A Novel Multi-touch Authentication Scheme for Mobile Devices

Dicheng Wu
Department of Mathematics and Computer Science
South Dakota School of Mines and Technology
Rapid City, SD 57701
Dicheng.Wu@mines.sdsmt.edu

Mengyu Qiao
Department of Mathematics and Computer Science
South Dakota School of Mines and Technology
Rapid City, SD 57701
Mengyu.Qiao@sdsmt.edu

Abstract

The enhanced performance and reduced cost have made mobile devices deeply penetrate into daily life and reform people’s habits in modern society. While people enjoy the convenient services and diversified contents provided through mobile devices, the prosperity of mobile device also leads to serious security concerns in mobile devices. User authentication plays an indispensable role in protecting computer systems and applications from unauthorized access. Many user authentication methods have been proposed and implemented to protect desktop computer system, but do not provide optimal security and convenience for the new generation of touchscreen-equipped devices. Therefore, there is especially high demand for a new user authentication method, which achieves high accuracy, usability, compatibility, and low cost for mobile devices. In this paper, we present a novel touchscreen-based authentication scheme that utilizing both static and dynamic features generated by different hand's gestures. We collect raw data including position, size, pressure, time of each individual touch-point generated by fingertip movements which correspond to distinct characters of gestures of different users. Then, we convert raw data to static and dynamic features to achieve accurate pattern recognition. Several volunteers are invited to help experiment the proposed scheme and collect sample data by performing different gestures for multiple times on different touch-screen devices. Afterwards, we run statistical analysis to identify discriminative features to reduce the complexity and enhance accuracy for classification. In the end, we apply and compare various machine learning approaches with selected features to build stable and robust classification models. As a proof-of-concept, a mobile app is developed to implement the proposed scheme for android tablet due to its API and hardware supports. When a user uses this app at first time, the app will ask the user to

sign up an account. Then, it leads the user to a sign-up screen and asks the user to enter a unique username and an email address. In the next step, the user is directed to another screen where he/she can select a preferred picture as the gesture background. Then, the app asks user to perform a gesture for three times to obtain initial gesture pattern data. In meantime, it also tests the similarities among the gestures. If an unstable pattern is detected, it will ask user to redo the gesture until the similarity meets pre-defined requirements. After successful registration, the user can sign in with the username and secret gesture. Each gesture will be evaluated by the classification model associated with the user account. Empirical research and experiments show that the proposed scheme overcome the drawbacks of the existing methods, and achieve high accuracy and usability for user authentication. Therefore, we believe it has great potentials to provide secure protection for systems, applications, and data in touch-screen equipped mobile devices.

1 Introduction

The enhanced performance and reduced cost have made mobile devices deeply penetrate into daily life and reform people’s habits in modern society. The attention and popularity gained by mobile devices are dramatically increased in recent years, and lots of existing IT companies tend to extend their products to mobile side in addition those tons of newly established companies solely focus on developing software on the mobile side. In consequence, a certain amount of people gradually switches their primary platform from PC to mobile devices and lots of sensitive data generated by those users is moved and stored in the mobile devices. However, people enjoy the convenient services and diversified contents provided through mobile devices, the prosperity of mobile device also leads to serious security concerns in mobile devices. It motivates those manufacturers of mobile devices to bring verities of solutions in order to resolve various security issues and protect users from malicious attackers. For example, Apple provides the product with a Touch ID which is a fingerprint recognition feature thus only the owner of the device can access it. The drawbacks of this scheme are it requires additional hardware to support this kind of feature and also it is not impossible to copy the fingerprint from the owner. To mitigate this kind of disadvantage, we developed an application based on android that does not require additional hardware and utilizing static and dynamic features of users therefore make the replication of user’s identity more difficult.

2 A User Authentication Application on Android

The application is mainly developed and tested on Google Nexus 7 tablet PC, and it provides a touchscreen-based scheme to authenticate users. To build a robust classification model, it is necessary to collect raw data which contains x-y points, pressure, size and time from users’ gesture drawing and we directly get those data by using motion event function in Android touchscree API and then store them into a mobile database. Then the idea is to utilize the static and dynamic features that extract from raw data to and lately feed those features to classifiers to verify whether the current user is legal when they are performing different gesture. If the user’s identity is verified, the classifier will classify the user into self-class which is denoted by 1 otherwise it will classify the user into non-self-class which is denoted 0. More specific, those features are distance, angle and pressure. Each finger has its own feature set that comprises5 features, and thus there are totally 25 features for each hand gesture. All those features are calculated in real-time and are based on raw data extracted from user gesture, after the calculation, the data is stored in database for classification usage. The flowchart of program is showed below.

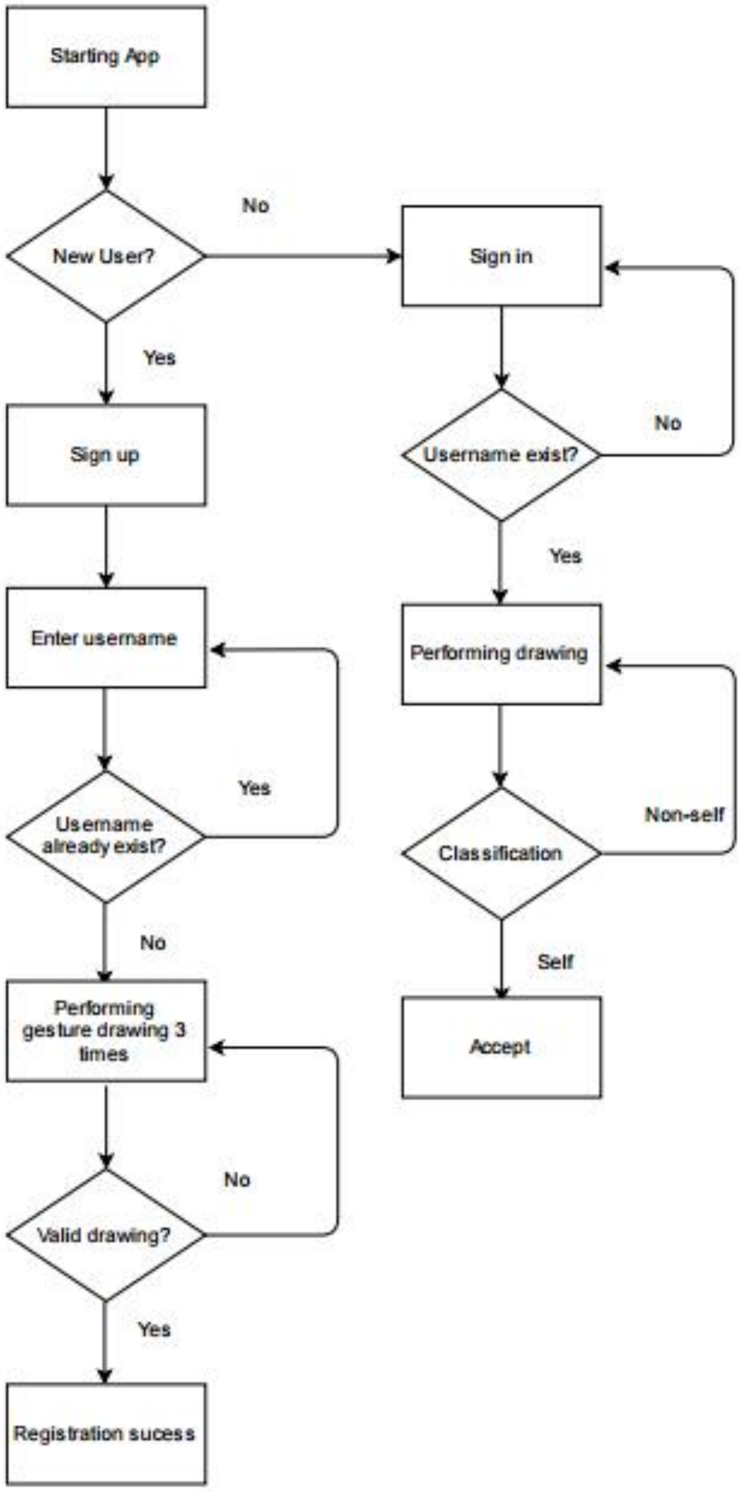


Figure 1: The Flowchart of the Application

In order to use the app, a user should have an account. If they don't, they will be asked to enter a unique username that has not been used before, because one device can be registered by several users and there may already have username registered by someone else. Afterwards, they are going to be asked to perform gesture drawing multiple times while calculating the features and later on the application will evaluate the similarity among those drawing based on features by applying Euclidean distance. The application would reject the drawings if the similarity is below the threshold, and then ask user to redraw. If the registration is successful, the features will be stored in database for later verification usage. As long as user has an account for this application they can sign into the system by drawing the same gesture and the features extracted from user will be feed to classifier in order to determine whether current user is genuine or not. The user will be accepted if they are classified as self otherwise they get rejected by the system.

3 Features Details

The features currently being used are distance, angle and pressure, and additional features will be added later. .

For the distance, we measure the length of tracks that user performs, and it is calculated by using distances between each pair of adjacent points and the summation of each individual distance to the total distance. The formula can be expressed as $distance_i = \sum_{j=0}^{n} \sqrt{(x_{i,j+1} - x_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2}$ (i=1,2,3,4,5) . The symbol i is the number which assigned to each finger from 1 to 5 and symbol j is the nth-of-points which is from 0 to number of total points - 1. $x_{i,j}$ is jth points on number of ith finger.

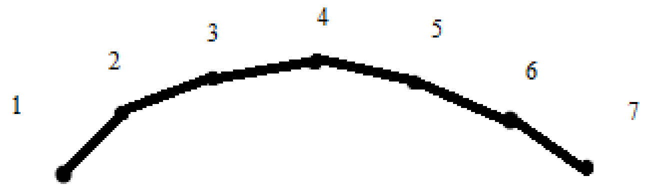


Figure 2: The One of the Tracks Generated by Gesture Drawing

For the angle value, it is calculated by measuring the angle of vectors that connecting points at certain interval, and then compute the accumulation of angle changes for the entire track.. The main reason of not using angle value between each two point is because it has some kind of perturbations which affect the result became too large and uncertain. Instead, we choose adjacent points and add up angles between those points. In this application, the point is calculated after the distance of track is calculated and it take points at certain percentage of the total distance. Here, we use 7.5%, 15%, 22.5%, 30% up to 100% as the cutoff points. The angle value is calculated by the formula of $angle_{i,j} = arctan\frac{y_{i,j+2} - y_{i,j}}{x_{i,j+2} - x_{i,j}}$ (i=1,2,3,4,5) , and it takes every other cutoff point to do the calculation by using arc tangent. The primary reason to do in this way is the density of points on the track differs on each location which may result in the percentage length largely differs from each other and it is inaccurate to just use adjacent pairs to calculate angle value. Therefore, we decide to use every other cutoff point as the angle value calculation reference.

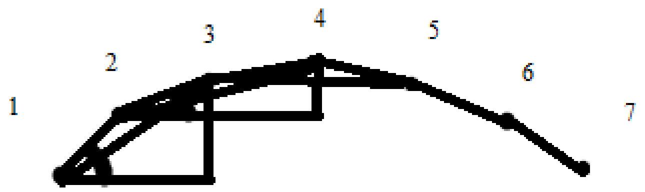


Figure 3: The Scheme of Angle Calculation.

After the each individual angle value has calculated, it needs to sum all those angle difference values and to get the accumulate angle value. The formula can be written as $acc\_angle_i = \sum_{j=1}^{n} |angle_{i,j} - angle_{i,j-1}|$(i=1,2,3,4,5;n=10) . It calculates the difference between successive angles in order to get the angle changing rate and then add them up.

For the pressure, it is easy to distinguish among different users since they tend to have different fingertip pressure, and it directly use the raw data extracted from user and make the averaging pressure. The formula can be written as $AvgPressure_i = \frac{1}{n} * \sum_{j=0}^{n} pressure_{i,j}$(i=1,2,3,4,5) . The n is total point and it adds up all n pressures and is divided by n.

4 Similarities Measuring

The similarities measuring is crucial for the application since the system has judge whether the gesture drawing meets the requirement and keeps the data consistent in order to get better perform in the training process. Here we adopt the idea of using Euclidean distance for the similarity checking. The formula can be written as $Similarity = \sqrt{\sum_{i=1}^{5} (feature_{1,i,j} - feature_{2,i,j})^2}$ (j=1,2,3) , where i denotes finger index and j denotes feature index. The application will ask the user to redraw a gesture if the similaritie is below the pre-defined threshold.

5 Finger Numbering

Another thing needs to be solved is finger numbering, since the motion event function labels finger by order of timestamp that finger touches on the screen. In other word, the fingers will be assigned label from 1 to 5 solely dependent upon the order that fingers touch on the screen. This would make the inconsistency of same finger labeled different number at different times, because the order of finger that touched on the screen will be likely random. In order to resolve the inconsistency, it is necessary to introduce a scheme that makes a stable labeling at each time. However, it is hard to directly manipulate on motion event function and make it acting the way we expect. Therefore, we decided to relabel the pre-labeled fingers in the way that assigns the labels by the real finger order. We discover that at each time when the gesture drawing is performing the thumb has largest distance towards the rest of fingers.After the thumb is determined, it is relative easy to figure out the rest of fingers.

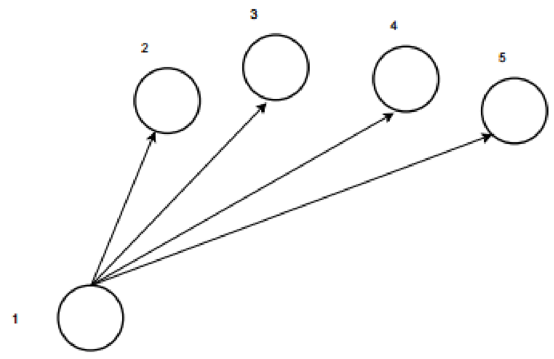


Figure 4: The Way to Calculate Distance in Finger Numbering Scheme

In the figure above, the circles illustrates the first location that each finger touchs the screen, and the thumb is labeled as 1 and the rest of fingers are successively labeled from 2 to 5 clockwisely. We calculate the summation of distance of each finger to the rest of fingers and choose the finger which has biggest value of summation of distance as the thumb. The formula can be wrritten as
$\max(\sum_{i=1}^{5} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2})$ $(\text{j=1,2,3,4,5 and j != i})$, wherei denotes current point and j denotes rest of the point.

6 Training Process

This is still in processing and we expect to complete it before the date of conference. We firstly gather data from different volunteers and ask them draw certain gesture on the screen then store feature data to the database. Afterwards, we extract those feature data from database as negative sample and our own drawing as positive sample and split them into training set and validation set. Since we only have limited number of sample and in order to determine the best training set, we choose to do the n-fold cross validation and pick the set that has highest as our initial training sample. For the classifier we choose to use SVM as our primary classier and the reason is it is easy to manipulate and less-prone to overfitting compare to neural networks since it is a convex optimization process while the neural nets suffers from local-optima and also it is more computational cheaper when the number of features grow. We also consider using the data augmentation on the sample data by applying Gaussian noise on the raw data in order to get more number of set of features. All training work is done offline and we plan to move this process to the mobile-side which will perform real-time training process when user registers an account.

7 Implementing Details

The program consists of two phases, registration phase and authentication phase. If the user doesn't have an account, he or she needs to sign up first. We use activity as container for the main page and design GUI on xml file. The main page provides the ability to user to sign into the system by proving a valid account and sign up a new account by clicking sign up button. It will check the database whether the entered username is existed, if so, it allows user to advance to the gesture drawing page otherwise it pops up the username does not exist message.

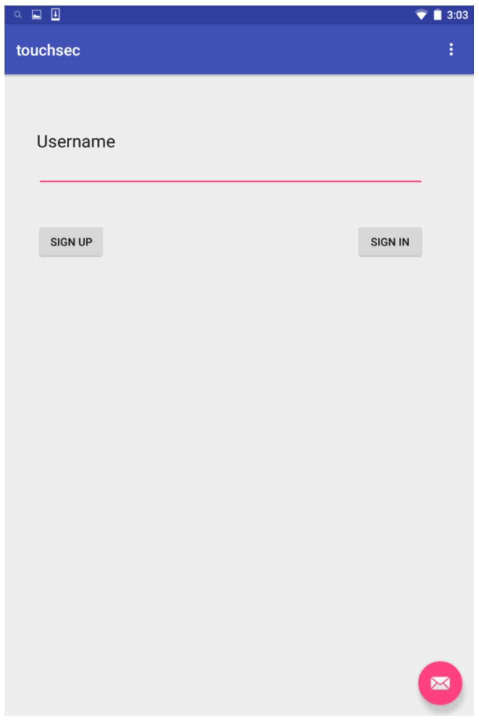
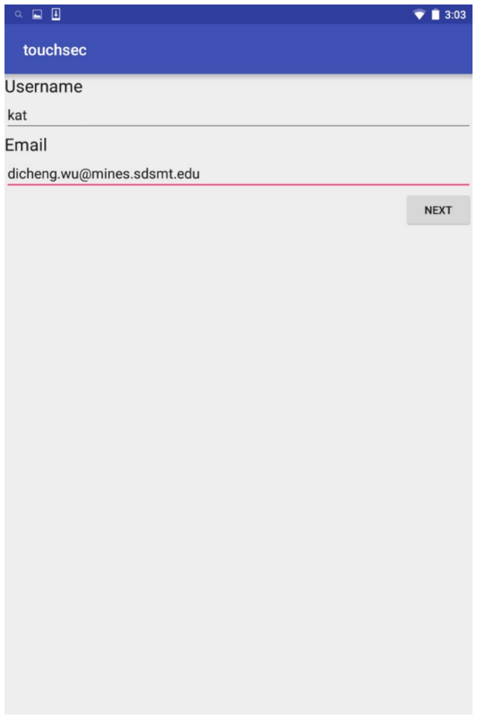


Figure 5: The Main Window

Once the user clicks the sign up button on main window, the application brings the users to the registration page which made up of textedits, textviews and a button, which allows them to enter the username and email address they want to use. It will pop up username already exist if user tries to use a name that already in the database. The user can click next button for next step when they are finishing fill out the information.



Username
kat
Email
dicheng.wu@mines.sdsmt.edu

Figure 6: The Registration Window

The user will be leaded to the image selection page which made up of thumbnails of available background image, which allows user to customize the background when they performing drawing. The purpose of this idea is to provide the user a reference which will help them to remember what position and shape they have been drawed last time and make the drawing more consistent and thus reducing the number of redraws.

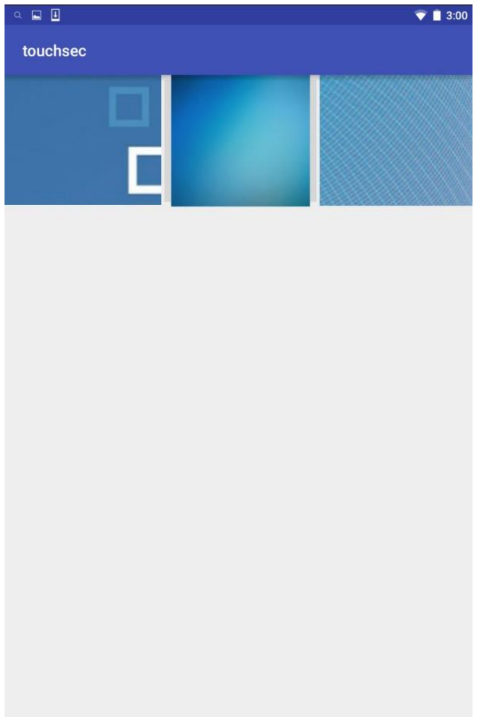


Figure 7: The Image Selection Window

The selected image will show up when they perform gesture drawing. The app also displays lines with different width along the touch tracks, and the width is dertermined by the pressure the user puts on the screen. The width is computed as $width = pressure + 10 * fingersize$. The reason for assigning a small factor to pressure is manly due to the fact that the device sometimes cannot record pressure very accurately.Intially, we implement it by using view and use ondraw to update the draws without using any preprocessing. We create a class that stores x-y coordinate, size, pressure and time, and vectors contain thoes objects. Each vector appends object comprising points information when the user performs drawing at each motionevent trigering. In the meantime, the ondraw function updates the graph according to the newly appended objects. As the result, the performance is not satisfactory especially the area of graph increasing because everytime the ondraw performs it needs to invalidate whole graph and it results the performance reducation with the accumulation of graph elements. The first attempt we make is to use partial invalidate with each induvidual finger instead of updating whole screen. To achieve this goal, we need to pass a rectangle with certain area as to invalidate function indicating the dirty region that needs to be updated. Since it has total 5 fingers, it is necessary to keep 5 distinct dirty regions. We also find that when calculating those dirty regions, the graph may consist of segements of lines with different width, therefore the area of dirty region is dependent upon width of segements of lines. To slove this problem, we develop a inflation function which will inflate the area of dirty region according to the width**.** In the end, althouth it is faster than previous version, the performance is still under our expection. The major reason is that the view has mutiple layers and every time the application invalidate ondraw the request needs to pass mutiple layers in oder to get graph updated. Thus we try to use surfaceview that is similar to a lightweight view and with less layers which means it has better performance campared with view. In order to use surfaceview, it needs to be ran in a sperate thread and with proper starting and ending functions. In addition, it is convenient to keep 2 canvas, one is for surfaceview and another is for main thread. In the surfaceview, we use a canvas to draw another canvas, which is updated on the main thread, onto the bitmap that contains the whole graph, the advantage to use bitmap is it is more efficient because it only need draw the current updated elements and all previous drawed elements are already store in the bitmap. In the main thread, it updates its canvas at everytime the motion event function is triggered. When the motion event is trigered, the surfaceview will start update its canvas by using canvas in main thread. We also set a frame limition in run function to prevent the thread blocking and keep a update flag for update canvas only once when the canvas in main thread is updated in order to prevent the graph from defroming. After all those modifications, the application performance is dramaticly improved. Finally, the user can perform gesture drawing.

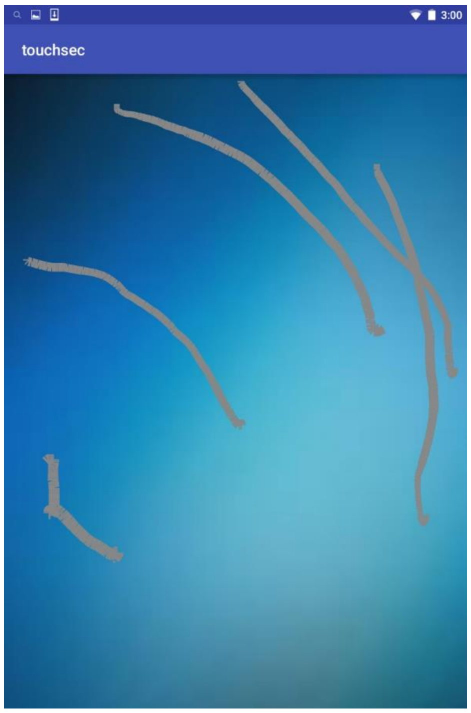


Figure 8: The Gesture Drawing Window

8 Conclusions

In this research, we studied the multi-touch user authentication scheme for touchscreen equipped mobile devices, and developed an android application as a proof of concept. The primary work focused on investigating touchscreen hardware and API performance, and enhancing the efficiency and accuracy of data acquisition and processing. The preliminary results show that the proposed scheme achieve high accuracy and efficiency for user authentication. The future work includes the application of difference machine learning methods for pattern classification and the deployment of classifiers to mobile platform in order to achieve real-time training and classification on mobile devices.

References

Qiao Mengyu, Suiyuan Zhang, Andrew H. Sung, and Qingzhong Liu. "A Novel Touchscreen-Based Authentication Scheme Using Static and Dynamic Hand Biometrics." 2015 IEEE 39th Annual Computer Software and Applications Conference (2015).