Final report of internship

JINQING XU

**Variable Definition Table**

|  |  |
| --- | --- |
| Variables | Definition |
|  | The normal vector of the tablet plane. |
| trial | Starting after clicking start button and ending when final lift up after a successful selection. |
| attempt | Starting after touch down and ending when lift up no matter selecting the target successfully or not. |
| entryFreq | The frequency of entry. |
|  |  |
|  |  |
|  |  |

**The Communication of leap motion and android**

* Challenge

There are still no SDK for the connection of leap motion and android.

So the two devices collect data separately. However, we need to find the relationship between 2D data and 3D data.

* Method

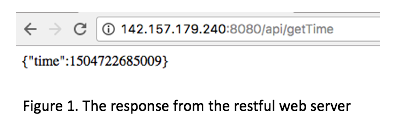
Since the android and leap motion programs run simultaneously, we can use the timestamp to locate the start and end of a trial in the leap motion data.

* Difficulties

We use the timestamp on the laptop to represent the time of a frame from leap motion data. And we use the system time of android to represent start-time and final-lift-up-time of a trial from android data. But the time of laptop and android can not always be simultaneously. Even if I do adjustment, the difference of timestamp will be greater as time goes on due to the difference of clock frequency in different devices. We found that the clock frequency in tablet is somehow slower than that in laptop.

* Solution

The solution is to use the internet to find the difference of timestamp dynamically. Firstly, I started a restful web server on the laptop. Whenever a GET request is received, it will respond with the current timestamp of the laptop as shown in Figure 1.

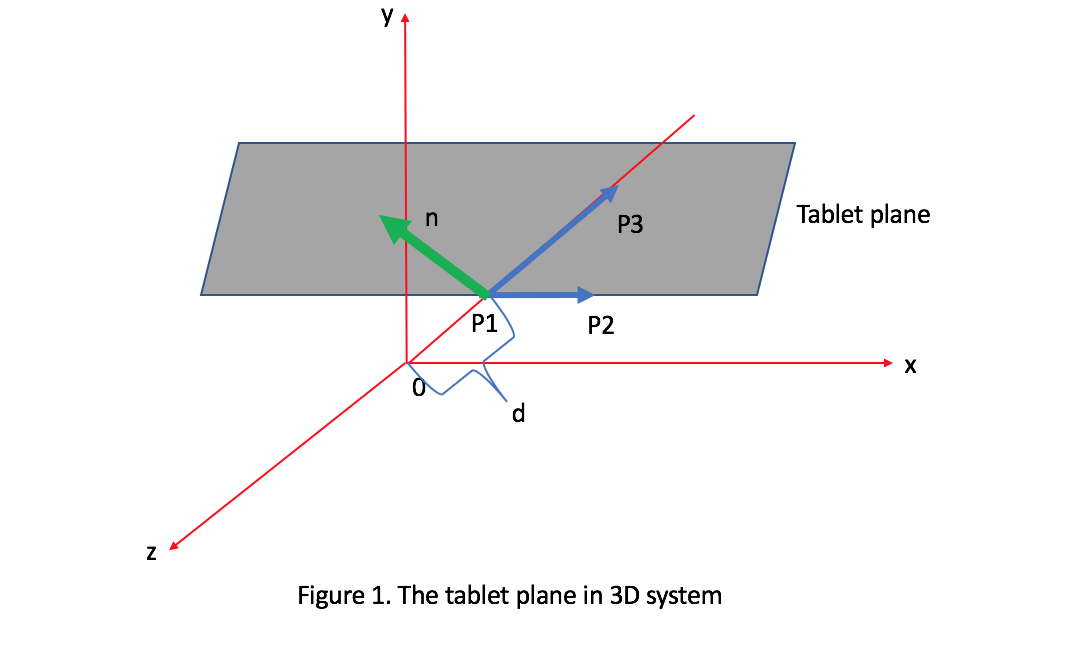


Secondly, at the initial stage of android program, it sends GET request to the web server to get the current timestamp of laptop. By comparing the timestamp from android system to that from laptop, we can find the timestamp difference of android relative to laptop. After that, each time we record timestamp in android system, we add that difference.

There is a tricky problem. After we send a request, there will be much network delay before we receive the response. To ensure the accuracy of timestamp from laptop, we need to take that delay into consideration. Before we send the request, we use beginClock to record the start time. After we receive response from the server, endClock is used to represent the end time. So Round-Trip Time is endClock minus beginClock. Since we get the system time of android after being responded, the delay should be half of RTT. Then the real timestamp from laptop should add up the delay on the base of the response value. Finally, we get the difference of timestamps from two devices.

**Measures from Mackenzie:**

* Variables that need to be defined in advance
* The normal vector of the tablet plane.



1. suppose P1,P2 and P3 are three points on the tablet.

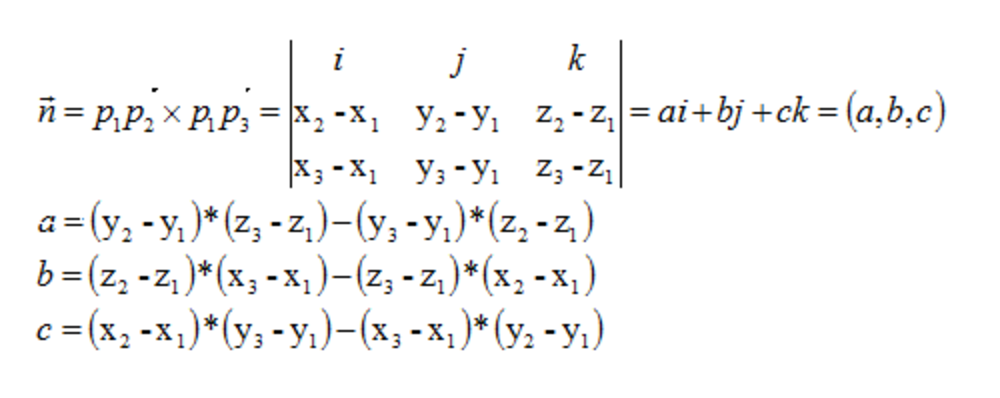
Let P1= (x1, y1, z1) P2 = ( x2, y2, z2) P3 = ( x3, y3, z3 )

Then

vector P1P2 = ( x2-x1 , y2-y1, z2-z1 )

vector P1P3 = ( x3-x1, y3-y1, z3-z1 )

Let be the normal vector of tablet.



1. Let the distance between O and P1 be d

Then P1 = ( 0,0,-d ) P2 = ( 1,0,-d ) P3 = ( 0,1,-d-1 )

1. Finally, we got the normal Vector of tablet = (0,1,1)

* Target Re-entry
* Definition

TRE means first enter the target, then left, and re-enter again. TRE is detected when the finger slides on the tablet. An entry behavior means the start point of the movement is outside the target and the end point of the movement is inside the target. Let **entryFreq** be the frequency of entry. We cumulate **entryFreq** and each time when it is above zero, we minus its value by 1 to get the re-entry frequency.

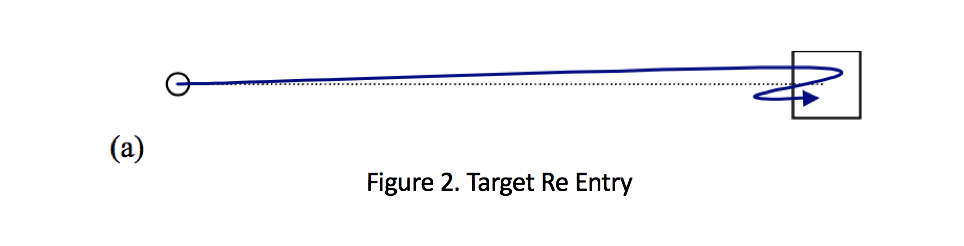
* Measures

1. TRE

The average re-entry frequency per attempt.

1. First TRE

The re-entry frequency of the first attempt.



* Movement Direction Change
* Definition

The direction change during a movement. There are changes in X, Y and Z direction.

The leap motion measures X, Y and Z value of the fingertip. When the current x is smaller than the previous one, the direction is **Left.** When the current y is smaller than the previous one, the direction is **Down.** When the current z is smaller than the previous one, the direction is **Backward**.

We record the previous direction and each time the current direction does not equal to the previous direction, a Movement Direction Change happens. Replace the previous direction with the current direction at the end of each loop.

* Task Axis Crossing
* Definition

The number of times that the paths of the finger cross the task plane.

Initially, we judge

* Movement Offset