Rajiv Gandhi University of Knowledge Technologies

RK Valley, Kadapa-516330

CURRENCY RECOGNITION

A mini project report

Submitted in the partial fulfillment of the requirements for the degree of

Bachelor of Technology in

Computer Science Engineering

By

1) **T BINESH** **R131983**

2) **M RAJINIKANTH** **R131186**

3) **D AMEER** **R131200**

4) **S MAHABOOB BASHA** **R131581**

Under the guidance of

Miss.Arpitha , Mr. Subhash

Department of Computer Science Engineering

Rajiv Gandhi University of Knowledge Technologies,

RK Valley, Kadapa, AP, 516330

CERTIFICATE

This is to certify that this work entitled “CURRENCY RECOGNITION” was successfully carried out by Mr.T BINESH(R131983),Mr.M RAJINIKANTH(R131186),Mr.D AMEER(R131200),Mr.S MAHABOOB BASHA(R131581) in partial fulfillment of the requirements leading to award the credits for mini project in Computer Science Engineering by Rajiv Gandhi University Of Knowledge Technologies during the academic year 2017-2018.

Project Guides

Miss.Arpitha Madam,

MR. Subhash Sir,

Department of CSE,

RGUKT, RK Valley, KADAPA.

ACKNOWLEDGEMENT

The successful completion of my project gives us an opportunity to convey our gratitude to each and every one who has been instrumental in shaping up the final outcome of this report.

We deeply indebted to our guide Miss Arpitha, Mr. Subhash (Faculty, Department of Computer Science Engineering,Rgukt Rkvalley), for valuable guidance and constant encouragement, constructive criticism and keen interest evinced throughout the course for my project work. we are really fortunate to associate myself with such an advising and helping guide in every possible way, at all stages, for the successful completion of my project work.

We would like to express my profound gratitude and sincere thanks to Miss.Arpitha, Mr. Subhash Faculty, Department of Computer Science Engineering , RGUKT_RK valley for providing valuable guidance in choosing project guides.

TABLE OF CONTENTS

ABSTRACT:

This project deals with detection of various currecy notes using image processing's OpenCV library and with the help of ORB(Oriented FAST and rotated BRIEF) algorithm which helps us in object recognition based on keypoints and feature detection points.First of all we are having different kind of currency notes and then test image will be given as input to our system then orb checks this test image with every image and counts the matching points for every image respectively. The image with the highest matching point count will be considered as our result means that is our actual currency note.

INTRODUCTION:

Now a days Everything works with money and currency notes makes more sense about money. At some places like bank large amount of money will be there , so counting of that money may takes more time when we do it with manual process, but if we use digital image processing techniques to do this task in a short span of time and it takes less cost. And

today fake currency is a biggest problem in the country now and it be resolved by using our system.

As we have to use ORB algorithm to achieve this. **ORB (**Oriented FAST and rotated BRIEF**)** is a fast robust local feature detector that can be used in computer vision tasks like object recognition or 3D reconstruction. It is based on the FAST keypoint detector and the visual descriptor BRIEF (Binary Robust Independent Elementary Features). Its aim is to provide a fast and efficient alternative to SIFT. Features from accelerated segment test**(FAST)** is a corner detection method, which could be used to extract feature points and later used to track and map objects in many computer vision task.

LITERATURE SURVEY REVIEW:

First of all we are having different kind of currency notes and then test image will be given as input to our system then orb checks this test image with every image and counts the matching points for every image respectively. The image with the highest matching point count will be considered as our result means that is our actual currency note.

This process is achieved by using **ORB (**Oriented FAST and rotated BRIEF**)** is a fast robust local feature detector that can be used in computer vision tasks like object recognition or 3D reconstruction. It is based on the FAST keypoint detector and the visual descriptor BRIEF (Binary Robust Independent Elementary Features). Its aim is to provide a fast and efficient alternative to SIFT. Features from accelerated segment test**(FAST)** is a corner detection method, which could be used to extract feature points and later used to track and map objects in many computer vision task.

CURRENCY RECOGNITION: Using ORB algorithm

ORB: An efficient alternative to SIFT or SURF in 2011. As the title says, it is a good alternative to SIFT and SURF in computation cost, matching performance and mainly the patents. Yes, SIFT and SURF are patented and you are supposed to pay them for its use. But ORB is not.

ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. First it use FAST to find keypoints, then apply Harris corner measure to find top N points among them. It also use pyramid to produce multiscale-features. But one problem is that, FAST doesn't compute the orientation. So what about rotation invariance? Authors came up with following modification.

EXISTING SYSTEMS:

There are so many Applications are available for currency recognition through image processing but they are developed by using SIFT operator.

Sift Operator:

we saw some corner detectors like Harris etc. They are rotation-invariant, which means, even if the image is rotated, we can find the same corners. It is obvious because corners remain corners in rotated image also. But what about scaling? A corner may not be a corner if the image is scaled. For example, check a simple image below. A corner in a small image within a small window is flat when it is zoomed in the same window. So Harris corner is not scale invariant.

Sift operator is classified into FIVE types:

1. Scale-space Extrema Detection
2. Keypoint Localizatio
3. Orientation Assignment
4. Keypoint Descriptor
5. Keypoint Matching

APPLICATONS:

- Able to Count the total worth of large amount of currency notes(Currency Counting Machine)
- Able to detect Fake Notes based on similarity matches count.
- It is to detect fake notes whenever we have large amount of currency.

ADVANTAGES :

- The drawback in sift is mathematically complicated and computationally heavy.
- In ORB mathematical computations are very easy and fast.
- SIFT is based on the Histogram of Gradients. That is, the gradients of each Pixel in the patch need to be computed and these computations cost time. It is not effective for low powered devices,while in case of ORB algorithm it takes less time cost and it is suitable for low power devices.
-

TECHNOLOGIES:

- Open CV
- Python
- Image Processing

SOURCE CODE:

cur.py:

```
import sys
sys.path
sys.path.insert(0, '/home/rgukt/Desktop/DIP_currency')
from utils import *
from matplotlib import pyplot as plt
import subprocess
from gtts import gTTS
```

```
max_val = 8
max_pt = -1
max_kp = 0
orb = cv2.ORB_create()
#test_img = read_img('files/test_100_2.jpg')
test_img = read_img('files/test_50_2.jpg')
#test_img = read_img('files/test_20_2.jpg')
#test_img = read_img('files/test_100_3.jpg')
#test_img = read_img('files/test_20_4.jpg')
original = resize_img(test_img, 0.4)
display('original', original)


(kp1, des1) = orb.detectAndCompute(test_img, None)
training_set = ['files/20.jpg', 'files/50.jpg', 'files/100.jpg',
'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 != 8:
      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')
```

utils.py:

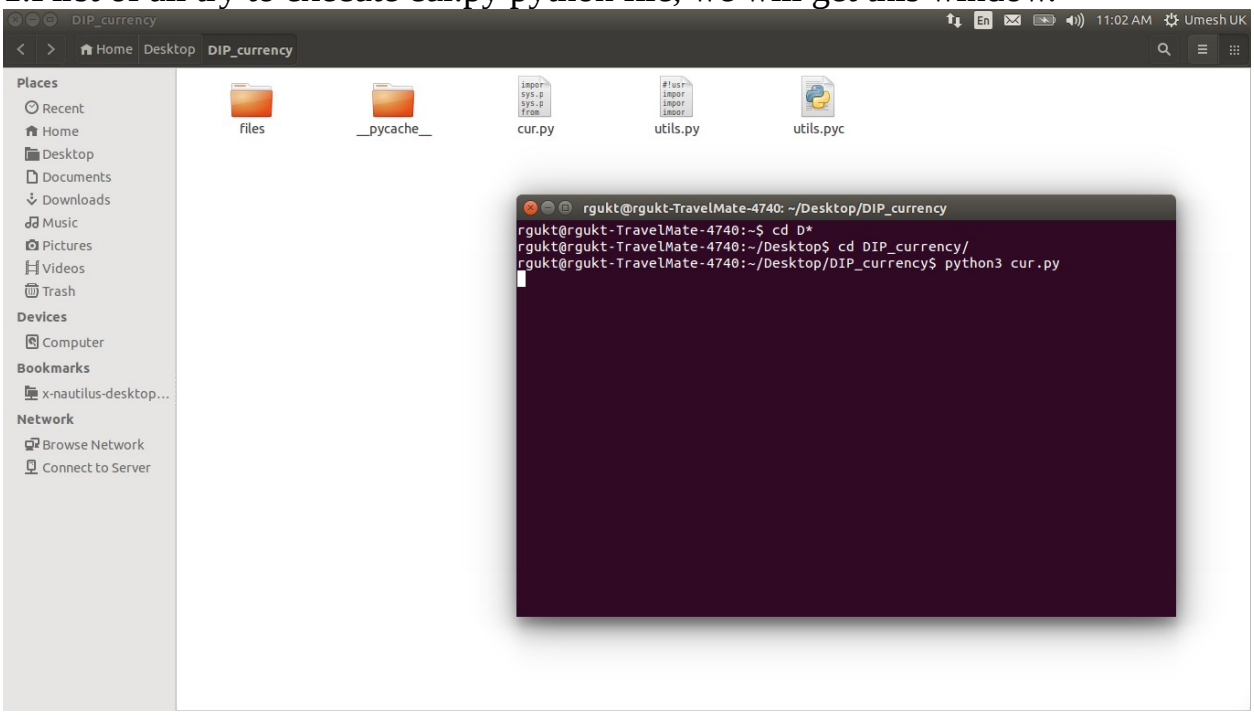
```
#!usr/bin/env python
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
from pprint import pprint
def read_img(file_name):
      img = cv2.imread(file_name)
      return img

def resize_img(image, scale):
      res = cv2.resize(image, None, fx=scale, fy=scale,
interpolation = cv2.INTER_AREA)
      return res

def img_to_gray(image):
      img_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
      return img_gray

def img_to_gaussian_gray(image):
      img_gray = cv2.GaussianBlur(img_to_gray(image), (5, 5), 0)
      return img_gray
def img_to_neg(image):
      img_neg = 255 - image
      return img_neg

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)
	return img_thresh

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')

def sobel_edge2(image):
	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

def canny_edge(image, block_size, ksize):
	img = cv2.Canny(image, block_size, ksize)
	return img

def laplacian_edge(image):
	img = cv2.Laplacian(image, cv2.CV_8U)
	return img

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

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

def dilate_img(image):
	img = cv2.dilate(image, np.ones((5,5), np.uint8))
	return img
```

```
def close(image):
	img = cv2.Canny(image, 75, 300)
	img = cv2.dilate(img, None)
	img = cv2.erode(img, None)
	return img

def harris_edge(image):
	img_gray = np.float32(image)

	corners = cv2.goodFeaturesToTrack(img_gray, 4, 0.03, 200,
None, None, 2,useHarrisDetector=True, k=0.04)
	corners = np.int0(corners)

	for corner in corners:
		x, y = corner.ravel()
		cv2.circle(image, (x, y), 3, 255, -1)
	return image

def histogram(image):
	hist = cv2.calcHist([image], [0], None, [256], [0, 256])
	plt.plot(hist)
	plt.show()

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()
def display(window_name, image):
	screen_res = 1440, 900
	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)
	cv2.namedWindow(window_name, cv2.WINDOW_NORMAL)
	cv2.resizeWindow(window_name, window_width, window_height)
	cv2.imshow(window_name, image)
	cv2.waitKey(0)
	cv2.destroyAllWindows()
```

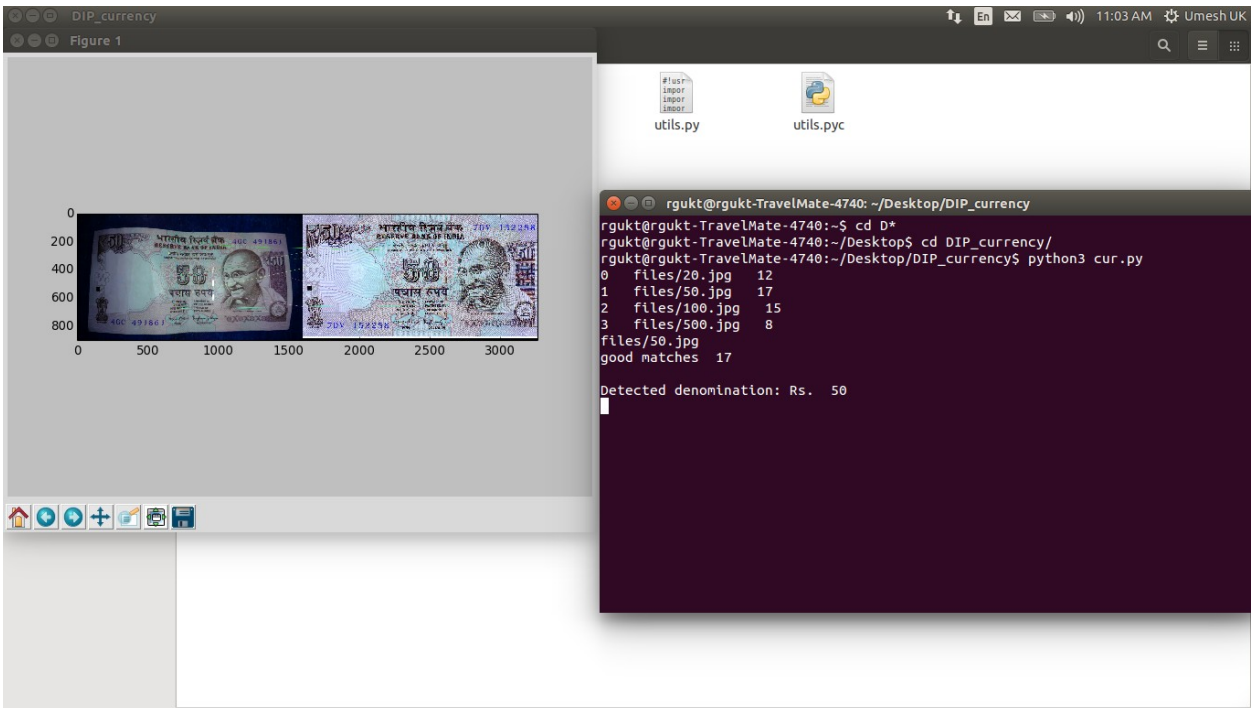
SAMPLE INPUT AND OUTPUT:

1.First of all try to execute cur.py python file, we will get this window.



2.A new test image window will open after excetuing cur.py file , which is our input test image to detect which currency note it is.

3.We will get match counts for every trained data set image and our program will select highest match counted value to detect the currency note.After that a plot will be displayed with the matched currency note along with test image.



CONCLUSION:

We have studied how similarities matches can be counted based on the keypoints and feature detection points which will be the most important criteria to compare two objects and checking the given test currency note is similiar to which training note is similiar to and declare that note as our final output and in near futre we can check whether given note is fake note or not based on the similarity of most points on original note with test note.

REFERENCES:

1. Digital Image Processing (English) 3rd Edition - Buy Digital Image Processing (English) 3rd Edition by Gonzalez, Rafael C|Author;Woods, Richard E|Author
2. *Rublee, Ethan; Rabaud, Vincent; Konolige, Kurt; Bradski, Gary (2011). "ORB: an efficient alternative to SIFT or SURF" (PDF). IEEE International Conference on Computer Vision (ICCV).*
3. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html
4. https://opencv.org/
5. https://anaconda.org/