Noman Khalid
01-235161-080

Image Forgery Detection

Bachelor of Science in Information Technology

Supervisor: Dr. Samabia Tehseen

Department of Computer Science
Bahria University, Islamabad

December 2019

Certificate

We accept the work contained in the report titled "Image Forgery Detection", written by Noman Khalid as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Information Technology.

Approved by . . . :

Supervisor: Dr. Samabia Tehseen (Associate Professor)

Internal Examiner: Dr. Arif ur Rahman (Associate Professor)

External Examiner: Dr. Anwar Ghani (Associate Professor)

Project Coordinator: Dr. Muneeb Gohar (Associate Professor)

Head of the Department: Dr. Muhammad Muzammal (Sr. Associate Professor)

December 13th, 2019

Abstract

Technological development in digital world has led to a huge increase in the popularity of digital images in all domains of life. However, sophisticated and easy to use photo editing software tools have made manipulation of images very easy. Thus there is a need to authenticate images especially in legal matters. The field of image authentication and forgery detection has gained huge popularity lately. A key domain in this regard is copy-move forgery detection. Copy move forgery involves copying a portion of an image and pasting it to a different location in same image, with a purpose to conceal facts. The main objective of this project is to design and develop a desktop application to detect image forgery in digital images.

Acknowledgments

First of all I would like to thank Al-Mighty Allah, the most beneficial and the most merciful. He gave me the strength to complete this major milestone of my degree program. I am also very thankful to my teachers, parents and friends who supported me both technically and morally at every stage of this project. I would like to express my gratitude to my supervisor Dr. Samabia Tehseen. Her technical guidance and support helped me complete this project and achieve its objective.

NOMAN KHALID
Islamabad, Pakistan

December 2019

*“Everybody should learn to program a computer,
because it teaches you how to think.”*

Steve Jobs

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

CMFD	Copy Move Forgery Detection
DCT	Discrete Cosine Transform
FN	False Negative
FP	False Positive
GUI	Graphical User Interface
IDE	Integrated Development Environment
RAD	Rapid Application Development
TN	True Negative
TP	True Positive

Chapter 1

Introduction

In the age of social media journalism, the importance of identification rather an image is legit or tampered has increased. Due to state-of-the-art technologies available for image editing such as Photoshop which makes it very easy to make image forgeries and it is getting harder to identify that an image is forged or not forged, and these images are one of the main sources of fake news which cause chaos in society. To define the integrity and authenticity of an image, a lot of research is done in the last few years to identify image forgery [1] [2]. Before any action must be taken on these fake images verifying the authenticity of these fake images is necessary.

There are two types of techniques to find out the image forgery. Active techniques and Passive (Blind) techniques [3]. In the Active method, there are two ways to find image forgery such as Digital Signature and Digital Watermarking both techniques depend on the information embedded at the time of image creation hence this method is not useful when dealing with images from unknown sources which will limit its applications. In the Passive method, no prior information of the image is required. There are many techniques each can detect special forgery in its own way. One of which is copy-move forgery detection. In copy-move forgery, a part of an image is copied and placed in another location within the same image to duplicate something of importance or to hide some information behind it that does not need to be displayed. You can see an example in Fig. 1.

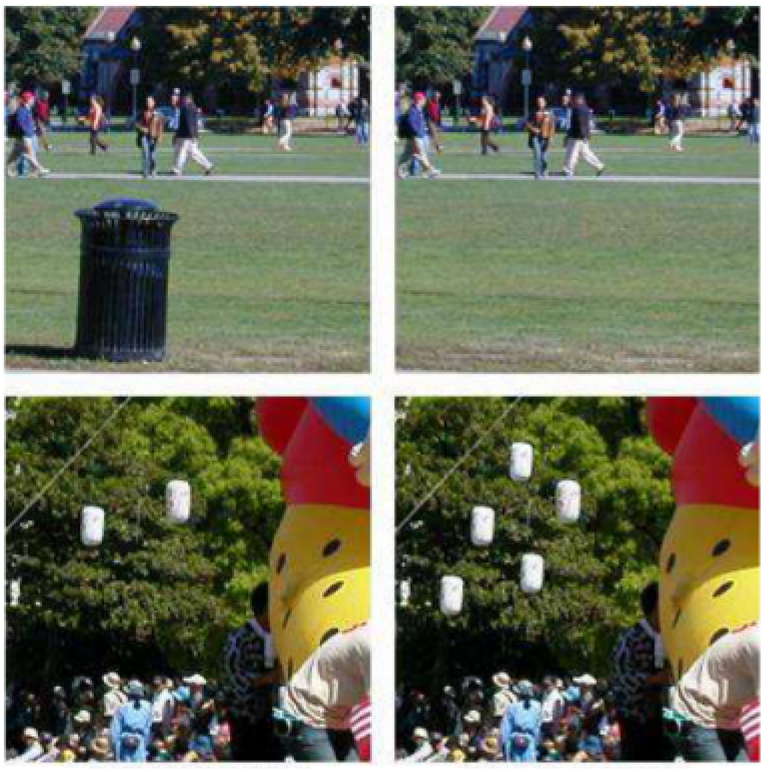


Figure 1.1: Copy-Move Forgery. Original (left), Forged (right)

To make it look more authentic often the image forger performs a series of post-processing after the manipulation of an image. Some examples of this post-processing are sharpening, blurring, JPEG compression and noise addition.

The purpose of this project is to determine whether a digital image has been tampered or not.

1.1 Problem Description

The purpose behind this project is to create an application which will use passive (blind) forensic technique such as copy-move forgery detection to determine whether an image has been tempered or not.

1.2 Project Objective

The main objective of this project is to design and develop a desktop application to detect image tampering in a digital image.

1.3 Project Scope

The primary objectives of developing this system are: The main scope of this application is to determine whether a digital image has been tampered using Passive forensic techniques such as copy-move forgery detection.

This project will not focus on Active forensic techniques.

1.4 Benefits

This project has real value for the following industries and domains which can benefit from it:

- Online Lenders and Banking Institutions
- Insurance Companies
- Social Media Companies
- Government and Local Authorities
- Recruitment Companies

This project can help the above domains in the image recognition for its integrity and authenticity.

Chapter 2

Literature Review

In this chapter all the concepts, methods and theories that are related with copy-move forgery detection and other applications will be discussed.

2.1 Related Works

Copied regions can be detected by using two techniques such as block-based and key point-based detection. The difference between these two techniques is that key point-based methods extract feature points only on particular regions from an image without any subdivisions of an image, therefore, it has few computational steps but cannot produce highly accurate results. The block-based methods subdivide the image into rectangular regions. A feature vector is computed for every region and similar feature vectors are subsequently matched for forgery detection, therefore, it has more computational steps and gives highly accurate results. Block-based techniques will be discussed in this chapter.

2.1.1 Exhaustive Search Technique

It is an easy-going approach to detecting copy-move forgery. A digital image is a representation of a real image as a set of numbers called pixels. Pixels can be stored and handled by a digital computer. For each pixel, the imaging device records a number that describes some property of this pixel such as the intensity of light or its color. The idea is to match each pixel value with other pixel values, starting from the top left corner of the image to the bottom right corner and mark the duplicated pixel [4]. Exhaustive search uses circularly shifted versions of a forged image to match with other parts. It reduces the computational complexity as a pixel value is matched twice with other pixel values, so half of the comparisons are reduced.

2.1.2 Block Match Technique

A good technique for detecting copy-move forgery is to verify if a set of blocks of pixels in a region of the image matches with another in a different region of the image. That is, the image is divided into n non-overlapping blocks, and each block is compared with the remaining ones. But, selecting the size of the block is difficult. If the size of the block is larger than the forged area, an exact match of the blocks does not result. If the size of the block is smaller, the forged area may cross the boundaries of adjacent blocks and then also exact matches would not result. If the size of the block size is made very small, the matching process becomes computationally intensive, particularly with large images. Also, uniform areas in the original image will be shown as duplicates. This kind of block matching can be termed as non-overlapped block matching.

2.1.3 Exact Match Technique

A better alternative is to select overlapping blocks. Blocks of size b x b pixels are selected from the top-left corner, moving right and down, to the bottom-right corner one pixel at a time along with the image. For each block, the pixel values are extracted by columns into a row of a two-dimensional array A with (M–b+1) (N–b+1) rows and b x b columns. Two identical rows in the matrix A correspond to two identical b x b blocks. To recognize the identical rows easily and quickly, the rows of the matrix A are lexicographically sorted. Matching rows can be easily searched by going through the rows of the ordered matrix A and looking for two successive rows that are identical.

2.1.4 Robust Match Technique

The best alternative to detect copy-move forgery is Robust Match where instead of matching the pixel representation of blocks, their robust representations are matched. One of the robust representations is the quantized DCT (Discrete Cosine Transform) coefficients. The advantage of DCT is that when it is applied on an image block it will give us quantized DCT coefficients and the signal energy is concentrated on first few coefficients of this image block whereas most other coefficients are negligibly small. Therefore the changes that occur in high frequencies due to the operations such as compression, retouching and noise addition do not affect these first few coefficients greatly.

2.2 Conclusion

Every technique has its own pros and cons, but performance evaluation of specific technique is based on the parameters like the size of a cloned region in pixels, type of transformation applied on image and compression ratio of an input image. Therefore, none of the techniques has a robust detection way of the cloned region. In this system modified robust match technique will be used to save as much execution time as possible while giving satisfactory results [5]. In this technique, the quantized coefficients of DCT of blocks are used as block features. Here the length of the feature of the blocks is reduced which reduces the execution time without affecting the quality of the result. Reduction in the length of the feature vector is possible, because, when the quantization of DCT coefficients of the blocks is done, there are many long run zeros. These are high-frequency DCT coefficients, which do not contribute to the quality of the image and therefore they can be omitted. The quantized DCT coefficients are read in zigzag order so only quantized low-frequency coefficients are taken. This algorithm is as efficient as a robust match while detecting a forgery in an image, but it will save 25 percent of execution time.

Chapter 3

Requirement Specifications

3.1 Existing System

There are some existing image forgery detection systems available on the internet, but all the tools ask you to upload the image online and it processes the operation on an image online and provides the resultant output to the user. These tools are very useful but sometime the user does not have the internet connection to detect the forgery in an image online or the user does not want to share the images online because of privacy issues.

3.2 Proposed System

The proposed system is going to be a desktop application that can run on windows operating system. The proposed system will detect the forgery in an image and display the results to the user. After starting the application, the user will select an image from the hard drive. The detection engine running in the background will detect the forged part in an image and display the result on screen in the form of masked image.

3.3 Requirements Specification

Requirement specifications will provide both the functional and non-functional requirements of the application.

3.4 Functional Requirements

The functional requirements of the application are as follows.

- User should be able to run the application.
- User should be able to open an image.
- System should be able to display masked image after performing operation on forged image.
- System should be able to save the created masked image in order to avoid loss.
- User should be able to exit the application.

3.5 Non-Functional Requirements

3.5.1 Usability

The system is very user friendly which will allow its users to access every functionality of system from single screen.

3.5.2 Privacy

The information of user will not be shared with anyone in public under any circumstances.

3.5.3 Reliability

The system will be reliable enough to allow user to access information without any errors.

3.5.4 Security

The system will keep the data of user secure from unauthorized access.

3.5.5 Availability

The system will be running at all time so that user can access information anytime.

3.5.6 Space Requirement

The system will manage hardware resources efficiently.

3.6 Use Cases

Use cases of the system are described in this section. Figure 3.1 shows the high level use case diagram of the whole system with actors as well as functionalities.

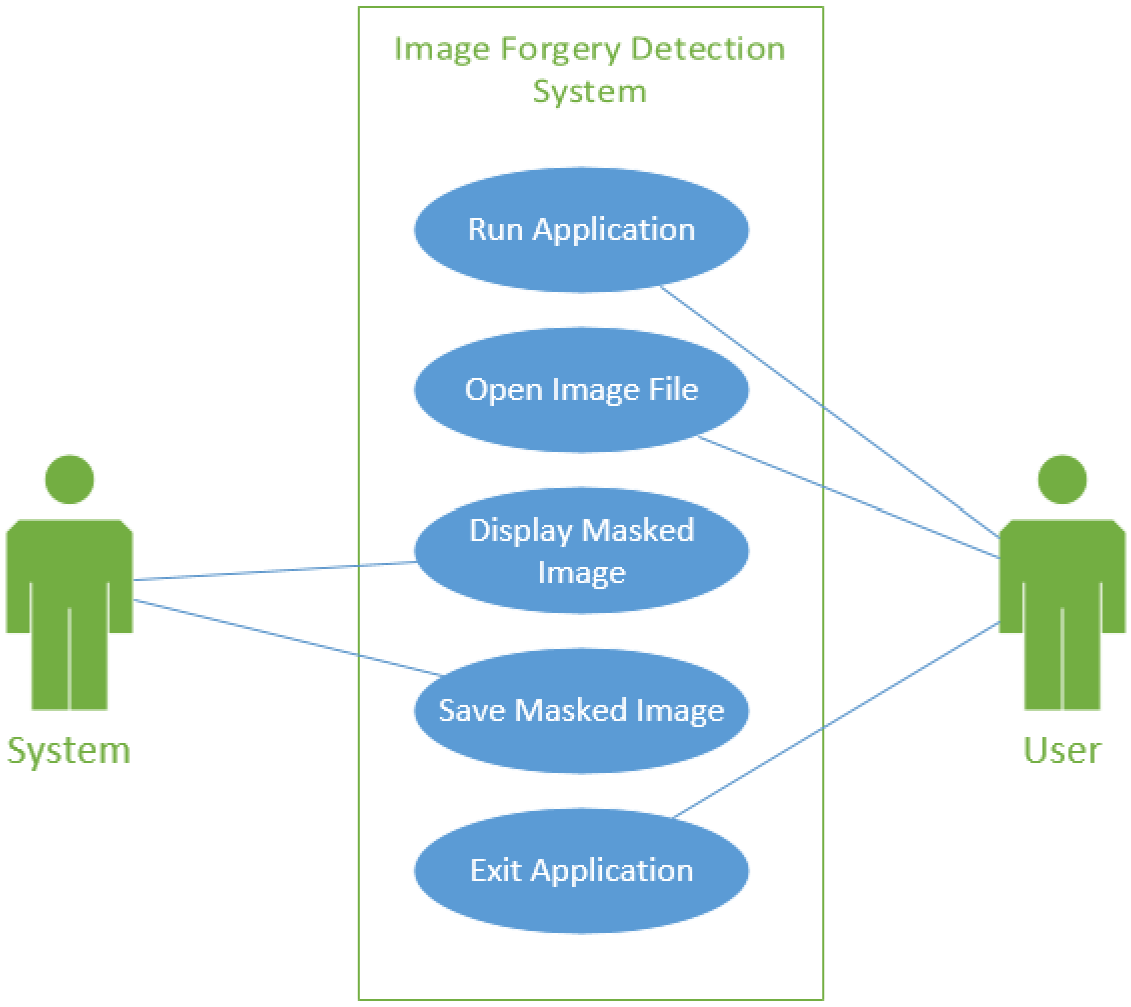


Figure 3.1: Use Case Diagram

3.6.1 Use Case: Run Application

Use Case ID	**CMFD-01**
TITLE	Run Application
PRIMARY ACTOR	User
DESCRIPTION	User may want to run the application from operating system
PRE-CONDITION	The application must be installed on operating system
POST-CONDITION	The application must be compatible with operating system
BASIC FLOW	1)User started the application by clicking the application icon 2)Application is ready to be used
ALTERNATIVE FLOW	1)Application stops responding 2)Application crashed
STIMULUS	User clicked the application icon from the operating system

Table 3.1: Use Case 1: Run Application

3.6.2 Use Case: Open Image File

Use Case ID	**CMFD-02**
TITLE	Open Image File
PRIMARY ACTOR	User
DESCRIPTION	To start forgery detection process user must open an image
PRE-CONDITION	Application must be installed on the device
POST-CONDITION	Image is opened in image block
BASIC FLOW	1)The application must be installed on device 2)User opens the application 3)User is prompted to open an image 4)User select desired option and proceeds
ALTERNATIVE FLOW	User exits the application without opening a file to proceed with
STIMULUS	User clicking on load image button
RESPONSE	A new or an existing file is either opened or application is exited

Table 3.2: Use Case 2: Open Image File

3.6.3 Use Case: Display Masked Image

Use Case ID	**CMFD-03**
TITLE	Display Masked Image
PRIMARY ACTOR	System
DESCRIPTION	Whatever the system has predicted in an image must be displayed on the screen
PRE-CONDITION	An image is created by the by system
POST-CONDITION	An image is displayed in image block
BASIC FLOW	1)System create the image 2)System displays the image in image block
ALTERNATIVE FLOW	If desired image is not created than show an error message
STIMULUS	Creation of an image by system
RESPONSE	Display masked image or error message

Table 3.3: Use Case 3: Display Masked Image

3.6.4 Use Case: Save Masked Image

Use Case ID	**CMFD-04**
TITLE	Save Masked Image
PRIMARY ACTOR	System
DESCRIPTION	After performing, operations on image system must save image in order to avoid the loss
PRE-CONDITION	System have made changes on an Image by performing operation on it
POST-CONDITION	Image is saved
BASIC FLOW	1)System give save command after doing some manipulation 2)Data or file is saved in device designated directory
ALTERNATIVE FLOW	Image is unable to be save due to some error
STIMULUS	Save command by system
RESPONSE	Image is either saved or it is not saved

Table 3.4: Use Case 4: Save Masked Image

3.6.5 Use Case: Exit Application

Use Case ID	**CMFD-05**
TITLE	Exit Application
PRIMARY ACTOR	User
DESCRIPTION	User may want to exit application; this must be done in a processed manner
PRE-CONDITION	Application is opened and running
POST-CONDITION	User exits the system or application
BASIC FLOW	1)The user started the application. 2)User may or may not have used the application 3)User clicks on exit button or give exit command with keys 4)User exit the application or system
ALTERNATIVE FLOW	1)User has chosen not to exit application 2)User may return to the application without closing it on exit application prompt 3)Application stops responding
STIMULUS	User clicked on exit application button
RESPONSE	User clicked on exit application button

Table 3.5: Use Case 5: Exit Application

Chapter 4

Design

In this chapter, system design, architecture, modules, interfaces and data for a system will be discussed.

4.1 System Architecture

Two-tier architecture will be used to build this application. The first tier is the system with which the user interacts through a graphical user interface. The second tier is the logical layer which handles processing and errors.

4.1.1 Presentation Layer

This layer is the interface of the application, with which the user is going to interact with the application.

4.1.2 Logical Layer

This layer handle all the functionalities and logical working of system such as it performs all the operations on data which is an image in our system and returns the result back to presentation layer so user can view the results. It also handles overall behavior of application such as responds to events that user initiate from presentation layer as well as loading, processing, displaying and saving of data which are images in this system.

4.2 Design Methodology

To design this application RAD (Rapid Application Development) methodology is used because the goal of this project is to produce a working version of the application as quickly as possible and improve if further in iterations after that. The application gets more refined and better as each iteration is completed. The early version of application is very rough but give a picture of what can be built. Each continuous iteration then looks more like the finished product.

4.3 Low Level Design

4.3.1 Sequence Diagram

Sequence Diagram defines exact flow of system with respect to time. It shows interaction between different classes of system and shows dynamic behavior of system.

4.3.1.1 Main Sequence Diagram

It shows all main actions that must be done to detect forgery in an image. It include all necessary actions like opening application, loading an image, detecting forgery, saving masked image, displaying forged image, showing forgery results and closing application.

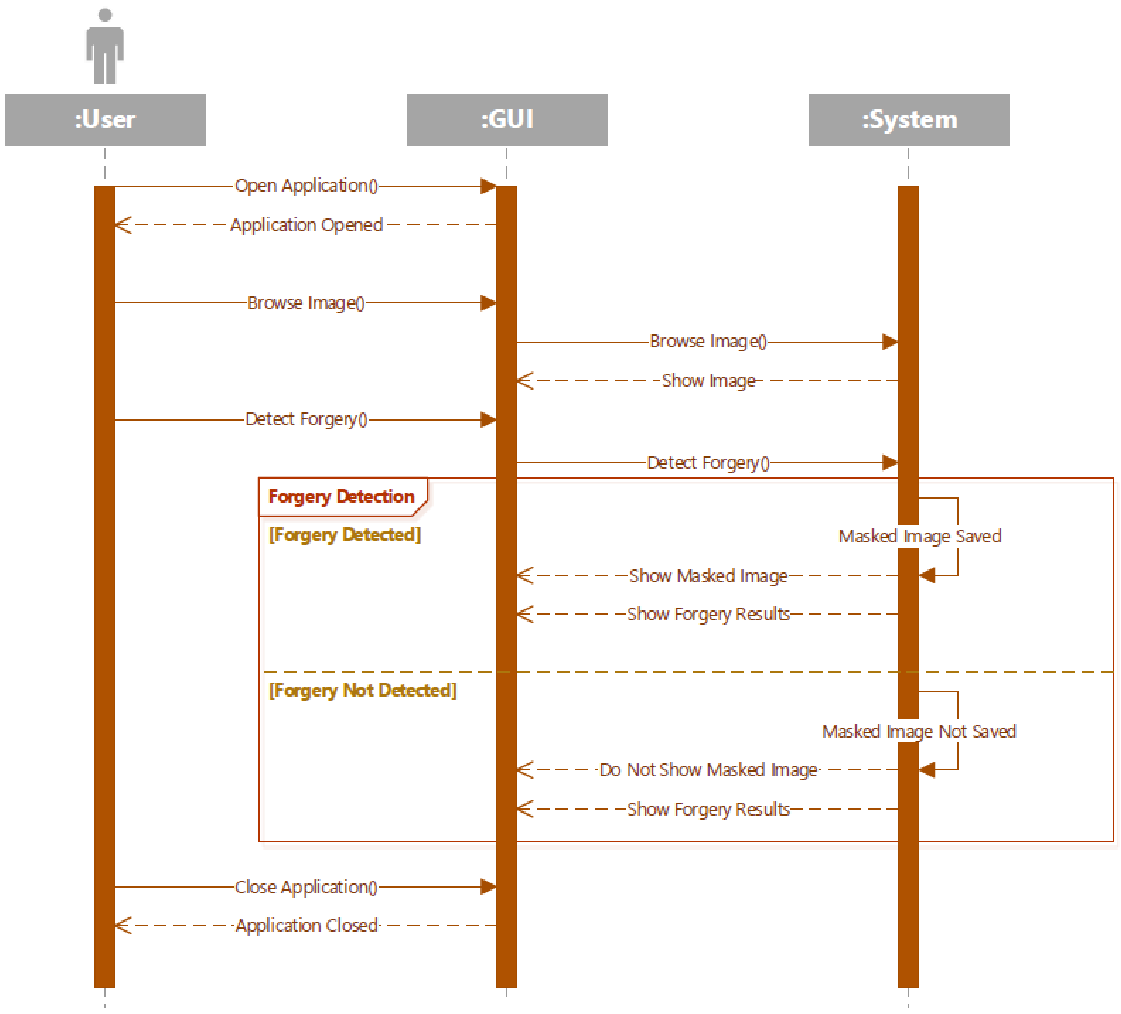


Figure 4.1: Main Sequence Diagram

4.3.1.2 Run Application Sequence Diagram

Running application is the first step of this system. It shows all the main action required to run this application.

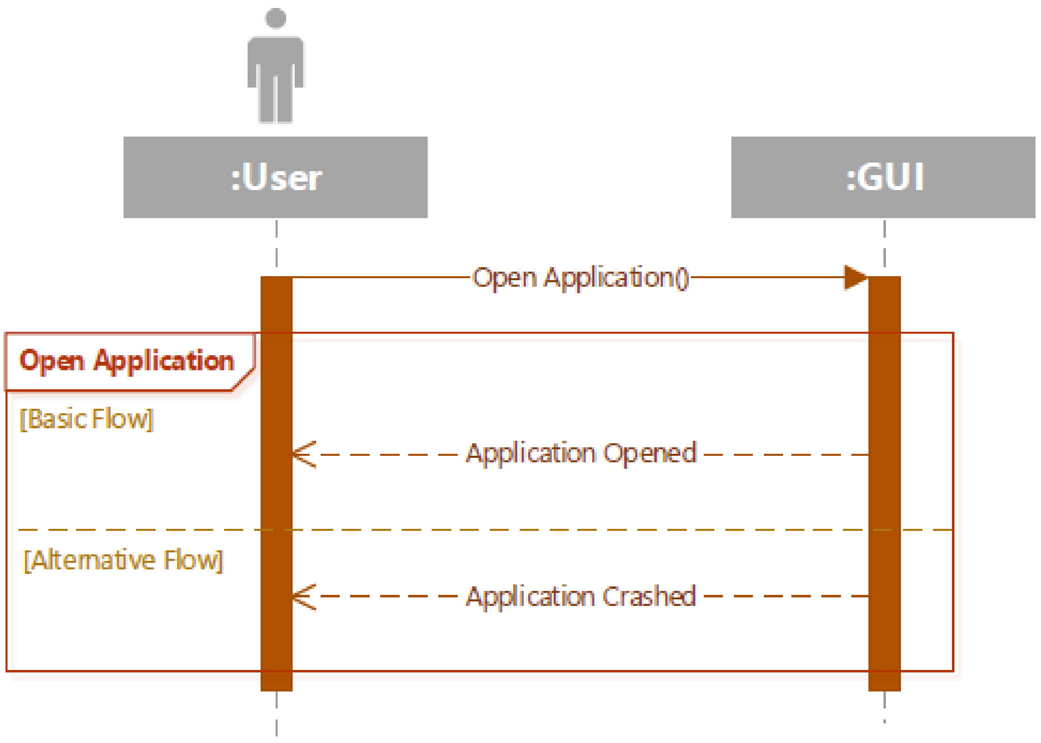


Figure 4.2: Run Application Sequence Diagram

4.3.1.3 Open Image File Sequence Diagram

User must load an image by browsing from hard drive to detect forgery in an image. It shows all the main actions required to load an image and display it in graphical user interface.

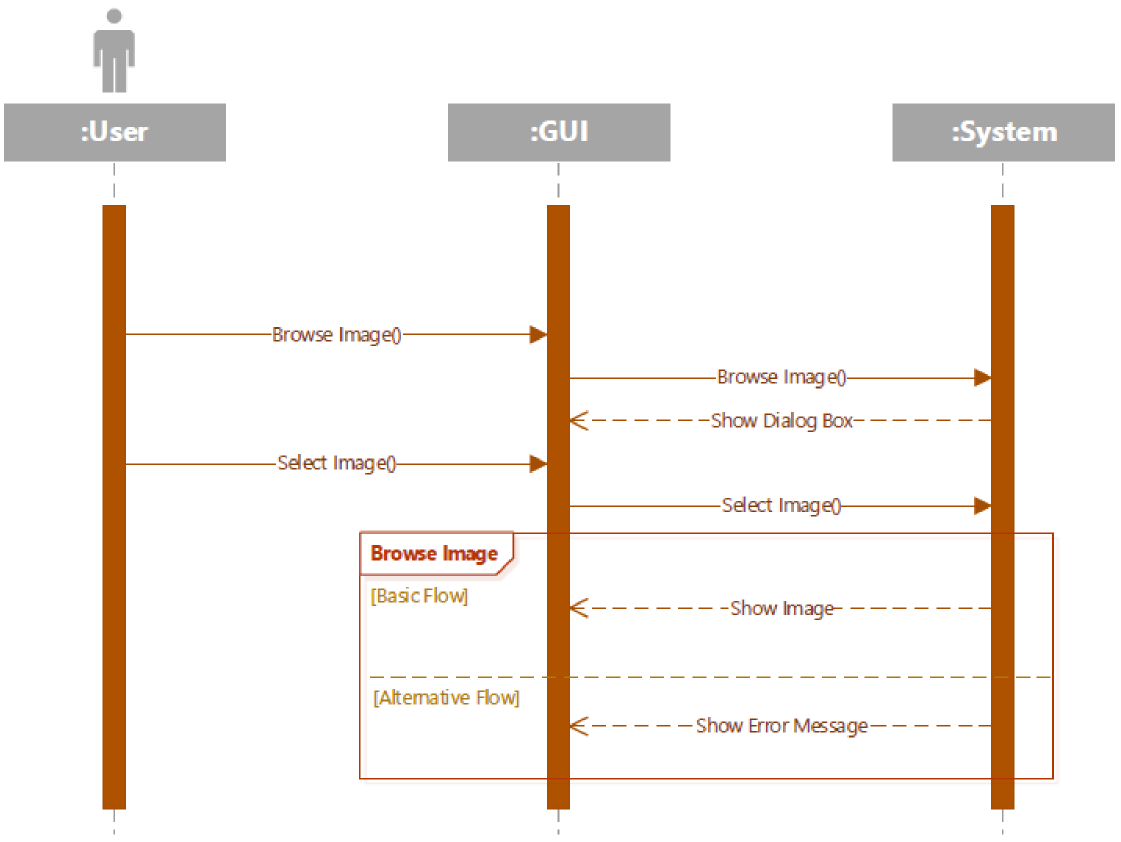


Figure 4.3: Open Image File Sequence Diagram

4.3.1.4 Save and Display Masked Image Sequence Diagram

System must save the masked image to show the forged region and results to the user. After saving this predicted forged region masked image it is displayed to user in graphical user interface. It shows all the main actions required to save and display masked image.

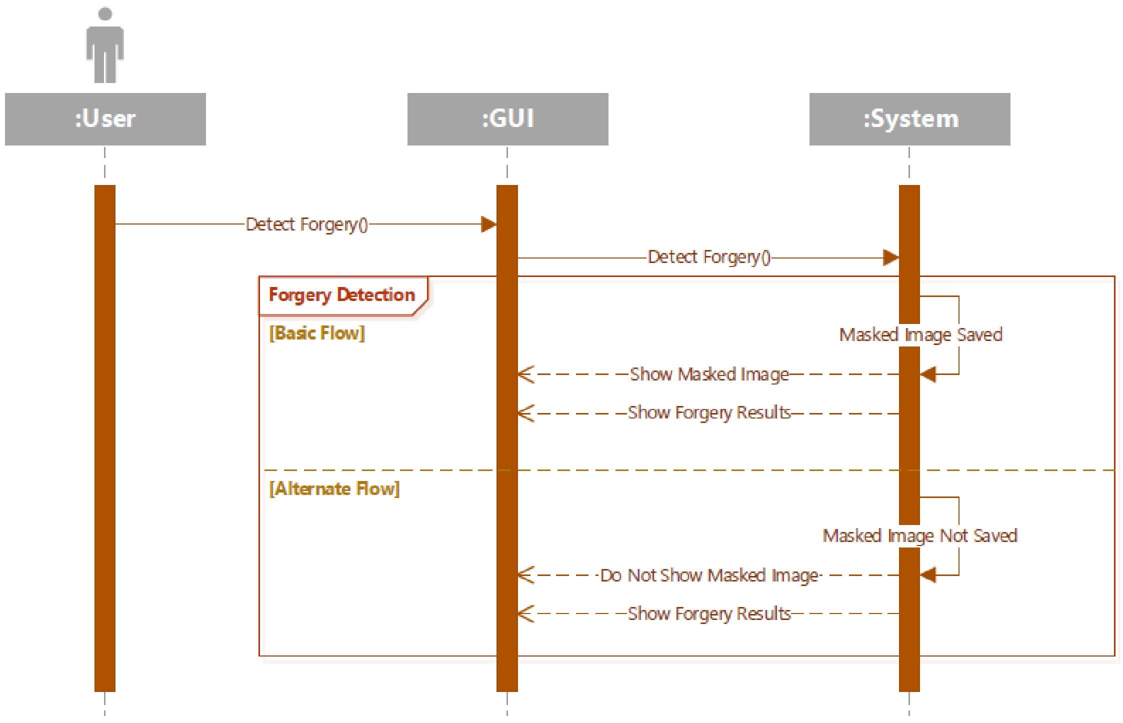


Figure 4.4: Save and Display Masked Image Sequence Diagram

4.3.1.5 Exit Application Sequence Diagram

Exiting application is the last step of this system if user want to exit application. It shows all the main action required to exit this application.

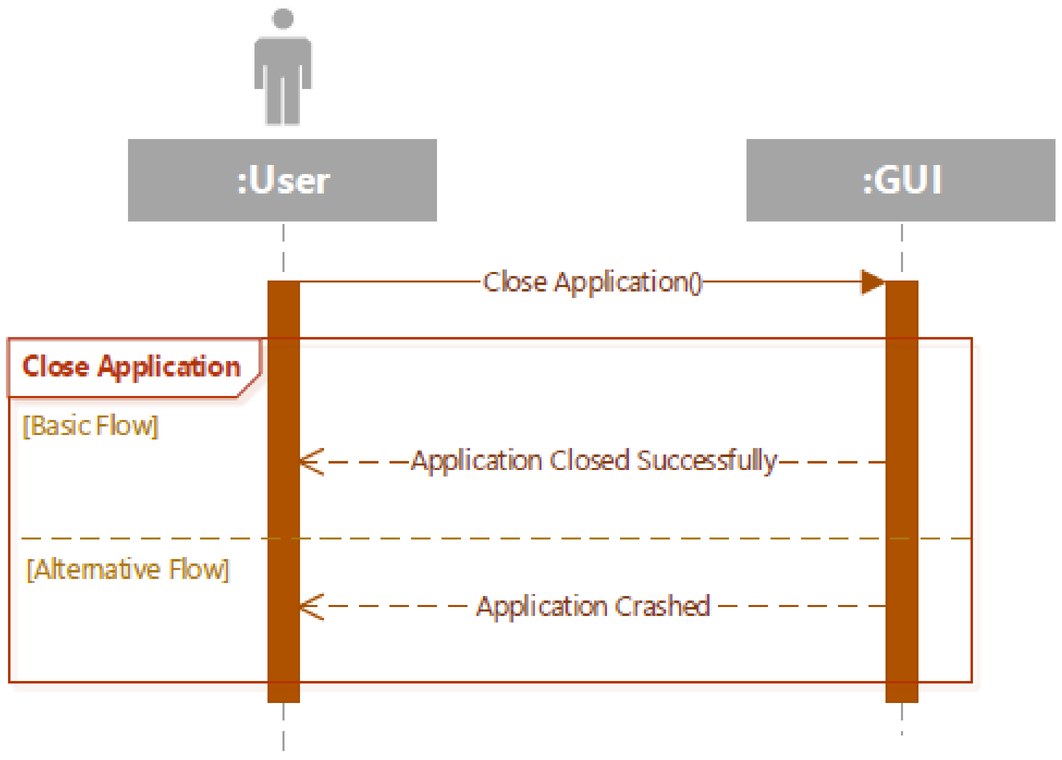


Figure 4.5: Exit Application Sequence Diagram

4.4 High Level Design

This section describe the high level design of this system. It show how the overall application will work as it shows the flow of application from input image to output results. The steps of copy-move forgery detection (CMFD) technique used in this application are shown in Fig. 4.1.

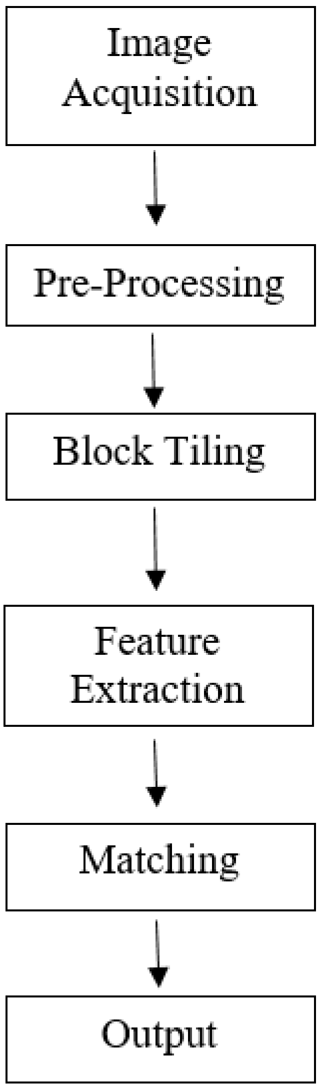


Figure 4.6: Workflow of CMFD

All these steps are explained in chapter 5 system implementation.

4.5 Database Design

4.5.1 Introduction

In this project the database I used is a database called CoMoFoD [6]. CoMoFoD is an image database for Copy-Move forgery detection and it has 200 images of resolution [512 x 512]. Images are categorized in 5 categories according to applied transformation. Also, six different postprocessing methods are applied to images in all categories.

4.5.2 Forgery method

Images are forged by copying a part of an original image and pasting it on a different location in the same image. The main goal was to embed the copied region into the original image content without leaving any visible traces of tampering. In some cases, copied part was transformed before changing its location. Several types of transformations are applied on these images, and grouped images in 5 categories according to applied transformation.

- **Translation**

  It is a process where only a region is copied and translated to the new location in an image without performing any transformation.

- **Rotation**

  It is a process where a region is copied and then translated and rotated to another location in an image.

- **Scaling**

  It is a process where a region is copied, and scaling is applied on it and translated to another location in an image.

- **Distortion**

  It is a process where a region is copied, and distortion is applied on it and translated to another location in an image

- **Combination**

  It is a process where two or more transformation are applied on a region that is copied before moving it to another location in an image.

Size of copied region differ from image to image. Smallest copied part in 512 x 512 images is 0.14 percent of image size. Biggest copied region in 512 x 512 images is 14.32 percent of image size.

In this application only translation transformation category is used for testing purpose.

4.5.3 Postprocessing methods

There are many different types of post processing methods that are applied to forged images with the aim of hiding tampering traces in this database. Post processing methods applied on all forged and original images are:

- JPEG compression.
- Contrast adjustments.
- Brightness change.
- Image blurring.
- Noise adding.
- Color reduction.

In this application all those images are used for testing purpose that have above post processing method applied on them.

4.6 GUI Design

Graphical user interface of this application is very simple and user friendly so it can be used by almost any user. Main screen of this application is shown in Fig. 4.2 and Fig. 4.3 below.

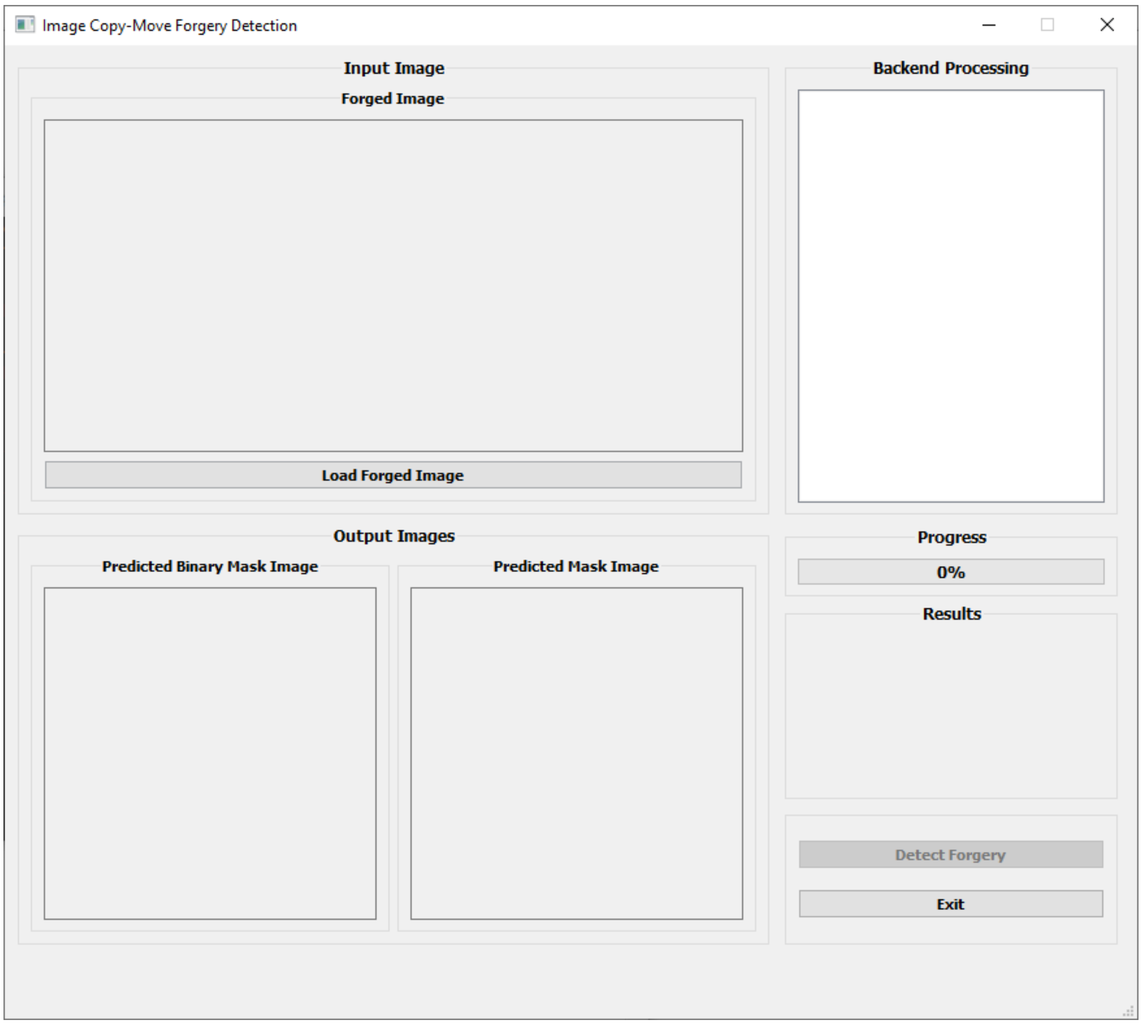
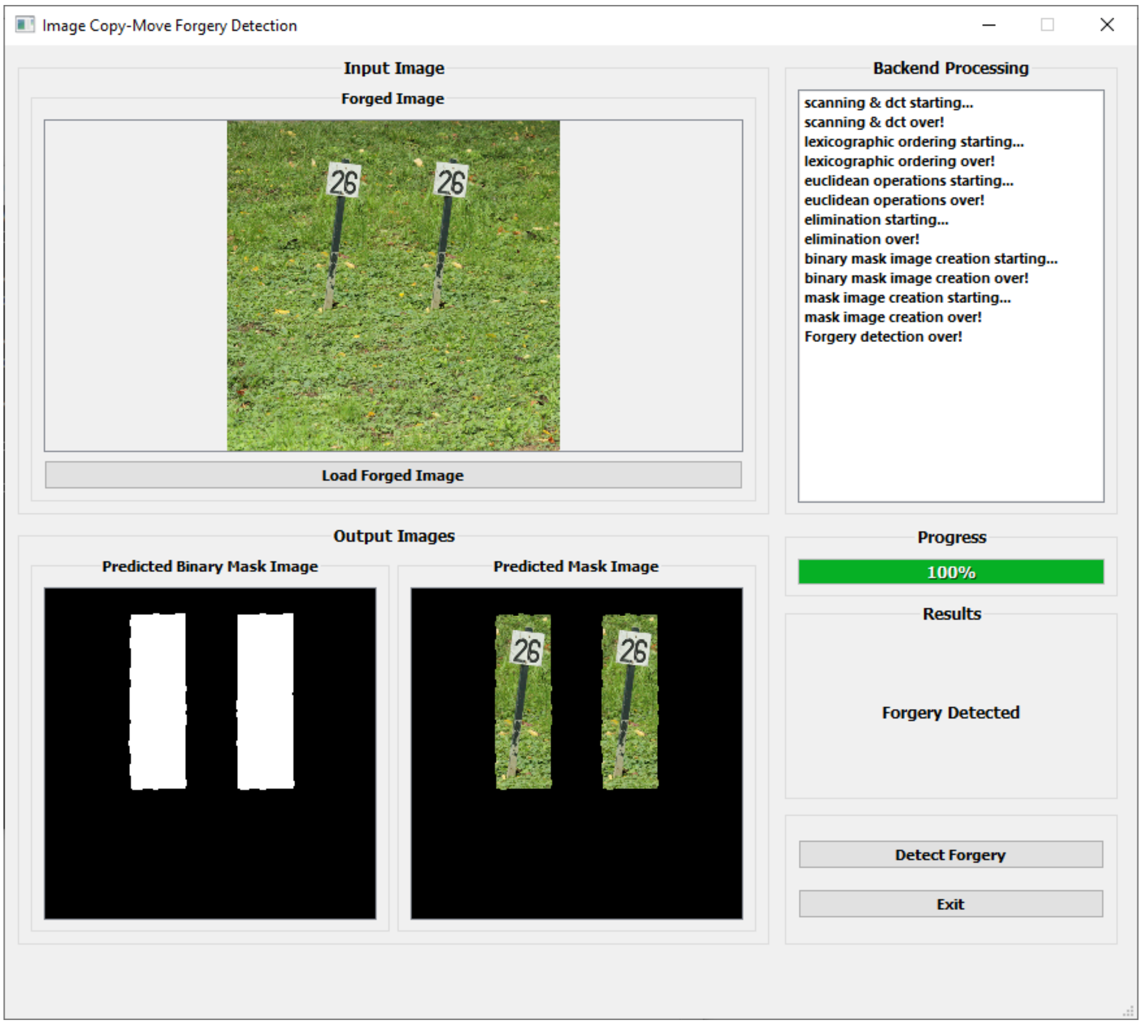


Figure 4.7: Main screen on startup



```
scanning & dct starting...
scanning & dct over!
lexicographic ordering starting...
lexicographic ordering over!
euclidean operations starting...
euclidean operations over!
elimination starting...
elimination over!
binary mask image creation starting...
binary mask image creation over!
mask image creation starting...
mask image creation over!
Forgery detection over!
```

Figure 4.8: Main screen after forgery detection

Chapter 5

System Implementation

In this chapter system implementation techniques are discussed in details as well as the tools and technologies that are used to develop this application.

5.1 Methodology

To structure this application different digital image processing techniques are used to detect forgery in an image.

5.1.1 Image Acquisition

In this step the user is asked to input an image when the image is acquired than next step in algorithm is followed.

5.1.2 Pre-Processing

The pre-processing applied here is to convert the color image into gray scale image.

5.1.3 Block Tiling

In the block tiling whole image is divided into small regions and feature information is extracted for each block and compared. The grayscale image is divide into overlapping blocks of size (b x b). It will be divided into (M-b+1)*(N-b+1) blocks. In this application b = 8.

5.1.4 Feature Extraction

For each block compute the DCT coefficients as the features. DCT is applied to the blocks in order to be resistant to compression, then apply the zigzag scan to the conclusion of the DCT and extracted the low frequency region. The purpose of this is to obtain a meaningful low-frequency segment in the image and obtain a vector of (1x64) after zigzag scanning.

After applying these operations to all blocks the vectors representing each block are listed one by one to obtain a matrix. The size of this matrix is 16 x blocks. To reduce the cost of comparing the vectors reorder these matrix columns by using lexicographic sorting (according to dictionary order). Now the similar vectors are close to each other, and also to avoid losing which block represents which vector, we added the initial coordinates of the blocks to the end of each (1x16) vector. Now the vector size is 1x18, but these two columns are not taken into account when sorting.

5.1.5 Matching

The similarities of the vectors with neighboring vectors are examined by applying Euclidean. If the calculated value is closer to 0 the two vectors are similar. The Euclidean threshold is a set to 3.5 in this application to get good results.

The vectors that cross the similarity threshold pass through the distance threshold 100. This purpose is not to associate the nearby areas. This process is done with Euclidean, but this time the initial coordinates of the blocks represented by vectors are used instead. In this process, if the distance between the two blocks is not sufficiently high, the minimum distance between the two vectors is 100 pixels. In this application good results are observed even in images with very similar regions such as desert and sky.

The directions of two vectors passing the distance threshold are calculated. The value of this direction is increased in direction space by 1, finally there is the greatest direction value in direction space and the blocks represented by the vectors in this direction are marked by white color in a blank image and this predicted binary mask image is saved on hard drive.

5.1.6 Output

First opening operation is performed on predicted binary mask image that is created in the last step to remove noise and isolate pixels that does not belong to the forged region. Next area of the two largest objects is calculated in pixels and compared by taking absolute difference of area for these largest two objects. If the absolute difference is less than 20 pixels and there are less than 4 objects in this predicted binary mask image than the image is forged otherwise it is not forged. Next masked image is created by taking and operation of input image and predicted binary mask image that is updated in this step. Output of the program will be this masked image that shows the forged region in image if the image is forged image otherwise this masked image is not displayed on screen. A message is also displayed on screen if image is forged or not.

5.2 Tools and Technology Used

This application is developed in python version 3.7 by using PyCharm 2019.2.3 (Professional Edition) IDE. For image processing and other operations OpenCV2 and NumPy is used. GUI is designed in PyQt5. All the diagrams are made by Microsoft Visio. All image dataset results are compiled in Microsoft Excel. This Report is written in Microsoft Word. This report is converted into LaTeX by using TeXstudio.

5.2.1 Python

Python is an interpreted, high-level, general-purpose programming language.

5.2.2 PyCharm

PyCharm is an integrated development environment used in computer programming, specifically for the Python language.

5.2.3 OpenCV2

OpenCV is a library of programming functions mainly aimed at real-time computer vision.

5.2.4 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

5.2.5 PyQt5

Python bindings for the Qt cross platform UI and application toolkit

5.2.6 Microsoft Visio

Microsoft Visio is a diagramming and vector graphics application.

5.2.7 Microsoft Excel

Microsoft Excel is a spreadsheet which is used for calculation, graphing tools, pivot tables, and a macro programming language called Visual Basic for Applications.

5.2.8 Microsoft Word

Microsoft Word is a word processor.

5.2.9 TeXstudio

TeXstudio is a full-featured LaTeX editor. The objective of TeXstudio is to make writing LaTeX documents as easy and convenient as possible. Some of the outstanding features of TeXstudio are a PDF viewer integrated with (almost) word level synchronization, live online preview, advanced syntax highlighting, live reference verification, citations, latex commands, spelling and grammar .

Chapter 6

System Testing and Evaluation

In this chapter performance testing and evaluation of the system will be discussed. Testing of the whole project is done by using the Manual Based Testing approach. Each component of the system has been tested manually. Application evaluation is done by using various tools and techniques on database. If this phase gives results according to the requirements than the testing is said to successful.

6.1 Graphical User Interface Testing

Graphical user interface testing is a process to evaluate how easily a user can interact with the application using the interface. The graphical user interface should be user friendly so that the user with can easily understand how to use the application by simply reading the text or images on buttons.

6.2 Usability Testing

The usability testing is done so that almost any user will be able to use this application very easily.

6.3 Software Performance Testing

Software performance testing is done to verify all the non-functional requirements of the system.

6.4 Compatibility Testing

Compatibility testing deals with the platform on which the application can be loaded. Since this application is desktop application so it should be run on a device containing windows operating system.

6.5 Load Testing

Load testing deals with the response time of the application. In this application the load testing is performed to measure the time this application takes to perform operations on an image and return results.

6.6 Test Cases

6.6.1 Test Case: Run Application

Test Case ID	**T-01**
Function To Be Tested	Run application
Initial State	The application must be installed on operating system
Input	User double click the application icon from the operating system
Expected Output	The application should be running successfully
Status	Pass

Table 6.1: Test Case 1: Run Application

6.6.2 Test Case: Open Image File

Test Case ID	**T-02**
Function To Be Tested	Open Image File
Initial State	Application must be running on operating system
Input	User click on load image button
Expected Output	Image should be opened in image block
Status	Pass

Table 6.2: Test Case 2: Open Image File

6.6.3 Test Case: Display Masked Image

Test Case ID	**T-03**
Function To Be Tested	Display Masked Image
Initial State	An image must be created by the by system
Input	Creation of an image by system
Expected Output	An image should be displayed in image block
Status	Pass

Table 6.3: Test Case 3: Display Masked Image

6.6.4 Test Case: Save Masked Image

Test Case ID	**T-04**
Function To Be Tested	Save Masked Image
Initial State	System must have made changes on an Image by performing operation on it
Input	Prediction of forged region in image by the system
Expected Output	Image should be saved
Status	Pass

Table 6.4: Test Case 4: Save Masked Image

6.6.5 Test Case: Exit Application

Test Case ID	**T-05**
Function To Be Tested	Exit Application
Initial State	Application must be running on operating system
Input	User click on exit application button and select yes when prompted
Expected Output	Application should be closed
Status	Pass

Table 6.5: Test Case 5: Exit Application

6.7 Performance Measures

The performance evaluation of a forgery detection algorithm can be done at two levels: at image level, where the focus is on the ability to detect if there is a forgery or not and at pixel level, where the accuracy of detecting the tampered regions. In this project both pixel level and image level evaluation is done.

At pixel level evaluation the important parameters are:

- **True Positive**

  These are correctly detected forged pixels. It is denoted as TP.

- **True Negative**

  These are correctly detected non-forged pixels. It is denoted as TN.

- **False Positive**

  These are pixels that have been falsely detected as forged. It is denoted as FP.

- **False Negative**

  These are forged pixels that are falsely missed. It is denoted as FN.

At image level evaluation the important parameters are:

- **True Positive**

  These are correctly detected forged images.

- **True Negative**

  These are correctly detected non-forged images.

- **False Positive**

  These are images that have been falsely detected as forged.

- **False Negative**

  These are forged images that are falsely missed.

The performance of the algorithm is evaluated using following metrics:

- **Precision**

  It is the probability that a detected forgery is truly a forgery and it is computed as:

  Precision = (TP) / (TP + FP)

- **Recall**

  It is the probability that a forged image is detected and it is computed as:

  Recall = (TP) / (TP + FN)

- **F1 Score**

  Both Precision and Recall is combined in a single value. It is also known as F1 Measure. It is computed as:

  F1 Measure = 2 * (Recall * Precision) / (Recall + Precision)

- **Accuracy**

  Accuracy is the ratio of correctly predicted observation to the total number of observations. It is computed as:

  Accuracy = (TP +TN) / (TP + FP + FN + TN)

Precision, recall, f1-mesure and accuracy evaluation matric scores will be used to report copy move forgery detection performance.

6.8 Test Results

6.8.1 Pixel Level Evaluation Testing

In order to test the application, tests were performed on different type of 540 forged images that have different post processing method applied on them. The performance results are listed in Table 6.1.

6.8.2 Image Level Evaluation Testing

In image level evaluation, a total of 570 images are considered out of which 540 were forged 30 were not forged (Original Images). The performance results are listed in Table 6.2.

Postprocessing method	**Average precision**	**Average recall**	**Average f1-measure**	**Average accuracy**	**Images with f1-mesure > 0.5**
No postprocessing	0.8506	0.8794	0.8485	0.9906	30
Brightness change, range (0.01, 0.95)	0.8532	0.8774	0.8465	0.9906	29
Brightness change, range (0.01, 0.9)	0.8630	0.8750	0.8465	0.9910	30
Brightness change, range (0.01, 0.8)	0.8465	0.8377	0.8196	0.9908	27
Color adjustments, range (0.01, 0.95)	0.8473	0.8795	0.8439	0.9898	29
Color adjustments, range (0.01, 0.9)	0.8428	0.8722	0.8373	0.9894	28
Color adjustments, range (0.01, 0.8)	0.8203	0.8768	0.8277	0.9882	28
Color reduction, 32 intensity levels	0.8578	0.8809	0.8524	0.9907	30
Color reduction, 64 intensity levels	0.8520	0.8911	0.8515	0.9903	29
Color reduction, 128 intensity levels	0.8515	0.8906	0.8509	0.9904	30
Image blurring, 3x3 averaging filter	0.8342	0.8592	0.8222	0.9886	27
Image blurring, 5x5 averaging filter	0.8127	0.8518	0.8082	0.9870	29
JPEG compression, quality factor 50	0.8812	0.8349	0.8363	0.9902	29
JPEG compression, quality factor 60	0.8582	0.8432	0.8272	0.9885	29
JPEG compression, quality factor 70	0.8666	0.8560	0.8408	0.9906	30
JPEG compression, quality factor 80	0.8585	0.8739	0.8425	0.9904	30
JPEG compression, quality factor 90	0.8535	0.8652	0.8396	0.9903	29
JPEG compression, quality factor 100	0.8531	0.8595	0.8363	0.9904	30
Overall Dataset Results	**0.8504**	**0.8659**	**0.8370**	**0.9898**	**523**

Table 6.6: Pixel Level Evaluation Results

Postprocessing method	Total Images	Correctly Detected Images	Image Level Accuracy
Original images	30	27	90%
No postprocessing	30	27	90%
Brightness change, range (0.01, 0.9)	30	27	90%
Brightness change, range (0.01, 0.8)	30	25	83%
Color adjustments, range (0.01, 0.95)	30	27	90%
Color adjustments, range (0.01, 0.9)	30	25	83%
Color adjustments, range (0.01, 0.8)	30	25	83%
Color reduction, 32 intensity levels	30	28	93%
Color reduction, 64 intensity levels	30	28	93%
Color reduction, 128 intensity levels	30	24	80%
Image blurring, 3x3 averaging filter	30	26	87%
Image blurring, 5x5 averaging filter	30	24	80%
JPEG compression, quality factor 50	30	26	87%
JPEG compression, quality factor 60	30	25	83%
JPEG compression, quality factor 70	30	28	93%
JPEG compression, quality factor 80	30	27	90%
JPEG compression, quality factor 90	30	28	93%
JPEG compression, quality factor 100	30	28	93%
Noise adding, variance = 0.0005	30	27	90%
Overall Dataset Results	**570**	**502**	**88%**

Table 6.7: Image Level Evaluation Results

Chapter 7

Conclusions

Image forgery detection application will allow the user to enter data in the form of an image and get the result in the form of image mask that highlights the copy move forged region in input image. It also show a message if whether image is forged or not. This application is completed to the best of my abilities and has been a great learning point for me. It can be improved greatly.

7.1 Future Improvements

This is the first version of this application with simple GUI and minimal functionalities because of time constraint. In future this system can be improved as follows.

7.1.1 Better GUI

GUI can be improved to make it more modern looking by adding multiple themes and other options so it can be used by all type of users because each user has different preferences. Some users like dark themes and some like light themes.

7.1.2 Multiple Platforms

The application is currently only developed for Windows Operating System. It will be released for other operating systems in the future.

7.1.3 Larger Dataset

The application is only tested on one dataset and only one type of transformation with multiple post processing methods applied over this single transformation. In the future, it can be tested over other transformations from this dataset and other dataset as well.

It is only possible by integrating multiple copy move forgery detection algorithms and techniques so it can handle almost any type of copy move forgery.

Bibliography

[1] Babak Mahdian and Stanislav Saic. A bibliography on blind methods for identifying image forgery. *Signal Processing: Image Communication*, 25(6):389–399, 2010. Cited on p. 1.

[2] Judith A Redi, Wiem Taktak, and Jean-Luc Dugelay. Digital image forensics: a booklet for beginners. *Multimedia Tools and Applications*, 51(1):133–162, 2011. Cited on p. 1.

[3] Tanzeela Qazi, Khizar Hayat, Samee U Khan, Sajjad A Madani, Imran A Khan, Joanna Kołodziej, Hongxiang Li, Weiyao Lin, Kin Choong Yow, and Cheng-Zhong Xu. Survey on blind image forgery detection. *IET Image Processing*, 7(7):660–670, 2013. Cited on p. 1.

[4] Sevinc Bayram, Husrev Taha Sencar, and Nasir Memon. A survey of copy-move forgery detection techniques. In *IEEE Western New York Image Processing Workshop*, pages 538–542. Citeseer, 2008. Cited on p. 5.

[5] Yanping Huang, Wei Lu, Wei Sun, and Dongyang Long. Improved dct-based detection of copy-move forgery in images. *Forensic science international*, 206(1-3):178–184, 2011. Cited on p. 7.

[6] Dijana Tralic, Ivan Zupancic, Sonja Grgic, and Mislav Grgic. Comofod—new database for copy-move forgery detection. In *Proceedings ELMAR-2013*, pages 49–54. IEEE, 2013. Cited on p. 23.