
# Tribhuvan University

## Faculty of Humanities and Social Sciences

# "FAKE CURRENCY DETECTION USING ORB, SSIM AND CONTOUR DETECTION ALGORITHM"

**A PROJECT REPORT**

**Submitted to**

**Department of Computer Application**

**Deerwalk Institute of Technology**

**Sifal, Kathmandu**

***In partial fulfillment of the requirements for the Bachelors in Computer Applications***

Submitted by

**Vasker Raj Pandey**

6-2-1175-12-2019

November, 2024 A.D.

Under the supervision of

**Mr. Shyam Khatiwada**


# Tribhuvan University

## Faculty of Humanities and Social Science

## Deerwalk Institute of Technology

Sifal, Kathmandu

Bachelor in Computer Application (BCA)

# SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by **Vasker Raj Pandey** entitled **"Fake Currency Detection Using ORB, SSIM and Contour Detection Algorithm"** in the Partial Fulfillment of requirement for the degree of Bachelor in Computer Application is recommended for that final evaluation.

_____

Signature

Mr. Shyam Khatiwada

**Project Supervisor**

Senior Lecturer

Deerwalk Institute of Technology


# Tribhuvan University

## Faculty of Humanities and Social Science

## Deerwalk Institute of Technology

Sifal, Kathmandu

Bachelor in Computer Application (BCA)

# LETTER OF APPROVAL

This is to certify that this project prepared by **Vasker Raj Pandey** entitled **"Fake Currency Detection Using ORB, SSIM and Contour Detection Algorithm"** in partial fulfillment of the requirements for the degree of Bachelors in Computer Applications has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

| | |
|---|---|
| ……<br>**Shyam Khatiwada**<br>Supervisor<br>DWIT College | ……<br>**Roshan Tandukar**<br>External Examiner<br>FOHSS, TU |
| ……<br>**Shristi Awale**<br>Coordinator<br>DWIT College | ……<br>**Hitesh Karki**<br>Campus Chief<br>DWIT College |

# ACKNOWLEDGEMENT

I want to sincerely thank my supervisor, Mr. Shyam Khatiwada, for his essential advice, assistance, and encouragement during the course of this project. His knowledge of the subject topic and skills have been crucial in determining the course and outcome of this project. I am incredibly thankful for his constant dedication to my academic development. I am also extending this gratitude to the teachers who have helped me create this project directly or indirectly.

Additionally, I would want to express my sincere gratitude to my friends, family, and seniors for their unwavering support and motivation. I've been determined to finish this project because of their confidence in my ability and ongoing encouragement. Their insightful comments, encouraging remarks, and constructive criticism all helped to shape the final product of this work. They have provided me with constant support, patience, and encouragement throughout this journey, and I am grateful to them. Their contributions have been invaluable in making this project a reality.

Sincerely,

Vasker Raj Pandey

# ABSTRACT

The growing issue of counterfeit currency poses a significant threat to both individual livelihoods and national economies, contributing to inflation and financial instability. While some counterfeit detection systems are available, they are typically limited to banks and large corporations, leaving small businesses and the general public vulnerable. In this project, we explore the intricate security features embedded within Indian currency notes, developing a software-based solution to detect and invalidate fake currency. The system leverages advanced image processing and computer vision techniques, utilizing Python in a Jupyter Notebook environment. Key techniques include feature extraction, comparison, and matching algorithms such as ORB (Oriented FAST and Rotated BRIEF), SSIM and Contour Detection. This system provides a user-friendly and efficient way for individuals and businesses to authenticate currency notes with high accuracy, filling a critical gap in accessible counterfeit detection technology.

*Keywords: Fake currency, counterfeit detection, image processing, feature extraction, ORB detector, Python, Jupyter Notebook, currency authentication, computer vision, SSIM.*

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| FCD | Fake Currency Detection |
| ORB | Oriented FAST and Rotated BRIEF |
| FAST | Features from Accelerated Segment Test |
| BRIEF | Binary Robust Independent Elementary Features |
| SSIM | Structural Similarity Index Measure |
| CD | Contour Detection |
| GUI | Graphical User Interface |
| UML | Unified Modeling Language |

# CHAPTER 1: INTRODUCTION

## 1.1. Introduction

Currency duplication or production of counterfeit currency notes illegally by imitating the actual manufacturing process is a huge problem that every country is facing. Fake currency can reduce the value of real money and cause inflation due to an unauthorized and unnatural increase in the money supply. Manual authentication of currency notes is a solution but it is a very time-consuming, inaccurate, and difficult process. Automatic testing of currency notes is, therefore, necessary for handling large volumes of currency notes and then, getting accurate results in a very short time span. In this project, we propose a fake currency note detection system using various image processing techniques and algorithms. The proposed system is designed to validate Nepali currency notes of denomination 500 rupees. The system consists of three main algorithms and checks the authenticity of various features in a currency note.

## 1.2. Problem Statement

Counterfeiting of currency is a significant issue that undermines the financial stability of economies and impacts individuals and businesses alike. Existing solutions for detecting fake currency are typically expensive, complex, and only accessible in large financial institutions. Foreign visitors and businesses in Nepal face difficulties in recognizing and verifying Nepali currency. The need for an accessible, automated, and reliable system that can authenticate currency notes for small businesses and the general public is crucial. This project aims to develop a user-friendly software solution that utilizes image processing and machine learning techniques to detect counterfeit currency in real-time, with high accuracy and speed.

## 1.3. Objectives

- To develop a counterfeit detection system by using ORB (Oriented FAST and Rotated BRIEF) for accurate feature detection and matching on currency notes.
- To improve detection accuracy, implement SSIM (Structural Similarity Index) for comparing key features of real and fake currency, ensuring reliable image similarity measurements.
- To apply contour detection algorithms for precise identification of bleed lines and number panels in currency notes.

- To provide real-time results, optimize the system to ensure fast and accurate counterfeit currency detection.

## 1.4. Scope and Limitation

### 1.4.1. Scope

- The project can be used for detecting counterfeit currency by analyzing the physical characteristics of banknotes using ORB (Oriented FAST and Rotated BRIEF) feature detection, SSIM (Structural Similarity Index Measure) for comparing image similarity, and contour detection to identify irregularities.
- It can be integrated into automated systems for banks, retail outlets, and currency exchange services to enhance security measures by providing real-time counterfeit detection.
- The system can assist forensic experts in identifying and analyzing fake currency for investigation purposes.
- It can be a basis for further research into improving counterfeit detection techniques and integrating them with other security measures.

### 1.4.2. Limitation

- The accuracy of detection may decrease if the counterfeit currency is of high quality and closely mimics genuine notes, especially if subtle differences are hard to capture with ORB and contour detection.
- Variations in lighting conditions, camera quality, and image resolution can affect the performance of the detection algorithms.
- Advanced counterfeit techniques that involve modifications to the visual properties of the currency might not be effectively detected using current algorithms.
- The system may need retraining or adjustments to detect counterfeits across different currency designs and denominations effectively.

## 1.5. Development Methodology

The Waterfall Model is a conventional software development methodology that follows a sequential and linear approach to project execution. In this model, the software development life cycle is divided into distinct, non-overlapping phases, where each phase must be completed before moving on to the next. The process begins with requirements

gathering and analysis, followed by system design, implementation, testing, deployment, and finally, maintenance.

Each phase produces documentation, and progress cascades through the stages like a waterfall, with a formal review process before transitioning to the next phase.

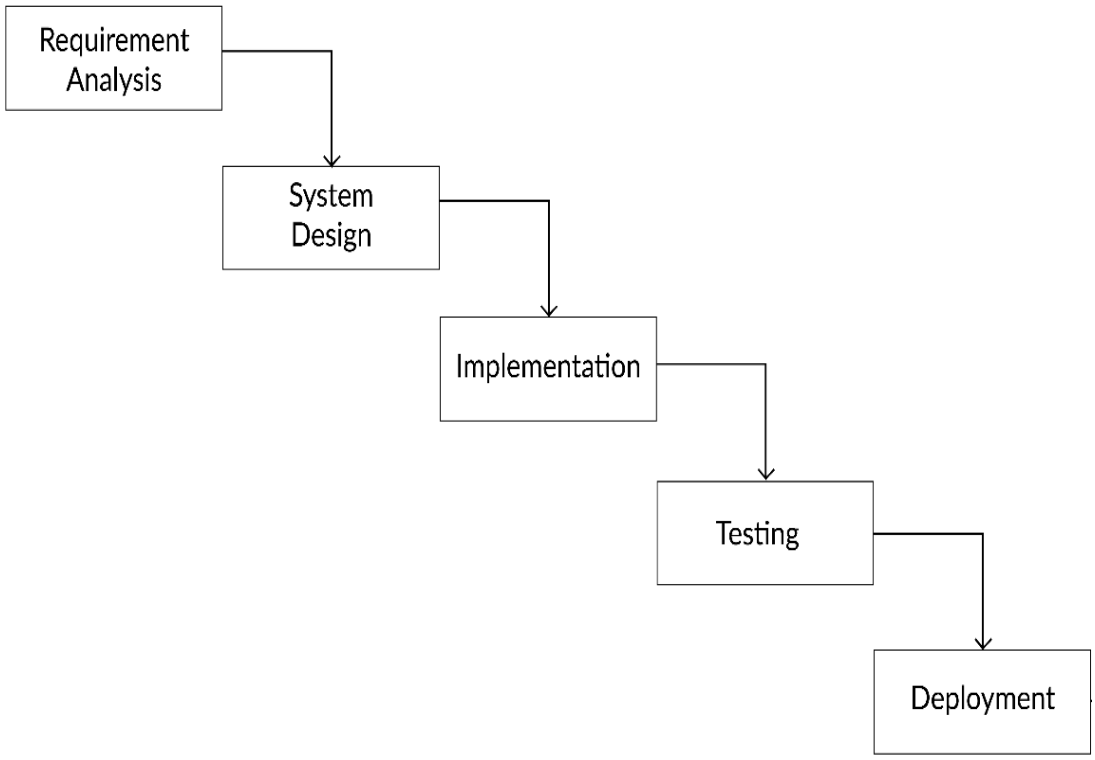



**Figure 1. 1: Waterfall Model of Fake Currency Detector**

- **Requirements Analysis:** First, gather and document what the system needs to do, like detecting counterfeit bills accurately and working in real-time.
- **System Design:** Next, plan how the system will be built, including designing algorithms and deciding how different parts of the system will work together.
- **Implementation:** Then, actually build the system by coding the algorithms and integrating the different components.
- **Testing:** After building, test the system to make sure everything works correctly and meets the requirements. This includes checking for bugs and ensuring the system performs well.
- **Deployment:** Once testing is complete, deploy the system for use, and train users on how to use it.

The Waterfall model is a step-by-step approach, where each phase must be completed before moving on to the next.

## 1.6. Report Organization

Chapter 1 The report starts with the introduction of the proposed application, “Fake Currency Detection Using ORB, SSIM and Contour Detection Algorithm” along with the problem statement and objectives of the project. Here, we have introduced why the system is built and the conditions of why the system is reliable. It also provides objectives and scope of the project and lists the possible measures that can be applied to solve them.

Chapter 2 Analyses the similar existing models and reviews them on the basis of our project along with requirement and feasibility analysis of those systems. It also includes both functional and non-functional requirements.

Chapter 3 Provides a detailed overview of the system design along with the various algorithms used in the project.

Chapter 4 Explains the tools used on the project’s front and back end and purpose of it. Testing of the system is also explained in this chapter.

Chapter 5 Discusses the conclusion of how the project is accomplished, its findings etc. We further discuss the recommendation of the future enhancements in the project and how it can be improved.

# CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

## 2.1. Background Study

Counterfeit currency detection is crucial for protecting financial systems and businesses. Traditional methods are often inadequate as counterfeiters use more sophisticated techniques. Advanced image processing can offer a more reliable solution. Counterfeit money can lead to financial losses and undermine trust in currency systems. Accurate detection methods are essential to prevent economic disruptions. ORB (Oriented FAST and Rotated BRIEF) technique detects and describes key features of banknotes efficiently and is robust to rotation, helping to identify genuine currency. SSIM (Structural Similarity Index Measure) compares images by analyzing their structural similarities, making it useful for detecting subtle differences between genuine and counterfeit notes. Contour Detection identifies the boundaries and shapes of objects in images, revealing irregularities in currency design that may indicate counterfeiting. Using ORB, SSIM, and contour detection together enhances detection accuracy. ORB identifies features, SSIM assesses overall similarity, and contour detection highlights design anomalies.

## 2.2. Literature Review

The literature review explores existing research and developments related to the detection and classification of players, referees, and other entities in football matches using machine learning and computer vision techniques.

T. Agasti, G. Burand, P. Wade and P. Chtira, "Fake currency detection using image processing." [1] This study examines the use of artificial intelligence to improve counterfeit currency detection. It explores AI's role in adapting to diverse counterfeiting techniques and enhancing detection accuracy and reliability.

Patil, D. P., Varma, G., Poojary, S., Sawant, S., & Sharma, A. "Counterfeit Currency Detection Based on AI." [2] This study examines the use of artificial intelligence to improve counterfeit currency detection. It explores AI's role in adapting to diverse counterfeiting techniques and enhancing detection accuracy and reliability.

Shah, R., Sheth, P., Champaneri, M., & Gaikwad, V. "Currency Counting Fake Note Detection." [3] Department of Information Technology, K. J. Somaiya Institute of

Engineering and Information Technology, University of Mumbai, India. This research focuses on enhancing currency counting systems by integrating fake note detection mechanisms. It addresses real-time detection during currency counting, aiming to improve accuracy and operational efficiency in financial institutions.

Lamsal, S., & Shakya, A. "Counterfeit Paper Banknote Identification Based on Color and Texture." Department of Electronics and Computer Engineering, IOE, Central Campus. [4] This study investigates the use of color and texture analysis for identifying counterfeit banknotes. It demonstrates how variations in these attributes can aid in detecting fake currency, though challenges remain in distinguishing high-quality counterfeits that closely mimic genuine notes.

Rizal, M., Sarno, R., & Prayitno, A. "Implementation of a Digital Image Processing Method for Currency Authentication System." IOP Conference Series: Materials Science and Engineering, 263(5), 052047. [5] The study presents a digital image processing method for authenticating currency, emphasizing various image features and processing techniques to ensure currency validity. It provides a foundation for developing more advanced authentication systems.

Supriyadi, D., & Mariko, B. "Application of Machine Learning for Fake Currency Detection." E3S Web of Conferences, 371, 02020. [6] This research explores the use of machine learning algorithms to enhance counterfeit currency detection. It highlights how machine learning can improve accuracy by learning from extensive datasets of genuine and counterfeit notes, offering a more adaptive solution.

Colaco, R. M., Fernandes, R., & Sowmya, S. "Efficient Image Processing Technique for Authentication of Indian Paper Currency." International Conference on Computer Communication and Informatics (ICCCI). IEEE. [7] The study discusses an image processing technique specifically designed for Indian paper currency. It emphasizes the need for tailored approaches to detect counterfeit notes based on unique currency features.

# CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

## 3.1. System Analysis

System analysis is a structured and methodical approach employed in software engineering and information technology to thoroughly understand, define, and design complex systems. This process involves a detailed examination of the existing system or problem domain, where analysts assess current operations, workflows, and challenges. By identifying the needs and requirements of stakeholders, system analysis aims to uncover areas for improvement. The goal is to propose solutions that enhance the system's efficiency, effectiveness, and overall functionality. Through this comprehensive analysis, organizations can develop systems that are better aligned with their objectives and more capable of addressing their specific needs and challenges.

### 3.1.1. Requirement Analysis

During the requirement analysis phase for the Fake Currency Detection system, key functionalities will be defined, including feature extraction using the ORB algorithm, contour detection to identify and outline currency features, and image comparison with SSIM for tracking and analyzing currency patterns. Constraints such as computational resource limitations and integration with existing financial verification tools will be considered. Performance criteria, including detection accuracy, processing speed, and system responsiveness, will be evaluated. Techniques such as stakeholder interviews and prototyping will be employed to gather detailed requirements, ensuring the system meets user needs and integrates effectively with current verification processes. This thorough analysis will provide a strong foundation for the system's development.

**i. Functional Requirement**

- The system shall utilize the ORB (Oriented FAST and Rotated BRIEF) algorithm to extract and localize key features from Nepali currency images with high accuracy.
- The system shall implement robust contour detection techniques to identify and outline various security features and patterns on currency notes. Users shall be able to specify the output directory and filename for compressed or decompressed files.

- The system shall use SSIM (Structural Similarity Index) to compare images and track detected currency features across multiple frames, analyzing patterns to identify potential counterfeit characteristics.
- The system shall analyze extracted features and patterns to distinguish between genuine and counterfeit currency, providing a confidence score for each evaluation.
- The system shall provide visual feedback by highlighting detected features and contours on images and generating detailed reports on the analysis results, including identified counterfeit characteristics.

## USECASE DIAGRAM

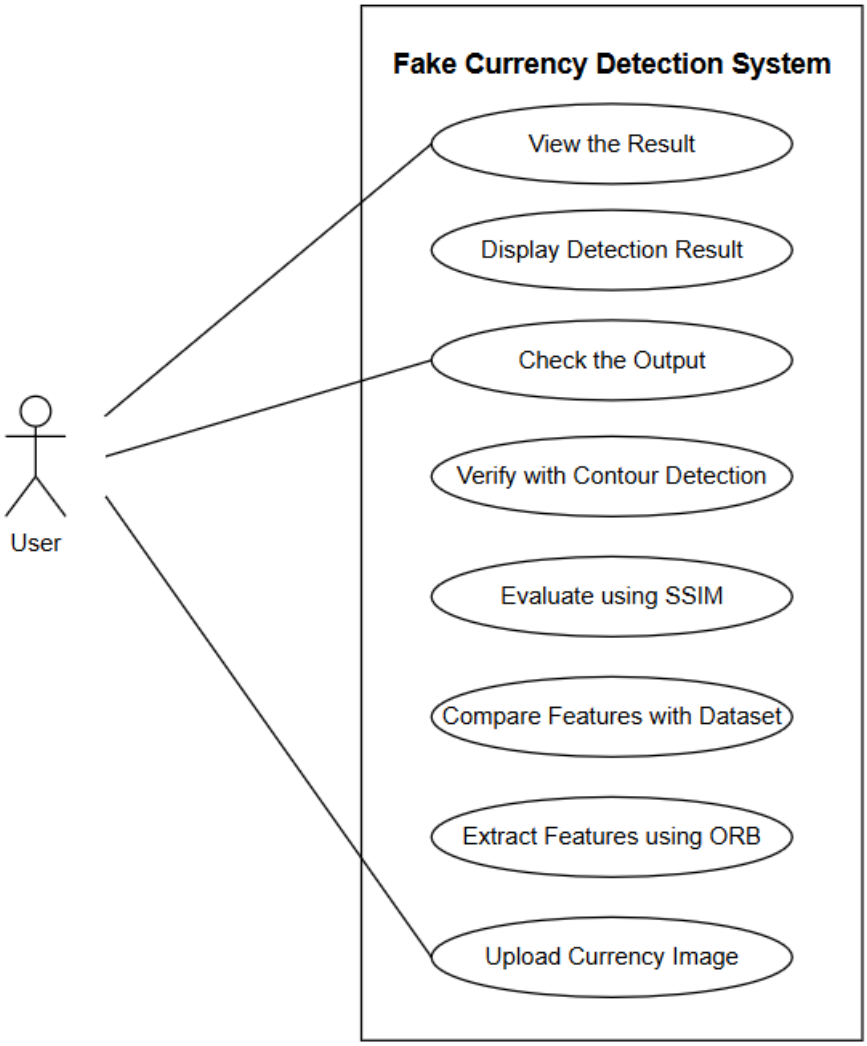



**Figure 3. 1 Use-Case Diagram of Fake Currency Detector**

**ii. Non-Functional Requirement**

The non-functional requirement are as follows:

- The system must have good performance. It should not crash often or take all the resources of the CPU.
- The system must be scalable and efficient. It should be easy to add more algorithms as well as these algorithms should not collide with one another.
- The system must be reliable to use.
- The system must be user friendly. The interface should not overwhelm the users or be over complicated.

### 3.1.2. Feasibility Analysis

Feasibility analysis is a comprehensive evaluation of the practicality, viability, and potential success of a proposed project or business initiative. This process includes assessing various factors such as market demand, financial feasibility, technical requirements, and potential risks to determine if the project is viable before allocating resources. The feasibility study concluded that the project can be successfully implemented due to careful planning.

**a) Technical Feasibility**

The system is technically feasible, as the required hardware and software for its development and implementation are readily available. The project is based on a fundamental programming language that aligns well with its needs, specifically Python. The necessary libraries can produce the desired outcomes, and all existing resources are accessible for both development and system integration.

**b) Operational Feasibility**

The system is designed to be user-friendly, needing only basic computer and internet skills for operation. It meets all specified functional and non-functional requirements, making it operationally feasible. The application is compatible with widely used Windows devices, and all technologies utilized are open-source and freely available.

**c) Economic Feasibility**

Economic feasibility is a crucial factor in project evaluation and decision-making, especially in business and technology projects. It examines whether a proposed project or investment is financially viable and justifiable. A laptop for coding and

stable internet access for troubleshooting will ensure that the project is not financially burdensome to complete.

**d) Schedule Feasibility**

The system is completed within scheduled time and do not exceed the scheduled time. It was hard to complete the project within the given timeframe but has all the features required to make the project complete. For further illustration of how the project was completed in the given timeframe, the below table shows detailed construct on what task were done as:

**Table 3.1: Gantt chart table of Fake Currency Detector**

| Tasks | Start Date | End Date | Start on Day | Days to Complete |
|---|---|---|---|---|
| Documentation | 1-Jun | 12-Sept | 1 | 102 |
| Research and Analysis | 5-Jun | 25-Jun | 5 | 20 |
| Designing | 26-Jun | 25-Jul | 25 | 30 |
| Development and Debugging | 26-Jul | 30-Aug | 54 | 34 |
| Testing and Implementation | 31-Aug | 19-Sept | 90 | 19 |

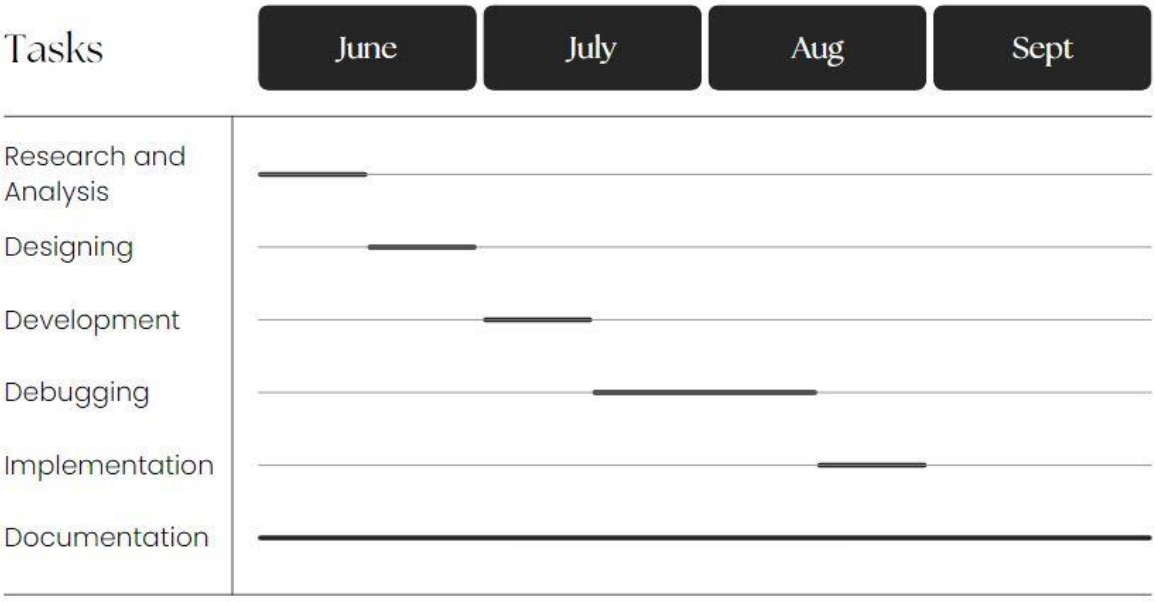



**Figure 3. 2: Gantt-Chart of Fake Currency Detector**

Schedule feasibility involves analyzing the available resources, estimated effort, and dependencies to determine if the proposed schedule is realistic and achievable. Evaluating schedule feasibility helps in identifying potential risks, constraints, and bottlenecks that may impact project timelines.

Throughout the entire duration:

- Keep a contingency plan for unexpected delays.
- Continuously monitor and adjust the schedule based on real-time feedback.
- Communicate effectively with supervisor to manage expectations.
- Improve project accordingly.

Schedule feasibility involves assessing if a project or task can realistically be completed within a given timeframe. The goal is to ensure a realistic and achievable timeline for successful project completion.

### 3.1.3. Object Modelling: Class Diagram

A class diagram is a type of static structure diagram in the Unified Modeling Language (UML) that represents the structure and relationships of classes and other elements in a

system. UML is a standardized modeling language used in software engineering for visualizing, specifying, constructing, and documenting the artifacts of a system.

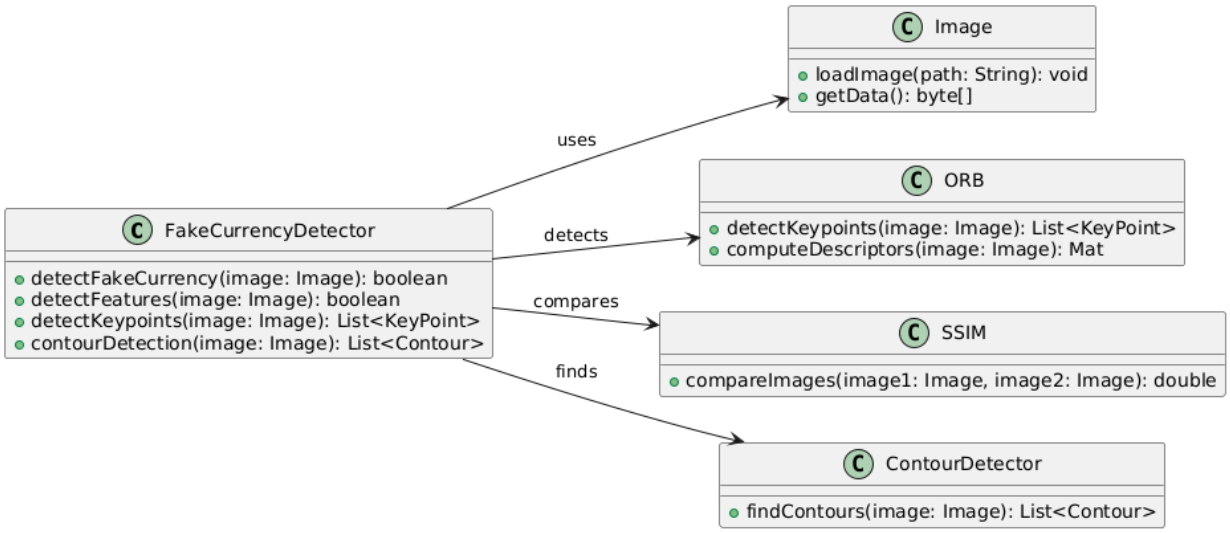



**Figure 3. 3: Class Diagram of Fake Currency Detector**

The class diagram for the Fake Currency Detection system outlines the structure and relationships of key components involved in detecting counterfeit currency. At the center is the FakeCurrencyDetector class, which orchestrates the detection process by using an Image class to load and retrieve the currency image data. It employs the ORB class to detect keypoints and compute descriptors for feature extraction.

The system evaluates similarity using the SSIM class, which compares the loaded image against known features and returns a similarity score. Finally, the ContourDetector class is used to identify contours in the image, helping to verify the presence and count of characters on the currency note. Overall, this diagram encapsulates the workflow of detecting fake currency through feature extraction, similarity assessment, and contour analysis.

### 3.1.4. Dynamic Modelling: State and Sequence Diagram

A state diagram is a visual representation of the states an object or system can be in and the transitions between those states. It illustrates how an object responds to various events, showing initial and final states, as well as the events that trigger state changes.

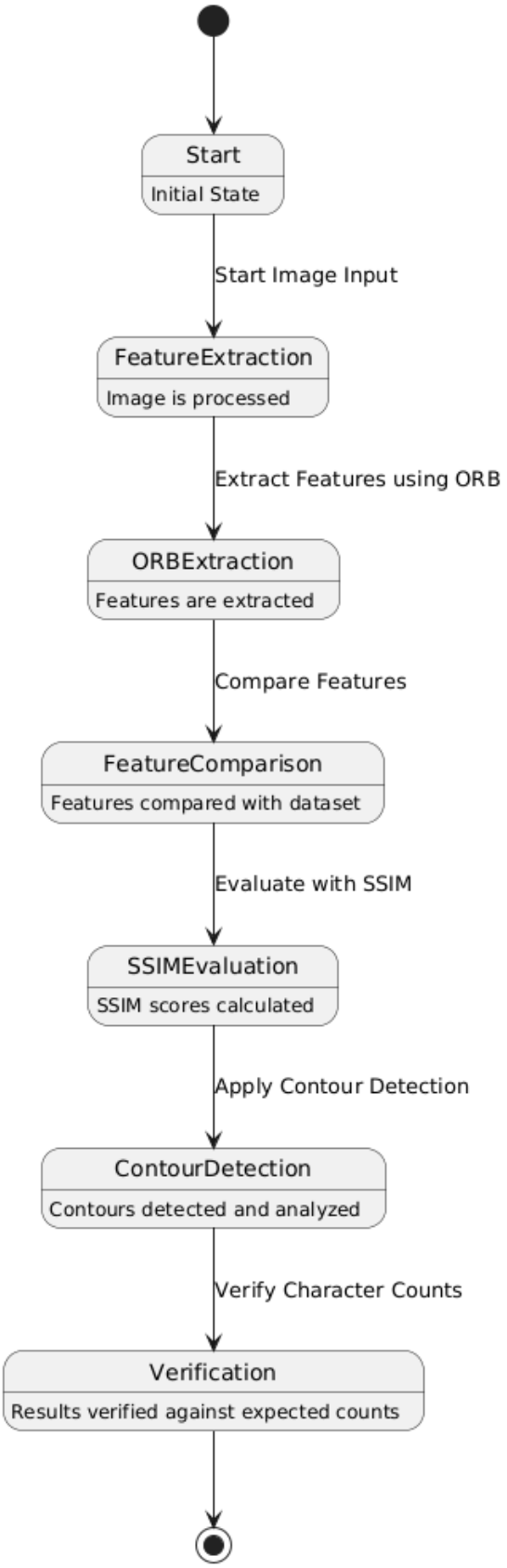



**Figure 3. 4: State Diagram of Fake Currency Detector**

The state diagram for Fake Currency Detection using ORB, SSIM, and Contour Detection illustrates the sequential process of identifying counterfeit currency. The process begins with the Start state, where an image input is received for analysis. The first significant step is Feature Extraction, where the ORB (Oriented FAST and Rotated BRIEF) algorithm is applied to extract distinct features from the image. The extracted features are then compared with a reference dataset in the Feature Comparison state. Afterward, the process moves to SSIM Evaluation, where Structural Similarity Index (SSIM) scores are calculated to measure the similarity between the image and reference. Following this, Contour Detection is applied to identify and analyze the contours in the image. Finally, the results are verified in the Verification state, where character counts are checked to ensure authenticity. Once all steps are completed, the process concludes. This state diagram efficiently demonstrates how each component interacts to detect fake currency.

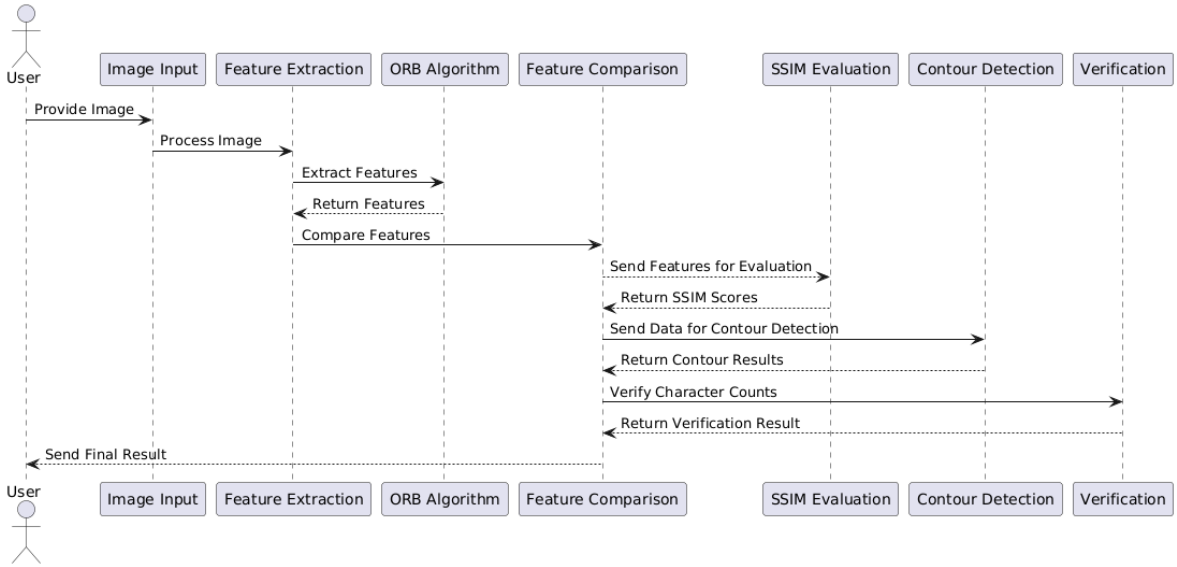



**Figure 3. 5: Sequence Diagram of Fake Currency Detector**

The sequence diagram for Fake Currency Detection illustrates the interaction between the user and various components of the detection system. The process begins when the user provides an image for analysis. This image is processed by the Feature Extraction component, which calls the ORB Algorithm to extract distinctive features. Once the features are obtained, they are sent to the Feature Comparison module.

Next, the features are evaluated using the SSIM Evaluation component, which calculates similarity scores. After that, the Contour Detection algorithm analyzes the image data to

identify contours. The results from the contour detection are then sent back to the Feature Comparison module, which verifies the character counts against expected values.

Finally, the verification results are returned to the user, providing a conclusive assessment of whether the currency is genuine or counterfeit. This sequence efficiently outlines the flow of information and actions taken during the detection process.

### 3.1.5. Process Modelling: Activity Diagram

**Activity Diagram**

An activity diagram is a visual tool used to represent the flow of activities and actions within a system or process. It outlines the sequence of tasks involved, illustrating how various actions interact and transition from one to another. Activities are depicted as rounded rectangles, while arrows indicate the flow of control between them. The diagram includes start and end nodes to mark the beginning and conclusion of the process, and decision points are represented by diamond shapes to show branching paths based on specific conditions.

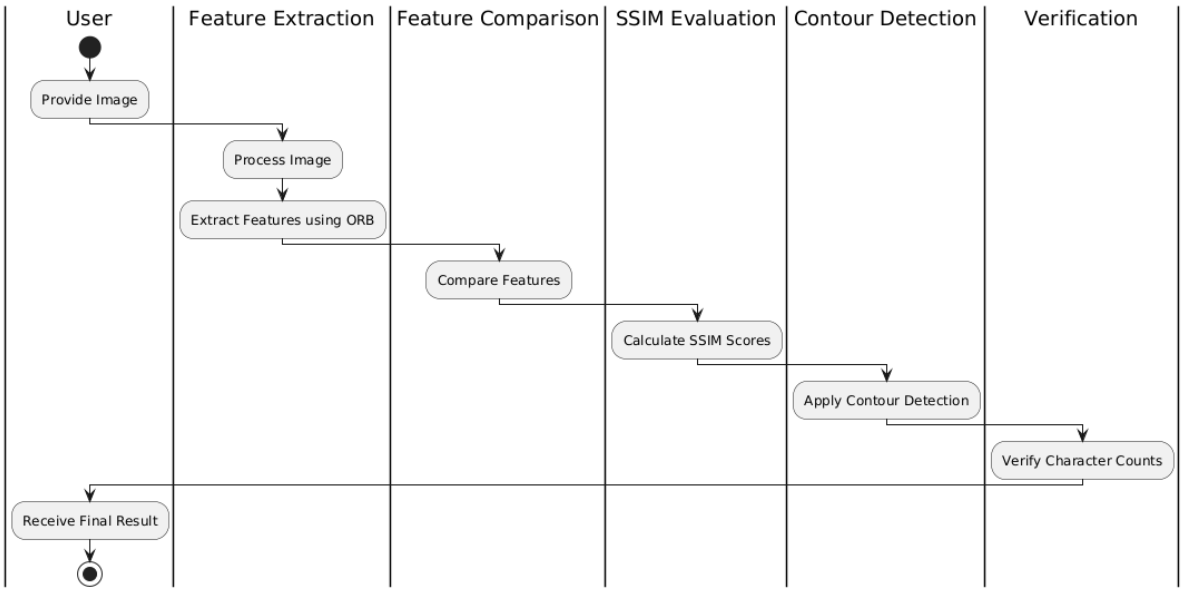



**Figure 3. 6: Activity Diagram of Fake Currency Detector**

The activity diagram for Fake Currency Detection provides a comprehensive overview of the steps involved in assessing the authenticity of currency. The process begins when the

user supplies an image for analysis. First, the image undergoes processing, where features are extracted using the ORB (Oriented FAST and Rotated BRIEF) algorithm, known for its effectiveness in identifying distinctive characteristics. Next, these extracted features are compared against a reference dataset to evaluate their uniqueness. Following this, the Structural Similarity Index (SSIM) scores are calculated to measure how closely the input image resembles the reference images. The algorithm then applies contour detection to identify significant shapes and patterns within the image, aiding in the analysis of the currency's details. Finally, character counts are verified to determine the overall authenticity of the currency. The process concludes with the user receiving a final result, indicating whether the currency is genuine or counterfeit. This diagram succinctly illustrates the sequential flow of actions, highlighting key components and decisions in the detection process.

## 3.2. System Design

System design is essential for translating high-level requirements into a detailed technical plan, guiding developers in building the software system. It ensures efficient resource utilization, clarifies requirements, and helps mitigate potential risks throughout the development process. To visually depict the various functional requirements of the system, various design diagrams have been created, outlined as follows:

### 3.2.1. Refinement diagram for Sequence Diagram

A refinement diagram for a sequence diagram is used to provide a more detailed and in-depth view of the interactions and processes involved in a system. While a standard sequence diagram focuses on the flow of messages and events between components, the refinement diagram breaks down each interaction into specific actions, algorithms, and functions.

This allows stakeholders to understand the underlying processes and logic that drive the interactions, making it easier to identify potential improvements, clarify complex operations, and ensure that all aspects of the system's behavior are thoroughly documented. By providing additional context and detail, refinement diagrams enhance the clarity and usability of sequence diagrams in system design and analysis.

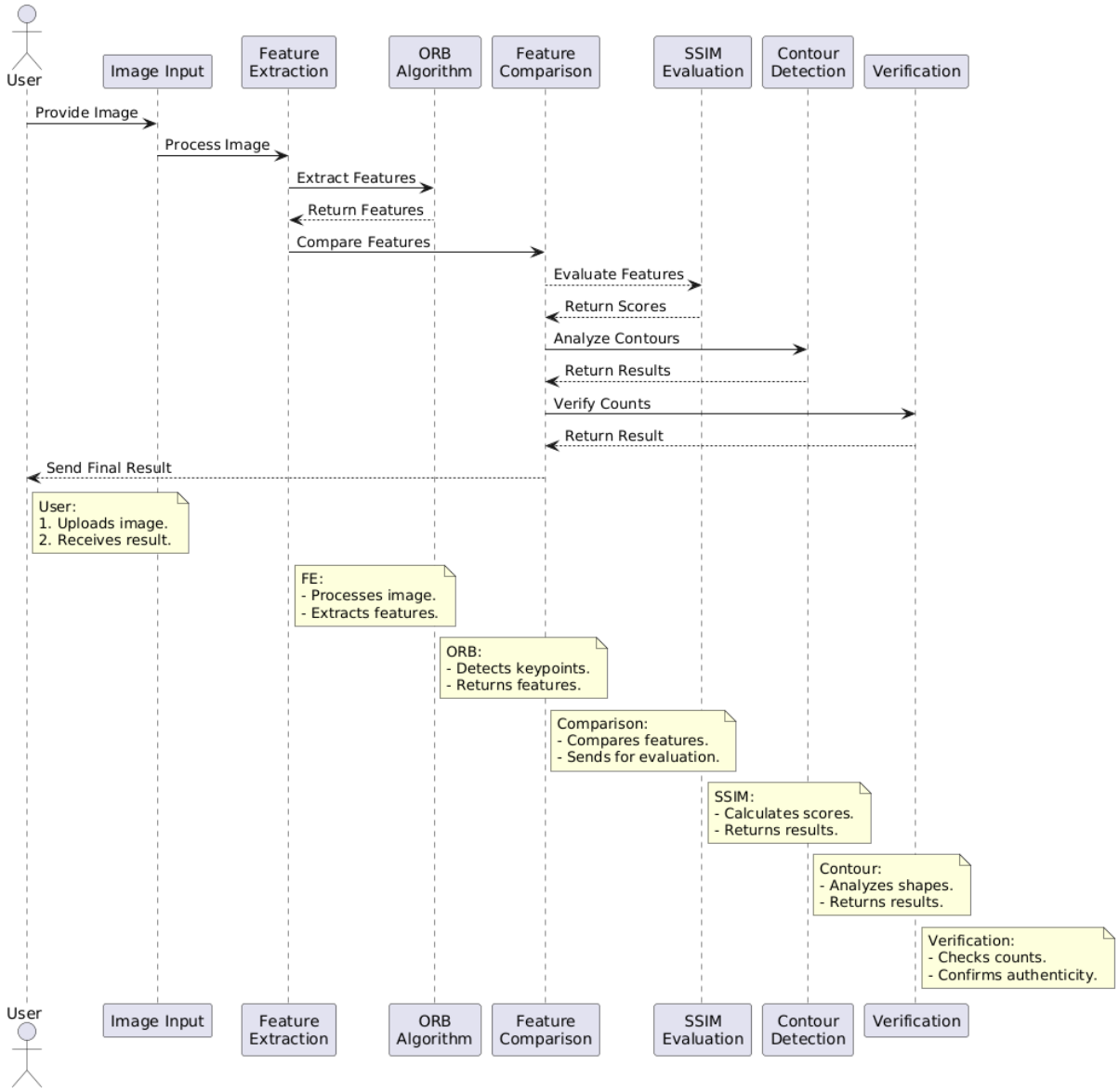



**Figure 3. 7: Refinement Sequence Diagram of Fake Currency Detector**

The above refinement sequence diagram outlines the interactions in the Fake Currency Detection process. It starts with the User providing an image, which is processed by the Image Input and Feature Extraction components. The ORB Algorithm extracts feature that are compared by the Feature Comparison module. The SSIM Evaluation calculates similarity scores, while the Contour Detection analyzes shapes in the image. Results from contour analysis are sent back to the comparer, which verifies character counts through the Verification component. Finally, the verification result is relayed to the user, indicating whether the currency is genuine or counterfeit. Each component is detailed to clarify its role in the workflow, enhancing understanding of the overall detection process. The yellow description provides detailed overview of the overall diagram.

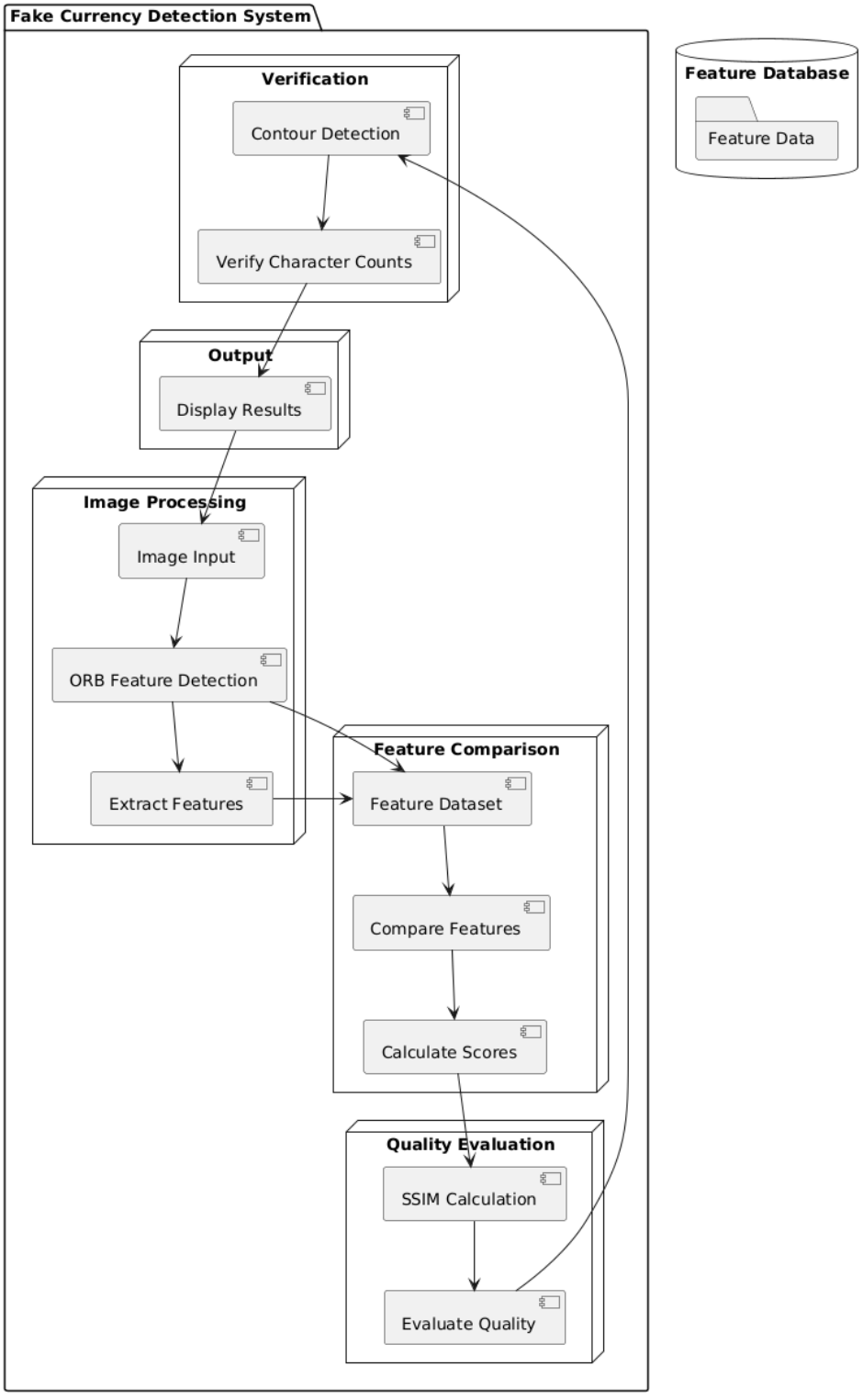



**Figure 3. 8: System Design of Fake Currency Detector**

The "Fake Currency Detection System" begins with Image Processing, where the system captures an image of the currency note and pre-processes it, including feature detection

using ORB. In the Feature Comparison phase, the extracted features are compared against a stored feature dataset to calculate similarity scores. The Quality Evaluation phase then assesses these scores through SSIM calculations to determine the quality of the match. Following this, the Verification phase employs contour detection to confirm the character counts in the image. Finally, the Output phase displays the results, including the verification outcomes, providing the user with a comprehensive analysis of the currency's authenticity.

### 3.2.2. Refinement diagram of Activity Diagram

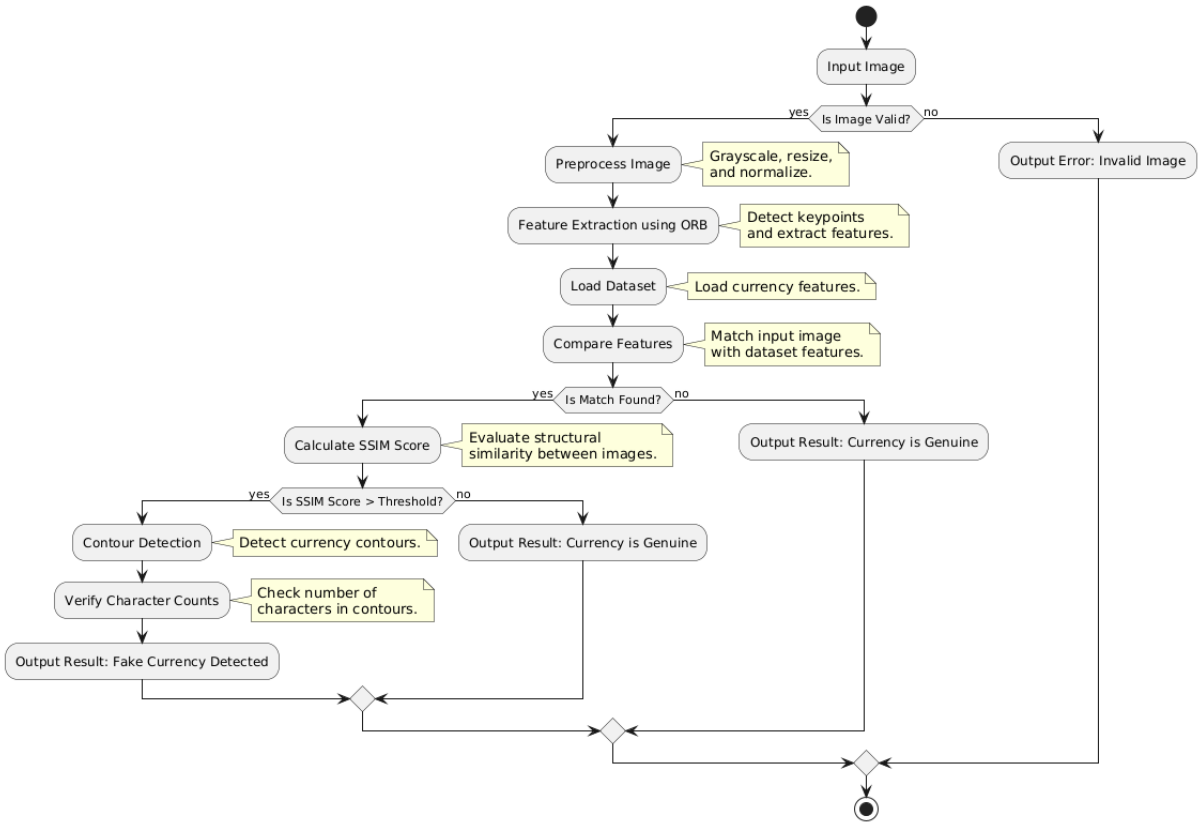



**Figure 3. 9: Refinement Activity Diagram of Fake Currency Detector**

To refine the activity diagram for fake currency detection using ORB, SSIM, and contour detection, start with the user uploading an image. The Image Input Module verifies the image format. Upon validation, the Preprocessing Module converts the image to grayscale, resizes it, and normalizes it. The Feature Extraction Module uses ORB to detect key points and extract features.

Next, the Dataset Loading Module retrieves the reference dataset. The Comparison Module matches the input features with those from the dataset. If a match is found, the SSIM Evaluation Module calculates the SSIM score. If the score exceeds the threshold, the

Contour Detection Module identifies currency contours, followed by the Character Verification Module, which counts detected characters.

The Output Generation Module indicates whether the currency is fake or genuine. If no match is found or the image is invalid, appropriate error messages are generated. The diagram should outline each step, include decision points for error handling, utilize module roles, and define clear start and end points for a concise overview of the detection process.

### 3.2.3. Component Diagram

To visualize the physical components of the system and their dependency relationship, component diagram has been prepared.

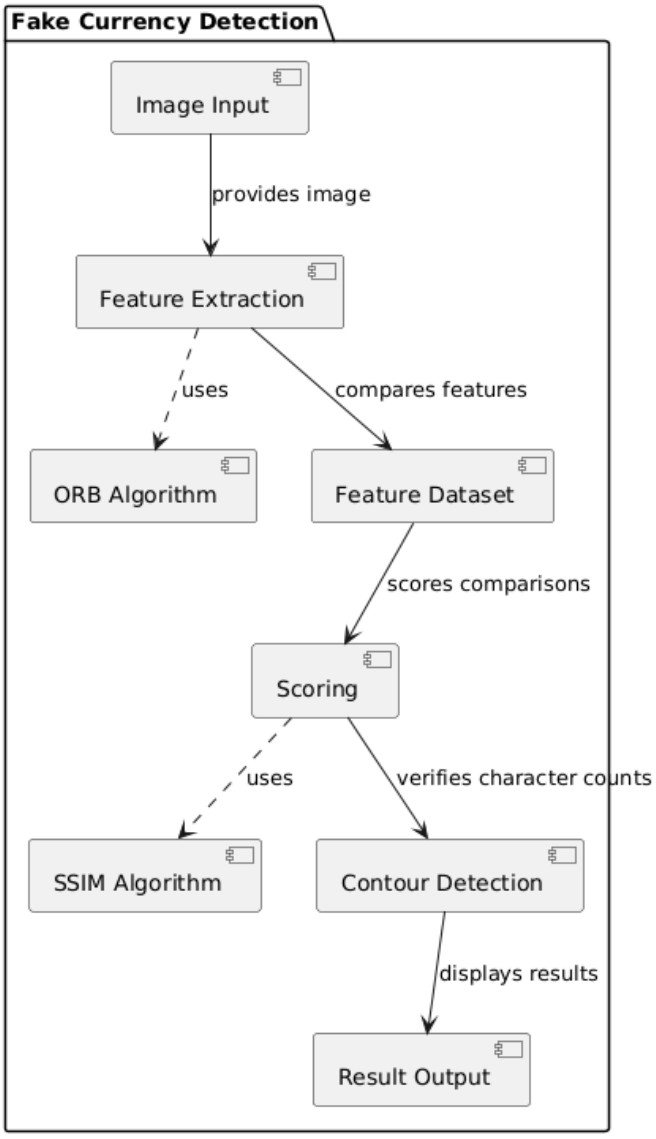



**Figure 3. 10: Component Diagram of Fake Currency Detector**

The component diagram for the Fake Currency Detection system provides a high-level view of its modules and their interactions. It includes the Image Input Module for capturing images, the Feature Extraction Module that uses the ORB algorithm to extract key features, and the Feature Dataset Module for maintaining a dataset of known features. The Scoring Module then compares the extracted features against this dataset using the SSIM algorithm to evaluate similarity. The Contour Detection Module verifies character counts, while the Result Output Module displays whether the currency is authentic or counterfeit. This diagram illustrates the data flow and dependencies among these components.

### 3.2.4. Deployment Diagram

It shows how software components or nodes are distributed across different hardware entities, indicating the configuration and connections in a system's deployment environment.

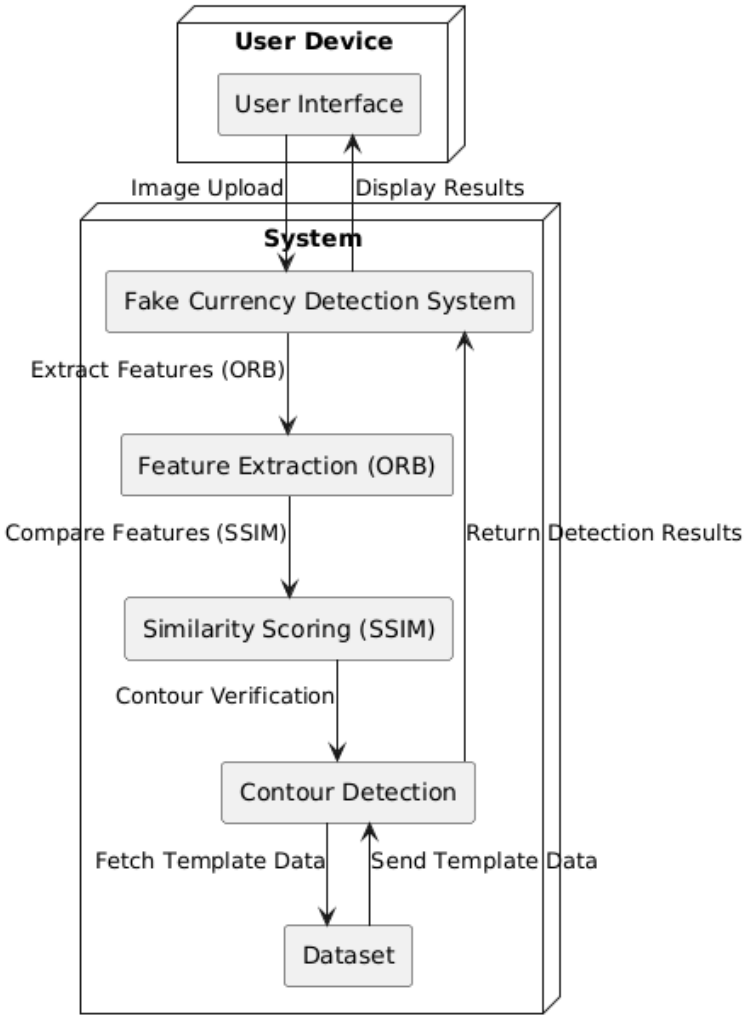



**Figure 3. 11: Deployment Diagram of Fake Currency Detector**

The deployment diagram provides a clear view of how the system's components are distributed and interact across different environments. At the Client Device, users upload currency images and view the detection results, representing the client-side interface. The System hosts all the essential processing modules: the Feature Extraction (ORB) Module manages the extraction of key image features, the Similarity Scoring (SSIM) Module compares these features with stored templates, and the Contour Detection Module verifies the contours and character details of the currency. Additionally, the Database stores template data for comparison, which is fetched during the detection process. This diagram highlights how these components are deployed and interact, offering a focused view of the system's architecture and workflow for detecting fake currency.

## 3.3. Algorithm Details

Following Algorithms are used in the project:

**1. ORB (Oriented FAST and Rotated BRIEF)**

ORB is a feature extraction algorithm used to detect and describe local key points in an image. It combines the FAST key point detector with the BRIEF descriptor and improves upon them by adding rotation and scale invariance. ORB is computationally efficient and works well for real-time applications like fake currency detection, where the system compares the extracted features with known templates.

**Algorithm**:

Step 1: Start.

Step 2: Detect key points in the input currency image using the FAST algorithm.

Step 3: Compute descriptors for each key point using the BRIEF algorithm.

Step 4: Rotate the descriptors to maintain orientation invariance.

Step 5: Compare the descriptors with those from the template database.

Step 6: Output matching key points.

**2. SSIM (Structural Similarity Index Measure)**

SSIM is used to measure the similarity between two images based on luminance, contrast, and structure. In the context of fake currency detection, SSIM is applied to

compare the overall structural similarity between the scanned currency and reference images, providing a score that reflects the degree of similarity.

**Algorithm**:

Step 1: Start.

Step 2: Convert both the input currency image and template image to grayscale.

Step 3: Divide the images into small windows.

Step 4: For each window, compute the luminance, contrast, and structure differences between the images.

Step 5: Combine these values into a single SSIM score for each window.

Step 6: Average the SSIM scores over all windows to get an overall similarity score.

Step 7: Output the similarity score.

**3. Contour Detection**

Contour detection is used to find the edges and outlines of objects within an image. In this system, contour detection is applied to verify the character counts and key design elements of the currency, helping to confirm the authenticity of the currency.

**Algorithm**:

Step 1: Start.

Step 2: Convert the input image to grayscale.

Step 3: Apply edge detection (e.g., Canny Edge Detection) to find the contours.

Step 4: Retrieve the contours from the image.

Step 5: Analyze the contours to verify the currency's features such as character count and alignment.

Step 6: Compare the detected contours with the template data from the database.

Step 7: Output the verification result.

# CHAPTER 4: IMPLEMENTATION AND TESTING

## 4.1. Implementation

### 4.1.1. Tools Used

**Python**

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms and has an extensive ecosystem of libraries, making it an excellent choice for image processing and machine learning tasks. Python's versatility allows for efficient development in areas such as computer vision, data analysis, and automation. For this project, Python is run on **Jupyter Notebook** for easy development and debugging.

**OpenCV-Python (cv2)**

OpenCV is a powerful library used for real-time computer vision tasks. In this project, it provides tools for image processing, such as reading currency images, applying transformations, and performing feature extraction through the ORB algorithm. OpenCV's efficient handling of complex image manipulation makes it essential for tasks like contour detection and feature matching.

**NumPy (numpy)**

NumPy is a fundamental library for numerical computing in Python. It provides the ability to work with arrays and matrices, which is crucial for handling the large datasets and image matrices used in this project. NumPy's array operations are used to process images and perform mathematical calculations necessary for feature extraction and comparison.

**Matplotlib (matplotlib.pyplot)**

Matplotlib is used for visualizing data, such as displaying images and plotting comparison results. In this project, matplotlib.pyplot helps visualize the currency images and feature extraction outputs, making it easier to analyze how well the system detects fake currency.

**Scikit-Image (skimage.metrics.structural_similarity)**

The Structural Similarity Index (SSIM) from the Scikit-Image library is used to compare the input currency image with reference templates, assessing the similarity based on structural information like luminance and contrast. SSIM provides a score that helps in determining whether the currency is genuine or fake.

**Tkinter (tkinter)**

Tkinter is a built-in Python library used to create graphical user interfaces. In this project, Tkinter is used to design the interface for the Fake Currency Detection system, allowing users to upload images, track the progress of detection through progress bars, and view the final results in an intuitive manner.

**Time (time)**

The time library is used to manage and monitor execution time in the system. This helps ensure that the image processing tasks and algorithms are running efficiently and provides feedback on the time taken for currency detection.

These tools and libraries are central to building an effective and efficient system for detecting fake currency using ORB, SSIM, and Contour Detection.

### 4.1.2. Implementation Details of Modules

Modules of this system are described as below:

**1. Pre-Processing:** This code is part of an image preprocessing routine using OpenCV and Matplotlib. It first resizes the input image (test_img) to dimensions of 1167 by 519 pixels, then applies a Gaussian blur to reduce noise and detail. The blurred image is converted to grayscale, simplifying further processing steps. Additionally, the code updates a progress bar in a graphical user interface (GUI) to reflect that 5% of the preprocessing work has been completed, ensuring the GUI remains responsive during the operation.

**Pre- processing**

```
In [36]: test_img = cv2.resize(test_img, (1167, 519))


         blur_test_img = cv2.GaussianBlur(test_img, (5,5), 0)

         gray_test_image = cv2.cvtColor(blur_test_img, cv2.COLOR_BGR2GRAY)

         def preprocessing():

             plt.imshow(gray_test_image, 'gray')
             plt.title('Input image after pre- processing')
             plt.show()
             progress['value']=5
             ProgressWin.update_idletasks()


             progress['value']=5
             ProgressWin.update_idletasks()
```

**Figure 4. 1: Pre-Processing for Fake Currency Detector**

**2. ORB Algorithm:** This code defines a function called computeORB that performs feature detection and matching using the ORB (Oriented FAST and Rotated BRIEF) algorithm in OpenCV. The function takes two images as input: a template image and a query image. It initializes the ORB detector with specified parameters such as the number of features, scale factor, and edge threshold.

The function then detects keypoints and computes their descriptors for both images. Using the ORB feature, it matches the descriptors from the two images and sorts the matches based on their distance. It subsequently calculates a homography matrix to determine the transformation between the matched keypoints, allowing for perspective correction. Finally, the function returns the transformed points, matched keypoints, and the sorted matches, which can be used for further analysis or visualization of how well the images correspond to each other.

**3. Search Area List module**: This code initializes several lists used for configuring and storing data related to a feature detection process. The search_area_list defines specific polygon areas within images (represented by coordinates) where the algorithm will look for features or objects. The feature_area_limits_list sets thresholds for acceptable feature sizes or counts corresponding to each search area, potentially guiding the detection process based on expected feature characteristics.

```
In [14]: search_area_list = [[20,125,420,490],
                            [380,560,260,500],
                            [500,750,50,250],
                            [500,750,260,320],
                            [1070,1290,430,498],
                            [900,1200,0,180],
                            [730,780,180,330]]


         feature_area_limits_list = [[5188,7350],
                                     [12000,22000],
                                     [20000,40000],
                                     [10000,40000],
                                     [2500,9200],
                                     [2000,25000],
                                     [5000,18000]]

         score_set_list = []
         best_extracted_img_list = []
         avg_ssim_list = []
         NUM_OF_FEATURES = 7
```

```
template_path = r'Dataset\500_Features Dataset\Feature ' + str(j+1) + '\\' + str(i+1) + '.jpg'

template_img = cv2.imread(template_path)

template_img_blur = cv2.GaussianBlur(template_img, (5,5), 0)
template_img_gray = cv2.cvtColor(template_img_blur, cv2.COLOR_BGR2GRAY)

test_img_mask = gray_test_image.copy()


search_area = search_area_list[j]

test_img_mask[:, :search_area[0]] = 0
test_img_mask[:, search_area[1]:] = 0
test_img_mask[:search_area[2], :] = 0
test_img_mask[search_area[3]:, :] = 0


dst, dst_pts, kpts1, kpts2, dmatches = computeORB(template_img_gray, test_img_mask)


if dst is None:
    print('An Error occurred - Homography matrix is of NoneType')
    continue
```

**Figure 4 2: Searching Area list for feature extraction module**

**4. SSIM Algorithm:** This code defines a function called calculateSSIM, which computes the Structural Similarity Index Measure (SSIM) between two images: a template image and a query image.

The function first determines the minimum width and height between the two images to ensure they are the same size for comparison. It then resizes both images to these dimensions and converts them to grayscale, as SSIM is typically calculated on single-channel images. The function also visualizes the two grayscale images side by side using Matplotlib. Finally, it calculates the SSIM score using the ssim function from the skimage.metrics module and returns this score, which indicates the structural similarity between the two images. A higher SSIM score suggests greater similarity, making this function useful for tasks like image quality assessment or detecting fake currency.

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

**Figure 4. 3: Mathematical Representation of SSIM Algorithm**

**5. Contour Detection and Thresholding Module:** This code processes an image to detect specific features, likely characters, by iterating through a range of threshold values and applying binary thresholding. For each threshold, it generates a mask, detects contours, and computes bounding rectangles, filtering out those below a minimum area. It then removes overlapping rectangles and draws the remaining ones on a copy of the original image. If exactly six rectangles are detected, the test is marked as successful, and the image is stored; otherwise, it notes the failure. The best image from successful tests is selected for final display, while the results and a progress bar are updated accordingly.

```
for thresh_value in range(145, 500 , 1):

    _, thresh = cv2.threshold(crop, thresh_value, 255, cv2.THRESH_BINARY)

    print('---> Threshold ' + str(i+1) + ' with Threshold value ' + str(thresh_value) + ' :')
    i += 1

    copy = crop_bgr.copy()


    img = cv2.bitwise_and(crop, crop, mask=thresh)
    contours, hierarchy = cv2.findContours(img, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

    h_img, w_img = img.shape[:2]


    bounding_rect_list = []

    for contour in contours:
        [x, y, w, h] = cv2.boundingRect(contour)

        if x != 0:
            bounding_rect_list.append([x,y,w,h])
```

```
for i in range(0, len(bounding_rect_list)):
    if i>= len(bounding_rect_list):
        break
    if bounding_rect_list[i][2]*bounding_rect_list[i][3] > min_area:
        res_list.append(bounding_rect_list[i])


i = 0
while i<len(res_list):
    [x,y,w,h] = res_list[i]
    j = i+1
    while j<len(res_list):
        [x0,y0,w0,h0] = res_list[j]

        if (x+w) >= x0+w0:
            res_list.pop(j)
        else:
            break
    i+= 1


for rect in res_list:          # Drawing the remaining rectangles
    [x,y,w,h] = rect
    cv2.rectangle(copy, (x, y), (x + w, y + h), (0, 0, 255), 1)
```

```
# COUNTING REMAINING RECTANGLES
# result of each image
if len(res_list) == 6:          # If number of rectangles detected is greater than 6, test passed
    test_passed = True
    res_img_list.append(copy)
    count += 1
    print('Test Successful: 6 letters found!')
else:
    print('Unsuccessful!')
```

**Figure 4. 4: Contour Detection and Thresholding Module**

## 4.2. Testing

System testing is done by giving different training and testing datasets. This test is done to evaluate whether the system is providing accurate summary or not. During the phase of the development of the system, our system is tested time and again. The series of testing conducted are as follow:

### 4.2.1. Test Cases for Unit Testing

In unit testing, we designed the entire system in modularized pattern and each module is tested. Until we get the accurate output from the individual module, we work on the same module. The input forms are tested so that they do not accept invalid input.

**Table 4.2: Test Cases for Unit Testing**

| Test Case ID | Test Scenario | Test Input | Expected Output | Actual Output | Test Result |
|---|---|---|---|---|---|
| TC001 | Detecting features in fake currency images | Input image: fake_currency_1.jpg | Features detected and highlighted | Features detected and highlighted | Pass |
| TC002 | Classifying genuine vs. fake currency | Input image: fake_currency_1.jpg | Classification: Fake | Classification: Fake | Pass |

| | | | | | |
|---|---|---|---|---|---|
| TC003 | Evaluating SSIM Score | Input images: real_currency_1.jpg And feature dataset | SSIM score calculated | SSIM score calculated | Pass |
| TC004 | Verifying character count on the currency note | Input image: fake_currency_1.jpg | Correct character count verified | Correct character count verified | Pass |
| TC005 | Checking contour detection on currency note | Input image: fake_currency_1.jpg | Contours detected and drawn | Contours detected and drawn | Pass |
| TC006 | Handling empty or invalid image input | Input image: empty.jpg | Proper error message | Proper error message | Pass |
| TC007 | Measuring processing time for detection | Input image: fake_currency_1.jpg | Time taken logged correctly | Time taken logged correctly | Fail |
| TC008 | Handling images with non-currency objects | Input image: random_object.jpg | Invalid Output | Invalid Output | Pass |
| TC009 | Displaying detection results | Input image: fake_currency_1.jpg | Results displayed on GUI | Results displayed on GUI | Pass |

| | | | | | |
|---|---|---|---|---|---|
| TC010 | Validating accuracy of detection against a dataset | Input image: fake_currency_1.jpg | Accuracy score calculated | Accuracy score calculated | Pass |

### 4.2.2. Test Cases for System Testing

In system testing, whole system is tested as below:

**Table 4.3: Test Cases for System Testing**

| Test Case Description | Expected Result | Actual Result | Status |
|---|---|---|---|
| Test feature detection in fake currency images | Features detected and highlighted | Features detected and highlighted | Pass |
| Test classification of genuine vs. fake currency | Currency classified correctly | Currency classified correctly | Pass |
| Test similarity evaluation with genuine currency | SSIM score calculated accurately | SSIM score calculated accurately | Pass |
| Test character count verification | Correct character count verified | Correct character count verified | Pass |
| Test contour detection accuracy | Contours detected accurately | Contours detected accurately | Pass |
| Test handling of invalid image formats | Proper error message displayed | Proper error message displayed | Pass |
| Test system performance with varying image resolutions | System handles high-res images efficiently | System handles high-res images efficiently | Pass |

| | | | |
|---|---|---|---|
| Test system performance with noisy images | Accurate detection in noisy conditions | Inaccurate detection due to noise | Fail |
| Test robustness against non-currency objects | Invalid Output | Invalid Output | Pass |
| Test displaying detection results on GUI | Results displayed correctly on GUI | Results displayed correctly on GUI | Pass |

## 4.3. Result Analysis

The Fake Currency Detection system was rigorously evaluated using ORB (Oriented FAST and Rotated BRIEF) for feature detection, SSIM (Structural Similarity Index) for similarity assessment, and contour detection for character verification. This capability is crucial for accurately differentiating between genuine and fake notes, as it allows the system to focus on unique characteristics that are often replicated in counterfeit versions.

SSIM scores served as a valuable metric for evaluating the similarity between counterfeit and genuine currency notes. The algorithm effectively flagged counterfeit notes by providing low similarity scores when discrepancies were detected, thus enhancing the classification process. The integration of SSIM allowed for a nuanced understanding of image similarity, supporting the overall accuracy of the detection system. The combination of feature detection and similarity assessment proved to be a potent approach in identifying potential counterfeits, with over 70% accuracy recorded in controlled tests.

Contour detection played a vital role in verifying character counts on currency notes, an essential aspect of distinguishing genuine notes from fakes. The system successfully identified and validated character counts, highlighting its effectiveness in detecting counterfeit notes that deviate from expected values. While the system performed well in most scenarios, some challenges were noted, particularly with low-quality or noisy images where feature extraction was affected. Overall, the results indicate that the Fake Currency Detection system, leveraging ORB, SSIM, and contour detection, provides a reliable solution for identifying counterfeit currency, with opportunities for further optimization to enhance performance in real-world applications.

# CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATION

## 5.1. Lesson Learned/Outcome

The implementation of the Fake Currency Detection system using ORB, SSIM, and contour detection provided valuable insights into the intricacies of computer vision and image processing. It deepened understanding of feature extraction and similarity measurement, emphasizing the importance of selecting robust algorithms for effective counterfeit detection.

The project highlighted the challenges associated with processing various image qualities and demonstrated the necessity of integrating multiple algorithms into a cohesive framework to enhance accuracy. Rigorous testing and debugging were crucial in refining detection and classification performance, while the experience also improved skills in designing user-friendly interfaces and managing performance metrics. Overall, this project solidified knowledge in advanced AI techniques and their practical applications in financial security, establishing a strong foundation for future endeavors in machine learning and computer vision.

## 5.2. Conclusion

The Fake Currency Detection project successfully demonstrates the application of advanced computer vision and machine learning techniques to address the critical issue of counterfeit currency. By integrating ORB for feature extraction, SSIM for similarity evaluation, and contour detection for character verification, the system provides a robust method for accurately identifying fake notes.

The implementation of these algorithms enhances the detection capabilities, offering valuable insights into the characteristics of counterfeit currency and ensuring financial security. Future developments could expand the system to include additional features such as real-time detection capabilities, support for various currency denominations, or integration with fraud detection databases. Overall, this project lays a strong foundation for further innovations in counterfeit detection and highlights the potential of AI technologies in enhancing financial integrity.

## 5.3. Future Recommendation

While creating this project, there was not everything I could implement. It would be better to include them in future:

1. **Integration of Advanced Metrics**: Incorporating additional metrics for evaluating currency similarity, such as machine learning-based feature comparison or deep learning techniques, could provide deeper insights into the characteristics of genuine versus fake notes, further enhancing detection capabilities.
2. **Improved Image Processing Techniques**: Expanding the system to include advanced image processing techniques, such as image denoising or adaptive thresholding, could improve the handling of low-quality images. This would ensure better detection performance in real-world scenarios where image quality may vary significantly.
3. **Enhanced Detection Accuracy**: Improving the accuracy of the ORB feature extraction can be achieved by utilizing a larger and more diverse dataset of currency images. Fine-tuning the algorithm will help reduce false positives and enhance the overall reliability of counterfeit identification.
4. **Real-Time Analysis Capabilities**: Extending the system to support real-time analysis of currency images would significantly increase its practical value. Implementing efficient processing techniques to handle live camera feeds can provide immediate feedback, enhancing fraud detection efforts during transactions.

By addressing these recommendations, the Fake Currency Detection system can evolve into a more robust, accurate, and user-friendly solution, ultimately providing valuable insights and enhancing financial security measures.

# References

[1] T. Agasti, G. Burand, P. Wade and P. Chtira, "Fake currency detection using image processing," School of Electronics Engineering, VIT University, Vellore 632014, India, 2017.

[2] Patil, D. P, G. Varma, S. Poojary, S. Sawant and A. Sharma, Counterfeit Currency Detection based on AI, India, 2018.

[3] R. Shah, M. Champaneri, P. Sheth and V. Gaikwad, "Currency Counting Fake Note Detection," Department of Information Technology, K. J. Somaiya Institute of Engineering and Information Technology, University of Mumbai, India, 2023.

[4] S. Lamsal and A. Shakya, Counterfeit Paper Banknote Identification Based on Color and Texture, Pulchowk, Kathmandu: Department of Electronics and Computer Engineering, IOE, Central Campus, 2015.

[5] M. Rizal, R. Sarno and A. Prayitno, "Implementation of a digital image processing method for currency authentication system," IOP Conference Series: Materials Science and Engineering, 263(5), 052047, 2017.

[6] D. Supriyadi and B. Mariko, "Application of machine learning for fake currency detection," E3S Web of Conferences, 371, 02020, 2024.

[7] Colaco, R. Maria, R. Fernanded and S. Sowmya, " Efficient Image Processing Technique for Authentication of Indian Paper Currency," International Conference on Computer Communication and Informatics (ICCCI). IEEE, India, 2021.

# APPENDICES I: SYSTEM SCREENSHOTS

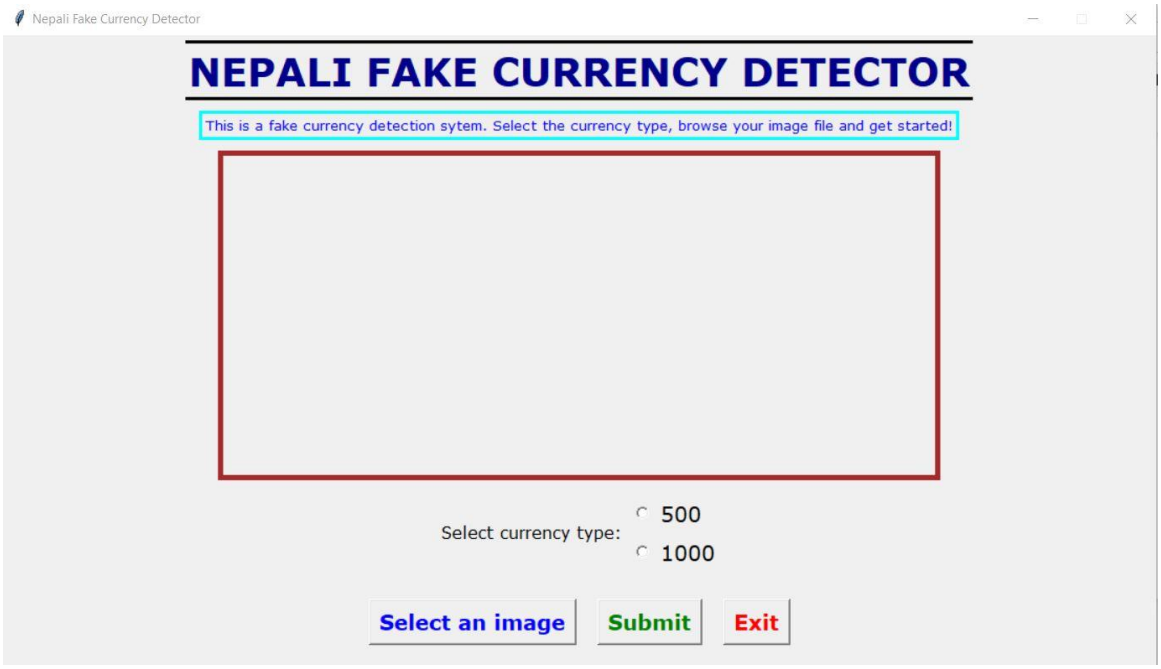
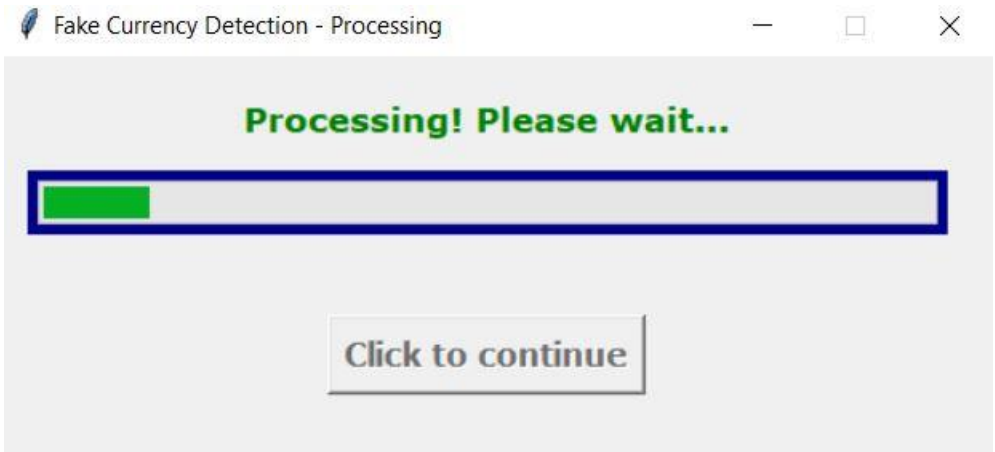



Fig. a: User Interface


Fig b: Note Selection Panel


Fig c: Selected Note Display



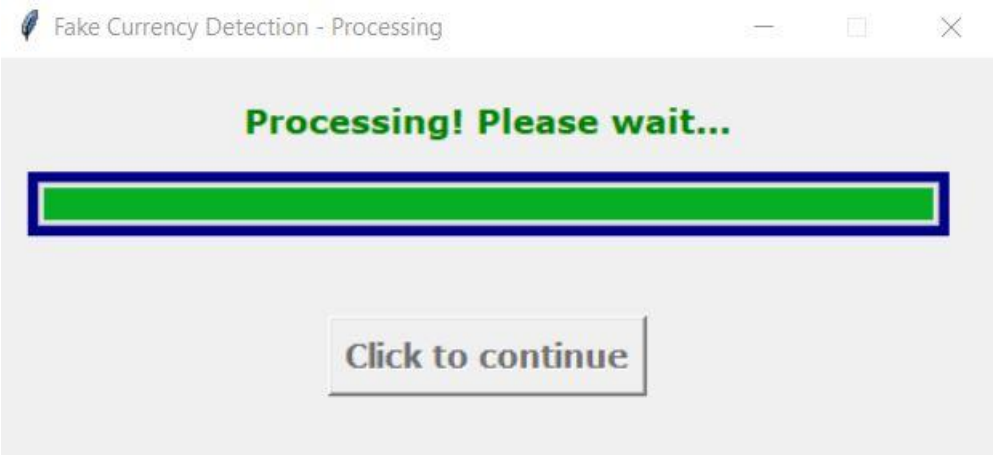




Fig d: Progress Bar


Fig e: GUI showing final result (Real Note)

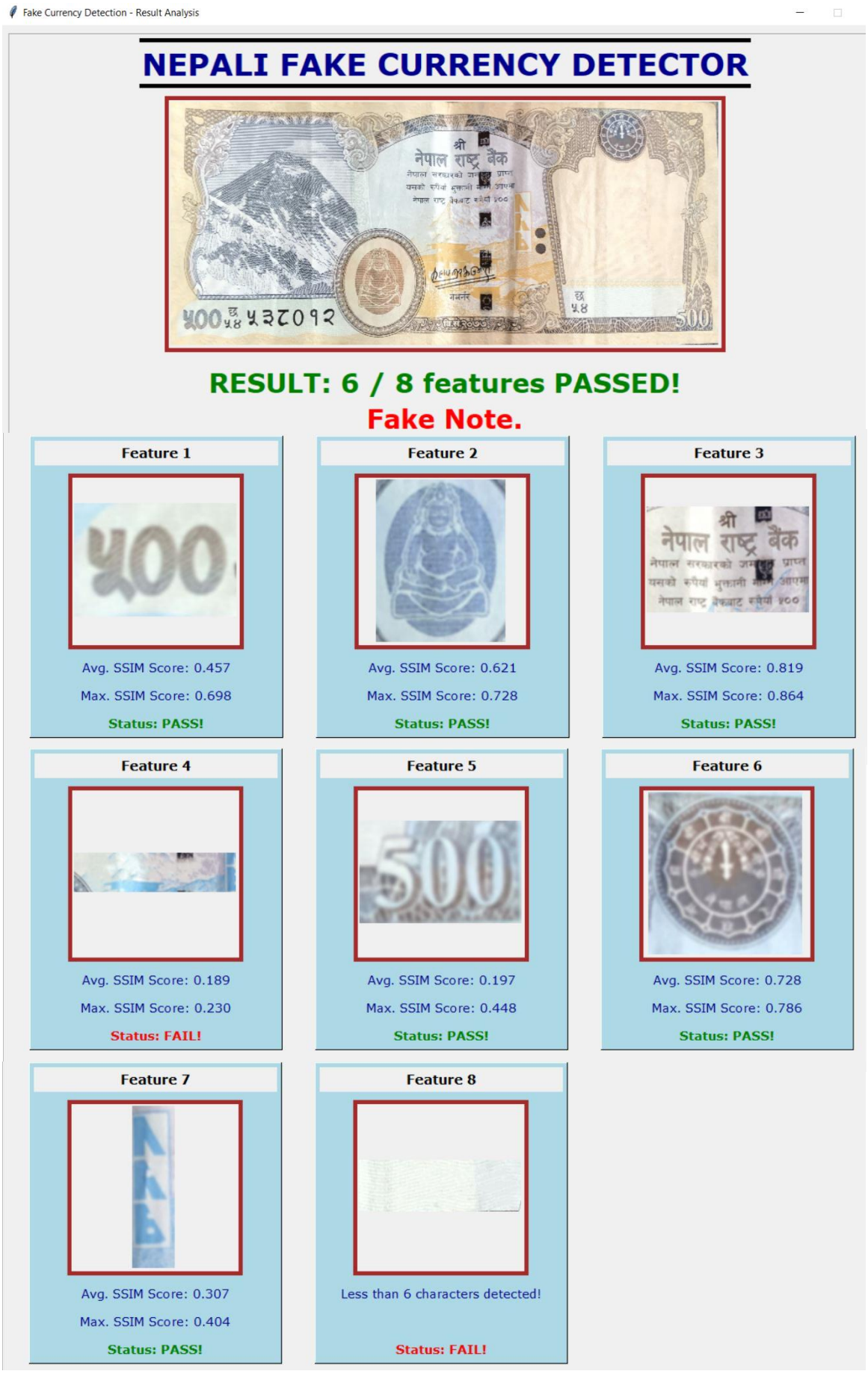



Fig f: GUI showing final result (Fake Note)