Texas A&M University

Department of Computer Science and Engineering

Final project

CSCE 624

Sketch Recognition

Author:

Shipeng Yang (130004503)   Aditya Vijayvergia (130004783)

Prof: Tracy Hammond

SIGNATURE ANALYSIS

Shipeng Yang
College Station, U.S.
sy600215@tamu.edu

Aditya Vijayvergia
College Station, U.S.
adityav@tamu.edu

BACKGROUND

This project is the final project of Sketch Recognition course of Texas A&M University. Course number is CSCE 624 and professor is Tracy Hammond.

ABSTRACT

In this project, our main goal to recognize if a signature is piratical or not. We analysis the speed and curvature of the stroke, bounding box and some other features to figure it out.

Keywords

Signature analysis, features, security.

INTRODUCTION

Signature is really important thing in people daily life. If I want to pay a check to my community, I need to sign my name on the check. When I use my credit card in Walmart, I also need to do my signature to verify my identity. It always help to recognize people's identity or show peoples' authority.

However, in most of business systems now, they don't really analysis our signatures to verify identifies. A lot of time, it will just care if there is a signature on a check or bill instead of checking if people really write his or her own signature. If we have a tool to compare out signature and the signature. in the database, it will be a valid way to protect customers or user. In addition, when we are trying to login our account such as Email or school account in a new computer, we are always be asked to verify our identity by sending message our answer some questions. Most of time we have no trouble with this, but if we can analysis people's signature, we will have one more valid way, which may be more convenient and it may spend less time. So we decide to make a signature analysis tool to do this.

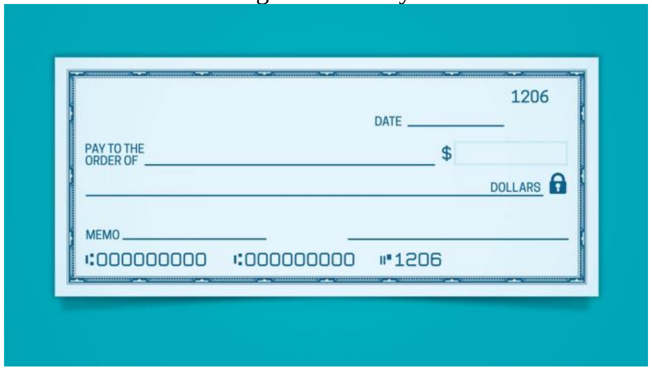


Figure1 : Personal check

We will let user to write his or her signature on a electric screen such as smart phone and touch screen laptop, or use mouth to write signature, so we can collect the data of their signatures and analyze it.

Prior work

We did some search online and read some paper from our professors, the most useful materials for us are Rubine's paper and Long's paper, which are from our professor. We use some of features from them in our project and I will figure them out later.

Implement

I. **Code, Tool and Software**:

We use JavaScript, Visual Studio Code and Git-hub to do this project.

Figure2: The tools we use in our program: JavaScript, Visual Studio Code and Git-hub.

II. Features

Basically stoke is a collection of points, and each point contain 3 information, which are x,y and time. Analyzing the stroke is trying to figure out how x,y and time are vary with each other. These three variables can give us the information we need.

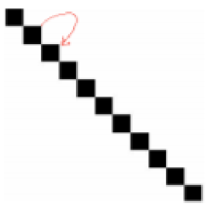


Figure 3: A stroke is a collection of points.

(1) Length of Diagonal

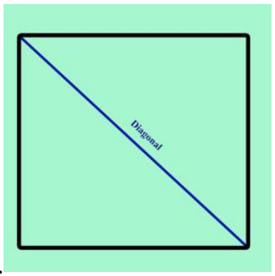
Make the whole stroke into a bounding box, and then calculate the length of Diagonal of bounding box.

Length of Diagonal is the best feature to show the size of this stroke, since height and width sometime will have extreme situation, diagonal can reduce this error properly. This feature is useful because user may user different size of screen to do signature.

(2) Angle of Diagonal

Calculate the angle of diagonal of bounding box.

This feature is good to show the total shape of stroke, which is the ratio between height and length. Similar signatures should have similar rate



.

Figure 4: Diagonal of bounding box

(3) Max Speed

Calculate max speed of the stroke.

Speed is useful because if someone try to write others' signature, he or she may not be that familiar with this handwriting, so speed might be slower. Same with average speed.

(4) Max relative speed

Max speed/ length of diagonal

Since people may do signature in different device, we need to avoid the error from size of device screen, my strategy is to make all features which are related to size of screen to relative features to avoid the error. The method is divide this feature by length of diagonal.

(5) Average speed

Average speed of the stroke.

(6) Average relative speed

Average speed/length of diagonal

(7) Total time

Total time of this stroke

Time can show if the user is familiar with this stroke and it can also express the length of stroke.

(8) Mid part relative speed

Only calculate mid part(from 33% to 67%) to avoid pen up and pen down error. Some people may spend some time to start writing and they may also stop a second after finishing the stroke. This feature can reduce this effect.

(9) High speed points match

Calculate the high speed points(over two times of average speed) in the stroke.

This is the most important feature in our project, which can show the high speed parts of the whole stroke. For the same signature, the high speed part should always be very similar. For example, in this signature, it is obviously the straight line is much faster then corners. In 'A' and 'l', they are really like straight line, it should have higher speed than other part.

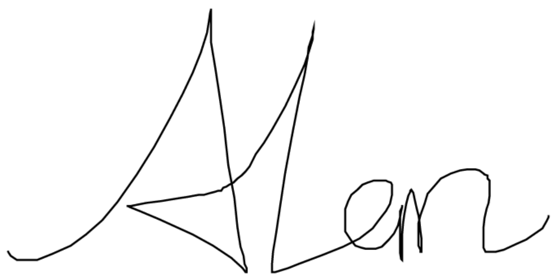
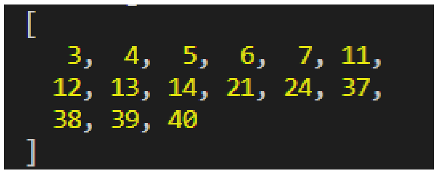


Figure 5: Template signature: Alan

However, if we let same people write signature two times, the number of points can be similar, but there are also some errors. For example, first time someone write a signature in a smart phone, but second time he write a signature on a laptop. It may contain different numbers of points. To reduce this error, I make all points shrink into 100 points, which show that in nth percentage, it reach the high speed points.

I tried 10 times signature can express the majority of the high speed point, the output is like this:

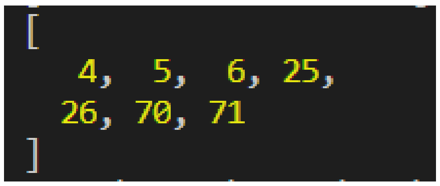


```
[
  3,  4,  5,  6,  7, 11,
  12, 13, 14, 21, 24, 37,
  38, 39, 40
]
```

Figure 6: High speed points from my signature

It fit my guess, there are three parts of high speed points, which are two straight stroke of 'A' and 'l'. In addition, we can also find the low speed points for each signature if necessary. Similar signature should also have same range low speed points.

However, I let my roommate to write my signature several times and also find the high speed points, which is like this:



```
[
  4,  5,  6, 25,
 26, 70, 71
]
```

Figure 7: High speed points from other signature

It is kind of similar with my signature, but since everyone has different preference, my roommate does not really familiar with my signature, so he may lose some high speed points or he may have some extra high speed points which are should not be

We have some special features in our program, which need to take several steps:

1.**Pre-processing :**

Removing noise

The data set contains some strokes that consist only of a single point. A single point is just a dot on sketch and is incapable of representing lines or corners. Hence such strokes are removed before the algorithm checks for presence of corners in a stroke.

Resampling

detecting extra corners. While resampling, same changes are made in the strokes like refining wavy lines to straight lines and redrawing curves with proper curvature. To record a stroke, the position of pen is recorded repeatedly after a fixed duration of time. Since different parts of sketch are drawn at different speeds, these points are not equidistant. The algorithm preprocess the strokes so the all the points on the stroke are at the same distance. This is done as-

- If the distance of next point p2 from the current point p1 is very small, Ignore the next point p2 and move to the point next to p2
- If the distance is more than the threshold t -
  1. Find the vector v in the direction of current point to the next point as

     v = position of next point - position of current point

  2. divide v by its magnitude to make it a unit vector
  3. make a new point p at a distance t from the current point in direction v
  4. make p as the current point and p2 stays as the next point
  5.

2. Algorithm

As discussed above, the algorithm simulates driving a car in high speed on a road in the shape of the stroke and detect collisions as corners. The major steps of the algorithm is as under.

Step 1: Defining the car

The car is defines by 2 parameters -

1. speed
2. maximum steering angle

Step 2 :Getting the direction of the road

The algorithm defines the direction of the road using the direction of a vector pointing from a point n points behind the current position of car to the current position of the car. This is also the distance the car will move in 1 unit time.

Step 3: Predicting the position of the car after 1 unit time

Assuming straight road The direction found in step 2 is used to predicts the position of a point n points ahead of the current position assuming that the car is driving on a straight road.

Step 4: Detecting curvature of road

After the prediction, the angle between the predicted point and actual point is calculated. This is the angle that the car needs to steer in order to avoid collision.

Step 5: Checking for corner

If the above angle is more than the maximum steering angle of the car, the current point is marked as a corner point.

Understanding the car parameters

This value n used in step 2 is the speed of the car as if we take a larger value of n (i.e., the car is moving at a higher speed) then the algorithm will be predicting a point that is even farther away from the current point. Hence the chances of error will be larger. In are simulation, it can be explained as a car moving in a higher speed, higher the speed, harder it will be to change the car's direction. Hence, the speed of the car n and the maximum steering angle are the two parameters that define the sensitivity of the algorithm towards corners.

Extracting Features from Segments:

After breaking the signature into segments, some features are saved from each of these segments. We then use these features to compare two signatures.

The detected features are as under:

1. number of segments
2. average speed of each segment
3. ratio of sum of angles in the segment and sum of absolute angles in the segment

In above features, feature 2 and 3 are arrays that store information for each segment. Hence, these are only 2 features but the amount of information they extract from the stroke is much higher.

We also use ratio of sum of angles and sum of absolute angles in the whole signature as it can combine with feature 3 above to give proper insight of how the signature is drawn.

III. Calculation

Firstly, we make all features in priority order, from high to low. Some of them are ancillary features and we removed it. After that, since each features have different importance, we give them different weight. The most important feature get the highest weight and the lowest important feature get the lowest weight. Then we calculate the difference score, which can be from negative score to thousands.

When a signature is from different people, it will give us a very high score, however, the different score will be low.

IV. Test

Test one: Same person to draw same signature with same speed

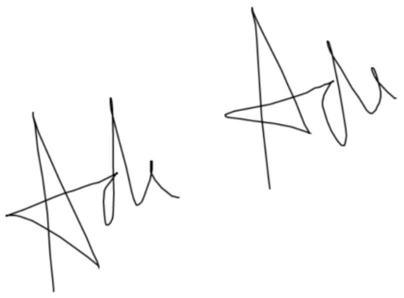


Figure 8: Signature from same people

Different point : 0

Which means there is no difference between two signature. We are 100 percent sure these two signature are from same people.

Test two: Same person to draw same signature but in different speed

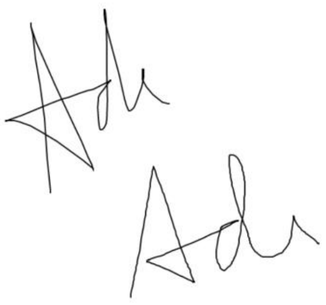


Figure 9: Signature from same people with different speed

Different score: 3359.28

It has a pretty high score which mean it looks like these two signature are not from same people. We do this situation since maybe someone know another person's signature, but he do not know how that person write it. Even absolute same signatures may have different speed in each part. Our system will be able to recognize it.

Test Three: Same person to draw same signature and same speed.

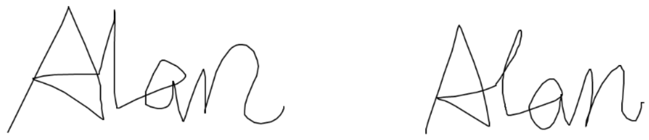


Figure 10: Signature from same people with tiny differences

Different score: 125

This time, same person write same signature twice, but there is a little difference between them, so it has a different score, but relative low. Which means these two signatures have high probability from same person.

Test Four: Different people to draw same signature.

Different score: 2036

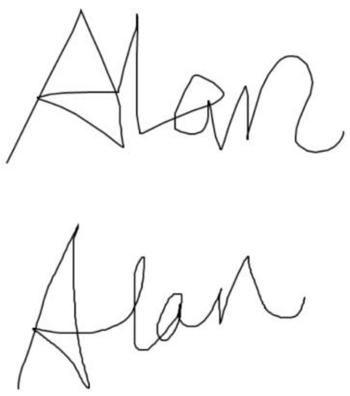


Figure 11: Signature from different people

It has high different score since the strokes are different are speed might also be different.

Test Five: Same person draw two different signatures

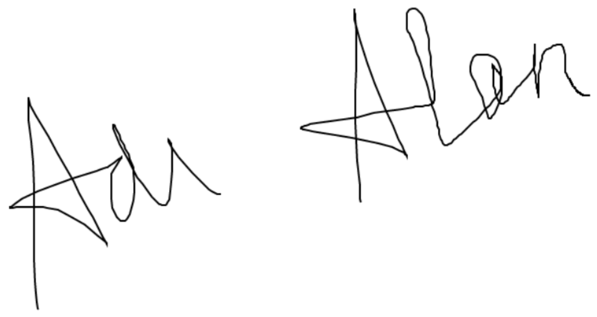


Figure 12: Different signature from same people

Different score: 4397

In this situation, it has the highest different score. Which means two signatures must be different.

V. Conclusion

Our main goal is to recognize if a signature is from the owner to protect customer’s money security. Basing on the test, we basically achieve this goal. It can recognize the signature.

We did a lot of tests and our program works well. Not only it can recognize the difference between two signatures, but also can realize if there are from different person. If this can be a tool which is used in password security system or bank system, it might be useful to protect people’s security.

VI. Future work

Since the time is limited, we don’t really add and test all features which are useful. We just use some features we think are useful. In the future, we need to do a lot of test to figure out what others features are useful and which current features is useless to provide a better algorithm.

In addition, we don’t have time to have a UI, which is necessary to let user use this tool. If we have time, after improving the functionality, I will build a UI for our project.

REFERENCES

1. Gabe Alvarez, Blue Sheffer and Morgan Bryant. Offline Signature Verification with Convolutional Neural Networks. Retrieved Dec 05, 2019 from

http://cs231n.stanford.edu/reports/2016/pdfs/276_Report.pdf

2. Krzysztof Cpalka, Marcin Zalasinski and Leszek Rutkowski. Feb 17, 2016. A new algorithm for identity verification based on the analysis of a handwritten dynamic signature. Retrieved Dec 06, 2019 from

https://www.sciencedirect.com/science/article/abs/pii/S1568494616300680

3. Sandhya Katiyar, Shubham Agarwal, Shubham Kaushal and Himanshu Vats. May, 2016. Signature Recognition and Verification System via Neural Network. Retrieved Nov 30,2019 from

https://pdfs.semanticscholar.org/c4fd/ab690ec68a06c590d866e7b1e3832b8f3cff.pdf

4. A.Chris Long, Jr., James A. Landay, Lawrence A.Rowe, and Joseph Michiels. Retrieved Dec 06 from

https://www2.eecs.berkeley.edu/Pubs/TechRpts/1999/CSD-99-1069.pdf

5. Dean Rubine. Specifying Gestures by Examples. 1991. Retrieved Dec 10 from

http://reports-archive.adm.cs.cmu.edu/anon/itc/CMU-ITC-099.pdf

6. Dean Harris Rubine. The Automatic Recognition of Gestures. Dec, 1991. Retrieved Dec 10 from

http://reports-archive.adm.cs.cmu.edu/anon/itc/CMU-ITC-099.pdf

7. Tevfik Metin Sezgin, Thomas Stahovich and Randall Davis. Sketch Based Interfaces: Early Processing for Sketch Understanding. May, 2001. Retrieved Dec 10 from

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.331.6262&rep=rep1&type=pdf

8. L. Eggli. Sketching with constraints. Master’s thesis, University of Utah, 1994.

9. J. A. Landay and B. A. Myers. Sketching interfaces: Toward more human interface design. IEEE Computer, vol. 34, no. 3, March 2001, pp. 56-64.

10. Christopher Laughman, Kwangduk Lee, Robert Cox, Steven Shaw, Steven Leeb, Les Norford, and Peter Armstrong. April 2003. Retrieved from

http://web.mit.edu/parmstr/www/pubs/PSAlaughman.pdf

11. Sabourin, R.: Off-line signature verification: Recent advances and perspectives. [30] 84–98

12. Leclerc, F., Plamondon, R.: Automatic signature verification and writer verification: The state of the art 1889–1993. Int. J. Pattern Recognit. Artif. Intell. 8, 643–660 (1993)

13. Impedovo, S., Simon, J., eds.: 1From Pixels to Features III: Frontiers in Handwriting Recognition. North Holland, Amsterdam (1992)

14. Yoshimura, I., Yoshimura, M.: Off-line verification of japanese signatures after elimination of background patterns. Int. J. Pattern Recognit. Artif. Intell. 8, 53–68 (1994)

15. Yoshimura, I., Yoshimura, M.: An application of the sequential dynamic programming matching method to off-line signature verification. [30] 299–310

16. Justino, E., Bortolozzi, F., Sabourin, R.: A comparison of SVM and HMM classifiers in the off-line signature verification. Pattern Recognit. Lett. 26, 1377–1385 (2005)

17. Justino, E., Bortolozzi, F., Sabourin, R.: Off-line signature verification using HMM for random, simple and skilled forgeries. 1031–1034 (2001)

18. Kalera, M., Srihari, S., Xu, A.: Off-line signature verification and idenitification using distance statistics. Int. J. Pattern Recognit. Artif. Intell. 18, 1339–1360 (2004)

19. Hairong, L., Wang, W., Wang, C., Zhuo, Q.: Off-line signature verification and forgery detection using fuzzy modeling. Pattern Recognit. 38, 341–356 (2005)

20. Hanmandlu, M., Yusof, M., Madasu, V.: Off-line chinese signature verification based on support vector machines. Pattern Recognit. Lett. 26, 2390–2399 (2005)