Structuring and Manipulating Hand-Drawn Concept Maps

Yingying Jiang[1,2] Feng Tian[2] Xugang Wang[2] Xiaolong Zhang[3] Guozhong Dai[1,2] Hongan Wang[1,2]

[1]State Key Laboratory of Computer Science,
Institute of Software,
Chinese Academy of Sciences
Beijing, China
Tel: +86-10-6266-1577
E-mail: {jyy,tf,wxg }@iel.iscas.ac.cn

[2]Intelligence Engineering Lab, Institute of Software,
Chinese Academy of Sciences
Tel: +86-10-6266-1577
E-mail: {dgz,wha}@iel.iscas.ac.cn
[3]The Pennsylvania State University
E-mail: lzhang@ist.psu.edu

ABSTRACT

Concept maps are an important tool to knowledge organization, representation, and sharing. Most current concept map tools do not provide full support for hand-drawn concept map creation and manipulation, largely due to the lack of methods to recognize hand-drawn concept maps. This paper proposes a structure recognition method. Our algorithm can extract node blocks and link blocks of a hand-drawn concept map by combining dynamic programming and graph partitioning and then build a concept-map structure by relating extracted nodes and links. We also introduce structure-based intelligent manipulation technique of hand-drawn concept maps. Evaluation shows that our method has high structure recognition accuracy in real time, and the intelligent manipulation technique is efficient and effective.

ACM Classification: H5.2 [Information Interfaces and Presentation]: User Interfaces. - Interaction styles; I.5.4 [Pattern Recognition]: Applications. - Signal processing.

General terms: Design, Algorithms

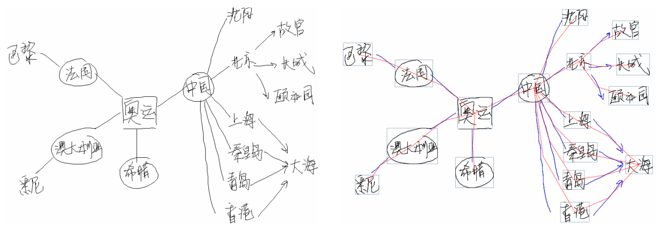
Keywords: Hand-drawn concept map, structure recognition, dynamic programming, graph partition.

INTRODUCTION

Concept maps are tools for organizing and representing knowledge [9]. In a concept map, important concepts are presented and relationships among concepts are visualized as lines connecting relevant concepts. Concept maps could be used for education and cooperation in schools and corporations [10]. People can use tools like MindManager, Inspiration or FreeMind to create concept maps with a keyboard and mouse.

With tools to support the creation and management of hand-drawn concept maps, users can focus on their primary tasks, such as learning. Pen-based devices like tablet PCs allow users to draw concept maps directly and naturally. However, current pen-based tools are weak in supporting hand-drawn concept maps. For example, MindManager can only handle concept nodes with pen gestures, not concept links; Inspiration allows users to draw nodes and links, but users must follow specific drawing rules and orders to make nodes and links recognizable; and sKEA [3] requires users to specify explicitly where a symbol starts and ends in concept maps. Thus, it is still difficult to create and manipulate hand-drawn concept maps in a natural and fluent fashion similar to what we do with a pen on paper.

This challenge is largely due to the lack of methods to recognize node and link components of hand-drawn concept maps. This paper proposes a structure recognition algorithm to extracts nodes, links, and their relationships from a hand-drawn concept map that is created without any constraint on drawing order. Figure 1 shows what our algorithm can deliver. Figure 1(a) is a hand-drawn concept map, and Figure 1(b) illustrates the extracted structure with our algorithm: strokes within a bounding box belong to a node block; link strokes are highlighted with blue; red lines are added to connect node blocks with link blocks.



(a) (b)

Figure 1. A hand-drawn concept map (a) and its extracted structure (b).

The following sections address related research, features of hand-drawn concept maps, structure understanding algorithm, intelligent manipulation technique, performance evaluation, conclusions and future work.

RELATED WORK

Sketch understanding has been studied by many researchers since 1970s. Research has been done on domain-specific sketch recognition, such as recognition of handwritten organic chemical formulae [11], hand drawn UML [5], and handwriting notes [8]. These methods address sketch understanding problems in their specific domains, but their domain-specific optimization approach makes them unfit for structure understanding of more general hand-drawn diagrams, such as concept maps.

Some researchers have investigated sketch understanding methods that can be used more broadly. As it is difficult to define grammars for various kinds of concept maps, the grammar-based sketch recognition methods proposed by Alvarado [1] and Shilman [12] is not fit for concept map structure recognition. The hierarchical sketch recognition proposed by Kara et al. [6] could not be applied directly to concept maps, in which link strokes can be very similar to some node strokes. Szummer's joint probabilistic model [14] can simultaneously group and recognize ink based on the dependencies among ink fragments and user feedback. Nevertheless, different node styles and link styles in concept maps make it difficult to define dependency context. The recognition-based segmentation method by Shilman et al. [13] used dynamic programming to segment strokes. However, the increase of stroke numbers results in sharp decrease of time efficiency. The recognition method by Gennari [4] is based on geometry and domain knowledge and is suitable for network-like diagrams that contain isolated, non-overlapping symbols. However, it requires strokes in one symbol be successive, and cannot handle concept maps, in which one node is not always drawn in one step.

Research has shown that to manipulate a hand-drawn diagram, techniques based on an underlying structure of the diagram are effective and efficient. For example, Ao et al. [2] showed that structuralizing raw digital ink as multiple hierarchies can facilitate selection tasks and improve task performance, and Li et al. [8] found that allowing user interaction with note semantics, rather than individual strokes, can better help notes manipulation.

CHARACTERISTICS OF HAND-DRAWN CONCEPT MAP

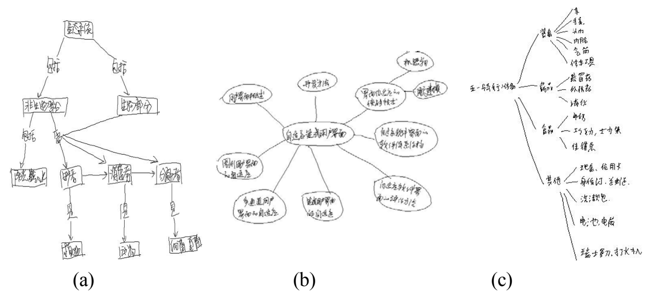


Figure 2. Three hand-drawn concept maps on paper.

We interviewed 12 people who often use concept maps in learning and asked them to draw concept maps with pen and paper. Figure 2 shows three concept maps from them. We identified the following features of hand-drawn concept maps:

- Concept maps can have various spatial layout styles and structural styles (tree or network) (Figure 2). However, they are diagrams with basic node and link elements.
- Concept nodes and links can also have different styles. Node may have a bounding shape (e.g., box, oval) or not. Link can be a line with arrow or a simple line. Most links are not curved.
- There are two kinds of relationships between nodes: a parent-child relationship, indicated by a unidirectional arrow connecting two nodes, or a brother relationship indicated by a non-directional line or a bidirectional arrow between two nodes.

During the interview, some people mentioned that while it was easy to create concept maps on paper, it was often difficult to edit them. They would like to have a tool that allows them to create a concept map easily and to edit and organize it conveniently. To address these needs, we developed a structure understanding algorithm to exact concept map structures and then designed structure-based intelligent manipulation technique to manipulate concept maps.

STUCTURE UNDERSTANDING ALGORITHM

The key to understanding concept map structure is the identification of concept nodes and links. In a hand-drawn concept map, node blocks and link blocks are often close to or even connected to each other, and segmenting nodes and links only based on stroke clustering is infeasible. One way to separate node blocks from link blocks is to recognize link strokes first and then use these link strokes as delimiters to get individual node blocks [6]. However, this approach has a problem in handling node strokes similar to link strokes, which would lead to over-segmentation result. We address this over-segmentation issue by developing an algorithm that combines dynamic programming and graph partition. Dynamic programming is used to extract optimal link blocks and node blocks and meanwhile to merge over-segmented node blocks. Graph partition is adopted to decompose a large graph into smaller subgraphs to improve the time efficiency.

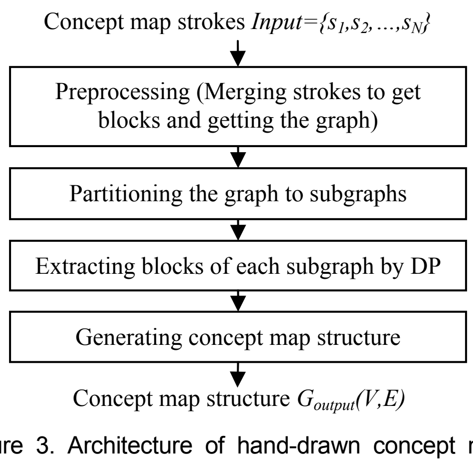

Concept map strokes $Input=\{s_1,s_2,\ldots,s_N\}$
Preprocessing (Merging strokes to get blocks and getting the graph)
Partitioning the graph to subgraphs
Extracting blocks of each subgraph by DP
Generating concept map structure
Concept map structure $G_{output}(V,E)$

Figure 3. Architecture of hand-drawn concept map structure understanding.

Figure 3 shows the architecture of our algorithm to extract the structure of a hand-drawn concept map. The first preprocessing step judges each stroke's type, merges non-link strokes, and represents the concept map as a graph. The second step partitions the graph into several subgraphs by a graph partitioning algorithm. The third step extracts node blocks and link blocks from each subgraph by using dynamic programming. The final step generates the concept map structure based on the blocks obtained from Step 3. The following sections provide more details of each step of our algorithm.

Preprocessing

The input of this step is a series of strokes *{$s_1, s_2, \ldots, s_N$}*, where s_i is the *ith* stroke and *N* is the total stroke number, and the output is a graph to store preprocessed blocks. Our algorithm first uses the $1 recognizer [15] to identify whether a stroke is a link stroke. Here, we assume a link stroke resembles a straight line or a line with arrow. Neighboring non-link strokes are merged to blocks, and a graph is created to hold these stroke blocks.

Whether two non-link strokes should be merged is based on their distance. If *dist(b_1, b_2)* is smaller than a threshold value and neither b_1 nor b_2 is a link stroke block, they are merged; otherwise, they are two separate blocks. Merged stroke blocks can be represented as {b_1, b_2, ..., b_n}, where $b_i=\{s_{i1}, s_{i2}, \ldots, s_{im}\}$.

With all stroke blocks after merging, a weighted undirected graph *G* can be built. The nodes in *G* represent stroke blocks and the edges in *G* indicate the relationships between stroke blocks. w_{ij} is the distance between blocks b_i and b_j. *weightThres* is set to be 1.5 times of the average diagonal length of the bounding boxes of all stroke blocks.

$$\begin{cases} G=(V,E,W) \\ V=\{b_i \mid i\in[1,n]\} \\ E=\{(b_i,b_j) \mid (i,j\in[1,n]) \wedge (i!=j) \wedge (w_{ij} < weightThres)\} \\ W=\{w_{ij} \mid (i,j\in[1,n])\} \end{cases}$$

Figure 4(a) shows 13 stroke blocks after pre-merging. Figure 4(b) is the graph *G* corresponding to Figure 4(a).

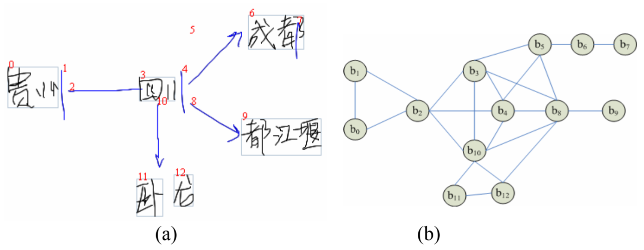


Figure 4. Preprocessing result: (a) stroke blocks; (b) a graph to represent stroke blocks and their relationships.

Graph Decomposition with Graph Partitioning

The second step of our algorithm is to partition the graph *G* into several subgraphs by graph partitioning. Our algorithm is based on the graph partitioning method in hMETIS [7].

We suppose *gpThres* is the maximum node number that a subgraph could have and it equals to 8 in this research. When the node number of *G* is less than *gpThres* or equals to *gpThres*, dynamic programming is applied to find the optimal block segmentation. If the node number is larger than *gpThres*, the graph is further divided into two subgraphs. The process iterates until the node numbers of all subgraphs are not larger than *gpThres.*

Figure 5 shows the graph partitioning result of the graph shown in Figure 4(b). There are two subgraphs, distinguished by their node colors. After graph partitioning, not all blocks in one subgraph are spatially close to each other, but blocks that belong to one concept node are in one subgraph.

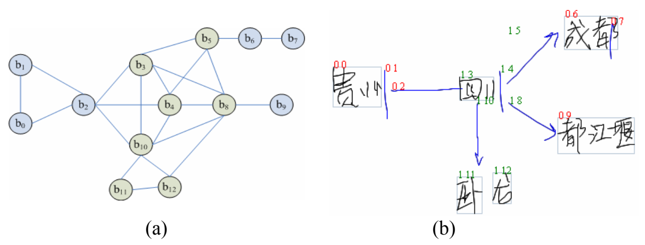


Figure 5. Graph partition result: (a) the graph after graph partition; (b) corresponding stroke blocks.

Block Extraction by Dynamic Programming

Dynamic programming can extract optimal stroke blocks and merge preprocessed over-segmented blocks. Our algorithm first builds a candidate block set *S*. A stroke block after preprocessing is a candidate block and a stroke block together with its adjacent blocks constitutes a candidate block as well.

Then, our algorithm calculates the reliability of each candidate block. The reliability of a block is determined by three factors: *densityFactor(V')* is the ratio of stroke length to its bounding box's diagonal length; *distFactor(V')* is in inverse proportion to the distances between constituent stroke blocks; *contextFactor(V')* is related to the relationships between constituent stroke blocks. If *V'* is a candidate block in *S* (*V'*∈ *S*), its reliability *R(V')* can be computed by the following formula.

$$R(V') = a * densityFactor(V') + b * distFactor(V') \\ + (1-a-b) * contextFactor(V')$$

a, b and (1- *a* - *b)* are coefficients.

The task of block extraction is to find the optimal candidate block set {V_1, V_2, ... V_M}, where $V_i = \{b_{i1}, b_{i2}, \ldots b_{iN'}\}$ is a candidate block and is composed of preprocessed blocks b_{i1}, b_{i2}, ... $b_{iN'}$. The following formula describes the approach to find optimal block segmentations by dynamic programming.

$$C(V) = \begin{cases} R(V), if\ |V| \le 1 \\ \max\limits_{(V' \subseteq V) \wedge (V' \in S)} \{R(V), \phi(R(V'), C(V-V'))\}, if\ (|V|>1) \wedge (V \in S) \\ \max\limits_{(V' \subseteq V) \wedge (V' \in S)} \{\phi(R(V'), C(V-V'))\}, else \end{cases}$$

C(V) is the reliability corresponding to the optimal block segmentation of *V*. It is the maximum value of the valid segmentations. *R(V)* is the reliability of a candidate block *V*. *ϕ(R(V'),C(V-V'))* defines the strategy to combine sub-problem *V'* and sub-problem *V-V'*, and is written as:

$$\phi(R(V'),C(V-V'))=\frac{|V'|*R(V')+|V-V'|*C(V-V')}{|V|}$$

where, $|V|$ is the number of nodes in V.

Concept Map Structure Extraction

The final concept map structure is represented in the form of an undirected graph $G_{output}(V,E)$. Every edge (V_i, V_j) satisfies $V_i.type!=V_j.type$, i.e., node blocks and link blocks are adjacent in a concept map.

$$\begin{cases} G_{output}=(V,E) \\ V=\{V_i \mid i\in[1,M]\} \\ E=\{(V_i,V_j) \mid (i,j\in[1,M])\wedge(i!=j)\wedge(V_i.type!=V_j.type)\} \end{cases}$$

Because a link can connect at most two nodes, a concept map structure can be extracted by identifying two node blocks for each link block. For complex concept maps, multiple node blocks may be found at one end of a link block and only the block that is closest to the end of the link is considered.

Figure 6 shows the structure extraction results. Figure 6(a) demonstrates stroke blocks extracted from two sub-problems in Figure 5 by dynamic programming. Figure 6(b) shows the relationships between nodes and links with added red lines to connect nodes and links.

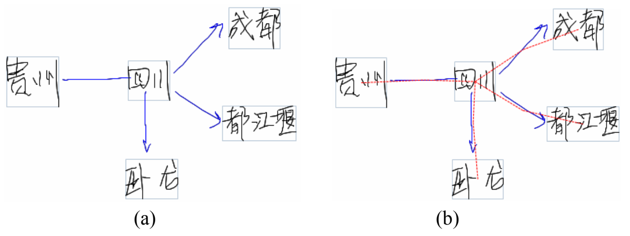


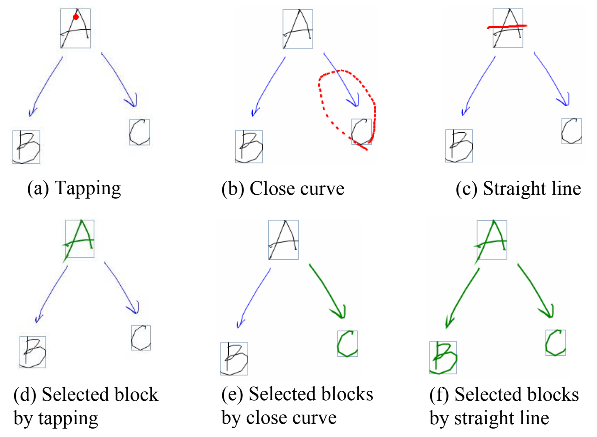
Figure 6. The structure extraction results: (a) stroke blocks after dynamic programming; (b) relationships between nodes and links

In addition, we extract parent-child relationships and brother relationships from the graph G. These relationships can be determined by the types of link blocks people draw. Such semantic information of relationships is critical to the design of manipulation technique.

INTELLIGENT MANIPULATION ON CONCEPT MAPS

As concept maps are often used for creating and refining knowledge representations, people need to modify concept maps frequently. We developed a set of pen gestures that allow users to handle concept maps by directly acting on extracted structures.

Three kinds of pen gestures were designed for selection at different levels (Figure 7): 1) tapping a stroke block to select the block – a link or a node; 2) drawing a closed curve to select multiple blocks; and 3) drawing a straight line over a node block to select the block as well as all its child node blocks and relevant link blocks. The third selection method is a semantic-based technique that considers the relationships among node blocks.



(a) Tapping
(b) Close curve
(c) Straight line
(d) Selected block by tapping
(e) Selected blocks by close curve
(f) Selected blocks by straight line

Figure 7. Three kinds of selection methods.

Similarly, we designed semantic translation based on the extracted structure (Figure 8). In addition, other pen gestures were designed for exchanging concept nodes, scaling, copying, deleting, and correcting errors.

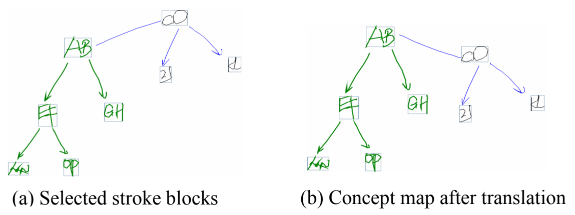


Figure 8. Examples of translating concept nodes.

EVALUATION

We first evaluated the effectiveness and efficiency of our algorithm. Our test was on a machine equipped with a 2.4GH CPU, 2G memory and a Wacom screen. Test data consisted of 45 hand-drawn concept maps from ten student subjects. The numbers of links and nodes in a concept map ranged from 10 to 40 and the average was 21.1 (SD=7.2).

Figure 9 shows the block extraction results of 45 concept maps. The average block extraction error rate is 4.82%. Most of these errors are over-segmentation errors that are caused by far-distant strokes and misrecognized link strokes. When node blocks and link blocks are correctly extracted, structure extraction accuracy is 99.5%. The errors are mainly caused by the incompleteness of concept maps themselves.

The average time to understand the structure of one concept map was about one second. Thus, our algorithm can easily be integrated into real-time tasks.

We also compared our intelligent manipulation technique with stroke-based manipulation technique. Twelve graduate students participated in an experiment that involved a task to change a given concept map to a targeted one. Two concept maps were used in the experiment. We collected each subject's task completion time.

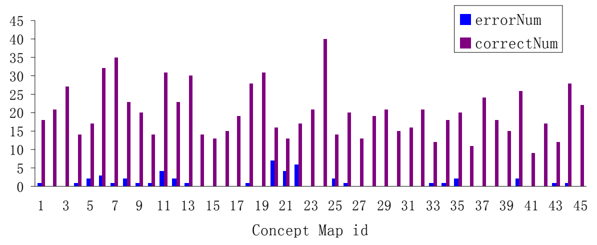


Figure 9. Accuracy of extracted blocks.

Figure 10 shows the average task completion times with two manipulation techniques. Our structure-based intelligent manipulation is significantly shorter than stroke-based manipulation technique (*p<.001*).

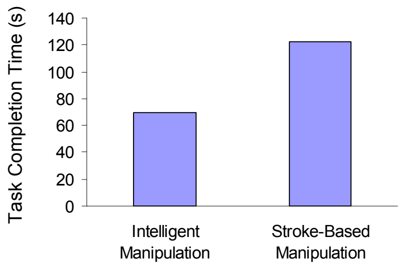


Figure 10. Efficiency comparison of intelligent manipulation and stroke-based manipulation.

CONCLUSION

This paper proposes a method to understand the structures of hand-drawn concept maps and a kind of structure-based intelligent manipulation technique. Evaluation shows both the algorithm and the manipulation technique is effective.

Although our algorithm is designed for concept maps, it can be extended to recognize sketches in other domains by using different reliability functions. Our method of combining dynamic programming and graph partitioning offers an effective approach to solve complex sketch recognition problems in real-time.

Currently, our algorithm could not recognize complex link types and the labels on the links. In the future, we will enhance our algorithm by addressing these limitations. In addition, we will integrate users' drawing habits in our algorithm to improve the accuracy of structure understanding.

ACKNOWLEDGEMENT

This research is supported by National Key Basic Research and Development Program of China under Grant No. 2009CB320804, the National Natural Science Foundation of China under Grant No. U0735004, No. 60503054, No. 60603073, and the National High Technology Development Program of China under Grant No. 2007AA01Z158.

REFERENCES

1. Alvarado, C. Multi-domain Sketch Understanding. PhD Thesis. Massachusetts Institute of Technology. September, 2004.
2. Ao, X., Li, J.F., Wang, X.G. and Dai, G.Z. Structuralizing digital ink for efficient selection. In *Proc. IUI 2006*, 148-154.
3. Forbus, K.D. and Usher, J.M. Sketching for knowledge capture: a progress report. In *Proc. IUI 2002*, 71-77.
4. Gennari, L., Kara, L.B., and Stahovich, T.F. Combining Geometry and Domain Knowledge to Interpret Hand-Drawn Diagrams. Computers & Graphics 29,4(2005), 547-562.
5. Hammond, T. and Davis, R. Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams. In *Proceedings of AAAI Spring Symposium on Sketch Understanding*, 2002, 59-66.
6. Kara, L.B. and Stahovich, T.F. Hierarchical parsing and recognition of hand-sketched diagrams. In *Proc. UIST 2004*, 13-22.
7. Karypis, G. and Kumar, V. hMETIS: A Hypergraph Partitioning Package (Version 1.5.3) [EB/OL]. http://glaros.dtc.umn.edu/gkhome/views/metis/hmetis/download.html.
8. Li, Y., Guan, Z.W., Wang, H.A., Dai, G.Z. and Ren, X.S. Structuralizing Freeform Notes by Implicit Sketch Understanding, In *Proceedings of AAAI Spring Symposium on Sketch Understanding*, 2002, 91-98.
9. Novak, J. D. and Cañas, A. J. The Theory Underlying Concept Maps and How to Construct Them. *Technical Report IHMC CmapTools 2006-01 Rev 01-2008.* Florida Institute for Human and Machine Cognition, 2008.
10. Novak, J.D. *Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations.* Lawrence Erlbaum Associates, Mahwah, NJ, 1998.
11. Ouyang, T.Y. and Davis, R. Recognition of Hand-Drawn Chemical Diagrams. In *Proc AAAI 2007*, 846-851.
12. Shilman M, Pasula H, Russell S, Newton R. Statistical visual language models for ink parsing. In *Proceedings of AAAI Spring Symposium on Sketch Understanding,* 2002, 126－132.
13. Shilman, M., Viola, P. and Chellapilla, K. Recognition and Grouping of Handwritten Text in Diagrams and Equations. In *Proc. IWFHR 2004*, 569－574.
14. Szummer, M. and Cowans. P. Incorporating Context and User Feedback in Pen-Based Interfaces. In *Proceeding of AAAI Fall Workshop on Making Pen-Based Interaction Intelligent and Natural,* 2004, 159-166.
15. Wobbrock, J.O., Wilson, A.D. and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proc. UIST 2007*, 159-168.