Fake Currency Detection using Image Processing

Vikram Baruah 18BEC0967

Abstract- The advancement of colour printing technology has increased the rate of fake currency note printing and duplicating the notes on a very large scale. Few years back, the printing could be done in a print house, but now anyone can print a currency note with maximum accuracy using a simple laser printer. As a result the issue of fake notes instead of the genuine ones has been increased very largely. India has been unfortunately cursed with the problems like corruption and black money. And counterfeit of currency notes is also a big problem to it. This leads to design of a system that detects the fake currency note in a less time and in a more efficient manner. The proposed system gives an approach to verify the Indian currency notes. Verification of currency note is done by the concepts of image processing. Python has been used to extract the features of the note. The proposed system has got advantages like simplicity and high performance speed. The result will predict whether the currency note is fake or not.

Introduction

There are 50 currencies all over the world, with each of them looking totally different. For e.g. the size of the paper may vary, and also color and pattern may different. People have to remember the features of each currency. This may cause some problems (e.g. wrong recognition), so people need an efficient and exact system to help their work. The aim of our proposed system is to help people who need to recognize their currency, and work with convenience and efficiency. For bank staffs, there is a Currency Sorting Machine which is an electronic device which helps them to recognize currencies. The main working processes of Currency Sorting Machine are image acquisition and recognitions. It has a technique named optical, mechanical and electronic integration, integrated with calculation, pattern recognition (high speed image processing). It is accurate and highly-efficient. But for most staffs, they have to keep a lot of different features and anti-fakes label for different commonly-used currency in their mind.

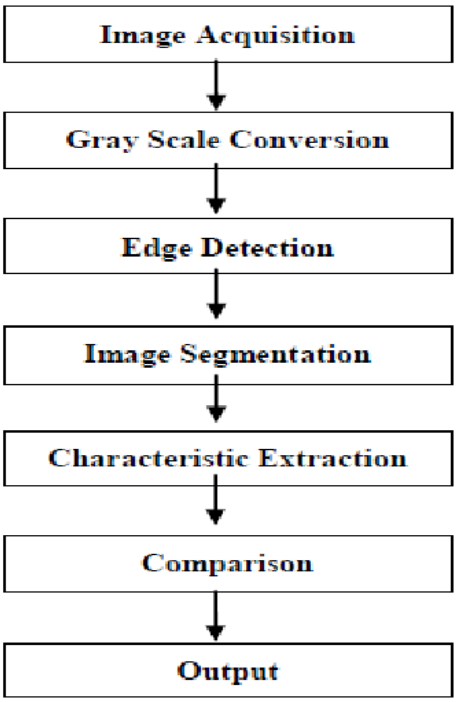
Objective

- To identify original currency note using Image processing techniques.
- System compare images of currency note to the stored images of original currency note images.
- To provide Cheaper and Accurate system to the user which can easily accessible and gives accurate recognition of currency notes.
- To develop user friendly web application of currency recognition system.
- To make available to common people quickly and easily so they can utilize anywhere and at any time.

System Description

- Input (Image Acquisition): A digital camera or scanner or phone is used for image pre-processing. The starting step of the paper currency recognition system would be image segmentation that means separating the note image from the background.
- Browsing: Proposed System browse these images file in the system and these image will be given for feature segmentation and template matching.
- Image processing: It is method to convert an image into digital form and perform some operations on picture or image, in order to obtaining an enhanced image or to extract some useful information from image or picture. Here, we use Template matching for finding small parts of image.
- Template matching: It is a technique in digital image processing for finding small parts of an image which match a template image. It can be used in manufacturing as a part of quality control, a way to navigate a mobile robot, or as a way to detect edges in images.
- Finally, we get output which shows the whether currency is Original or Duplicate. After applying Template matching Algorithm, so person can know whether note is real or fake.

Algorithm- The image is read and resized to desire aspect ratio and stored in the variable, original. The keypoints and descriptors are stored in 2 variables. An array is declared to store the original set of currency note images to which the test data will be compared. Both the original image and test image are compared on the basis of their Euclidian distances and if the distance is less than a threshold then the image is matched and it is real, else the note is fake.



Code

Code for feature extraction-

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
from pprint import pprint

# read image as is
def read_img(file_name):
        img = cv2.imread(file_name)
        return img
# resize image with fixed aspect ratio
def resize_img(image, scale):
        res = cv2.resize(image, None, fx=scale, fy=scale, interpolation = cv2.INTER_AREA)
        return res
# convert image to grayscale
def img_to_gray(image):
        img_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        return img_gray
# gaussian blurred grayscale
def img_to_gaussian_gray(image):
        img_gray = cv2.GaussianBlur(img_to_gray(image), (5, 5), 0)
        return img_gray
# convert image to negative
def img_to_neg(image):
        img_neg = 255 - image
        return img_neg
# binarize (threshold)
def binary_thresh(image, threshold):
        retval, img_thresh = cv2.threshold(image, threshold, 255, cv2.THRESH_BINARY)
        return img_thresh
def adaptive_thresh(image):
        img_thresh = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 8)
        # cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst]) → dsta
        return img_thresh
# sobel edge operator
def sobel_edge(image, align):
        img_horiz = cv2.Sobel(image, cv2.CV_8U, 0, 1)
        img_vert = cv2.Sobel(image, cv2.CV_8U, 1, 0)
        if align == 'h':
                return img_horiz
        elif align == 'v':
                return img_vert
        else:
                print('use h or v')
# sobel edge x + y
def sobel_edge2(image):
        # ksize = size of extended sobel kernel
```

```
        # ksize = size of extended sobel kernel
        grad_x = cv2.Sobel(image, cv2.CV_16S, 1, 0, ksize=3, borderType = cv2.BORDER_DEFAULT)
        grad_y = cv2.Sobel(image, cv2.CV_16S, 0, 1, ksize=3, borderType = cv2.BORDER_DEFAULT)

        abs_grad_x = cv2.convertScaleAbs(grad_x)
        abs_grad_y = cv2.convertScaleAbs(grad_y)

        dst = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)
        return dst
# canny edge operator
def canny_edge(image, block_size, ksize):
        # block_size => Neighborhood size
        # ksize => Aperture parameter for the Sobel operator

        # 350, 350 => for smaller 500
        # 720, 350 => Devnagari 500, Reserve bank of India

        img = cv2.Canny(image, block_size, ksize)
        # dilate to fill up the numbers
        #img = cv2.dilate(img, None)
        return img


# laplacian edge
def laplacian_edge(image):
        # good for text
        img = cv2.Laplacian(image, cv2.CV_8U)
        return img


# detect countours
def find_contours(image):
        (_, contours, _) = cv2.findContours(image, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
        contours = sorted(contours, key = cv2.contourArea, reverse = True)[:5]
        return contours


# median blur
def median_blur(image):
        blurred_img = cv2.medianBlur(image, 3)
        return blurred_img


# dialte image to close lines
def dilate_img(image):
        img = cv2.dilate(image, np.ones((5,5), np.uint8))
        return img
```

```
# calculate histogram
def histogram(image):
        hist = cv2.calcHist([image], [0], None, [256], [0, 256])
        # cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])
        plt.plot(hist)
        plt.show()
# fast fourier transform
def fourier(image):
        f = np.fft.fft2(image)
        fshift = np.fft.fftshift(f)
        magnitude_spectrum = 20 * np.log(np.abs(fshift))

        plt.subplot(121), plt.imshow(image, cmap='gray')
        plt.title('Input Image'), plt.xticks([]), plt.yticks([])

        plt.subplot(122), plt.imshow(magnitude_spectrum, cmap='gray')
        plt.title('FFT'), plt.xticks([]), plt.yticks([])

        plt.show()
# calculate scale and fit into display
def display(window_name, image):
        screen_res = 1440, 900  # MacBook Air

        scale_width = screen_res[0] / image.shape[1]
        scale_height = screen_res[1] / image.shape[0]
        scale = min(scale_width, scale_height)
        window_width = int(image.shape[1] * scale)
        window_height = int(image.shape[0] * scale)

        # reescale the resolution of the window
        cv2.namedWindow(window_name, cv2.WINDOW_NORMAL)
        cv2.resizeWindow(window_name, window_width, window_height)

        # display image
        cv2.imshow(window_name, image)

        # wait for any key to quit the program
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

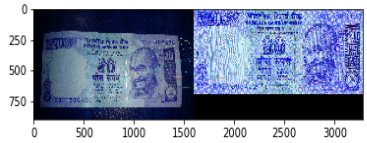
Main code-

```
"""
from utils import *
from matplotlib import pyplot as plt

max_val = 8
max_pt = -1
max_kp = 0
orb = cv2.ORB_create()
test_img = read_img(r'C:\Users\vikra\Desktop\currency-detector-opencv-master\files\500.jpg')
(kp1, des1) = orb.detectAndCompute(test_img, None)
training_set = [r'C:\Users\vikra\Desktop\currency-detector-opencv-master\files\20.jpg',
                r'C:\Users\vikra\Desktop\currency-detector-opencv-master\files\50.jpg',
                r'C:\Users\vikra\Desktop\currency-detector-opencv-master\files\100.jpg',
                r'C:\Users\vikra\Desktop\currency-detector-opencv-master\files\500.jpg']
for i in range(0, len(training_set)):
    train_img = cv2.imread(training_set[i])
    (kp2, des2) = orb.detectAndCompute(train_img, None)
    bf = cv2.BFMatcher()
    all_matches = bf.knnMatch(des1, des2, k=2)
    good = []
    for (m, n) in all_matches:
        if m.distance < 0.789 * n.distance:
            good.append([m])
    if len(good) > max_val:
        max_val = len(good)
        max_pt = i
        max_kp = kp2
    print(i, ' ', training_set[i], ' ', len(good))

if max_val >14:
    print(training_set[max_pt])
    print('good matches ', max_val)

    train_img = cv2.imread(training_set[max_pt])
    img3 = cv2.drawMatchesKnn(test_img, kp1, train_img, max_kp, good, 4)

    note = str(training_set[max_pt])[6:-4]
    print('\nDetected denomination: Rs. ', note)
    (plt.imshow(img3), plt.show())
else:
    print('No Matches')

```

Results-

For real currency-


```
In [26]: runfile('C:/Users/vikra/untitled0.py', wdir='C:/Users/vikra')
0   C:\Users\vikra\Desktop\currency-detector-opencv-master\files\20.jpg   30
1   C:\Users\vikra\Desktop\currency-detector-opencv-master\files\50.jpg   16
2   C:\Users\vikra\Desktop\currency-detector-opencv-master\files\100.jpg   15
3   C:\Users\vikra\Desktop\currency-detector-opencv-master\files\500.jpg   12
C:\Users\vikra\Desktop\currency-detector-opencv-master\files\20.jpg
good matches  30

Detected denomination: Rs.  rs\vikra\Desktop\currency-detector-opencv-master\files\20
```




```
[[<DMatch 000001E40FCA8310>], [<DMatch 000001E40FC76030>], [<DMatch 000001E40FC76670>], [<DMatch 000001E40FC76A30>], [<DMatch 000001E40FC80470>],
[<DMatch 000001E40FC80570>], [<DMatch 000001E40FC80AB0>], [<DMatch 000001E40FCA3CD0>], [<DMatch 000001E40FCA3EF0>], [<DMatch 000001E40FC92530>],
[<DMatch 000001E40FC92730>], [<DMatch 000001E40FC92C30>]]
```

For fake currency-


```
In [30]: runfile('C:/Users/vikra/untitled0.py', wdir='C:/Users/vikra')
0   C:\Users\vikra\Desktop\currency-detector-opencv-master\files\20.jpg   8
1   C:\Users\vikra\Desktop\currency-detector-opencv-master\files\50.jpg   12
2   C:\Users\vikra\Desktop\currency-detector-opencv-master\files\100.jpg   11
3   C:\Users\vikra\Desktop\currency-detector-opencv-master\files\500.jpg   6
No Matches
[[<DMatch 000001E40FCA8090>], [<DMatch 000001E40FCA8450>], [<DMatch 000001E40FFAAE90>], [<DMatch 000001E4100124B0>], [<DMatch 000001E410066EB0>],
[<DMatch 000001E410064CF0>]]
```

CONCLUSION AND FUTURE SCOPE

By using digital image processing, analysis of Currency image is more accurate as well as this method is efficient in terms of cost and time consuming compared to existing techniques. Python is used for this analysis. Day by day research work is increasing in this field and various image processing techniques are implemented in order to get more accurate results. The proposed system is worked effectively for extracting features of Indian currency images. Extracted features of currency image will be using for currency value recognition as well as for its verification. In Future, Application based system shall be designed to get proper result whether currency image is fake or genuine. The same system can be developed for the remaining Indian currency notes and other country's currency notes.

References

[1] Li Liu, Yue Lu ―An Image-Based Approach to Detection of Fake Coins‖ in IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY June 2017.

[2] A Roy ―Machine-assisted authentication of paper currency: an experiment on Indian bank notes‖ International Journal on Document Analysis and Recognition, 18(3): 271-285, 2015.

[3] Sun. Ke, ―Detection of Counterfeit Coins and Assessment of Coin Qualities‖ IEEE Conference 2015.

[4] L. Liu ―Variable-length signature for near-duplicate image matching‖ IEEE Conference 2015.

[5] Jongpil Kim ―Ancient Coin Recognition Based on Spatial Coding‖ IEEE Conference 2015.

[6] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 2nd ed., Prentice Hall India, ISBN- 81-203-2758-6, 2006.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[7] Fake Indian Currency Note [Online]. Available: https://en.wikipedia.org/wiki/Fake_Indian_currency_note

[8] Chinmay Bhurke, Meghana Sirdeshmukh, Prof. Mrs. M.S.Kanitkar,―Currency Recognition Using ImageProcessing‖

International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 5, May 2015

[9] Trupti Pathrabe G and Swapnili Karmore 2011 Int. J. CompTrends Tech 152-156

[10] Eshita Pilania, Bhavika Arora, ―Recognition of Fake Currency Based on Security Thread Feature of Currency International Journal Of Engineering And Computer Science, ISSN: 2319-7242

[11] P. Julia Grace, Ph.D., A. Sheema, ―A survey on Fake Indian Paper Currency Identification System‖ Grace et al., International Journal of Advanced Research in Computer Science and Software Engineering 6(7),July- 2016, pp. 340-345 ISSN: 2277 128X.