Identity Verification using shape and Geometry of Human Hand

A Project Report
VII Semester

of

Under Graduate Program

B.Tech. (IT)

under the guidance of

Dr Satish Kumar Singh

Submitted By:
Amit Kumar Gautam
IIT2013200

Signature:

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this project entitled "Identity verification using shape and geometry of human hand", submitted in final evaluation of the seventh semester of Bachelor of Technology (B.Tech) program, in Information Technology at Indian Institute of Information Technology, Allahabad, is an authentic record of my original work carried out under the guidance of Dr. Satish K Singh, due acknowledgements have been made in the text of the project to all other material used. This work was done in full compliance with the requirements and constraints of the prescribed curriculum.

Place: IIIT Allahabad
Date : 19/12/16

Student:
Amit Kumar Gautam
IIT2013200

ACKNOWLEDGEMENT

Author would like to express his sincere gratitude to Dr. Satish Kumar Singh without whose able guidance the on going proceedings of the project would have been a mammoth task. Author would like to specially thank Mr Kush Kumar for his immensive support and motivation. Author would also like to pay his gratitude to people who helped me in whole development phase directly or indirectly.

Table of contents:

Title Page no

Abstract……………………………………………………………….5

Problem Definition………………………………………………6

Introduction…………………………………………………….....6

Literature Survey……………………………………………....8

Methodology…………………………………………………...10

Implementation : Gray Conversion……………………………….…10

Binary Extraction………………………………..10

Contour formation…………..……………………………..11

Valley and Peak extraction…………………………………13

Knuckle Point Extraction and Distance Vector Calculation... 14

Comparison……………………………………15

Software Requirements……………………………………..16

Hardware Requirements…………………………………….16

Proposed Improvements…………………………………….16

References…………………………………………………...17

ABSTRACT:

An identity verification system is implemented using hand shape and geometry. Features of hand are extracted only from the contour hence one image acquisition device is sufficient. A stable reference point is extracted from the contour which is rotation invarient in comparison to centroid of contour. This stable reference point is used for extraction of peak and valley features from contour later. Distance and orientation of each oint of contour is required as shape descriptor. Seven distances are used to encode the geometrical information of the hand. Proposed features and fusion methods are tested over IITD contactless dataset of 240 subjects.

Problem Definition:-

Biometric systems are demand of era as they cannot be easily tricked. They are user friendly too as their are PIN or password free so any user can use it no matter his exposure to technology. Even illiterate people can use biometric systems whereas they may fail in front of security systems with PIN, password or signature.

I am trying to implement a biometric system with lower algorithmic complexity and cost friendly implementation. System will be multimodal for better reliability so that it can be used in identification and recognition too.

Introduction:

Biometrics has gained lot of interest among researchers because of its exponentially growing applications in several areas and its being highly reliable and dependable as far as security is concerned. Different physiological and behavioral traits are used for biometric authentication. Hand based biometrics is considered in this research due to its advantages of low cost, low computational complexity, low template size and more user friendly. Uni-modal systems face major problems of noisy data and intra-class variations (Mansoor, Masood, & Mumtaz, 2011). Such problems can easily be reduced by developing multimodal biometric systems. Hand shape and geometry are easily integrated with different modalities such as latent palm-print (Dai & Zhou, 2011), fingerprint (Cappelli, Ferrara, & Maltoni, 2010), finger geometry (Malassiotis, Aifanti, & Strintzis, 2006), palm-vein (Zhou & Kumar, 2011), Finger-Knuckle-Print (Gao, Yang, Qian, & Zhang, 2014) to develop multimodal biometric systems for various applications of different specifications. Hand geometry (Ross & Jain, 1999) and hand shape (Duta, 2009; Ferrer & Morales, 2011; Su, 2008) have been developed and investigated in the literature. Recently, researchers have reported various new multi-model biometric systems based on the hand geometry, hand shape and their combinations with other biometrics in literature. Hand shape and hand geometry are widely adopted because these can be easily captured with any image capturing device. Moreover, the JUET database used in this paper are created using a simple document scanner. Basically hand shape characterizes the boundary of the hand (i.e. relation of one point with other points at the silhouettes), whereas, the hand geometry characterizes the relation of one part of the hand with other part such as relation between the length of fingers, palm, hand etc. The hand shape and hand geometry only requires the contour of the hand as input. It provides more flexibility to the users and also the number of points o be processed (i.e. only contour points) becomes small which decrease the time complexity of the algorithm. The main problem with the hand shape and hand geometry is that the hand contour can differ too much with the movement of the fingers which is solved in this paper by adopting the finger registration concept.

Biometrics has a gained a lot of interest for researchers because of its increasing scope in various fields. With each day devices are becoming more powerful and their capacity is also increasing therefore biometrics can be applied to more and more places like identification, logging, security, monitoring etc. Systems who use biometric algorithms try to hunt for behaviorial and physiological features. Hand based biometrics can be preferred over other biometric methods because of its low implementation cost, low computational complexity, low template size and user friendly interface. Uni-model systems face major problems with inter class variations therefore multi-modal biometric systems are usually preferred over uni-modal systems as they tend to be more accurate because of many thresholds. In this project user end system consists of raspberry pi with a camera and network connectivity. Camera acquires image of hand and extracts skin region only to derive the contour. This contour is used for peak and valley extraction. As fingerprint patterns are not necessary as feature therefore both contact and contactless images can be used. But contact images are preferred because they are height invariant while contactless image acquisition can show some variation between different captures with respect to height.

Raspberry pi is low end ARM based computing system developed by Raspberry pi foundation whose main intent was to teach kids programming. Device used in this project has 1 GB of RAM and consists of 1.2 Ghz quad-core ARM Cortex A53 processor. It contains 4 USB ports with inbuilt bluetooth and WiFi connectivity.

Python is programming language developed by Google employee Guido Von Rossum and due to its open source nature it has gained a lot of popularity since its release. Due to presence of many frameworks python can be used in image processing, database management, machine learning, system programming, GUI development etc. Numpy framework of python enables to do matrix operations while easygui makes it easier to build GUI applications.

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel’s research center it was later supported by Willow Garage. Currently it is maintained by Itseez. This library is cross-platform and free for use under the open-source BSD license.

Literature Survey:

Use of hand images for verification is demand of the era in which password based systems are being deprecated. Many software products who were doing their identity verification using passwords shifted their algorithm to something else. For example Windows operating system’s locking pattern was based on the password but with Windows 8 onwards Microsoft has face identity verification system which can be used aternatively. Many mobile companies are now launching their mobile handsets which can only be unlocked by fingerprint pattern of user. They have inbuilt fingerprint scanner which scans the finger pattern and matches with stored pattern. Google is also reportedly working on a verification system in which person does not have to enter the password but as soon as he logs in the google, an activation link will be sent to his mobile phone. He will be able to log in that particulkat system only after clicking on the link sent to device. The use of hand images for the biometrics verification has become most popular due to their high user acceptance among the available hand-based biometrics such as fingerprint, palm print, hand geometry, palm vein and finger knuckle (Guo, Zhou & Wang, 2014; Zhang, Zhang, Zhang, & Zhu, 2010). Fotak, Koruga, and Baca (2012) have explored new trends in hand geometry. They presented contact based hand image features as elastic graph using graph theory and intend to make it contactless and combine it with handwritten signature in future. After detailed analysis, they mentioned that hand geometry achieves medium security with benefits of low cost, low computations, low storage requirement and user friendly. Accuracy may get affected due to rings, swelling in fingers unless extra precautions during preprocessing are not taken. Contactless hand geometry is emerging as standard biometric in present time due to ease of use. We worked over both type of datasets contact and contactless dataset. The problems associated with the contactless dataset is that the image may become larger if the hand is near to the captur- ing device and vice versa and also the hand can be placed at any orientation. Some researchers also tackled this problem and worked on the contactless images (De-Santos-Sierra, S\xc3\xa1nchez- Avila, del Pozo, & Guerra-Casanova, 2011; Michael, Connie, Hoe, & Jin, 2010). In De-Santos-Sierra et al. (2011), the authors have used the feature points of the fingers and palms of the hand and reported 2.3% EER, whereas we derived shape features from the hand contour and fused it with the hand geometrical features. Recently, Takeuchi, Manabe, and Sugawara (2013) have utilized the hand shape with the handwriting motion in the air to model a multimodel soft biometric verification system. This method is able to achieve the good verification performance over 10 subjects but it also requires special cameras such as RGB and depth camera which is not desirable for an efficient and low cost person verifica- tion system. Obviously, to form the feature vector from the hand- writing motion requires more computation which restricts its applicability in real time. Whereas, in our case both type of hand features are extracted from the same image and even preprocess- ing step is same for both features which reduces the time complex- ity of the system. The finger nail plates are very first used by the Kumar, Garg, and Hanmandlu (2014) for biometric authentication purpose. This method first finds the nail plate ROI of the Index, Middle and Ring fingers from the hand image and then used Haar wavelet and independent component analysis for the feature extraction. This method has used texture based features of nail plate which is more prone to the noises such as nail polish etc. Whereas, in our method, only contour information is used to find the shape and geometrical features so the effect of noise is removed inherently. Mohd Asaari, Suandi, and Rosdi (2014) have presented the fusion of finger vein and finger geometry to design a multimodel biometric recognition system. This method is able to achieve 1.78% EER while used a single complex imaging device to capture the finger vein and geometry. There are two drawbacks of this method; the first one is the lesser flexibility to the users and second one is with the use of only fingers which cannot be more discriminative as compared to the full hand of the person, moreover, the spoofing in fingers is also easy. Still the researchers are trying to incorporate the finger based biometric system for person authentication (Kumar, 2014).

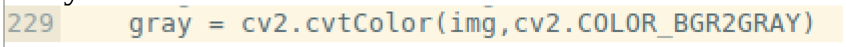
METHODOLOGY -

We will do the process for image matching as follows

1. Preprocessing

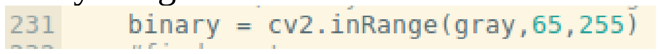
In this part we did some preprocessing on the image before using it in algorithm for matching. First we will convert the image in grayscale format. After that using some threshold value we will convert the image into a binary image which will consist of only balck n white colour. The reigon of fingers,thumb and palm will be in white colour and the rest part of the image will be in black colour because we dont want to use any type of noise and background of hand image. For implementation we used threshold value 65 to convert garayscale image to binary image.

Gray conversion



```
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

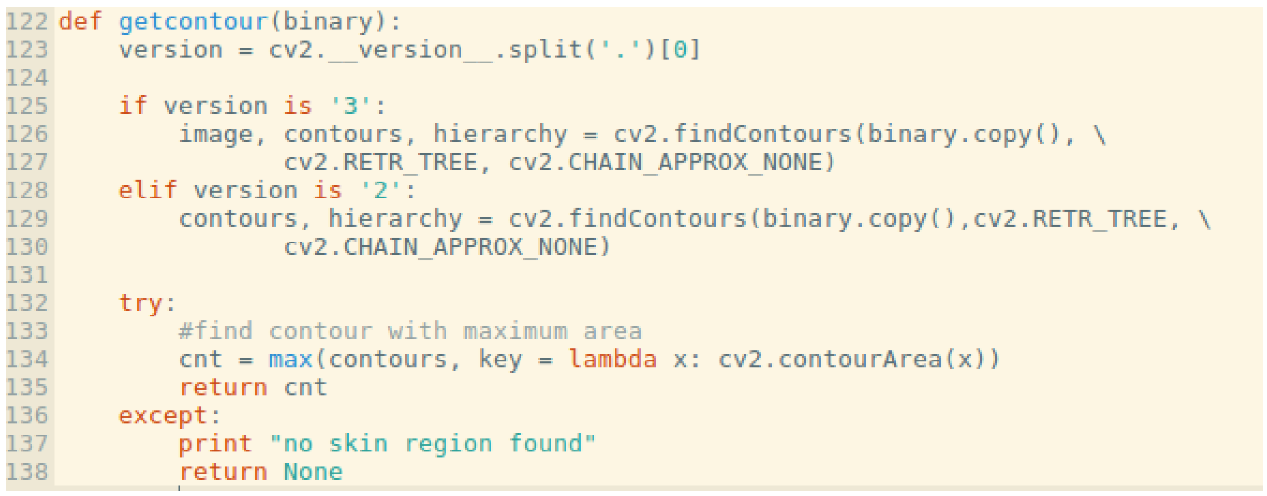
Binary image creation



```
    binary = cv2.inRange(gray,65,255)
```

2. Contour Extraction

After converting the image into a binary image we exctracted the contour using open cv.

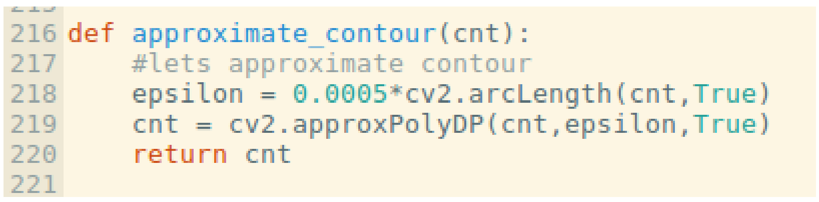


```
def getcontour(binary):
    version = cv2.__version__.split('.')[0]

    if version is '3':
        image, contours, hierarchy = cv2.findContours(binary.copy(), \
                cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
    elif version is '2':
        contours, hierarchy = cv2.findContours(binary.copy(),cv2.RETR_TREE, \
                cv2.CHAIN_APPROX_NONE)

    try:
        #find contour with maximum area
        cnt = max(contours, key = lambda x: cv2.contourArea(x))
        return cnt
    except:
        print "no skin region found"
        return None
```

We also handled the version of opencv used for contour exctraction because in open cv2 and opencv3 the method used for contour extraction is different.

3. Contour Approximation

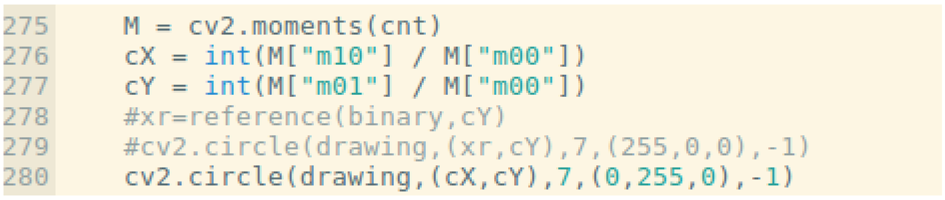
After contour exctraction we approximate the contour because the contour on the finger tips wasn’t smooth.



```
def approximate_contour(cnt):
    #lets approximate contour
    epsilon = 0.0005*cv2.arcLength(cnt,True)
    cnt = cv2.approxPolyDP(cnt,epsilon,True)
    return cnt

```

4. Generating Moment Of contour and Exctraction Of Centroid Co-Ordinates

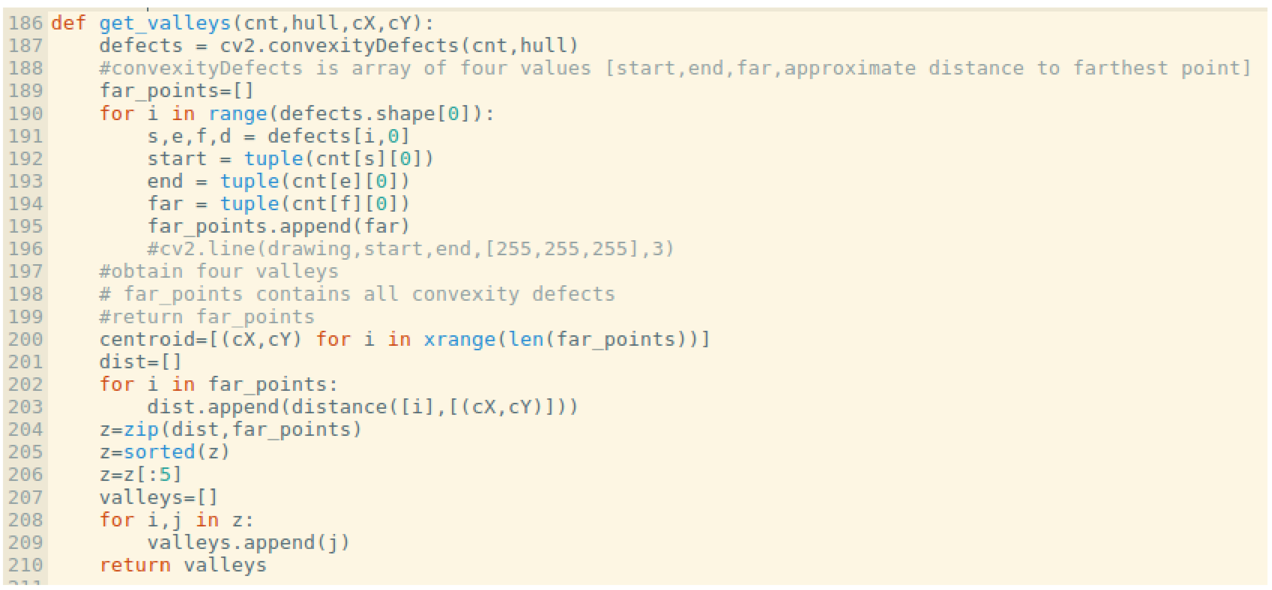
After contour approximation we find the moment of contour



```
    M = cv2.moments(cnt)
    cX = int(M["m10"] / M["m00"])
    cY = int(M["m01"] / M["m00"])
    #xr=reference(binary,cY)
    #cv2.circle(drawing,(xr,cY),7,(255,0,0),-1)
    cv2.circle(drawing,(cX,cY),7,(0,255,0),-1)
```

5. Getting Valley Co-Ordinates

we used convex hull for valley point detection.as we have alreay found the contour. We used convex hull and contour for valley point extraction. For valley point detection we used convexity deffect between convex hull and and the contour we created. We get 5 valley points using this.

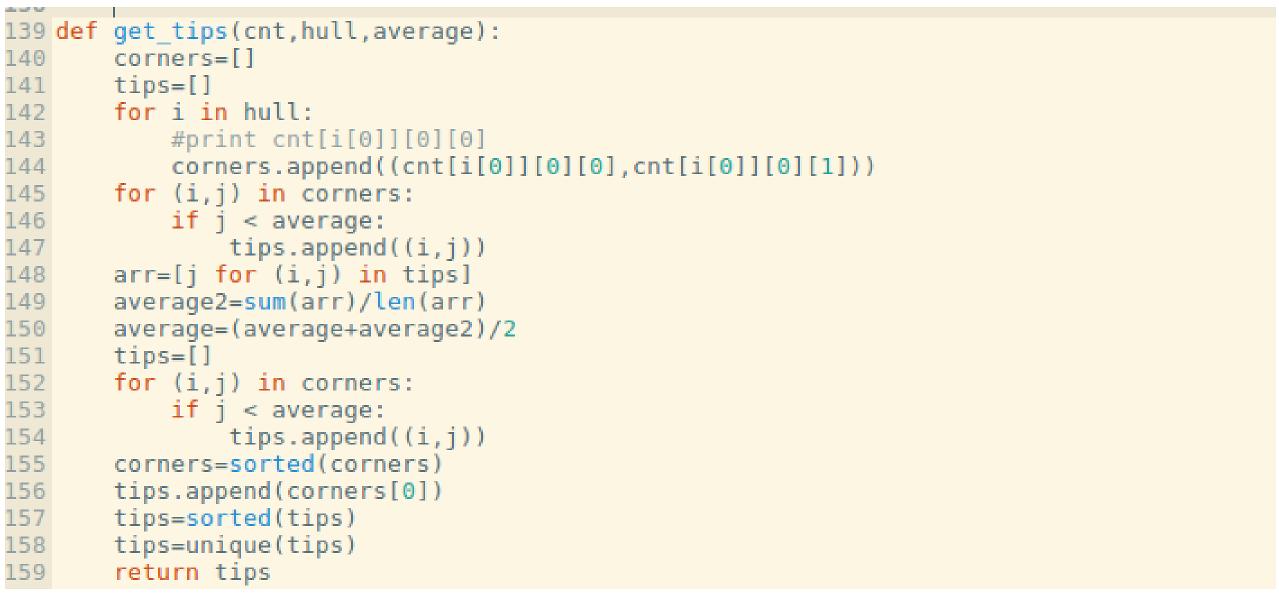


```
def get_valleys(cnt,hull,cX,cY):
    defects = cv2.convexityDefects(cnt,hull)
    #convexityDefects is array of four values [start,end,far,approximate distance to farthest point]
    far_points=[]
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])
        far_points.append(far)
        #cv2.line(drawing,start,end,[255,255,255],3)
    #obtain four valleys
    # far_points contains all convexity defects
    #return far_points
    centroid=[(cX,cY) for i in xrange(len(far_points))]
    dist=[]
    for i in far_points:
        dist.append(distance([i],[(cX,cY)]))
    z=zip(dist,far_points)
    z=sorted(z)
    z=z[:5]
    valleys=[]
    for i,j in z:
        valleys.append(j)
    return valleys
```

6. Getting Tip Points Co-Ordinates

after exctracting tip points. We also get the tip points. Using the centroid and the average of valley points. We first average the all valley points. Which lie among the valley points. After this we calculated the distance between centroid and the average valley point. For each tip point extraction we addthis distance to the valley point to a new point . Now all the points on convex hull which are above the new point we got. Now all these points will be tip point of fingers

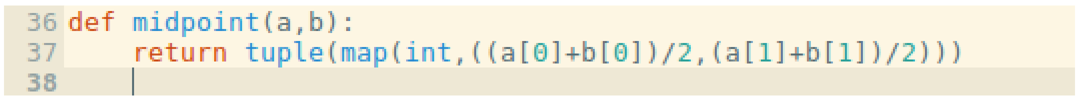
for extraction of the tip point of thumb we will find the nearest point to the new point on the convex hull downward side.



```
def get_tips(cnt,hull,average):
    corners=[]
    tips=[]
    for i in hull:
        #print cnt[i[0]][0][0]
        corners.append((cnt[i[0]][0][0],cnt[i[0]][0][1]))
    for (i,j) in corners:
        if j < average:
            tips.append((i,j))
    arr=[j for (i,j) in tips]
    average2=sum(arr)/len(arr)
    average=(average+average2)/2
    tips=[]
    for (i,j) in corners:
        if j < average:
            tips.append((i,j))
    corners=sorted(corners)
    tips.append(corners[0])
    tips=sorted(tips)
    tips=unique(tips)
    return tips
```

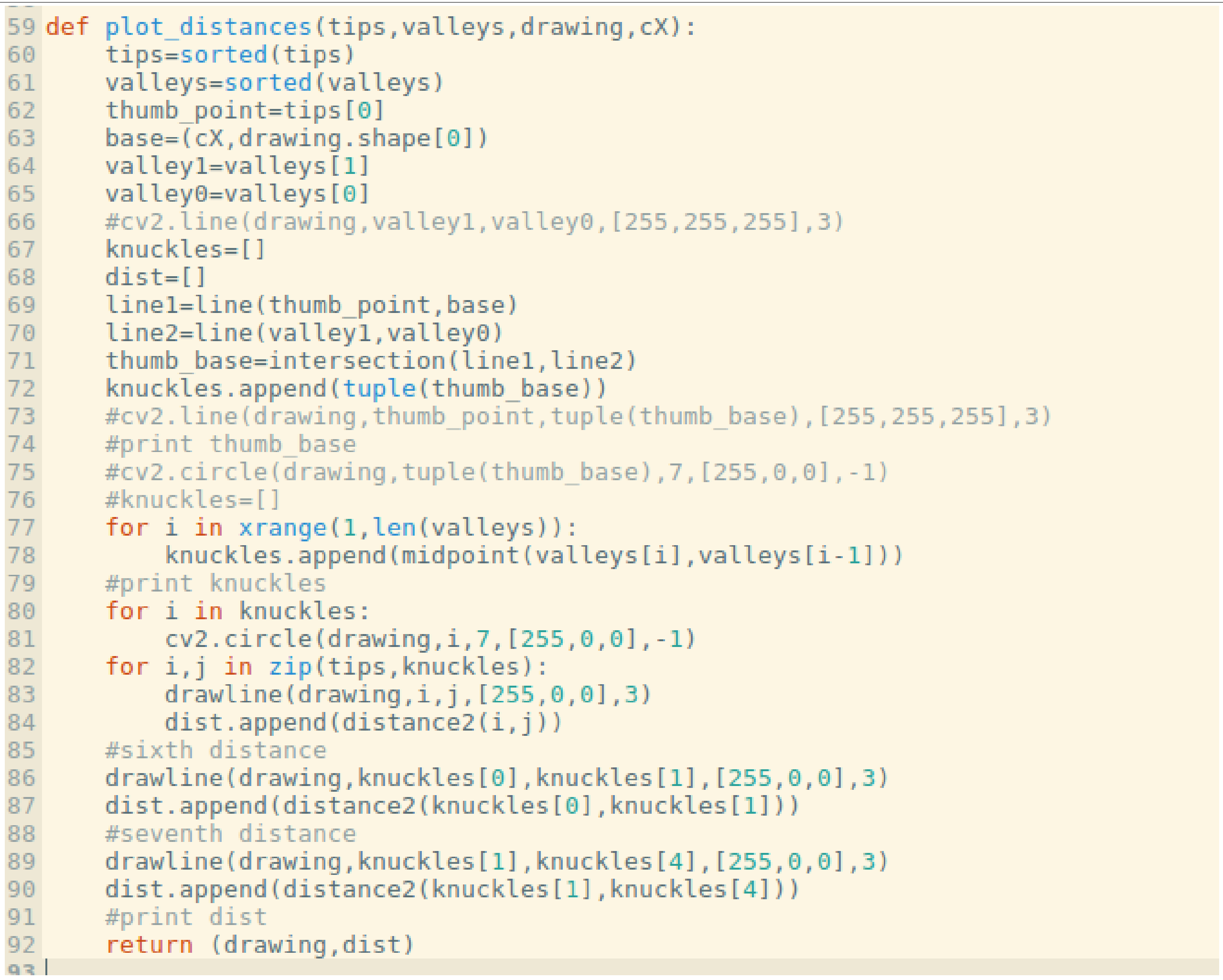
7. Getting Knuckle Points

For knuckle points extraction we take the mid point of five valley points which will give us 4 point with respect to each finger. For the knuckle point of thumb we used the valley points of the thumb and adjacent fingers valley point. We draw a line using valley point of thumb and valley point of te finger which was closer to the thumb. And one more line using the tip point of thumb and drop of centroid point. The intersection point of these two lines will be our knuckle point of thumb.



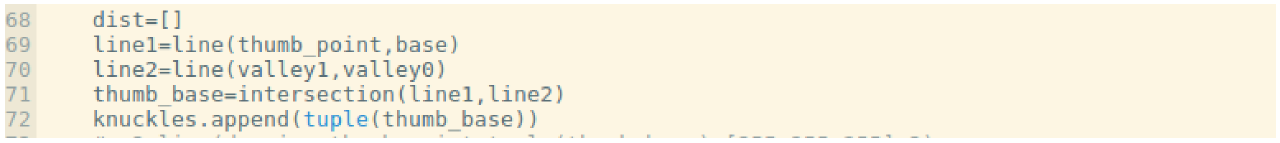
```
def midpoint(a,b):
    return tuple(map(int,((a[0]+b[0])/2,(a[1]+b[1])/2)))
```

Above code is used for finding midpoint of the valley points.



```python
def plot_distances(tips,valleys,drawing,cX):
    tips=sorted(tips)
    valleys=sorted(valleys)
    thumb_point=tips[0]
    base=(cX,drawing.shape[0])
    valley1=valleys[1]
    valley0=valleys[0]
    #cv2.line(drawing,valley1,valley0,[255,255,255],3)
    knuckles=[]
    dist=[]
    line1=line(thumb_point,base)
    line2=line(valley1,valley0)
    thumb_base=intersection(line1,line2)
    knuckles.append(tuple(thumb_base))
    #cv2.line(drawing,thumb_point,tuple(thumb_base),[255,255,255],3)
    #print thumb_base
    #cv2.circle(drawing,tuple(thumb_base),7,[255,0,0],-1)
    #knuckles=[]
    for i in xrange(1,len(valleys)):
        knuckles.append(midpoint(valleys[i],valleys[i-1]))
    #print knuckles
    for i in knuckles:
        cv2.circle(drawing,i,7,[255,0,0],-1)
    for i,j in zip(tips,knuckles):
        drawline(drawing,i,j,[255,0,0],3)
        dist.append(distance2(i,j))
    #sixth distance
    drawline(drawing,knuckles[0],knuckles[1],[255,0,0],3)
    dist.append(distance2(knuckles[0],knuckles[1]))
    #seventh distance
    drawline(drawing,knuckles[1],knuckles[4],[255,0,0],3)
    dist.append(distance2(knuckles[1],knuckles[4]))
    #print dist
    return (drawing,dist)
```

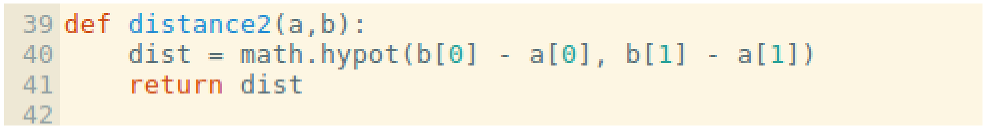
The above written code is used for for finding all knuckle points. The following part of above code is used for finding the knuckle point of thumb.



```python
    dist=[]
    line1=line(thumb_point,base)
    line2=line(valley1,valley0)
    thumb_base=intersection(line1,line2)
    knuckles.append(tuple(thumb_base))
```

8. Getting Distance Between Knuckle Point and Tip Point

After getting kncuckle point with respect to thumb and finger. we calculated the the distance between between tip point and knuckle point with respect to each finger and thumb.



```python
def distance2(a,b):
    dist = math.hypot(b[0] - a[0], b[1] - a[1])
    return dist
```

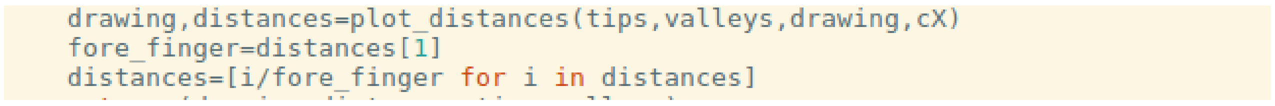
We calculated total seven distance. Out of seven , five distances are between knuckle point and tip points. One distance between knuckle point of thumb and knuckle point of nearest finger to the thumb. One more distance which was calculated between knuckle point of forefinger’s and index finger’s knuckle point.

From here we get total 7 distance which we will use for matching.

9. Normalizing The Distance

As we know that the length of the fingers of a aged man and a child is vary in real life. So avoiding this problem we normalized the distances we get.

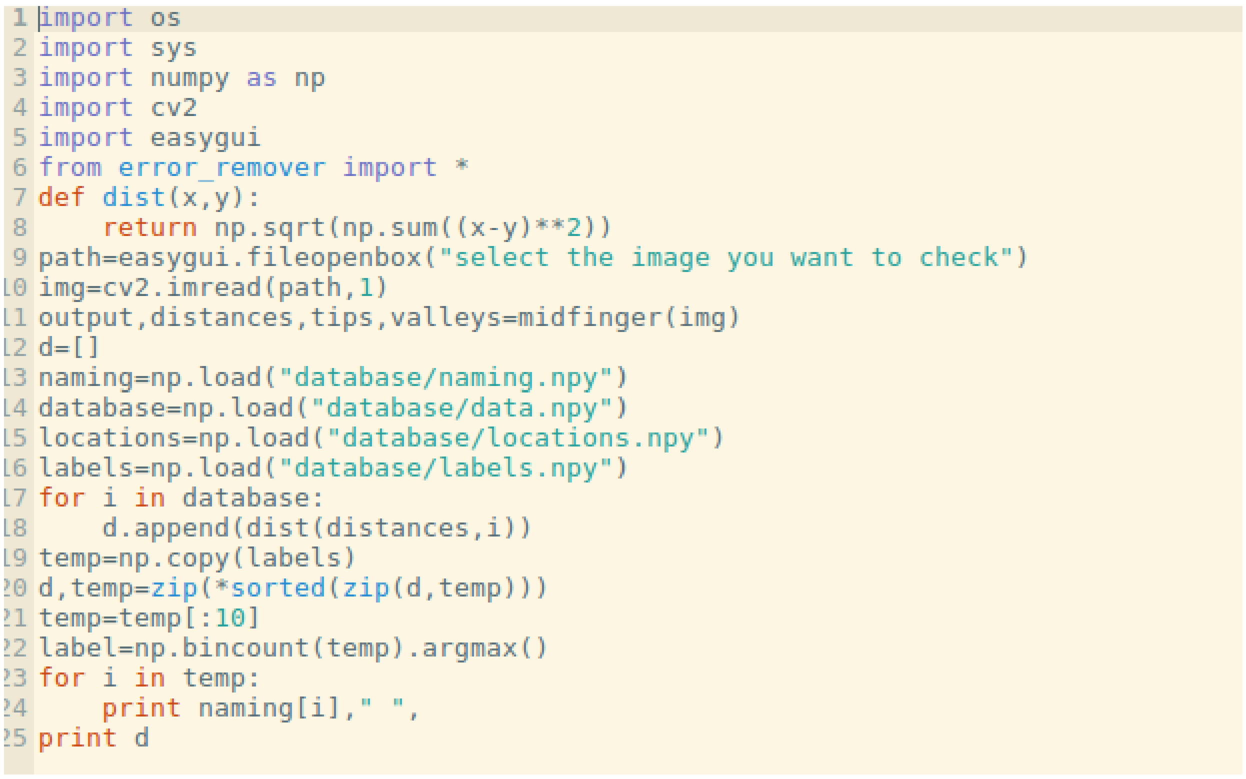
For normalization we didivded all the distance by the distance of forefinger. Now the distance we have is normalized distance.



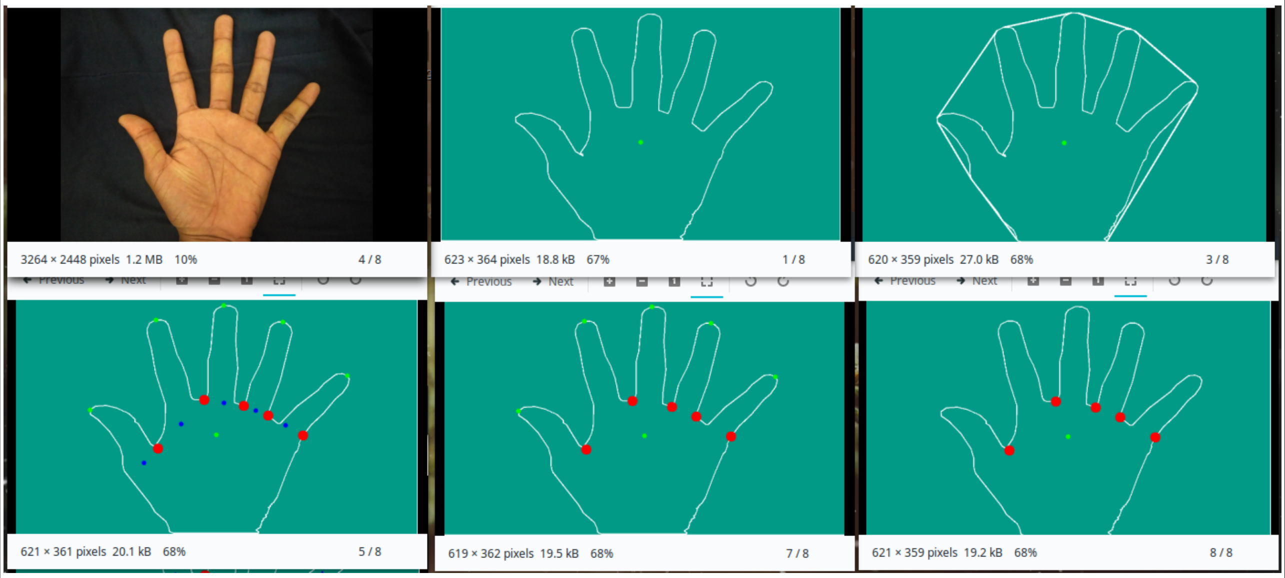
```
    drawing,distances=plot_distances(tips,valleys,drawing,cX)
    fore_finger=distances[1]
    distances=[i/fore_finger for i in distances]
```

10. Matching

for matching we will use the distance we have calculated. We will match the live system distances with the distances of each and every image in our databbase. From which image we will get minimum distance we will finally predict the identity of that person.



```
import os
import sys
import numpy as np
import cv2
import easygui
from error_remover import *
def dist(x,y):
    return np.sqrt(np.sum((x-y)**2))
path=easygui.fileopenbox("select the image you want to check")
img=cv2.imread(path,1)
output,distances,tips,valleys=midfinger(img)
d=[]
naming=np.load("database/naming.npy")
database=np.load("database/data.npy")
locations=np.load("database/locations.npy")
labels=np.load("database/labels.npy")
for i in database:
    d.append(dist(distances,i))
temp=np.copy(labels)
d,temp=zip(*sorted(zip(d,temp)))
temp=temp[:10]
label=np.bincount(temp).argmax()
for i in temp:
    print naming[i]," ",
print d
```


3264 × 2448 pixels 1.2 MB 10% 4 / 8
623 × 364 pixels 18.8 kB 67% 1 / 8
620 × 359 pixels 27.0 kB 68% 3 / 8
621 × 361 pixels 20.1 kB 68% 5 / 8
619 × 362 pixels 19.5 kB 68% 7 / 8
621 × 359 pixels 19.2 kB 68% 8 / 8
Previous
Next

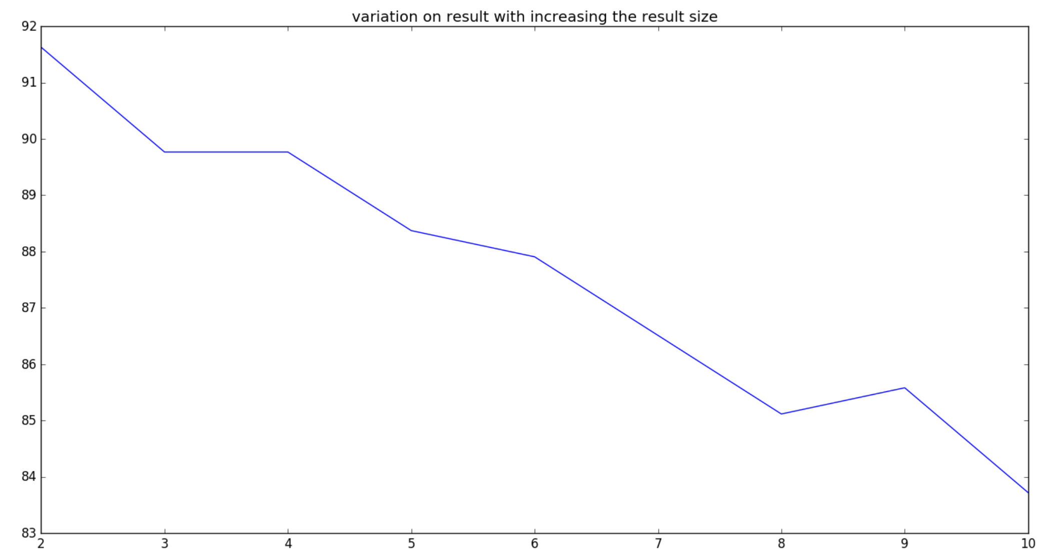
Result Analysis:

After running algorithm over a database of 17 classes with 15 template each we observed that an accuracy of 95% is observed. But database is not isolated.

Running same algorithm over isolated database gives 89% success but in this case matching image is ignored which means best match is ignored because best match is same image from the database.

For above two runs we considered 10 best distances and label with the maximum count was our result.

As we started to reduce the best number of matches, result was being better which clearly indicates that matches of same class are coming first and wrong matches are coming later. Below plot verifies above statement.



Software Rquirements:

1. Linux Mint or any ubuntu based operating system.

2. Python 2.7

3. Opencv2

4. Numpy

5. Easygui

6. Tkinter

7. python-imaging-tk

8. raspbian OS for rspberry pi

Hardware Requirements:

1. One machine for server while another for client (raspberry pi here).

2. raspberry pi display (optional)

3. raspberry pi camera

4. Power and network connectivity.

Proposed Improvements:

1. Use machine learning clustering algorithms instead of simple euclidean distance.

2. Extraction of angular features and making higher order feature by combining them with distance features.

References:

1.**Identity verification using shape and geometry of human hands** by Shefali Sharma, Shiv Ram Dubey, Satish Kumar Singh, Rajiv Saxena and Rajat Kumar Singh

2.**Online Palmprint Identification** by David Zhang, Wai-Kin Kong, Jane You, and Michael Wong.