P-3318

IMPLICATION OF DIGITAL IMAGE FORENSICS VIA INTRINSIC FINGERPRINT

PROJECT REPORT

Submitted by

J.NAVEEN GUPTHA	**71206205029**
R.PARTHIBAN	**71206205032**
R.SRIRAM	**71206205049**

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY:: CHENNAI 600025

APRIL 2010

BONAFIDE CERTIFICATE

Certified that this project report "IMPLICATION OF DIGITAL IMAGE FORENSICS VIA INTRINSIC FINGERPRINT " is the bonafide work of "J.NAVEENGUPTHA, R.PARTHIBAN, R.SRIRAM", who carried out the project work under my supervision.

[illegible]

SIGNATURE
Dr. LS Jayashree Ph.D
Head of the Department
Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641 006.

[illegible]

SIGNATURE
Ms. P.Shenbagam M.E
Lecturer
Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641 006.

The candidates with University Register No 71206205029, 71206205032and 71206205049 were examined by us in the project viva-voice examination held on 20-4-2010

[illegible] 20/4/10
INTERNAL EXAMINER

[illegible] 20.4.10
EXTERNAL EXAMINER

DECLARATION

We,

J.NAVEENGUPTHA	Reg.No:	71206205029
R.PARTHIBAN	Reg.No:	71206205032
R.SRIRAM	Reg.No:	71206205049

Hereby declare that the project entitled " **IMPLICATION OF DIGITAL IMAGE FORENSICS VIA INTRINSIC FINGERPRINT",** submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) degree, is a record of original work done by us under the supervision and guidance of Ms P.Shenbagam M.E., Lecturer, Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place:Coimbatore

Date : 19-4-2010

[illegible] J.Naveenguptha

R.Parthiban R.Parthiban

[illegible] R.Sriram

Project Guided by

[illegible]

[Ms P.Shenbagam M.E]

ACKNOWLEDGEMENT

We express our sincere thanks to our Chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc., F.I.E.,** Co-Chairman Dr. B.K. **Krishnaraj Vanavarayar,** Correspondent **Shri.M.Balasubramaniam and Director Dr.J.Shanmugam** for all their support and ray of strengthening hope extended. We are immensely grateful to our Principal **Dr.Ramachandran,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy,** Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project co-ordinator, **Dr.L.S Jayashree** Prof, Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide and extend our gratitude towards **,Ms P.Shenbagam M.E.** Lecturer, Department of Information Technology for her helpful guidance and valuable support given to us throughout this project.

We also extend our heartfelt thanks to our class advisor, **Ms N.Rajathi M.E.** Asst.Prof, Department of Information Technology for providing us her support which really helped us.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

ABSTRACT

ABSTRACT

Digital multimedia contents are often transmitted over networks without any protection. This raises serious security concerns since the receivers/subscribers do not know what processes have been applied to multimedia data, and neither do they know whether this copy comes from a trusted source. Therefore, it is critical to provide forensic tools to identify the history of operations applied to multimedia data. A novel method for image forgery detection is proposed. The method exploits the fingerprints left by JPEG compression. The fingerprints are obtained by estimating the quantization matrix and are used to detect different forgeries such as copy-paste, cropping, rotation, and brightness changes. Algorithms were developed for detecting forgeries on a single JPEG compressed image and for the more difficult scenario where the forged image is decompressed.

The proposed algorithms' performance improves with increasing quality factor unlike block-artifact based detection schemes. Thus it would provide a good complement to the reliability offered by blocking-artifact based schemes at lower quality factors. A database of around 100 images was used for testing.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

INTRODUCTION

1.1 General:

In recent days the images sent through the internet are tapped and altered without the knowledge of the owner and they are misused by altering the image.

1.2 Solution:

To prevent this the computer forensics is used.

Computer forensics is a branch of forensic science pertaining to legal evidence found in computers and digital storage media. Computer forensics is also known as digital forensics.

The goal of computer forensics is to explain the current state of a digital artifact. The term digital artifact can include a computer system, a storage medium (such as a hard disk or CD-ROM), an electronic document (e.g. an email message or JPEG image) or even a sequence of packets moving over a computer network. The explanation can be as straightforward as "what information is here?" and as detailed as "what is the sequence of events responsible for the present situation?”

The field of computer forensics also has sub branches within it such as firewall forensics, network forensics, database forensics and mobile device forensics.

There are many reasons to employ the techniques of computer forensics:

- In legal cases, computer forensic techniques are frequently used to analyze computer systems belonging to defendants (in criminal cases) or litigants (in civil cases).
- To recover data in the event of a hardware or software failure.
- To analyze a computer system after a break-in, for example, to determine how the attacker gained access and what the attacker did.
- To gather evidence against an employee that an organization wishes to terminate.
- To gain information about how computer systems work for the purpose of debugging, performance optimization, or reverse-engineering.

Special measures should be taken when conducting a forensic investigation if it is desired for the results to be used in a court of law. One of the most important measures is to assure that the evidence has been accurately collected and that there is a clear chain of custody from the scene of the crime to the investigator---and ultimately to the court. In order to comply with the need to maintain the integrity of digital evidence, British examiners comply with the Association of Chief Police Officers (A.C.P.O.) guidelines. These are made up of four principles as follows:-

1. No action taken by law enforcement agencies or their agents should change data held on a computer or storage media which may subsequently be relied upon in court.
2. In exceptional circumstances, where a person finds it necessary to access original data held on a computer or on storage media, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions.

3. An audit trail or other record of all processes applied to computer based electronic evidence should be created and preserved. An independent third party should be able to examine those processes and achieve the same result.
4. The person in charge of the investigation (the case officer) has overall responsibility for ensuring that the law and these principles are adhered to.

1.3 Objective of our project:

The aim of the project is to detect the image as morphed or not by intrinsic characteristics like pixel values, RGB, Histogram. The main modules of the project are DCT image blocking, Image matching, User interface design.

The initial approach taken was based on the assumption that the Q matrix of the untampered image is known. Such a situation arises when knowledge of the image acquisition process uniquely identifies the Q matrix used for JPEG compression. In this scenario, the forged image is saved as an un-compressed bitmap. The parameter that is used as a metric for determining forgery is related to average distortion. For each block in the image, this is calculated as a function of the remainders of the DCT coefficients with respect to the original Q matrix. Finally gives the absolute difference between input and source image.

Large values of this measure indicate that the particular block of the image is very different from the one that is expected and, hence is likely to belong to a forged image. Averaged over the entire image, this measure can be used for making a decision about authenticity of the image. The threshold for classification is calculated from a training data set.

1.3.1 Single compression quantization matrix detection

An important step in the general approach towards using the above described methods for forgery detection is the estimation of the quantization table used for compression. Initially, our method revolved around estimating the quantizer step based on the peaks that occur in the histogram of the DCT coefficients. However, this method failed to provide a reliable estimate of the Q matrix because of the rounding operations that take place during the DCT transform cause the peaks in the histogram to spread out thereby making them difficult to detect. However, transforming the histogram to the frequency domain by obtaining the power spectral density makes the peaks more prominent and the quantization step easy to estimate.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1FEASIBILITY STUDY

2.1.1 EXISTING SYSTEM

Copy-Move forgery a part of the image itself is copied and pasted into another part of the same image. This is usually performed with the intention to make an object “disappear” from the image by covering it with a segment copied from another part of the image. Textured areas, such as grass, foliage, gravel, or fabric with irregular patterns, are ideal for this purpose because the copied areas will likely blend with the background and the human eye cannot easily discern any suspicious artifacts.

Because the copied parts come from the same image, its noise component, color palette, dynamic range, and most other important properties will be *compatible* with the rest of the image and thus will not be detectable using methods that look for incompatibilities in statistical measures in different parts of the image. To make the forgery even harder to detect, one can use the feat.

2.1.2 PROPOSED SYSTEM

We propose the parameter that is used as a metric for determining forgery as related to average distortion. For each block in the image, this is calculated as a function of the remainders of the DCT coefficients with respect to the quantization matrix (Q matix) used. The forgery methods investigated using this method include rotation, copy- paste, cropping, and brightness, all of which are easily achievable using both commercially and freely available software. One of the baseline assumptions of this approach is that only one type of forgery is applied per image In addition, even though the methods discussed here are easily extendible to color images, only grayscale images were investigated.

The quantization step introduces distinct fingerprints within each block and across boundaries of adjacent blocks. These artifacts provide built in reliable fingerprints that allow passive integrity checks. In addition, they can also be used in the estimation of the quantization matrix used for compression.

2.2 HARDWARE REQUIREMENT SPECIFICATIONS:

Processor	:	Pentium IV
Speed	:	Above 500 MHz
RAM capacity	:	256 MB
Floppy disk drive	:	1.44 MB
Hard disk drive	:	40 GB
Monitor	:	15” Color Monitor
Keyboard	:	Multimedia Keyboard
Mouse	:	Scroll Mouse

2.3 SOFTWARE REQUIREMENT SPECIFICATIONS:

Operating System	:	Platform Independent
Front-end used	:	Java
Back-end used	:	Mysql

2.4 SOFTWARE OVERVIEW

Java:

Java is a new computer programming language developed by Sun Microsystems. Java has a good chance to be the first really successful new computer language in several decades. Advanced programmers like it because it has a clean, well-designed definition. Business likes it because it dominates an important new application, Web programming.

Java Has Several Important Features:

- A Java program runs exactly the same way on all computers. Most other languages allow small differences in interpretation of the standards.
- It is not just the source that is portable. A Java program is a stream of bytes that can be run on any machine. An interpreter program is built into Web browsers, though it can run separately. Java programs can be distributed through the Web to any client computer.
- Java applets are safe. The interpreter program does not allow Java code loaded from the network to access local disk files, other machines on the local network, or local databases. The code can display information on the screen and communicate back to the server from which it was loaded.

A group at Sun reluctantly invented Java when they decided that existing computer languages could not solve the problem of distributing applications over the network. C++ inherited many unsafe practices from the old C language. Basic was too static and constrained to support the development of large applications and libraries.

Today, every major vendor supports Java. Netscape incorporates Java support in every version of its Browser and Server products. Oracle will support Java on the Client, the Web Server, and the Database Server. IBM looks to Java to solve the problems caused by its heterogeneous product line.

The Java programming language and environment is designed to solve a number of problems in modern programming practice. It has many interesting features that make it an ideal language for software development. It is a high-level language that can be characterized by all of the following buzzwords:

Features

Sun describes Java as

- Simple
- Object-oriented
- Distributed
- Robust
- Secure
- Architecture Neutral
- Portable
- Interpreted
- High performance

- Multithreaded
- Dynamic.

Overview of the Swing

The Swing package is part of Java Foundation Classes (JFC) in the Java platform. The JFC encompasses a group of features to help people build GUIs; Swing provides all the components from buttons to split panes and tables.

The Swing package was first available as an add-on to JDK 1.1. Prior to the introduction of the Swing package, the Abstract Window Toolkit (AWT) components provided all the UI components in the JDK1.0 and 1.1 platforms. Although the Java2 Platform still supports the AWT components, we strongly encourage using Swing components instead. You can identify Swing components because their names start with J. The AWT button class, for example, is named Button, whereas the Swing button class is named JButton. In addition, the AWT components are in the java.awt package, whereas the swing components are in the javax.swing package.

As a rule, programs should not use "heavyweight" AWT components alongside Swing components. Heavyweight components include all the ready-to-use AWT components, such as Menu and ScrollPane, and all components that inherit from the AWT canvas and Panel classes. When Swing components (and all other "lightweight" components) overlap with heavyweight components, the heavyweight component is always painted on top.

Swing Features and Concepts:

It introduces Swing's features and explains all the concepts we need to be able to use Swing components effectively.

Swing Components and the Containment Hierarchy:

Swing provides many standard GUI components such as buttons, lists, menus and textareas, which we combine to create our program's GUI. It also includes containers such as windows and tool bars.

Layout Management:

Containers use layout managers to determine the size and position of the components they contain. Borders affect the layout of Swing GUIs by making Swing components larger. We can also use invisible components to affect layout.

Event Handling:

Event handling is how programs respond to external events, such as the user pressing a mouse button. Swing programs perform all their painting and event handling in the event-dispatching thread.

Painting:

Painting means drawing the components on-screen. Although it's easy to customize a component's painting, most programs don't do anything more complicated than customizing a component's border.

Threads and Swing:

If we do something to a visible component that might depend on or affect its state, then we need to do it from the event-dispatching thread. This isn't an issue for many simple programs, which generally refer to components only in event -handling code. However, other programs need to use the invoke Later method to execute component-related calls in the event-dispatching thread.

More Swing Features and Concepts:

- Icons
- Actions
- Pluggable Look and Feel support
- Support for assistive technologies
- Separate data and state models

Features that JComponent Provides:

Except for the top-level containers, all components that begin with J inherit from the JComponent class. They get many features from JComponent, such as the ability to have borders, tool tips, and a configurable look and feel. They also inherit many convenient methods.

Icons:

Many Swing components—notably buttons and labels—can display images. We specify these images as Icon objects.

Actions:

With Action objects, the Swing API provides special support for sharing data and state between two or more components that can generate action events. For example, if we have a button and a menu item that perform the same function, consider using an Action object to coordinate the text, icon, and enabled state for the two components.

Pluggable Look and Feel:

This Feature enables the user to switch the look- and-feel of Swing components without restarting the application. The Swing library supports cross-platform look-and-feel---also Java look-and-feel--- that remains the same across all platforms wherever the program runs. The native look-and-feel is native to whatever particular system on which the program happens to be running,

including Windows and Motif. The Swing library provides an API that gives great flexibility in determining the look-and-feel of applications. It also enables you to create our own look-and-feel.

Support for Assistive Technologies:

Assistive technologies such as screen readers can use the Accessibility API to get information from swing components. Because support for the Accessibility API is built into the Swing components, our swing program will probably work just fine with assistive technologies, even if we do nothing special.

Separate Data and State Models:

Most noncontainer Swing components have models. A button (JButton), for example, has a model (ButtonModel) that stores the button's state—what its keyboard mnemonic is, Whether it's enabled, selected, or pressed, and so on.

Some components have multiple models. A list (JList), for example, uses a ListModel to hold the list's contents, and a ListSelectionModel to track the list's current Selection.

DESIGN AND DEVELOPMENT

3.DESIGN AND DEVELOPEMENT

SYSTEM DESIGN

System design is described as a process of planning a new business system or more to replace or to compliment an existing system. The system design states how a system will meet the requirements identified during the system analysis

System design focuses mainly on four distinct attributes. They are data structure, software architecture, interface representation and algorithmic details.

It describes a solution of approaching to the creation of new system. System design is a transmission from a user oriented document to a document oriented to programmers. It goes through a logical and physical design. The key points followed at the times of designing are:

- ✓ Preparing input and output specification
- ✓ Data flows and stores
- ✓ Preparing security and control specification
- ✓ Temporary and permanent collection of data
- ✓ A walk through before implementation
- ✓ Process

Reviewing the study phase activities and making decisions about which function are to be performed by the hardware, software, and human ware started in the design phase. The output, input and file design for each of the programs was done. Finally the generalized systems were explained to the management for approval.

The steps involved in designing phase were:

- ✓ The function to be performed is identified
- ✓ The input, output and file design is performed
- ✓ The system and component cost requirements is specified
- ✓ The design phase report is generated.

3.1 DATA FLOW DIAGRAM

A data flow diagram also known as "**bubble chart**" has the purpose of clarifying system requirements and identifying major transformation that will become program in system design. So it is the starting point of the phase that functionally decomposes the requirement specification down to the lowest level of details. A DFD contains series of bubbles joined by lines.

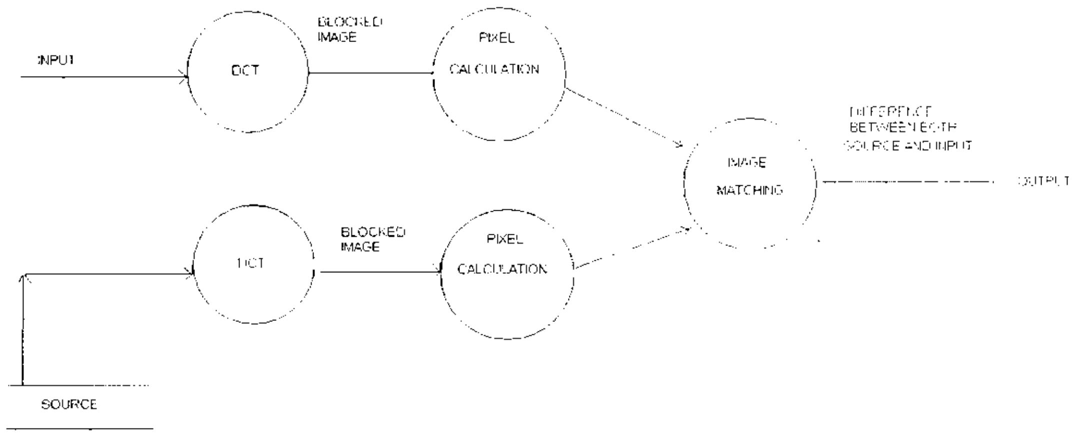


Fig 1

3.2 DESIGN PROCESS

3.2.1 DATABASE DESIGN

The goal of database design is to generate a set of relations that allows storing information easily. The database is designed in relational model in which the data are organized into entities and relations between them.

In the flow diagrams, the names are given to data flows, process and data stores. Although the names are descriptive of the data, they do not give details. the DFD gives the details of the fields used. A data dictionary has many advantages in improving analysis of user communication by establishing consistent definitions of terms elements and procedures.

In the database we upload only the source images into it and they are stored as a BLOB.

3.2.2 INPUT DESIGN

The input design is the process of entering data to the system. The goal of input design is to enter data the computer as accurate as possible. Here inputs are designed effectively so that errors made by the operations are minimized. The inputs to the system have been designed in such a way that manual forms and the inputs are coordinates where the data elements are common to the source document and to the input. The input is acceptable and understandable by the users who are using it.

The quality of the system input determines the quality for system output. Input specification describes the manner in which data entered the system process.

Input design is the process of converting user-originated inputs to a computer-based format. Input data are collected and organized into group of similar data. Once identified, appropriate input media are selected for processing.

The input design also determines the user to interact efficiently with the system. Input design is a part of overall system design that requires special attention because it is the common source for data processing error. The goal of designing input data is to make entry easy and free from errors.

User interface design

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interaction yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centered around their expertise, whether that be software design, user research, web design, or industrial design.

A user interface module is designed for language editor, language conversion options and conversion result option everything available in screen. This module provides user friendly GUI options.

DCT Image blocking

The discrete cosine transform (DCT) domain and statistical analysis based on binning techniques have been used to estimate the quantization matrices. Image manipulations such as contrast changes, Gamma correction and other image non-linearity's have been modeled and higher order statistics such as the bi spectrum have been used to identify them.

Inconsistencies in noise patterns, JPEG compression, or lighting, and alternations in correlations induced by color interpolation caused while creating a tampered picture have been used to identify inauthentic images.

Image Matching

RGB Matching Compare DCT blocks to calculate difference from input image and source images.

Histogram matching is a method in image processing of color adjustment of two images using the image histograms. It is possible to use histogram matching to balance detector responses as a relative detector calibration technique. It can be used to normalize two images, when the images were acquired at the same local illumination (such as shadows) over the same location, but by different sensors, atmospheric conditions or global illumination.

3.2.3 OUTPUT DESIGN

The output design was done so that results of processing could be communicated to the users. The various outputs have been designed in such a way that they represent the same format that the office and management used to.

Computer output is the most important and direct source of information to the user. Efficient, intelligible output design should improve the systems relationships with the user and help in decision making. A major form of output is a hardcopy from the printer

Output requirements are designed during system analysis. A good starting point for the output design is the Data Flow Diagram (DFD). Human factors reduce issues for design involves addressing internal controls to ensure readability

3.3 SYSTEM IMPLEMENTATION

The purpose of System Implementation can be summarized as follows:

It makes the new system available to a prepared set of users (the deployment), and positioning on-going support and maintenance of the system within the performing organization (the transition). At a finer level of detail, deploying the system consists of executing all steps necessary to educate the consumers on the use of the new system, placing the newly developed system into production, confirming that all data required at the start of operations is available and accurate, and validating the business functions that interact with the system are functioning properly. Transitioning the system support responsibilities involves changing from a system development to a system support and maintenance mode of operation, with ownership of the new system moving from the Project Team to the Performing Organization.

System implementation is the important stage of project when the theoretical design is tuned into practical system. The main stages in the implementation are as follows:

- Planning
- Training
- System testing and
- Changeover Planning

Planning is the first task in the system implementation. Planning means deciding on the method and the time scale to be adopted. At the time of implementation of any system people from different departments and system analysis involve. They are confirmed to practical problem of controlling various activities of people outside their own data processing departments. The line managers controlled through an implementation coordinating committee. The committee considers ideas, problems and complaints of user department, it must also consider;

- The implication of system environment
- Self selection and allocation form implementation tasks
- Consultation with unions and resources available
- Standby facilities and channels of communication

The main modules in the project are image blocking, image matching and user interface design.

IMAGE BLOCKING:

In this module the image is split into blocks of 8x8 image blocks. This is based on the assumption that the Q matrix of the untampered image is known. Such a situation arises when knowledge of the image acquisition process uniquely identifies the Q matrix used for JPEG compression. In this scenario, the forged image is saved as an uncompressed bitmap.

The parameter that is used as a metric for determining forgery is related to average distortion. For each block in the image, this is calculated as a function of the remainders of the DCT coefficients with respect to the original Q matrix as follows:

$$r = \sum_{i=1}^{8} \sum_{j=1}^{8} r_{ij} \cdot e^{r_{ij}} \qquad (1)$$

Where

$$r_{ij} = \widehat{mod}(D_{ij}, Q_{ij})$$

And

mod(a,b), gives the absolute difference between a and the closest multiple of b.

Large values of this measure indicate that the particular block of the image is very different from the one that is expected and, hence is likely to belong to a forged image. Averaged over the entire image, this measure can be used for making a decision about authenticity of the image. The threshold for classification is calculated from a training data set.

Since processing of each block remains slow we take whole input image as a single block and according to the size of the input image the source image is compared hence it is faster than the processing of each block.

After this the pixel values are calculated for input image at first the pixel values are extracted then by using shifting function we calculate red, green and blue values for the whole block. At first the pixel values will be in the binary format by left shifting 16 bits we get red, by left shifting 8 bits we get Green value and finally we get blue value. Hence using these values we create a matrix of red, green and blue values. This is used for comparing the source and input images.

IMAGE MATCHING:

RGB Matching Compare DCT blocks to calculate difference from input image and source images.

Histogram matching is a method in image processing of color adjustment of two images using the image histograms. It is possible to use histogram matching to balance detector responses as a relative detector calibration technique. It can be used to normalize two images, when the images were acquired at the same local illumination (such as shadows) over the same location, but by different sensors, atmospheric conditions or global illumination.

Here as per the size of the input image the comparison is done with the source images during run time. Finally the result is ordered based on the difference between DCT coefficient's that is calculated for both input and source images and time taken for processing is also calculated and displayed.

USER INTERFACE DESIGN:

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interaction yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centered around their expertise, whether that be software design, user research, web design, or industrial design.

A user interface module is designed for language editor, language conversion options and conversion result option everything available in screen. This module provides user friendly GUI options.

Here the various panels used are main panel, input selection panel, browser panel, type of search selection panel, processing panel, result panel.

The main panel there are two options to upload the image and to process the image. The upload option is used to upload the source images in the database. The image process option is used to compare the image which will lead to the input selection panel. In the input selection panel using browser panel we can select the input image. Then by clicking next button this panel will lead to type of search panel like original image and cropped image search according to the selection the process will be continued. After this selection the actual process will take place that is image matching is done. After the process completes total time taken to process is displayed and the results are organized based on their difference. The least difference will be at the top and maximum difference will at the bottom.

PERFORMANCE EVALUATIO

4.PERFORMANCE EVALUATION

4.1TESTING

Testing is a series of different tests that whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all work should verify that all system element have been properly integrated and performed allocated function. Testing is the process of checking whether the developed system works according to the actual requirement and objectives of the system.

The philosophy behind testing is to find the errors. A good test is one that has a high probability of finding an undiscovered error. A successful test is one that uncovers the undiscovered error. Test cases are devised with this purpose in mind. A test case is a set of data that the system will process as an input. However the data are created with the intent of determining whether the system will process them correctly without any errors to produce the required output.

4.2 SYSTEM TESTING

It is the stage of implementation, which ensures that system works accurately and effectively before the live operation commences. It is a confirmation that all are correct and opportunity to show users that the system must be tested with text data and show that the system will operate successfully and produce expected results under expected conditions

Software testing is a crucial element of software quality assurance and represents the unlimited review of specification, design and coding. Testing represents an interesting anomaly for the software. During the earlier definition and development phase, it was attempted to build the software from an abstract concept to tangible implementation

Testing Objectives:

- Testing is the process of analyzing program with the intent of discovering errors.
- A good test case is one that has high probability of finding undiscovered error.

Types of Testing:

- Unit testing
- Integration testing
- Validation testing
- Output testing
- User acceptance testing

4.2.1Unit Testing

All modules were tested and individually as soon as they were completed and were checked for their correct functionality.

4.2.2Integration Testing

The entire project was split into small program, each of these single programs gives a frame as an output. These programs were tested individually; at last all these programs where combined together by creating another program where all these constructors were used. It give a lot of problem by not functioning is an integrated manner.

The user interface testing is important since the user has to declare that the arrangements made in frames are convenient and it is satisfied. When the frames where given for the test, the end user gave suggestion. Based on their suggestions the frames where modified and put into practice.

4.2.3Validation Testing:

At the culmination of the black box testing software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of test i.e., Validation succeeds when the software function in a manner that can be reasonably accepted by the customer.

4.2.4Output Testing

After performing the validation testing the next step is output testing of the proposed system. Since the system cannot be useful if it does not produce the required output. Asking the user about the format in which the system is required tests the output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is on screen and another one is printed format. The output format on the screen is found to be corrected as the format was designed in the system phase according to the user needs. And for the hardcopy the output comes according to the specifications requested by the user.

ONCLUTION AND FUTUTR E ENHANCEMENT

5. CONCLUSION AND FUTURE ENHANCEMENT

The algorithms that we have developed are able to detect forgeries in both single and double JPEG compressed images with reasonable accuracy. Our test results show that the accuracy of detection improves with increase in Quality Factor. Intuitively one might expect the opposite trend, since blocking artifact increases with increasing compression. However, our method is not based on blocking artifact but rather on the metric that is calculated whose effectiveness decreases with higher compression. Hence, a combination of the new approach with the block wise distortion measure approach will significantly improve forgery detection by complementing each other's performance. Among the different kinds of forgeries that we worked on, cropping and rotation were the easiest to detect and copy-paste forgery was probably the toughest.

In future, different schemes for image forgery identification would be analyzed. However, these schemes mainly revolve around using the Quantization matrix only. Our test showed that this approach provides decent results but better methods can probably be devised in order to improve the accuracy.

APPENDIX

APPENDIX

7.1 SOURCE CODE

ImagePreviewPanel.java

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;

import javax.swing.BorderFactory;
import javax.swing.JPanel;
import javax.swing.border.Border;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EtchedBorder;

public class ImagePreviewPanel extends JPanel {
        private static final long serialVersionUID = -2984690692269464775L;
        private BufferedImage image;
        private int x, y, width, height;
        private double scale;
        private boolean scaleToSize;
        private boolean center;

        public ImagePreviewPanel(boolean scaleToSize, boolean center) {
                this.scaleToSize = scaleToSize;
        this.center = center;
        Border borderEtched = BorderFactory.createEtchedBorder(
                        EtchedBorder.RAISED, Color.blue, Color.cyan);
        setBorder(new CompoundBorder(borderEtched, borderEtched));

}

public void setImage(BufferedImage newImage) {
        if (newImage != image) {
                image = newImage;

                setRect(0, 0, 0, 0);
        } else {
                repaint();
        }
}

public void setRect(int x, int y, int width, int height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;

        repaint();
}

public static BufferedImage getScaledImage(BufferedImage originalImage,
                double scale) {
        AffineTransform at = new AffineTransform();
        at.setToScale(scale, scale);
        AffineTransformOp tx = new AffineTransformOp(at,
                        AffineTransformOp.TYPE_BILINEAR);
            BufferedImage newImage = tx.filter(originalImage, null);
            return newImage;
        }

        public void paint(Graphics g) {
            g.clearRect(0, 0, getWidth(), getHeight());
            if (image != null) {
                if (image.getWidth() < getSize().width
                                && image.getHeight() < getSize().height &&
scaleToSize) {
                        scale = 1;
                } else {
                        if (image.getWidth() < image.getHeight()) {
                                scale = (double) getSize().height
                                                / (double) image.getHeight();

                                if (image.getWidth() * scale > getSize().width) {
                                        scale *= (double) getSize().width
                                                        / ((double) image.getWidth()
* scale);
                                }
                        } else {
                                scale = (double) getSize().width
                                                / (double) image.getWidth();

                                if (image.getHeight() * scale > getSize().height) {
                                        scale *= (double) getSize().height
                                        / ((double) image.getHeight() * scale);
                                }
                        }
                }
                int xMargin = 0, yMargin = 0;
                if (center) {
                        xMargin = (getWidth() - (int) (image.getWidth() * scale)) /
2;
                        g.drawImage(getScaledImage(image, scale), xMargin, 0,
null);
                } else {
                        g.drawImage(getScaledImage(image, scale), 0, 0, null);
                }

                if (width != 0 && height != 0) {
                        Graphics2D g2d = (Graphics2D) g;
                        g2d.setPaint(Color.RED);
                        g2d.setStroke(new BasicStroke(3f));
                        if (center) {
                                g2d.draw(new Rectangle((int) (x * scale) +
xMargin,
                                                (int) (y * scale) + yMargin, (int)
(width * scale),
                                                (int) (height * scale)));
                        } else {
                                g2d.draw(new Rectangle((int) (x * scale),
                                                (int) (y * scale), (int) (width * scale),
                                                (int) (height * scale)));
                        }
                }
        }
    }
```

```
public class DigitalImageForensicsMain extends JFrame implements
ResultsListener,ActionListener {
        private static final long serialVersionUID = -3928892796166773 85L;
        static JDesktopPane deskView;
        private static Preferences preferences;
        private static String PREFERENCES_FILE_NAME = "preferences.dat";
        File imageFile = null;

        private JTabbedPaneWithCloseIcons mainTabbedPane;
        JInternalFrame internalFrame;
        public DigitalImageForensicsMain() {
                toInitialize();
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
                setSize(screenSize.width, screenSize.height - 30);
                setTitle("Digital Image Forensics");
                setJMenuBar(addMenubar());
                setContentPane(deskView);
                setVisible(true);
        }
        public void displayResults(ProgressPanel progressl,
                ShowResultsPanel resultsl, String titlel) {
                final ProgressPanel progress = progressl;
                final ShowResultsPanel results = resultsl;
                final String title = titlel;
                SwingUtilities.invokeLater(new Runnable() {
                        public void run() {
                                mainTabbedPane.remove(progress);
                                mainTabbedPane.add(title, results);
                                mainTabbedPane.setSelectedComponent(results);
                        }
```

```
			});

		}
		public void actionPerformed(ActionEvent ae) {
			if (ae.getActionCommand() == "Exit") {
				System.exit(0);
			}
			if (ae.getActionCommand() == "Upload Files") {
				deskView.add(new ImageToDatabase());
			}
			if (ae.getActionCommand() == "Image Process") {
				try {
					FileInputStream fis = new
FileInputStream(PREFERENCES_FILE_NAME);
					ObjectInputStream ois = new ObjectInputStream(fis);

					preferences = (Preferences) ois.readObject();
					ois.close();
				} catch (FileNotFoundException ex) {
					setPreferences(new Preferences());
				} catch (IOException ex) {
					ex.printStackTrace();
				} catch (ClassNotFoundException ex) {
					ex.printStackTrace();
				}

				showNewWizard();
			}
		}
		private void showNewWizard() {
			SetupWizard setupWizard = new SetupWizard(this, true, preferences);
```

```
            Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
            setupWizard.setLocation((dimension.width-570)/2, (dimension.height-
500)/2);

            setupWizard.setVisible(true);
            int returnStatus = setupWizard.getReturnStatus();

            if (returnStatus == SetupWizard.OK_OPTION) {
                    ProcessedImage chosenImage = new ProcessedImage(setupWizard
                                    .getSourceImage());
                    imageFile = setupWizard.getSourceImage();
                    String imageName = setupWizard.getSourceImage().getName();
                    boolean saveSettings = setupWizard.saveSettings();

                    if (setupWizard.getMode() == ImageProcessor.SIMILAR_IMAGE)
{
                            ProgressPanel similarImageProgress = new
ProgressPanel(this,
                                            chosenImage, imageName,
setupWizard.getSearchImages(),
                                            setupWizard.getSimilarSettings());
                            mainTabbedPane.addTab(imageName,
similarImageProgress);

        mainTabbedPane.setSelectedComponent(similarImageProgress);



        preferences.updatePreferences(setupWizard.getSimilarSettings(),
                                            setupWizard.getSourceImage(),
setupWizard
```

```
                                                            .getSearchFolder(),
saveSettings);

                        } else if (setupWizard.getMode() ==
ImageProcessor.WITHIN_IMAGE) {
                              ProcessedImage searchImage = new
ProcessedImage(setupWizard
                                          .getSearchImage());
                              ProgressPanel findInProgress = new ProgressPanel(this,
                                          chosenImage, imageName, searchImage,
setupWizard
                                                      .getWithinSettings());
                              mainTabbedPane.addTab(imageName, findInProgress);
                              mainTabbedPane.setSelectedComponent(findInProgress);


          preferences.updatePreferences(setupWizard.getWithinSettings(),
                                          setupWizard.getSourceImage(),
setupWizard
                                                      .getSearchFolder(),
saveSettings);

                        } else if (setupWizard.getMode() ==
ImageProcessor.SIMILAR_POSTERIZED_IMAGE) {
                              ProgressPanel similarPosterizedImageProgress = new
ProgressPanel(
                                          this, chosenImage, imageName,
setupWizard
                                                      .getSearchImages(),
setupWizard
```

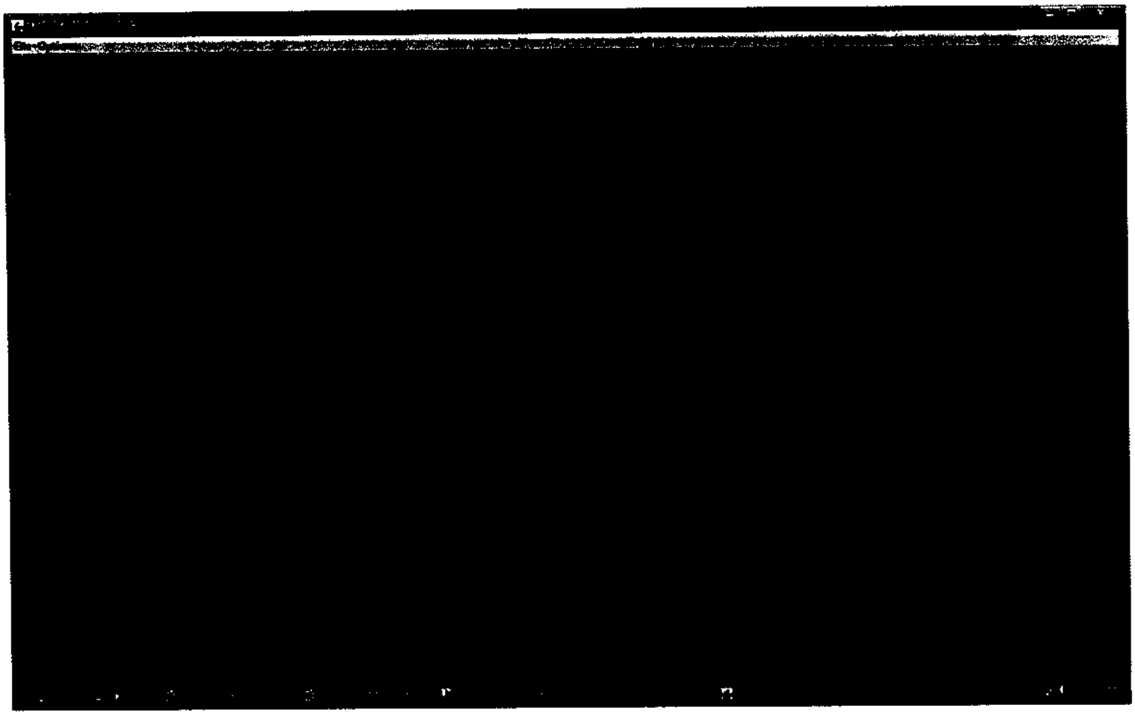
```
        .getSimilarPosterizedSettings());
                                mainTabbedPane
                                                .addTab(imageName,
similarPosterizedImageProgress);
                                mainTabbedPane

        .setSelectedComponent(similarPosterizedImageProgress);

                                preferences.updatePreferences(setupWizard
                                                .getSimilarPosterizedSettings(),
setupWizard
                                                .getSourceImage(),
setupWizard.getSearchFolder(),
                                                saveSettings);

                        } else if (setupWizard.getMode() ==
ImageProcessor.SIMILAR_POSTERIZED_BLOCK_IMAGE) {
                                ProgressPanel similarPBProgress = new
ProgressPanel(this,
                                                chosenImage, imageName,
setupWizard.getSearchImages(),
                                                setupWizard.getSimilarPBSettings());
                                mainTabbedPane.addTab(imageName, similarPBProgress);

        mainTabbedPane.setSelectedComponent(similarPBProgress);

                                preferences.updatePreferences(setupWizard
                                                .getSimilarPBSettings(),
setupWizard.getSourceImage(),
                                        setupWizard.getSearchFolder(),
saveSettings);

                    } else {
                          ProgressPanel findInDirProgress = new ProgressPanel(this,
                                        chosenImage, imageName,
setupWizard.getSearchImages(),
                                        setupWizard.getWithinSettings());
                          mainTabbedPane.addTab(imageName, findInDirProgress);

      mainTabbedPane.setSelectedComponent(findInDirProgress);


      preferences.updatePreferences(setupWizard.getWithinSettings(),
                                        setupWizard.getSourceImage(),
setupWizard
                                                    .getSearchFolder(),
saveSettings);

                    }

                    setPreferences(preferences);
                    internalFrame.add(mainTabbedPane);
                    internalFrame.setClosable(true);
                    internalFrame.setBounds((dimension.width-
1000)/2,(dimension.height-700)/2,1000,600);
                    internalFrame.setVisible(true);
                    deskView.add(internalFrame);


               }
```

```
	}

	private JMenuBar addMenubar() {
		JMenuBar menuBar = new JMenuBar();
		JMenu mnuFile, mnuOptions;
		JMenuItem mtmExit, mtmUpload, mtmProcess;

		mnuFile = new JMenu("File");
		mtmExit = new JMenuItem("Exit", 'x');
		mnuFile.add(mtmExit);
		menuBar.add(mnuFile);

		mnuOptions = new JMenu("Options");
		mtmUpload = new JMenuItem("Upload Files", 'u');
		mtmProcess = new JMenuItem("Image Process", 'p');
		mnuOptions.add(mtmUpload);
		mnuOptions.add(mtmProcess);
		menuBar.add(mnuOptions);

		mtmExit.addActionListener(this);
		mtmUpload.addActionListener(this);
		mtmProcess.addActionListener(this);

		return menuBar;
	}
	public static void setPreferences(Preferences newPreferences) {
		preferences = newPreferences;

		try {
			FileOutputStream fos = new
FileOutputStream(PREFERENCES_FILE_NAME);
```

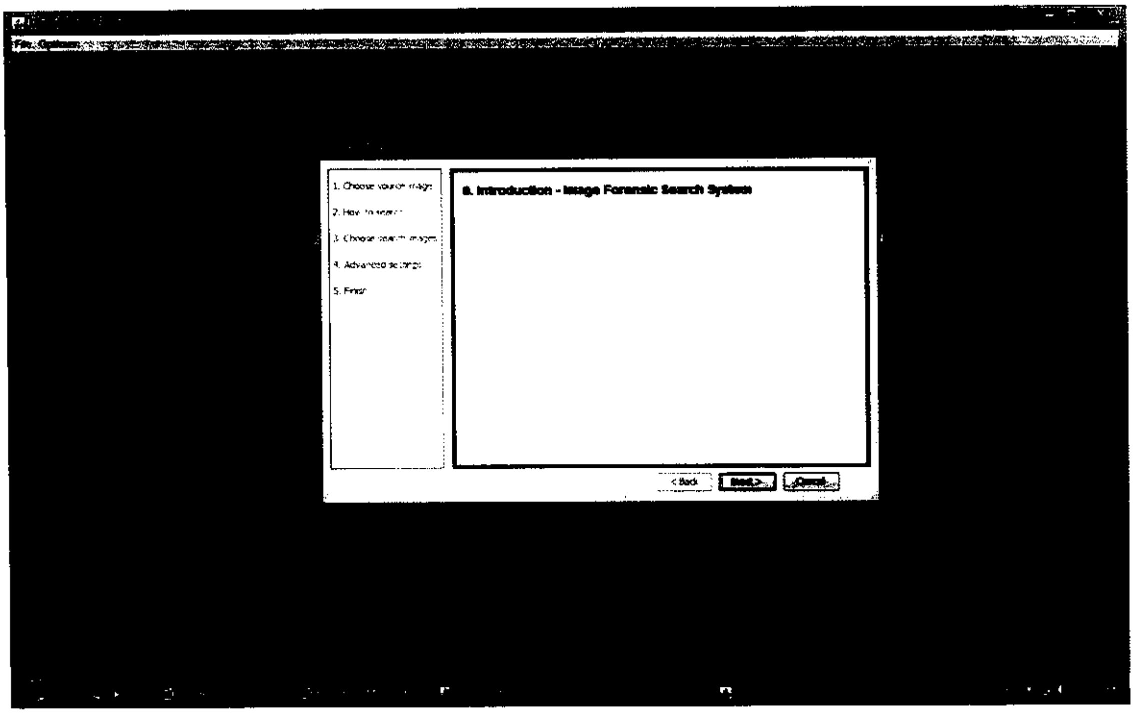
```
                ObjectOutputStream oos = new ObjectOutputStream(fos);
                oos.writeObject(preferences);
                oos.close();
            } catch (FileNotFoundException ex) {
                ex.printStackTrace();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
    }

    public static Preferences getPreferences() {
            return preferences;
    }



    private void toInitialize() {
            try {

    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
            } catch (Exception e) {
                e.printStackTrace();
            }
            internalFrame = new JInternalFrame();
            deskView = new JDesktopPane();
            mainTabbedPane = new JTabbedPaneWithCloseIcons();
    }
    public static void main(String[] args) {
            new DigitalImageForensicsMain();
    }

}
```

7.2SCREEN SHOTS:

MAIN FORM:

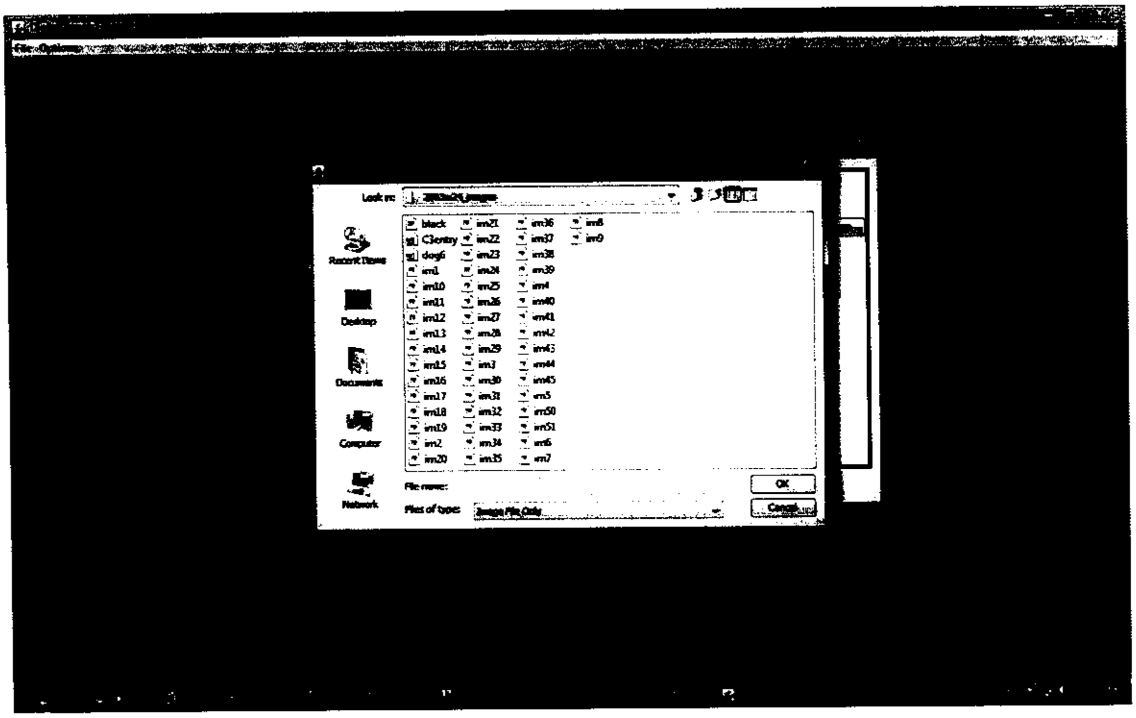


SEARCH PANEL



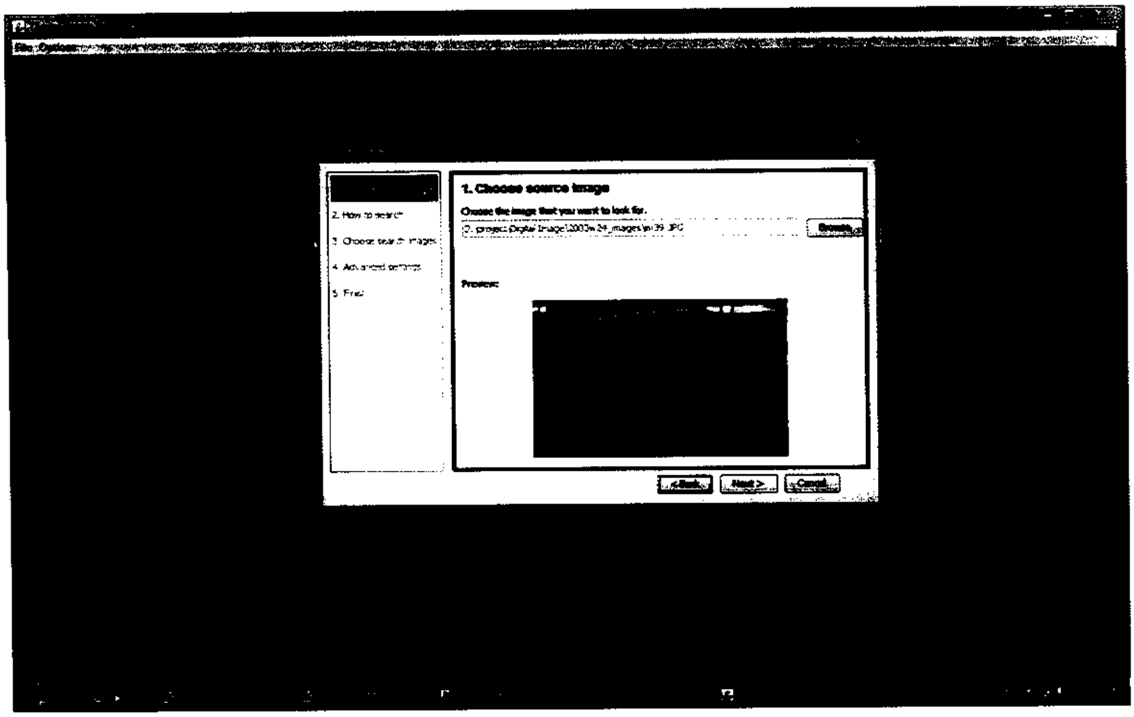
0. Introduction - Image Forensic Search System

INPUT SELECTION PANEL

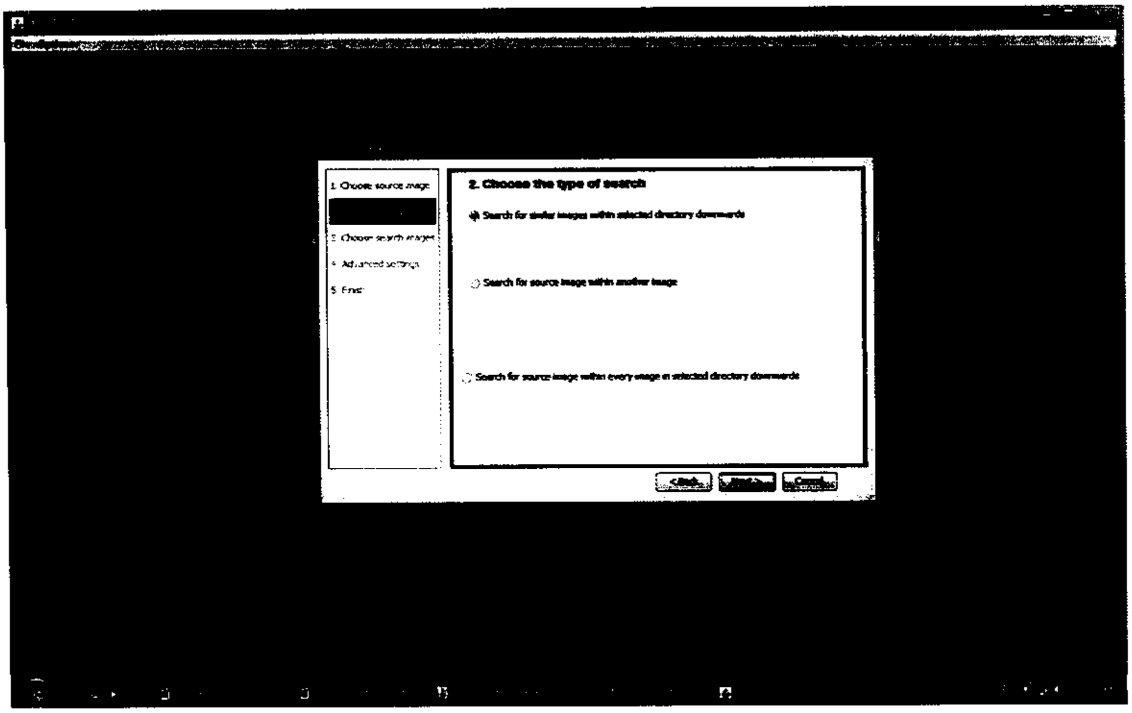


```
black    im21   im36   im8
C3entry  im22   im37   im9
dog6     im23   im38
im1      im24   im39
im10     im25   im4
im11     im26   im40
im12     im27   im41
im13     im28   im42
im14     im29   im43
im15     im3    im44
im16     im30   im45
im17     im31   im5
im18     im32   im50
im19     im33   im51
im2      im34   im6
im20     im35   im7
```

INPUT PANEL:


File Options
2. How to search
3. Choose search images
4. Advanced settings
5. Final
1. Choose source image
Choose the image that you want to look for.
D:\project\Digital Image\2003w24_images\im39.JPG
Browse
Preview:
< Back
Next >
Cancel

TYPE OF SEARCH CHOOSING PANEL:



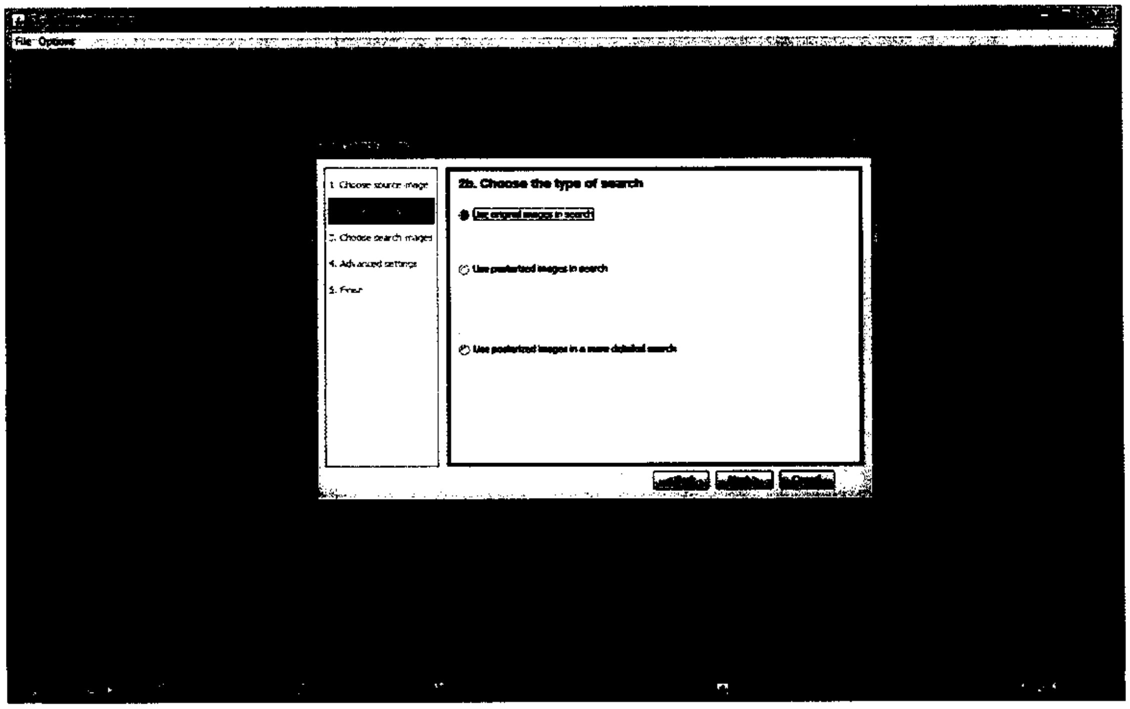
1. Choose source image
3. Choose search images
4. Advanced settings
5. Finish

2. Choose the type of search

[✓ Search for similar images within selected directory downwards]

Search for source image within another image

Search for source image within every image in selected directory downwards



1. Choose source image
3. Choose search images
4. Advanced settings
5. Finish

2b. Choose the type of search

[✓ Use original images in search]

Use posterized images in search

Use posterized images in a more detailed search

SOURCE IMAGES SELECTING PANEL:

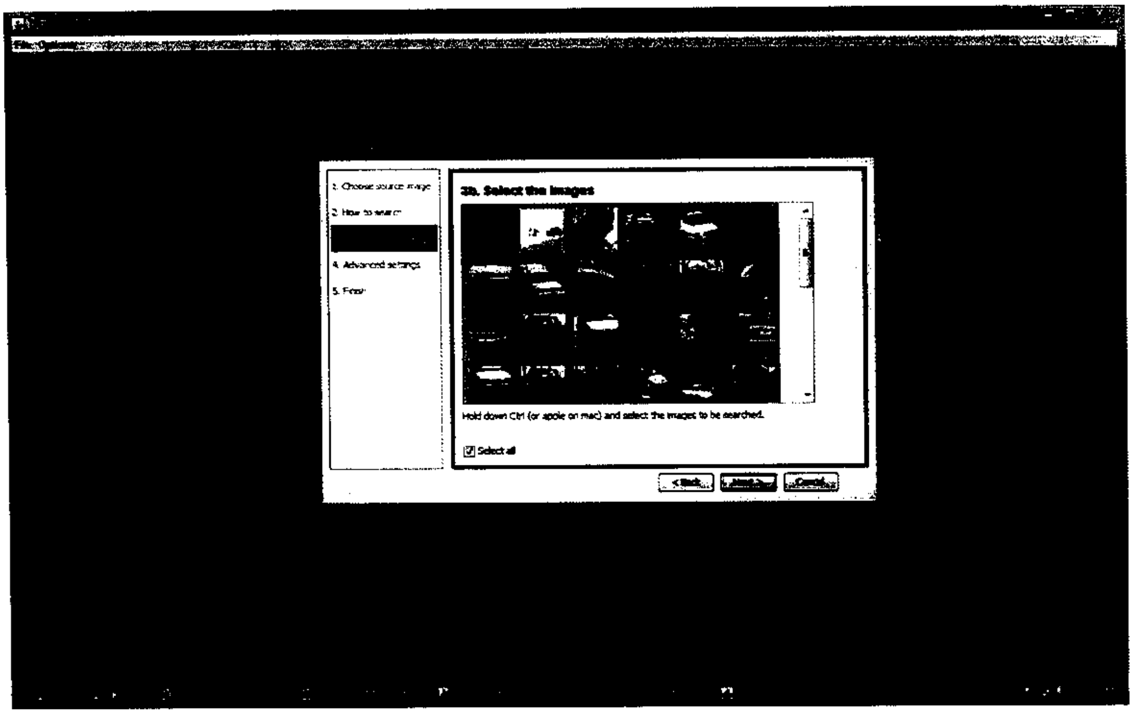
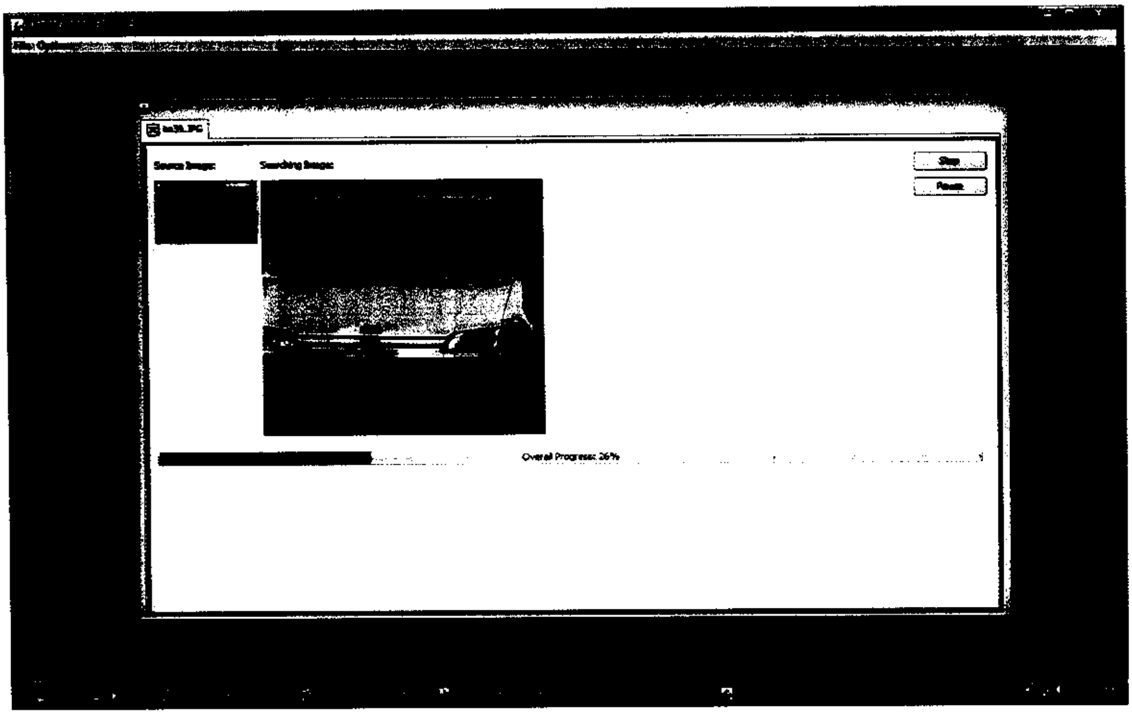
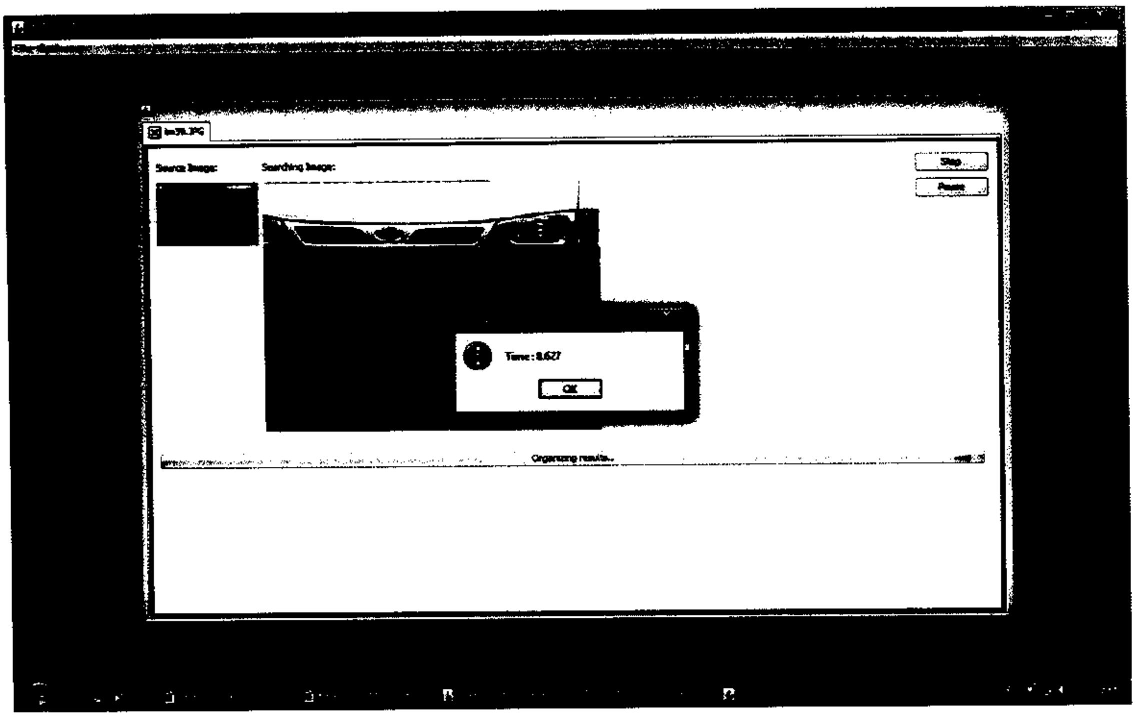


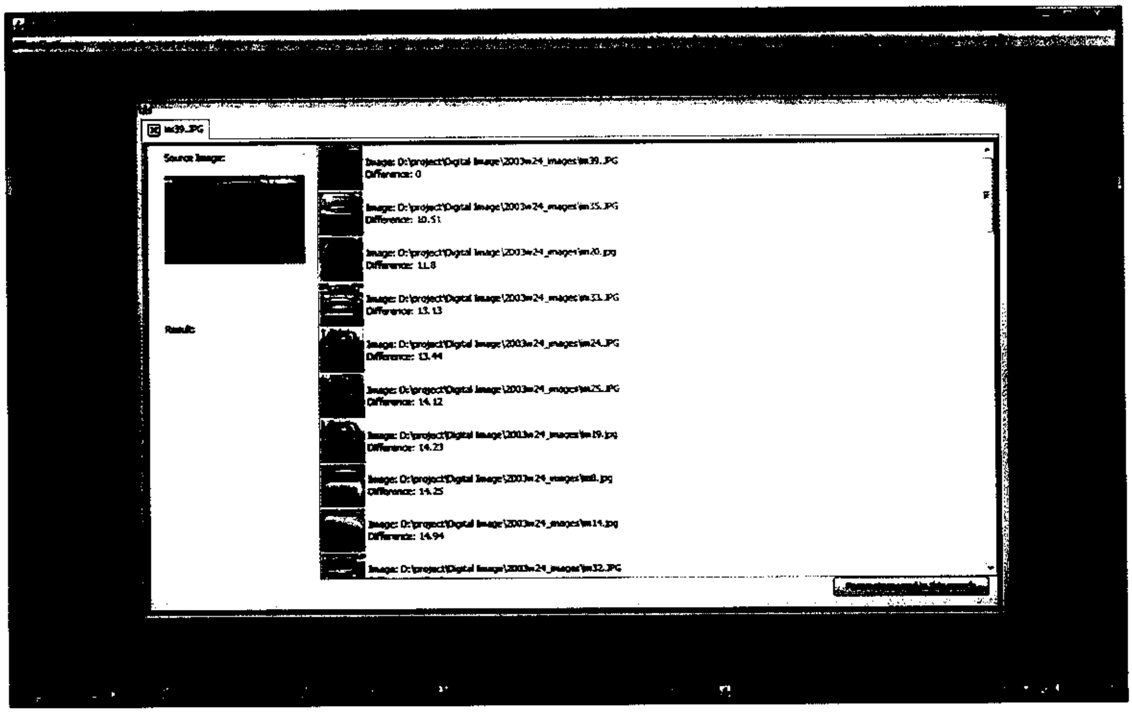
IMAGE PROCESSING PANEL:



PANEL SHOWING TOTAL TIME TAKEN TO PROCESS:



RESULT PANEL:



```
Image: D:\project\Digital Image\2003w24_images\im39.JPG
Difference: 0
Image: D:\project\Digital Image\2003w24_images\im35.JPG
Difference: 10.51
Image: D:\project\Digital Image\2003w24_images\im20.jpg
Difference: 11.8
Image: D:\project\Digital Image\2003w24_images\im33.JPG
Difference: 13.13
Image: D:\project\Digital Image\2003w24_images\im24.JPG
Difference: 13.44
Image: D:\project\Digital Image\2003w24_images\im25.JPG
Difference: 14.12
Image: D:\project\Digital Image\2003w24_images\im19.jpg
Difference: 14.23
Image: D:\project\Digital Image\2003w24_images\im8.jpg
Difference: 14.25
Image: D:\project\Digital Image\2003w24_images\im14.jpg
Difference: 14.94
Image: D:\project\Digital Image\2003w24_images\im32.JPG
```

REFERENCES

REFERENCES

1. Digital Image Forensics via Intrinsic Fingerprints Ashwin Swaminathan, *Student Member, IEEE,* Min Wu, *Senior Member, IEEE,* and K. J. Ray Liu, *Fellow, IEEE 2008.*
2. Image Forgery Identification Using JPEG Intrinsic Fingerprints A. Garg, A. Hailu, and R. Sridharan.
3. Scott Oaks, Henry Wong, Mike Loukides (Editor), "Java Threads Java Series", O'Reilly & Associates.
4. Patrick Naughton, Herbert Schildt, "Java™ 2: The Complete Reference", Third Edition, Tata McGraw-Hill Publishing Company Limited.
5. David Flanagan, "Java in a Nutshell", 2nd Edition, May 1997.
6. Mark Grand and Jonathan Knudsen, "Java Fundamental Classes Reference", Tata McGraw-Hill Publishing Company Limited, 1st Edition, May 1997.
7. Dietel & Deitel, Java How To Program, BPB Publishers, New Delhi, 2000.
8. O'reilly, Java Swings, Tata McGraw Hill, Fifth Edition, 2002.
9. Broukshickle, Thinking in Java, Calgotia Publishers, New Delhi, 2001
10. Ian Darwin "Java Cookbook", First Edition O'reilly June 2001
11. David Flanagan, "Java in a Nutshell", 2nd Edition, May 1997.
12. Elias M awar, "System Analysis and Design ", Second Edition Galgotia publication 1996

http://www.google.com

http://www.java2s.com

http://www.netbeans.org

http://www.roseindia.net

http://sun.java.com

http://www.jexamples.com

http://www.javadb.com/

http://www.jsresources.org/examples/

http://www.esus.com/

http://www.javadocexamples.com/