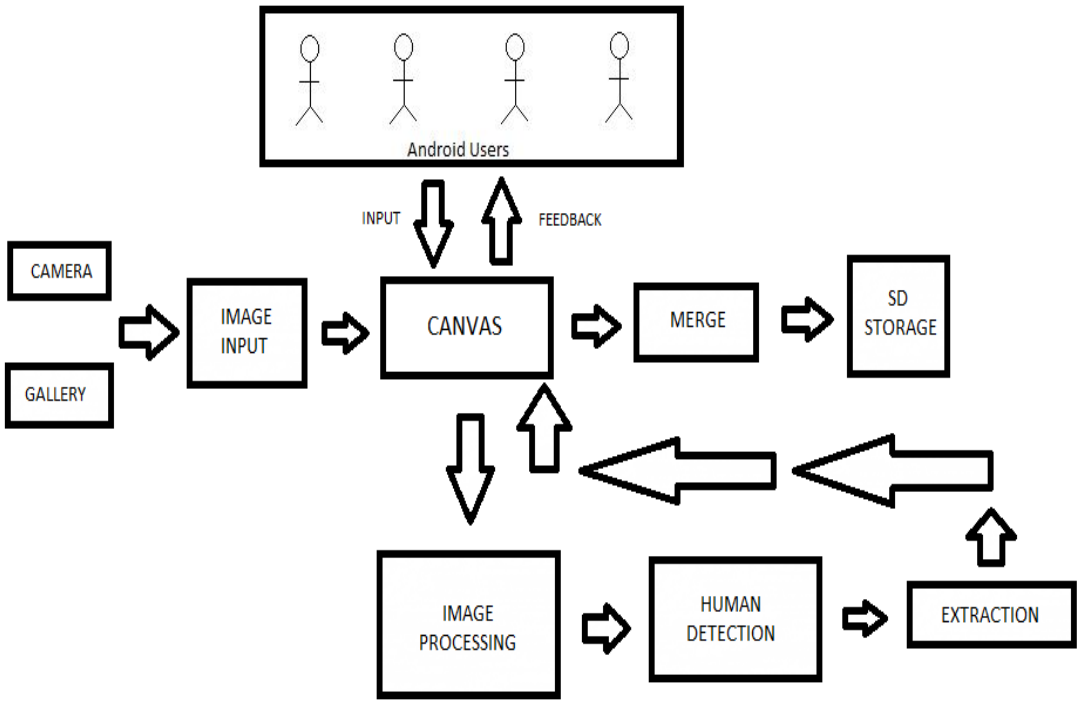


Figure 3.5

Chapter 4: Design and Development

4.1 Introduction

How can computers understand the visual world of humans? In this document and in the creation of this application, we will treat vision as a process of inference from noisy and uncertain data and emphasizes probabilistic, statistical, data-driven approaches to observe and identify contours in images.

By identifying and recognizing objects, via manual cropping, the desired object can be extracted from the image and then be stitched on top of another image of the user's choosing.

The contours would be recognized by the help of some of the following different well proven techniques, such as Edge and Corner Detection, The Canny Edge Detector, The Sobel operator, Harris Corner Detection and others like Hough transformations. To improve the primary or the secondary images, our system will provide the user the ability to apply different filters to the images, including basic filters such as Mean Blur, Gaussian Blur, Median Blur, Sharpener, Dilation, Erosion and different Threshold filters such as Binary Threshold and Threshold to Zero filters. To make the editing and cloning of a selection region seamless we will make use of Poisson partial differential equation with Dirichlet boundary conditions which specifies the Laplacian of an unknown function over the domain of interest, along with the unknown function values over the boundary of the domain.

Theextentofthechangesrangesfromslightdistortions to complete replacement by novel content.

By mixing suitably the gradient of the source image with that of the destination image, it also becomes possible to add transparent objects convincingly. Furthermore, objects with complex outlines including holes can be added automatically without the need for painstaking cutting out.

The purpose of this app is to give the end user easy access to various image editing and image extraction techniques with the help of an easy to use graphic user interface and minimal learning effort.

4.2 Scope of the Development Project

The project aims at exploring, tinkering and modifying the image editing and our implementation of the *Poisson Equations*.

The scope of work is limited to extracting an object from the input image and merging it to a background image meanwhile providing a variety of image enhancing functionalities.

4.3SystemArchitectureDescription

This Section overview of application, its higher and lower levels details and user interfaces.

4.3.1OVERVIEW OF MODULES/COMPONENTS

piClone comprises of following components:

1. Graphical User Interface
    a. Activity
    b. Fragment
2. Image Processing
    a. Cropping Image
    b. Edge Detection
        i. Difference of Gaussian
        ii. The Canny Edge Detection
        iii. The Sobel
    c. Image Filters
        i. Mean Blur
        ii. Median
        iii. Gaussian

iv. Threshold

v. Erode

vi. Sharpen

d. Image Merging

e. Image Blending

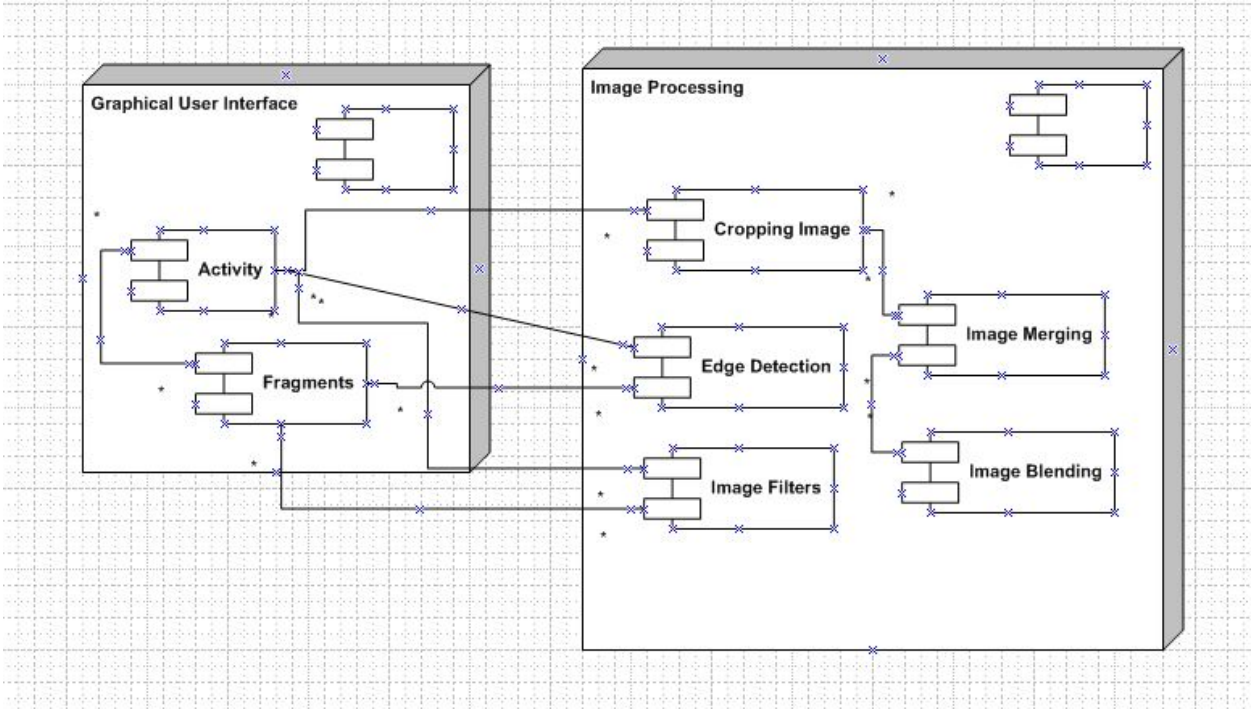


Fig. 4.3.1: *Component diagram for piClone*

Graphical User Interface is part of presentation layer in the system architecture.

Graphical User Interface directly interacts with the user. He/she provides an input for the required action (through this component) and it displays its output respectively.

Image Processing has 5 sub components.

Image Cropping is to get the input from the user that would help choose which portion or object of the image, the user is interested in so that it can be extracted.

Edge Detection, The function of this sub-component is process the Image using various Edge Detection Algorithms and to display the output in an image view.

Image Filter, The function of this sub-component is processing the Image using various Image Filters and to display the output in an image view.

Image Merging, The function of this sub-component is to merge both the primary image containing the desired object and the background Image together to form a new Image.

Image Blending, The function of this sub-component is to blend both the primary image containing the desired object and the background Image together in the new Image.

4.3.2 Structure and Relationships

This section covers the overall technical description of piClone. It shows the working of application in perspective of different point-of-views and also shows relationships between different components.

Use Case Diagram

Fig. 4.3.2.1: *Use Case*

Actors

Primary Actor(s): Application User, *Secondary Actor(s):* Gallery, Camera

Use Cases

1. Select Image
2. Capture Image
3. From Gallery
4. Apply Filter
5. Detect Edges
6. Select Object
7. Extract Object
8. Add Background
9. Merge Object with Background
10. Blend Object with Background
11. Gaussian Blur
12. Mean Blur
13. Median Blur
14. Bla Bla Blur
15. Threshold
16. Dilute
17. Sharpen
18. The Canny
19. Difference of Gaussian
20. The Sobel

Use Case Description

Use Case 1

Use Case Name	**Select Image**
Primary Actor	Application User
Secondary Actor	Gallery, Camera
Normal Course	• Application User selects an Image from Gallery • Application User takes an Image using Mobile Camera • Application User clicks the next button to enter the application
Alternate Course	If the Application User does not select an image to be edited, the application will generate an error and will ask user to choose an image.
Pre-Condition	The application should be able to access android mobile phone gallery and camera.
Post Condition	The Application User can now choose what to do with the selected image.
Extends	*Apply Filter, Detect Edges, Select Object*
Include	*Capture Image, From Gallery*
Assumptions	The Application User Selects an image from mobile Gallery or takes an image from mobile camera and move forward in application.

Use Case 2

Use Case Name	**Apply Filter**
Primary Actor	Application User
Secondary Actor	N/A

Normal Course	Application User chooses any one of the 7 given Filters to apply on the Selected image.
Alternate Course	If the Application User does not want to choose any Filter, which is to be applied on the selected image, the application user will have choice of moving back to the pervious interface.
Pre-Condition	All Filters must be available to be applied on any selected image as chosen by the application user. Filters should be working properly.
Post Condition	An image with the desired Filter applied on it will be displayed for the Application User and will be saved in Galley.
Extends	*N/A*
Include	*Gaussian Blur, Median Blur, Mean Blur, Erode, Threshold, Dilute, Sharpen*
Assumptions	The Application User Selects the Filter which he/she wants to apply on the selected image. The Output image which is the Filtered image is displayed on the screen and saved in Gallery.

Use Case 3

Use Case Name	**Detect Edges**
Primary Actor	Application User
Secondary Actor	N/A
Normal Course	Application User chooses any one of the 3 given Edge Detection Methods to apply on the Selected image.

Alternate Course	If the Application User does not want to choose any Edge Detection Method, which is to be applied on the selected image, the application user will have choice of moving back to the pervious interface.
Pre-Condition	All Edge Detection Method must be available to be applied on any selected image as chosen by the application user. Edge Detection Method should be working properly.
Post Condition	An image with the desired Edge Detection Method applied on it will be displayed for the Application User and will be saved in Galley.
Extends	*N/A*
Include	*The Canny, Difference of Gaussian, The Sobel*
Assumptions	The Application User Selects the Edge Detection Method which he/she wants to apply on the selected image. The Output image will be displayed on the screen and saved in Gallery.

Use Case 4

Use Case Name	**Select Object**
Primary Actor	Application User
Secondary Actor	N/A
Normal Course	Application User chooses any object he/she wants to choose from the Selected image, by making a bounding box around that object. Application user can choose as many objects as he/she wants one at a time.
Alternate Course	If the Application User does not want to choose any object from the selected image, he/she will have choice of moving back to the pervious interface.

Pre-Condition	All
Post Condition	A bounding box around the object to be selected is made and the object is ready to be extracted.
Extends	*N/A*
Include	*Extract Object*
Assumptions	The Application User Selects the Object which he/she wants to Extract from the selected image, by creating a bounding box around that object.

Use Case 5

Use Case Name	**Extract Object**
Primary Actor	Application User
Secondary Actor	N/A
Normal Course	Application User extracts the selected object (The Object around which a bounding box is made) from the Input Image, by pressing the button "Extract Object".
Alternate Course	If the Application User does not want to Extract the selected Object, he/she will have choice of moving back to the pervious interface.
Pre-Condition	The Object that needs to be Extracted must first be Selected by making a bounding box around it.
Post Condition	The Selected Object is Extracted from the input image and saved in android mobile phone's Gallery.
Extends	*Add Background*
Include	*N/A*
Assumptions	The Application User Extracts the Object which he/she has Selected from the Input image.

Use Case 6

Use Case Name	**Add Background**
Primary Actor	Application User
Secondary Actor	N/A
Normal Course	Application User selects an image from Gallery or Take an image using mobile camera. This image is used as Background for the Extracted objects from other Images.
Alternate Course	If the Application User does not want to use the Selected image as Background image for the extracted objects, he/she can save the selected image in mobile's Gallery and move back to previous interface.
Pre-Condition	The application should be able to access android mobile phone gallery and camera.
Post Condition	The Application User can now choose what to do with the selected image.
Extends	*Merge Object with Background*
Include	*Capture Image, From Gallery*
Assumptions	The Application User Selects an image from mobile Gallery or takes an image from mobile camera and chooses what to do with that image.

Use Case 7

Use Case Name	**Merge Object with Background**
Primary Actor	Application User
Secondary Actor	N/A

Normal Course	Application User selects an Object from Gallery that he/she wants to merge with the Background Image. The Selected object is placed on the desired location of the Background Image and merged with it.
Alternate Course	N/A
Pre-Condition	The Object and the Background image must be saved in Gallery. Application User must be able to choose the Object and the Background Image.
Post Condition	The Object is merged on the Background Image at the desired location.
Extends	*N/A*
Include	*Blend Object with Background*
Assumptions	Application User selects an Object from Gallery that he/she wants to merge with the Background Image. The Selected object is placed on the desired location of the Background Image and merged with it.

Use Case 8

Use Case Name	**Blend Object with Background**
Primary Actor	Application User
Secondary Actor	N/A
Normal Course	Application User blend the object with the Background Image, using OpenCV core libraries, by pressing the button "Blend Object".
Alternate Course	If Application user does not want to Blend the Object with the Background Image, he/she can save the image, with the object merged, in the Gallery.

Pre-Condition	The Object must be merged on the Background image.
Post Condition	The image, which is result of Object being blended with the Background Image, is displayed and saved in mobile phone's Gallery.
Extends	*N/A*
Include	*N/A*
Assumptions	Application User blend the object with the Background Image, using OpenCV core libraries, by pressing the button "Blend Object".

Use Case 9

Use Case Name	**Capture Image**
Primary Actor	Application User
Secondary Actor	Camera
Normal Course	Application User uses the application to access the mobile phone's camera and captures image, which is displayed and saved in Gallery.
Alternate Course	N/A
Pre-Condition	Android mobile phone must have a working camera.
Post Condition	The new image is saved in android mobile phone's Gallery and is available to application user to use.
Extends	*N/A*
Include	*N/A*
Assumptions	Application User uses the application to access the mobile

	phone's camera and captures image, which is displayed and saved in Gallery.

Use Case 10

Use Case Name	**From Gallery**
Primary Actor	Application User
Secondary Actor	Gallery
Normal Course	Application User uses the application to access the android mobile phone's Gallery.
Alternate Course	N/A
Pre-Condition	Android mobile phone must have an Image in Gallery.
Post Condition	The image is available to application user to use.
Extends	*N/A*
Include	*N/A*
Assumptions	Application **User** uses the application to access the android mobile phone's Gallery.

Sequence Diagrams

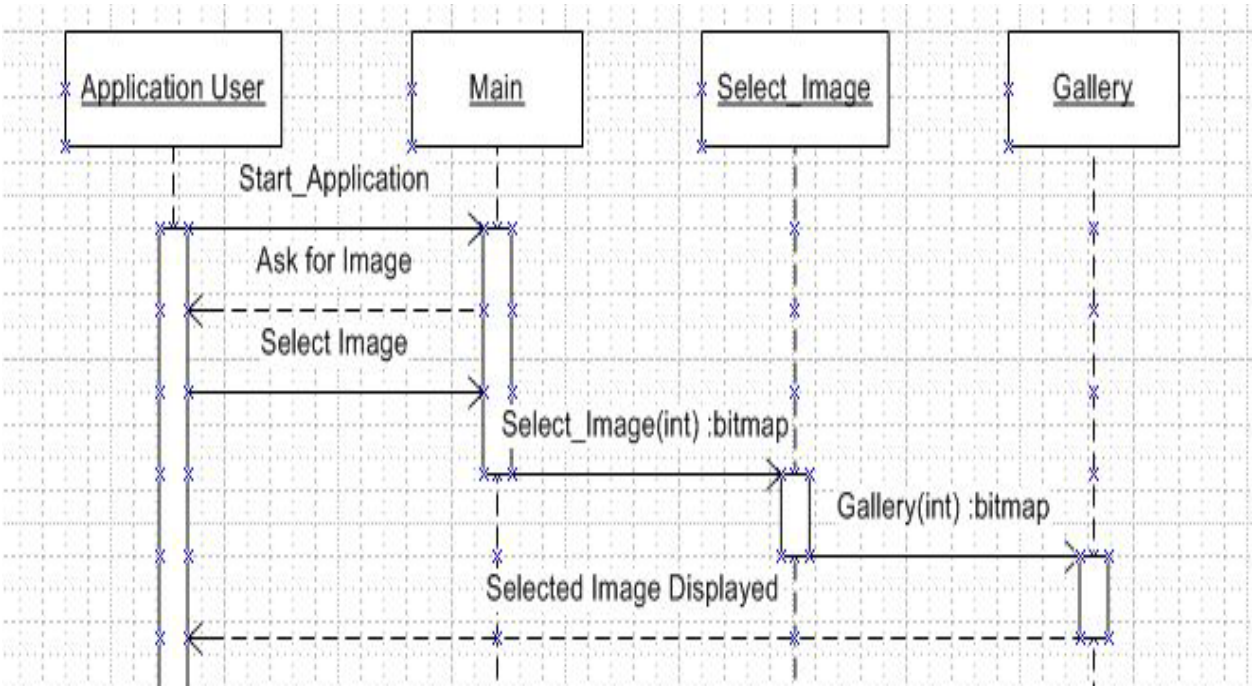


Fig. 4.3.2.2: Select Image from Gallery

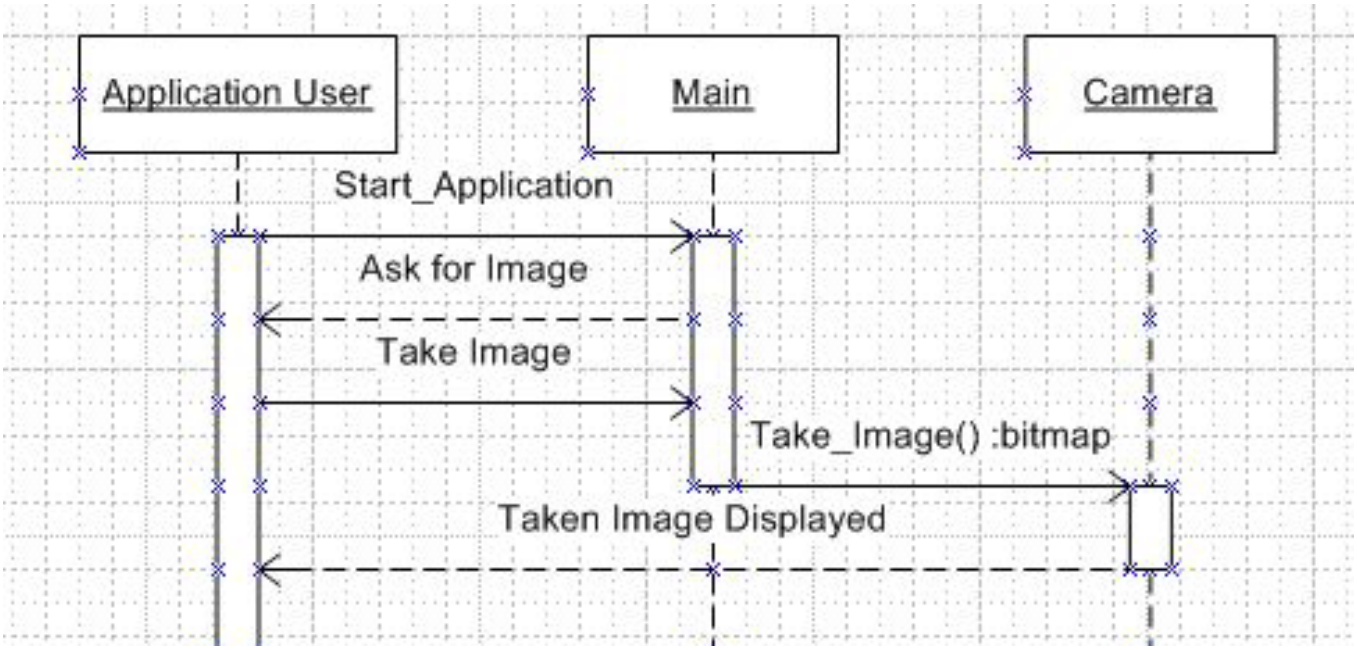


Fig. 4.3.2.3: Capture Image from Camera

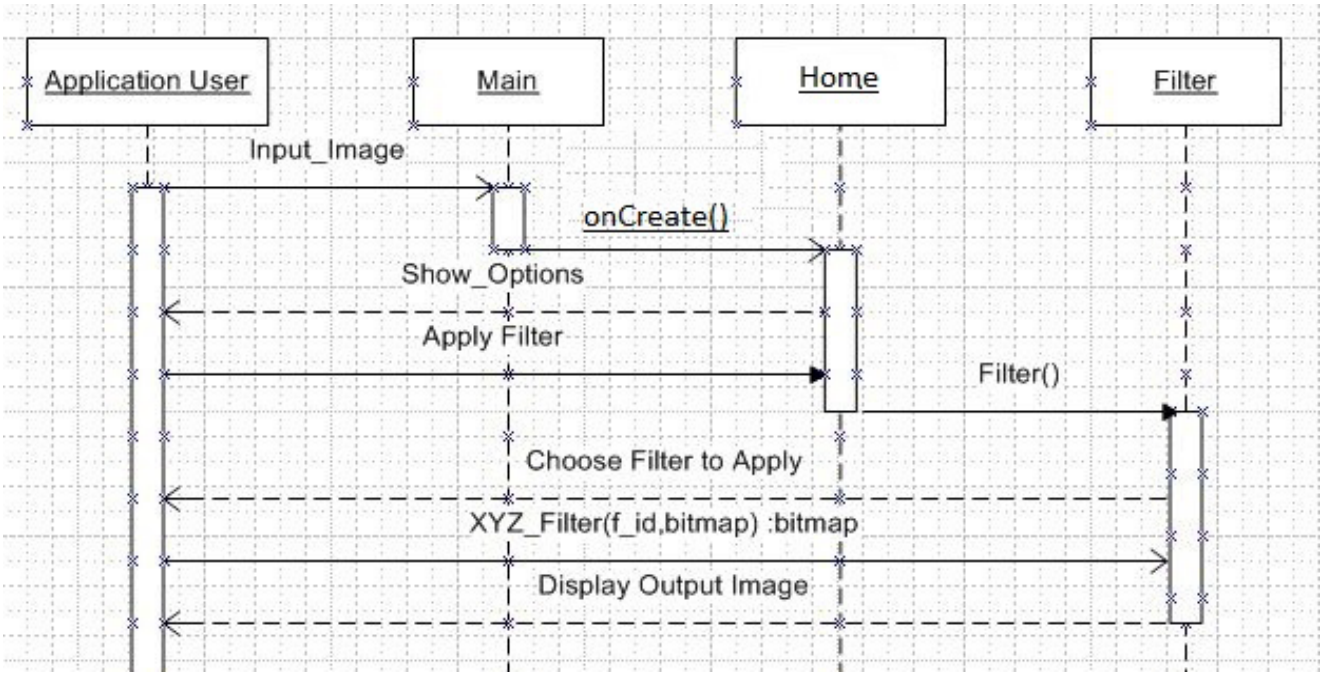


Fig. 4.3.2.4: Apply Filter to Image

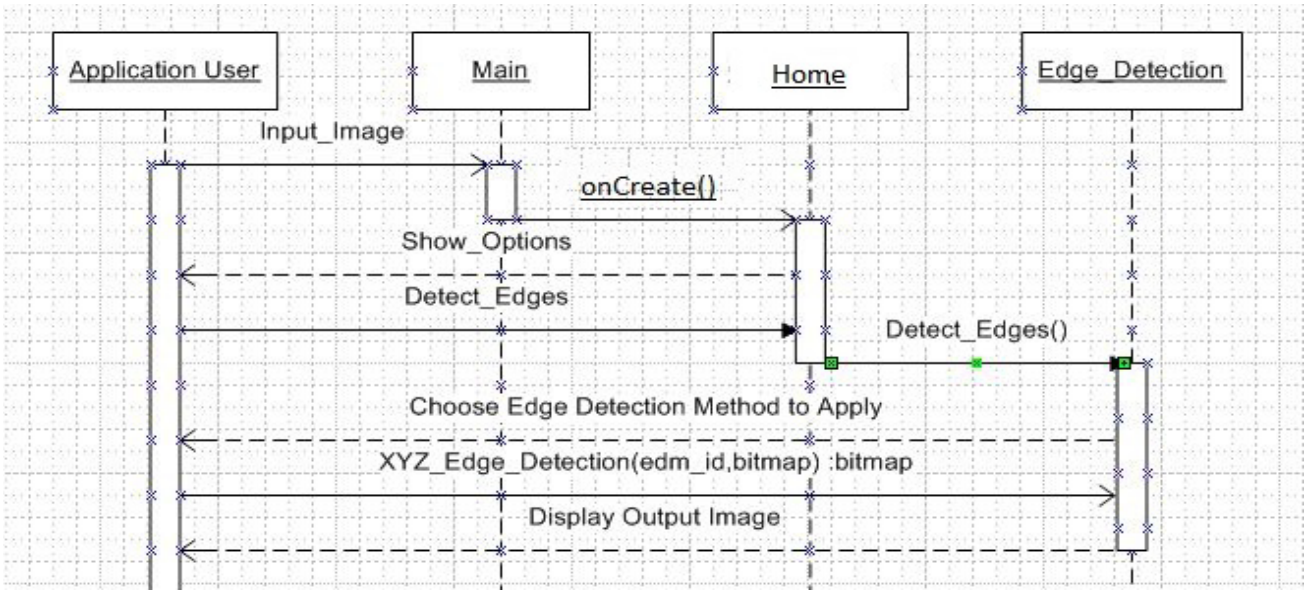


Fig. 4.3.2.5: Detect Edges

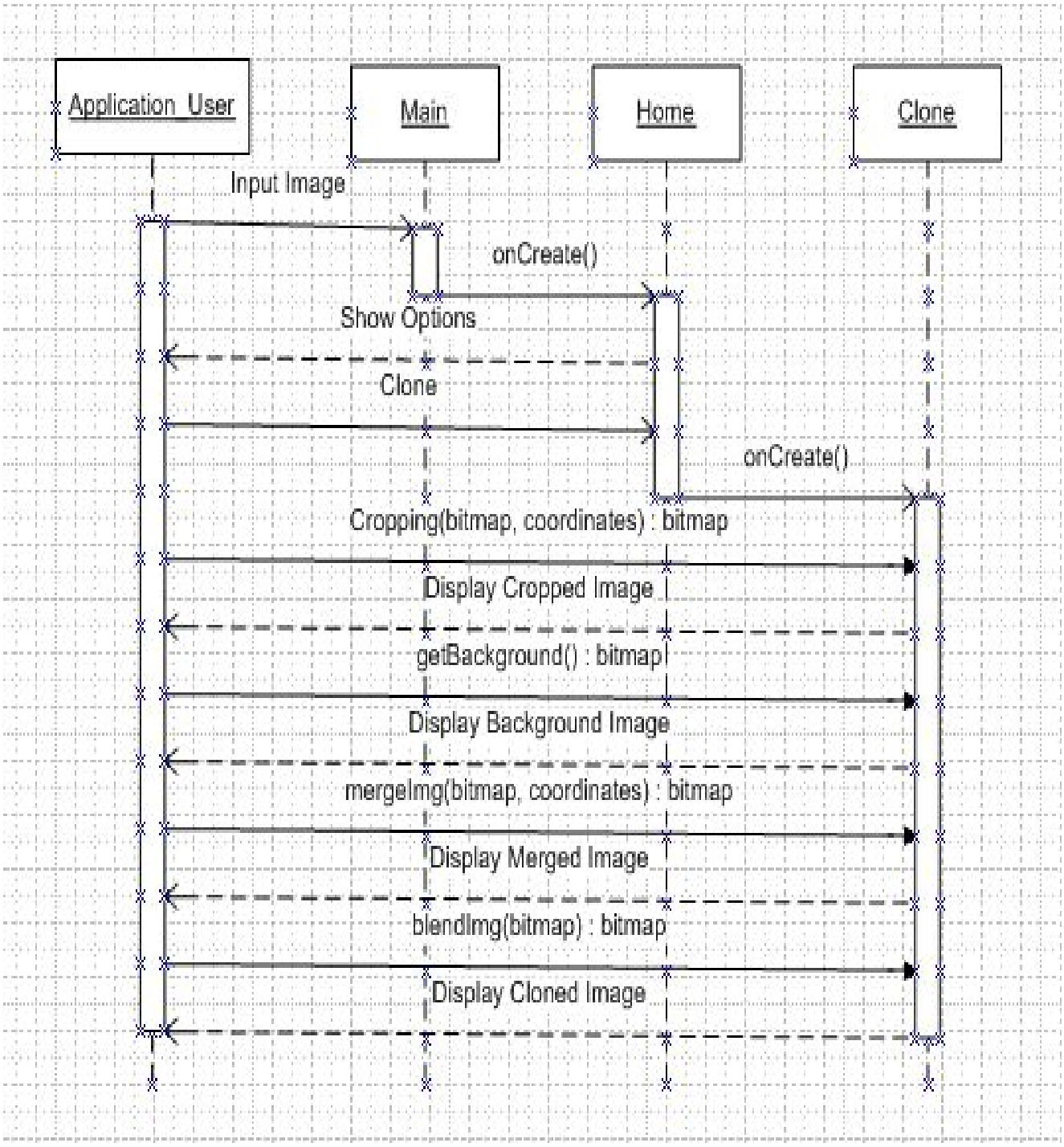


Fig. 4.3.2.6: Clone Image

Class Diagram

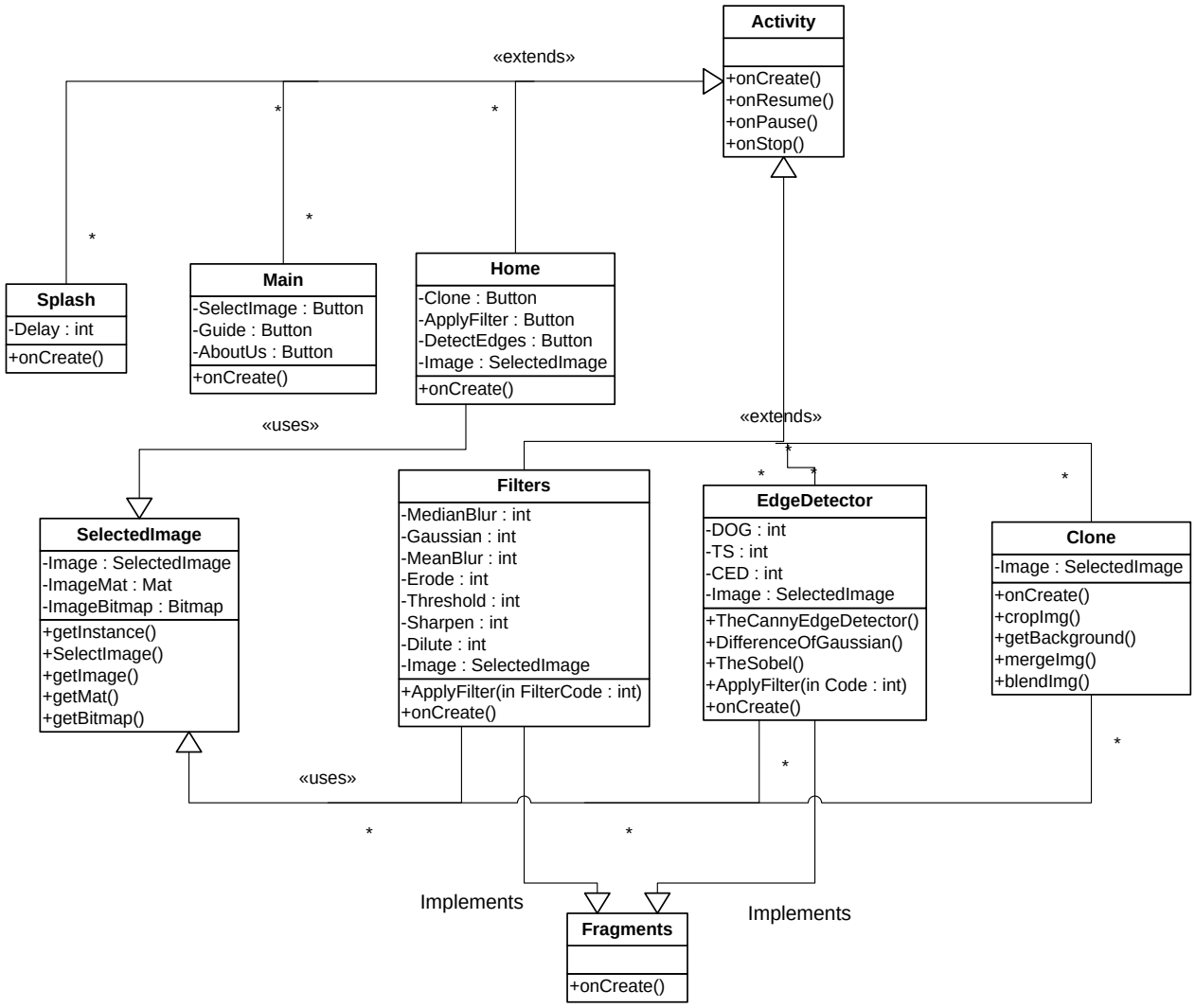


Fig. 4.3.2.7: Class *diagram for piClone*

Classes	Description
Activity	This is the main Activity Class, all Activities extends this Activity Class.
Splash	This class extends Activity class. It is responsible to display the Splash screen. The Splash screen features a logo of the application, and 3 other logos, the one on the left is of Military College of Signals (MCS), the one on the right is of National University of Science and Technology (NUST). The one in the center is the logo of Bull Incarnadine which is the Parent Company. Underneath is the name of the Application, followed by the names of the creators.
Home	This class also extends Activity. Here the user can choose from 3 different options. He can start using the core features of the app by clicking on the **Select Picture**. Can access help by clicking on the **Guide** button and can get info about the app and its founders or give feedback by getting access to the contacts of the founders by clicking on the **About Us** button. By Clicking on the Select Picture, the user would be asked to choose picture either by selecting it from the phone Gallery or by capturing a picture using the phone Camera.
Main	This class extends Activity. After the picture is selected, the Main Screen Activity would be displayed which would show the selected picture on an Image View. Here the user is further prompted as to what he/she wants to do with the picture. Either **Apply Filters** or **Detect Edges** or the user can select the core feature, **Clone** Option to start the cloning process.
Filter	The user can apply various filters to the image if he selects the Apply Filter option. If the user does select this option, a screen

	similar to Fig 2.2.8.4 would appear where there are two Image Views, one that displays the original picture and the other that displays the picture after a Filter is applied to it. The name of the Filter would be displayed to identify the Filter. This Image View would be placed in a Fragment that can be swiped left and right to change different filters to see a more dynamic and easy flow between changing of filters
EdgeDetector	The user can select the **Detect Edges** option from the Main Screen Activity which leads to the user been given choices to apply various algorithms. **The Detect Edges** Activity and Fragments is similar to that of **Apply Filter** Activity. **Fig** shows the **Detect Edges** Activity screen where **The Canny Edge Detector** has been applied to the Image.
Clone	The user can select the **Clone** option from the Main Screen Activity which takes the user to the **Crop Screen Activity**. **Fig** shows the user being able to choose the object that is desired by trapping it in a bounding box. To confirm his selection he can press the **Save** button, or if he decides to cancel, the user can press **Cancel**. When the desired region containing the desired object is selected, a new cropped Image is made. This then leads to the Add Background Image Screen, where the user is prompted to select a base Image from either the phone Gallery or the Phone Camera. After which the user can place the cropped image to which ever location on top of the base image, and select confirm. After which the two images would then be merged. **Fig** shows an example cropped Image, which is then placed on top of the base image as shown in **Fig** and then Merged. After the two Images have been merged together, blending

	would take place to make the image look more natural and real as shown in **Fig.** This completes the cloning procedure after which the user can save or discard image. Then the user would be taken to the Main Screen Activity (**Fig**) with the new Blended Image as the selected Image.
SelectedImage	This is a singleton method. Here the selected image's URI would be stored. The image's Bitmap and Mat would be saved here and be easily accessible using simple get methods.
Fragments	In the Activities that implements fragments, pager adapter can be used to create a beautiful flow between different fragments. When different filters or edge detection would be applied to different images, they could be easily shown on different Fragments.

Logical View (State Transition Diagram)

The State Transitions occurring in the application are shown in **Fig. 4.3.2.8** below:

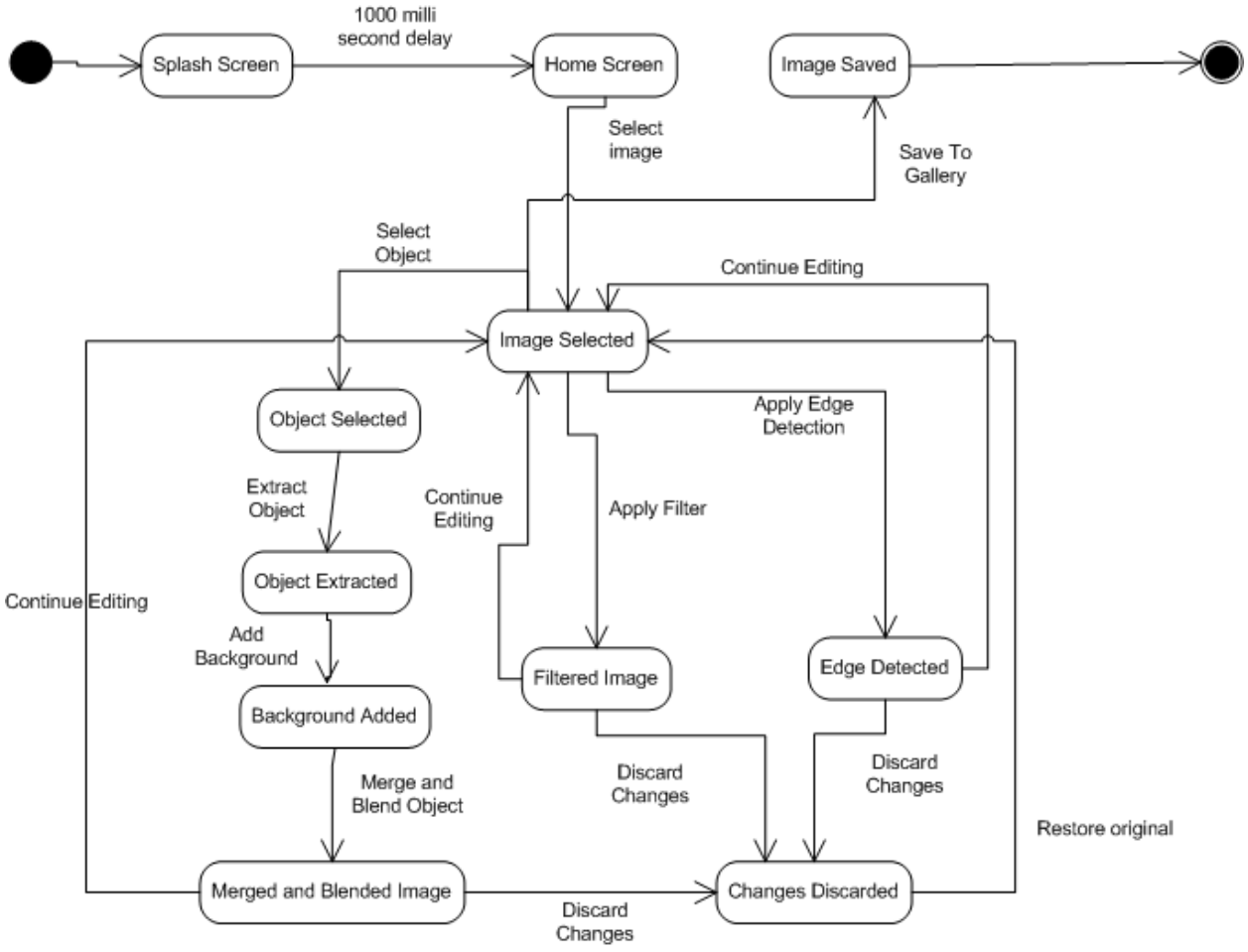


Fig. 4.3.2.8: *State diagram for piClone*

Dynamic view (Activity Diagram)

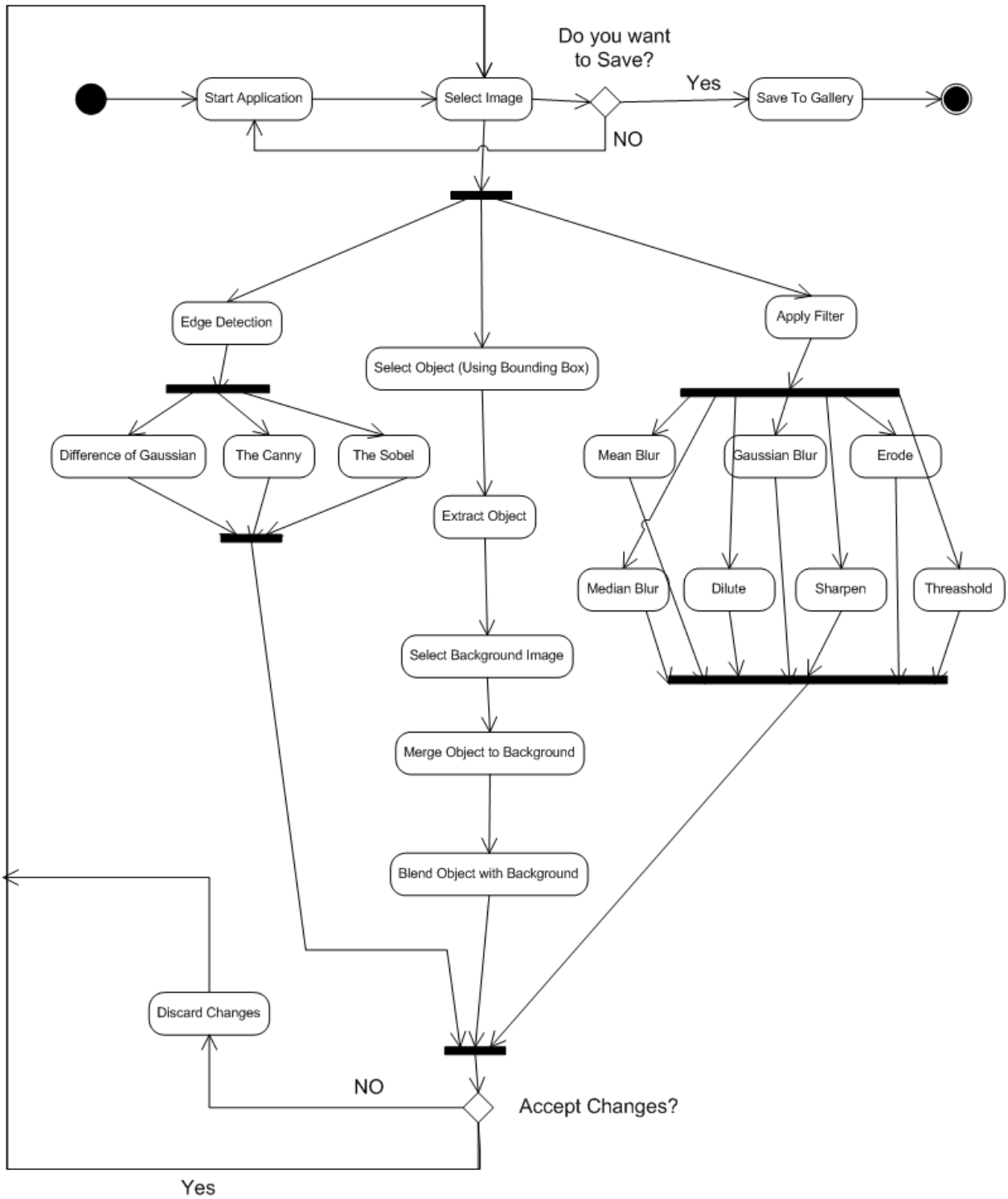


Fig. 4.3.2.9: *Activity diagram for piClone*

In activity diagram, the dynamic view of the system is shown. All the activities are shown concurrently with their respective start and end states.

Android Activity Lifecycle

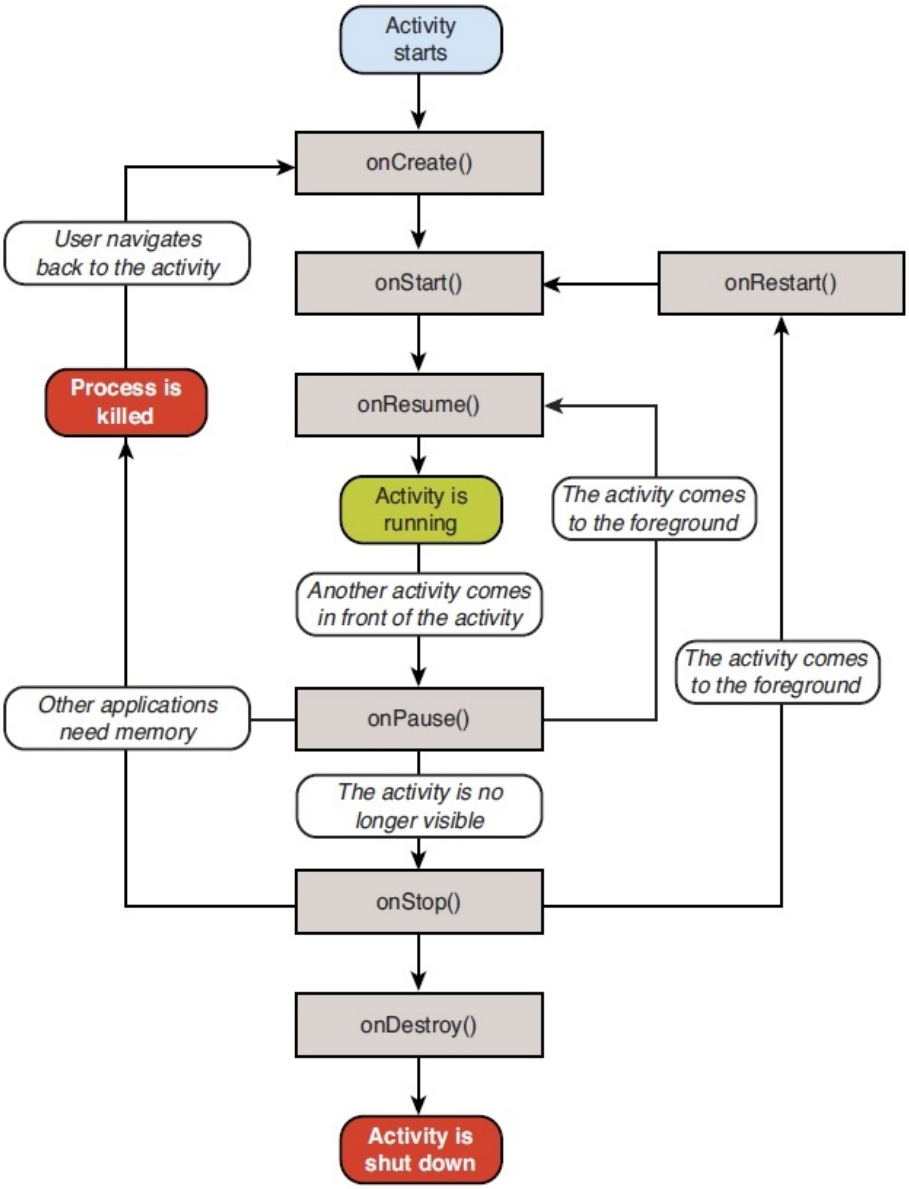


Fig. 4.3.2.10:*Activity Life Cycle diagram used inpiClone*

Android Fragment Lifecycle

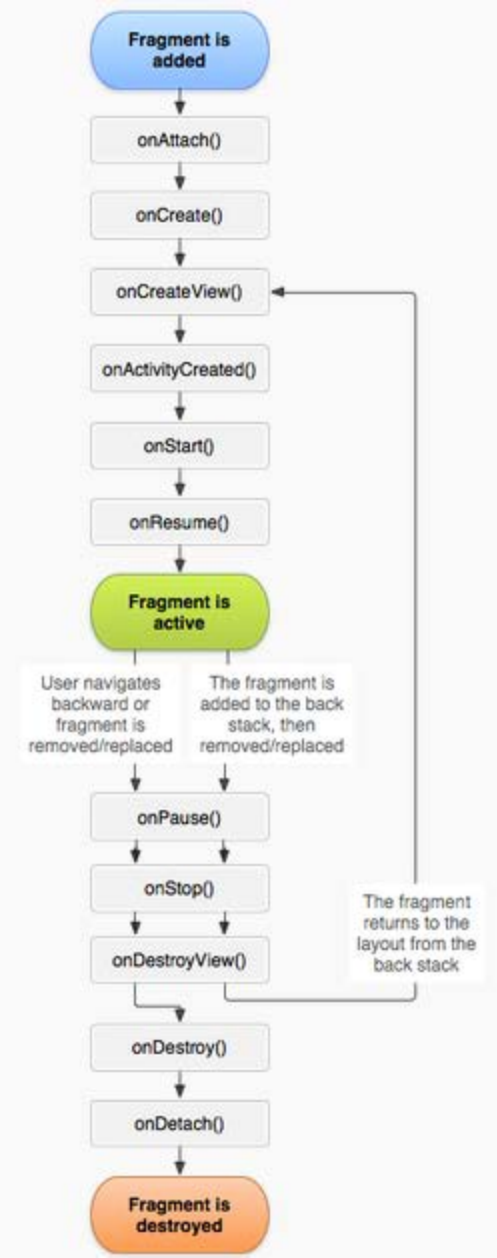


Fig. 4.3.2.11: *Fragment Lifecycle diagram used inpiClone*

Structure Chart

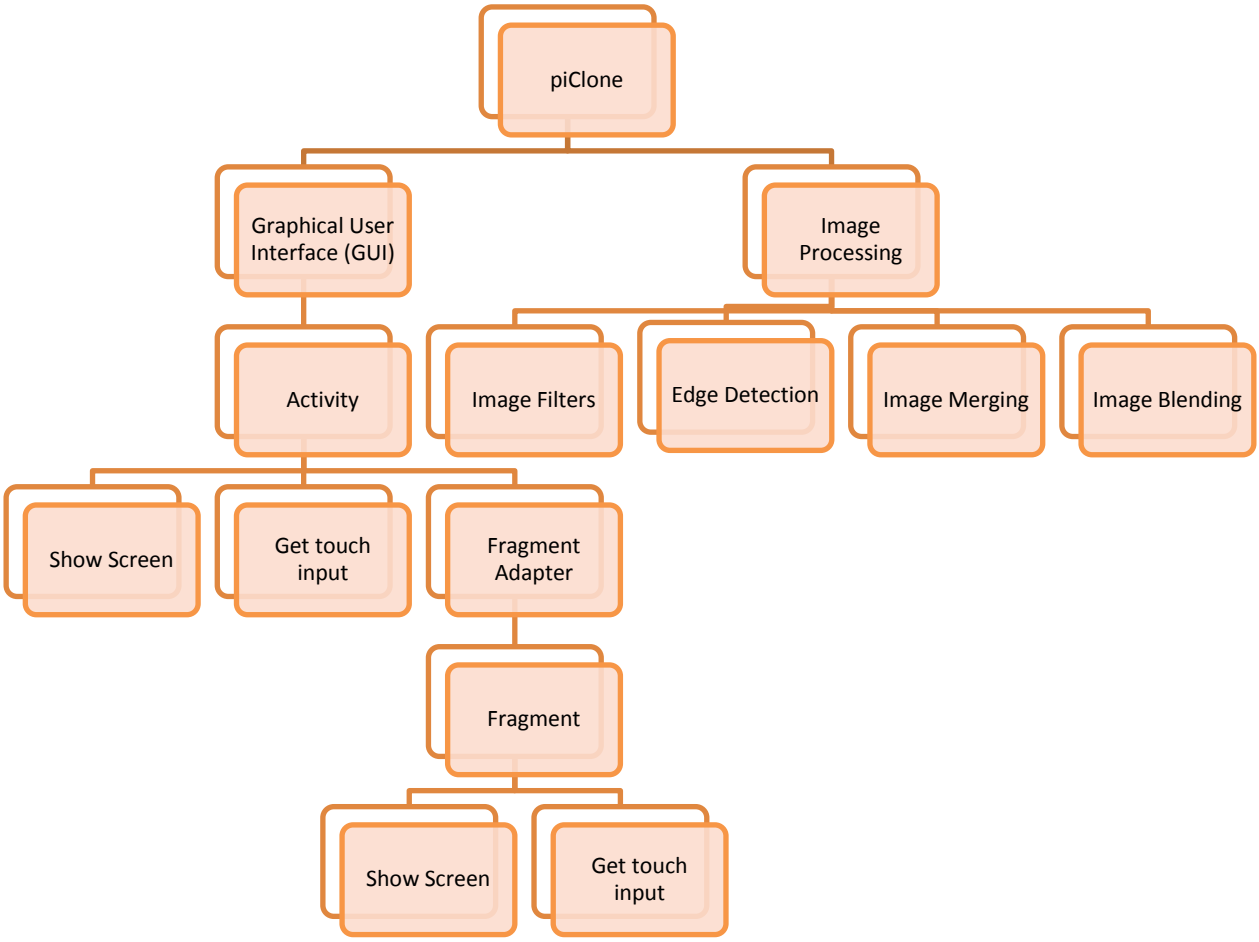


Fig. 4.3.2.12: *Structure Chart diagram used inpiClone*

Work Breakdown Structure

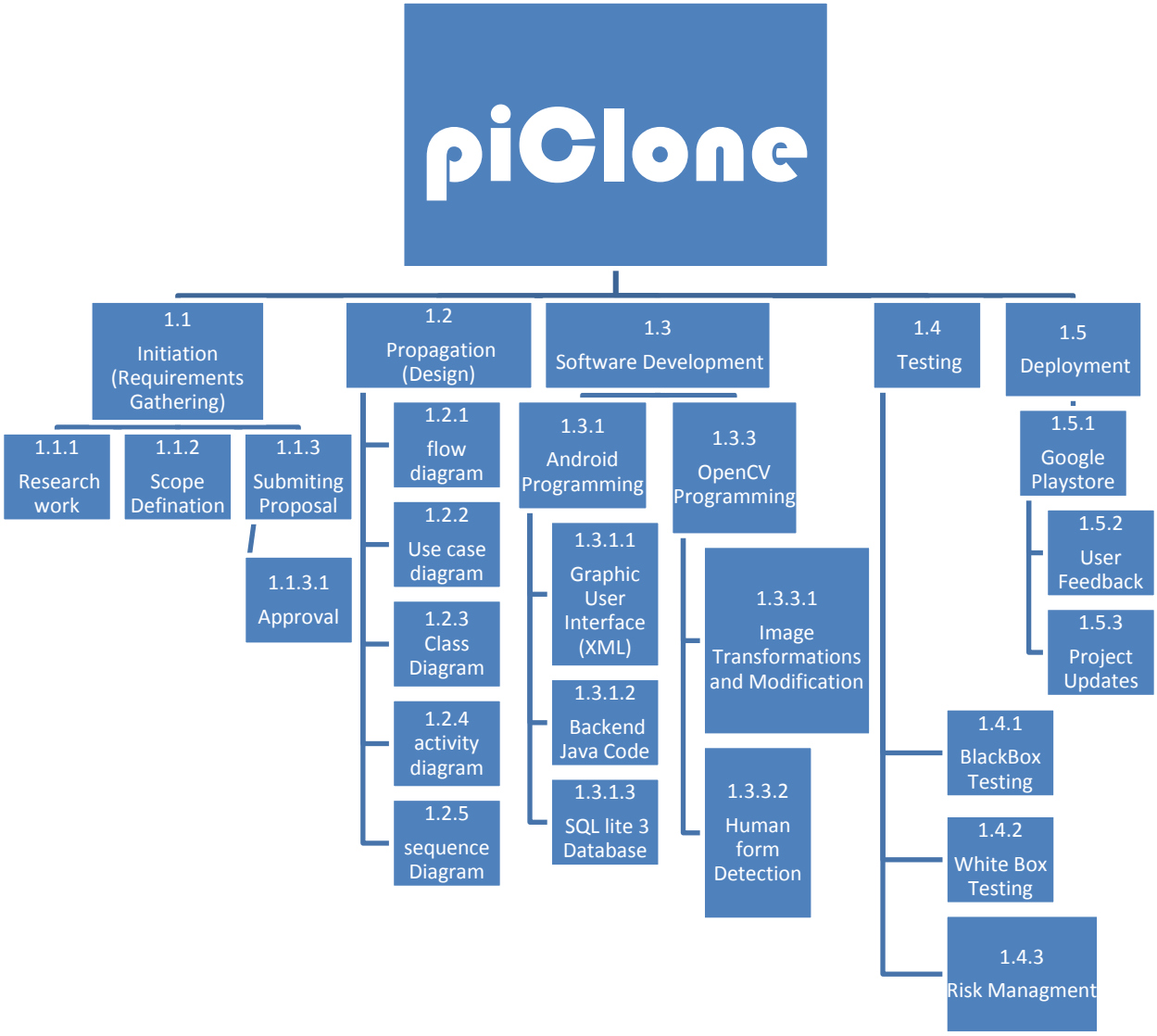


Fig. 4.3.2.13:*Fragment Lifecycle diagram used inpiClone*

User Interface

The Splash screen features a logo of the application, and 3 other logos, the one on the left is of Military College of Signals (MCS), the one on the right is of National University of Science and Technology (NUST). The one in the center is the logo of Bull Incarnadine which is the Parent Company. Underneath is the name of the Application, followed by the names of the creators.

Fig. 4.3.2.14: *Splash Screen*

The splash screen will appear for a limited time of 1000 mille seconds, after which the Home Screen Activity would be displayed to the user. Here the user can choose from 3 different options. He can start using the core features of the app by clicking on the **Select Picture**. Can access help by clicking on the **Guide** button and can get info about the app and its founders or give feedback by getting access to the contacts of the founders by clicking on the **About Us** button.

Fig. 4.3.2.15: *Home Screen*

By Clicking on the Select Picture, the user would be asked to choose picture either by selecting it from the phone Gallery or by capturing a picture using the phone Camera. After the picture is selected, the Main Screen Activity would be displayed which would show the selected picture on an Image View. Here the user is further prompted as to what he/she wants to do with the picture. Either **Apply Filters** or **Detect Edges** or the user can select the core feature, **Clone** Option to start the cloning process.

Fig. 4.3.2.16: *Main Screen*

The user can apply various filters to the image if he selects the Apply Filter option. If the user does select this option, a screen similar to fig would appear where there are two Image Views, one that displays the original picture and the other that displays the picture after a Filter is applied to it. The name of the Filter would be displayed to identify the Filter. This Image View would be placed in a Fragment that can be swiped left and right to change different filters to see a more dynamic and easy flow between changing of filters.

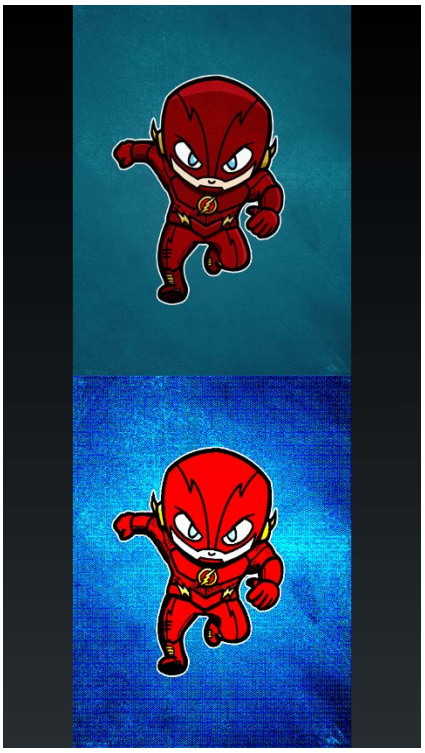


Fig. 4.3.2.17: *The bottom image has been applied with* ***Binary Threshold Filter***

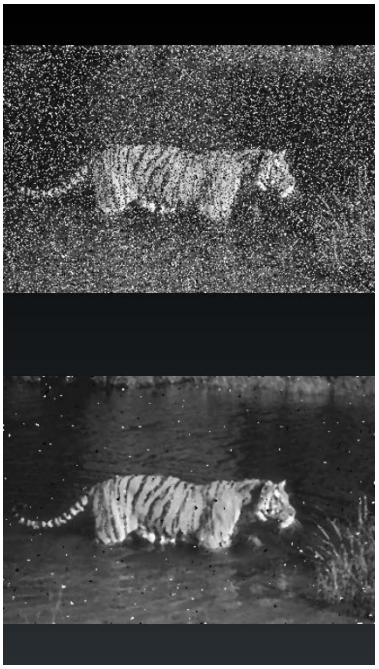


Fig. 4.3.2.18: *The salt and pepper noise is removed in the bottom image by* ***Median Filter***

The user can select the **Detect Edges** option from the Main Screen Activity which leads to the user been given choices to apply various algorithms. **The Detect Edges** Activity and Fragments is similar to that of **Apply Filter** Activity. **Fig** shows the **Detect Edges** Activity screen where **The Canny Edge Detector** has been applied to the Image.

Fig. 4.3.2.19: *The Canny Edge Detector Algorithm Applied to the Selected Image*

The user can select the **Clone** option from the Main Screen Activity which takes the user to the **Crop Screen Activity**.The user being able to choose the object that is desired by trapping it in a bounding box. To confirm his selection he can press the **Save** button, or if he decides to cancel, the user can press **Cancel**.

Fig. 4.3.2.20: *Cropping Image*

When the desired region containing the desired object is selected, a new cropped Image is made. This then leads to the Add Background Image Screen, where the user is prompted to select a base Image from either the phone Gallery or the Phone Camera. After which the user can place the cropped image to which ever location on top of the base image, and select confirm. After which the two images would then be merged. Shows an example cropped Image, which is then placed on top of the base image as shown and then Merged.

Fig. 4.3.2.21: Cropped Image

Fig. 4.3.2.22: Merged Image

After the two Images have been merged together, blending would take place to make the image look more natural and real as shown**.** This completes the cloning procedure after which the user can save or discard image. Then the user would be taken to the Main Screen Activity with the new Blended Image as the selected Image.

Fig. 4.3.2.23: Blended Image (cloning completed)

4.3.3 Detailed Description of Components

Graphical User Interface

Identification	*Name:* Graphical User Interface *Location:* Presentation layer of the system architecture
Type	UI component
Purpose	The user directly interacts with this component. He/she provides an input for the required action (through this component) and it displays its output respectively. This component fulfills following functional requirements (as specified in SRS Document) related to user interaction in the application: **REQ-1:** Application should allow the user to select the desired image either by choosing from gallery or by capturing from camera. **REQ-2:** The user can then choose how to modify or edit the selected image. **REQ-3:** The application should display the selected Image. **REQ-4:** The application should display the result Image after going though different Image Processing. **REQ-5:** The application should allow the user to save progress when changes are made. **REQ-6:** The application should help guide the user to use the application. **REQ-7:** The application should be able to transition from and to Activities when required.

	REQ-8: The application should make the use of fragments where necessary.
Function	This component has two major functions; take input from the user and display all application activities and fragments. It takes input from user in form of touch events, and provides a graphical output accordingly.
Subordinates	This component has two subordinates; one is responsible for input, the other for output. The input subordinate satisfies all functional requirements (mentioned in the SRS Document) that require user input: REQ-2, REQ-3 and REQ-7. The output subordinate satisfies all functional requirements (mentioned in the SRS Document) that provide output: REQ-1, REQ-2, REQ-3, REQ-4, REQ-5, REQ-6, REQ-7, REQ-8.
Dependencies	It interacts with *Image Processing* (refer Section 3.3), whenever a user interacts with the application. This component is dependent on the *Graphical User Interface* whereas no component depends upon it.
Interfaces	All user interfaces defined in section 2.3 are part of it. The user input and output on screens will be shown using these interfaces. It will provide external interface to *Process Data* in form of inputs taken from the user. It will display guide Toasts like:

	1. Image Selected 2. Filter applied 3. Edges detected 4. Image cropped 5. Image merged 6. Image cloned successfully 7. Image Saved 8. Unable to Save *(Gallery related)* 9. Unable to Retrieve *(Gallery related)*
Resources	**Hardware:** Touch screen enable user to interact with the application. It will require a screen to display the application. The screens will be run by using internal memory i.e.; RAM, Processor of the Android Mobile Device or Tablet. **Software:** Android Studio to design interface for display on screen.
Processing	Takes user input in form of touch screen and shapes the output according to user intent. (Refer Section 6.1)
Data	Image processing etc…

Image Processing

This component has 5 sub-components:

Image Cropping

Identification	*Name:* Image Cropping *Location:* Application Logic layer of the system architecture
Type	Sub-component
Purpose	Following functional requirements mentioned in SRS are fulfilled by this sub-component:

	REQ-9: The application should be able to select the desired object by allowing the user to place a bounding box around it as desired. **REQ-10:** The application should extract the desired segment of the Image enclosed by the bounding box.
Function	The function of this sub-component is to get the input from the user that would help choose which portion or object of the image, the user is interested in so that it can be extracted. For selecting the Image region, it takes input from *touch screen* component (refer Section 3.5) and shows output on the *Graphical User Interface* component (refer Section 3.1). The cropped Image would then be showed to the user as output on the Graphical User Interface component.
Subordinates	It has two subordinates, *Create new bitmap* and *copying the desired region from the image to the bitmap*.
Dependencies	This dependent is dependent on *Graphical User Interface* (refer Section 3.5) Some of the functions of this component are: 'Imgproc.rectangle(contoursFrame, new Point(rect.x, rect.y), new Point(rect.x+rect.width, rect.y +rect.height), new Scalar(255,0,0) , 1,8,0)' etc.
Interfaces	It provides internal and external interface to component 3.1

	'Graphical User Interface' in form of input which it gets from the user and then displays the output image after processing
Resources	Hardware: RAM and Processor of the system will be utilized. Software: OpenCV core Libraries.
Processing	The component handles the cropping of the Image in the application. It takes input in form of key touch events from other components (refer fig 2.1.1), aids the user using bounding box and generates a cropped Image according to the user intent. (Refer section 6.5).
Data	Bitmap, Image Mat and Vector points.

Edge Detection

Identification	*Name:* Edge Detection *Location:* Application Logic layer of the system architecture
Type	Sub-Component
Purpose	Following functional requirements are fulfilled by this sub-component: **REQ-11:** The application should allow the user to choose from various Edge Detection Algorithms. **REQ-12:** The application should display the output image after applying the selected Edge Detection Algorithm.
Function	The function of this sub-component is process the Image using various Edge Detection Algorithms and to display the output in an image view. Edge Detection Algorithms are stated as follows: a. Difference of Gaussian

	b. The Canny Edge Detection c. The Sobel It takes the Edge Detection options input from the component 3.1 'Graphical User Interface' and then opens up the corresponding Activity screen to show the result.
Subordinates	It has two subordinates, GUI interaction and Image Processing. GUI interaction will ask the user to select Edge Detection Algorithm of his choice and it will fulfill Req-16 of functional requirements mentioned above.
Dependencies	This subcomponent provides service to the component 3.1 'GUI' Some of the Functions of this component are: Mat DifferenceOfGaussian(Mat originalMat), Mat Soble(Mat originalMat).
Interfaces	It provides internal and external interface to component 3.1 'Graphical User Interface' in form of input which it gets from the user and then displays the output image after processing.
Resources	Hardware: RAM and Processor of the system will be utilized. Software: OpenCV core libraries
Processing	The component handles the Detection of Edges of the Image in the application. It takes input in form of key touch events from other components (refer fig 2.1.1), displays the output in an Image view according to the user intent. (Refer section 6.5).
Data	Bitmap, Image Mat and Vector points.

Image Filters

Identification	*Name:* Image Filters *Location:* Application Logic layer of the system architecture
Type	Sub-Component
Purpose	Following functional requirements are fulfilled by this sub-component: **REQ-13:** The application should allow the user to choose from various Image Filters. **REQ-14:** The application should display the output image after applying the selected Image Filters.
Function	The function of this sub-component is processing the Image using various Image Filters and to display the output in an image view. These Filters include the following: d. Mean Blur e. Median f. Gaussian g. Threshold h. Erode i. Sharpen It takes the Filter Image options input from the component 3.1 'Graphical User Interface' and then opens up the corresponding Activity screen to show the result.
Subordinates	It has two subordinates, GUI interaction and Image Processing.

	GUI interaction will ask the user to select Image Filter of his choice and it will fulfill Req-16 of functional requirements mentioned above.
Dependencies	This subcomponent provides service to the component 3.1 'GUI'. Some of the function of these components are: Mat DifferenceOfGaussian(Mat originalMat), Mat Soble(Mat originalMat).
Interfaces	It provides internal and external interface to component 3.1 'Graphical User Interface' in form of input which it gets from the user and then displays the output image after processing.
Resources	Hardware: RAM and Processor of the system will be utilized. Software: OpenCV core libraries
Processing	The component handles the Detection of Edges of the Image in the application. It takes input in form of key touch events from other components (refer fig 2.1.1), displays the output in an Image view according to the user intent. (Refer section 6.5).
Data	Bitmap, Image Mat and Vector points.

Image Merging

Identification	*Name:* Image Merging *Location:* Application Logic layer of the system architecture
Type	Sub-Component
Purpose	Following functional requirements are fulfilled by this sub-component:

	REQ-15: The application should allow the user to select a background Image from either the phone Gallery or capture it using the phone camera, using Android APIs. **REQ-16:** The application should allow the user to place his primary cropped image to the desired location of the background Image using Drag and Drop feature. **REQ-17:** The application should combine the two images, primary and the background to make a new Image after getting the go ahead from the user.
Function	The function of this sub-component is to merge both the primary image containing the desired object and the background Image together to form a new Image. It takes the Merge Image options input from the component 3.1 'Graphical User Interface' and then opens up the corresponding Activity screen to show the result. Control of the application is transferred to component 3.2.5 'Image blending'.
Subordinates	It has two subordinates, GUI interaction and Image Processing. GUI interaction will ask the user to select Background Image of his choice and place the primary Image to the desired vector location in relation to the background image and it will fulfill Req-19 of functional requirements mentioned above.
Dependencies	This subcomponent provides service to the component 3.1 'GUI'. Some of the function of these components are: Mat DifferenceOfGaussian(Mat originalMat), Mat Soble(Mat originalMat).

Interfaces	It provides internal and external interface to component 3.1 ‘Graphical User Interface’ in form of input which it gets from the user and then displays the output image after processing. Control of the application is transferred to component 3.2.5 ‘Image blending’.
Resources	Hardware: RAM and Processor of the system will be utilized. Software: OpenCV core libraries, Android APIs
Processing	The component handles the Merging of the Image in the application. It takes input in form of key touch events from other components (refer fig 2.1.1), displays the output in an Image view according to the user intent. (Refer section 6.5).
Data	Bitmap, Image Mat and Vector points.

Image Blending

Identification	*Name:* Image Blending *Location:* Application Logic layer of the system architecture
Type	Sub-Component
Purpose	Following functional requirements are fulfilled by this sub-component: **REQ-18:** The application should after the images have been merged to form a new image, be able to make them blend into one another in order to look more natural and realistic.
Function	The function of this sub-component is to blend both the primary image containing the desired object and the background Image together in the new Image.

Subordinates	It has one subordinate, Image Processing.
Dependencies	This subcomponent provides service to the component 3.1 'GUI'. Some of the function of these components are: Mat PissonBlending(Mat originalMat), Mat SimpleBlend(Mat originalMat).
Interfaces	It provides external interface to component 3.1 'Graphical User Interface' displaying the output image after processing.
Resources	Hardware: RAM and Processor of the system will be utilized. Software: OpenCV core libraries,
Processing	The component handles the Blending of the Images in the application. It displays the output in an Image view according to the user intent. (Refer section 6.5).
Data	Bitmap, Image Mat and Vector points.

4.4 Reuse and Relationship to other products

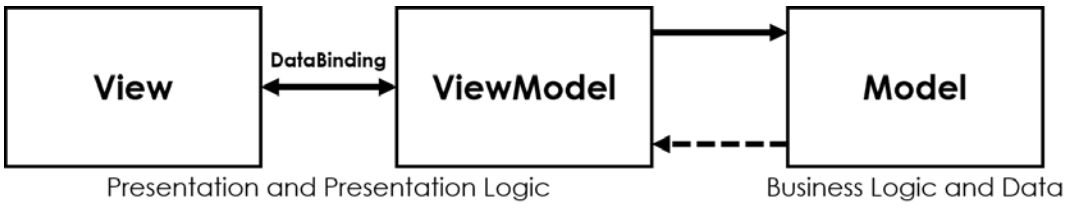
piClone project can be extended to make a complete image editor application to Photoshop an image. All the code will be available on GitHub for anyone to learn and use in their projects. More functionalities can be introduced like: edit or render text, vector graphics (especially through clipping path), 3D graphics and video

4.5 Design and Tradeoffs

The piClone is an interactive application that uses a simple, easy to use user interface. There are a few activity classes but more might be added depending on the addition of more features. The Singleton Selected Image Class will help pass the resources of an image with relative ease. The application is designed as 2-layered with the Activity Classes forming the upper layer for user interface and the lower layer providing the algorithms and functionalities of image processing using only internal libraries in Android like the OpenCV itself. The design does not support the usage of any external library so in case of requiring to use any external functionalities the design might go under tremendous changes.

Interface of the system is distinct from the application logic. Layered architecture is used to isolate application logic from the user interface. It can be modeled using **Multitier Layered Architecture** consisting of three layers i.e.; presentation, application logic and Device Storage. Presentation layer corresponds to elements of the user interface such as text, checkbox item etc., and application logic layer controls the communication of data between the presentation and the Storage layer, and is the part where the main logic, user actions and working of the system is defined. In general, it controls the complete behavior of the system, while the Storage layer is responsible for handling, fetching and storage.

As per the interface and business logic goes, piClone applications follow Model View View Model (MVVM) design pattern **[9]**, which is largely based on model-view-controller (MVC) pattern. It is a specific implementation targeted at UI development platforms which support event-driven programming on the Android platforms using XML and Java.



Chapter 5: Testing and Evaluation

5.1 Introduction

This test plan chapter describes the appropriate strategies, process and methodologies used to plan, execute and manage testing of the piClone Android application project. The test plan will ensure that piClone meets the customer requirements at an accredited level.

Manual Testing will be followed which includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. Each Unit will be tested separately and then will be integrated with other units, therefore Unit Testing and Integration testing will be followed. For each unit Black box Testing is done and for combined units Acceptance Testing is done.

The test scope includes the Testing of all functional, application performance and use cases requirements listed in the *requirement document*

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed.

This document includes the plan, scope, approach and procedure of piClone test. The pass/fail criteria of the test items are also defined. The Test Plan document documents and tracks the necessary information required to effectively define the approach to be used in the testing of the product.

5.2 Test Items

Based on the piClone requirements and design description, application modules of mobile Android application and non-functional scenario will be tested. The Requirements Defined in Software Requirements Specification and the Design entities as explained in Software Design Document will be tested.

5.3 Features tested

Following Features are tested:

a) Ability to Select Desired Image from Gallery.
b) Ability to take a new image using camera.
c) Ability to crop the desired object from the selected image.
d) Ability to set background by selecting the desired background image.
e) Ability to put the desired object on the Background Image.
f) Ability to move the desired object on the Background image at the desired coordinates.
g) Ability to Merge and Blend the Desired Object on the Background Image.
h) Ability to Apply Filter on the selected image with accuracy.
i) Ability to detect edges of the desired image accurately.

5.4 Approach

Acceptance test will be executed based on this acceptance test plan. And after all test cases are executed, a test report will be summarized to show the quality of piClone. Following test approaches will be used in test execution:

- **Unit test**. Developers are responsible for unit test as white-box testing. The implementation of each module and individual component will be verified separately.
- **Integration test**. After the unit test is passed above the defined quality threshold, testers will execute the integration test cases. After all the modules are integrated, it's crucial to test the product as a black-box. End-to-end scenarios will be tested to ensure the communication functionality.
- **Regression test**.After developers fix the bug in one feature, regression test will be executed by testers to ensure that the other functions are not affected.
- **Field test**. Firstly, untrained end users recreate one or more existing (but narrow) mass observation events in the piClone Android Application. A number of observers will be invited to help with evaluation. After that, post event questionnaires will be used to collect quantitative usage data as well as qualitative data and further improvement will be taken into consideration.
- **Positive and negative testing design technique**. This approach will be combined with unit test and integration test. Test cases are designed in obvious scenarios, which ensure that all functional requirements are satisfied. What's more, different test cases will also be covered to show how the system reacts with invalid operations.

5.5 Item Pass/Fail Criteria

Details of the test cases are specified in section Test Deliverables. Following the principles outlined below, a test item would be judged as pass or fail.

- Preconditions are met
- Inputs are carried out as specified
- The result works as what specified in output => Pass
- The system doesn't work or not the same as output specification => Fail

5.6 Suspension Criteria and Resumption Requirements

Any bugs found can be fixed by developers quickly and no need to start the testing process from the beginning. However, when major bugs will block the some test cases as they are interdependent and the testing has to be paused. The test will restart from the very beginning until the major error is solved.

5.7 Test Deliverables

Following are the Test Cases:

Test Case Name	Feature Choice Menu
Test Case No	1
Description	Testing Feature Choice Menu
Testing Technique Used	Unit Testing
Preconditions	Application should be installed in Android Operating System
Input Values	piClone application
Valid Inputs	Open the piClone application
Steps	Select the piClone android application installed in Android Operating System
Expected Output	piClone android application opens displaying Feature Choice Menu to the user
Actual Output	piClone android application opens displaying Feature Choice Menu to the user

Test Case Name	Clone Feature
Test Case No	2
Description	Testing the Clone Feature
Testing Technique Used	Unit Testing and Acceptance Testing
Preconditions	The Application must be opened displaying Feature Choice Menu
Input Values	Clone , Apply Filter, Edge detection
Valid Inputs	Choose Clone Feature by clicking on the clone button
Steps	First select the piClone android application installed in Android Operating System, then choose the Clone Feature which is displayed on Feature Choice Menu
Expected Output	The Choose Image Choice menu should open displaying two choices for selecting image
Actual Output	The Choose Image Choice menu opens displaying two choices for selecting image

Test Case Name	Select Image from Gallery (Clone Feature)
Test Case No	3
Description	Testing Select Image from Gallery Feature for Clone Feature
Testing Technique Used	Unit Testing
Preconditions	The user must have selected Clone Feature from Feature Choice Menu
Input Values	Choose from Gallery, Take Image using Camera
Valid Inputs	Choose Select Image form Gallery option by clicking on the Choose from Gallery button
Steps	First select the piClone android application installed in Android Operating System then choose the Clone Feature which is displayed on Feature Choice Menu then choose Select Image from Gallery displaying on Choose Image Menu
Expected Output	The Gallery of the android mobile phone should be accessed by piClone and should enable user to select desired image from Gallery
Actual Output	The Gallery of the android mobile phone is accessed by piClone enabling user to select desired image from Gallery

Test Case Name	Take Picture using Camera (Clone Feature)
Test Case No	4
Description	Testing Take Picture using Camera Feature for Clone Feature
Testing Technique Used	Unit Testing
Preconditions	The user must have selected Clone Feature from Feature Choice Menu
Input Values	Choose from Gallery, Take Image using Camera
Valid Inputs	Choose Take Image using Camera option by clicking on the Take Image using Camera button
Steps	First select the piClone android application installed in Android Operating System then choose the Clone Feature which is displayed on Feature Choice Menu then choose Take Image using Camera displaying on Choose Image Menu
Expected Output	The Camera of the android mobile phone should be accessed by piClone and should enable user to Take an Image
Actual Output	The Camera of the android mobile phone is accessed by piClone enabling user to Take an Image

Test Case Name	Choose the desired object (Select Image from Gallery)
Test Case No	5
Description	Testing Choose the desired object for Select Image from Gallery Feature in Cloning process
Testing Technique Used	Integration Testing
Preconditions	The user must have selected Select Image from Gallery Feature from Choose Image Menu
Input Values	
Valid Inputs	Crop the desired object from the Image which is selected from Gallery
Steps	First select the piClone android application installed in Android Operating System then choose the Clone Feature which is displayed on Feature Choice Menu then choose Select Image from Gallery Feature displaying on Choose Image Menu then choose the desired object by cropping that object from the selected image
Expected Output	The cropped object should be saved in Gallery
Actual Output	The cropped object is saved in Gallery

Test Case Name	Choose the desired object(Take Image using Camera)
Test Case No	6
Description	Testing Choose the desired object for Take Image using Camera Feature in Cloning process
Testing Technique Used	Integration Testing
Preconditions	The user must have selected Take an Image using Camera Feature from Choose Image Menu
Input Values	
Valid Inputs	Crop the desired object from the Image which is taken using camera
Steps	First select the piClone android application installed in Android Operating System then choose the Clone Feature which is displayed on Feature Choice Menu then choose Take Image using Camera Feature displaying on Choose Image Menu then choose the desired object by cropping that object from the Image which is taken
Expected Output	The cropped object should be saved in Gallery
Actual Output	The cropped object is saved in Gallery

Test Case Name	Set the Background (Select Image from Gallery)
Test Case No	7
Description	Testing Set the Background Feature for Select Image from Gallery Feature in Cloning process
Testing Technique Used	Regression Testing
Preconditions	The user must have cropped the desired object which he needs to merge on the Background
Input Values	
Valid Inputs	Select an image that needs to be Background Image for the desired object to be placed on
Steps	First select the piClone android application installed in Android Operating System then choose the Clone Feature which is displayed on Feature Choice Menu then choose Select Image from Gallery Feature displaying on Choose Image Menu then choose the desired object by cropping that object from the Image which is selected from Gallery then Choose the Background Image which is the image on which you want to place the desired object
Expected Output	The selected background image should be displayed and the desired object which was the result of cropping the object from Selected image should be placed on the Background Image and should be able to move as desired by the user
Actual Output	The selected background image is displayed and the desired object which was the result of cropping the object from Selected image is placed on the Background Image and is able to be moved as desired by the user

Test Case Name	Set the Background (Take Image using Camera)
Test Case No	8
Description	Testing Set the Background Feature for Take Image using Camera Feature in Cloning process
Testing Technique Used	Regression Testing
Preconditions	The user must have cropped the desired object which he needs to merge on the Background
Input Values	
Valid Inputs	Select an image that needs to be Background Image for the desired object to be placed on

Steps	First select the piClone android application installed in Android Operating System then choose the Clone Feature which is displayed on Feature Choice Menu then choose Take Image using Camera Feature displaying on Choose Image Menu then choose the desired object by cropping that object from the Image which is taken from Camera then Choose the Background Image which is the image on which you want to place the desired object
Expected Output	The selected background image should be displayed and the desired object which was the result of cropping the object from Selected image should be placed on the Background Image and should be able to move as desired by the user
Actual Output	The selected background image is displayed and the desired object which was the result of cropping the object from Selected image is placed on the Background Image and is able to be moved as desired by the user

Test Case Name	Image Blending (Select Image from Gallery)
Test Case No	9
Description	Testing Image Blending Feature for Select Image from Gallery Feature in Cloning process
Testing Technique Used	Unit Testing, Integration Testing and Regression Testing
Preconditions	The Desired Cropped Object must be placed on the selected background image at the desired coordinates
Input Values	
Valid Inputs	Select the Image Blending Button
Steps	First select the piClone android application installed in Android Operating System then choose the Clone Feature which is displayed on Feature Choice Menu then choose Select Image from Gallery Feature displaying on Choose Image Menu then choose the desired object by cropping that object from the Image which was selected from Gallery then Choose the Background Image which is the image on which you want to place the desired object then Click on the Image Blending Button to Complete the Cloning process
Expected Output	A Cloned Image should be the Output which should be displayed and stored in Gallery.
Actual Output	A Cloned Image is the Output which is displayed and stored in Gallery.

Test Case Name	Image Blending (Take Image using Camera)
Test Case No	10
Description	Testing Image Blending Feature for Take Image using Camera Feature in Cloning process
Testing Technique Used	Unit Testing, Integration Testing and Regression Testing
Preconditions	The Desired Cropped Object must be placed on the selected background image at the desired coordinates
Input Values	
Valid Inputs	Select the Image Blending Button
Steps	First select the piClone android application installed in Android Operating System then choose the Clone Feature which is displayed on Feature Choice Menu then choose Take Image using Camera Feature displaying on Choose Image Menu then choose the desired object by cropping that object from the Image which was taken using camera then Choose the Background Image which is the image on which you want to place the desired object then Click on the Image Blending Button to Complete the Cloning process
Expected Output	A Cloned Image should be the Output which should be displayed and stored in Gallery.
Actual Output	A Cloned Image is the Output which is displayed and stored in Gallery.

Test Case Name	Apply Filters Feature
Test Case No	11
Description	Testing the Apply Filters Feature
Testing Technique Used	Unit Testing
Preconditions	The Application must be opened displaying Feature Choice Menu
Input Values	Clone , Apply Filter, Edge detection
Valid Inputs	Choose Apply Filters Feature by clicking on the Apply Filters button
Steps	First select the piClone android application installed in Android Operating System, then choose the Apply Filters Feature which is displayed on Feature Choice Menu
Expected Output	The Filters Choice menu should open displaying choices for

	selecting the Desired Filter to apply
Actual Output	The Filters Choice menu is opened displaying choices for selecting the Desired Filter to apply

Test Case Name	Choosing Filters Feature
Test Case No	12
Description	Testing the Choosing Filters Feature
Testing Technique Used	Unit Testing, Integration Testing
Preconditions	The Application must be opened displaying Filters Choice Menu
Input Values	Any number of Filters
Valid Inputs	Choose any Filter Feature by clicking on that Filter button
Steps	First select the piClone android application installed in Android Operating System, then choose the Apply Filters Feature which is displayed on Feature Choice Menu then choose the Filter you want to use
Expected Output	The Image Choose Menu is opened, displaying options to Select image from Galley or Take new image using camera
Actual Output	The Image Choose Menu is opened, displaying options to Select image from Galley or Take new image using camera

Test Case Name	Select Image from Gallery (Apply Filter Feature)
Test Case No	13
Description	Testing Select Image from Gallery Feature for Apply Filter Feature
Testing Technique Used	
Preconditions	The user must have selected Apply Filter Feature from Feature Choice Menu
Input Values	Choose from Gallery, Take Image using Camera
Valid Inputs	Choose Select Image form Gallery option by clicking on the Choose from Gallery button
Steps	First select the piClone android application installed in Android Operating System then choose the Apply Filter Feature which is displayed on Feature Choice Menu then choose Select Image

	from Gallery displaying on Choose Image Menu
Expected Output	The Gallery of the android mobile phone should be accessed by piClone and should enable user to select desired image from Gallery
Actual Output	The Gallery of the android mobile phone is accessed by piClone enabling user to select desired image from Gallery
Status	Three

Test Case Name	Take Picture using Camera (Apply Filter Feature)
Test Case No	14
Description	Testing Take Picture using Camera Feature for Apply Filter Feature
Testing Technique Used	Unit Testing
Preconditions	The user must have selected Apply Filter Feature from Feature Choice Menu
Input Values	Choose from Gallery, Take Image using Camera
Valid Inputs	Choose Take Image using Camera option by clicking on the Take Image using Camera button
Steps	First select the piClone android application installed in Android Operating System then choose the Apply Filter Feature which is displayed on Feature Choice Menu then choose Take Image using Camera displaying on Choose Image Menu
Expected Output	The Camera of the android mobile phone should be accessed by piClone and should enable user to Take an Image
Actual Output	The Camera of the android mobile phone is accessed by piClone enabling user to Take an Image

Test Case Name	Filter
Test Case No	15
Description	Testing Filtered Feature for the Selected Image
Testing Technique Used	Unit Testing, Integration Testing
Preconditions	The user must have selected the image to which the user wants to apply Filter
Input Values	Filter
Valid Inputs	Choose Filter option by clicking on the Filter Button

Steps	First select the piClone android application installed in Android Operating System then choose the Apply Filter Feature which is displayed on Feature Choice Menu then Select the image you want to apply Filter on then Select the Filter Button to apply the selected Filer on the choosen Image
Expected Output	The new Filtered Image appears on the new interface and should be saved in Gallery of Android Mobile phone
Actual Output	The new Filtered Image appears on the new interface and is saved in Gallery of Android Mobile phone

Test Case Name	Edge Detection Feature
Test Case No	16
Description	Testing the Detect Edge Feature
Testing Technique Used	Unit Testing, Integration Testing
Preconditions	The Application must be opened displaying Feature Choice Menu
Input Values	Clone , Apply Filter, Edge detection
Valid Inputs	Choose Edge detection Feature by clicking on the Apply Filters button
Steps	First select the piClone android application installed in Android Operating System, then choose the Edge Detection Feature which is displayed on Feature Choice Menu
Expected Output	The Edge Detection Choice menu should open displaying choices for selecting the Desired Edge Detection Method to use
Actual Output	The Edge Detection Choice menu is opened displaying choices for selecting the Desired Edge Detection Method to use

Test Case Name	Choosing Edge Detection Method Feature
Test Case No	17
Description	Testing the Choosing Edge Detection Method Feature
Testing Technique Used	Unit Testing
Preconditions	The Application must be opened displaying Edge Detection Methods Choice Menu
Input Values	Any number of Edge Detection Methods
Valid Inputs	Choose any Edge Detection Method by clicking on that method button
Steps	First select the piClone android application installed in Android

	Operating System, then choose the Edge Detection Feature which is displayed on Feature Choice Menu then choose the Edge Detection Method you want to use
Expected Output	The Image Choose Menu is opened, displaying options to Select image from Galley or Take new image using camera
Actual Output	The Image Choose Menu is opened, displaying options to Select image from Galley or Take new image using camera

Test Case Name	Select Image from Gallery (Edge Detection Feature)
Test Case No	18
Description	Testing Select Image from Gallery Feature for Edge Detection Method Feature
Testing Technique Used	Unit Testing
Preconditions	The user must have selected Edge Detection Method from Feature Choice Menu
Input Values	Choose from Gallery, Take Image using Camera
Valid Inputs	Choose Select Image form Gallery option by clicking on the Choose from Gallery button
Steps	First select the piClone android application installed in Android Operating System then choose the Edge Detection Feature which is displayed on Feature Choice Menu then choose Edge Detection Method displaying on Choose Image Menu
Expected Output	The Gallery of the android mobile phone should be accessed by piClone and should enable user to select desired image from Gallery
Actual Output	The Gallery of the android mobile phone is accessed by piClone enabling user to select desired image from Gallery

Test Case Name	Take Picture using Camera (Feature)
Test Case No	19
Description	Testing Take Picture using Camera Feature for Edge Detection Feature
Testing Technique Used	Unit Testing
Preconditions	The user must have selected Edge Detection Method from Feature Choice Menu
Input Values	Choose from Gallery, Take Image using Camera

Valid Inputs	Choose Take Image using Camera option by clicking on the Take Image using Camera button
Steps	First select the piClone android application installed in Android Operating System then choose the Edge Detection Feature which is displayed on Feature Choice Menu then choose Take Image using Camera displaying on Choose Image Menu
Expected Output	The Camera of the android mobile phone should be accessed by piClone and should enable user to Take an Image
Actual Output	The Camera of the android mobile phone is accessed by piClone enabling user to Take an Image
Test Case Name	Detect Edges
Test Case No	20
Description	Testing Detect Edges Feature for the Selected Image
Testing Technique Used	Unit Testing, Integration Testing and Regression Testing
Preconditions	The user must have selected the image for which user wants to Detect Edges
Input Values	Detect Edges
Valid Inputs	Choose Detect Edges option by clicking on the Detect Edges Button
Steps	First select the piClone android application installed in Android Operating System then choose the Edge Detection Feature which is displayed on Feature Choice Menu then Select the image you want to Detect Edges for then Select the Detect Edges Button to apply the selected Edge Detection Method on the choosen Image
Expected Output	The new Image appears on the new interface and should be saved in Gallery of Android Mobile phone
Actual Output	The new Image appears on the new interface and is saved in Gallery of Android Mobile phone

5.8 Environmental Needs

Hardware

- Mobile with Android platform

Software

- Mobile Platform: Android 3.0/3.1 or later (Eclair Based on Linux Kernel 2.6.29 or later
- Eclipse 3.4 (Ganymede) or 3.5 (Galileo) with ADT Plugin

5.9 Responsibilities, Staffing and Training Needs

Responsibilities

- HaiderMushtaq is responsible for Acceptance Testing
- MianBasitMahmood is responsible for Integration Testing
- Hussain Ali is responsible for testing each separate unit that is Unit Testing.

Skills

- Skills needed to test piClone using QACenter

5.10Risks and contingencies

We have tried to test on various android platforms as much as possible, but it's impossible to test for all android platforms. What's more, mobile Android application is tested on Android devices and is tested on limited mobiles, thus we cannot predict the system behavior on the other mobile platforms (eg. iPhone, Blackberry, Symbian platform etc.). Further investigation is required to verify and improve piClone.

Chapter 6: Future Work

Researchers can use this Project as a guide to understand the “Image Cloning”. They can use this as a base for upgrading and adding new features. They can also use this for developing a new project by using this a reference material.The project developed could then be used as a basis for further work in the field of Image Processing.

It can be evolved into a bigger and more complex system with more features and functionality. The application can be enhanced to further include more features using which Images could be edited and modified in a more seamless and effective manner.

Anything dealing with Image Processing is highly unpredictable and tedious because there are infinite possibilities of the Images that are to be dealt with. As Image Cloning is part of Image Processing therefore there can be infinite number of scenarios where cloning using our system may not be as effective and as seamless as hoped. Therefore, Future work can include Perfecting Cloning for a wide range of possibilities.

Chapter 7: Conclusion

We workedto find a solution to Image Cloning, which includes Image Extraction from a selected Image and Merging of the extracted piece of selected image on the base Image, which can be implemented for an android application. It will be our algorithm that will be able to merge (clone) these 2 images seamless and effective manner .We developed an algorithm that can merge 2 images in such a way that the output image is as realistic as possible. An android application, using which the user could easily perform cloning and edit and modify images in a seamless and effective manner, was the final product.

Using the generic framework of guided interpolation, we have introduced a variety of tools to edit in a seamless and effortless manner the contents of an image selection. The extent of possible changes ranges from replacement by, or mixing with, another source image region, to alterations of some aspects of the originalimage inside the selection, such as texture, illumination, or color. An important common characteristic of all these tools is that there isnoneedforpreciseobjectdelineation, incontrastwiththeclassic tools that address similar tasks. This is a valuable feature, whether one is interested in small touch-up operations or in complex photomontages. It is for instance possible to insert an object while flattening its texture to make it match the style of a texture-free destination. Finally, it is worth noting that the range of editing facilities derived in this paper from the same generic framework could probably be extended further. Appearance changes could for instance alsodealwiththesharpnessofobjectsofinterest, thusallowingthe user to make apparent changes of focus.

The purpose of this app is to give the end user easy access to various image editing and image extraction techniques with the help of an easy to use graphic user interface and minimal learning effort. Using this app the end user can create amazing breathtaking pictures which can then be saved to local storage or can then be shared among friends using the social media

Bibliography

http://www.cs.huji.ac.il/~danix/mvclone/files/mvc-final-opt.pdf

http://www.connellybarnes.com/work/class/2013/cs6501/proj2/

http://www.developer.android.com/reference/android/app/Activity.html

http://www.embedded.com/design/programming-languages-and-tools/4406164/Developing-OpenCV-computer-vision-apps-for-the-Android-platform

http://www.eavise.be/mastertheses/BilliauwsBonjean.pdf

Appendix

Description:

An android application that takes in two pictures from the user. One picture can be selected as the base picture or as the background. From the other picture, the selected object will be cloned on to the base picture. Gradients can be modified, boundary conditions can be applied, and then everything would be "integrated" using a linear solver to go back to color domain of the base image.

ScopeofWork:

This is an image processing application for the android phone that can be used to take pictures of Objects and then strip away the background, replacing it with a new one. An additional feature will be to modify images by applying filters and edge detection techniques.

AcademicObjective:

Understand the image cloning process. Working with images on Android and then developing a complete android based application

End Goal Objective:

Develop a simple, easy to use Android application that would require minimum user input to produce beautiful image mergers.

Pseudo code for components

HomeActivity.java

```
public class HomeActivity extends Activity {

public static final int MEAN_BLUR = 1;
public static final int GAUSSIAN_BLUR = 2;
public static final int MEDIAN_BLUR =3 , SHARPEN =4 ,DILATE = 5, ERODE = 6, THRESHOLD =7 ,
        ADAPTIVE_THRESHOLD =8, DIFFERENCE_OF_GAUSSIAN = 9 ,CANNY = 10,
        SOBEL = 11, HARRIS = 12, HOUGHL = 13 , HOUGHC = 14, CONTOURS = 15;

   Button bMean, gBlur, meBlur, shr, dil, ero, ada, DoG, TCED, Sobel, Harris, HoughL, HoughC, Contours,
Threashold;


   @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_home);
bMean = (Button)findViewById(R.id.bMean);
gBlur = (Button)findViewById(R.id.gBlur);
meBlur = (Button)findViewById(R.id.meBlur);
shr = (Button)findViewById(R.id.shr);
dil = (Button)findViewById(R.id.dil);
ero = (Button)findViewById(R.id.ero);
ada = (Button)findViewById(R.id.ada);
DoG = (Button)findViewById(R.id.DoG);
      TCED = (Button)findViewById(R.id.TCED);
Sobel = (Button)findViewById(R.id.Sobel);
      Harris = (Button)findViewById(R.id.Harris);
HoughL = (Button)findViewById(R.id.HoughL);
```

```
HoughC = (Button)findViewById(R.id.HoughC);
      Contours = (Button)findViewById(R.id.Contours);
Threashold = (Button) findViewById(R.id.Threashold);

Threashold.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
           Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", THRESHOLD);
startActivity(i);
        }
      });

Contours.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
           Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", CONTOURS);
startActivity(i);
        }
      });

HoughC.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
           Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", HOUGHC);
startActivity(i);
        }
```

```
    });

HoughL.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", HOUGHL);
startActivity(i);
        }
    });

Harris.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", HARRIS);
startActivity(i);
        }
    });

Sobel.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", SOBEL);
startActivity(i);
        }
    });
```

```
TCED.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", CANNY);
startActivity(i);
        }
      });

DoG.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", DIFFERENCE_OF_GAUSSIAN);
startActivity(i);
        }
      });

ada.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", ADAPTIVE_THRESHOLD);
startActivity(i);
        }
      });

ero.setOnClickListener(new View.OnClickListener() {
        @Override
```

```
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", ERODE);
startActivity(i);
        }
      });

shr.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", SHARPEN);
startActivity(i);
        }
      });

dil.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", DILATE);
startActivity(i);
        }
      });

bMean.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
```

```
MainActivity.class);
i.putExtra("ACTION_MODE", MEAN_BLUR);
startActivity(i);
        }
      });

gBlur.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", GAUSSIAN_BLUR);
startActivity(i);
        }
      });

meBlur.setOnClickListener(new View.OnClickListener() {
        @Override
public void onClick(View v) {
            Intent i = new Intent(getApplicationContext(),
MainActivity.class);
i.putExtra("ACTION_MODE", MEDIAN_BLUR);
startActivity(i);
        }
      });

   }

}
```

MainActivity.java

```
public class MainActivity extends Activity {

private final int SELECT_PHOTO = 1;
privateImageViewivImage, ivImageProcessed;
   Mat src;
staticint ACTION_MODE = 0;



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

ivImage = (ImageView)findViewById(R.id.ivImage);
ivImageProcessed =
            (ImageView)findViewById(R.id.ivImageProcessed);
      Intent intent = getIntent();
if(intent.hasExtra("ACTION_MODE")) {
         ACTION_MODE = intent.getIntExtra("ACTION_MODE", 0);
      }


   }

   @Override
publicbooleanonCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
   }
```

```
   @Override
publicbooleanonOptionsItemSelected(MenuItem item) {
int id = item.getItemId();
if (id == R.id.action_load_image) {
        Intent photoPickerIntent = new
Intent(Intent.ACTION_PICK);
photoPickerIntent.setType("image/*");
startActivityForResult(photoPickerIntent, SELECT_PHOTO);
Log.d("Haider", "Activity: Started ");
return true;
      }
returnsuper.onOptionsItemSelected(item);
   }

public void onActivityResult(intrequestCode, intresultCode, Intent imageReturnedIntent){
super.onActivityResult(requestCode, resultCode, imageReturnedIntent);
Log.d("Haider", "OnActivityResult: Started ");
switch(requestCode) {
case SELECT_PHOTO:
if(resultCode == RESULT_OK){
try {
//Code to load image into a Bitmap and convert it to a Mat for processing.

final Uri imageUri = imageReturnedIntent.getData();
finalInputStreamimageStream =
getContentResolver().openInputStream(imageUri);
final Bitmap selectedImage =
BitmapFactory.decodeStream(imageStream);
src = new Mat(selectedImage.getHeight(),
selectedImage.getWidth(), CvType.CV_8UC4);
Utils.bitmapToMat(selectedImage, src);
switch (ACTION_MODE){
```

```
//Add different cases here depending on the required operation
caseHomeActivity.MEAN_BLUR:
Imgproc.blur(src, src, new Size(3, 3));
break;
caseHomeActivity.GAUSSIAN_BLUR:
Imgproc.GaussianBlur(src, src, new Size(3, 3), 0);
break;
caseHomeActivity.MEDIAN_BLUR:
Imgproc.medianBlur(src, src, 3);
break;
caseHomeActivity.SHARPEN:
                Mat kernel = new Mat(3,3,CvType.CV_16SC1);
kernel.put(0, 0, 0, -1, 0, -1, 5, -1, 0, -1, 0);
Imgproc.filter2D(src, src, src.depth(), kernel);
break;
caseHomeActivity.DILATE:
                Mat kernelDilate = Imgproc.getStructuringElement(
Imgproc.MORPH_RECT, new Size(3, 3));
Imgproc.dilate(src, src, kernelDilate);
break;
caseHomeActivity.ERODE:
                Mat kernelErode = Imgproc.getStructuringElement(Imgproc.MORPH_ELLIPSE, new Size(5,
5));
Imgproc.erode(src, src, kernelErode);
break;
caseHomeActivity.THRESHOLD:
Imgproc.threshold(src, src, 100, 255, Imgproc.THRESH_BINARY);
break;
caseHomeActivity.ADAPTIVE_THRESHOLD:
Imgproc.cvtColor(src, src, Imgproc.COLOR_BGR2GRAY);
Imgproc.adaptiveThreshold(src, src, 255, Imgproc.ADAPTIVE_THRESH_GAUSSIAN_C,
Imgproc.THRESH_BINARY, 3, 0);
break;
```

```
caseHomeActivity.DIFFERENCE_OF_GAUSSIAN:
src = DifferenceOfGaussian(src);
break;
caseHomeActivity.CANNY:
Imgproc.cvtColor(src,src,Imgproc.COLOR_BGR2GRAY);
Imgproc.Canny(src, src, 10, 100);
break;
caseHomeActivity.SOBEL:
src = Sobel(src);
break;
caseHomeActivity.HARRIS:
src = HarrisCorner(src);
break;
caseHomeActivity.HOUGHL:
src = HoughLines(src);
break;
caseHomeActivity.HOUGHC:
src = HoughCircles(src);
break;
caseHomeActivity.CONTOURS:
src = Contours(src);
break;

                }
//Code to convert Mat to Bitmap to load in an ImageView. Also load original image in imageView
                Bitmap processedImage = Bitmap.createBitmap(src.cols(),
src.rows(), Bitmap.Config.ARGB_8888);
Utils.matToBitmap(src, processedImage);
ivImage.setImageBitmap(selectedImage);
ivImageProcessed.setImageBitmap(processedImage);

            } catch (FileNotFoundException e) {
```

```
e.printStackTrace();
                }
            }
break;
      }
   }

privateBaseLoaderCallbackmOpenCVCallBack = new
BaseLoaderCallback(this) {
            @Override
public void onManagerConnected(int status) {
switch (status) {
caseLoaderCallbackInterface.SUCCESS:
//DO YOUR WORK/STUFF HERE
break;
default:
super.onManagerConnected(status);
break;
                }
            }
         };

   @Override
protected void onResume() {
super.onResume();
OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_0_0,
this,
mOpenCVCallBack);
   }

public Mat DifferenceOfGaussian(Mat originalMat)
   {
```

```
        Mat grayMat = new Mat();
        Mat blur1 = new Mat();
        Mat blur2 = new Mat();
//Converting the image to grayscale
Imgproc.cvtColor(originalMat
                , grayMat, Imgproc.COLOR_BGR2GRAY);
//Bluring the images using two different blurring radius
Imgproc.GaussianBlur(grayMat, blur1, new Size(15, 15), 5);
Imgproc.GaussianBlur(grayMat, blur2, new Size(21, 21), 5);
//Subtracting the two blurred images
        Mat DoG = new Mat();
Core.absdiff(blur1, blur2, DoG);
//Inverse Binary Thresholding
Core.multiply(DoG, new Scalar(100), DoG);
Imgproc.threshold(DoG, DoG, 50, 255
                , Imgproc.THRESH_BINARY_INV);
//Converting Mat back to Bitmap
returnDoG;
    }

public Mat Sobel( Mat originalMat )
    {
        Mat grayMat = new Mat();
        Mat sobel = new Mat(); //Mat to store the result
//Mat to store gradient and absolute gradient respectively
        Mat grad_x = new Mat();
        Mat abs_grad_x = new Mat();
        Mat grad_y = new Mat();
        Mat abs_grad_y = new Mat();
        List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
//Converting the image to grayscale
Imgproc.cvtColor(originalMat, grayMat, Imgproc.COLOR_BGR2GRAY);
```

```
//Calculating gradient in horizontal direction
Imgproc.Sobel(grayMat, grad_x, CvType.CV_16S, 1, 0, 3, 1, 0);
//Calculating gradient in vertical direction
Imgproc.Sobel(grayMat, grad_y, CvType.CV_16S, 0, 1, 3, 1, 0);
//Calculating absolute value of gradients in both the direction
Core.convertScaleAbs(grad_x, abs_grad_x);
Core.convertScaleAbs(grad_y, abs_grad_y);
//Calculating the resultant gradient
Core.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 1, sobel);
//Find contours :



Imgproc.threshold(sobel, sobel, 200, 255, Imgproc.THRESH_BINARY);

      Mat copy = sobel.clone();

Imgproc.findContours(copy, contours, new Mat(), Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE );

// Approximate contours to polygons + get bounding rects
      MatOfPoint2f approxCurve = new MatOfPoint2f();

for (int i=0; i<contours.size(); i++)
      {
         //Convert contours(i) from MatOfPoint to MatOfPoint2f
         MatOfPoint2f contour2f = new MatOfPoint2f(contours.get(i).toArray() );
         //Processing on mMOP2f1 which is in type MatOfPoint2f
doubleapproxDistance = Imgproc.arcLength(contour2f, true)*0.02;
Imgproc.approxPolyDP(contour2f, approxCurve, approxDistance, true);

         //Convert back to MatOfPoint
MatOfPoint points = new MatOfPoint(approxCurve.toArray() );
```

```
        // Get bounding rect of contour
Rectrect = Imgproc.boundingRect(points);

        // draw enclosing rectangle (all same color, but you could use variable i to make them unique)
Imgproc.rectangle(sobel, new Point(rect.x,rect.y), new Point(rect.x+rect.width,rect.y+rect.height), new Scalar(255, 0, 0),1, 8,0);
      }

      Toast toast = Toast.makeText(this, "Total Contours: " + contours.size(), Toast.LENGTH_LONG);
toast.show();



returnsobel;
   }

public Mat HarrisCorner(Mat originalMat) {
      Mat grayMat = new Mat();
      Mat corners = new Mat();
//Converting the image to grayscale
Imgproc.cvtColor(originalMat, grayMat, Imgproc.COLOR_BGR2GRAY);
      Mat tempDst = new Mat();
//finding corners
Imgproc.cornerHarris(grayMat, tempDst, 2, 3, 0.04);
//Normalizing harris corner's output
      Mat tempDstNorm = new Mat();
Core.normalize(tempDst, tempDstNorm, 0, 255, Core.NORM_MINMAX);
Core.convertScaleAbs(tempDstNorm, corners);
//Drawing corners on a new image
      Random r = new Random();
for (int i = 0; i <tempDstNorm.cols(); i++) {
for (int j = 0; j <tempDstNorm.rows(); j++) {
double[] value = tempDstNorm.get(j, i);
```

```
if (value[0] > 150)
Imgproc.circle(corners, new Point(i, j), 5, new Scalar(r.nextInt(255)), 2);
        }
      }
//Converting Mat back to Bitmap
return corners;
  }

public Mat HoughLines(Mat originalMat)
  {
     Mat grayMat = new Mat();
     Mat cannyEdges = new Mat();
     Mat lines = new Mat();
//Converting the image to grayscale
Imgproc.cvtColor(originalMat, grayMat,Imgproc.COLOR_BGR2GRAY);
Imgproc.Canny(grayMat, cannyEdges,10, 100);
Imgproc.HoughLinesP(cannyEdges, lines, 1, Math.PI / 180, 50, 20, 20);
     Mat houghLines = new Mat();
houghLines.create(cannyEdges.rows(), cannyEdges.cols(), CvType.CV_8UC1);
     //Drawing lines on the image
for(int i = 0 ; i <lines.cols() ; i++)
     {
double[] points = lines.get(0,i);
double x1, y1, x2, y2;
x1 = points[0];
        y1 = points[1];
x2 = points[2];
        y2 = points[3];
        Point pt1 = new Point(x1, y1);
        Point pt2 = new Point(x2, y2);
//Drawing lines on an image
Imgproc.line(houghLines, pt1, pt2, new Scalar(255, 0, 0), 1);
```

```
    }
//Converting Mat back to Bitmap
returnhoughLines;
  }

public Mat HoughCircles(Mat originalMat)
  {
    Mat grayMat = new Mat();
    Mat cannyEdges = new Mat();
    Mat circles = new Mat();
//Converting the image to grayscale
Imgproc.cvtColor(originalMat, grayMat,Imgproc.COLOR_BGR2GRAY);
Imgproc.Canny(grayMat, cannyEdges,10, 100);
Imgproc.HoughCircles(cannyEdges, circles,
Imgproc.CV_HOUGH_GRADIENT, 1, cannyEdges.rows() / 15);
//, grayMat.rows() / 8);
    Mat houghCircles = new Mat();
houghCircles.create(cannyEdges.rows(),cannyEdges.cols(), CvType.CV_8UC1);
//Drawing lines on the image
for(int i = 0 ; i <circles.cols() ; i++)
    {
double[] parameters = circles.get(0,i);
double x, y;
int r;
      x = parameters[0];
      y = parameters[1];
      r = (int)parameters[2];
      Point center = new Point(x, y);
//Drawing circles on an image
Imgproc.circle(houghCircles, center, r,
new Scalar(255, 0, 0), 1);
    }
```

```
//Converting Mat back to Bitmap
returnhoughCircles;
   }

public Mat Contours(Mat originalMat)
   {
      Mat grayMat = new Mat();
      Mat cannyEdges = new Mat();
      Mat hierarchy = new Mat();
      List<MatOfPoint>contourList = new ArrayList<MatOfPoint>();
//A list to store all the contours
      //Converting the image to grayscale
Imgproc.cvtColor(originalMat,grayMat,Imgproc.COLOR_BGR2GRAY);
Imgproc.Canny(grayMat, cannyEdges,10, 100);
//finding contours
Imgproc.findContours(cannyEdges,contourList, hierarchy,Imgproc.RETR_LIST,
Imgproc.CHAIN_APPROX_SIMPLE);
//Drawing contours on a new image
      Mat contours = new Mat();
contours.create(cannyEdges.rows(), cannyEdges.cols(),CvType.CV_8UC3);
      Random r = new Random();
for(int i = 0; i <contourList.size(); i++)
      {
Imgproc.drawContours(contours, contourList, i, new Scalar(r.nextInt(255), r.nextInt(255),r.nextInt(255)), -1);
      }
//Converting Mat back to Bitmap
return contours;
   }       }
```