Vol.19 No.3, 2017

Original Paper

An Algorithm to Detect Midair Multi-Clicks Gestures

Hani Karam[*1], Jiro Tanaka[*2]

Abstract – Selection mechanism gestures are used in Natural User Interfaces (NUI) to designate elements in a User Interface. They usually involve simple gestures with limited interactions. Being able to use more than one gesture simultaneously increases the vocabulary of the interactions. In this paper, we present MultiX Click, a new algorithm to detect midair multi-click gestures. Our approach allows the detection of multiple midair finger clicks using a depth sensor. To show the potential of our algorithm, we implemented a midair multi-click keyboard and a midair piano that use simultaneous multi-clicks. In the midair multi-click keyboard, we mixed single and multiple clicks with the ability to retrieve the location of a click, and as a result we were able to increase the gesture vocabulary. This paper explains in detail the algorithm we used to detect multi-clicks. We also explain about some preliminary experiments for evaluating it.

Keywords : Midair click detection, Multi-click Detection, Natural User Interface, Leap Motion, Midair keyboard

1. Introduction

Natural User Interfaces (NUI) involve using the body as a means for inputting command to a computer system. They usually consist of some kind of sensors that are used to detect specific gestures which would be translated into commands. In traditional Graphical User Interfaces (GUIs), users point to items (buttons, icons, etc.) by using an input device, usually a mouse, and select a given item by performing a click operation. However, in NUI, hand gestures are mainly used as a way to interact with a system. It has been suggested that gestures make an intuitive and natural way to communicate with a computer [1]. A User Interface (UI) usually contains multiple elements, and a selection mechanism, just like a GUI's mouse click, is needed to designate a given item from a UI; this applies to NUI as well. Many gesture-based selection mechanisms have been proposed [2]~[5]. Most of the existing selection gestures are unintuitive or limited to one selection command, and cannot be used to make a more interactive UI. Moreover, existing selection mechanisms miss the ability of performing multiple simultaneous selections. Adding this capability can increase the selection gestures vocabulary by allowing combinations of simultaneous gestures to define new interactions. In this paper, we present MultiX Click , a new approach to detect simultaneous explicit finger clicks using a depth sensor. We also implemented a midair multi-click keyboard and a midair piano as applications to our algorithm.

*1: Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba [hani@iplab.cs.tsukuba.ac.jp]
*2: Graduate School of Information, Production and Systems, Waseda University [jiro@computer.org]

2. Related Work in Selection Mechanisms

It has been suggested that gestures can be a good candidate for controlling a computer [6], [7]. Wachs et al. [1] point out that gestures should be designed to be intuitive for users; that is, they should be designed to resemble the action they are linked to. Kato et al. [2] used a pinching gestures as a selection technique. While efficient, it is limited in terms of usability, as it can only be used for simple selections. A similar pinching technique was used in [8] to select and drag windows in a user interface, and suffers from the same limitations. Lee et al. [9] defined a click by identifying the index finger, then determining whether it got closer to the camera and back. This makes it limited to the index finger, and cannot be used for precise interactions. [3] detected thumb and index clicks which were used for basic selection only. In their research [4] Harrison et al. detected clicking gestures with depth cameras by using flooding techniques; that is, a click is only detected if the finger executing the gesture touches a solid object. This forces the users to touch an object and the interaction cannot be performed in midair. A midair finger click interface was developed for Augmented Reality systems [10]. The only usable finger was the

index, and the interaction was limited to just a selection mechanism. [5] uses double crossing, an improvement over regular crossing, where a target must be crossed twice to be selected. This reduced false detections, but is constrained to a basic selection. Kulshreshth et al. [11] count fingers to select an item that was previously assigned a specific number as its ID: when the number of extended fingers matches the ID, the item is selected. This forces the items to be numbered, and is only limited to ten elements. Moreover, the fingers cannot be used for any other gestures as they are dedicated for counting. Additionally, in all the above mentioned works, the selection interactions are sequential: users cannot perform simultaneous multiple selections, and thus the interaction is limited to selections that are executed one after the other.

3. Our Previous work

In this section, we introduce our previous work that is related to our current research. In [12], we had introduced the "Three Fingers Clicking Gesture" to detect a click through a depth camera. To detect an index-click, we first measured the angle between that finger and the hand's palm. Whenever the angle crossed a defined *threshold*, a click was generated. The Three Fingers Clicking gesture was ergonomically poor. It also suffered from some limitations such as users could only use the index finger for clicking. To overcome these limitations, we introduced "Depth Click" [13].

In Depth Click, we first detect the 3D coordinates of the Thumb (T), Index (I), Middle Finger (M), Ring Finger (R), Small Finger (S), as well as the palm center (P). Then we create the vectors $\overrightarrow{PT}$, $\overrightarrow{PI}$, $\overrightarrow{PM}$, $\overrightarrow{PR}$ and $\overrightarrow{PS}$. We also retrieve the normal vector $\overrightarrow{N}$ perpendicular to the palm. To detect a click we first compute the angle α between $\overrightarrow{N}$ and any of the previously mentioned vectors, we then compute the complement of this angle $\theta = 90 - \alpha$. θ is the angle by which the finger is bent, relatively to the palm. It serves as a value that is used for a click detection by comparing it to a *threshold*; a click is detected whenever a finger crosses this threshold. This made it possible to detect clicks using any finger. Furthermore, a click was now separated into "Click Down" and "Click Up". A click was hereby defined as a Click Down followed by a Click Up; this gave it more flexibility.

4. MultiX Click

Even though we were able to detect the click of any of the fingers in Depth Click, the clicking mechanism was limited to an individual generic click detection, without being able to take advantage of the fact that clicking of multiple fingers can be realized. With the ability to perform simultaneous multi-click gestures, the vocabulary of the interaction can be greatly enriched, not just by clicking with multiple fingers, but also by *how* the fingers are being clicked. Since our system can detect the clicking action of multiple fingers simultaneously (Fig. 1), we would like to tap into this potential and use it to augment the gestures vocabulary. That was one condition for being able to detect and use multi-clicks. The other condition is the ability of the system to detect "continuous clicks", that is, detecting a click as long as the fingers are held down. Since our system already splits a click action into "click down" and "click up", we used this advantage to detect multi-clicks whenever a finger is in "click down" mode. This was used in the Midair multi-click keyboard (6.) and the Midair Piano (7.), both of which are applications to our algorithm. The midair multi-click keyboard uses "click down" to detect multi-clicks (just like the midair piano); however, it also uses an extra approach which is using a click down in one hand while using its position to trigger the right modifier key. This is explained in details in 6. 1.

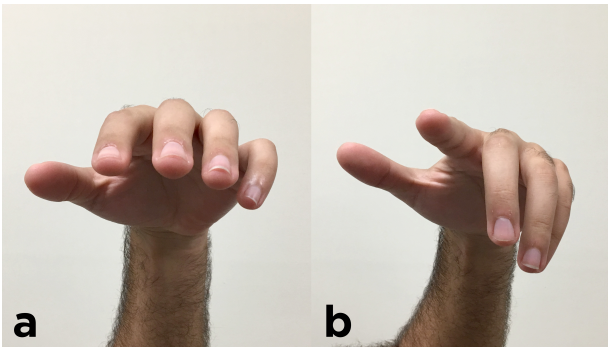

Fig. 1 a shows a hand in a relaxed position (no clicking). b shows a multi-click performed by the middle, ring and small fingers.

5. System overview

5.1 Hardware

To implement our prototype, we used a Leap Motion sensor, which runs at approximately 100 fps (the frame rate varies depending on the available resources). A 23-inch monitor with Full HD resolution was used as the output device. The prototype has been implemented on a computer equipped with an Intel Core i5 3.2 GHz CPU and 4GB of RAM.

5.2 Software

To test our algorithm, we have implemented a midair multi-click keyboard and a midair piano. We explain about them in details in 6. and 7. respectively. We used the Leap Motion SDK to detect the hands, as well as to retrieve the 3D positions of the fingers, palm center, and the normal vector perpendicular to the palm. We used those points to construct the various vectors, angles, and distances used throughout our algorithm. Onscreen rendering was implemented in SFML, and the entire prototype was written in C++.

6. Midair multi-click keyboard

To test MultiX Click, we have implemented a "midair multi-click keyboard", which can be used to input the alphabet characters, some special keys, as well as the "Shift", "Control" and "Alt" modifier keys.

6.1 Triggering the modifier keys

A problem arises when we are mixing both multi-clicks interactions with single-finger clicks interactions; in some situations, multiple functionalities might overlap if the same finger is used to detect a click-down, and later is detected as a regular click once it goes back behind its threshold. While our system supports this kind of gestures, we have designed the single-finger clicks and multi-clicks to be mutually exclusive, and only one of them can be used at a given time. To accomplish this, we have defined a circle-shaped threshold with the sensor as its center, and the multi-clicks detection will only be activated if the users' hands are outside the circle (that is, if the distance from the palm center to the sensor is greater than the circle's radius). In our system, we used a radius of 80 mm; using a bigger value will increase the chance of having the hand outside the detection range of the sensor, and will hence decrease its accuracy. To further increase the vocabulary of the interactions, we used the location of the hand's palm center to determine its relative position to the sensor, in the horizontal plane. To accomplish this, we measure the angle ∠ XOP between vector OP (where O is the sensor's center, and P is the palm center) and the X axis. We have defined eight angle values to determine the position of the hand. Each angle forms an arc of 45 °, and thus these eight angles fit inside a circle. If this angle's value falls within a given arc, it will then correspond to the position related to the arc. This angle is measured when a click is detected outside the circle threshold. Fig. 2 illustrates the arcs and the hand's position depending on the arc, and Algorithm 1 explains how the detection is achieved.

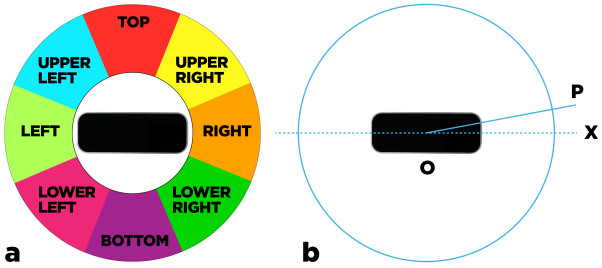

Fig. 2 a shows the distribution of the arcs around the circle. b shows how to find the position by measuring the angle ∠ XOP, as long as the hand is outside the circle. Here, P stands for the palm center (position of the hand), O stands for the origin (position of the sensor), and X corresponds to a point on the X axis, positioned to the right of O

Algorithm 1 Detection of a multi-click

procedure OnClick
  if ($\overline{OP}$ < radius) **then**
    return
  end if
  Determine angle ∠ POX // OX is the horizontal X axis
  Deduce position from ∠ POX
end procedure

6.2 Midair multi-click keyboard implementation

In our research, we suppose that the users will utilize the midair multi-click keyboard in the same way they do a physical QWERTY keyboard. The keys are assigned to fingers in the same way too. For example, the small finger of the left hand can tap the

"Q, A or Z" keys. The possible keys on our keyboard are the twenty-six alphabet keys, in addition to the following four keys: semicolon (;), comma (,), period (.) and forward slash (/). The space key is also usable. To input text using the midair multi-click keyboard, users place their hands about 20cm above the Leap Motion sensor and then perform midair clicks as shown in Fig. 3.

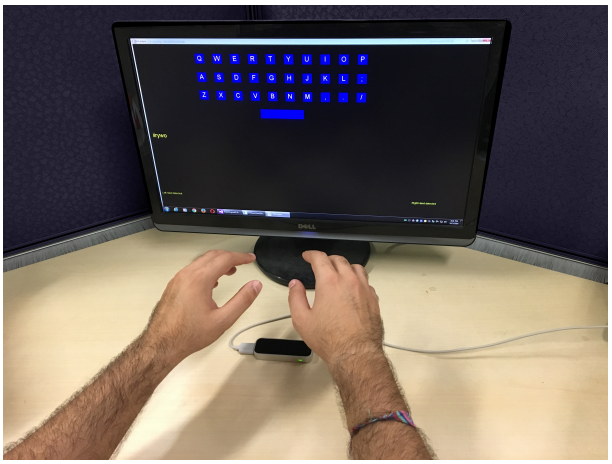

Fig. 3 Midair multi-click keyboard prototype

But the main advantage of our keyboard is that it can detect modifier keys such as "Shift", "Control" or "Alt" in a multi-click fashion: users can click and hold a modifier key with one hand, while inputting regular keys with the other hand, just as it is done when typing on a physical keyboard. To input a "modifier key - regular key" combination, we assume that the hand inputting the modifier key is outside the circle defined in 6. 1. In this position, a thumb-click-down is executed. Upon detection of a click-down gesture, the position of the hand palm is then retrieved, and is used to determine the angle $\angle$ XOP. This angle is then used to determine the position of the click-down regarding the predefined locations in the circle (Fig. 2). The final step would be determining which hand performed the gestures. If the right hand executed the click, then the following algorithm is applied:

- A Shift-click occurs when the thumb is continuously clicking in the right arc (Fig. 2-a)
- A Control-click occurs when the thumb is continuously clicking in the lower right arc (Fig. 2-a)
- An Alt-click occurs when the thumb is continuously clicking in the bottom arc (Fig. 2-a)

Conversely, if the left hand performed the gestures, then the modifier click is detected in the following way:

- A Shift-click occurs when the thumb is continuously clicking in the left arc (Fig. 2-a)
- A Control-click occurs when the thumb is continuously clicking in the lower left arc (Fig. 2-a)
- An Alt-click occurs when the thumb is continuously clicking in the bottom arc (Fig. 2-a)

To detect a regular key tap, we first consider that the hand is inside the circle defined in 6. 1. We then apply the following steps:

- If a thumb click was detected from either hand, a space key tap is generated, regardless of the position of the click.
- If a middle finger, ring finger, or little finger click was detected, we deduce the row of the tap, then generate a click of the corresponding key.
- In the case of an index click, we deduce both the row and the column. We then generate the appropriate key tap.

To detect the row of a given tap, we determine the position of the hand relatively to the Leap Motion sensor, and use it in the following way:

- If the hand is right above the sensor, and inside a given margin, then we assume that the click occurred in the middle row of the keyboard.
- If the hand moves away from the user's body, and beyond the above mentioned middle row margin, we determine that the finger tap occurred in the upper row.
- Conversely, if the hand moves closer to the user's body, and below the middle row margin, then we suppose that the finger tap took place in the lower row.

The index finger can click two columns: a "primary" one (the column closest to the middle finger) and a "secondary" one (the column closest to the thumb). To estimate the column clicked by an index finger, we suppose the users will move their index away from the middle finger and closer to the thumb (for the secondary column) or the other way around (for the primary one column). By detecting this variation, we can approximate the intended column. Knowing the clicking finger, the hand, the row (and column, in the case of an index finger), it is easy to determine the key.

7. Air Piano Application

Our system can detect different forms of simultaneous multiple clicks, such as a click-down in one hand mixed with a regular click in the other (as used in the midair multi-click keyboard). Another possibility is detecting all the click-down gestures that are occurring in one hand. As an application to this situation, we have created an air piano. An air piano is an ideal platform to test our multi-click detection system, especially that when pianists are playing the piano, they hit multiple keys and generate multiple notes at the same time. Our system can detect continuous click-down motions, so we designed the air piano to keep on playing given notes as long as the relative fingers are in click-down position; this replicates sustained notes in a regular piano. Once fingers are not in click-down position (after they perform the click-up gestures), the respective notes stop playing. We have extended this to both hands, so users can play up to ten notes simultaneously.

In our version of the air piano, we have covered two octaves (14 white keys, 10 black keys); each hand would operate an octave. The left hand is assigned to the first octave; to right hand to the second. We first start by explaining the hand movement assumptions that we used to simulate playing the piano. We then explain how each hand operates its assigned octave separately (Fig. 4).

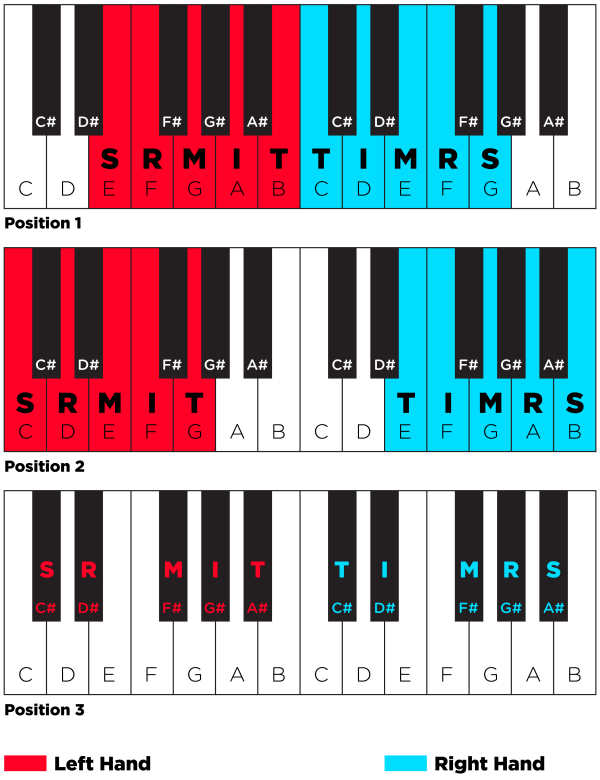

Fig. 4 Keys detection in Air Piano. The letters S, R, M, I and T correspond respectively to the Small, Ring, Middle, Index and Thumb fingers. Position 1 shows how the left and right hands are assigned to the inner white keys. In Position 2, they are assigned to the outer white keys. Position 3 illustrates how the fingers hit the black keys

A piano's keyboard has two rows: the lower row which consists of white keys, and the upper row which consists of black keys. When pianists are playing the piano, we suppose that they move their hands up (away from their torso) to reach the black keys, and down (close to their torso to hit the white keys). To replicate this in our system, we supposed that the Leap Motion sensor sits in the center of our piano version (between the two octaves). We used the same angles model that we have previously utilized in the detection of the Shift, Control and Alt keys: whenever a left hand is in the upper left arc of the circle (Fig. 2 - a), it will be considered that the users are trying to hit the black keys corresponding to that hand. If the right hand is in the upper right arc of the circle (Fig. 2 - a), then we consider that the users want to reach the black keys related to the right hand. The next step is assigning fingers to the keys. In the case of black keys, each octave has five, so the fingers are directly assigned to the keys in the following manner: in the left hand, the small, ring, middle, index and thumb fingers are respectively assigned to the first octave's C#, D#, F#, G# and A# keys. The right hand's assignment's order will be reversed: the second octave's C#, D#, F#, G# and A# keys will respectively be assigned to the thumb, index, middle, ring and small fingers (Fig. 4 - Position 3).

To hit the white keys, we assume that when the left and right hands are respectively in the left and right arcs (Fig. 2 - a), the users want to hit the white keys. Unlike the black keys, which amount to five in each octave and can thus be directly assigned to the five fingers of each hand, there are seven white keys per octave, and hence a different approach is required to hit them. Therefore, we used the same approach that we had previously used in 6. 1 to detect when to trigger multiple selection. We used a radius (Fig. 2 -

b) of 80 mm as a threshold. If the distance from the palm center to the Leap Motion sensor is less than this threshold, then we suppose that the users are trying to hit the inner notes of an octave; otherwise, they will be attempting to hit the outer notes. In the case of the left hand, the small, ring, middle, index and thumb fingers are respectively assigned to:

- E, F, G, A, B notes of the first octave, in the case of the *inner* notes (Fig. 4 - Position 1)
- C, D, E, F, G notes of the first octave, in the case of the *outer* notes (Fig. 4 - Position 2)

As for the right hand, the thumb, index, middle, ring and small fingers are respectively assigned to:

- C, D, E, F, G notes of the second octave, in the case of the *inner* notes (Fig. 4 - Position 1)
- E, F, G, A, B notes of the second octave, in the case of the *outer* notes (Fig. 4 - Position 2)

Our midair piano supports the simultaneous multi-click detection of MutiX Click, and thus allows the users to play multiple notes at the same time. It also produces an audio output of the corresponding played note to simulate a real piano operation. Visual feedback is also given to the users by changing the color of the key that was hit to green. Both the audio and visual feedbacks continue to operate as long as the corresponding key is hit (that is, as long as the assigned finger is in click-down position).

The participants of the experiment also tried the air piano, and we have received multiple comments. Fig. 5 shows the Midair Piano prototype in action.

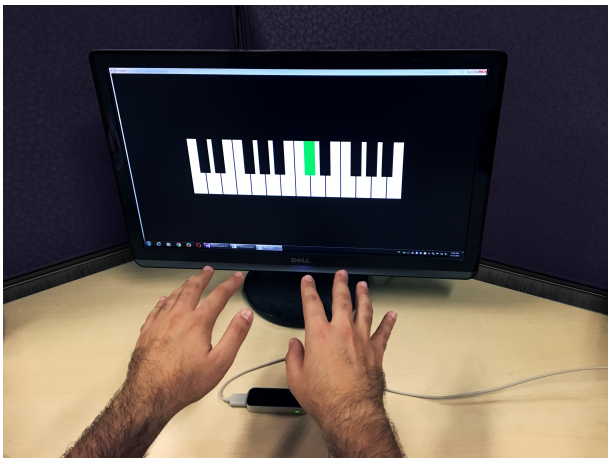

Fig. 5 Midair Piano prototype. The active key's color is changed to green as a visual feedback to the user

8. Evaluation

To evaluate MultiX Click, we conducted three different experiments.

8.1 Experiment 1

In the first experiment, we wanted to evaluate the effect of multi-clicking on the input accuracy. To evaluate our system, 12 participants (all males) aged between 20 and 29 (average 23.75, S.D 2.42) were recruited; 9 amongst them were computer science / engineering students. All were right-handed. A brief introduction was given to the participants on how to carry out the experiment. They were asked to practice using the system for about 10 minutes. To perform the experiment, the participants placed their elbows on the desk, and their hands about 20 cm above the Leap Motion sensor, as can be seen in Fig. 3. In one variation of the experiment, the participants were asked to input all thirty-one keys while using the three modifier keys Shift, Control, and Alt. First, they start by using the right hand to input the modifier keys by using a click down gesture. While keeping the click-down gesture executed, they then, with their left hand, input a key. Each "key - modifier key" combination was entered five times (Shift + Q, Shift + Q, Shift + Q, Shift + Q, Shift + Q, Ctrl + Q, Ctrl + Q, Ctrl + Q, Ctrl + Q, Ctrl + Q, Alt + Q, Alt + Q, Alt + Q, Alt + Q, Alt + Q) They started by inputting the keys corresponding to the upper row of the keyboard (Q, W, E, R, T), then moved on to the middle row (A, S, D, F, G). Finally, they continued with the bottom row (Z, X, C, V, B). Next, they switch hands: the input modifier keys with their left hand, while at the same time inputting regular keys with their right hand. They started with the upper row (Y, U, I, O, P), middle row (H, J, K, L, ;) and bottom row (N, M, comma, period, forward slash). Finally, they entered the space key with their preferred hand, while simultaneously entering the modifier keys with the opposite hand. The other variation of the experiment is similar to the previous one. The only exception is that participants would not enter any modifier key here. The participants were allowed to take a break whenever they felt tired, which was almost once after each row (six times per experiment). To prevent the order effect, the order of the experiment variations was changed for every participant.

8. 2 Questionnaire 1

After completing the experiments, the participants were asked to fill out the following questionnaire, to which they could respond on a five point Likert scale (-2 = Strongly negative answer, 2 = Strongly positive answer):

1. To what extent does multiple selection's usage feel natural?
2. To what extent is multiple selection's usage easy?
3. To what extent did multiple selection operate as you thought?

The participants were also offered the opportunity to enter comments, if any, freely.

8. 3 Results of Experiment 1

Each participant inputted a key 15 times, while at the same time inputting a modifier key with the opposite hand. As a result, each key involved 2 separate inputs, and thus the total inputs per key were 30. This was repeated for all 31 keys of the keyboard, which sums up to 30 * 31 = 960 in the multi-click variation. In the other variation, each participants executed 15 * 31 = 465 inputs, for a total of 1,425 input per participant. All in all, the 12 participants performed 17,100 inputs. We have analyzed the input accuracy for each experiment variation. Fig. 6 and Fig. 7 show the results per modifier key for the regular input variation and modifier key variation respectively.

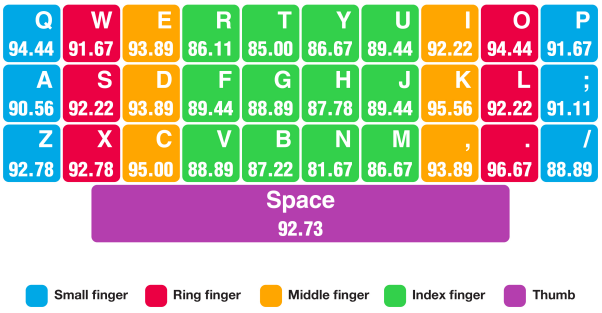

Fig. 6 Success rate (%) by key without using Multi-click

8. 4 Results of Questionnaire 1

The results of the questionnaire are shown in Table 1

The result of Question 1 shows that the participants agreed that the usage of multiple selection is natural, whereas the result of Question 2 indicates that they marginally agreed that it was easy. Finally

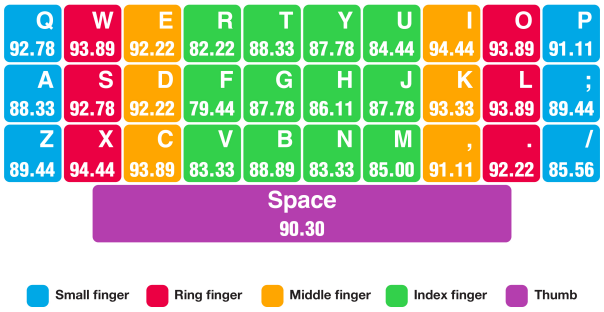

Fig. 7 Success rate (%) by key while using Multi-click

Table 1 Questionnaire 1 results

Question	Mean	S.D
Question 1	0.83	0.72
Question 2	0.58	0.79
Question 3	0.50	0.67

result of Question 3 indicates that the participants marginally agreed that multiple selection operated as they thought.

8. 5 Experiment 2

In this experiment, we wish to evaluate the effect of multi-clicking on the input speed in our air keyboard. 12 participants (all males) aged between 20 and 26 (average 23.42, S.D 1.62) were recruited; 5 amongst them were computer science / engineering students. All were right-handed. A brief introduction was given to the participants on how to carry out the experiment. They were asked to practice using the system for about 10 minutes. The participants were asked to input the following 5 sentences:

1. "1 Year Has 7 Months With 31 Days"
2. "There are 102248 arachnida species on Earth"
3. "HTML5 is better than HTML4"
4. "Mount Tsukuba Is 877 Meters High"
5. "Star Wars Was Released On May 25th 1977"

The participants had to respect the case of the letters as they appeared in the sentences. They were asked to input each sentence in two different ways. In the first one, they were able to input capital letters by clicking down on the Shift key with one hand, and selecting the appropriate key with the other hand. Since our keyboard does not have keys to input numerical values, we mapped the numbers to a combination of Control button and the keys from the middle row (a, s, d, f, g, h, j, k, l, semi-colon). Ctrl + a would generate the number 1, Ctrl + s would

generate the number 2, and so on. Control + semi-colon would generate the number 0. In the second method, we added a "Capital Lock" and a "Number Lock" buttons on the left and right side of the keyboard respectively. Those two buttons can then be activated and deactivated in the same way as the modifier keys:

- The Capital Lock button is activated/deactivated when a thumb click occurs in the left arc (Fig. 2-a)
- The Number Lock button is activated/deactivated if a thumb click is detected in the right arc (Fig. 2-a)

Once the Capital Lock key is activated, the inputted text following that will be in upper case. To return back to lower case, it has to be deactivated. The Number Lock key operates in the same way. As with the previous method, the keys from the middle row (a, s, d, f, g, h, j, k, l, semi-colon) are mapped to the numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 0. To prevent the order effect, the order of the sentences and that of the input method were chosen randomly for each participant.

8. 6 Results of Experiment 2

We have measured the average time in seconds to complete inputting the sentence in both input methods. Furthermore, we analyzed the collected data using a paired t-test ($\alpha = 0.05$). Table 2 shows the results.

Table 2 Results of Experiment 2. Single-click refers to the method where the Capital and Number lock keys were used. Multi-click refers to the input method where the Shift and Control keys were used to input capital letters and numbers respectively. Diff shows the difference of speed between the two methods

Sentence	Single-click	Multi-click	T-test (p)	Diff.
1	52.21s (S.D. 14.22)	38.65s (S.D. 15.10)	0.013	26%
2	43.30s (S.D. 12.06)	43.07s (S.D. 14.74)	0.967	1%
3	31.37s (S.D. 8.50)	33.45s (S.D. 10.06)	0.590	6%
4	44.20s (S.D. 12.49)	34.00s (S.D. 9.84)	0.038	23%
5	57.10s (S.D. 15.61)	43.02s (S.D. 11.16)	0.019	25%

8. 7 Experiment 3

In this experiment, we wanted to evaluate the multi-click detection in Air Piano. For this purpose, 10 participants (all males) aged between 22 and 26 (average 23.5, S.D 1.08) were recruited; 8 amongst them were computer science students. 9 were right-handed. A brief introduction was given to the participants on how to carry out the experiment. They were asked to practice using the system for about 5 minutes. In this experiment, the participants were asked to input 3 different piano chords, 10 times with each hand. To prevent the learning effect, the hand, and the chord that was executed by that hand were selected at random each time. Piano chords involve hitting multiple notes at the same time. The following chords were used:

- C Major, using C, E and G notes
- C sus 2, using C, D and G notes
- C sus 4, using C, F and G notes

As explained in 7., the notes C, D, E, F and G are respectively mapped to the small, ring, middle, index and thumb fingers of the left hand, and to the thumb, index, middle, ring and small fingers of the right hand. Therefore, the fingers used to perform the chords are different for each hand. Table 3 and Table 4 show the fingers used respectively in the left and right hands to execute the chords.

Table 3 Fingers used in the left hand to execute the chords. The corresponding note is shown in parenthesis

Chord	Finger 1	Finger 2	Finger 3
C Major	Small (C)	Middle (E)	Thumb (G)
C sus 2	Small (C)	Ring (D)	Thumb (G)
C sus 4	Small (C)	Index (F)	Thumb (G)

Table 4 Fingers used in the right hand to execute the chords. The corresponding note is shown in parenthesis

Chord	Finger 1	Finger 2	Finger 3
C Major	Thumb (C)	Middle (E)	Small (G)
C sus 2	Thumb (C)	Index (D)	Small (G)
C sus 4	Thumb (C)	Ring (F)	Small (G)

8. 8 Results of Experiment 3

We evaluated the chords detection in Air Piano. Each chord is inputted with specific fingers. For a positive detection, only the required fingers had to be detected. Any other condition (less than the required fingers were detected, more than the required fingers were detected, no detection at all) was considered negative and was rejected. We duly note that all chords executions were detected, and thus the results reflect whether the right or wrong combination of fingers was detected. Table 5 shows the results of the chords detection rate.

Table 5 Results of the average chords detection rate per hand (the standard deviation is shown in parenthesis)

Hand	C Major	C sus 2	C sus 4
Left	44% (3.13)	83% (1.77)	83% (1.70)
Right	34% (4.30)	79% (1.79)	91% (0.88)

8.9 Questionnaire 2

After completing the experiment, the participants were asked to fill out the following questionnaire, to which they could respond on a five point Likert scale (-2 = Strongly negative answer, 2 = Strongly positive answer):

1. To what extent is the C Major chord input easy?
2. To what extent is the C sus 2 chord input easy?
3. To what extent is the C sus 4 chord input easy?

The participants were also offered the opportunity to enter comments, if any, freely.

8.10 Results of Questionnaire 2

The results of the questionnaire are shown in Table 6

Table 6 Questionnaire 2 results

Question	Mean	S.D
Question 1	-0.40	1.43
Question 2	1.50	0.53
Question 3	0.80	0.92

8.11 Air Piano impressions

We have received many comments on the Air Piano application, mostly positive. One participant commented that he learned piano for six years and was very satisfied with the air piano. Another stated that he was able to play a section of the "Do-Re-Mi song" after some practice. Other comments included "the air piano was very natural to use", "the air piano was fun", and "I think people can use the air piano to practice on the train". Others commented that "changing octaves was a bit difficult" and "unintentional keys were detected some times".

9. Discussion

In this section, we discuss the results of the three experiments and the possibilities of our system. We start by explaining the target of the midair multi-click recognition rate. We have defined the following three levels for the recognition rate:

- Level 0: 90% accuracy.
- Level 1: 99% accuracy.
- Level 2: 99.9% accuracy.

We have also defined the following 3 levels for input speed:

- Level 0: 8 strokes per minute - we can input in any way.
- Level 1: 40 strokes per minute - we can input with some stress.
- Level 2: 200 strokes per minute - we can input with no stress.

The target recognition rate and the target speed will depend on applications. For the Midair Keyboard application, our initial goals were Level 1 for accuracy and Level 1 for speed. For the Air Piano application, our initial goals were Level 0 for accuracy and Level 1 for speed. As can be seen from the results of Experiments 1 and 2, the Midair Keyboard goals have almost been satisfied.

Experiment 1 shows that we can type letters with a reasonable recognition rate. If we use multi-selection, we can also type with a reasonable recognition rate, i.e., multi-click does not lower the recognition rate drastically.

Experiment 2 shows that we can type regular English sentences with a certain speed. Comparing single-click and multi-click, multi-click seems to provide us with shorter input time when the words start with a capital letter. On the other hand, when consecutive numbers are appearing, multi-click does not shorten the input time.

Experiment 3 shows the detection and recognition rates of multi-clicks. Previous research [14], [15] shows that human fingers do not move independently, and that motions were produced in other fingers even if the subjects were asked to move just one finger. Due to the constraints of the hand anatomy, some rates dropped. But we can say that the above accuracy goal for Air Piano has been satisfied. Moreover, since we can play the Air Piano in real time, we can say Level 1 speed has also been realized.

By introducing finger clicking, and using different fingers, we can input gesture sequences in a smoother way, which is a great advantage compared to one-shot finger shape gesture. By explicitly detecting finger clicks, we were able to have a great control over the interaction. We were able to precisely define the beginning and the end of gestures by using clicks. Our system successfully detects single clicks, "hybrid

multi-clicks" using both hands (two clicks, one coming from each hand, are detected simultaneously) and "pure multi-clicks" (multiple fingers clicked in one hand). Moreover, in the latter case, the identity of the fingers is also detected and not just the number of multi-clicking fingers (i.e. a multi-click with the index and thumb is different than a multi-click with the index and ring fingers, even though in both cases two fingers are clicked). Identifying the fingers that click in a single hand expands the vocabulary of the interaction in that given hand. Finally, combining single clicks, hybrid multi-clicks and pure multi-clicks, such as mixing pure multi-clicks in both hands, gives rise to multiple possibilities and thus notably increases the overall vocabulary of midair clicks.

10. Other Possible Applications

A potential application for our system is a midair guitar player. Since our system can distinguish multi-clicks this can be used to estimate the chords that a player is trying to play.

Typically, the ability to use multi-clicking enriches the gesture interaction vocabulary. This increase of vocabulary is very flexible too: the combination of multi-clicks can be used to differentiate gestures. For example, simultaneous clicking of different fingers (in the same hand) can generate different gestures, as has been shown in Experiment 3. This can be mixed with regular clicks from the other hand. A possible application to this scenario is a secure input system for passwords or PINs. For example, let's suppose that a user's PIN is 123. Each of the digits 1, 2 and 3 will be inputted using a regular click in one hand, while simultaneously performing different multi-clicks in the other hand. Therefore, the PIN is not just 123 anymore, but a combination of numbers in one hand and multi-clicking with different fingers in the other hand, which makes its input more secure and decreases the chances of being guessed. Another possible application to multi-clicking is 3D multi-touch gestures, the same way multi-touch is used in smartphones or tablets by using multiple fingers. For example, we can click with the index and middle fingers simultaneously, then swipe the hand to trigger a scroll depending on the swipe direction. Moreover, a 2-finger click can be used to generate a right-click action, and a pinch with thumb and index fingers can be used to zoom in or out of a picture.

11. Conclusion and Future Work

In this paper, we described MultiX Click, a new midair multi-click detection algorithm. Our algorithm allows the detection of simultaneous clicks executed using multiple fingers. Moreover, we were able to combine the detection of multi-clicks with regular clicks, and thus increasing the gestures interaction vocabulary. As an application to our technique, we have implemented a midair QWERTY keyboard with Shift, Control and Alt modifier keys. Users can utilize these modifier keys by clicking down on them; they can use a normal click on any other key, as if using a physical keyboard. We have also successfully implemented an air piano as an application to simultaneous multi-clicks where users can play multiple notes together by simultaneously performing midair clicks with multiple fingers. The multi-click usage was found to be natural. In our future work, we would like to explore the possibility of using a combination of heuristics and machine learning to further increase the accuracy of our algorithm.

Reference

[1] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand-gesture applications. *Communications of the ACM*, Vol. 54, No. 2, pp. 60–71, February 2011.

[2] Haruhisa Kato and Hiromasa Yanagihara. Pacman ui: Vision-based finger detection for positioning and clicking manipulations. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '13, pp. 464–467, 2013.

[3] Daniel Vogel and Ravin Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pp. 33–42, 2005.

[4] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. Omnitouch: Wearable multitouch interaction everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pp. 441–450, 2011.

[5] Takashi Nakamura, Shin Takahashi, and Jiro Tanaka. Double-crossing: A new interaction technique for hand gesture interfaces. In *Computer-Human Interaction*, pp. 292–300, 2008.

[6] Jakub Segen and Senthil Kumar. Look ma, no mouse! In *Communications of the ACM*, pp. 102–109, 2000.

[7] Christian von Hardenberg and François Bérard. Bare-hand human-computer interaction. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, pp. 1–8, 2001.

[8] Jinha Lee, Alex Olwal, Hiroshi Ishii, and Cati Boulanger. Spacetop: Integrating 2d and spatial

3d interactions in a see-through desktop environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pp. 189–192, 2013.

[9] Unseok Lee and Jiro Tanaka. Finger identification and hand gesture recognition techniques for natural user interface. In *APCHI '13 Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, pp. 274–279, 2013.

[10] Atsushi Sugiura, Masahiro Toyoura, and Xiaoyang Mao. A natural click interface for ar systems with a single camera. In *Proceedings of Graphics Interface 2014*, pp. 67–75, 2014.

[11] Arun Kulshreshth and Jr. Joseph J. LaViola. Exploring the usefulness of finger-based 3d gesture menu selection. In *CHI '14 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1093–1102, 2014.

[12] Hani Karam and Jiro Tanaka. Two-handed interactive menu: An application of asymmetric bimanual gestures and depth based selection techniques. In *HCII '14 Proceedings of the 16th International Conference, HCI International 2014*, pp. 187–198, 2014.

[13] Hani Karam and Jiro Tanaka. Finger click detection using a depth camera. In *AHFE '15 Proceedings of the 6th International Conference on Applied Human Factors and Ergonomics and the Affiliated Conferences*, pp. 5381–5388, 2015.

[14] Charlotte Häger-Ross and Marc H. Schieber. Quantifying the independence of human finger movements: Comparisons of digits, hands, and movement frequencies. In *The Journal of Neuroscience*, pp. 8542–8550, 2000.

[15] Zong-Ming Li, Shouchen Dun, Daniel A. Harkness, and Teresa L. Brininger. Motion enslaving among multiple fingers of the human hand. In *Motor Control*, pp. 1–15, 2004.

（received Dec. 2, 2016，revised May 3, 2017）

Biography

Hani Karam

Hani Karam is a PhD candidate in the Department of Computer Science at the University of Tsukuba, Japan. His research interests include human-computer interaction and midair finger tap interfaces. He received his B.S. in Computer Science at the Université Libanaise - Faculté des Sciences II, Lebanon in 2002.

Jiro Tanaka （Member）

Jiro Tanaka is a Professor of Graduate School of IPS, Waseda University. His research interests include ubiquitous computing, interactive programming, and computer-human interaction. He received a BSc a MSc from University of Tokyo in 1975 and 1977. He received a PhD in computer science from University of Utah in 1984. He worked at Department of Computer Science, University of Tsukuba as an Associate Professor and a Professor from 1993 to 2016. He is a member of ACM, IEEE and IPSJ.