VeriLeap

Security System with 3D Gesture Recognition

Zhao Pengran
School of Computing, NUS
A0105541U
a0105541@u.nus.edu

Cheng YingJie
School of Computing, NUS
A0105709H
a0105709H@u.nus.edu

Yip Jiajie
School of Computing, NUS
A0101924R
a0101924@u.nus.edu

Chang Yan Qian
School of Computing, NUS
A0098998R
a0098998@u.nus.edu

Glen Ko
School of Computing, NUS
A0096766J
a0096766@u.nus.edu

Norman Hugh Anderson
School of Computing, NUS
Academic Supervisor

ABSTRACT

Motion detection peripherals are becoming more affordable and commonplace in many homes today. This paper explores the feasibility and authentication opportunities of a static gesture recognition system, implemented on top of one such motion detection system, *Leap Motion*. If deemed successful, security systems such as these can be integrated with existing applications to provide an additional layer of security, or perhaps even serve as a standalone, quick and convenient way of authenticating users.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Authentication and Verification. *Based on the ACM Computing Classification System*: *http://www.acm.org/class/1998/*

General Terms

Algorithms, Security, Verification.

Keywords

Leap Motion, Security, Gesture, Recognition, Authentication

1. INTRODUCTION

It is observed that there is a general trend toward more affordable and much more accurate forms of motion detection intended for use in the average person's home. Microsoft's *Kinect* and Sony's *PlayStation Camera* are some examples of peripherals that are marketed at inexpensive prices to consumers. We believe that devices such as these have the potential to serve as basic security systems for authentication and verification of identity, leading us to conduct a deeper study into the feasibility of such a proposition.

We chose the *Leap Motion* device as a starting point for our proof-of-concept, seeing as to how it has quick and hassle-free integration with most personal computers. Based on this cross-platform design methodology, we decided to design our product to support this functionality as well and it should work on most mainstream web browsers. *Leap Motion* also boasts accurate measurements of hand gestures, which unlike many other motion detection systems does not have to take into account large body movements. This seemed to be the natural starting platform for a desktop or laptop user, as it would be unlikely that such a user would be willing to perform gestures that require more than just a couple of simple poses he can perform without standing up.

In the sections that follow, we describe the features of ***VeriLeap*** as well as some of its limitations. We also explore the possibility of it being extended and perhaps integrated with other applications or systems, finishing off the paper with a conclusive decision on whether more effort should be directed into 3D motion detection based security systems.

2. Leap Motion

The Leap Motion device is a proprietary consumer-targeted motion detection device intended for home and office use, primarily targeted at applications related to art, music games, virtual and augmented reality [1].

Leap Motion claims to track human hands at up to 200 frames a second [2] accurately. With fairly comprehensive developer support and other explorations with using this platform as a gesture recognition system, we decided that it was the natural choice to design a prototype security system around.

3. DEFINITION OF TERMS

In this paper we will primarily be dealing with gestures of one or more human hands. In order to minimize ambiguity of the different terms used to describe gestures, we define the following:

i. Pose – A static pose that one or both hands of the same individual makes, presented to the motion-capture device

ii. Gesture – A dynamic movement that one or both hands of the same individual makes, presented to the motion-capture device

iii. Action – Either a Pose or a Gesture that has been successfully recorded by the motion-capture device and is used as the basis for authentication against other candidate Poses or Gestures.

4. RESEARCH AND SIMILAR WORKS

4.1.1 Previous Attempts

Initially, we wanted to see if the Leap Motion device could accurately and reliably record biometric features of a human hand. The Leap device could purportedly determine the length and width of each individual finger (and thumb) segment. Hence, we then tried to create a system where a user could present his hand(s), and be automatically logged into his account without any other information. A paper [3] described the unparalleled accuracy of using the Random Forest algorithm to find accurate matches in biometric data. However, we found that the Random Forest algorithm that we used to find matches took far too long (>3 seconds for only 5 entries, with 20 trees) to return real-time results. We wanted to avoid this as we were aiming for a responsive and real-time authentication system.

4.1.2 LeapTrainer.js

We then decided to take another approach and try to authenticate based on gestures alone, with the verification process being independent of biometric measurements. After some experimentation with the LeapTrainer.js[4] library, it was apparent that static gestures could be recognized accurately and quickly with leap motion. We then decided to take the direction of using static gestures (Poses) and short dynamic gestures (Gestures) to authenticate the user.

5. VERILEAP

In this section, we will highlight the implantation details of VeriLeap, as well as why we chose to make certain design choices over others.

5.1 Design Principles

5.1.1 Proof-of-Concept

VeriLeap was designed to primarily be a curt yet solid proof-of-concept that consumer motion detection systems can be used as tools to implement secure authentication systems. Upon plugging in a Leap Motion device, users can then *Register* and subsequently *LogIn* to a server from their browser. VeriLeap encompasses the entire UI/UX components, as well as the back-end verification process on the server side.

5.1.2 Cross-Platform Compatibility

With cross-platform compatibility as part of our main concerns, we decided to implement VeriLeap on JavaScript so that it can be run on most mainstream browsers in use today.

5.1.3 Assumptions

We operate under the assumption that the transmission between client (user's preferred web-browser) and server is secure, as it is not the focus of our product. Also, we assume our database of users and relevant log in information is secure, and do not take into account security controls in this area.

5.2 User Interface

VeriLeap allows a user to *Register* his/her username, followed by three Poses or Gestures recorded by the system sequentially. He/she may then *LogIn* to the system by presenting his or her username, followed by the same three hand-gestures in sequence provided at the time of registration. Upon successful verification of identity, the user will be shown a page indicating that he has successfully logged in.

5.2.1 Registration

Registration is simple and intuitive, with the user being prompted to perform a series of actions before an account can successfully be made. An image of the person's hand is computed on the fly and displayed on the screen for visual confirmation of the hand gesture he wishes to record.

5.2.2 Login

Logging in to the system is straightforward as well, and the user only has to enter his username before being automatically asked to present his three hand gestures one immediately after the other without needing to perform any interaction other than his hand gestures. The user can choose to remove his hand from the Leap Motion device if he needs time to remember his hand gestures, and the system will only proceed once a hand gesture has been detected and recorded.

5.3 Underlying algorithm

A considerable part of the verification algorithm used in VeriLeap draws both code and inspiration from LeapTrainer.js. As such, in this section, we will attempt to describe the functions borrowed from this open source library.

5.3.1 Training Algorithm: Pose versus Gesture

The algorithm recognizes the difference between hand gestures of a static (Pose) and dynamic (Gesture) nature. During the training, frames are continuously recorded, and checked if any major movements are performed.

To discern poses from gestures, the system checks two things:

1. If there is a **hand** moving above the minimum recording velocity
2. If there is a **fingertip** moving above the minimum recording velocity

If any of the above conditions are met, the system will record a Gesture. If neither are a met, a Pose will be recorded instead.

For Poses, the system makes sure that the hand is mostly still before capturing a single frame with details of that particular scene presented to the Leap device, and uses it as data for that particular Action. For Gestures, the system sends all frames detailing the movement of the hand(s), and uses that as data for that particular action.

The frame(s) consists of the x, y and z coordinates of the palm center, and all fingers that are present in the device's view. They are stored in a JSON file and sent to the server side for registration or verification.

5.3.2 Homogenizing of Recorded Action

We resample the gesture in order to create gestures of homogeneous lengths, in order to make different gestures more comparable in the event that the hand is placed at different view distances from the device.

We then scale each homogenized gesture to homogenous variances, so as to provide for differentiation between the same gestures but at different scales.

Finally, we translate the gesture to the provided centroid of the view-space, in order to recognize gestures that are the same but occur at different points in the view.

5.3.3 Recognition Algorithm

When the user attempts to log in, he will be asked to present 3 candidate Poses or Gestures in sequence. For each instance, the system will attempt to first discern if the user has presented a Pose or Gesture using the same algorithm as the one used during Training.

Once this has been done, the recorded data is put through a function that correlates this to the corresponding action stored in the database. The correlation value is a heuristically defined value ranging from 0-100, where 100 indicates a 100% match of candidate and stored data. If the correlation value is beyond a certain value, we say it is a match, and move on to the next gesture. We understand that there cannot be a 100% correlation, and hence have set the correlation threshold to be 91, meaning that anything with a confidence level of 90 below will result in a negative result.

This correlation value is computed as follows:

1. For each finger $\boldsymbol{i}$ in the **recorded Action**, measure the Euclidean distance $\boldsymbol{E_{ij}}$ between every other finger $\boldsymbol{j}$ in the **candidate action**.
2. We multiply this distance by weights $\boldsymbol{W_{ij}}$ based on the number of fingers apart $\boldsymbol{i}$ and $\boldsymbol{j}$ are from each other to get a weighted distance $\boldsymbol{D_{ij} = E_{ij} \times W_{ij}}$
3. Get the sum of all of the weighted distances to get $\boldsymbol{S_{RC} = \sum D_{ij}}$
4. Repeat this for the **candidate Action**, in comparison to the **recorded action** to get $\boldsymbol{S_{CR} = \sum D_{ji}}$. This value is different from the value in (3), given different weightage.
5. Take the minimum of the two values $\boldsymbol{S_{min} = min(S_{RC}, S_{CR})}$
6. This value $\boldsymbol{S_{min}}$ is mapped to a percentage value to give a score ranging from 1-100 to get ***Correlation***. This is the correlation score that we give to the two gestures.

We also make sure the user needs to verify 3 individual Actions in sequence so that the chance for false-positives is minimized. This also makes it such that the chance for an attacked to achieve a fluke verification is minimized, as they would have to achieve this unlikely feat another two times.

5.4 Action Database

The recorded actions for each user are stored in a server-side database, and are accessed during registration and logging in from the VeriLeap client. Usernames are checked to be unique as per standard, so users cannot register with duplicate usernames. Users will also not be able to retrieve their Actions, and can only write actions upon registration.

When logging in, the candidate data is sent to the server to perform the recognition algorithm. The server then sends back a "Pass" or "Fail" indication back to the client. Note that we do not reveal even the correlation value, so that any attackers cannot know by trial and error if they are getting closer to the correct Pose/Gesture. Note also that at no point in time is the recorded Action revealed to anyone but the server itself for use in the verification.

5.5 Known Vulnerabilities

The focus of our work has been directed mainly at the motion-detection portion of the whole system, and hence have left out some potentially vulnerable avenues for attack. The vulnerabilities highlighted in this section have been acknowledged by the development team, and recommends these issues to be ironed out before use in more critical security systems.

i. **Pose/Gesture data is temporarily stored client-side**
When the user Registers with the VeriLeap client, the JSON file is temporarily stored on the client side. This exposes a Man-in-the-Middle vulnerability, as he might be able to retrieve the Action data during the Register phase

ii. **Plaintext transmission of data**
The Pose/Gesture data is transmitted in plaintext to the server, and also stored in plaintext in the database. We understand this is highly undesirable and is unacceptable in secure systems. This will have to be fixed if intended for more critical systems.

6. INTEGRATION AND EXTENSIBILITY

Currently, VeriLeap is presented as a front-facing webpage that only allows users to *Register* and *LogIn* into the system. We believe that it can be easily extended and integrated into a multitude of web-based platforms as an added layer of security or perhaps as a replacement of the whole authentication system.

6.1 Integration

6.1.1 Integration with other Smart Devices

VeriLeap servers can be hosted on a secure location and then used to authenticate users remotely, and then subsequently grant them access to the platform they are intending to log into. This serves an alternative to traditional passphrase verification via entering keystrokes on a keyboard. This opens up security options for not only the Personal Computer, but also the increasingly many types of smart devices that are able to connect to the internet. Smart TVs, game consoles and perhaps even simple household safes could be integrated with VeriLeap so that keyboards are not needed, allowing for a clean aesthetic experience without compromising on security. We believe this can be easily done since VeriLeap runs mostly on JavaScript, which should be able to be handled by any device that can run a modern web browser.

6.2 Extensibility

6.2.1 Improved Training Procedure

Currently, we have implemented VeriLeap to use only one pose or gesture as basis for its training; one training gesture for gesture #1, one gesture again for gesture #2 and so on. This was mainly to enhance the user experience, seeing as to how we expect users to not want to repeat the same gesture too many times. Despite this, we can significantly improve the sample data by making the user present each Pose or Gesture multiple times for each recorded Action. This added training data will make it such that anomalies in gestures get minimized, leading to a reduction in false-negatives during verification.

6.2.2 Two-Factor Authentication (2FA)

2FA can also be easily implemented on our system, where users can be instructed to present specific gestures to the Leap Motion device in place of a traditional One-Time Password (OTP). Users are then be required to enter their mobile numbers or personal e-mails upon registration with the system, and go through the standard means of verifications that they are indeed the owner of the phone number or email account. The instructions sent in the one-time message could be a number of things which include:

i. A picture of the hand pose to be presented to the Leap Device

ii. Textual instructions to perform a specific pose or gesture to the Leap Device

iii. An instruction to present their three candidate Poses/Gestures in a specific order. (for example, the user might be asked to present the 2nd, 3rd, then 1st Poses/Gestures that they provided at registration time instead of 1st, 2nd then 3rd as intuitively expected)

These not only make it such that replay attacks will be more difficult, but also make it impossible for people who are in direct line of sight of your hand-movements to log in simply based on the actions they see you performing to your Leap device.

6.2.2.1 Alpha Implementation of Two-Factor Authentication (2FA)

An alpha prototype of a 2FA system was created for the base VeriLeap program, using a variant of method **i.** from above. We pre-record a number of hand gestures to store in the database as "one-time passwords". The user is then notified via e-mail when attempting to log into his account.

The user is then shown a pose, and asked to perform this hand pose as a replacement for one of his three gestures. A sample email can be seen below.

It is apparent that hand gestures cannot be replicated with high accuracy just by comparing it with a two-dimensional image. We attempted to perform a workaround by lowering correlation threshold the swapped gesture in particular to be at a score of 70. Despite the lowering of the correlation threshold and hence allowing for a higher possibility for brute force attacks to be successful, the value added to the security system from a 2-factor authentication system (from the points in the previous sub-section) is believed to be greater than the potential vulnerabilities.

This vulnerability is also combatted with the implementation of a timeout. Each gesture sent to the user is only valid for a set amount of time (currently 5 minutes), which makes it even more difficult for the attacker to try out a significant number of possibilities.

7. CONCLUSION

Upon preliminary study of the currently explored methods of biometric and gesture recognition capabilities of the Leap Motion device, it was apparent that recognition of basic static and dynamic gestures would be a very responsive security system that could be implemented on a browser based platform.

VeriLeap was designed with all of this kept in mind, seeking to be easily extensible and quickly integrated with existing web applications.

VeriLeap appears to successfully achieve the aims of what we set out to achieve – a cross-platform authentication system implemented on a household motion-detection device. We consider this proof-of-concept as a success, and believe that it can definitely serve as a prototype for future security systems developed of this nature.

8. ACKNOWLEDGMENTS

We would like to express our appreciation for the author of LeapTrainer.js, Robert O'Leary, for without his work we would have no basis to build VeriLeap upon.

Our sincerest thanks to Hugh Anderson (Associate Professor), for his guidance and help in making this work a possibility, and to the School of Computing, National University of Singapore for providing us with the material means and support, without which this work would not have seen completion.

9. REFERENCES

[1] Leap Motion homepage
https://www.leapmotion.com/about-us

[2] Leap Motion product desciption
https://www.leapmotion.com/product/desktop

[3] Leap Motion for Authentication via Hand Geometry and Gestures
http://www.springer.com/cda/content/document/cda_downloaddocument/9783319203751-c2.pdf?SGWID=0-0-45-1518080-p177470387

[4] LeapTrainer.js by Robert O'Leary, under open-source license by MIT
https://github.com/roboleary/LeapTrainer.js