Visual Feature Detection System for Counterfeit Indian Currency Notes

CS22B1003 CH Pranay
CS22B1045 S Nikhil
CS22B1055 S Vignesh

1. Introduction

Counterfeit currency is a major issue that affects economies by reducing the value of real money and causing inflation. Manual verification of currency notes is time-consuming and unreliable, making automatic detection essential for efficiency and accuracy. This project proposes a fake currency detection system for Indian ₹500 and ₹2000 notes using image processing techniques. The system uses three algorithms: the first performs steps like image acquisition, preprocessing, grayscale conversion, and feature comparison using ORB and SSIM; the second checks the bleed lines; and the third verifies the number panel. The system provides quick and accurate results, offering a user-friendly and efficient alternative to manual detection.

2. PROBLEM STATEMENT

To test the authenticity of Indian currency notes by preparing a system which takes the image of currency bill as input and gives the final result by applying various image processing and computer vision techniques and algorithms.

A. Objectives

- The main objective of the project is to identify the fake Indian currency notes through an automated system by using Image processing and computer vision techniques
- The system should have high accuracy.
- The system should be able to give the final results in a short time.
- The system should have an User-friendly interface, to make it convenient to use and understand.

3. METHODOLOGY

A. Preparation of Dataset

• The first step is the preparation of a dataset containing images of different currency notes (both fake and real) and images of different features of each of the currency notes

• The dataset will contain the following repositories:

Sub- dataset for Rs. 500 currency notes

1. Images of real notes
2. Images of fake notes
3. Multiple images of each security feature (template)

Sub- dataset of Rs. 2000 currency notes (Similar structure)

The various security features that we are considering are: (for Rs. 500 currency notes- Total 10 features)

- Rs. 500 in Devanagari and English script (2 features)
- Ashoka pillar Emblem (1 feature)
- RBI symbols in Hindi and English (2 features)
- 500 rupees written in hindi (1 feature)
- RBI logo (1 feature)
- Bleed Lines on Left and right side (2 features)
- Number Panel (1 feature)

B. Image Acquisition

Next, the image of the test-currency note is taken as input and fed it into the system. The image should be taken from a digital camera or preferably, using a scanner. The image should have a proper resolution, proper brightness and should not be hazy or unclear. Blurred images and images with less detail may adversely affect the performance of the system.

C. Pre-processing

Next, the pre-processing of the input image is done. In this step, first the image is resized to a fixed size. A fixed size of image makes a lot of computations simpler. Next up, image smoothening is performed by using Gaussian Blurring method. Gaussian blurring removes a lot of noise present in the image and increases the efficiency of the system.

D. Gray- scale conversion

Gray scale conversion is mainly used because an RGB image has 3 channels whereas a gray image has only one channel. This makes the computation and processing on images much more easier in the case of gray scaled images.

E. Algorithm- 1 : For feature 1- 7

1) Feature detection and matching using ORB:

After completing the necessary processing of the image, feature detection and matching is done using ORB. Our dataset already contains the images of different security features present in a currency note

(total 10). Further, we have multiple images of varying brightness and resolutions corresponding to each security feature (6 templates for each feature). Using the ORB algorithm,each security feature is detected in the test image. To make the searching of the security feature (template image) easier and more accurate, a search area will be defined on the test currency image where that template is most likely to be present. Then, ORB will be used to detect the template in the test image and the result will be highlighted properly with a marker. This process will be applied for every image of each security feature present in the data-set and every time the detected part of the test image will be highlighted properly using proper markers.

Fig. 1: ORB Feature detection and Matching

2) Feature Extraction:

Now, using ORB location of each template has been detected in the input image within the highlighted area. The highlighted area is then cropped by slicing the 3D pixel matrix of the image. Next, we apply Gray scaling and Gaussian blur to further smoothen the image and now our feature is ready to be compared with the corresponding feature in our trained model.

Fig. 2: Features in 2000 currency bill

3) Feature comparison using SSIM :

From the previous step, the part of the test currency image which matches with each of the templates will be generated. In this method, the original template will be compared with the extracted feature and then a score will be given for the similarity between the two images using SSIM.

$SSIM(x, y) = (2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2) / (\mu^2_x + \mu^2_y + c_1)(\sigma^2_x + \sigma^2_y + c_2)$

The Structural Similarity Index (SSIM) is a scoring system that quantifies the image quality degradation that is caused by processing such as data compression or by losses in data transmission. Basically it looks for similarity between two images. It is a part of skimage library and uses the above mentioned formula to calculate similarity. It returns a value between -1 and 1. Closer the SSIM to 1, higher is the similarity. So, for every security feature, the SSIM value between each image of that security feature and the corresponding extracted feature from the test image will be calculated. Then, the mean SSIM for each security feature is calculated and stored.

F. Algorithm 2: For feature 8 and 9

Every currency note contains bleed lines near the left and right edges. There are 5 lines in case of 500 currency note and 7 lines in case of Rs. 2000 currency near each of the two sides. This algorithm is being used to count and verify the number of bleed lines present in the left and right sides of a currency note. (feature 8 and 9)

1) Feature Extraction: In the first step, the region in which the bleed lines are present are extracted by cropping the image. So, a part near the left and right edges of the input currency note image is carefully extracted.

2) Image Thresholding: In the 2nd step, the image is thresholded using a suitable value. This ensures that only the black bleed lines remain on a white background and makes further processing quite easy.

3) Calculation of number of bleed lines: The 3rd step involves calculation of number of bleed lines. In this step, first we iterate over each column of the thresholded image. Then we iterate over each pixel of each column. Then, we calculate the number of black regions in each column by increasing a counter whenever current pixel of the column is white and the immediate next pixel is black. Similarly we, count number of black regions for each column, but, if the number of black regions is too large (>= 10), then that column is erroneous and it is discarded. Finally, the average count of black regions is calculated by considering the non- erroneous columns only and the result is displayed as the number of bleed lines. This count should be approximately 5 for Rs 500 currency notes and 7 for Rs 2000 currency notes.

G. Algorithm 3: For feature 10

Every currency note contains a number panel in the bottom right part where the serial number of the currency note is displayed. The number of characters present in the number panel should be equal to 9 (neglecting the space between thecharacters). This algorithm performs various operations and finally counts the number of characters present in the number panel.

H. Image Thresholding (with multiple values):

1. **Image Thresholding (Multiple Values):** The image is thresholded starting from a value (e.g., 90) and increased in steps of 5 until it reaches 150 or detects 9 valid characters. This ensures optimal visibility of the number panel.
2. **Contour Detection:** Contours are detected in the thresholded image to identify character shapes.
3. **Bounding Rectangles:** Bounding boxes are drawn around each detected contour.
4. **Error Elimination:** Rectangles that are too large, too small, nested within others, or too high in the image are removed to filter out noise.
5. **Character Count:** The remaining rectangles are counted as valid characters. The process repeats for each threshold value until the algorithm confirms 9 characters in 3 consecutive iterations or hits the threshold limit.

I. Displaying Output :

Finally, the result of all algorithms is displayed to the user. The extracted image of each feature and the various important data collected for each feature is displayed properly in a GUI window. Further, the status (Pass/ Fail) of each feature is displayed along with the details. Finally the total number of features that have passed successfully for the input image of currency note is displayed and based upon that it is decided whether the note is fake or not. The entire GUI is programmed in python itself using tkinter library.

Fig. 3: Input image of currency note

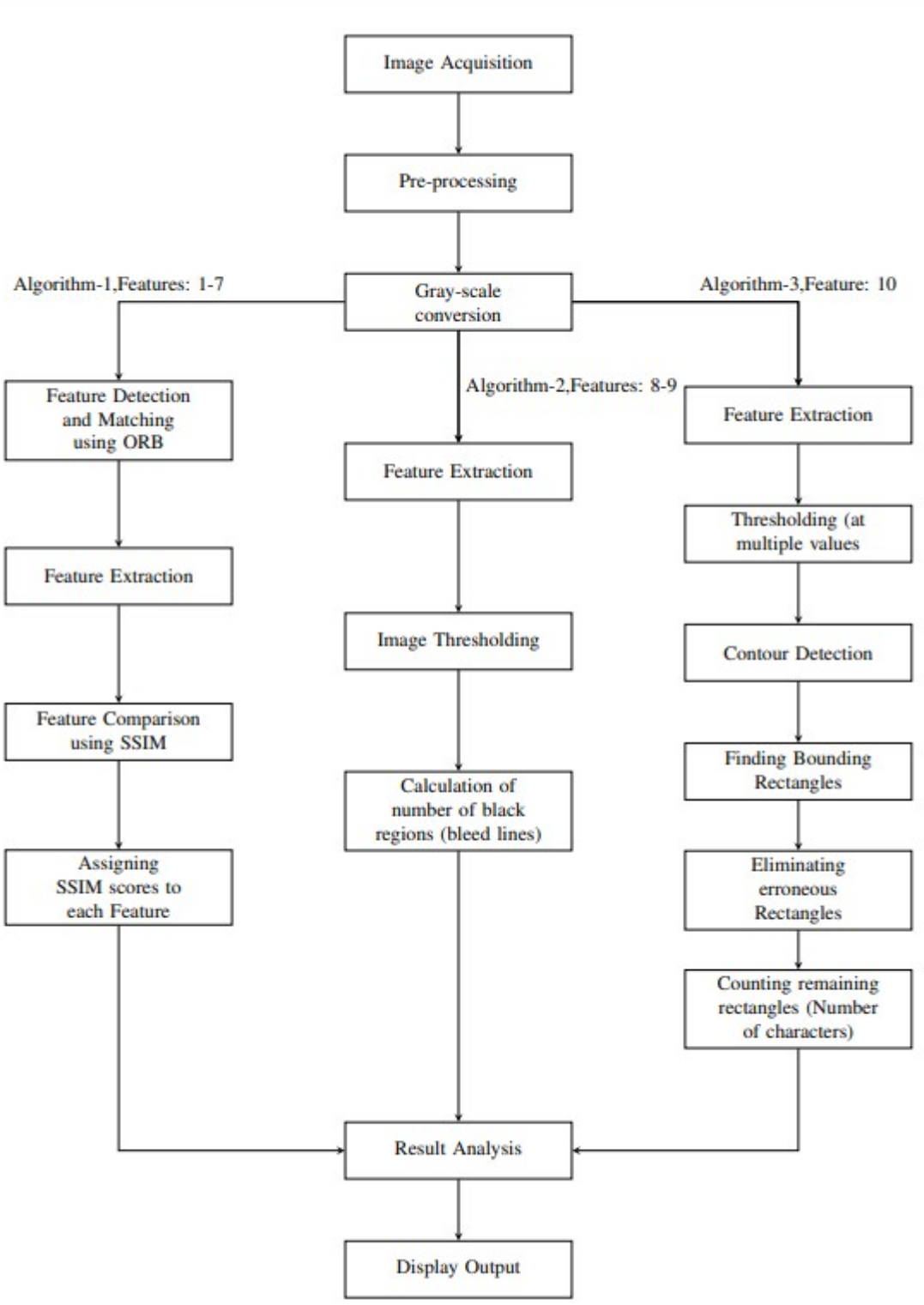

Fig. 4: Flow Diagram

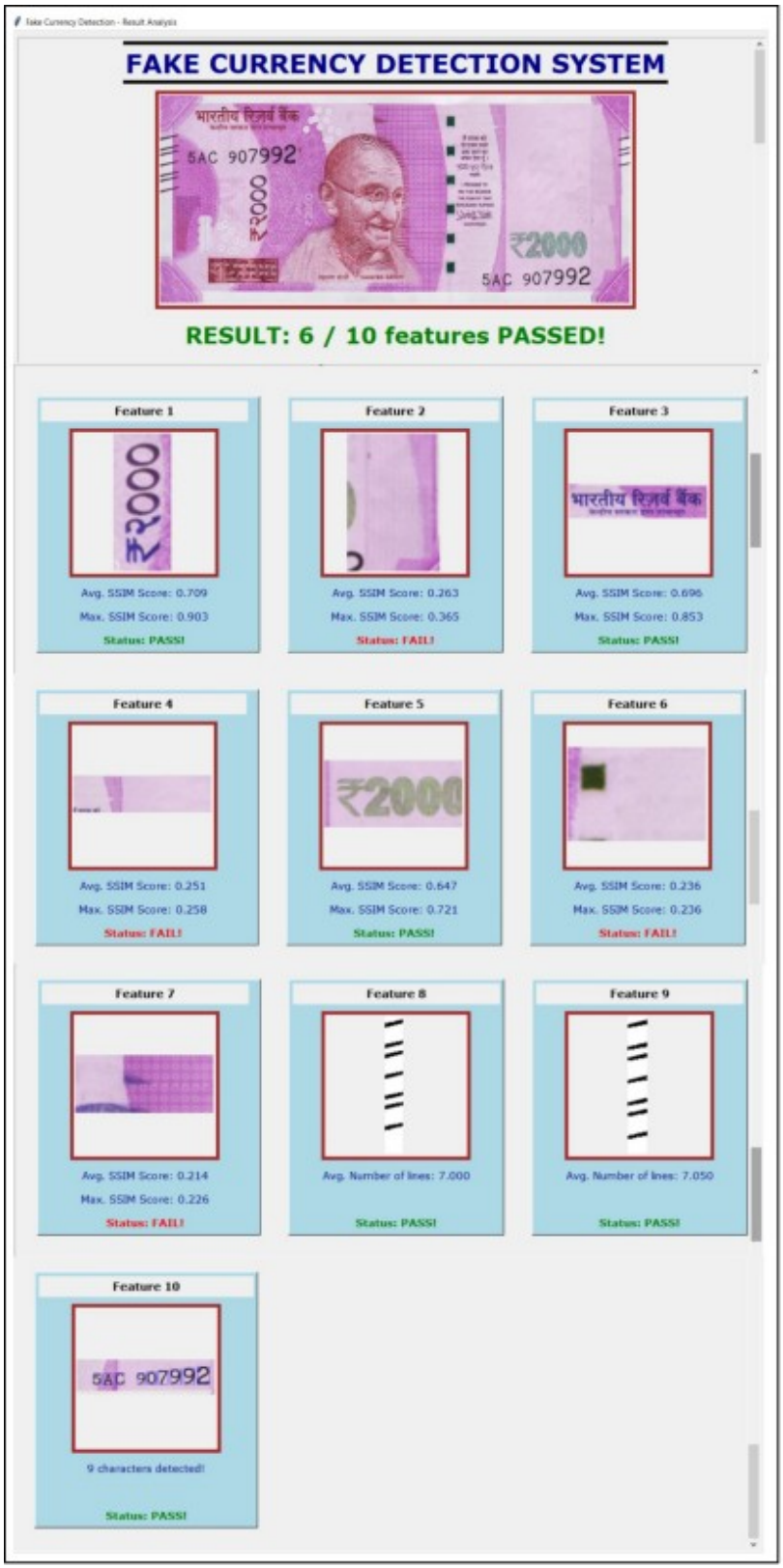

Fig. 5: GUI showing final result(Fake note)

4. RESULTS

The proposed system authenticates the input image of currency note through image processing. The input image passes through various algorithms in which the image is processed and each extracted feature is thoroughly examined. The results are calculated in the following manner:

- **Algorithm 1 (Feature 1-7):**

This algorithm finally collects the average SSIM score and the max. SSIM score for each feature. A feature passes the test and is real if the average SSIM score is greater than a minimum value (this value has to be decided after proper testing). A feature also passes the test if the max. SSIM score is too high (probably greater than 0.8).

- **Algorithm 2 (Feature 8-9):**

This algorithm finally returns the average number of bleed lines present in the left and right sides of a currency note. Each feature passes the test if the average number of bleed lines is closer to 5, in case of Rs 500 currency note, and 7, in case of 2000 currency note.

- **Algorithm 3 (Feature 10):**

This algorithm finally returns the number of characters present in the number panel of the currency note. This feature passes the test if the number of characters detected is equal to 9 (for at least one threshold value).

5. Individual Contribution

Tasks	Pranay	Nikhil	Vignesh
Preparation of dataset	✔	✔	✔
Algorithm 1(SSIM)	✔		✔
Algorithm 2(Image Thresholding)		✔	✔
Algorithm 3(Contour Detection)	✔	✔	✔
GUI		✔	
Elimination of bugs	✔	✔	
Integration of progress bar (GUI)		✔	
Controller notebook (Driver)	✔		✔
Preparation of final report	✔		✔

6. Conclusion

This project demonstrates a practical application of image processing for real-world problems like fake currency detection. Using simple yet effective techniques like SSIM, grayscale conversion, and resizing, we've built an end-to-end system that is functional, scalable, and user-friendly.

7. Code Files and Descriptions

The following files contains the source code files used in the project. Each file is listed and described briefly to provide an understanding of its functionality and purpose in the fake currency detection system:

1. **`test2.ipynb`**
   This notebook is used to analyze and process ₹2000 currency notes. It performs image preprocessing, extracts features, and uses SSIM to compare input notes against a reference database to determine authenticity.
2. **`test1.ipynb`**
   Similar to the `2000_testing` notebook, this file handles ₹500 currency notes. It follows the same methodology using image processing and similarity checks to detect fake notes.
3. **`tk1.ipynb`**
   This notebook provides a Tkinter-based GUI where users can input the currency note image for testing. It includes file selection, input validation, and triggers the appropriate backend functions.
4. **`tk2.ipynb`**
   This file displays the result of the fake currency detection process through a GUI window. It shows whether the note is genuine or fake and displays the similarity score (SSIM).
5. **`main.ipynb`**
   This acts as the main control script. It connects all other notebooks, handles user interaction flow, and ensures the correct functions are called in sequence. It also manages the GUI flow and integrates the progress bar and message updates.

8. Database Attachment

A comprehensive database of Indian currency note images has been included with the project files in the form of a ZIP archive. This dataset is structured to support the fake currency detection system and is divided into categories based on denomination and authenticity.

```
├── 500_dataset/                  # Contains input images of ₹500 notes (genuine/test samples)
├── 500_Features/                 # Cropped feature regions from ₹500 currency notes
├── 500_Features Dataset/         # Feature dataset for ₹500 notes used in similarity matching
├── 2000_dataset/                 # Contains input images of ₹2000 notes (genuine/test samples)
├── 2000_Features/                # Cropped feature regions from ₹2000 currency notes
├── 2000_Features Dataset/        # Feature dataset for ₹2000 notes used in similarity matching
├── fake_500/                     # Images of fake ₹500 notes
├── fake_2000/                    # Images of fake ₹2000 notes
```

Fig. 6: Folder Structre

1. **500_dataset / 2000_dataset**
   These folders contain the original full-sized note images used for testing. The GUI allows users to pick one of these for detection.
2. **fake_500 / fake_2000**
   These include samples of forged or tampered currency notes. These are used to evaluate the system's detection accuracy.
3. **500_Features / 2000_Features**
   These folders contain extracted ROIs (Regions of Interest) from each note – such as watermarks, numerals, and color strips.
4. **500_Features Dataset / 2000_Features Dataset**
   These are refined datasets that include only the cropped feature samples used for SSIM comparison with test inputs.