A graph modeling strategy for multi-touch gesture recognition

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

A graph modeling strategy for multi-touch gesture recognition

Zhaoxin Chen and Eric Anquetil
INSA de Rennes/IRISA, France
zhaoxin.chen@irisa.fr
eric.anquetil@irisa.fr

Harold Mouchère and Christian Viard-Gaudin
IRCCyN/IVC - UMR CNRS 6597
Ecole Polytechnique de l'Université de Nantes, France
harold.mouchere@univ-nantes.fr
christian.viard-gaudin@univ-nantes.fr

***Abstract*—In most applications of touch based human computer interaction, multi-touch gestures are used for directly manipulating the interface such as scaling, panning, etc. In this paper, we propose using multi-touch gesture as indirect command, such as redo, undo, erase, etc., for the operating system. The proposed recognition system is guided by temporal, spatial and shape information. This is achieved using a graph embedding approach where all previous information are used. We evaluated our multi-touch recognition system on a set of 18 different multi-touch gestures. With this graph embedding method and a SVM classifier, we achieve 94.50% recognition rate. We believe that our research points out a possibility of integrating together raw ink, direct manipulation and indirect command in many gesture-based complex application such as a sketch drawing application.**

Keywords— Multi-touch, gesture recognition, graph embedding

I. INTRODUCTION

With the development of touchscreen technology, touch-based interaction gains a lot of popularity in the recent decade. A highly sensitive touchscreen with its underlying recognition system allows many different gesture commands such as flicking, pinching, scrolling, etc., which gives users a more convenient way to interact with computing devices.

Basically, there are three kinds of touch-based interaction, 1) Raw ink, which records the raw freehand sketching data, e.g. a handwriting notepad or a paint application [1]. 2) Direct manipulation, which manipulates the interface when a real-time feedback is needed from the system to the user such as scaling, panning, etc. [2, 3]. 3) Indirect command, where an input gesture is treated as a shortcut for triggering a corresponding action [4-6]. Prior works show that multi-touch gestures are mostly employed for direct manipulation while indirect command studies focus more on single-finger gestures.

In this paper, we present a novel multi-touch gesture recognition strategy for indirect command issue, where the recognition procedure is launched after a multi-touch gesture being performed. The shape and relationship between strokes may have a certain meaning that can be associated with a certain action. It allows the users to perform a multi-touch gesture as a shortcut for a command rather than seeking an explicit button or a keyboard shortcut. This idea of using gesture as command is not quite a new one, one of our recent application *Varchitect* (Found in Windows Store) enables the user to efficiently draw a pre-defined object with a single stroke gesture (Fig. 1). But to the best our knowledge, few studies address the multi-touch gesture as indirect command issue. The motivation of our work is to extend the variety of command gestures by employing multi-touch gesture.

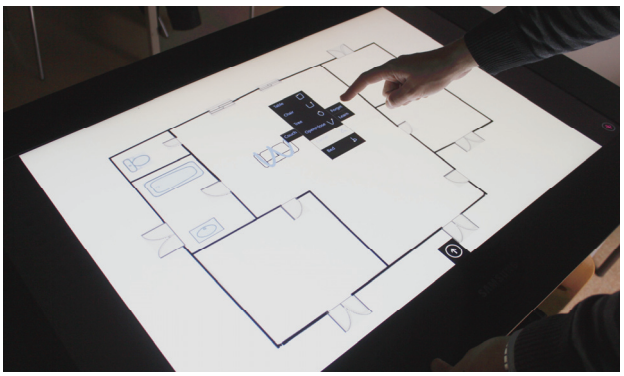


Fig. 1. User may perform a gesture to efficiently draw a pre-define object in our *Varchitect* application.

At this stage of our research, we focus on the problem to recognize a multi-touch gesture from its temporal, spatial and shape information. We begin with a discussion of previous research works related to the problem of gesture recognition (Section 2). We then present the approach we use to model a gesture by graph. The definition of the multi-touch gesture model by a graph is the main contribution of this paper. This approach is enriched by the presentation of a graph embedding strategy that results in an explicit embedding of graphs from graph domain in a real vector space (Section 3). In Section 4, the recognition performance of our strategy is explored by using LibSVM [9] classifier on a dataset containing a total of 1800 samples of 18 different multi-touch gestures. The associated discussions illustrate how we want to implement this approach on a sketch drawing application.

II. RELATED WORK

To begin with, it is important first to distinguish between direct manipulation recognition and indirect command recognition. The former should measure the spatial relationship between fingers and give a continuous feedback during fingers moving. For example, a scaling action requires the system continuously changing the target size in accordance with the fingers gathering or spreading. As for the latter, the motion's global features, such as the shape of trajectory becomes more useful. The recognition is done after gesture ending and triggers a discrete action for the system.

A large number of works concern gestures that directly manipulate the interface. Oh et al. [3] calculated motion likelihoods as feature during fingers' moving. Three basic motions, translation, scaling and rotation, can be recognized if their likelihood values exceeds a certain threshold. Lu and Li [2] proposed a Gesture Code approach where a finger's event sequence is used to learn a state machine for recognition. But learning based on individual motion segment does not allow the capture of a motion's global features. A very similar method from Kin et al. [10] specified a sequence of touch events that comprises a gesture as a regular expression. These methods commonly extract the dynamic features for recognition.

Concerning indirect commands, Madhvanath et al. [5] presented their GeCCo which uses finger gestures as command and control. Dynamic Time Warping (DTW) with k nearest neighbor (KNN) is employed for recognition, but only single-finger gesture is supported. Appert and Zhai [6] also investigated using stroke gestures as shortcut. They concluded that stroke shortcuts can be as effective as keyboard shortcut in actual performance but have cognitive advantages in learning and recall. Bailly et al. [4] introduced a Finger-Count interaction where the system just need to count the number of finger contacts. Twenty five different input configurations can be expressed when using both hand.

It is worth to note that most of the studies on multi-touch gesture are limited in a small set of commands. Few studies focus on recognizing multi-touch gesture as indirect commands especially when we need to deal with a large number of different gestures.

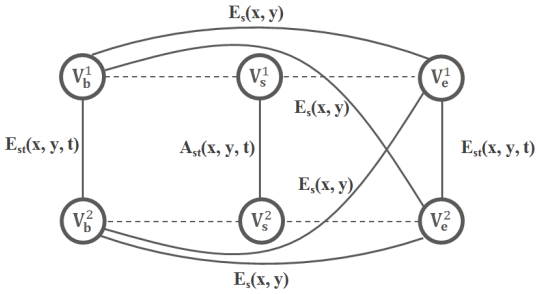
III. GRAPH MODELING AND EMBEDDING

We introduce here our new recognition system for multi-touch gestures. We will first discuss the graph modeling approach where the strokes and their spatial and temporal relations are represented by a set of vertices and edges in the graph. We then introduce a graph embedding strategy to encode the graph into a fixed size vector to make the pattern recognition task easier.

A. Graph modeling

A multi-touch gesture modeling should consider three kinds of information: spatial, temporal and shape information. A spatial information shows the relative position of each single stroke with reference to the others inside the gesture, while a temporal information illustrates the written order between the different strokes and the duration of each one. The last information should retain knowledge about the intrinsic shape of the stroke, allowing to distinguish between a simple straight line from a more complex curve.

In a first step, we consider representing each stroke by three vertices in the graph. Consider a two stroke gesture as an example (Fig. 2), each stroke is depicted by a begin vertex (V_b), a stroke vertex (V_s) and an end vertex (V_e). The spatial and temporal relationships between vertices are described by the edges $E_s(x, y)$, $E_{st}(x, y, t)$ and $A_{st}(x, y, t)$. The subscripts s and st indicate that only spatial or both spatial and temporal relationships are measured between the vertices, respectively. The differences between E_{st} and A_{st} will be explained in the following paragraph.



$\mathbf{V_b}$: Begin vertex — $\mathbf{E_s}$: Spatial relationship edge
$\mathbf{V_s}$: Stroke vertex — $\mathbf{E_{st}}$: Spatial and temporal relationship edge
$\mathbf{V_e}$: End vertex — $\mathbf{A_{st}}$: Spatial and temporal relationship edge with Allen's labels

Fig. 2. An example of a general graph modeling for a two strokes gesture.

To complete the preceding representation, and make it specific for a given spatial-temporal configuration, the edges are marked with one or several discrete labels. We make use of the Allen's relations [11] which originally characterized the inferences about time by discrete labels. Table I shows the illustration and examples of Allen's relations.

TABLE I. EXAMPLES OF ALLEN'S RELATIONS

Relation	Example	Relation	Example
Before (B)	Str_1: —— Str_2: ——	During (D)	Str_1: —— Str_2: ——
Equal (E)	Str_1: —— Str_2: ——	Start (S)	Str_1: —— Str_2: ——
Meet (M)	Str_1: —— Str_2: ——	Finish (F)	Str_1: —— Str_2: ——
Overlap (O)	Str_1: —— Str_2: ——		

In our approach, we extend the applicability of these relations on characterizing also the spatial relationships. There are two general cases for labeling the edges.

1) Edges between stroke vertices (A_{st}): The Allen's relations are used to measure the relationships with respect to time, x-axis position and y-axis position. Consider the *flick* gesture in Fig. 3(a), Fig. 3(b) shows the spatial relationships between strokes with respect to the x and y axis. According to the Allen's relations, the label *Equal_X (E_X)* and *Before_Y (B_Y)* are associated to the edge A_{st}. Fig 3(c) depicts that the two strokes of *flick* were performed simultaneously. Therefore an *Equal_Time (E_T)* label is assigned to the edge A_{st}. Finally the edge A_{st} will be:

$$A_{st}(x, y, t) = \{E\_X, B\_Y, E\_T\} .$$

2) Edges between extremity vertices (E_s and E_{st}): Since the extremity vertices represent each of the finger-down and finger-up positions, only *Equal* property for the time (*E_T*), x-

axis position (*E_X*) and y-axis position (*E_Y*) is used for characterizing the edges between vertices. Fig. 3(d) shows the graph of a *flick* gesture with all the edges labeled. The label E_{st}={E_X, E_T} between V_b^1 and V_b^2 indicates the two starting points are written in the same region on x-axis and in the same time. Same property will also be found between the two V_e vertices. The label E_{st}={E_Y} between V_b^1 and V_e^1 means that they are written in the same region on y-axis. Noted that not all edges between extremity vertices have to be generated if the *Equal* property is not satisfied. Comparing to the general case in Fig. 2, the edges between V_b^1 and V_e^2, V_b^2 and V_e^1 in Fig. 3(b) are removed since these two points are not located in the same position on neither x-axis nor y-axis.

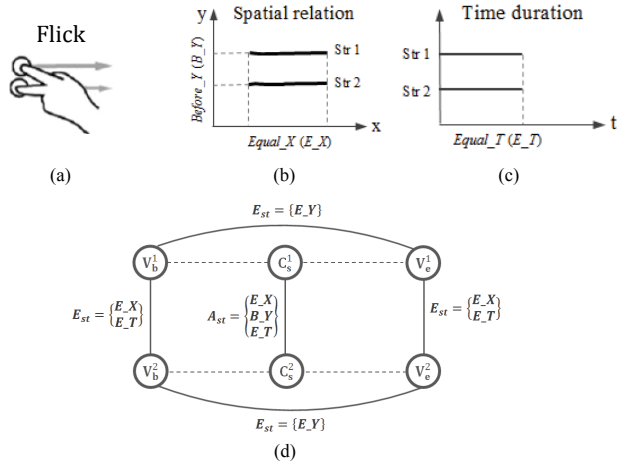


Fig. 3. a) A *flick* gesture. b) Spatial relationship between strokes. c) Temporal relationship between strokes. d) Graph model with labels.

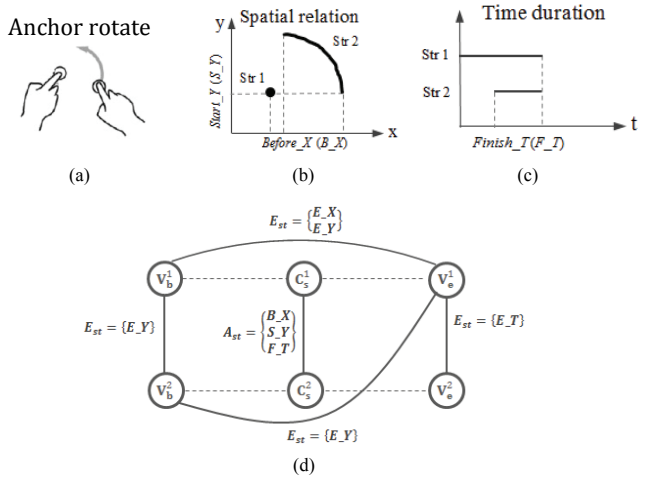


Fig. 4. a) An *anchor rotate* gesture. b) Spatial relationship between strokes. c) Temporal relationship between strokes. d) Graph model with labels.

Fig. 4 gives another example of an *anchor rotate* gesture. Comparing to the *flick* gesture, since they have different spatial relationships, a graph with different structure and labels is shown in Fig. 4(d). It proves that our graph modeling strategy has the ability to characterize different multi-tocuh gesture with different graph models. These differences could be used by a classifier to make a recognition.

B. Stroke shape quantization

In this step we intend to integrate the shape information into our graph strategy in order to differentiate the gestures which have same graph but different stroke shape. To describe the shape of the gesture, we will use the feature set HBF49 proposed in [8]. The shape feature will be studied from two perspectives:

*1) Local Shape Feature (**LSF**):* A multi-touch gesture may be composed by various shapes of strokes. We describe the shape of each stroke by a feature vector from HBF49. Each stroke will then be represented by a label extracted from a codebook previously built with a clustering algorithm (K-means). The corresponding label will be assigned to the stroke vertex of that stroke in the graph. Shown in Fig. 3(d), the labels of vertices V_s have been replaced by the labels of the cluster, the strokes belong to.

*2) Global Shape Feature (**GSF**):* We also extract the shape features for the entire gesture. However, this HBF49 feature vector will not be integrated into the graph but concatenated after the graph embedding feature vector. This GSF can be regarded as a complement for our graph embedding strategy. More details will be explained in the experimental section.

C. Graph embedding

A graph embedding method aims to transform the graph structure into a feature vector for the benefit of using statistical classification method. In this paper, we adopt a graph embedding approach introduced by Sidere et al. [12]. The basic idea of this approach is to build a matrix where each row is relative to a label of vertices or edges while each column corresponds to a sub structures P_j of the graph. The value of the matrix at $[L_i, P_j]$ is the number of occurrences of the label L_i in each sub graph P_k which is isomorphic to the sub structures P_j. The construction of the vectorial representation can then be performed by transforming the matrix into vector feature space.

In our case, we empirically choose three sub structures which are one vertex, two vertices with one edge and three vertices with three edges for the column of the matrix. The row involves the seven Allen's relations for three aspects (time, x-axis and y-axis), cluster number (typically 9) and labels of begin and end for the vertex. On the whole, 32 labels will be related to the rows. Accordingly, a feature vector with a length of 96 will be generated after the graph embedding.

IV. EXPERIMENTS

We conducted experimental evaluation of our proposed recognition method over a multi-touch gesture dataset which will be depicted in the following subsection. We make use of LIBSVM for the classifier to test the recognition rate by using our graph embedding features. The result is then compared with our previous work, HBF49 features, which is used as a baseline method for evaluating symbol recognition system.

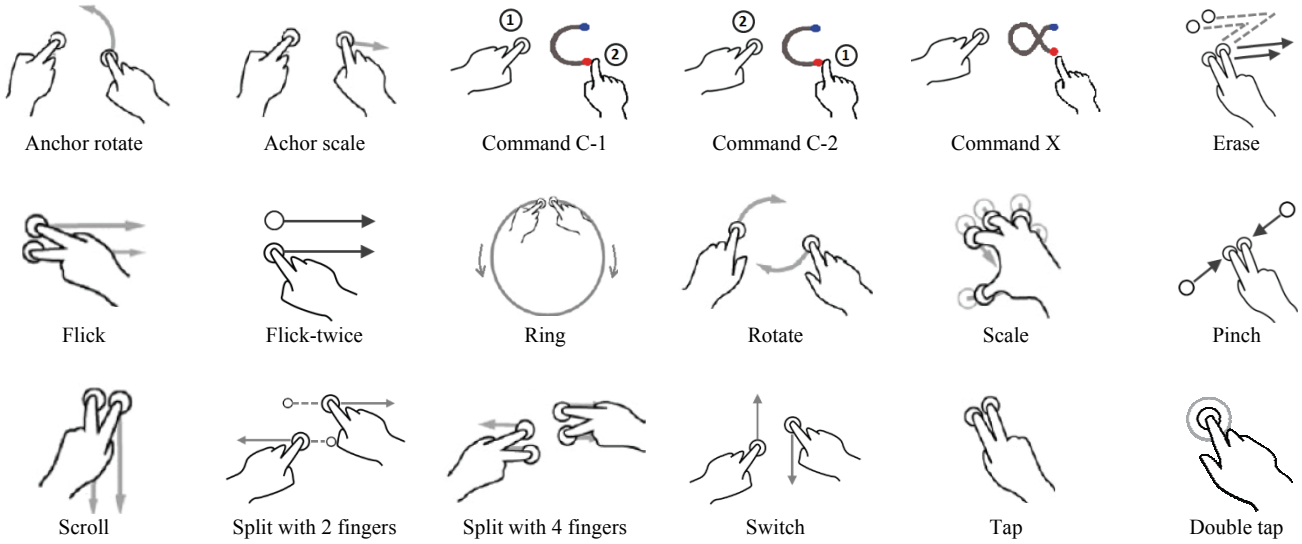


Fig. 5. Multi-touch gestures prototypes in our experimental dataset.

A. Dataset

Since a public dataset for multi-touch recognition can hardly be found, we recently collected a total of 1,800 multi-touch gesture samples, written by 10 persons, for the experiment. This dataset contains 18 different kinds of multi-touch gestures which could be used as command (Fig. 5). These gestures are composed of points, linear segments and arcs with a varying number of strokes. Note that most of them have an apparent distinction according to their shape except two pairs, *Command C-1* versus *Command C-2* and *Flick* versus *Flick twice*. The gesture *C-1* is performed firstly the 'dot' in the left side, then the 'C' shape in the right side. On the contrary, gesture *C-2* is performed firstly the 'C' shape and then the 'dot'. As for the other pair, the *Flick* gesture is performed by two fingers flicking simultaneously, whereas the *Flick-twice* gesture is done by one finger but flicking twice. These two pairs of gestures have the same shape respectively but are different from a temporal point of view. We introduce them in the dataset to evaluate the capability of our recognition method to discriminate gestures from temporal information.

B. Classifier

Since the graph features have been embedded in a vector features, we chose SVM as the classifier. The LIBSVM with a Gaussian kernel and default parameters appeared to provide adequate capability for gesture classification.

C. Experimental design

To fully evaluate our graph embedding strategy, five experiments were conducted with different methods or method combinations respectively.

- *Graph*: In the first experiment, we tried to evaluate our graph modeling strategy without shape information. Only spatial and temporal information were integrated in the graph for recognition.
- *GSF*: In this second experiment, we use directly the HBF49 features without any graph modeling. This could be considered as a baseline system. .
- *Graph+GSF*: In this trial, the feature vector of GSF was simply concatenated after the graph features from the first experiment to achieve a conjoint feature vector, the length of feature vector would be 87+49=136.
- *Graph(LSF)*: The fourth experiment evaluated the performance of integrating the LSF into the graph. The cluster number K is set to an optimal value 9. The comparison of different values of K will be presented below.
- *Graph(LSF)+GSF*: Finally, we combined the graph information with both global and local shape information. It was achieved by concatenating the feature vector of GSF after the vector obtained in the fourth experiment.

All the experiments adopted a 5-cross-validation (CV) scheme for testing writer-independent (WI) performance.

D. Experimental result

Table II summarizes the recognition rates obtained by different methods or their combination.

The results show that the graph strategy, containing only spatial and temporal information, obtains 87.50% recognition rate which is lower than the HBF49 based GSF method (90.44 %). With a deeper investigation from the confusion matrix in Fig. 5, most of the misclassifications by graph strategy happens between the gesture *Command C-1*, *Command C-2* and *Command X* because they have similar spatial relationships between strokes (Fig. 6(a)). We can note that GSF is able to classify the majority of multi-touch but fails to make a distinction between *Command C-1* versus *Command*

C-2 and *Flick* versus *Flick-twice* since they are similar in shape, respectively (Fig. 6(b)).

TABLE II. RECOGNITION RATE OBTAINED BY DIFFERENT METHODS

Method	Length of features	Recognition rate (%)	Std. Deviation
Graph (without shape feature)	87	87.50	0.037
GSF	49	90.44	0.034
Graph + GSF	136	90.11	0.035
Graph (LSF)	96	92.56	0.013
Graph (LSF) + GSF	145	94.50	0.020

The Graph+GSF method results in 90.11% which is slightly lower than GSF method. However, when we integrate the shape information of each stroke inside the graph by using clustering method, the recognition rate of Graph (LSF) rises to 92.56%. In accordance to what can be expected, the final experiment, Graph (LSF) + GSF which integrates all the information together achieves the best recognition rate, 94.50%,

	C1	C2	X	F	F-2
C1	34	17	35	0	0
C2	6	77	15	0	0
X	16	19	60	0	0
F	0	0	0	**99**	1
F-2	0	0	0	3	**96**

(a) Graph (Without shape feature)

	C1	C2	X	F	F-2
C1	**99**	1	0	0	0
C2	36	63	0	0	1
X	0	2	**98**	0	0
F	0	0	0	82	18
F-2	0	0	1	42	56

(b) GSF

	C1	C2	X	F	F-2
C1	49	17	34	0	0
C2	2	82	16	0	0
X	16	12	72	0	0
F	0	0	0	**92**	8
F-2	1	0	0	3	**95**

(c) Graph + GSF

	C1	C2	X	F	F-2
C1	64	24	10	0	0
C2	29	68	0	0	1
X	2	4	81	0	0
F	0	0	0	**99**	1
F-2	0	0	1	2	**96**

(d) Graph (LSF)

	C1	C2	X	F	F-2
C1	75	14	8	0	0
C2	21	77	1	0	0
X	2	0	88	0	0
F	0	0	0	**100**	0
F-2	1	0	0	3	**95**

(e) Graph (LSF) + GSF

C1: Command C-1 **C2**: Command C-2 **X**: Command X

F: Flick **F-2**: Flick-twice

Fig. 6. Confusion matrix of some typical misclassified gestures of different classification methods. The row relates to the ground truth.

that is significantly better than others. Meanwhile, by evaluating the standard deviation of 5-cross-validation, the two strategies with LSF inside produce smaller variations than the other three strategies in the WI situation.

The cluster number K used in the clustering process is also a factor of great concern for the recognition. The comparison results of the recognition rate under different values of K are illustrated in Fig.7.

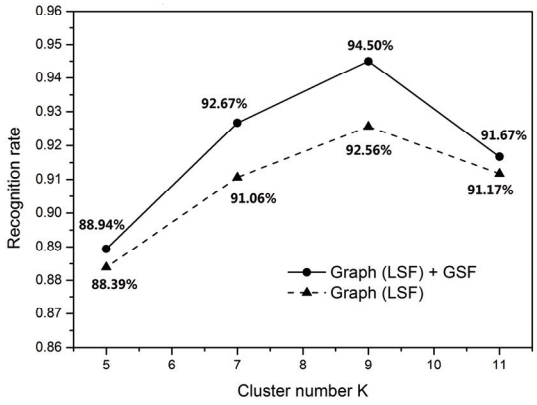


Fig. 7. Performance evaluation of different values of cluster number K

It shows that the same trend in the relationship between cluster number K and recognition rate can be observed from both methods. The peak appears when K is chosen to 9. Neither too large nor too small cluster number are able to well perform. The comparison also proves that the Graph (LSF) + GSF method is always better than the Graph (LSF) method.

V. CONCLUSION AND FUTURE WORK

In this paper, we have investigated a method to recognize the multi-touch gestures. Unlike many other works, we study this problem from a new perspective considering a multi-touch gesture as an indirect command. We believe that three kinds of information, spatial, temporal and shape information, of the gesture should be processed for the recognition. We first proposed a graph modeling method which measures the spatial and temporal relationships between strokes of the gesture. In order to integrate the shape information into the graph, a clustering method is employed to label the shape of the stroke inside the graph as a local shape feature. Another globe shape feature is extracted with our previous baseline method HBF49 features.

We are currently further developing our graph based recognition technologies. Instead of using discrete prototypes, we are considering using numerical features to give a more precise measurement for the spatial and temporal information. In another effort, our proposal of multi-touch gesture as indirect command is planned to be integrated into a sketch drawing application to evaluate its usability. We hope this strategy may offer a more convenient switching mode between raw ink, direct manipulation and indirect command.

ACKNOWLEDGMENT

The authors would like to thank Somia Rahmoun and Boussad Ghedamsi for fundamental research in their master's internship early in this study.

REFERENCES

[1] Tevfik Metin Sezgin, Thomas Stahovich, Randall Davis, "Sketch based interfaces: early processing for sketch understanding", ACM

SIGGRAPH 2006 Courses, July 30-August 03, 2006, Boston, Massachusetts.

[2] Hao Lü , Yang Li, “Gesture coder: a tool for programming multi-touch gestures by demonstration”, Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems, May 05-10, 2012, Austin, Texas, USA.

[3] Chi-Min Oh, Md. Zahidul Islam, Chil-Woo Lee, "MRF-based Particle Filters for Multi-touch Tracking and Gesture Likelihoods," CIT, page 144-149. IEEE Computer Society, 2011.

[4] Gilles Bailly , JöRg MüLler , Eric Lecolinet, “Design and evaluation of finger-count interaction: Combining multitouch gestures and menus”, International Journal of Human-Computer Studies, v.70 n.10, p.673-689, October, 2012.

[5] Sriganesh Madhvanath, Dinesh Mandalapu, Tarun Madan, Naznin Rao, Ramesh Kozhissery, “GeCCo: Finger gesture-based command and control for touch interfaces”, IHCI 2012: 1-6.

[6] Caroline Appert , Shumin Zhai, “Using strokes as command shortcuts: cognitive benefits and toolkit support”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, April 04-09, 2009, Boston, MA, USA

[7] Yang Li, Ken Hinckley, Zhiwei Guan, and James A. Landay, “Experimental analysis of mode switching techniques in pen-based user interfaces”, CHI, page 461-470. ACM, 2005.

[8] Adrien Delaye, Eric Anquetil, “HBF49 feature set: A first unified baseline for online symbol recognition”, Pattern Recognition, 46(1):117-130, 2013.

[9] C.-C.Chang, C.-J.Lin, “LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27.

[10] Kenrick Kin , Björn Hartmann , Tony DeRose , Maneesh Agrawala, “Proton: multitouch gestures as regular expressions”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, May 05-10, 2012, Austin, Texas, USA.

[11] James F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM, v.26 n.11, p.832-843, Nov. 1983.

[12] Nicolas Sidère, Pierre Héroux, Jean-Yves Ramel, "Vector Representation of Graphs: Application to the Classification of Symbols and Letters," Proceedings of ICDAR, pp.681-685, 2009 10th International Conference on Document Analysis and Recognition, 2009.