Unistroke Gestures on Multi-Touch Interaction: Supporting Flexible Touches with Key Stroke Extraction

Yingying Jiang[1] **Feng Tian**[1,3] **Xiaolong Zhang**[2]
[1]Intelligence Engineering Lab, Institute of Software, Chinese Academy of Sciences, Beijing, China
{jyy, tf}@iel.iscas.ac.cn
[2]The Pennsylvania State University
lzhang@ist.psu.edu

Wei Liu[1] **Guozhong Dai**[1,3] **Hongan Wang**[1,3]
[3]State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
water_wei@263.net
{dgz, wha}@iel.iscas.ac.cn

ABSTRACT
Gesture inputs on multi-touch tabletops usually involve multiple fingers (more than two) and casual touchdowns or liftoffs of fingers. This flexibility of touch gestures allows more natural user interaction, but also poses new challenges for accurate recognition of multi-touch gestures. To address these challenges, we propose a new approach to recognize flexible multi-touch stroke gestures on tabletops. Based on a user study on multi-touch unistroke gestures, we develop a gesture recognition method by extracting key strokes embedded in flexible multi-touch input. Our evaluation study result shows that this method can greatly improve the recognition accuracy of flexible multi-touch unistroke gestures on tabletops.

Author Keywords
Multi-touch interaction, flexible gesture inputs, unistroke gesture recognition.

ACM Classification Keywords
H.5.2. [**Information Interfaces And Presentation**]: User Interfaces – *Interaction styles*; I.5.4. [**Pattern Recognition**]: Applications – *Signal processing*.

INTRODUCTION
Multi-touch interaction has become more and more ubiquitous. Devices with multi-touch functions support touch-based inputs, either by fingers or by hands. When interacting with devices with small interaction surfaces (e.g., mobile phones), users largely rely on gestures produced by one or two fingers. When interacting with large display devices (e.g., tabletops), however, users often apply gestures that involve multiple fingers [11] and casual touchdowns or liftoffs of fingers. Devices that can tolerate multiple fingers and casual touchdowns and liftoffs of fingers offer more flexible and natural user interaction. However, this flexibility also leads to the difficulty in accurate gesture recognition.

Currently, many gesture recognition algorithms are based on stroke gestures (e.g., scrolling). Compared with other gestures, such as static hand shape gestures, stroke gestures are produced by the movement of fingers or hands, and can express richer meanings. Thus, they have been widely used in pen- and touch-based computing [2, 3, 10]. However, existing methods on stroke gesture recognition can hardly support flexible multi-touch stroke gestures that consist of multiple strokes by multiple fingers that are on and off the surface frequently.

In this paper, we present a method to recognize flexible multi-touch unistroke gestures. As the basic gesture type, a unistroke gesture only consists of one stroke. In multi-touch interactions, a multi-touch unistroke gesture may have multiple strokes produced by different touch points (e.g., multiple fingers), but can be essentially represented by a key stroke that is extracted from multiple strokes by considering the spatial relationship of these strokes.

Figure 1 shows the idea of key stroke with three gestures to draw a circle: one finger only (the first row), two fingers (the second row), and one finger with casual hand touch on the screen surface (the third row). Here, each row has four figures to show fingers involved (the leftmost), sampled touch points and their links based on temporal adjacency (the second from the left), multi-touch strokes color-coded based on input fingers (the second from right), and extracted key strokes (the rightmost). The gesture by one finger only is a unistroke, so no extraction is necessary. The key strokes in other two multi-touch gestures are extracted by removing temporally overlapping points (see more details in later sections).

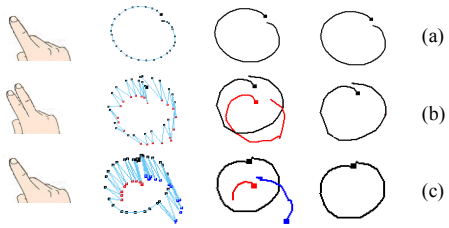

Figure 1. Different drawing styles of a unistroke circle.

IUI'12, February 14–17, 2011, Lisbon, Portugal.

RELATED WORK

Multi-touch gesture recognition has been researched in recent years to extend simple touch gestures. Roudaut et al. [4] developed a method to recognize different thumb gestures (e.g., rolling, sliding). Westerman [8] proposed a recognition approach that can track and distinguish different fingers. Holz et al. [1] explored a method to improve gesture recognition accuracy by using fingerprints to deduce finger posture and user ID. Wilson et al. [9] used the shape of the hand contact area, rather than just the point by finger, in gesture design. Li [3] allowed searching by drawing stroke gestures on touch screen mobile devices. Currently, stroke gesture recognition algorithms are largely about the recognition of unistroke gestures [2, 3, 10].

While these multi-touch gesture recognition and stroke gesture recognition algorithms usually work well, they are weak when processing complex finger gestures that involve multiple fingers and the casual touchdown and liftoff of fingers. Some research [11] has shown that users prefer more flexible gestures that are not limited to one or two fingers and allow more casual movement of hands and fingers.

In summary, existing methods on multi-touch gesture recognition cannot effectively process stroke gestures with flexible touch inputs, which are important to natural user interaction with large-screen devices (e.g., tabletops). Our goal here is to address this problem with a multi-touch unistroke gesture recognition method.

MULTI-TOUCH UNISTROKE GESTURE RECOGNITION

To our knowledge, our work is the first of its kind that explores the methods to recognize multi-touch unistroke gestures with flexible touch inputs. Our work included three components: a user study to establish basic understanding of flexible multi-touch unistroke gestures, the development of multi-touch unistroke gesture recognition algorithms, and the evaluation of our approach.

User Study

Intuitively, although multiple fingers are involved in flexible multi-touch gesture inputs, the movement of these fingers may not be totally independent. Rather, they should move in a coordinated way, as shown in Figure 1b. To know better about the movement of multiple fingers, we conducted a user study to collect data about the finger movement in multi-touch stroke gestures on tabletops.

Subjects, Apparatus, and Procedure

Twelve graduate students (9 male and 3 female) were recruited to perform gestures on a touch table. None of the participants was familiar with tabletop interaction before the study. The touch table used in the study was produced by VR VISION [7], and was based on Laser Light Plane Illumination (LLP) technique.

In the study, a subject was first given five minutes to get familiar with the multi-touch screen by trying a demo provided by Touchlib [6]. Then, the subject was asked to draw five types of gestures: line, circle, triangle, rectangle, and multiline. Each gesture was repeated four more times. The subject was told to choose whatever ways she or he liked. Furthermore, to collect a subject's natural drawing style, we provided no visual feedback of gesture input.

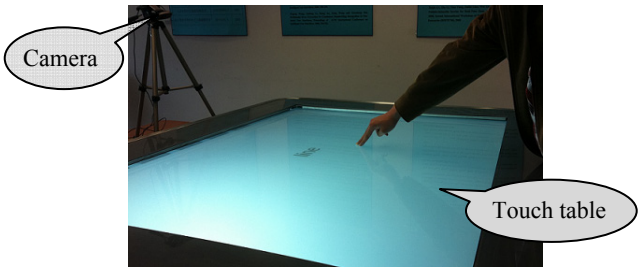

Figure 2. A scene of data collecting.

TouchLib was used to detect finger blobs and store blobs to files. The processes of producing gestures were observed and recorded with an upfront camera, as shown in Figure 2.

Results

By analyzing the collected data, we found that almost one-third (32%) of gestures were drawn by more than one finger. However, the movements of multiple fingers involved in gestures are synchronized: the shapes (strokes) drawn by these synchronized fingers were often similar.

We further measured the similarities of multi-touch strokes by comparing the widths, heights and distances of stroke pairs in the same multi-touch gesture. The mean of the *heightRatio* of stroke pairs (the ratio of heights of bounding boxes of two strokes in a stroke pair) is 1.112 (*SD* = 0.517), the mean of the *widthRatio* (the ratio of the widths of two strokes) is 1.046 (*SD*=0.526), and the mean of *distRatio* (the ratio of the distance between two synchronized points to the average distance of all synchronized points) is 1.000 (*SD* = 0.346). From the results, we can see that during the drawing process, the heights and widths of two synchronized strokes were similar, and the distances between synchronous fingers tended to be stable.

Multi-Touch Unistroke Gesture Recognition

Suppose the sampled touch points be represented as $\{(id_1, x_1, y_1, t_1), \ldots, (id_i, x_i, y_i, t_i), \ldots, (id_N, x_N, y_N, t_N)\}$, id_i indicates the finger index of the ith touch point, (x_i, y_i) is the coordinates, t_i is the time, and N is the number of touch points. The key stroke S consists of a sequence of points and AddPt(x, y) would append a point with the coordinate (x, y) to S.

A conventional recognition algorithm will take all touch points as the key stroke:

$$\text{AddPt}(x_i, y_i), \quad 1 \le i \le N$$

Thus, casual touchdowns or multiple finger inputs may affect the recognition results.

Our approach to recognize a multi-touch unistroke gesture includes three steps (Figure 3). First, touch inputs are sampled. Second, a key stroke is extracted from multi-touch inputs. Third, a unistroke gesture recognizer is adopted to identify the type of a gesture. Conventional gesture recognition approaches mentioned above can be regarded as a special case of our approach, without the step of key stroke extraction.

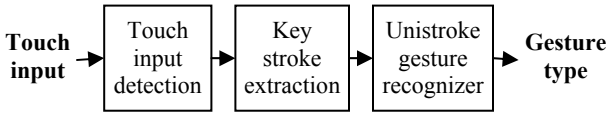

Figure 3. Recognition of multi-touch unistroke gestures.

Key stroke extraction can be done in many ways. In our research, we explored three techniques.

Removing temporally overlapped points: this technique eliminates temporally overlapped, redundant strokes by only keeping the touch point detected earlier when multiple touch points are detected at the same time:

$$\text{AddPt}(x_i, y_i),\ (i=1)\text{or}(1<i\le N,\ t_i != t_{i-1}).$$

Taking the centroid of temporally overlapped points: this method takes the centroid points of temporally overlapped points to compose the key stroke:

$$\text{AddPt}\left(\underset{t_i=t_j, j\in[1,N]}{ave}(x_j), \underset{t_i=t_j, j\in[1,N]}{ave}(y_j)\right),\ 1\le i\le N$$

Averaging adjacent points: this method averages adjacent points to reduce the influence of flexible inputs on the recognition:

$$\begin{cases}\text{AddPt}(x_i, y_i) & (i=1)or(i=N)\\ \text{AddPt}\left(\frac{x_i+x_{i+1}}{2}, \frac{y_i+y_{i+1}}{2}\right) & 1<i<N\end{cases}$$

Evaluation

To understand the effectiveness of our approach, we conducted an experiment. We implemented the three key stroke extraction methods and integrated them with TouchLib [6] (touch point sampling) and $1 recognizer [10] (unistroke recognizing). We also implemented a recognition method without key stroke extraction as the baseline condition. Our study compared these four methods in recognizing flexible multi-touch unistroke gestures.

Subjects, Apparatus, and Procedure

Twelve graduate students were recruited (7 male and 5 female). Five of them had participated in the aforementioned user study. They were all right-handed. We used the same touch table in the user study. Similarly, we provided no visual feedback to subjects' inputs and users were told to perform gestures naturally.

We collected data for 16 unistroke gestures that have been also used in the research to evaluate the $1 recognizer [10]. Figure 4 shows a sample of these 16 gestures, and these gestures are usually used for such activities as executing commands (e.g., deletion), selecting objects, and creating symbols. Each subject had ten trials. In each trial, the subject was asked to finish all 16 gestures one by one. In total, we collected 1920 touch gestures (16 × 10 × 12).

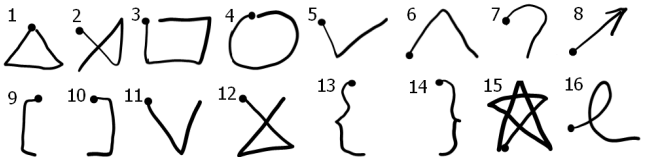

Figure 4. A sample of unistroke gestures used in the study. Dots were added to indicate gesture starting positions.

We then used four methods to recognize gesture inputs. Here, we label four methods as T1 (no key stroke extraction), T2 (removing temporally overlapped points), T3 (taking the centroid of temporally overlapped points), and T4 (averaging adjacent points).

Results

We first examined the error rates when different numbers of templates are used. As shown in Figure 5, the error rates of four techniques (T1-T4) decrease as the training samples increase, and the error rates of T2, T3 and T4 are lower than that of T1 consistently.

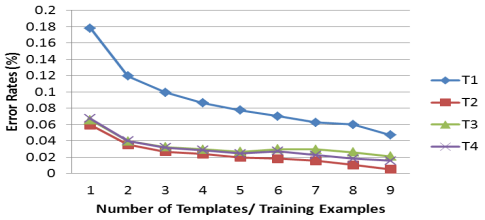

Figure 5. Recognition error rates with training data.

With just one training sample, the error rate of T2, T3 and T4 are 5.94% (*SD*=0.050), 6.58% (*SD*=0.053), and 6.74% (*SD*=0.052) respectively, while the error rate of T1 is 17.81% (*SD*=0.074). One-way ANOVA shows that the performance differences among four methods are statistically significant ($F_{3,60}$=15.387, p<.001). Post-hoc analysis (Tukey HSD) indicates that T2, T3 and T4 are significantly better than T1 (p<.001). One-way ANOVA does not show significant differences among T2, T3, and T4 ($F_{2,45}$=.108, p=.897).

We also examined the error rates of four methods on top N matching (N = 1, 2, and 3). Figure 6 shows the results of the Top-1, Top-2, and Top-3 with one training template. Here, "candidates" refers to possible gesture classes, and "top-n candidates" are the first n gestures returned from recognition algorithms based on their recognition confidence scores. As shown, T2, T3 and T4 outperform T1 in all cases.

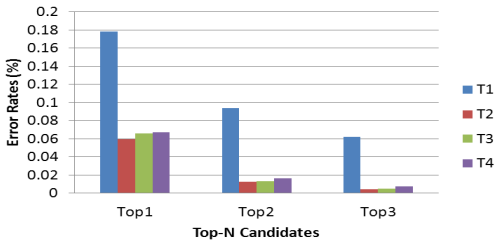

Figure 6. Recognition error rates with top-N candidates.

Figure 7 shows the recognition results of all 16 gestures. As shown, T2, T3 and T4 are better than T1 across all gestures. The gestures with more recognition errors are mainly caused by misclassification of similar gestures, such as gesture No. 5 (the check sign) and gesture No. 11 (the letter v) in Figure 4.

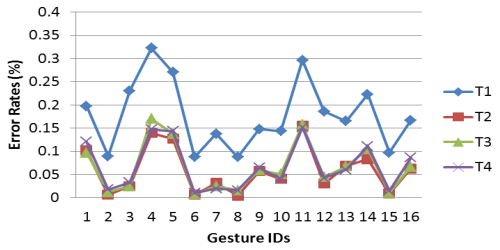

Figure 7. Recognition error rates of different gestures.

As for the time efficiency, when one template is used for each gesture, the average time for T1, T2, T3, T4 are 5.99ms, 5.86ms, 5.89ms, and 5.92ms respectively. When more templates are used, the time needed increases. No significant difference was found among four methods.

DISCUSSION AND CONCLUSION

The results of our evaluation study show that our approaches to recognize multi-touch unistroke gestures are more accurate than the traditional method. This is because key stroke extraction is considered before the application of the stroke recognizer. The three key stroke extraction methods we proposed yield comparable performances, and are all better than the conventional method. Thus, our approach can better support flexible multi-touch unistroke gestures that involve multiple fingers and casual touchdowns or liftoffs of fingers.

It should be pointed out that the recognition accuracies of all four methods used in our experiment are lower than the results reported in other studies, such as [10]. This is largely related to poorer gestures produced by our subjects because of two reasons. First, our data were collected on a tabletop, rather than pen devices in other literature. Tabletop gestures are more flexible than pen gestures. Second, our data collection process provided no visual gesture feedback while other studies usually offered such feedback. Thus, our subjects obtained fewer visual cues in evaluating and correcting their gestures. In this sense, our results are not directly comparable with others.

Accidental and deliberate multiple touches may exist in one system. One way to distinguish them is to include both our algorithm and multi-touch gesture algorithm in the system. Then, those deliberate multi-touch gestures will score higher in the multi-touch algorithm than in our algorithm, while unistroke gestures will score higher in ours than in the multi-touch one. For those ambiguous gestures, we can either use learning algorithms to ask users to specify what result is intended and remember the result, or use other methods, such as [5], to disambiguate different types of gestures automatically.

In summary, we have explored a method to recognize flexible multi-touch stroke gestures. Our studies show that using key stroke extraction can greatly improve the recognition of flexible multi-touch unistroke gestures involving multiple fingers and complex finger behaviors. In the future, we will extend our work by further extending our algorithm for *multi-stroke* gestures, *multi-user* gestures, and small screen devices.

ACKNOWLEDGEMENT

This research is supported by the National Key Basic Research and Development Program of China under Grant No. 2009CB320804 and the National Natural Science Foundation of China under Grant No. 61100151,61170182, and U0735004.

REFERENCES

1. Holz, C. and Baudisch, P. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In CHI 2010, 581-591.
2. Li, Y. Protractor: A fast and accurate gesture recognizer. In CHI 2010, 2169-2172.
3. Li, Y. Gesture Search: A Tool for Fast Mobile Data Access. In UIST 2010, 87-96.
4. Roudaut, A., Lecolinet, E. and Guiard, Y. MicroRolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In CHI 2009, 927-936.
5. Schwarz, J., Mankoff, J. and Hudson, S.E. Monte carlo methods for managing interactive state, action and feedback under uncertainty. In UIST 2011, 235-244.
6. TouchLib. http://www.nuigroup.com/touchlib/
7. VR VISION. http://www.vrvision.cn/index.asp
8. Westerman, W. Hand tracking, finger identification, and chordic manipulation on a multi-touch surface. PhD thesis, University of Delaware, 1999.
9. Wilson, A.D., Izadi, S., Hilliges, O., Garcia-Mendoza, A. and Kirk, D. Bringing physics to the surface. In UIST 2008, 67-76.
10. Wobbrock, J.O., Wilson, A.D. and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In UIST 2007, 159-168.
11. Wobbrock, J.O., Morris, M.R. and Wilson, A.D. User-defined gestures for surface computing. In CHI 2009, 1083-1092.