CURRENCY RECOGNITION AND CONVERSION

Hasaranga Abeywickrama
S/19/303

INTRODUCTION

Develop an image processing system to recognize currency notes from different countries by identifying visual features.

Convert recognized currency into another type by leveraging exchange rates, focusing on visual identification without using machine learning or complex classification models

THE CHALLENGE OF CURRENCY DETECTION

- Multiple currencies worldwide, each with distinct designs and features.
- Traditional detection methods rely on Optical Character Recognition (OCR), which is complex and prone to errors.
- Need for a more efficient and accurate method for detecting currency notes.

APPROACH: TEMPLATE MATCHING

- Template Matching: Using parts of known currency notes (templates) to match with user-uploaded images.
- Objective: Detect the currency note, its value, and convert to local currency (LKR).
- Advantages: Simple, scalable, and avoids the complexity of OCR.

PROGRAM WORKFLOW

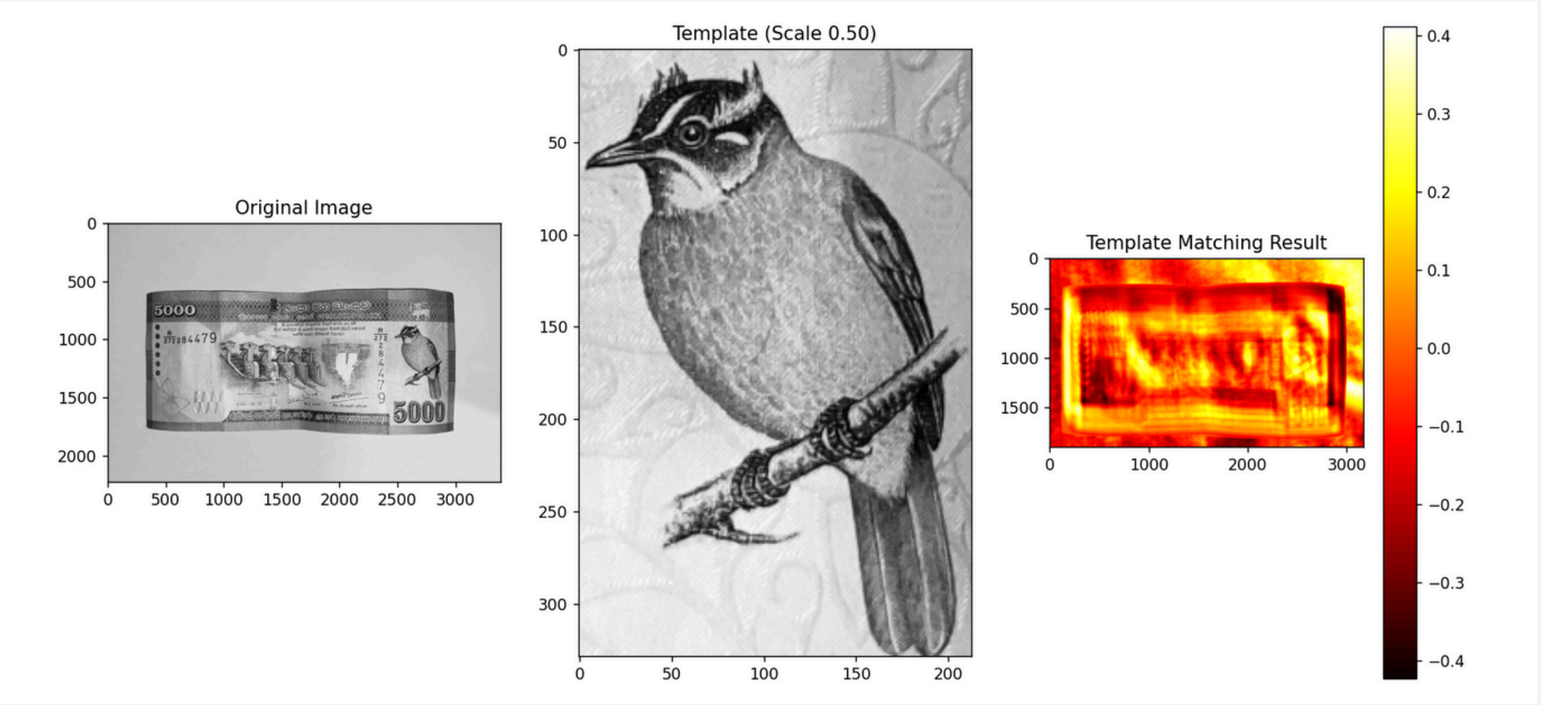
Steps ,

1. Load User Image: Load and preprocess the image uploaded by the user.
2. Template Matching: Compare the user image with stored templates using multi-scale matching.
3. Detection: Identify the best matching template, marking the region on the image.
4. Fetch Exchange Rate: Use an API to get the exchange rate for the detected currency.
5. Output: Display the detected currency, its value, and its equivalent in LKR.

CODE SNIPPET

TEMPLATE MATCHING

```
result = cv2.matchTemplate(image, resized_template, cv2.TM_CCOEFF_NORMED)
(_, max_val, _, max_loc) = cv2.minMaxLoc(result)
```



COMMON CHALLENGES

- Scale Mismatch: User images might differ in scale compared to templates.
- Solution: Implemented multi-scale template matching to account for this.
- Lighting Variations: User images may have different lighting conditions.
- Solution: Using normalized cross-correlation for template matching.

FETCHING EXCHANGE RATES

- Use the ExchangeRate API to fetch live currency exchange rates.
- Show the JSON response and how it’s parsed to extract rates.
- Example snippet:

```python
def get_exchange_rate(currency_code):
    try:
        response = requests.get(f'https://api.exchangerate-api.com/v4/latest/{currency_code}')
        rates = response.json()['rates']
        lkr_rate = rates.get('LKR', None)
        return lkr_rate
    except Exception as e:
        print(f"Error fetching exchange rate: {e}")
        return None
```

RESULTS

```
Detected: LKR 5000 note.
Exchange Rate: 1 LKR = 1 LKR
Value in LKR: 5000 LKR
```

FUTURE IMPROVEMENTS

- More Robust Template Matching: Improve tolerance to rotation and perspective changes.
- More Currency Support: Add support for additional currencies and denominations.
- Mobile App: Extend the project to a mobile application for on-the-go currency detection.
- Improved Performance: Explore GPU acceleration for faster processing.

THANK YOU!