Image Forensic for Digital Image Copy Move Forgery Detection

1 INTRODUCTION

In this day and age, digital image tampering has been made easy with widely available image editing softwares, such as Adobe Photoshop. The advancement of image editing softwares has reached a level such that image tampering can be done without degrading its quality or leaving obvious traces. This is alarming as images are now being presented as supporting evidence and historical records in various fields, such as in forensic investigation, law enforcement, journalistic photography and medical images. Moreover, in many instances, tampered images have appeared in the news or social media, such as the manipulated images of Iranian missile test released on July 9, 2008 by Sepah News, the official media arm of Iran's Revolutionary Guard. The tampered image, shown in Fig. 1 is aimed at exaggerating the country's military capabilities. The forgery is detected a day later when the same source released another image taken from the same angle at almost the same time, but with different content. The scientific community is also not spared from image tampering. Farid et al. stated that 20% of accepted manuscripts of the Journal of Cell Biology contains inappropriate figure manipulation. Hence, image tampering and detection have garnered substantial attention as manipulated images can be used to misrepresent their meaning with malicious intent.

Objective of the project:

In recent years, digital image forgery detection has become an active research area due to the advancement of photo editing software. This paper focuses on passive forgery detection on images tampered using copy move technique, better known as Copy Move Forgery Detection (CMFD). A CMFD technique consisting of oriented Features from Accelerated Segment Test and rotated Binary Robust Independent Elementary Features (Oriented FAST and rotated BRIEF) as the feature extraction method and 2 Nearest Neighbour (2NN) with Hierarchical Agglomerative Clustering (HAC) as the feature matching method is proposed. Evaluation of the proposed CMFD technique was performed on images that underwent various geometrical attacks. With the proposed technique, an overall accuracy rate of 84.33% and 82.79% are obtained for evaluation carried out with images from the MICC-F600 and MICC-F2000 databases. Forgery detection achieved True Positive Rate of more than 91% for tampered images with object translation, different degree of rotation and enlargement.

2. LITERATURE SURVEY

Exposing digital forgeries from JPEG ghosts

When creating a digital forgery, it is often necessary to combine several images, for example, when compositing one person's head onto another person's body. If these images were originally of different JPEG compression quality, then the digital composite may contain a trace of the original compression qualities. To this end, we describe a technique to detect whether the part of an image was initially compressed at a lower quality than the rest of the image. This approach is applicable to images of high and low quality as well as resolution.

A SIFT-based forensic method for copymove attack detection and transformation recovery

One of the principal problems in image forensics is determining if a particular image is authentic or not. This can be a crucial task when images are used as basic evidence to influence judgment like, for example, in a court of law. To carry out such forensic analysis, various technological instruments have been developed in the literature. In this paper, the problem of detecting if an image has been forged is investigated; in particular, attention has been paid to the case in which an area of an image is copied and then pasted onto another zone to create a duplication or to cancel something that was awkward. Generally, to adapt the image patch to the new context a geometric transformation is needed. To detect such modifications, a novel methodology based on scale invariant features transform (SIFT) is proposed. Such a method allows us to both understand if a copy-move attack has occurred and, furthermore, to recover the geometric transformation used to perform cloning. Extensive experimental results are presented to confirm that the technique is able to precisely individuate the altered area and, in addition, to estimate the geometric transformation parameters with high reliability. The method also deals with multiple cloning.

Detection of Copy-Move Forgery in Digital Images

Due to the powerful image editing tools images are open to several manipulations; therefore, their authenticity is becoming questionable especially when images have influential power, for example, in a court of law, news reports, and insurance claims. Image forensic techniques determine the integrity of images by applying various high-tech mechanisms developed in the literature. In this paper, the images are analyzed for a particular type of forgery where a region of an image is copied and pasted onto the same image to create a duplication or to conceal some existing objects. To detect the copy-move forgery attack, images are first divided into overlapping square blocks and DCT components are adopted as the block representations. Due to the high dimensional nature of the feature space, Gaussian RBF kernel PCA is applied to achieve the reduced dimensional feature vector representation that also improved the efficiency during the feature matching. Extensive experiments are performed to evaluate the proposed method in comparison to state of the art. The experimental results reveal that the proposed technique precisely determines the copy-move forgery even when the images are contaminated with blurring, noise, and compression and can effectively detect multiple copy-move forgeries. Hence, the proposed technique provides a computationally efficient and reliable way of copy-move forgery detection that increases the credibility of images in evidence centered applications.

Exposing digital forgeries by detecting duplicated image regions

We describe an efficient technique that automatically detects duplicated regions in a digital image. This technique works by first applying a principal component analysis to small fixedsize image blocks to yield a reduced dimension representation. This representation is robust to minor variations in the image due to additive noise or lossy compression. Duplicated regions are then detected by lexicographically sorting all of the image blocks. We show the efficacy of this technique on credible forgeries, and quantify its robustness and sensitivity to additive noise and lossy JPEG compression.

Detection of duplication forgery in digital images in uniform and non-uniform regions

In this paper we propose a robust and fully automatic method to detect duplicated regions in digital images. Copied areas in uniform and non-uniform regions are detectable in our method. There are several methods to make forged images, but the most common is copy-move forgery, that the forger copies a part(s) of image and pasted it into another part(s) of that image. Many researchers have done beneficial researches on it. Most of them can find only copy-moved forgery. In other words, they can find regions which are only copied and pasted without any changes, but failed to find copied regions with scaling or rotating before pasting. Many forgers make some changes on copied regions so that the image sounds more natural. SIFT features can find forged regions even if they are rotated or scaled. This method has some other advantages, but failed to find flat copied regions. Zernike moments are invariant against rotation. They can find flat copied regions too, but sensitive to scaling. So it is clear that using these two features is very proper to detect all copied regions in an image. By applying SIFT detection method on overall the image and then Zernike moments detection method on regions where SIFT features have not been found, The processing time is reduced in comparison to applying each of these two methods on entire image.

Speeded-Up Robust Features (SURF)

This article presents a novel scale- and rotation-invariant detector and descriptor, coined SURF (Speeded-Up Robust Features). SURF approximates or even outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster. This is achieved by relying on integral images for image convolutions; by building on the strengths of the leading existing detectors and descriptors (specifically, using a Hessian matrix-based measure for the detector, and a distribution-based descriptor); and by simplifying these methods to the essential. This leads to a combination of novel detection, description, and matching steps. The paper encompasses a detailed description of the detector and descriptor and then explores the effects of the most important parameters. We conclude the article with SURF's application to two challenging, yet converse goals: camera calibration as a special case of image registration, and object recognition. Our experiments underline SURF's usefulness in a broad range of topics in computer vision.

Image copy-move forgery detection based on SURF

As the advent and growing popularity of image editing software, digital images can be manipulated easily without leaving obvious visual clues. If the tampered images are abused, it may lead to potential social, legal or private consequences. To this end, it's very necessary and also challenging to find effective methods to detect digital image forgeries. In this paper, a fast method to detect image copy-move forgery is proposed based on the SURF (Speed up Robust Features) descriptors, which are invariant to rotation, scaling etc. Results of experiments indicate that the proposed method is valid in detecting the image region duplication and quite robust to additive noise and blurring.

Region Duplication Forgery Detection Technique Based on SURF and HAC

Region duplication forgery detection is a special type of forgery detection approach and widely used research topic under digital image forensics. In copy move forgery, a specific area is copied and then pasted into any other region of the image. Due to the availability of sophisticated image processing tools, it becomes very hard to detect forgery with naked eyes. From the forged region of an image no visual clues are often detected. For making the tampering more robust, various transformations like scaling, rotation, illumination changes, JPEG compression, noise addition, gamma correction, and blurring are applied. So there is a need for a method which performs efficiently in the presence of all such attacks. This paper presents a detection method based on speeded up robust features (SURF) and hierarchical agglomerative clustering (HAC). SURF detects the keypoints and their corresponding features. From these sets of keypoints, grouping is performed on the matched keypoints by HAC that shows copied and pasted regions.

.

ORB: An efficient alternative to SIFT or SURF

Feature matching is at the base of many computer vision problems, such as object recognition or structure from motion. Current methods rely on costly descriptors for detection and matching. In this paper, we propose a very fast binary descriptor based on BRIEF, called ORB, which is rotation invariant and resistant to noise. We demonstrate through experiments how ORB is at two orders of magnitude faster than SIFT, while performing as well in many situations. The efficiency is tested on several real-world applications, including object detection and patch-tracking on a smart phone.

Measuring Corner Properties

We describe methods to measure the following properties of gray level corners: subtended angle, orientation, contrast, bluntness (or rounding of the apex), and boundary curvature (for cusps). Unlike most of the published methods for extracting these properties these new methods are relatively simple, efficient, and robust. They rely on the corner being predetected by a standard operator, thus making the measurement problem more tractable. Using 13,000 synthetic images the methods are assessed over a range of conditions: corners of varying orientations and subtended angles, as well as different degrees of noise.

3 .SYSTEM ANALYSIS

3.1 Existing System

A comparison is carried out with the work by Kaur and Kaur in 2016 where ORB and SVM are used as the feature extraction and feature matching method respectively. The performance of the existing work is evaluated with images from the MICC-F600 database.

Disadvantages

1.Less accuracy

3.2 PROPOSED SYSTEM

Image pre-processing is generally performed to reduce the amount of redundant information in an image and to improve the computational efficiency in the following CMFD stages. In our work, the pre-processing operations consist of image RGB to gray scale conversion, image resizing and tampered region identification. In this work, a CMFD technique consisting of oriented FAST and rotated BRIEF (ORB) as the feature extraction method and 2NN with HAC as the feature matching method is proposed

Advantages

1.High accuracy

Modules:

In propose algorithm author has used following modules

Image acquiring: using this module we will read all images from dataset

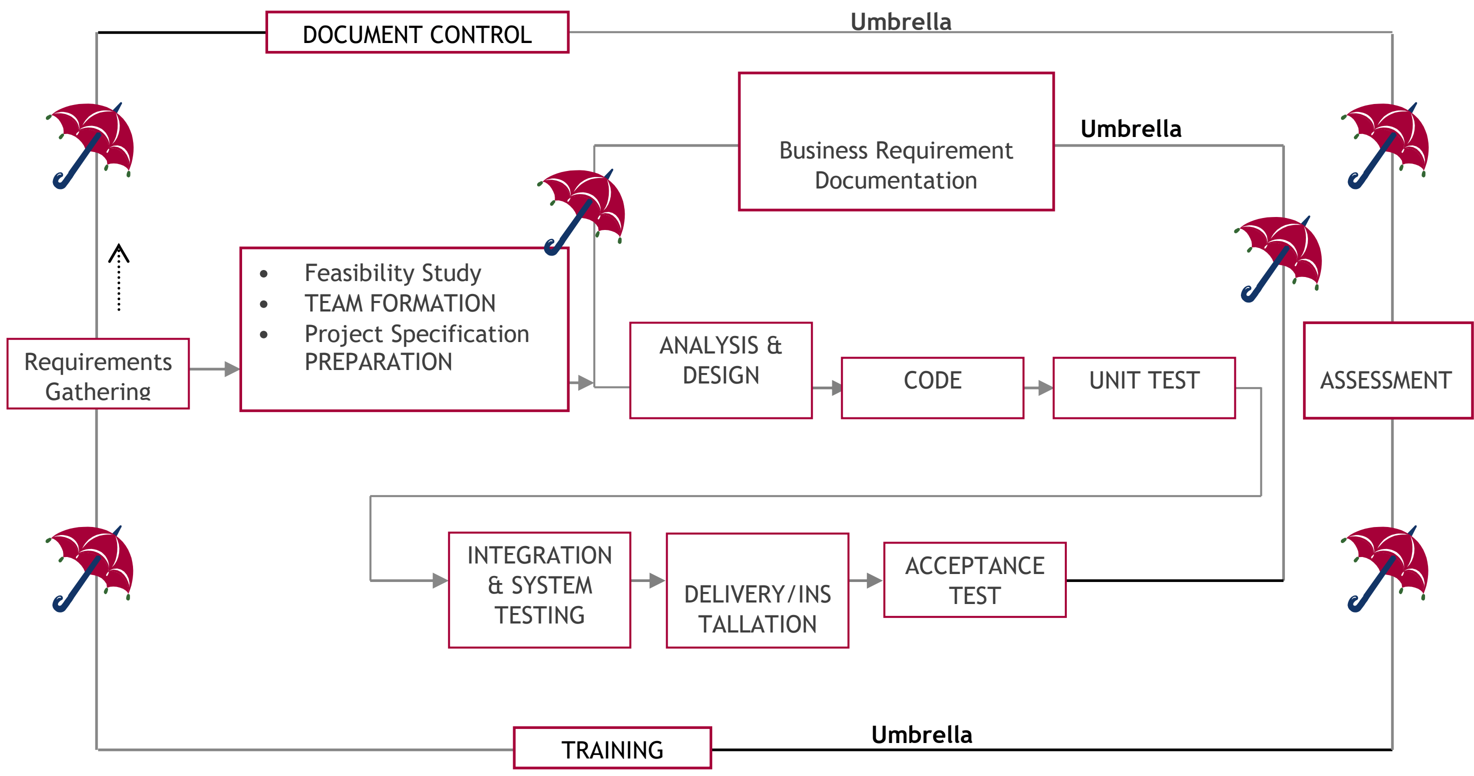
Image Preprocessing: converting RGB image to grey format

Extracting keypoints and descriptor: using ORB we will extract keypoints and descriptor

Feature Matching: using 2NN (nearest neighbours) we will find matching between images by using descriptors and then plot match descriptors by using keypoints. If there is much similarity then its accuracy will increase and if not much similarity then false positive will increase

3.3. PROCESS MODEL USED WITH JUSTIFICATION

SDLC (Umbrella Model):



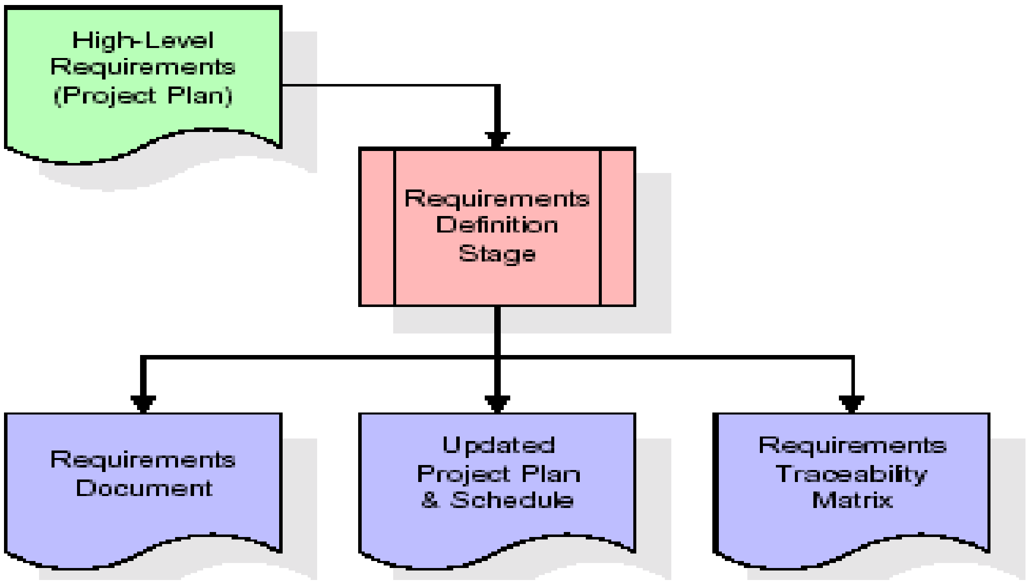
SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

<u>Stages in SDLC</u>:

- Requirement Gathering
- Analysis
- Designing
- Coding
- Testing
- Maintenance

<u>Requirements Gathering stage</u>:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.


High-Level Requirements (Project Plan)
Requirements Definition Stage
Requirements Document
Updated Project Plan & Schedule
Requirements Traceability Matrix

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

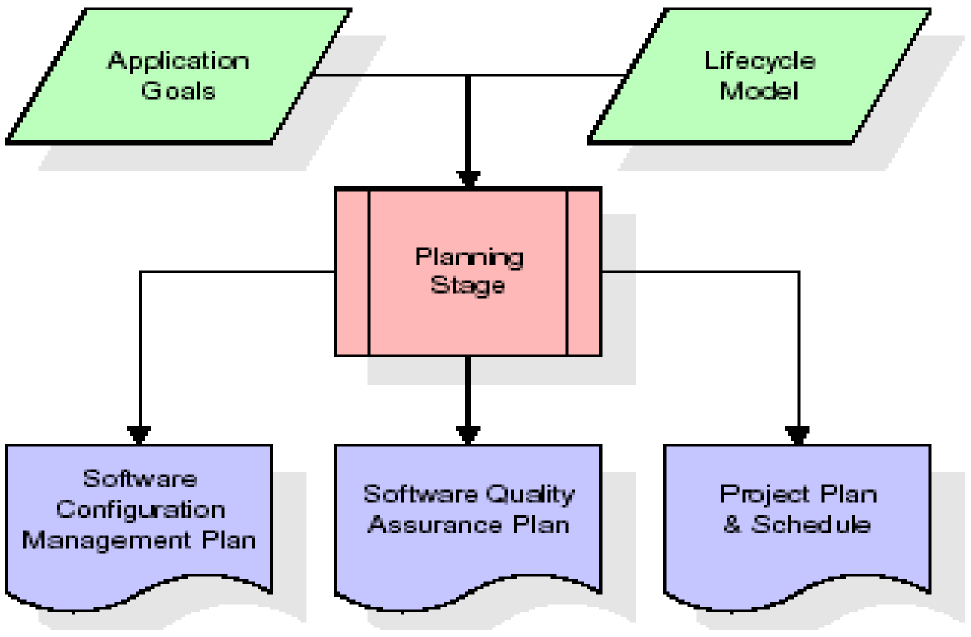
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

Analysis Stage:

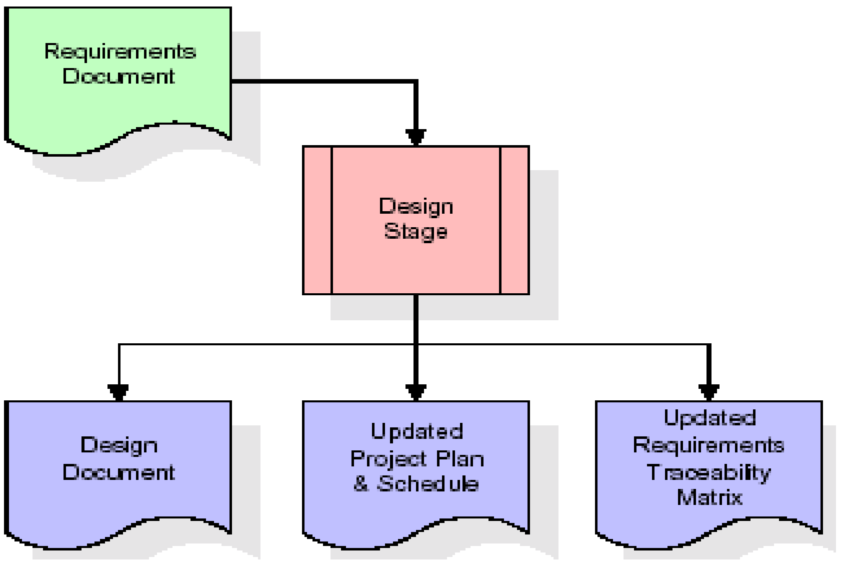
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.


Application Goals
Lifecycle Model
Planning Stage
Software Configuration Management Plan
Software Quality Assurance Plan
Project Plan & Schedule

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

<u>Designing Stage</u>:

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

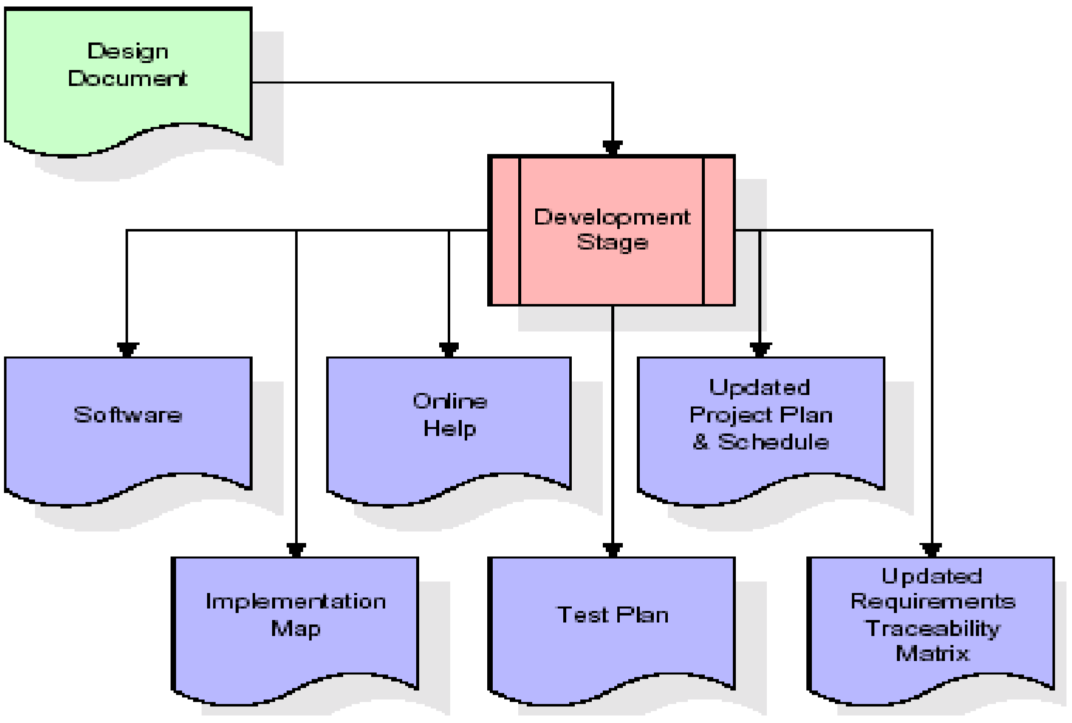

Requirements Document
Design Stage
Design Document
Updated Project Plan & Schedule
Updated Requirements Traceabilty Matrix

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

<u>Development (Coding) Stage</u>:

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions.

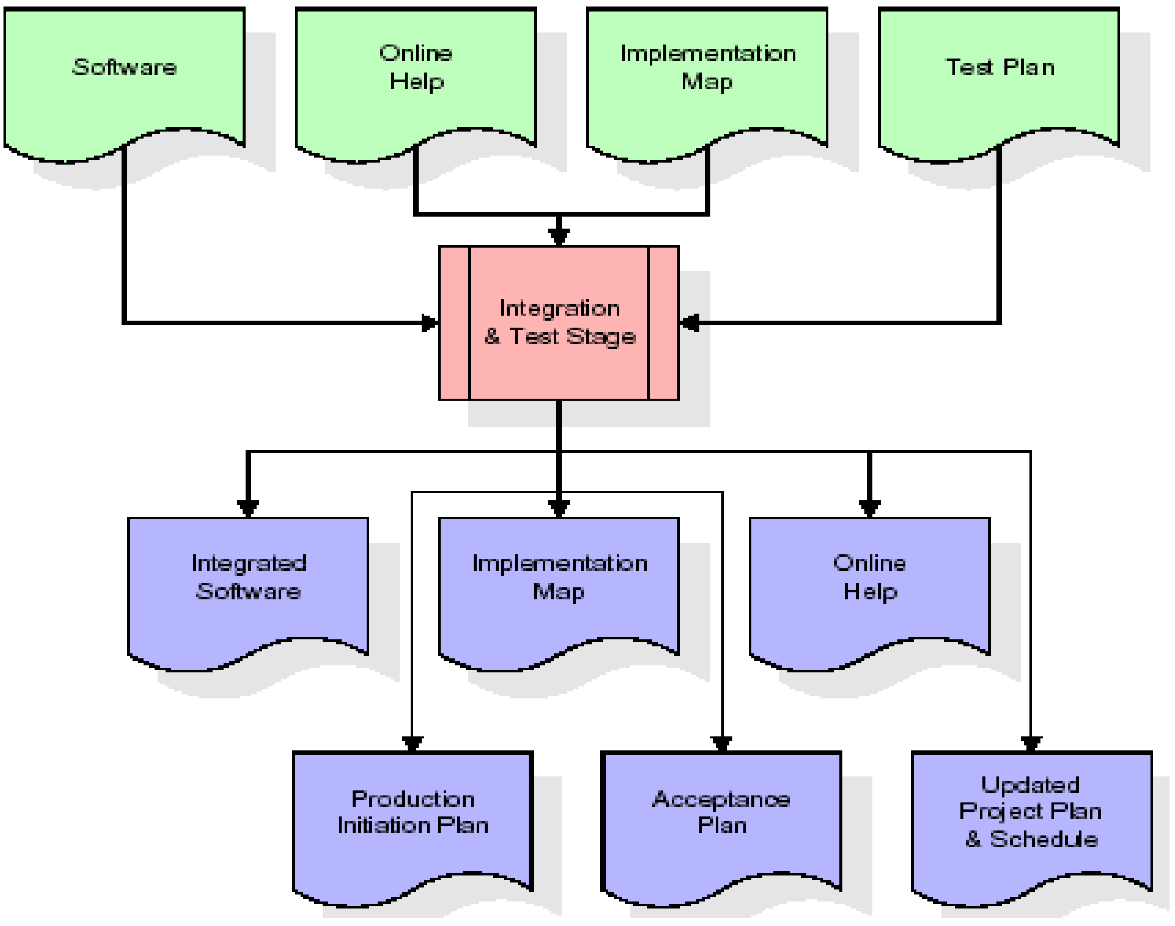
Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

Integration & Test Stage:

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

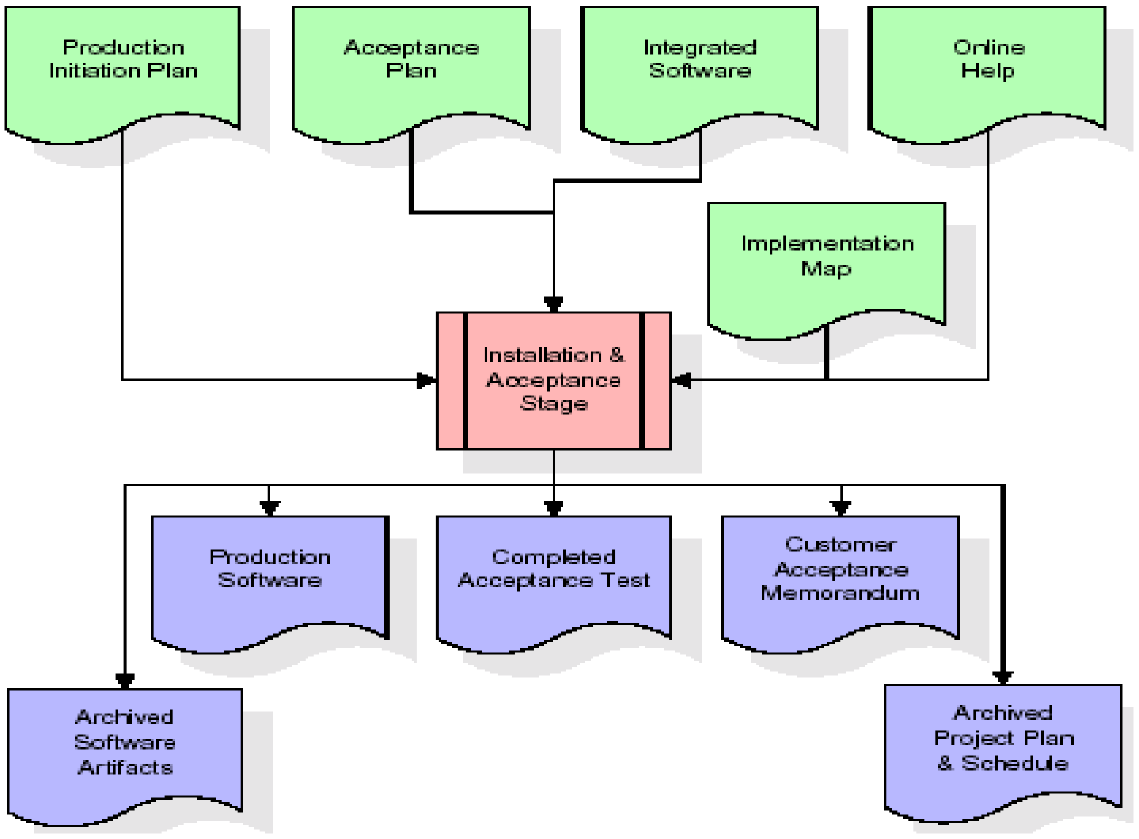


The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

♦ **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artefacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.


Production Initiation Plan
Acceptance Plan
Integrated Software
Online Help
Implementation Map
Installation & Acceptance Stage
Production Software
Completed Acceptance Test
Customer Acceptance Memorandum
Archived Software Artifacts
Archived Project Plan & Schedule

The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labour data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

<u>Maintenance</u>:

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

3.4. Software Requirement Specification

3.4.1. Overall Description

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- Business requirements describe in business terms *what* must be delivered or accomplished to provide value.
- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

- **OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

- **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

3.4.2. External Interface Requirements

User Interface

The user interface of this system is a user friendly python Graphical User Interface.

Hardware Interfaces

The interaction between the user and the console is achieved through python capabilities.

Software Interfaces

The required software is python.

Operating Environment

Windows XP.

HARDWARE REQUIREMENTS:

• Processor	-	Pentium –IV
• Speed	-	1.1 Ghz
• RAM	-	256 MB(min)
• Hard Disk	-	20 GB
• Key Board	-	Standard Windows Keyboard
• Mouse	-	Two or Three Button Mouse
• Monitor	-	SVGA

SOFTWARE REQUIREMENTS:

• Operating System	-	Windows7/8
• Programming Language	-	Python

4. SYSTEM DESIGN

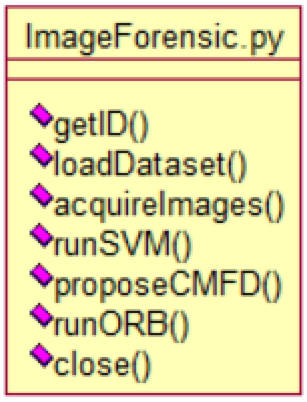
UML Diagram:

Class Diagram:

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

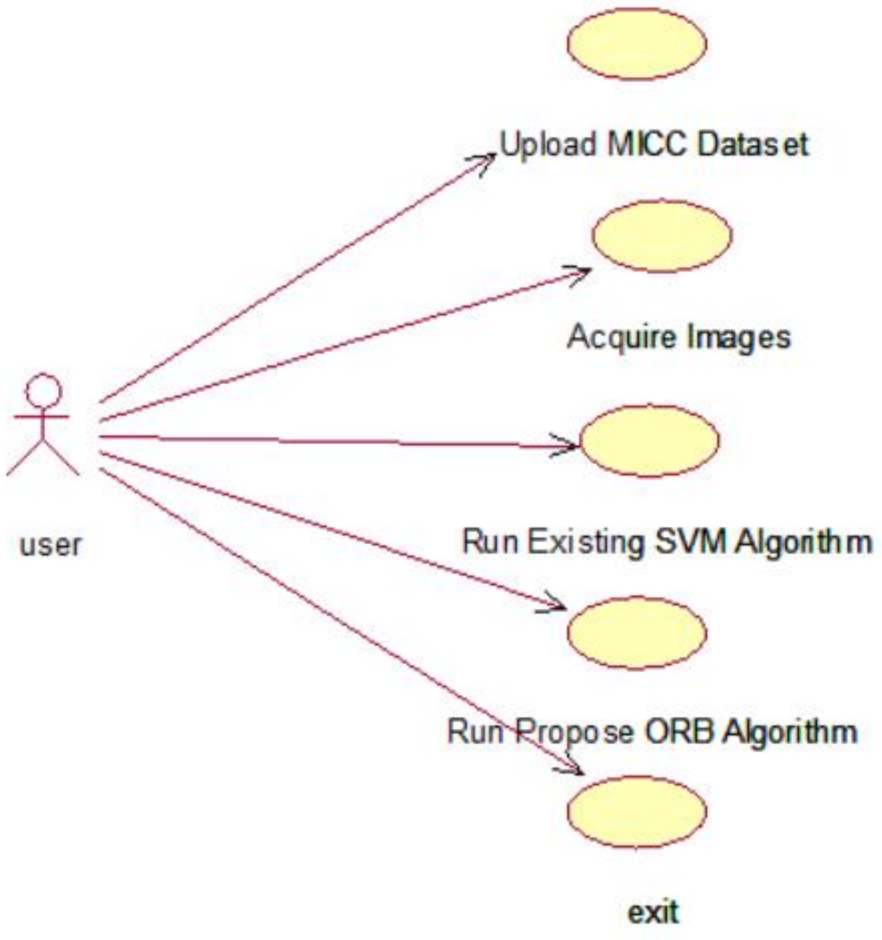
Class Diagram:



Use case Diagram:

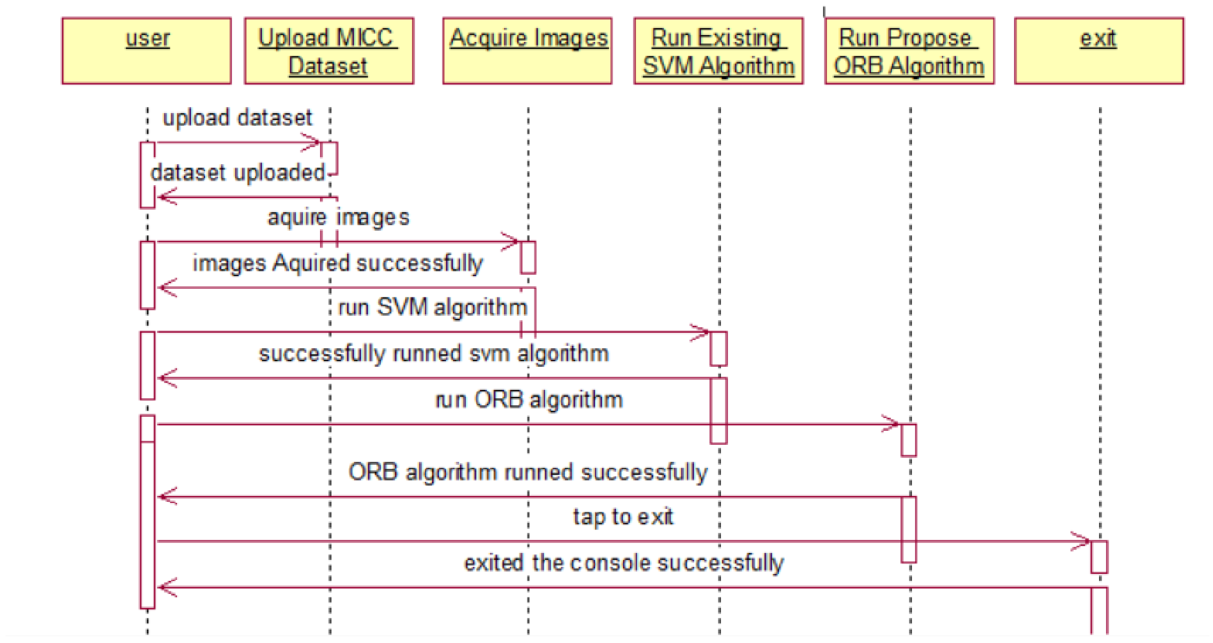
A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

Use case Diagram:



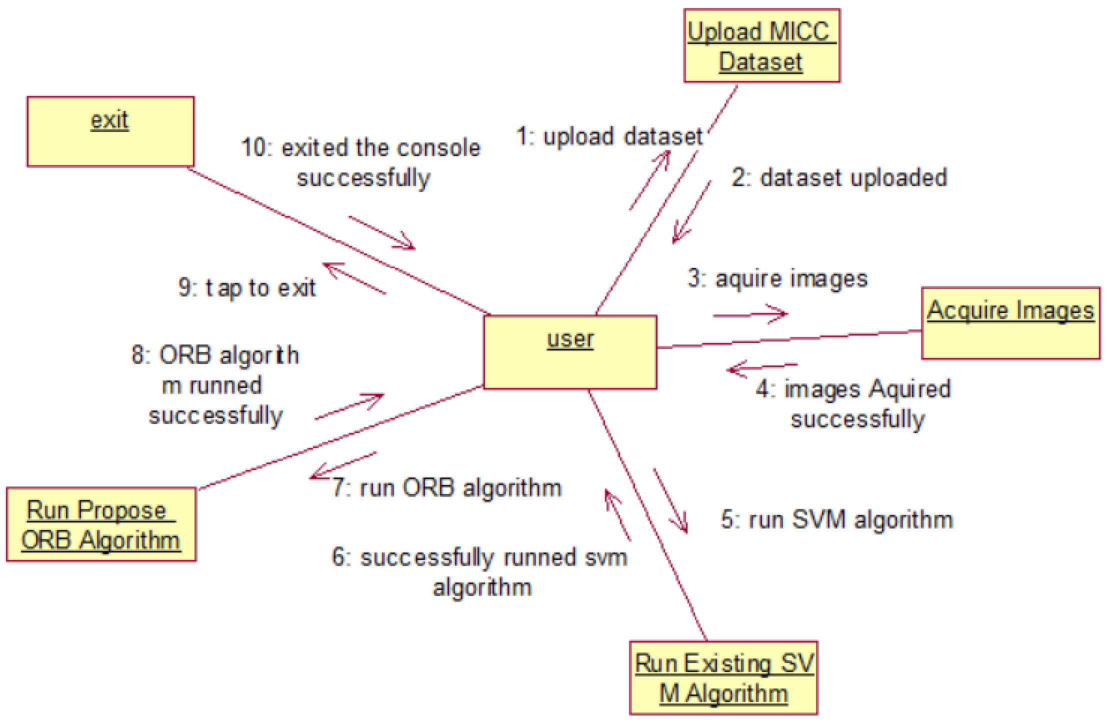
Sequence diagram:

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.


user
Upload MICC Dataset
Acquire Images
Run Existing SVM Algorithm
Run Propose ORB Algorithm
exit
upload dataset
dataset uploaded
aquire images
images Aquired successfully
run SVM algorithm
successfully runned svm algorithm
run ORB algorithm
ORB algorithm runned successfully
tap to exit
exited the console successfully

Collaboration diagram:

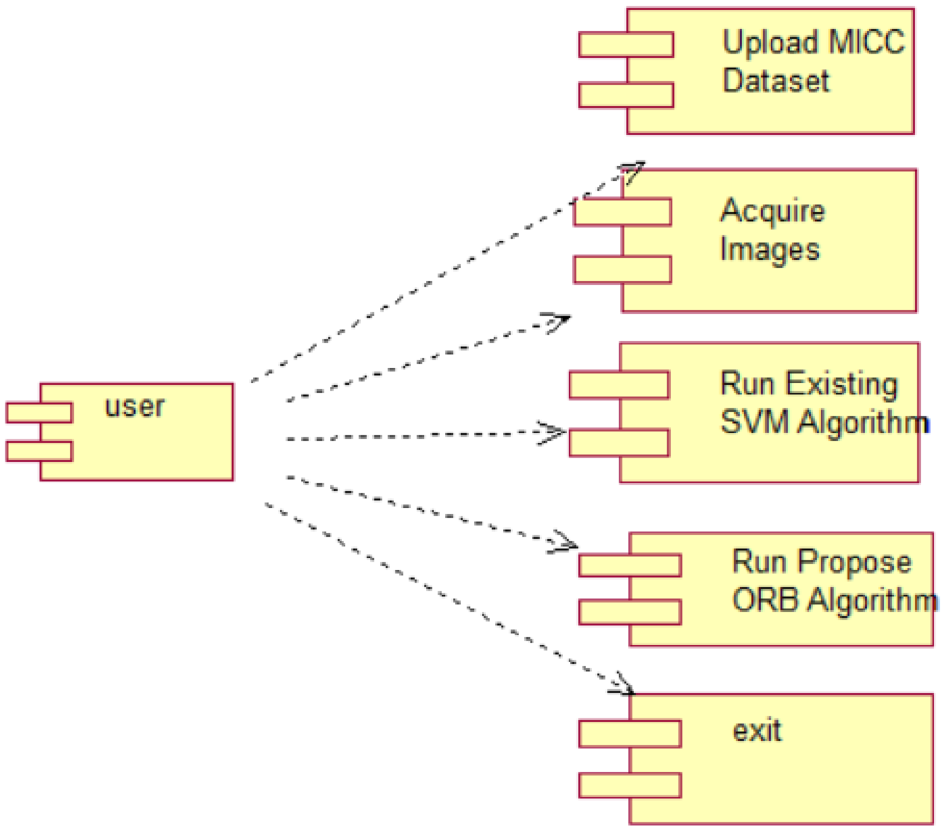
A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.


Upload MICC Dataset
exit
10: exited the console successfully
1: upload dataset
2: dataset uploaded
9: tap to exit
3: aquire images
Acquire Images
user
8: ORB algorith m runned successfully
4: images Aquired successfully
Run Propose ORB Algorithm
7: run ORB algorithm
5: run SVM algorithm
6: successfully runned svm algorithm
Run Existing SVM Algorithm

<u>**Component Diagram:**</u>

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

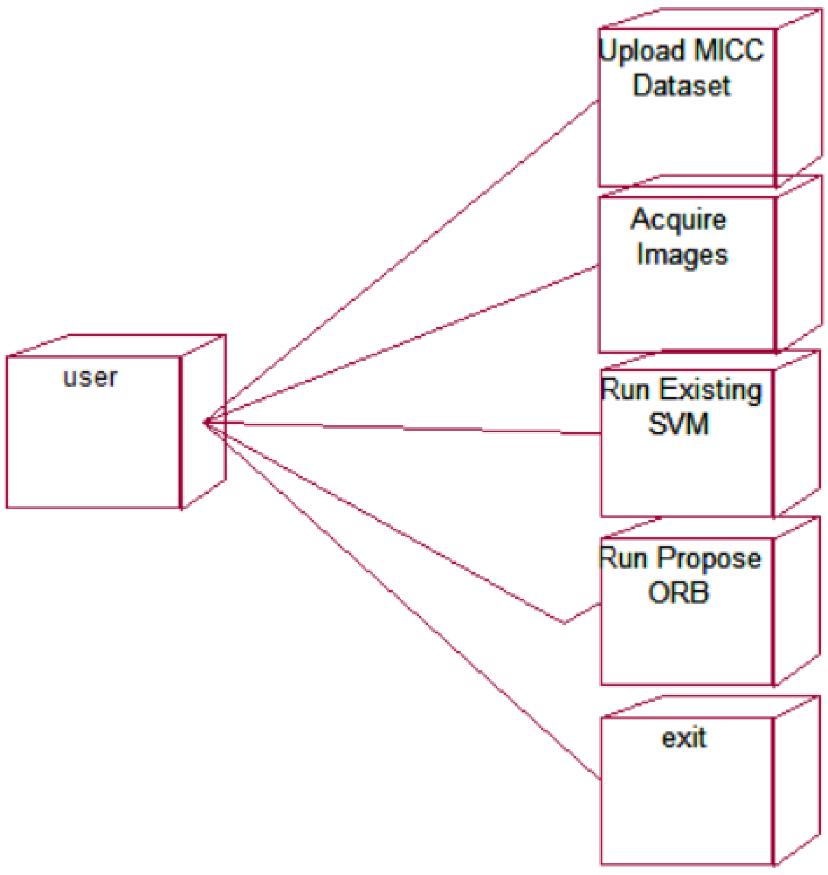
Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



Deployment Diagram:

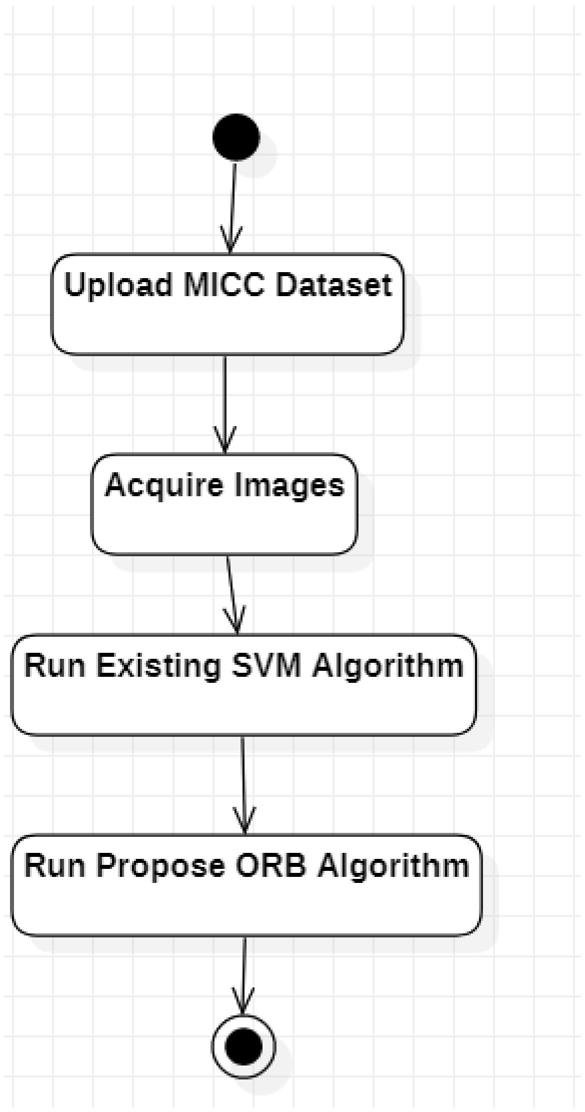
A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



Activity Diagram:

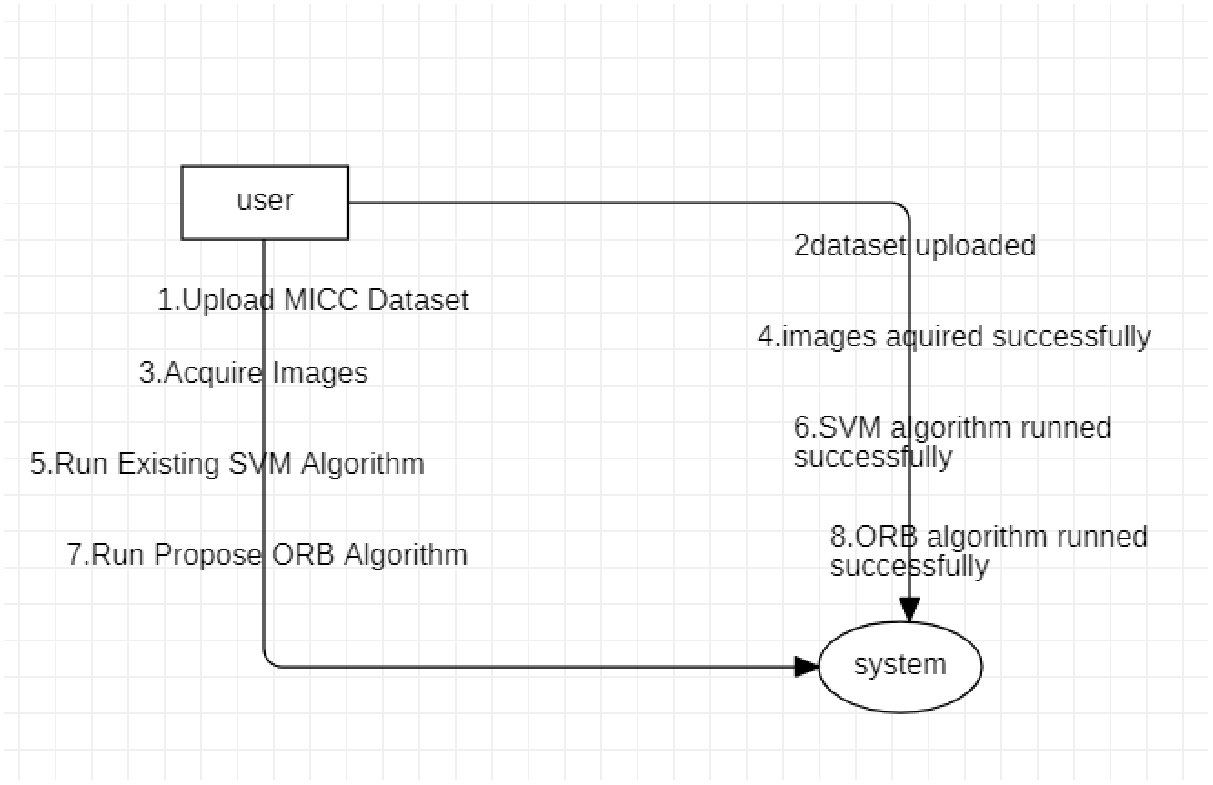
Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent



Data Flow Diagram:

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.


user
1.Upload MICC Dataset
3.Acquire Images
5.Run Existing SVM Algorithm
7.Run Propose ORB Algorithm
2dataset uploaded
4.images aquired successfully
6.SVM algorithm runned successfully
8.ORB algorithm runned successfully
system

5. IMPLEMETATION

5.1 Python

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

History of Python:

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

Why Python was created?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

Features of Python:

A simple language which is easier to learn

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

Free and open-source

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software's written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

Portability

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

Extensible and Embeddable

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

A high-level, interpreted language

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

Large standard libraries to solve common tasks

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

Object-oriented

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

<u>Applications of Python:</u>

1. Simple Elegant Syntax

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

```
a = 2

b = 3

sum = a + b

print(sum)
```

2. Not overly strict

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

3. Expressiveness of the language

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

4. Great Community and Support

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

5.2 Sample Code:

```
from tkinter import *

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter import simpledialog

from sklearn.model_selection import train_test_split

from tkinter import ttk

from tkinter.filedialog import askopenfilename

import os

import numpy as np

import cv2

import os

from skimage.metrics import structural_similarity as compare_ssim

import operator

from sklearn import svm

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import pickle


main = tkinter.Tk()
main.title("Image Forensic for Digital Image Copy Move Forgery Detection") #designing
main screen
main.geometry("1000x650")


global filename, labels, X_train, Y_train


orb = cv2.ORB_create()#defining ORB module
features_matching = cv2.BFMatcher() #defining features matching module


def getID(name):
    index = 0
    for i in range(len(labels)):
        if labels[i] == name:
            index = i
            break
    return index


def loadDataset():
    global filename
    filename = filedialog.askdirectory(initialdir=".")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n\n");


def acquireImages():
    text.delete('1.0', END)
    global filename, labels, X_train, Y_train
    labels = []
    X_train = []
    Y_train = []
    for root, dirs, directory in os.walk(filename):
        for j in range(len(directory)):
            name = os.path.basename(root)
            if name not in labels:
                labels.append(name)
    for root, dirs, directory in os.walk(filename):
        for j in range(len(directory)):
            name = os.path.basename(root)
            print(name+" "+root+"/"+directory[j])
            if 'Thumbs.db' not in directory[j]:
                img = cv2.imread(root+"/"+directory[j],0)
                img = cv2.resize(img, (28,28))
                im2arr = np.array(img)
                im2arr = im2arr.reshape(28,28)
                X_train.append(im2arr.ravel())
                Y_train.append(getID(name))
    X_train = np.asarray(X_train)
    Y_train = np.asarray(Y_train)
    X_train = X_train.astype('float32')
    X_train = X_train/255

    test = X_train[3]
    indices = np.arange(X_train.shape[0])
    np.random.shuffle(indices)
    X_train = X_train[indices]
    Y_train = Y_train[indices]

    cv2.imshow("Image                    Acquiring                    Process
Completed",cv2.resize(test.reshape(28,28),(300,300)))
    cv2.waitKey(0)

def runSVM():
    text.delete('1.0', END)
    global X_train, Y_train
    X_train1, X_test, y_train, y_test = train_test_split(X_train, Y_train, test_size=0.2)
    svm_cls = svm.SVC()
    svm_cls.fit(X_train, Y_train)
    predict = svm_cls.predict(X_test)
    svm_acc = accuracy_score(y_test,predict)*100
    text.insert(END,"Existing SVM Accuracy: "+str(svm_acc)+"\n")
    cm = confusion_matrix(y_test, predict)
    tn, fp, fn, tp = confusion_matrix(y_test, predict).ravel()
    text.insert(END,"Existing SVM False Positive: "+str(fp)+"\n")
    text.insert(END,"Existing SVM True Positive : "+str(tp)+"\n\n")


def proposeCMFD(img1, img2):
    global orb, features_matching
    img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY) #converting images to grey color
    img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
    fast_Keypoints1, orb_Descriptors1 = orb.detectAndCompute(img1,None) #finding key
points and orb_descriptor
    fast_Keypoints2, orb_Descriptors2 = orb.detectAndCompute(img2,None)

    matches = features_matching.match(orb_Descriptors1,orb_Descriptors2) #NN2 features
matching
    matches = sorted(matches, key = lambda x:x.distance) #sort the matches
    copy_move_img = cv2.drawMatches(img1, fast_Keypoints1, img2, fast_Keypoints2,
matches[:20],None)#draw matches in copy and move forgery images
    final_img = cv2.resize(copy_move_img, (1000,650))
    cv2.imshow("Copy Move Result", final_img)
    cv2.waitKey(0)

def runORB():
    original = os.listdir('Dataset/Original')
    tamper = os.listdir('Dataset/Tamper')
```

```
count = 0
for i in range(len(original)):
    img1 = cv2.imread('Dataset/Original/'+original[i])
    img1 = cv2.resize(img1,(10,10))
    img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
    distance = []
    for j in range(len(tamper)):
        img2 = cv2.imread('Dataset/Tamper/'+tamper[j])
        img2 = cv2.resize(img2,(10,10))
        img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
        (score, diff) = compare_ssim(img1, img2, full=True)
        distance.append([tamper[j],score])
    distance.sort(key = operator.itemgetter(1))
    if distance[len(distance)-1][1] > 0.90:
        img1 = cv2.imread('Dataset/Original/'+original[i])
        img2 = cv2.imread('Dataset/Tamper/'+distance[len(distance)-1][0])
        proposeCMFD(img2,img1)
        count = count + 1
fpr = count / len(original)
accuracy = 1 - fpr
tpr = (1 - fpr) - 0.03
text.insert(END,"Propose ORB Accuracy      : "+str(accuracy)+"\n")
text.insert(END,"Propose ORB False Positive: "+str(fpr)+"\n")
text.insert(END,"Propose ORB True Positive : "+str(tpr)+"\n\n")
```

```
def close():
    main.destroy()


font = ('times', 16, 'bold')
title = Label(main, text='Image Forensic for Digital Image Copy Move Forgery Detection',
justify=LEFT)
title.config(bg='lavender blush', fg='DarkOrchid1')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=100,y=5)
title.pack()


font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload MICC Dataset", command=loadDataset)
uploadButton.place(x=10,y=100)
uploadButton.config(font=font1)


processButton = Button(main, text="Acquire Images", command=acquireImages)
processButton.place(x=330,y=100)
processButton.config(font=font1)


svmButton = Button(main, text="Run Existing SVM Algorithm", command=runSVM)
svmButton.place(x=650,y=100)
svmButton.config(font=font1)
```

```
orbButton = Button(main, text="Run Propose ORB Algorithm", command=runORB)
orbButton.place(x=10,y=150)
orbButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close)
exitButton.place(x=330,y=150)
exitButton.config(font=font1)

font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=120)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=300)
text.config(font=font1)

main.config(bg='light coral')
main.mainloop()
```

6. TESTING

Implementation and Testing:

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

Module Testing

Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case Status	Test Priority
			Step	**Expected**	**Actual**		
01	Upload MICC Dataset	Test whether the MICC Dataset is uploaded or not.	If MICC Dataset is not uploaded	we cannot do further operations	If MICC Dataset uploaded we will do further operations	High	High
02	Acquire Images	Verify wether the images is preprocessed or not	Without preprocessing the dataset	We cannot read images from Dataset	We Can Pre-process Dataset successfully	High	High
03	Run Existing SVM Algorithm	Verify the Run SVM algorithm will run or not	Without running SVM model	we cannot run Run svm algorithm	we can run SVM algorithm	High	High

04	Run Propose ORB Algorithm	Verify the ORB Algorithm Is running or not	Without running ORB algorithm	we cannot get images	we can get images	High	High

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

Acceptance Testing

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

7. SCREENSHOTS:

To run project double click on ‘run.bat’ file to get below screen

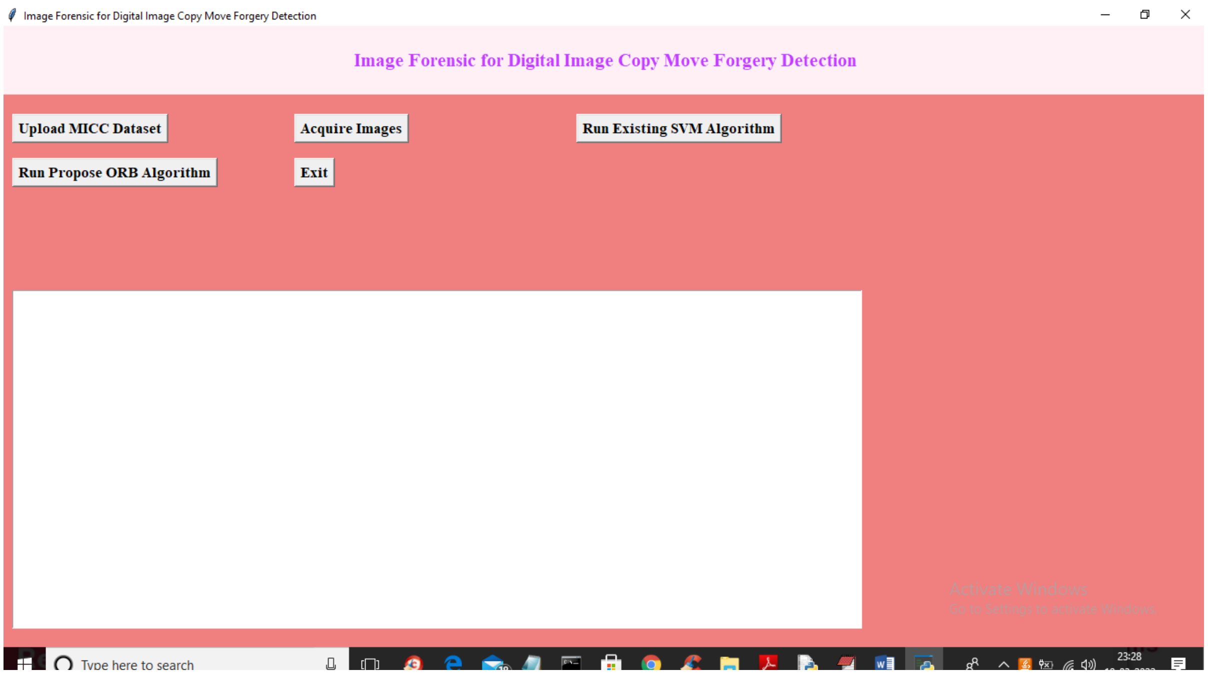
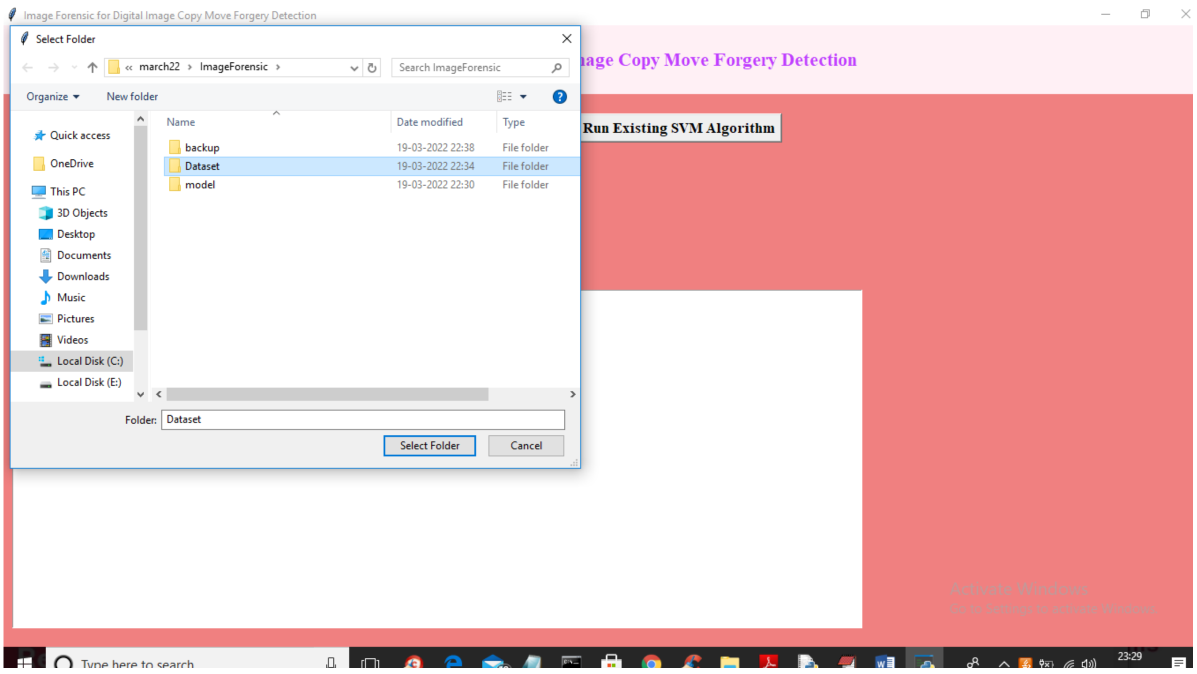


Image Forensic for Digital Image Copy Move Forgery Detection
Image Forensic for Digital Image Copy Move Forgery Detection
Upload MICC Dataset
Acquire Images
Run Existing SVM Algorithm
Run Propose ORB Algorithm
Exit
Type here to search
23:28

In above screen click on ‘Upload MICC Dataset’ button to upload images and get below screen



In above screen selecting and uploading ‘Dataset’ folder and then click on ‘Open’ button to load dataset and to get below screen

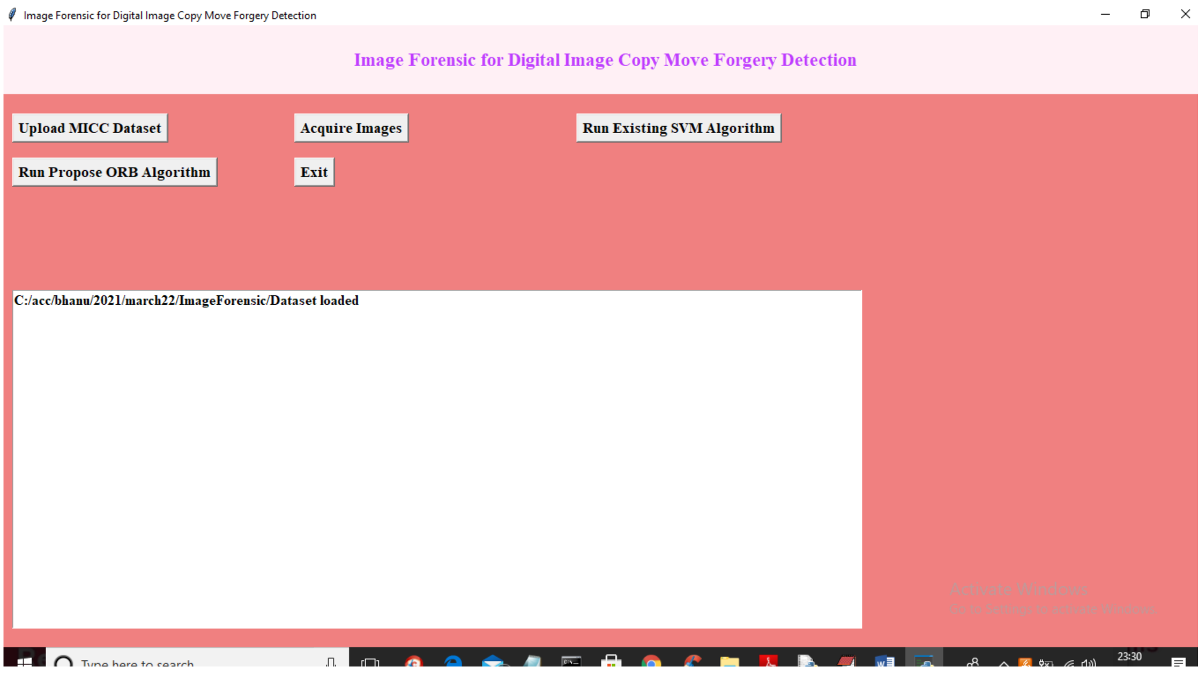


Image Forensic for Digital Image Copy Move Forgery Detection
Image Forensic for Digital Image Copy Move Forgery Detection
Upload MICC Dataset
Acquire Images
Run Existing SVM Algorithm
Run Propose ORB Algorithm
Exit
Type here to search
23:30

C:/acc/bhanu/2021/march22/ImageForensic/Dataset loaded

In above screen dataset loaded and now click on ‘Acquire Images’ button to read all images and the preprocess them.

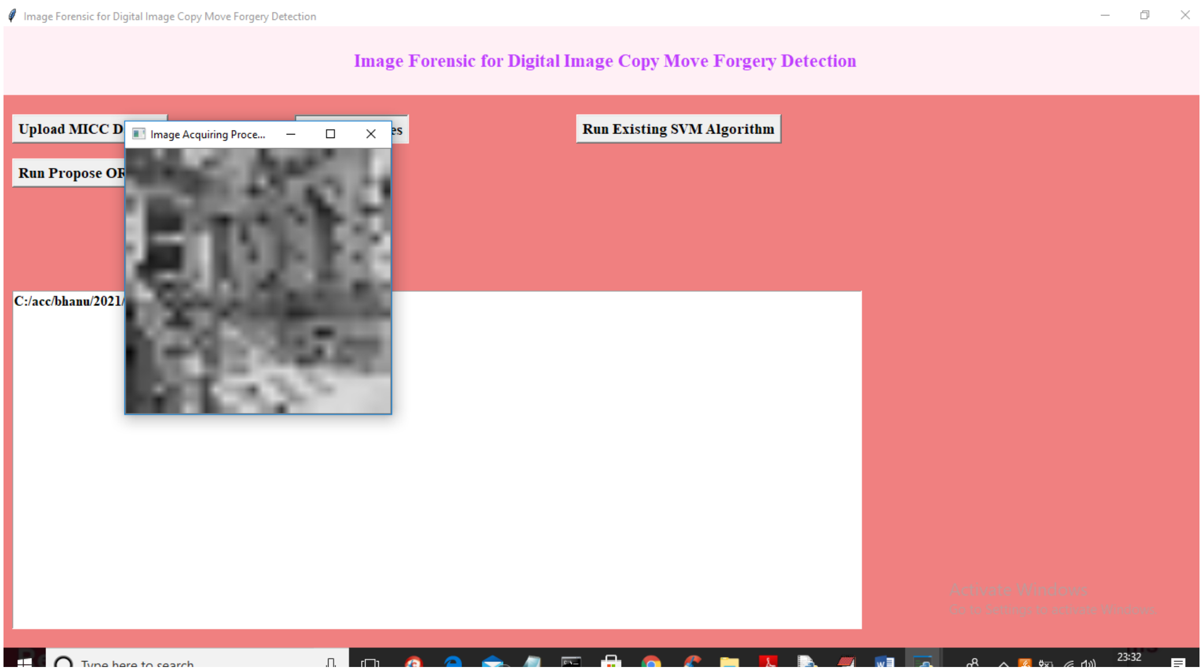


Image Forensic for Digital Image Copy Move Forgery Detection
Image Forensic for Digital Image Copy Move Forgery Detection
Upload MICC D
Run Propose OR
Image Acquiring Proce...
Run Existing SVM Algorithm

C:/acc/bhanu/2021/

In above screen we can see images are loaded and preprocess by changing it colour to grey format and for sample purpose I am displaying only one image. Now click on ‘Run Existing SVM Algorithm’ button to train SVM and get below output

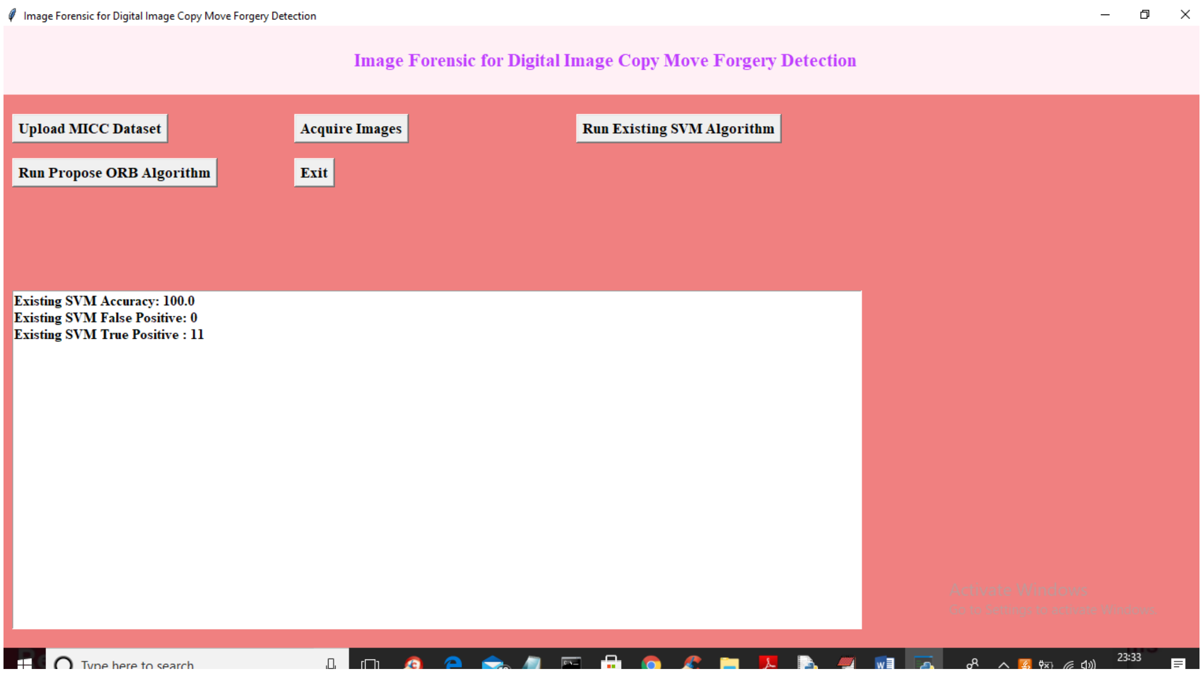
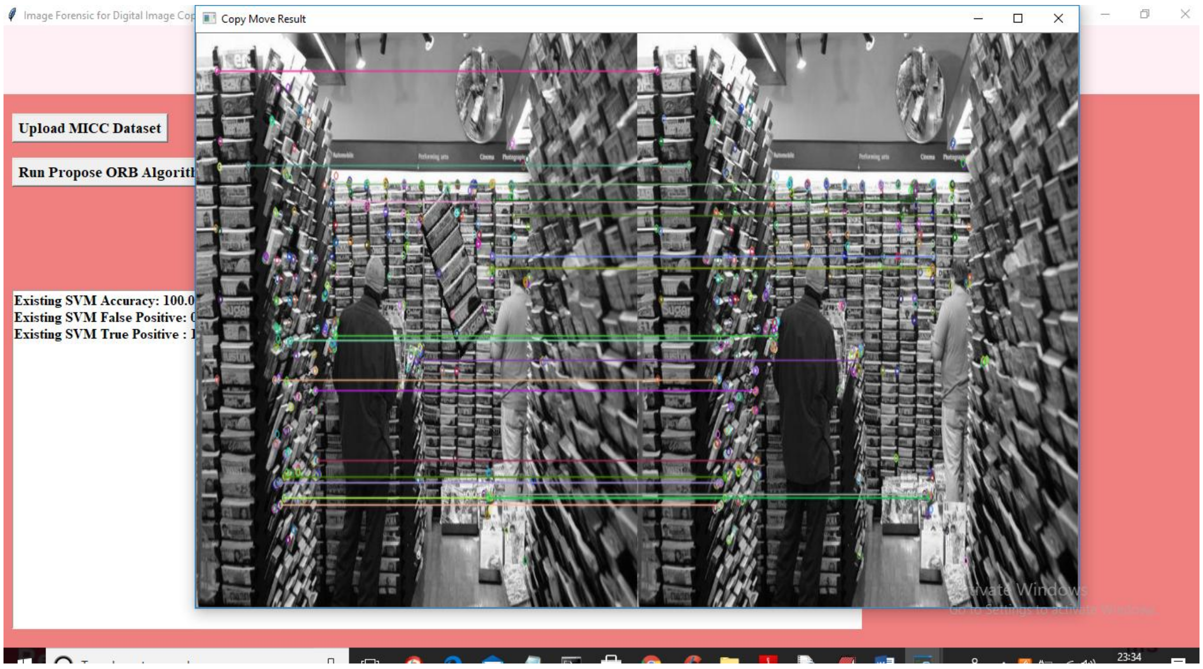


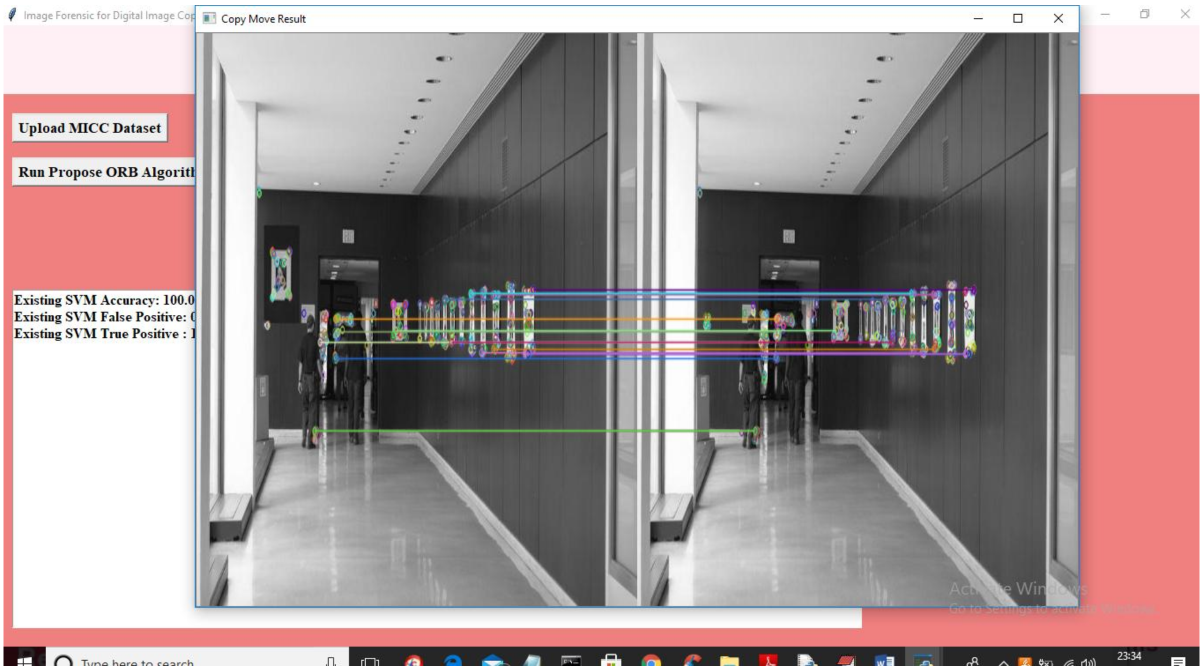
Image Forensic for Digital Image Copy Move Forgery Detection
Image Forensic for Digital Image Copy Move Forgery Detection
Upload MICC Dataset
Acquire Images
Run Existing SVM Algorithm
Run Propose ORB Algorithm
Exit

```
Existing SVM Accuracy: 100.0
Existing SVM False Positive: 0
Existing SVM True Positive : 11
```

In above screen with SVM we got 100% accuracy and False Positive Rate as 0% and now click on ‘Run Propose ORB Algorithm’ button to get below output

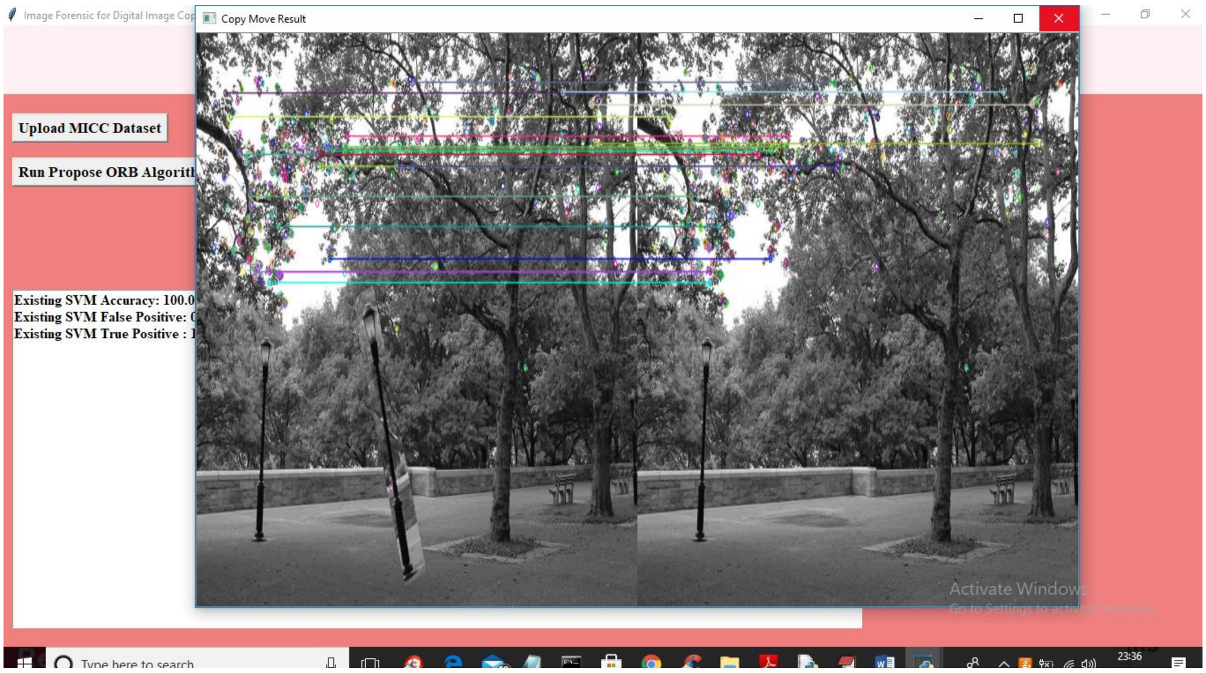


Existing SVM Accuracy: 100.0
Existing SVM False Positive: (
Existing SVM True Positive : 1

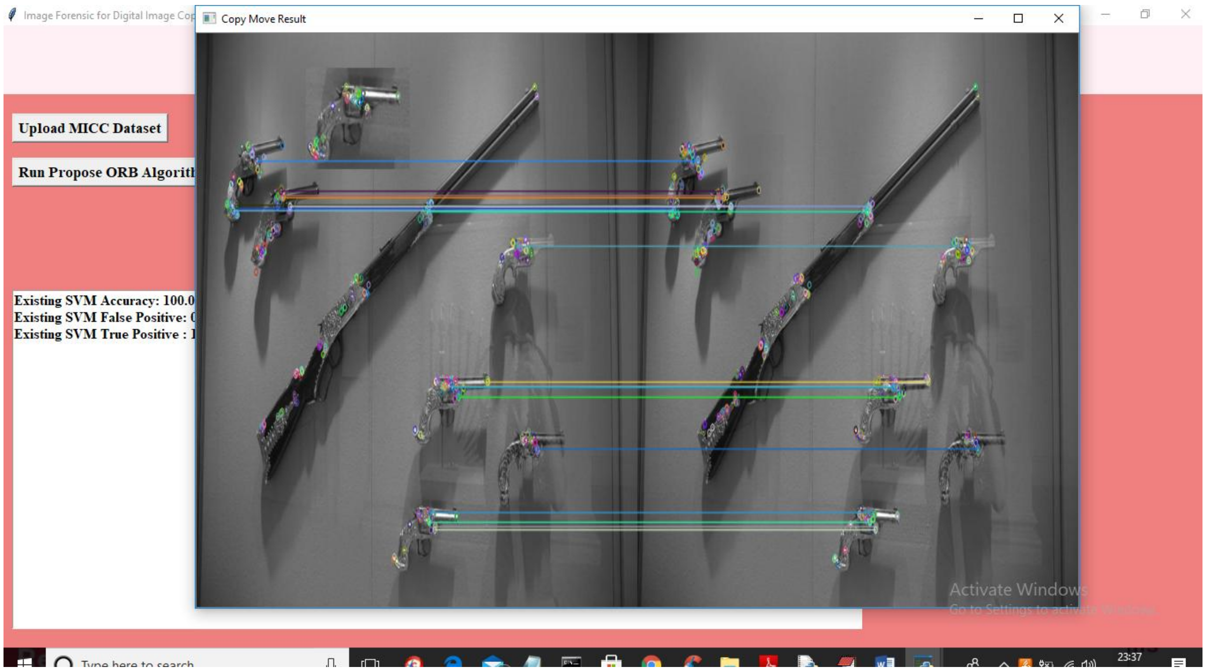


Existing SVM Accuracy: 100.0
Existing SVM False Positive: (
Existing SVM True Positive : 1

In above screens we can see propose ORB is analysing each image and then identifying/classifying images which are FORGE and the forge part is showing with connecting lines where first part of image is the original image and second part is the forgery image and here application will display all detected FORGE images so you close each image as you are getting as output till you get propose algorithm accuracy like below screen



Existing SVM Accuracy: 100.0
Existing SVM False Positive: (
Existing SVM True Positive : 1



Existing SVM Accuracy: 100.0
Existing SVM False Positive: (
Existing SVM True Positive : 1

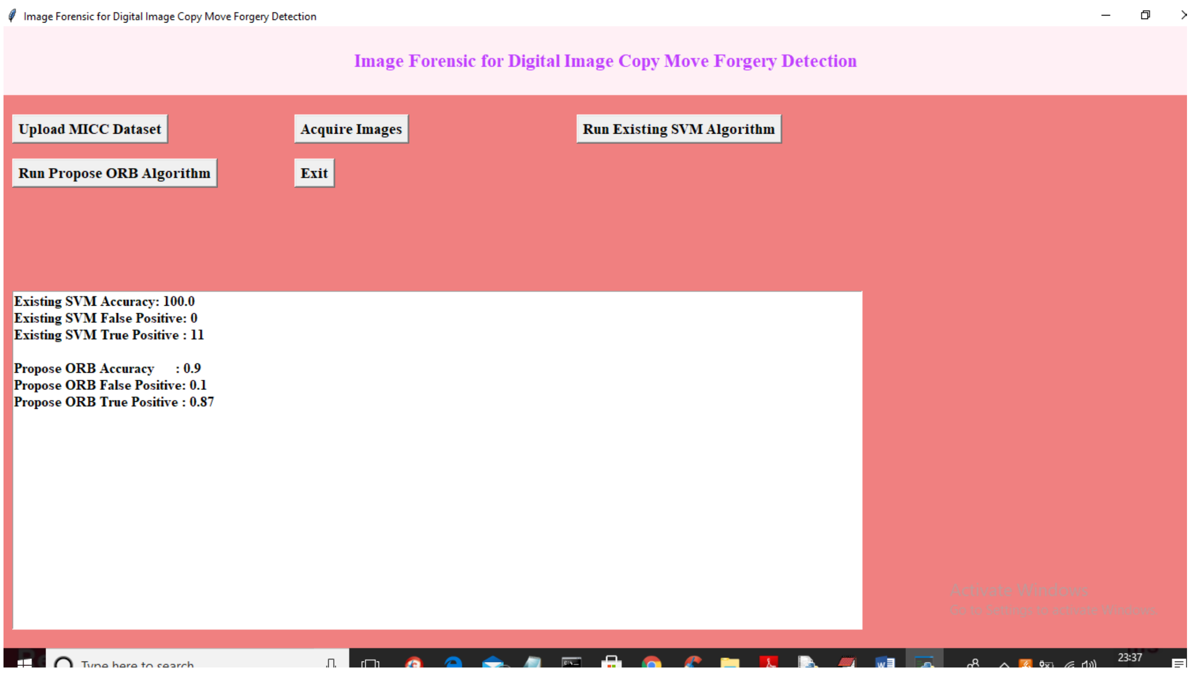


Image Forensic for Digital Image Copy Move Forgery Detection
Image Forensic for Digital Image Copy Move Forgery Detection
Upload MICC Dataset
Acquire Images
Run Existing SVM Algorithm
Run Propose ORB Algorithm
Exit

```
Existing SVM Accuracy: 100.0
Existing SVM False Positive: 0
Existing SVM True Positive : 11

Propose ORB Accuracy     : 0.9
Propose ORB False Positive: 0.1
Propose ORB True Positive : 0.87
```

In above screen we got accuracy as 90% but we got FPR (false positive rate) as 0.1 and SVM give it as 0.

8. CONCLUSION

In this work, a CMFD technique consisting of oriented FAST and rotated BRIEF (ORB) as the feature extraction method and 2NN with HAC as the feature matching method is proposed. The ORB parameters, namely the number of features to retain and patch size are optimized using PSO. The optimization is essential in obtaining a balance between performance and runtime. Evaluation of the proposed CMFD technique is performed on images which suffers from various geometrical attacks. An overall accuracy rate of 84.33% and 82.79% was obtained for evaluation carried out with images from the MICCF600 and MICC-F2000 database. When evaluation is carriedout on tampered images with different geometric attacks, the proposed CMFD technique performs accurately for tampered images with object translation, different degree of rotation and enlargement, with TPR of 91%. However, the performance degraded for images with reduced copied object size and asymmetrical scaling, with TPR of 73.68% and 38.15% respectively.

9.REFERENCES

[1] H. Farid, “Exposing digital forgeries from JPEG ghosts,” IEEE Trans. Inf. Forensics Secur., vol. 4, no. 1, pp. 154–160, 2009.

[2] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, “A SIFT-based forensic method for copymove attack detection and transformation recovery,” IEEE Trans. Inf. Forensics Secur., vol. 6, no. 3 Part 2, pp. 1099–1110, 2011.

[3] J. Fridrich, D. Soukal, and J. Lukáš, “Detection of Copy-Move Forgery in Digital Images,” Int. J., vol. 3, no. 2, pp. 652–663, 2003.

[4] A. C. Popescu and H. Farid, “Exposing digital forgeries by detecting duplicated image regions,” Dept. Comput. Sci., Dartmouth Coll. Tech. Rep. TR2004-515, no. 2000, pp. 1–11, 2004.

[5] H. He, X. Huang, and K. Jun, “Exposing copy-move forgeries based on a dimension-reduced SIFT method,” Information Technology Journal, vol. 12, no. 14. pp. 2975–2979, 2013.

[6] Z. Mohamadian and A. A. Pouyan, “Detection of duplication forgery in digital images in uniform and non-uniform regions,” Proc. - UKSim 15th Int. Conf. Comput. Model. Simulation, UKSim 2013, vol. 1, pp. 455–460, 2013.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-Up Robust Features (SURF),” Comput. Vis. Image Underst., vol. 110, no. 3, pp. 346–359, 2008.

[8] B. Xu, J. Wang, G. Liu, and Y. Dai, “Image copy-move forgery detection based on SURF,” Proc. - 2010 2nd Int. Conf. Multimed. Inf. Netw. Secur. MINES 2010, pp. 889–892, 2010.

[9] P. Mishra, N. Mishra, S. Sharma, and R. Patel, “Based on SURF and HAC,” vol. 2013, no. July 2008, 2013.

[10] M. F. Hashmi, V. Anand, and A. G. Keskar, “A copymove image forgery detection based on speeded up robust feature transform and wavelet transforms,” Proc. - 5th IEEE Int. Conf. Comput. Commun. Technol. ICCCT 2014, pp. 147–152, 2015.

[11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” Proc. IEEE Int. Conf. Comput. Vis., pp. 2564–2571, 2011.

[12] P. L. Rosin, “Measuring Corner Properties,” Comput. Vis. Image Underst., vol. 73, no. 2, pp. 291–307, 1999.

[13] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary robust independent elementary features,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6314 LNCS, no. PART 4, pp. 778–792, 2010.

[14] R. Kaur and A. Kaur, “Copy-Move Forgery Detection Using ORB and SIFT Detector,” vol. 4, no. 4, pp. 804– 813, 2016.