AirMouse: a touchless input device built on the top of the physical mouse idiom

Giovanni Cuffaro
E-mail address
cuffaro.giovanni@gmail.com

Abstract

In this project I studied how to emulate the behavior of a physical mouse using the 3D hand model generated by the Leap Motion device. I used machine learning techniques for recognizing the "mouse grab" pose and a small set of mouse gestures. The main idea is to simulate the mouse behavior without touching any physical object but using the same mouse idiom learned using a computer. The touchless mouse or AirMouse was evaluated with a usability test to assess user opinions and by a set of timed tests to collect an objective measure of the performance and an estimation of the learning curve.

1. Introduction

AirMouse is an input system which uses the Leap Motion Controller[1] 3D hand model to emulate the hand poses and gestures of physical mouse without touching or moving anything. The Leap SDK creates a 3D model of user's hand that is used, thanks to machine learning techniques, to recognize poses and gestures in order to move the pointer and perform actions usually related to a physical mouse.

My work rests on the concept expressed in [1]: users have deeply internalized the concept of mouse interaction as if it was a feature of the physical-natural world. Almost everybody knows how to use a mouse and my work assumes that it should be easy to use a touch-less device if the hand pose and gestures are similar to those they are familiar to.

AirMouse is able to distinguish when the user wants to move the pointer from when he is entering or leaving the sensor area; it allows the user to simulate a click, left or right, to complete a drag and drop operation and to scroll a page up and down.
The last section of this paper reports the usability test protocol and results with a brief discussion on the average user experience.

[1]http://www.leapmotion.com

Related work

Most of the work concerning the Leap Motion controller is about gesture recognition to create gestural input to access specific functions [2] or only in certain application, like a music player [3]. Instead, I propose a solution to use the sensor as a cursor-like navigation device. Software available on the Leap App Store [4, 5, 6] virtually simulate a touch screen in front of the user, proposing different gestures similar to "point and tap". The main difference with the other projects is that *AirMouse* wants to emulate the poses and gestures the user does when using a physical mouse. In addition most of the proposed solutions are not capable of understanding user's intent to interact with the system while I tried to solve this problem using a null rejection strategy.

2. Tools and techniques employed

The main tools and techniques I employed are:

Leap Motion controller a natural HCI device which creates a 3D model of the hands, described in section 2 [7].

Leap Motion SDK for accessing the 3D model data generated by the Leap Motion Controller.

Gesture recognition toolkit a machine learning library specifically designed for real-time gesture recognition[8].

XLib for controlling the mouse pointer behavior programmatically (movement, click, scroll wheel)[9].

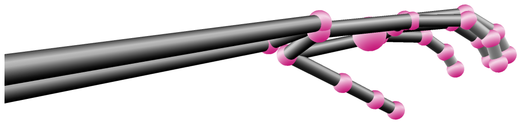
Leap Motion Controller

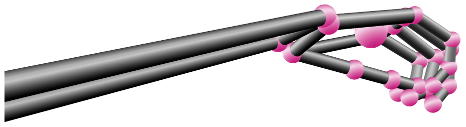
The Leap Motion controller is a small USB device which is designed to be placed on a physical desktop, facing upward. It uses three infrared LEDs which generate patternless IR light and 2 cameras able to acquire up to 200 frames per second.
The Leap Motion software receives the video stream and synthesizes a 3D hand model by comparing the 2D frames

generated by the two cameras (stereo vision). Leap Motion SDK allows the developers to access both high level data, like predefined gestures and derived features, like grab strength, and low level model information like tips position, velocity and angles between fingers.

Static Gestures

In this work I considered 3 static hand poses: *Mouse Grab* gesture, shown in figure 1(a), employed to move the mouse pointer and the *Pinch* gesture (with all the fingers), shown in figure 1(b), to start a drag operation.
A third pose was introduced to aid the classifier null rejection by giving it some example of an open hand, figure 1(c), meaning to stop moving the pointer and release if dragging.


(a) Mouse grab pose (move pointer)


(b) Pinch pose (drag)

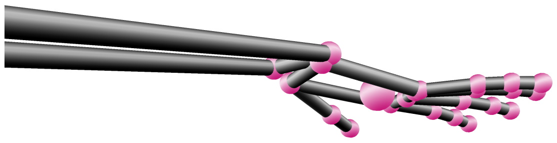

(c) Null hand pose (release control)

Figure 1. Static poses visualized through the 3D Leap Visualizer

Support-Vector Machine Classifier

Static poses classification and prediction is accomplished by a SVM classifier. Support-Vector Machine (SVM) is a supervised learning algorithm used for classification and regression analysis, based on the mapping of characteristics extracted from instances (feature vectors) to points in space. SVM create a set of hype-planes in a multi-dimensional

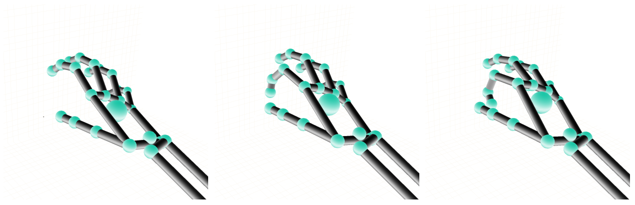


Figure 2. 3 frames grabben during the click gesture and visualized through the 3D Leap Visualizer

space, which divide points into different classes.
Support Vector Machines is a maximum margin classifier, because the resulting hyper-planes maximize the distance between the ”nearest” vectors of different classes. These vectors take the name of support vectors. In addition to performing linear classification, SVMs can perform a non-linear classification using what is called the *Kernel trick*, mapping inputs into high-dimensional feature spaces. For this purpose I used the Radial Basis Function kernel (RBF). In my work I used SVM implementation of GRT library which is based on libsvm[2].

Features

The feature vector is composed by a global element indicating how close a hand is to being a fist, generated by the leap motion SDK, and, for each finger, the 3D position of the tip relative to the stabilized[3] position of the palm center and the angle between the finger and the estimated direction of the hand.
These features are invariant enough to classify gestures from different people and they give a stable response on slightly modified poses.

Training

The training set was created by asking 10 people to replicate the 3 hand poses and recording about 1000 samples for person for each pose while moving their hand above the sensor.

Dynamic Gestures

Dynamic gestures are used for detecting left and right click simulating the way in which the user press the buttons of a physical mouse. I used the continuous HMM as a classifier for triggering the mouse click, one classifier for the right and one for the left button. The scrolling gesture does not emulate the mouse behavior to keep the system simple

[2]https://www.csie.ntu.edu.tw/ cjlin/libsvm/
[3]The SDK provides both stabilized and raw position

and usable. It is triggered by a rotation of the open hand perpendicularly to the horizontal plane, clockwise rotation stands for scroll down while anti-clockwise stands for scroll up. Scrolling gesture is generated from the finger circle gesture available directly from the Leap Motion SDK.

Continuous Hidden Markov Model

HMM (Hidden Markov Model) is a statistical Markov model in which the system being modeled is assumed to satisfy the Markov property[4] process with unobserved (hidden) states.

Features

Because of the dynamic nature of the click gesture, I used as features the 3D fingertip (index or middle) speed relative to the other 4 fingertips velocity. The feature vector is basic and needs a quick and clean movement to be triggered but it minimizes the false detection rate.

Training

HMM training require a significant number of samples so the HMM model has been trained on my own hand by recording 100 samples for right and left click gesture. I assumed that the feature vector is enough invariant to different shaped or sized hands because it is only related to fingertips relative speeds.

Implementation

Static poses

The SVM classifier is used to detect the hand pose. If it recognizes the mouse grab pose the program will start to move the pointer which follows user's palm position. I used the non-stabilized palm position because it allows a better control over the pointer and, even if it shows a rougher interaction, the feedback is far better.

The response in motion of the pointer is based on the hand movement multiplied by a configurable speed factor. In order to enable precision movement of the pointer, the speed factor is decreased when the user does slow movements and reset smoothly to its default value when the movement become faster. In addiction, when the hand movement is bounded to a small area, a *dead zone* is activated to let the pointer stay still even if the user is not able to keep the hand perfectly stable. If the classifier detects the pinch hand pose the system starts moving the pointer as described before but emulating the holding down of the left mouse button (dragging). When the classifier detects the open hand pose the control is released and if the system was in a dragging status the left button is virtually released to stop the dragging operation.

[4]Meaning that the future state depends only on the previous state and on probability of transition between those states

Dynamic gestures

The click gestures are detected by 2 HMM classifiers, one for the left click and one for the right click. HMM usually triggers more than once for each click so I added a debouncer function. The debouncing time can be set in the configuration file. The scroll gesture is derived directly from the SDK finger circle gesture. I added a control on the circle completion in order to start the scrolling operation only after at least an entire circle is drawn. The amount of scrolling is proportional to the radius of the circle enabling precision scrolling.

3. Usability test

The users are asked to complete a series of routine tasks supervised by a moderator with the aim of estimating system:

- Learnability
- Easiness of use
- Accuracy
- Fatigue and Comfort
- Consistency

During the usability test the right click gesture was disabled to simplify the interaction.

3.1. Methodology

Before starting the test the system is introduced to the user through a video presentation of the hand poses and gestures while the supervisor explain briefly the main features of the device.

He also gives some useful tips for the best experience and answers user's questions, if any.

Test protocol

The testing protocol is composed by 5 consecutive parts:

1. **Timed test:** 5 colored squares are presented on the screen, the user has to follow the instruction on the screen and click the right square. The application registers the total time spent to complete the 16 instructions, the average time for clicking a different square from the previous (move and click) and the average time to click the same square as previous (click). Clicks recognized while on the wrong square are considered errors and recorded. Figure 5 shows the main view of the test.

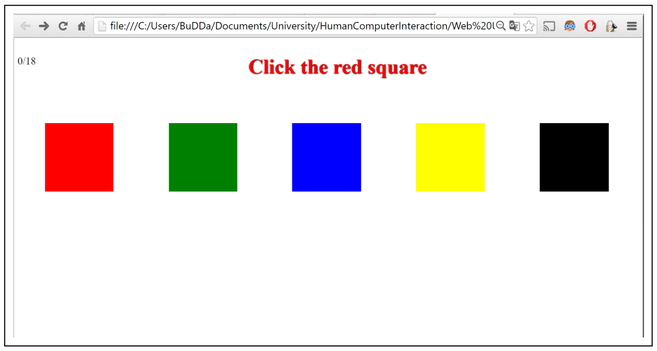
2. **Task oriented test**: The user is asked to complete a series of routine tasks, the supervisor is allowed to help the user by giving some tips.

   (a) Open Firefox browser
   (b) Open a new tab (plus button in the tab bar)
   (c) Open the website in the first panel
   (d) Scroll down until a red button is found
   (e) Click the red button (it will open Google News)
   (f) Click the second news
   (g) Close the tab
   (h) Open a new tab
   (i) Swap the third and the fourth panel
   (j) Close Firefox

3. **Timed test again**

4. **Free use:** The user is left free to use device to navigate Google Maps; the suggested task is to find his house on the map (by dragging the map / zooming in and out) and than use Google street to navigate in front of the house main door.

5. **Last timed test**


file:///C:/Users/BuDDa/Documents/University/HumanComputerInteraction/Web%20

0/18

Click the red square

Figure 3. Main view of the quantitative test showing the instruction in red and the 5 colored squares.

The study involved 16 participants between the ages of 20 and 40 who are unfamiliar with this project.
The location of the usability evaluation is a room where the user can test the system in privacy and without too many distractions. The Leap Motion controller was placed on the table in front of a big wall screen connected to the computer; the user was asked to use the device standing in front of the screen.
After have completed the test a little reward is given to the users to thank for their time and patience.
The test is supposed to require about 10 minutes.

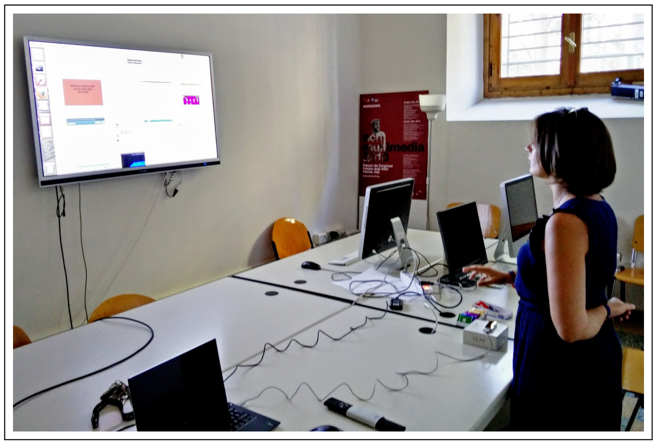


Figure 4. Picture taken during a test. The user is standing in front of the screen while using the Leap Motion device.

3.2. Results

Qualitative test

The system usability is evaluated by asking the users to answer 19 SEQ[5] over a 7-point Likert scale to assess their opinion. For negative scores I asked explanation in order to understand better the issue.
Most of the participants (80%) are satisfied with the AirMouse system. The majority of participants agree they learned to use the input device after a little practice and that most people will learn to use it quickly even if half of the users think the help give during the test is important to understand the working principle.
Nobody reported difficulty remembering the gestures or felt too much uncomfortable during operation but the position and the posture are considered good by very few users (25%). Most of the participants agree the test required a fair effort and tasks completion did not require too much mental concentration. Most of them complained especially when using the click gesture on very little GUI elements, despite the accuracy of the device is considered sufficiently positive.
Even though participants' average rating is 5.1/7 not everybody liked AirMouse conceptual similarity with a physical mouse (1.8 standard deviation of the answers). Operation speed and smoothness didn't satisfy completely the participants, only 44% of them think they completed the tasks quickly. See table 3.2 for a detailed report. Table 3.2 reports the fatigue ratings. Arm is the most strained part followed by the wrist and the hand, especially the index finger, but the physical effort is considered acceptable by most of the participants; only one of them somewhat agrees that actuation required a lot of force.

[5]Single Ease Question

N	Question	Mean rating	σ	% Agree
1	Task completion required too much effort	3.2	1.6	19%
2	Task completion required mental concentration	3.6	1.3	19%
3	I was able to complete the tasks very quickly	4.2	1.5	44%
4	The device is accurate	4.9	1.4	69%
5	Actuation required a lot of force	2.1	1.2	6%
6	The operation is very smooth	4.6	1.5	56%
7	I felt very uncomfortable during operation	2.5	0.8	0%
8	The system is very easy to use	4.5	1.2	50%
9	The gestures and hand poses are difficult to remember	1.7	0.8	0%
10	I learned to use the system better after a little while	5.7	1.3	81%
11	The system is illogical or inconsistent	1.3	0.5	0%
12	Position and posture are comfortable	3.9	1.2	25%
13	Help given during the test is very important to understand how the device works	4.9	1.2	50%
14	Most people will learn to use the system quickly	5.3	1.3	75%
15	Overall, I am satisfied with the system	5.7	1.0	81%
16	I like the similarity with a physical mouse	5.1	1.8	63%

Table 1. Detailed report. Single Ease Question on a **1** *(Strongly disagree)* to **7** *(Strongly agree)* scale.
Percent Agree: (%) Ratings greater than 4

Body part	Mean rating	σ
Hand fatigue	3.1	1.6
Wrist fatigue	3.2	1.4
Arm fatigue	3.9	1.5

Table 2. Detailed report about fatigue.

Mean time	I Attempt	II Attempt	III Attempt
Total	67 557	54 693	50 525
Click	2 927	2 265	1 614
Move & Click	5 100	4 009	4 019

Table 3. Quantitative test, average timings in milliseconds.
Click is the time between two click on the same square.
Move & Click is the time between two click on different squares.
Each error in considered as a 3 seconds penalty on total time.

Quantitative test

The timed test aim is to understand how fast the average user is able to learn to use the device. The test is focused on the click gesture and on the switching between the mouse grab pose and the release/null pose.

As described in 3.1 users were asked to repeat the test 3 times: at the beginning, after the task oriented part and as last step of the test. The majority of the participant understood very quickly how the mouse grab pose worked and how to switch to the null pose but the click gesture required a lot of effort initially.
After some advice most of them realized the best movement to perform and they were able to complete the 16 instructions in a reasonable amount of time. The plot in figure 5 shows that going from the first attempt to the last the average time decreases. Most of the users improved their time in the second attempt but some of them made a worse result during the last test. This can be explained by the hand/finger fatigue after 10 minutes of continuous operation.

Table 3.2 reports the mean recorded value fort the first, second and third user's attempt.
According to the results all of the participants are able to enhancing their ability improving the total time of about 18 seconds if we compare the first try with the last. Click gesture is considered difficult but, according to the results, most of the users keep improving their ability during all the test, going from an average of 3 seconds during the first attempt to 1.5 of the last. *Move & Click* timing highlights that moving the pointer is easier to learn than the click gesture, after a little improvement from the first to the second test, there is no further enhancement in the third attempt.

Conclusion

In this study I tested a different idea for touch-less pointer-like interaction with a computer: *AirMouse*.
The results of the test show that the learning curve of the system is good, the idea to consider mouse poses and gestures as natural interaction can be interesting. Leap Motion provides great accuracy for static hand and fingers but the data for the moving hand are usually noisy and dynamic gesture recognition suffers because of this roughness. It has not been possible to simulate a physical mouse in all of its parts because of the sensor accuracy and the different interaction caused by having nothing physical in your

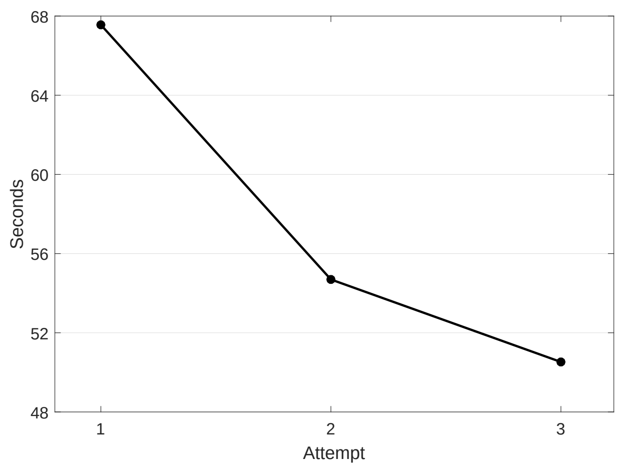


Figure 5. Plot of the average completion time (with 3 seconds penalty for each error) of the 3 timed tests.

hand while using the system, for example the pinch pose is used instead of simulating the continuous left click and the circle gesture is used instead of simulate the mouse wheel scrolling.

The usability test showed some weaknesses, in summary:

Click gesture: at the beginning all of the participants had some troubles learning how to perform the click gesture correctly because of the lack of feedback given by the absence of a physical item in user's hand. This gesture caused annoyance to most of the users.

Precision click: clicking usually causes unwanted hand movements and the precautions taken[6] often were not enough to avoid the pointer to move out the button. Only some of the participants waited the system to stabilize the cursor position before clicking.

Drop after drag: releasing the drag causes the same problems of the click making difficult to perform precision drops.

A touch-less interface like AirMouse has to deal with the lack of perceptible affordances: users are usually not able to discover by themselves how to interact with the system, they can only try some actions and wait for a feedback from the system. There are no signifiers, you can't just imagine how to scroll up and down, and there are no buttons to click. The lack of physical constraints makes easy to move the hand outside the viewing area of the Leap Motion and creates confusion about the right hand-pose because there is nothing physical to grab.

In conclusion I can affirm that the used idiom makes the learning curve smooth, despite some issues with the click gesture recognition, but the posture during the operation does not allow a prolonged usage.

AirMouse may be a useful alternative in situations in which using conventional input devices (mouse and touchscreen) is difficult for example in public spaces where physical objects are dirty and can be easily vandalized. An other scenario may be the computer usage during a surgery in a a sterile operating room without physically touching a device and destroying the sterile field.

[6]The dead zone and the speed factor reduction

References

[1] Hans-Christian Jetter, Harald Reiterer, and Florian Geyer. "Blended Interaction: understanding natural human–computer interaction in post-WIMP interactive spaces". In: *Personal and Ubiquitous Computing* 18.5 (2014), pp. 1139–1158.

[2] Leap Motion. *Shortcuts*. 2014. URL: `https://apps.leapmotion.com/apps/shortcuts/`.

[3] mareXim. *AirWMP*. 2016. URL: `https://apps.leapmotion.com/apps/airwmp-the-windows-media-player-plugin/`.

[4] Leap Motion. *Touchless*. 2013. URL: `https://apps.leapmotion.com/apps/touchless-for-windows/`.

[5] Touchless. *AirInput*. 2014. URL: `https://apps.leapmotion.com/apps/airinput/`.

[6] Nu-Tech. *Mudra Mouse*. 2016. URL: `https://apps.leapmotion.com/apps/mudra-mouse/`.

[7] Inc Leap Motion. *Leap Motion*. 2016. URL: `https://www.leapmotion.com`.

[8] Nicholas Edward Gillian and Joseph A Paradiso. "The gesture recognition toolkit." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 3483–3487.

[9] *Xlib*. URL: `http://www.x.org`.