HAL
archives-ouvertes.fr

Recognize multi-touch gestures by graph modeling and matching

Zhaoxin Chen, Eric Anquetil, Harold Mouchère, Christian Viard-Gaudin

▶ **To cite this version:**

Zhaoxin Chen, Eric Anquetil, Harold Mouchère, Christian Viard-Gaudin. Recognize multi-touch gestures by graph modeling and matching. Céline Rémi; Lionel Prévost; Eric Anquetil. 17th Biennial Conference of the International Graphonomics Society, Jun 2015, Pointe-à-Pitre, Guadeloupe. 2015, Drawing, Handwriting Processing Analysis: New Advances and Challenges. <hal-01165768>

HAL Id: hal-01165768
https://hal.univ-antilles.fr/hal-01165768

Submitted on 20 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Recognize multi-touch gestures by graph modeling and matching

Zhaoxin CHEN [a b], Eric Anquetil [a] , Harold Mouchére [b] and Christian Viard-Gaudin [b]
[a] *INSA de Rennes, Avenue des Buttes de Coësmes, F-35043 Rennes, France*
UMR IRISA, Campus de Beaulieu, F-35042 Rennes, France
[b] *LUNAM / UNIVERSITY OF NANTES / IRCCYN, Rue Christian Pauc, 44000, Nantes, France*
{zhaoxin.chen, eric.anquetil}@irisa.fr, {harold.mouchere, christian.viard-gaudin}@univ-nantes.fr

Abstract. Extract the features for a multi-touch gesture is difficult due to the complex temporal and motion relations between multiple trajectories. In this paper we present a new generic graph model to quantify the shape, temporal and motion information from multi-touch gesture. To make a comparison between graph, we also propose a specific graph matching method based on graph edit distance. Results prove that our graph model can be fruitfully used for multi-touch gesture pattern recognition purpose with the classifier of graph embedding and SVM.

1. Introduction

Due to the recent prevalence of multi-touch devices, multi-touch gesture recognition has gained a large interest in the last decade. It is possible to consider two different application frameworks for multi-touch gestures: gestures as direct manipulation or indirect command. The former study is usually based on low-level touch events (e.g., touch-down, touch-move, touch-up) and has been extensively applied in image or map view application (e.g., pinch or spread two fingers to scale an image). The latter is much similar to the character recognition as the recognition procedure is conducted after the entire gesture being performed and triggers a corresponding operation (e.g., copy, paste, etc.). To the best our knowledge, most research in the latter are based on mono-touch gesture. Our work focuses on the difficulties on multi-touch gesture recognition and tries to make it possible for indirect command.

Unlike mono-touch gesture recognition which tracks the movement of a single point of input, multi-touch gesture often tracks many points of contact in parallel as they appear, move and disappear. The recognition for multi-touch gestures is challenging because of the complex chronological relation between the fingers' trajectories. Consider a two-stroke gesture under a multi-touch input context. Although the shape of the strokes is fixed, the writing order could vary from in-sequence to full-synchronizing as explained in Figure 1. In the work (Hao Lu & Yang Li, 2012), the authors present a GestureCode system recognizing multi-touch gestures with state machines. They achieved around 75% accuracy on a 15-gesture database. In (Kenrick Kin et al., 2012), the Proton++ describes the multi-touch gestures as regular expressions of touch event symbols. GeForMt (Kammer et al., 2010) uses context-free grammars to model multi-touch gestures.

In our prior work (Zhaoxin et al., 2014), we use graph to characterizes the stroke shape information on nodes and stroke spatial and temporal information on edges. In most of the cases, this approach could well recognize the multi-touch gestures which are alike in shape but different in stroke order. However, it only uses some discrete attributes as many previous works (Kenrick Kin et al., 2012) to indicate the state or shape of the strokes. In this paper, we propose an improvement of our prior work that quantifies these relations with numeric features to characterize the stroke shapes, spatial relationships, motion relationships and temporal relationships in the gesture.

2. Represent a multi-touch gesture by graph

Extracting global features has been proven to be efficient for handwriting character, symbol, icon or gesture recognition where the discrimination is mainly made according to the shape of the sample (Don Willems et al., 2009). For the multi-touch gesture case, since each stroke of gesture is no longer only written in sequence, the temporal relation as well as the relative motion relation of strokes should also be characterized. Therefore, in our graph model, we propose to use the node to describe the shape of each stroke and edge to indicate their temporal and motion relation. In the following paragraphs, the graph will be denoted by $g = (V, E)$, where V denotes a finite set of nodes, E denotes a set of edges.

2.1 Preprocessing and sub-stroke representation

In general, an on-line handwriting data is a set of consecutive points which records the coordinate (x, y), time (t) for each input event. It is unwise to use raw coordinate point as node because the complexity of graph matching in the following procedure is exponential in the number of nodes. We consider to segment each raw strokes S_i into a sequence of straight line sub-strokes $(s_1, \dots, s_n)$ using polygonal approximation. The Ramer-Douglas-Peucker algorithm (Urs Ramer, 1972) is used to find a polygonal chain with few segments that serves as an accurate approximation of the stroke. Each sub-stroke s_i is then represented by a node u_i in the graph with an attribute of 4 components, $u_i = (p_i, a_i, l_i, t_i)$, where p_i is the position (unit square bounding box coordinates) of the center of the sub-stroke, a_i is its angle relative to the horizontal, l_i length of the sub-stroke, t_i contains start

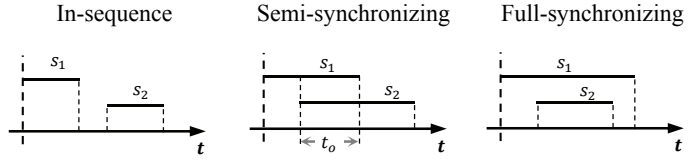
and end times. Each feature is normalized to [0,1]. Thus, assume a multi-touch gesture with n strokes, the node set V is represented as

$$V = \{U_1, \dots, U_n\} = \{(u_1^1, \dots, u_{m_1}^1), \dots, (u_1^n, \dots, u_{m_n}^n)\}, \quad [1]$$

where U_i is the subset which contains the nodes belonging to the same stroke S_i. u_i^j is the node which represents the ith sub-stroke of the jth stroke. Once the node set V is obtained, a link edge e^l is built from u_i^j to u_{i+1}^j which indicates the written sequence of a stroke. Since the direction attribute has been contained in node (as angle), the edge only indicates the concatenating relation between nodes.

2.2 Representation for temporal and motion relation between strokes

Differ from the mono-touch handwritten gesture, the multiple strokes in a multi-touch gesture are not performed stroke after stroke. The relative temporal relation between each two strokes could be in-sequence, semi-synchronizing or full-synchronizing. To quantify these relations, we define a fuzzy relative temporal relation function $f_t(s_1, s_2)$ as expressed by equation [2] and illustrated in figure 1,

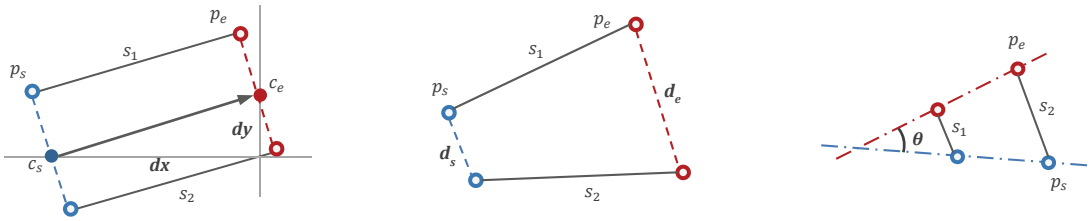


$$f_t(s_1, s_2) = \begin{cases} 0 & \text{in-seq} \\ \min(1, \frac{t_o}{\min(t_1, t_2)}) & \text{semi-sync} \\ 1 & \text{full-sync} \end{cases} \quad [2]$$

Figure 1. Relative temporal relation between two strokes

where t_o is the time of overlap detected in the semi-synchronizing case, t_1 and t_2 are the time duration for the two strokes. In our graph, this relation function is implemented on each pair of nodes (sub-strokes) which do not belong to same stroke. When value of the relation function is larger than a threshold as $f_t(s_1, s_2) > \lambda$, a synchronization edge e^s is built between the corresponding two nodes.

We now proceed to the modeling of motion information. The synchronization relation obtained above indicates the existence of a relative movement between sub-strokes. This relative movement is the key feature of a multi-touch gesture against a mono-touch gesture. Plenty of previous works (Kenrick Kin et al., 2012) made effort to extract the motion information from the movement. Inspired from the work by (Chi-Min Oh et al., 2011), we measure three kinds of motion from the movement: translation, rotation and scaling. Figure 2 illustrates the computation of the three types of motion between sub-strokes.



(a) Translation motion $m_t(s_1, s_2) = \sqrt{dx^2 + dy^2}$

(b) Scaling motion $m_s(s_1, s_2) = d_e - d_s$

(c) Rotation motion $m_r(s_1, s_2) = \theta / 2\pi$

Figure 2. Three types of motion between sub-strokes, p_s and p_e represent the start points pair and end points pair, respectively. Since all the gestures are normalized inside a unit square bounding box, these three values have the same range in [0,1].

Note that it is unnecessary to compute these motion features between all two sub-strokes. The relative movement is meaningful only if two sub-strokes are written synchronously. In other words, these motion features are computed when a synchronization edge e^s is existed between two nodes. Therefore, we assign these three values as the attributes of synchronization edge, $e^s = (m_t, m_s, m_r)$.

On the other hand, in order to capture the sequential relation between the sequential written strokes, we introduce a sequence edge e^q that directs from the last sub-stroke of previse stroke to the first sub-stroke of the following stroke in our graph. This edge has an attribute t_d as $e^q = (t_d)$, which indicates the time delay between sequential written strokes.

The example in figure 3 considers a bimanual multi-touch gesture performed by a *holding* with left hand and writing 'z' character with right hand synchronously. The preprocessing step firstly resamples the gesture to 4 sub-strokes $(s_1, \dots, s_4)$. Then, we convert each sub-strokes into node in the graph with its shape and duration information as its attribute, i.e. $u_i = (p_i, a_i, l_i, t_i)$. Finally, two link edges e^l and three synchronization edges e^s (since the two strokes are performed synchronously, no sequence edge is built for this gesture) are built between nodes.

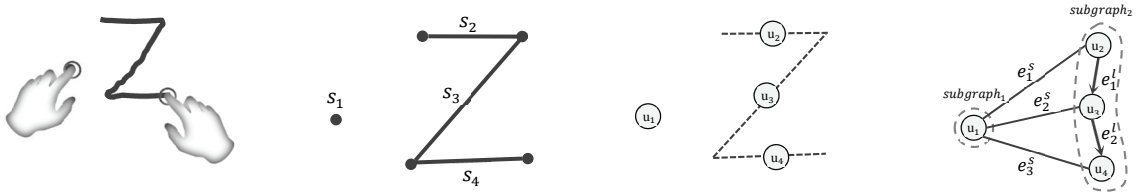


Figure 3. Illustration of the graph representation. (a) Raw input multi-touch gesture. (b) Resampling the raw data to 4 sub-strokes using polygonal approximation. (c) Each sub-stroke s_i is represented by a node u_i with its features (p_i, a_i, l_i, t_i) as attribute. (d) Link edge e^l and synchronization edge e^s are built between nodes.

2.3 Graph matching algorithm

We adopt graph edit distance to compare the dissimilarity between two graphs. Since each graph has multiple sub-graphs, the first step is to find which sub-graph should be compare to which one from one graph to the other. This is a typical assignment problem that can be solved by Munkres' algorithm. The distance of each sub-graph pair is computed by DTW algorithm based on the sequence of nodes $u_i = (p_i, a_i, l_i)$. Optimal matching is then found by Munkres' algorithm that total cost of the sub-graph assignment is minimized. Once we have the nodes' alignment path achieved by DTW, the edit operations on nodes and edges could be inferred from the nodes' DTW path. Consequently, the edit distance of two graph is defined as:

$$d(G_1, G_2) = \sum_{i=1}^{n} c(u_i, u_{\varphi(i)}) + \sum_{i=1}^{m} c(e_i, e_{\varphi(i)})$$

where $c(u_i, u_{\varphi(i)})$ denotes the edit cost for nodes pair, $c(e_i, e_{\varphi(i)})$ is the edit cost for edges pair, φ denotes the assignment function which maps the nodes and edges of the first graph G_1 to the second graph G_2.

3. Experiments

We describe below a series of experiments to evaluate our graph modeling and graph matching algorithm. The recognition system and database we use will be firstly introduced in this section.

Since we use the graph edit distance to measure the dissimilarity between graphs, a simplest recognition method is the k-Nearest Neighbors (kNN) which classifies an object by a majority vote of its neighbors. Meanwhile, we also implement a graph embedding algorithm introduced in (Kaspar Riesen et al., 2010) to represent the graphs in vector space. The basic idea of this graph embedding algorithm is to represent a graph with a number of dissimilarities to a set of prototypical graphs. Prototypes are iteratively selected from training set under a prototype selection principles named *Targetsphoere* (also introduced in (Kaspar Riesen et al., 2010)). A SVM classifier is then trained for the feature vector classification.

We designed a specific gesture database of 4346 samples collected from 21 participants. Based on these gestures, we defined a rotation dependent set (Set45) with 45 different classes and a rotation independent set (Set31) with 31 different classes (Figure 3). Both gesture sets have same content but different labels on some special cases. These gestures are distinguished from shape, temporal and motion information and can be generally separated to two categories. Category A contains bimanual or unimanual multi-touch gestures in which the trajectories are synchronously performed. Gestures in this category vary in shape, numbers of touch finger and motion. To validate the modeling of temporal information, category B contains 4 kinds of multi-stroke gestures which have same shape as some cases in category A but vary in written order.

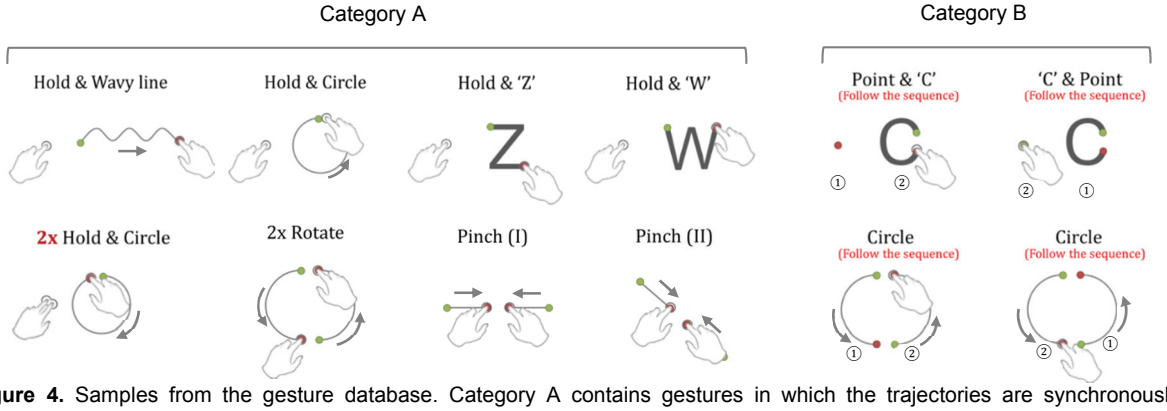
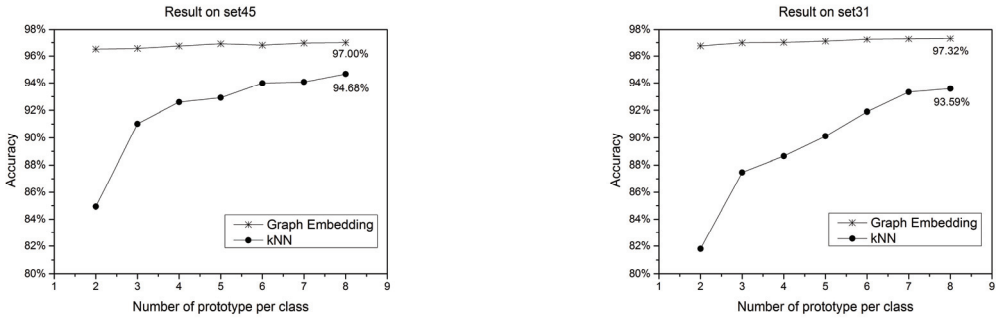


Figure 4. Samples from the gesture database. Category A contains gestures in which the trajectories are synchronously performed. Note that the last two gestures Pinch (I) and Pinch (II) are allocated to different classes in rotation dependent set (Set45) but integrated in the same class in rotation independent set (Set31). Category B consists of two sequential mono-touch gestures which are distinguished from written order.

We conducted our experiments in a 5-fold cross validation and writer independent scheme. In each fold, samples from 6 writers are used for training and 15 writers for test.

4. Results

We compare the recognition accuracy between kNN and graph embedding regarding different numbers of prototypes. Results are given in figure 5.



(a) (b)

Figure 5. Comparisons of the two recognition systems on both datasets. Accuracies are given as a function of the number of prototypes per class.

The experiments on set45 in Figure 5(a) show that with the increasing of the prototype number, the accuracy of graph embedding system keeps stable around 97% and performs significantly better than kNN system, which increases from 84.93% to 94.68%. Note that even with a limited number of prototypes, the graph embedding with SVM classifier could still obtain a good performance while kNN is not efficient enough. Same trend is also viewed in the result on set31. The graph embedding system performs as good as in set45 while accuracy of kNN system has a slightly decreasing from 94.68% to 93.59%. Results prove that our graph modeling , matching and embedding framework is powerful to extract the discrimination information for multi-touch gesture pattern recognition.

5. Conclusion

In this paper, we propose a new method for modeling the shape, relative temporal and motion information in multi-touch gesture by a model of graph. We extract these information and quantify them as attribute on the nodes and edges in the graph. A way to make a comparison between graphs is also offered that can be used for graph classification. Results show that with the help of graph embedding and SVM classifier, our graph model has the ability to effectively distinguish different multi-touch gestures. Base on this model, in our future work we aim at developing a strategy to detect the pattern of multi-touch gesture at runtime as in (Kenrick Kin, 2012).

References

Hao Lü, Yang Li (2012). Gesture Coder: A tool for programming multi-touch gestures by demonstration. Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). ACM, New York, NY, USA, 2875-2884.

Kenrick Kin, Björn Hartmann, Tony DeRose, & Maneesh Agrawala. (2012). Proton: multitouch gestures as regular expressions. Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). ACM, New York, NY, USA, 2885-2894.

Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. (2012). Proton++: a customizable declarative multitouch framework. Proc. of the 25th annual ACM symposium on User interface software and technology (UIST '12). ACM, New York, NY, USA, 477-486.

Kammer, D., Wojdziak, J., Keck, M., and Taranko, S. (2010) Towards a formalization of multi-touch gestures. Proc. ITS 2010 (2010), 49–58.

Zhaoxin Chen, Eric Anquetil, Harold Mouchère, Christian Viard-Gaudin. (2014). A graph modeling strategy for multi-touch gesture recognition. In 14th International Conference on Frontiers in Handwriting Recognition, Pages 259-264, 2014.

Don Willems, Ralph Niels, Marcel van Gerven, and Louis Vuurpijl. (2009). Iconic and multi-stroke gesture recognition. Pattern Recogn. 42, 12 (December 2009), 3303-3312.

Chi-Min Oh, Md. Zahidul Islam, and Chil-Woo Lee. (2011). MRF-based Particle Filters for Multi-touch Tracking and Gesture Likelihoods. Proc. of the 2011 IEEE 11th International Conference on Computer and Information Technology (CIT '11). IEEE Computer Society, Washington, DC, USA, 144-149.

Kaspar Riesen and Horst Bunke. (2010). Graph Classification and Clustering Based on Vector Space Embedding. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Kaspar Riesen and Horst Bunke. (2009). Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision computing 27(4), 950-959.

Urs Ramer. (1972). An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing, 1(3), 244–256