Exploring Applications of Microsoft's Xbox Kinect:
Using a Depth Map and Feature Points as a Password

Ahmad Faiz Abdull Rashid
Victor Ferrero

Abstract

As more and more personal information is stored in the cloud the need for enhanced security is at an all-time high. With increased computing power, hackers can now figure out passwords much more easily than before. This coupled with the added pay-off that hacking can have has created a situation ripe for stealing our personal information. The days of passwords as strings of characters are numbered, and we must look to alternate methods of securing our computer systems. In this paper, we analyze the effectiveness of using a picture of a person as that person's password. Specifically, we explore the potential effectiveness of using a depth map along with twenty feature points in the human skeleton to create a pose based password.

Introduction

Passwords are nothing new to the world of computers. Present notions of a password are that it is a string of characters chosen by the user. Recently emerging technologies are attempting to change this notion, most notably Microsoft's Surface tablet uses an image upon which the user must make some sort of gesture on the tablets glass screen in order to gain access. Ideally, we want something that only the user can recreate, and can recreate easily. An image of the user where biometric information is extracted and/or taken into account has potential to fulfill the aforementioned requirement.

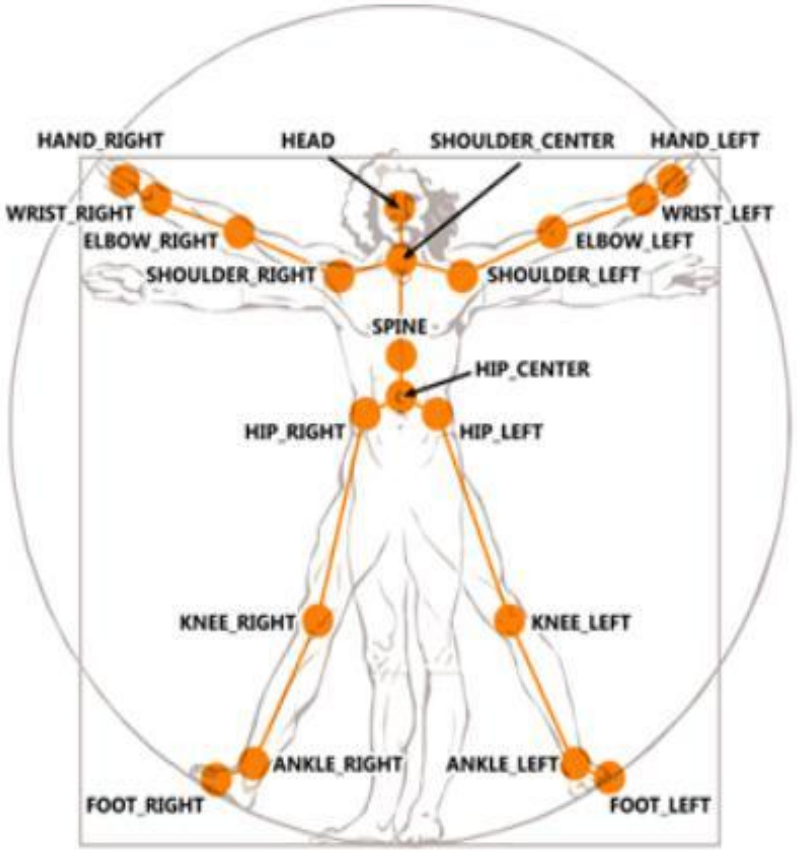
Implementation

Terms and Definitions

"User" will be used to refer to the person who has set the password
"Person" will refer to anyone who has not set the password

Microsoft Kinect (very brief overview)

In this project we used the Xbox Kinect to create a depth map and extract twenty feature points from the human skeleton. Each feature point is given an (x,y,z) value in millimeters. The (x,y) correspond to millimeters away from the optical center, z is the distance from the camera.
The location of the twenty feature points on the human skeleton are shown in the following image:



The Kinect accomplishes this task in two stages. First it produces a depth map which it creates by projecting a known IR pattern onto the scene. By measuring the deformation of this IR pattern, the Kinect is able to assign a Z distance component to every pixel in the image. The feature points are assigned using the depth map to locate possible human skeletons, and a Decision Forest which actually assigns the feature points. For more information about the Kinect, see Microsoft's documentation. All of our code for taking the image uses the Xbox Kinect SDK (version 1.6) along with the .NET framework (version 4.0). The (x,y,z) values for each point are saved to a text file when the image of the user is taken. We also saved RGB images for each photo taken. Both RGB images, as well as the text file feature vectors will be made available.

The training set

The most important element of a password is the ability of the user to recreate it in the future. Due to human error and hardware error it simply is not practical to expect a user to exactly recreate each X, Y, and Z value for all twenty feature points. In light of this observation we had to find a way to deem a user's login attempt as "close enough" to their set password. To solve this problem we used a training set along with dynamic thresholding and thresholding constants. The training set consists of ten to fifteen images of the user. It is very important that the user move his arms in between each image before resorting back to the original pose. This is intended to create natural variation in a person's ability to recreate their password. The average (x,y,z) values for each feature point in the training set is what was used as the feature vector for the password. Expressed more formally, the password is:

Feature Point p1 X = (x1 + x2 + x3 + ….. + Xn) / N
Feature Point p1 Y = (y1 + y2 + y3 + ….. + Yn) / N
Feature Point p1 Z = (z1 + z2 + z3 + ….. + Zn) / N
Where N = number of images in the training set
This is done for all twenty feature points.

Each feature point's X, Y, and Z value is given a maximum allowed threshold. If future replications of the feature point's (x,y,z) values fall within these allowed threshold then the feature point is considered to match the one in the password. This threshold is calculated by searching through the training set and finding the users worst replication of the individual X, Y, and Z values for a given feature point. More formally:

X threshold for Feature Point p =
max(|x1-x2|, |x1-x3|, …. , |x1-x20| , |x2-x3|, |x2-x4|, …. |x19- x20|) + X threshold constant
where Xn is that feature points X value from the Nth image in the training set

Y threshold for Feature Point p =
max(|y1-y2|, |y1-y3|, …. , |y1-y20| , |y2-y3|, |y2-y4|, …. |y19- y20|) + Y threshold constant
where Yn is that feature points Y value from the Nth image in the training set

Z threshold for Feature Point p =
max(|z1-z2|, |z1-z3|, …. , |z1-z20| , |z2-z3|, |z2-z4|, …. |z19- z20|) + Z threshold constant
where Zn is that feature points Z value from the Nth image in the training set

The constants that we experimented with were X=200mm, Y=80mm, Z=300mm. These constants are very important for several reasons:

1. Some feature points just do not create enough variability. These are typically points that are not very involved in the user's gesture: head, spine, hip etc. But, in future replications of the password these points may still have some small variability given that the user is not standing in the exact same position as they did during the training set.
2. Because the user did not physically move closer or farther from the camera during the training set (this was intentional), some Z variability was observed, but just not enough to allow for reliable recreation of the password in the future.
3. They significantly aid the user in future replications of the password, without significantly increasing the rate of false positives given that the user and the other person attempting to login have different body proportions

Examples

User successfully recreates password:

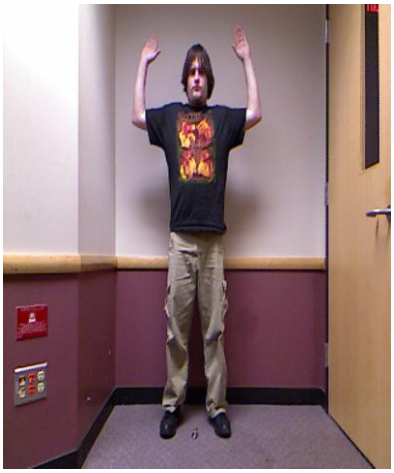


Figure 1

Figure 2

Figure 1 was an image from a training set of 15. Figure 2 was an image outside of the training set. Our algorithm classified Figure 2 as a match.

User fails to recreate password:

Figure 3

Figure 4

Figure 3 is the same image as figure 2. Figure 4 is deemed not a match.

Person recreates password

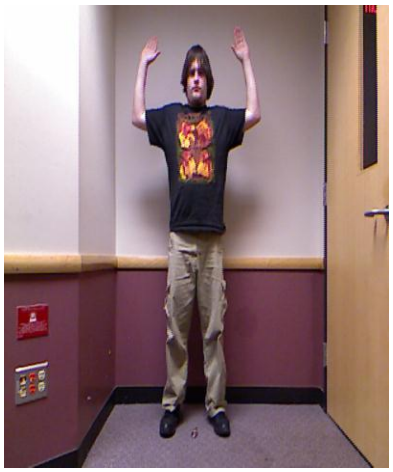


Figure 5

Figure 6

Figure 6 is incorrectly deemed to be a match of Figure 5(same image as figures 3,4)

Person fails to recreate password:

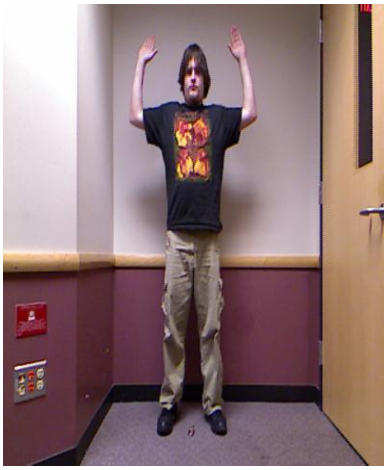


Figure 7

Figure 8

Person fails to recreate password:

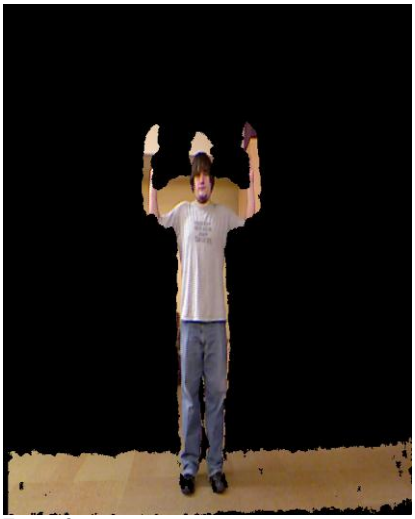


Figure 9

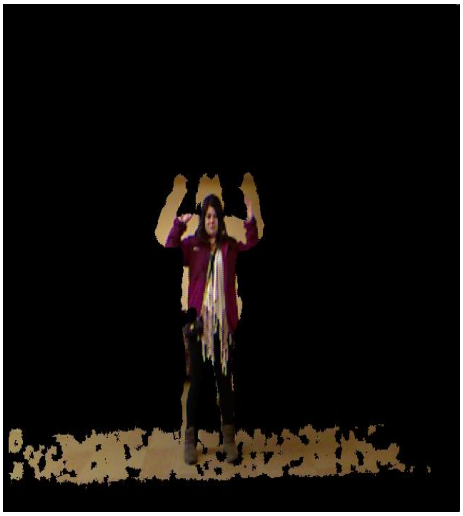


Figure 10

Figures 9 and 10 were taken with an old version of our program which filtered out the background. All twenty feature points still get an (x,y,z) value. Each are from a training set and neither one will match the other one's training sets. This is a good example of a situation where even though the user's pose is known, if the body proportions are different then the password is not matched.

Limitations

This method of using a depth map and feature points as a password does have some limitations. The main limitation that was observed occurs in the case where someone who is logging in has similar body proportions as the user. In this case all the person needs to do is successfully recreate the pose that the user was in when they set the password. Granted this extra bit of information (the pose) could prevent an unauthorized login, the fact remains that if the person has similar body proportions they have the ability to recreate the user's password. This represents a security concern.

Other limitations to consider: practicality and camera location since the (x,y,z) values of each feature point are determined relative to the optical center and distance from the camera, it becomes very difficult for the user to login if the camera has moved because they must now take this into account.

Improvements

1. Encryption. Encrypting feature vectors was beyond the scope of this project, but should be a part of any real world implementation
2. Incorporate depth maps and facial feature point extraction as a means to reduce or ideally eliminate false positives. This could also allow for a facial pose to be a part of the user's password
3. Assign each feature point its own X, Y, and Z threshold constant instead of using the same constant for each feature point

Conclusions

Our findings lead us to conclude that a depth map along with feature point extraction does have the potential to be used as a password in much the same way that a string of characters is used. The ability of the user to reliably recreate his password was shown which demonstrates that this is a viable option. We do acknowledge that this presented mechanism is not flawless. A person with similar body proportions does have the ability to recreate a user's password. However, it is important to realize that current passwords (strings of characters) are not flawless either. If some person knows the correct string of characters then they are guaranteed to be able to break the password. However, in a real world implementation of our password, simply knowing what the password is does not guarantee access. One must know what the pose of the user was, his distance from the camera when the password was generated, as well as be able to physically recreate the user's pose which require that they have similar body proportions as the user. Upon closer examination it would appear as though our mechanism has potential to be more secure. We believe that this project demonstrates the effectiveness of using an image of a person as a password, and hope that it insights more research in this field.