Understanding, Manipulating and Searching Hand-Drawn Concept Maps

YINGYING JIANG and FENG TIAN, Chinese Academy of Sciences
XIAOLONG (LUKE) ZHANG, The Pennsylvania State University
GUOZHONG DAI and HONGAN WANG, Chinese Academy of Sciences

Concept maps are an important tool to organize, represent, and share knowledge. Building a concept map involves creating text-based concepts and specifying their relationships with line-based links. Current concept map tools usually impose specific task structures for text and link construction, and may increase cognitive burden to generate and interact with concept maps. While pen-based devices (e.g., tablet PCs) offer users more freedom in drawing concept maps with a pen or stylus more naturally, the support for hand-drawn concept map creation and manipulation is still limited, largely due to the lack of methods to recognize the components and structures of hand-drawn concept maps. This article proposes a method to understand hand-drawn concept maps. Our algorithm can extract node blocks, or concept blocks, and link blocks of a hand-drawn concept map by combining dynamic programming and graph partitioning, recognize the text content of each concept node, and build a concept-map structure by relating concepts and links. We also design an algorithm for concept map retrieval based on hand-drawn queries. With our algorithms, we introduce structure-based intelligent manipulation techniques and ink-based retrieval techniques to support the management and modification of hand-drawn concept maps. Results from our evaluation study show high structure recognition accuracy in real time of our method, and good usability of intelligent manipulation and retrieval techniques.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*User-centered design; interaction styles; theory and methods*

General Terms: Design, Algorithms, Human Factors

Additional Key Words and Phrases: Hand-drawn concept map, recognition, retrieval, intelligent manipulation

ACM Reference Format:
Jiang, Y., Tian, F., Zhang, X., Dai, G., and Wang, H. 2011. Understanding, manipulating and searching hand-drawn concept maps. ACM Trans. Intell. Syst. Technol. 3, 1, Article 11 (October 2011), 21 pages.
DOI = 10.1145/2036264.2036275 http://doi.acm.org/10.1145/2036264.2036275

Partial results of this work appeared as a conference short paper [Jiang et al. 2009] at Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI'09).
This research was supported by the National Key Basic Research and Development Program of China under Grant no. 2009CB320804, The National Natural Science Foundation of China under Grant no. U0735004, and the National High Technology Development Program of China under Grant no. 2007AA01Z158 and no. 2009AA01Z337.
Authors' addresses: Y. Jiang (corresponding author) and F. Tian, Institute of Software, Chinese Academy of Sciences, Beijing, China; email: jyy.ios@gmail.com; X. Zhang, College of Information Sciences and Technology, Pennsylvania State University, PA 16802; G. Dai and H. Wang, State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China.

DOI 10.1145/2036264.2036275 http://doi.acm.org/10.1145/2036264.2036275

1. INTRODUCTION

Concept maps are often used as a tool for knowledge organization, representation, and sharing [Coffey et al. 2002; Novak and Cañas 2008; Tergan 2005]. In a concept map, important concepts are presented as text-based nodes and relationships among concepts are visualized as line-style links to connect relevant concepts. Concept maps could be used for education and cooperation in schools and corporations [Novak 1998; Novak and Gowin 1984]. In the process of text analysis, a concept map could be used to structure textual knowledge and achieve better recall of information [Halimi 2006]. Currently, people can use tools like MindManager,[1] Inspiration,[2] or FreeMind[3] to create concept maps with a keyboard and mouse. This keyboard-and-mouse approach suffers two problems. First, it usually requires users to follow specific task structures (e.g., two concept nodes must be created before their link can be generated), which could distract users from their primary tasks. Second, it becomes inefficient and ineffective to use keyboard and mouse on devices that support more direct interaction, such as pen-based and touch-based computers, because such devices usually only provide a virtual keyboard and offer gesture-based interaction that is different from mouse behaviors.

Hand-drawn concept map tools can help to overcome these two challenges. They allow users to write and sketch directly with gestures. However, existing tools are still weak when processing hand-drawn concept maps. It is difficult to create hand-drawn concept maps in a natural and fluent fashion. For example, MindManager can only handle concept nodes with pen gestures, not concept links; Inspiration requires users to follow specific drawing rules and orders to make nodes and links recognizable; sKEA [Forbus and Usher 2002] demands explicit information about where a symbol starts and ends in concept maps; and Dashboard[4] allows drawing of concept maps, while the meaning and structure of concept maps are not understood. Moreover, users also face challenges in other high-level tasks, such as manipulating and searching hand-drawn concept maps.

The problem is largely due to the lack of methods to recognize and structure node and link components of hand-drawn concept maps. This article proposes a recognition algorithm to extracts nodes, links, and their relationships from a hand-drawn concept map that is created without any constraint on drawing order. Meanwhile, a retrieval algorithm is designed for hand-drawn concept maps. Figure 1 shows what our recognition algorithm can deliver. Figure 1(a) is a hand-drawn concept map about the Olympics. All handwritten concept nodes are in Chinese (English translation is provided by authors to indicate the meaning of each node). Figure 1(b) illustrates the extracted structure and the recognition result with our algorithm. In this figure, strokes belonging to the same node block are put a bounding box and the text below each bounding box is the recognition result of the node block; link strokes are also recognized and thick dashed lines are added to connect node blocks and link blocks. These dashed lines can help users evaluate whether the relationships between node blocks and link blocks are correctly understood by the algorithm.

The article is structured as the following. First, we review related research, and then outline a set of features commonly seen from hand-drawn concept maps based on our user interviews. Next, we describe the details of the recognition algorithm and the retrieval algorithm, and present a set of intelligent manipulation techniques for the management and modification of concept maps based on recognized structures.

[1] MINDMANAGER. http://www.mindjet.com/products/mindmanager_pro/default.aspx.
[2] INSPIRATION. http://www.inspiration.com/productinfo/inspiration/index.cfm.
[3] FREEMIND. http://freemind.sourceforge.net/wiki/index.php/Main_Page.
[4] DABBLEBOARD. http://www.dabbleboard.com/.

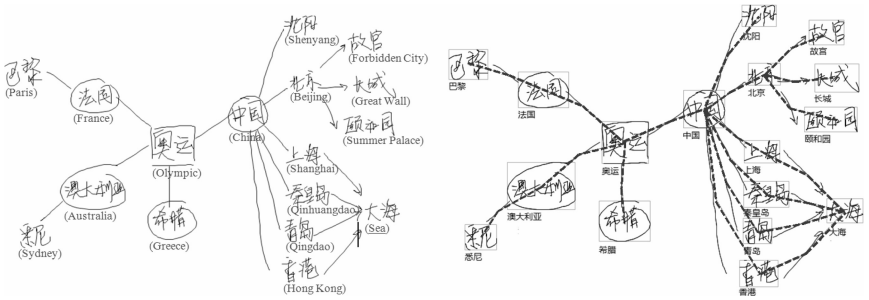


Fig. 1. A hand-drawn concept map (a) and its extracted structure (b).

Furthermore, we introduce our research on evaluating the accuracy and usability of our approach. Finally we discuss the results of our research and conclude the article with future research directions.

2. RELATED WORK

Because our interest is in hand-drawn concept maps, literature reviewed here concerns previous research on sketch understanding, sketch manipulation, and sketch retrieval.

2.1. Sketch Understanding

Sketch understanding has been studied by many researchers since the 1970's. Research has been conducted on domain-specific sketch recognition, such as recognition of handwritten mathematical expressions [Chan and Yeung 2000], handwritten organic chemical formulae [Ouyang and Davis 2007], hand-drawn UML [Hammond and Davis 2002], and handwriting notes [Li et al. 2002]. These methods address sketch understanding problems in their specific domains, but this domain-specific optimization approach cannot be directly applied in the structure understanding of more general, out-of-domain hand-drawn diagrams, such as concept maps. Chik et al. [2007] tried to understand pen-based mind-maps by identifying the first node as the central node and then finding subnodes connected to it recursively.

Some researchers have investigated sketch understanding methods that can be used more broadly. Alvarado [2004] and Shilman et al. [2002] proposed grammar-based sketch recognition methods, but due to the difficulty in defining grammars for various types of concept maps, this approach is inappropriate to concept map structure recognition. The method by Kara and Stahovich [2004] achieved diagram recognition with a hierarchical approach to parse and recognize strokes, but it faces challenges when recognizing concept maps because of the similarity between link strokes and some node strokes. While the joint probabilistic model [Szummer and Cowans 2004] can simultaneously group and recognize inks based on dependencies among ink fragments and user feedback, different node styles and link styles in concept maps pose challenges in defining context dependency. The recognition-based segmentation method by Shilman et al. [2004] used dynamic programming to segment strokes. However, the increase of stroke number can dramatically decrease the time efficiency of this method. The recognition method by Gennari et al. [2005] is based on geometry and domain knowledge and is suitable for network-like diagrams that contain isolated, nonoverlapping symbols. It requires strokes in one symbol be successive and cannot handle concept maps, in which one node is not always drawn in one step.

2.2. Sketch Manipulation

Research has shown that to manipulate a hand-drawn diagram, techniques based on an underlying structure of the diagram are effective and efficient [Ao et al. 2006; Jiang et al. 2009, 2010; Li et al. 2002]. For example, Ao et al. [2006] showed that structuralizing raw digital ink as multiple hierarchies can facilitate selection tasks and improve task performance, and Li et al. [2002] found that allowing user interaction with note semantics, rather than individual strokes, can better help notes manipulation.

Some sketch manipulation tools have been designed. Stretch-A-Sketch [Gross 1994] is a pen-based drawing program that combines recognition of hand-drawn glyphs with constraint-based maintenance of spatial relationships. However, its recognizer does not fit for hand-drawn concept maps that have different node and link styles, and the program can only handle local constraints, not global spatial constraints. The structured editing tool by Moran et al. [1995] can detect implicit structures that humans see in drawing, but this technique only supports editing list, text, table, and outline structures over handwritten scribbles and typed text. Scanscribe [Saund et al. 2003] is a novel image editing program emphasizing easy selection and manipulation of material found in documents based on visual perceptual organization theories (e.g., the Gestalt laws). It is acted on offline sketch which is different from concept maps with strokes, however. The pen-based mind-mapping tool by Chik et al. [2007] can recognize hand-drawn hierarchical structures and allow the revision of the recognized structure. However, this tool can only handle hierarchical structures and has a strict requirement for the order of objects to be drawn (e.g., a parent node must be drawn before any child node).

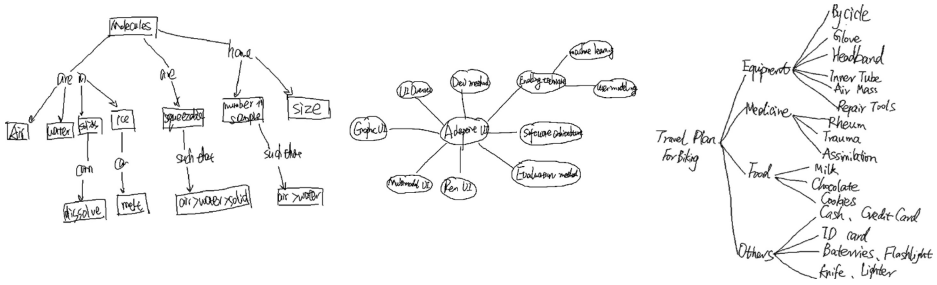
2.3. Sketch Retrieval

Researchers have studied methods for sketch and online document retrieval. Liang and Sun [2008] used biased SVM classification for sketch retrieval by matching query sketch with sketches in libraries based on their weighted spatial distance. Leung [2003] proposed a method to compare the features of sketches with global and local matching. However, these methods, which are primarily based on stroke correspondence, are inappropriate for hand-drawn concept map retrieval, because concept maps with similar concept structures may have different graphics appearances. As for online document retrieval tasks, InkSeine allows searching in formal texts by recognizing users' handwriting into text [Hinckley et al. 2007]; Jain and Namboodiri [2003] proposed a method to index and retrieve online handwritten documents; and Jawahar et al. [2009] tried to retrieve online handwriting documents by synthesizing and matching handwritten data. However, these methods can only support search based on sketched text, and cannot be used for searching concept maps, which have graph structures.

Graph matching is widely used in areas such as image analysis, document processing, biometric identification, image databases, and video analysis [Conte et al. 2004]. In Zhou et al. [2005], a graph-matching approach was used for intelligent multimedia retrieval. Cordella et al. [2001] presented a graph-matching algorithm to address the problems of graph and graph-subgraph isomorphism. As hand-drawn concept maps have graph structures, it is possible to retrieve concept maps using graph-matching methods. However, to the best of our knowledge, we have not found such graph-matching techniques used in hand-drawn concept map retrieval.

3. CHARACTERISTICS OF HAND-DRAWN CONCEPT MAP

To design tools that can handle the complexity of concept maps created by different people and with different styles, we first conducted an interview study to capture



(a) (b) (c)

Fig. 2. Three hand-drawn concept maps on paper. (a) a concept map about high-school chemistry; (b) a diagram on the relationship among some concepts in human-computer interaction; (c) a bike travel plan.

common structural features of hand-drawn concept maps. We interviewed 12 people who often use concept maps. Two of them have used computer-based concept maps before. We asked them to draw concept maps with pen and paper. Figure 2 shows three concept maps from them.

Based on drawings collected from subjects, we identified the following features of hand-drawn concept maps.

— Concept maps can have various spatial layout styles, such as top-down style shown in Figure 2(a), center-out style in Figure 2(b), or left-right style Figure 2(c). In addition, they can have different structural styles: as a tree, or as a network. However, despite different layout and structural styles, all concept maps are diagrams with basic node and link elements.
— Concept nodes and links can also have different styles. Nodes may have a bounding shape (e.g., box, oval) or not. A 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, which is indicated by unidirectional arrows connecting two nodes, or a brother-brother relationship indicated by a nondirectional line or a bidirectional arrow between two nodes.
— In addition to these representation features, we also observed people's different drawing habits. Some people drew all concept nodes first and then added links between them, while some drew links and nodes in random order. As for concept nodes with bounding shapes, some people drew node content first and then the bounding shape, while some did in the opposite way. Furthermore, people did not always finish a node in one step. They may draw part of a node first and then finish it after other links or nodes. In addition, most links are drawn in one stroke. Although different users have different preferences for drawing styles, we did not find a correlation between users' experience of computer-based concept maps and the concept map styles they drew.

During the interview, some people mentioned that while it was easy to create concept maps on paper, it was often difficult to edit them. Moreover, it was inconvenient to retrieve concept maps in their notes. They would like to have a computer-based concept map tool that allows them to create a concept map easily and at the same time to edit, organize, and retrieve it conveniently. In addition, users who have used computer-based concept maps were glad and excited to use a hand-drawn concept map tool, largely because hand-drawn concept maps are more intuitive and flexible.

To address these needs, we developed a hand-drawn concept map understanding algorithm to extract concept map structures and then designed intelligent manipulation techniques to manipulate concept maps through extracted structures. Moreover, we designed retrieval technique for hand-drawn concept maps.

4. UNDERSTANDING OF HAND-DRAWN CONCEPT MAPS

4.1. Hand-Drawn Concept Map Understanding Algorithm

The key to understanding a concept map is to extract 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 [Kara and Stahovich 2004]. However, this approach has a problem in handling node strokes similar to link strokes, which would lead to an over-segmentation result. We address this over-segmentation issue by developing an algorithm that combines dynamic programming [Cormen et al. 2001] 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.

Our algorithm has five steps to understand a concept map based on hand-drawn strokes. The first preprocessing step judges each stroke's type, merges nonlink strokes, and represents the concept map as a graph. The nodes of the graph are stroke blocks and the edges are relationships between stroke blocks (e.g., their distance). 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 fourth step generates the concept map structure based on the blocks obtained from step 3. The final step recognizes concept nodes to get a semantic concept map. The following sections provide more details of each step of our algorithm. (The pseudocodes are given in Appendix.)

4.1.1. 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 [Wobbrock et al. 2007] 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 nonlink strokes are merged to blocks, and a graph is created to hold these stroke blocks.

Whether two nonlink strokes should be merged is based on their distance. If the distance between two stroke blocks is smaller than a threshold value and neither stroke block is a link stroke block, they are merged; otherwise, they are two separate blocks. Merged stroke blocks can be represented as $\{b_1, b_2, \ldots, b_n\}$, where b_i=$\{s_{i1}, s_{i2}, \ldots, s_{im}\}$.

With all stroke blocks after merging, a weighted undirected graph G can be built:

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

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.

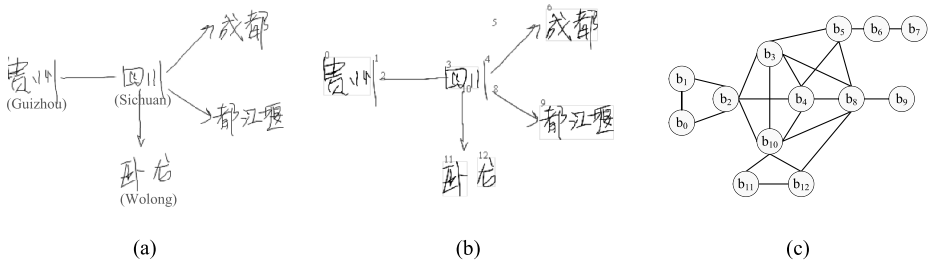


Fig. 3. Preprocessing result: (a) stroke blocks; (b) a graph to represent stroke blocks and their relationships.

Figure 3(a) shows a hand-drawn concept map about some provinces and cities in China. The nodes were written in Chinese. (English translation is provided by authors to show the meaning of each node. Here, "Guizhou" and "Sichuan" are two neighboring provinces in Southwest China, and "Chengdu", "Dujiangyan", and "Wolong" are three cities in the Sichuan province.) Figure 3(b) shows 13 stroke blocks after premerging. Some of them are over-segmented, such as (0, 1), (3, 4), (6, 7), and (11, 12). The vertical stroke in stroke block 1 is recognized as a line stroke, but it is actually a stroke that is part of the character in stroke block 0. Similarly, stroke blocks 4 and 7 are not line strokes, and should be with stroke blocks 3 and 6, respectively. Stroke blocks 11 and 12 are recognized as two stroke blocks because of their distance, but they are actually two characters of one single concept context. Figure 3(c) is the graph G corresponding to Figure 3(b). In G, edges connected to node b_4 include (b_4, b_2), (b_4, b_3), (b_4, b_5), (b_4, b_8), and (b_4, b_{10}), while the stroke block b_4 is adjacent to blocks b_2, b_3, b_5, b_8, and b_{10} in Figure 3(b).

4.1.2. Graph Decomposition with Graph Partitioning. The second step of our algorithm is to partition the graph G into several subgraphs with a graph partitioning algorithm. The time efficiency of block extraction with dynamic programming is greatly influenced by the node number of G, so a divide-and-conquer strategy was adopted to first partition G into several smaller subgraphs and then dynamic programming was used to extract blocks in each subgraph. This graph partitioning technique has been widely used in such areas as VLSI design, transportation management, and data mining. Our algorithm is based on the graph partitioning method in hMETIS [Karypis and Kumar 1998]. Given a graph $G = (V, E)$, the time to partition G to two subgraphs is $O(|V|+|E|)$ [Karypis and Kumar 1998].

Figure 4 shows the flowchart of our graph partitioning.

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*.

4.1.3. 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'),* which is the ratio of stroke length to its bounding box's diagonal length; *distFactor(V')*, which is the inverse of the distance between constituent stroke blocks; and *contextFactor(V')*, which is related to

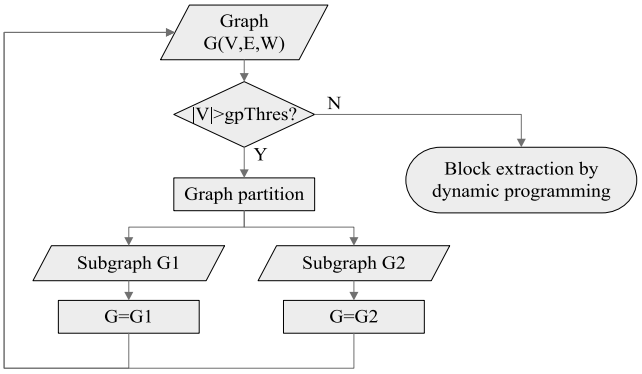


Fig. 4. The flowchart of graph partition.

the relationships between constituent stroke blocks. If V' is a candidate block in S ($V' \in S$), its reliability $R(V')$ can be computed by the following formula.

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

where a, and b are coefficients.

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

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

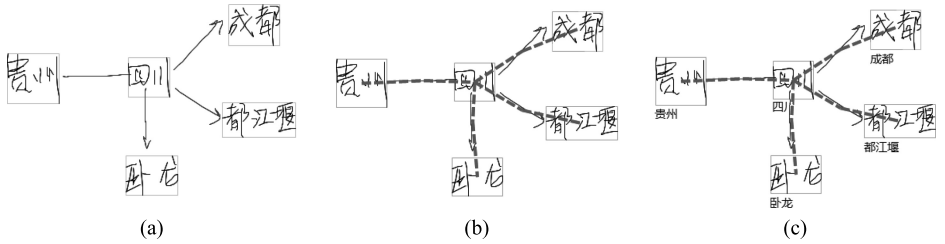
$C(V)$ is the reliability corresponding to the optimal block segmentation of V. It is the maximum value of the valid segmentations. $R(V)$ is the reliability of a candidate block V. $\phi(R(V'), C(V - V'))$ defines the strategy to combine subproblem V' and subproblem $V - V'$, and is written as

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

where $|\boldsymbol{V}|$ is the number of nodes in V.

When $|\boldsymbol{V}| \le \mathbf{1}$, there's no subproblem. When $|\boldsymbol{V}| > \mathbf{1}$ and V is in the candidate set S, the optimal reliability of V is the maximum value of V, or the maximum value of all combinations of V' and $V - V'$; otherwise, the optimal reliability is the maximum value of all combinations of V' and $V - V'$. A hash mechanism is used to store the optimal partition results of subproblems and their reliabilities.

4.1.4. 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 $v_i.type!=v_j.type$, that is, node blocks and link blocks are adjacent in a concept map. In



(a) (b) (c)

Fig. 5. The structure extraction and recognition results: (a) stroke blocks after dynamic programming; (b) relationships between nodes and links; (c) recognition result of concept nodes.

addition to the type attribute $v_i.type$, v_i has another attribute $v_i.text$ which indicates the recognition result of concept node v_i.

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

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. In this situation, only the block that is closest to the end of the link is considered. Following this approach, all node blocks can be tied to relevant link blocks. As a result, the structure of a concept map is extracted.

In addition, we extract parent-child relationships and brother relationships from the graph. These relationships can be determined by the types of link blocks people drew, as described in Section 3.

4.1.5. Concept Node Recognition. Concept maps with semantic meanings may be used for concept map searching. We recognize each concept node to get its corresponding text. For each node v_i with node type ($v_i.type$=*node*), our concept node recognition method extracts its attribute $v_i.text$. For a link node v_j ($v_j.type$=*link*), $v_j.text$ is set to empty string.

A concept node may have a bounding box or not. So the first step to recognize a concept node is to identify bounding strokes from handwritten texts in a node. The bounding strokes are those strokes by which text strokes are enclosed. In addition, the bounding strokes are often drawn at the beginning or at the end of the concept node. We distinguish bounding strokes from text strokes by utilizing the spatial and temporal relations between them. For handwritten texts, we used Microsoft Tablet PC Platform SDK[5] to recognize users' handwriting.

Figure 5 shows the effects of concept map structure extraction and node recognition. Figure 5(a) demonstrates stroke blocks extracted from two subproblems by dynamic programming. Figure 5(b) shows the relationships between nodes and links of the concept map with added thick dashed lines to connect nodes and links. Figure 5(c) illustrates the recognition results of concept nodes.

4.2. Hand-Drawn Concept Map Retrieval Algorithm

We also designed an algorithm to support concept-map retrieval based on a hand-drawn query. Our algorithm focuses on the structure and semantics of a sketched

[5]MICROSOFT TABLET PC PLATFORM SDK. http://www.microsoft.com/downloads/details.aspx?FamilyID=4B14B74A-27E4-42C4-862F-273F6302EA4F&displaylang=en&displaylang=en.

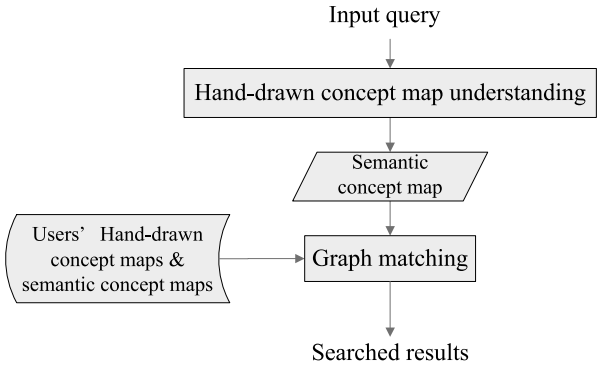


Fig. 6. Flowchart of hand-drawn concept map retrieval algorithm.

query and searches concept maps by using the extracted concept map from the hand-drawn query. The process of our hand-drawn concept map retrieval algorithm is shown in Figure 6.

As shown, in our algorithm, an input query is first understood to get its corresponding semantic concept map. Then, the graph matching algorithm is applied to retrieve concept maps that match the query.

4.2.1. Similarity between Concept Map Nodes. As the graph matching algorithm is based on the semantic concept maps, we need to define the similarity between concept nodes.

Given concept nodes a and b, the following formula defines whether these two nodes are the same.

$$same(a, b) = \begin{cases} true, & \text{if } (a.type == b.type) \wedge (sim(a.text, b.text) > simThres) \\ false, & \text{else} \end{cases}$$

When a and b have the same node type and the similarity between $a.text$ and $b.text$ is larger than $simThres$, they are considered to be the same.

The similarity between $a.text$ and $b.text$ is calculated by considering Levenshtein Distance (LD) [Levenshtein et al. 1966] and the longest common substring between them. As the recognition of users' handwriting could not be perfect, LD allows finding concept nodes with some character recognition errors. For example, users' handwriting of the word "flash" may be misrecognized as "flush". As the LD between "flash" and "flush" is small, we can still find "flash" with the misrecognized word. By adopting the longest common substring, searching based on a substring of a concept node is possible. For example, "bread" could be used to search "bread and milk".

$$sim(a.text, b.text) = \max\left(\frac{1}{\alpha + \beta * edit\_distance(a.text, b.text)}, \frac{commonSubNum(a.text, b.text)}{\min\left(len(a.text), len(b.text)\right)}\right)$$

4.2.2. Graph Matching Algorithm. When a user provides a concept map sketch for search, our algorithm intends to find all the similar concept maps and subconcept maps. Thus, the retrieval of hand-drawn concept maps can be considered as a graph-subgraph isomorphism problem. Moreover, as the hand-drawn concept map consists of nodes and links that contain attributes, it can be seen as an attributed relational graph. The graph-matching algorithm VF2 proposed by Cordella et al. [2001] could be used to

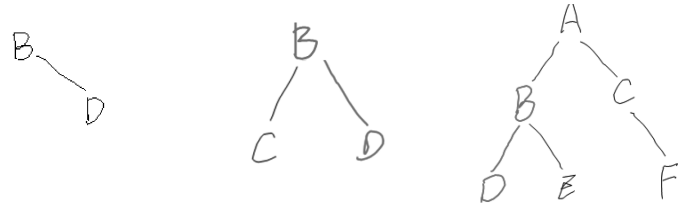


Fig. 7. The hand-drawn concept map for searching (a) and the corresponding searched results (b-c).

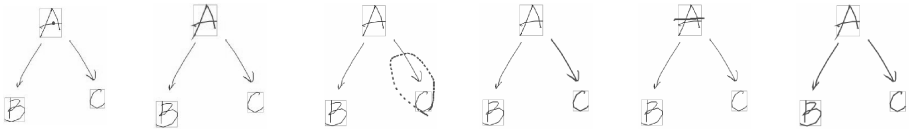


Fig. 8. Three kinds of selection techniques.

solve the graph-subgraph isomorphism problem on attributed relational graphs and could work with very large graphs.

Our retrieval algorithm is based on the VF2 algorithm and our implementation is based on the graph-matching library VFLib[6]. Suppose $G1$ is the graph corresponding to the input query and $G2$ is a graph corresponding to a hand-drawn concept map in the storage, our graph-matching algorithm tries to find a mapping M between $G1$ and a subgraph of $G2$.

Figure 7 illustrates how our retrieval algorithm works. Figure 7(a) is a hand-drawn query for concept maps. Figure 7(b) and Figure 7(c) are the search results and the highlighted strokes correspond to the query input.

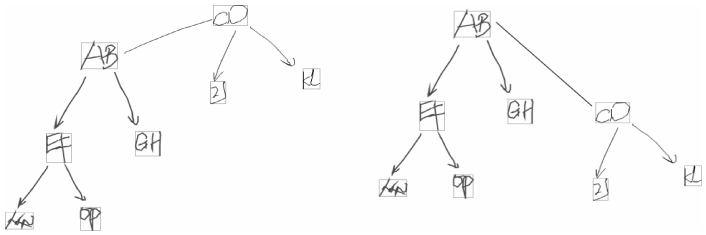
5. INTELLIGENT MANIPULATION OF CONCEPT MAPS

Based on the hand-drawn concept map understanding algorithm and retrieval algorithm, we design structure-based manipulation techniques and ink-based retrieval techniques for hand-drawn concept maps.

As concept maps are often used for creating and refining knowledge representations, people need to modify concept maps frequently. Therefore, natural and efficient manipulation techniques are very important to a concept map drawing tool. To support natural and efficient manipulation of hand-drawn concept maps, we developed a set of pen gestures that allow users to handle concept maps by directly acting on extracted structures.

Selection is essential for editing concept maps. Operations like scaling and translating are targeted for selected objects. With structures extracted from hand-drawn concept maps, structuralized selection becomes feasible. We designed three pen gestures to select stroke blocks at different levels. Figure 8 demonstrates these gestures. A user can select components in a concept map in three different ways: (1) tapping a stroke block to select the block, either 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

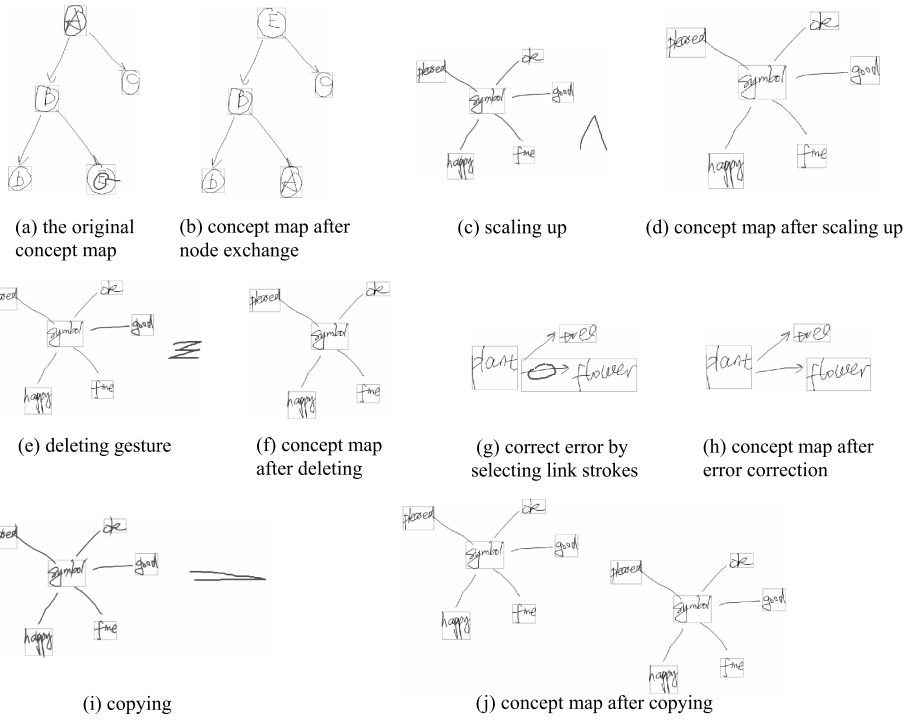
[6]VFLib. http://amalfi.dis.unina.it/graph/db/vflib-2.0/doc/vflib.html.



(a) selected stroke blocks

(b) concept map after translation

Fig. 9. Translating part of a concept map without changing its structure.



(a) the original concept map

(b) concept map after node exchange

(c) scaling up

(d) concept map after scaling up

(e) deleting gesture

(f) concept map after deleting

(g) correct error by selecting link strokes

(h) concept map after error correction

(i) copying

(j) concept map after copying

Fig. 10. Examples of pen gestures for manipulating hand-drawn concept maps.

method is a semantic-based technique that considers the relationships among node blocks.

This structure-based selection can simplify certain manipulation tasks. For example, a user can easily move part of a concept map part around while altering its structure as shown in Figure 9.

In addition, other pen gestures were designed for manipulating concept maps (Figure 10). The user can exchange concept nodes (Figure 10(a) and 10(b)), scale a concept map (Figure 10(c) and 10(d)), delete part of a concept map (Figure 10(e) and 10(f)), correct errors (Figure 10(g) and 10(h)), and copy a concept map (Figure 10(i) and 10(j)). These gestures are similar to those designs in systems like MindManager.

We also implemented some simple gesture tools that allow users to correct drawing errors (e.g., erasing sketches, redrawing sketches). It should be noted that error

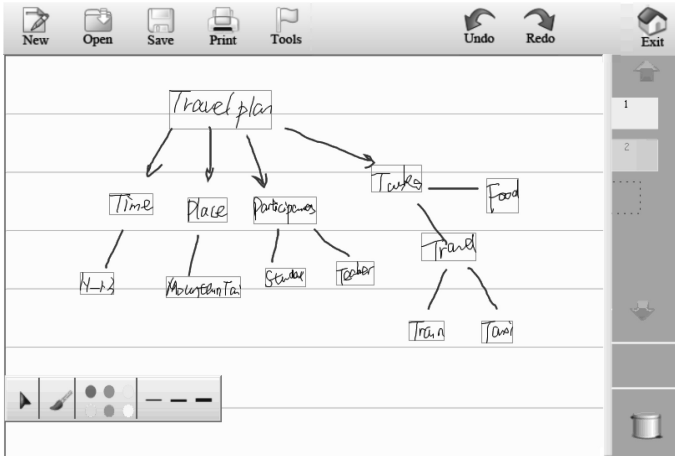


Fig. 11. The user interface of ConceptSketch.

correction techniques have been studied by many researchers [Ao et al. 2007; Wang et al. 2006] and are not the focus on this research. Our implementation of error correction tools aims to improve the effectiveness of our system.

6. EVALUATION

Based on our concept map understanding algorithm and the proposed intelligent manipulation and searching techniques, we designed a pen-based concept map drawing tool, called ConceptSketch (Figure 11), to support drawing, manipulating, and searching concept maps.

With the prototype, we conducted a study to evaluate the accuracy of concept map understanding, and the usability of our intelligent manipulation techniques. We also collected user feedback on the use of our prototype.

6.1. Accuracy of Concept Map Understanding

Our evaluation on algorithm accuracy focused on the accuracy of understanding hand-drawn concept maps. We did not examine the accuracy of concept map retrieval separately. This is because among three factors involved in the retrieval accuracy, that is, structure extraction accuracy, concept node recognition accuracy, and graph-matching accuracy, the first factor is the accuracy of understanding the concept map and the other two are determined by the performances of the Microsoft Platform SDK and VFLib, respectively, which are beyond the scope of this research.

To evaluate structure understanding accuracy, we focused on the effectiveness and efficiency of the algorithm. Our test was on a machine equipped with a 2.4 GH CPU, 2G memory and a Wacom screen. Test data consisted of 45 hand-drawn concept maps from ten student subjects. As for layout styles, 15 concept maps had a top-down style, 15 had a left-right style, and 15 had a center-out style. Twenty five concept maps had tree structure while the others had network structure. About one-third of concept maps had different node and link styles. The numbers of links and nodes in a concept map ranged from 10 to 40 and the average was 21.1 (SD = 7.2).

Concept map structure understanding accuracy consists of two parts: stroke block extraction accuracy and structure extraction accuracy. Figure 12 shows the block extraction results of 45 concept maps. The average error rate for block extraction is 4.82% (SD = 0.07). The median of the error rate is 2.78%. Most of these errors

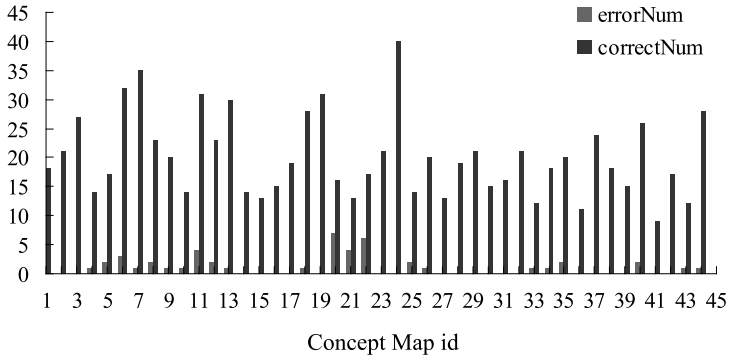


Fig. 12. Accuracy of extracted blocks.

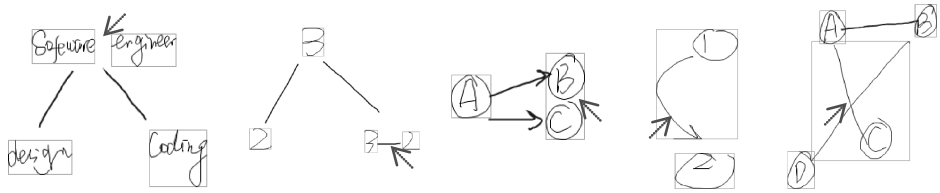


Fig. 13. Some examples where the algorithm does not work well.

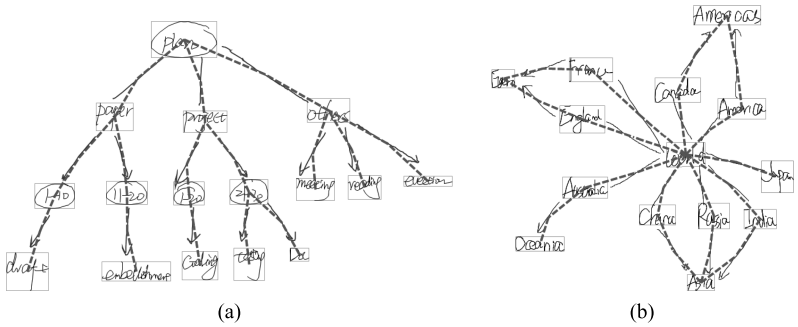


Fig. 14. Examples of concept map structure understanding.

are over-segmentation errors, that is, strokes that should belong to one block are segmented into several blocks. Under-segmentation errors, that is, strokes that should belong to different blocks are grouped into one block, were also observed. Two factors were found to contribute to over-segmentation errors: distant strokes, as shown in Figure 13(a), and misrecognized link strokes in Figure 13(b). Under-segmentation errors are largely due to the closeness of node blocks, shown in Figure 13(c) and, again, misrecognized link blocks in Figure 13(d) and 13(e).

When node blocks and link blocks are correctly extracted, structure extraction accuracy is high under our algorithm at 99.5%. The errors are mainly caused by the incompleteness of concept maps themselves.

Figure 14 shows extracted structures from two hand-drawn concepts maps. As seen, our algorithm can correctly generate concept map structures despite variations on node style, link style, and concept map layout.

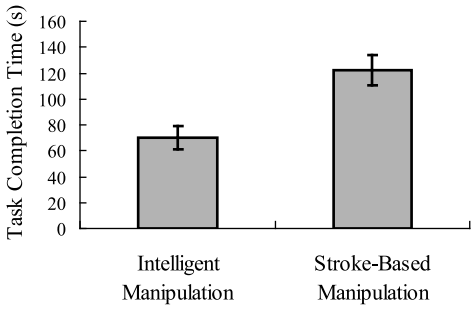


Fig. 15. Efficiency comparison of intelligent manipulation and stroke-based manipulation.

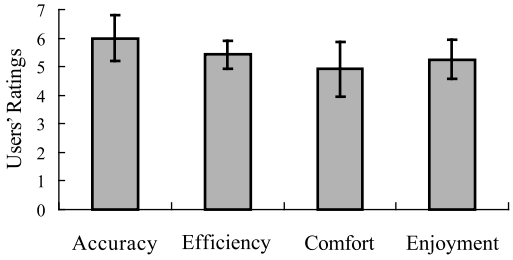


Fig. 16. Users' ratings.

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

6.2. Usability of Manipulation Techniques

We also compared our intelligent manipulation techniques with stroke-based manipulation techniques. The participants were asked to change a given concept map to a targeted one. Two concept maps were used in the experiment. Six subjects were asked to modify the first concept map with our intelligent manipulation techniques and then to change the second concept map with a stroke-based manipulation technique. The other six users used the stroke-based method to transform the first concept map and intelligent manipulation techniques for the second one. We collected each subject's task completion time.

Figure 15 shows the average task completion times with two manipulation techniques. Our structure-based intelligent manipulation is significantly shorter than the stroke-based manipulation technique ($p < .001$).

6.3. User Feedback

We also asked each subject to answer a post-test questionnaire to grade the accuracy, efficiency, comfort, and enjoyment of intelligent manipulation and retrieval techniques, all in a 7-level Likert scale (1-very bad, 7-very good). Subjects could also provide comments and suggestions in an open-ended question.

Figure 16 exhibits the results of subjective evaluation about the intelligent manipulation technique and retrieval technique. As shown, our techniques were well received by subjects.

Users also gave us some useful comments on our work. Some users suggested that the system support automatic layout of hand-drawn concept maps. Some pointed out that the system should handle more complex concept maps, such as concept maps with

curved or multistroke links. Some mentioned that it would be useful if the system could support more functions for concept maps, such as the comparison or the merging of concept maps.

7. DISCUSSION AND CONCLUSION

This article presented an approach to understand and retrieve hand-drawn concept maps. By combining dynamic programming and graph partitioning, our algorithm extracts node blocks and link blocks of hand-drawn concept maps and builds a concept-map structure by relating nodes and links. Meanwhile, our algorithm supports retrieval of hand-drawn concept maps based on graph matching. We implemented a system prototype, ConceptSketch, and integrated a set of structure-based intelligent manipulation techniques and a concept map retrieval method based on sketch query. The manipulation techniques allow users to manipulate concept maps by directly handling object blocks, rather than fragmental strokes. Our evaluation results show that our algorithm is effective and our structure-based manipulation techniques and retrieval technique are effective and welcomed by users.

The contributions of our research are twofold. First, our algorithm can greatly enhance hand-drawn concept map tools and make such tools more intuitive and natural for users to create concept maps and diagrams. Our algorithm allows the construction of more advanced tools to manipulate concept maps. For example, by extracting structures of hand-drawn concept maps with our algorithm, such operations as reorganizing a concept map, sorting the concepts involved in a concept map, and exporting a concept to other formats become possible. Furthermore, with extracted meanings of nodes, additional metadata on the nodes can be added to the concept map to make the concept map more comprehensive. For example, a search tool can be built so that users can take the extracted meaning of a node as a query to search through a database to identify other relevant concepts or to find related media (e.g., pictures or videos) to elaborate the node.

Second, our structure-oriented method can expand the scope of application of hand-drawn tools into other areas in which hand-drawn structures are important. Although our structure understanding algorithm is designed for concept maps, it can also be extended to recognize sketches in other domains, such as chemistry, math, physics, and engineering (computer-aided design), 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.

Our research has some limits. First, our algorithm still produces some errors (less than 5%) in understanding concept maps. These errors, although very small, are largely related to block extraction, that is, over- and under-segmentation of blocks due to far- and short-distant stroke blocks, as shown in Figure 13(a) and 13(c). Two kinds of strategies could be used to address these errors. On the one hand, more robust algorithms should be explored to reduce recognition errors. For example, our graph partitioning method may produce a structure that does not meet a user's intent, although this rarely happens in our study. It would probably be solved by developing a graph partitioning algorithm specialized for concept maps. On the other hand, we could let users help the system understand concept maps. By suggesting users to draw in-block strokes closely and between-block strokes distantly, we expect higher recognition accuracy could be achieved. Second, our algorithm cannot handle more complicated link styles, such as curved links shown in Figure 13(d), links with labels, and crossed links in Figure 13(e). To address this problem, more advanced algorithms are needed to identify curve sketches, to distinguish labels from links and from other nodes, and to separate crossed links.

We will extend our research in several directions. First, we will enhance our algorithm by addressing the limitations discussed previously, in particular supporting diverse link types and recognizing link labels. Second, we will integrate users' drawing habits in our algorithm to improve the accuracy of concept map understanding. Furthermore, we will explore structure-based interaction techniques, such as automatic layout, to help people better organize and manage hand-drawn concept maps. In addition, our concept map retrieval technique is effective to search a hand-drawn concept map by using a subconcept map, but the search may fail when the mapping between the input query and a hand-drawn concept map does not satisfy the edge-preserving constraint. We will try to use inexact graph-matching techniques in concept map retrieval algorithm.

APPENDIX

Concept Map Structure Understanding Algorithm

Step 1. Preprocessing: merging strokes to get blocks and getting the graph

```
PREPROCESSING(strokes)
        foreach stroke si in strokes do
            block ← {si}
            APPEND(blocks, block)
         foreach block bi in blocks do
            foreach block bj in blocks do
                // bi.type is obtained by $1 recognizer
                if DIST(bi,bj)<thres, bi.type!=LINK and bj.type!=LINK then
                        b′ ← MERGE(bi,bj)
                        INSERT(blocks, i, b′)
                        REMOVE(blocks, bi)
                        REMOVE(blocks, bj)
        foreach block bi in blocks do
            ADDNODE(graph, bi)
        foreach block bi in blocks do
            foreach block bj in blocks do
                weight ← DIST(bi, bj)
                if weight<weightThres then
                        ADDEDGE(graph, bi, bj, weight)
        return graph
```

Step 2. Graph decomposition with graph partitioning

```
DECOMPOSITION(graph)
        if NUM_NODE(graph)< thres then //Here we set thres=8
            APPEND(subGraphs, graph)
        else
            tempGraphs = PARTITION-BY-SHMETIS(graph)
```

```
        foreach graph gi in tempGraphs do
            tempSubGraphs = DECOMPOSITION(gi)
            APPEND(subGraphs, tempSubGraphs)
    return subGraphs
```

Step 3. Block extraction by dynamic programming

```
BLOCK-EXTRACTION(graph)
    candidates ← GENERATE-CANDIDATES(graph)
    V ← GETNODES(graph)
    EXTRACT-RECURSIVE(candidates, optimalPath, V)
    Get newBlocks according to optimalPath
    return newBlocks

GENERATE-CANDIDATES (graph)
    foreach node vi in graph do
        b ← vi //vi corresponds to a block in graph
        b.value ← a*DENSITYFACTOR(b)+b*DISTFACTOR(b)+
        c*CONTEXTFACTOR(b)
        APPEND(candidates, b)
        foreach node vj in graph do
            if !HASEDGE(vi,vj) then continue
            b′ ← MERGE(vi,vj)
            b′.value ← a*DISTFACTOR(b′)+b*DENSITYFACTOR(b′)+
            c*CONTEXTFACTOR(b′)
            APPEND(candidates, b′)
            foreach node vk in graph do
                if !HASEDGE(vi,vk) and !HASEDGE(vj,vk) then continue
                b″ ← MERGE(vi,vj,vk)
                b″.value ← a*DENSITYFACTOR(b″)+b*DISTFACTOR(b″)+
                c*CONTEXTFACTOR (b″)
                APPEND (candidates, b″)
    return candidates

EXTRACT-RECURSIVE(candidates, optimalPath,V)
    if NUM(candidates)<1 then
       return 0
    maxValue ← 0
    foreach candidate V′ in candidates do
```

```
        value1 ← V′.value
        subPath1 ← {i|bi belongs to V′}
        if (V−V′). hasCalculated then
            value2 ← (V−V′).value
            subPath2 ← (V−V′).path
        else
            tempCandidates ← {candidates without any blocks in V′}
            value2 ← EXTRACT-RECURSIVE (tempCandidates, subPath2,
            V−V′)
            (V−V′).value ← value2
            (V−V′).path ← subPath2
        value ← (|V′|*value1+ |V−V′|*value2)/|V|
        if (maxValue<value) then
            maxValue ← value
            APPEND(subPath2, subPath1)
            optimalPath ← subPath2
            V.hasCalculated ← true
    return maxValue
```

Step 4. Concept map structure extraction

```
EXTRACT-STRUCTURE(blocks, graph)
    foreach block bi in blocks do
        ADDNODE(graph, bi)
    foreach block bi in blocks do
        foreach block bj in blocks do
            if bi.type is LINK, bj.type is NODE, and DIST(bi,bj)<thres then
                ADDEDGE(graph, bi, bj)
```

Step 5. Recognizing concept nodes

```
RECOGNIZE-CONCEPTNODES(graph)
    foreach node vi in graph do
        if vi.type is LINK then
          vi.text ← ""
        else
          textStrokes ← GETTEXTSTROKES(vi)
          vi.text ← RECOGNIZE(textStrokes) // using Microsoft Tablet
          PC Platform SDK
```

REFERENCES

ALVARADO, C. 2004. Multi-Domain sketch understanding. Ph.D. thesis. Massachusetts Institute of Technology, Cambridge, MA.

AO, X., LI, J. F., WANG, X. G., AND DAI, G. Z. 2006. Structuralizing digital ink for efficient selection. In *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI'06)*. ACM Press, 148–154.

AO, X., WANG, X., TIAN, F., DAI, G. Z., AND WANG, H. 2007. Crossmodal error correction of continuous handwriting recognition by speech. In *Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI'07)*. ACM, New York, 243–250.

CHAN, K. F. AND YEUNG, D. Y. 2000. Mathematical expression recognition: A survey. *Int. J. Document Anal. Recogn. 3*, 1, 3–15.

CHIK, V., PLIMMER, B., AND HOSKING, J. 2007. Intelligent mind-mapping. In *Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces (OzCHI'07)*. Vol. 251. ACM, New York, 195–198.

COFFEY, J. W., HOFFMAN, R. R., CAÑAS, A. J., AND FORD, K. M. 2002. A concept map-based knowledge modeling approach to expert knowledge sharing. In *Proceedings of the IASTED International Conference on Information and Knowledge Sharing*. 212–217.

CONTE, D., FOGGIA, P., SANSONE, C., AND VENTO, M. 2004. Thirty years of graph matching in pattern recognition. *Int. J. Patt. Recogn. Artif. Intell. 18*, 3, 265–298.

CORDELLA, L. P., FOGGIA, P., SANSONE, C., AND VENTO, M. 2001. An improved algorithm for matching large graphs. In *Proceedings of 3rd IAPR-TC15 Workshop Graph-Based Representations in Pattern Recognition*. 149–159.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms* 2nd Ed. MIT Press, Cambridge, MA.

FORBUS, K. D. AND USHER, J. M. 2002. Sketching for knowledge capture: A progress report. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI'02)*. ACM, New York, 71–77.

GENNARI, L., KARA, L. B., AND STAHOVICH, T. F. 2005. Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Comput. Graph. 29*, 4, 547–562.

GROSS, M. D. 1994. Stretch-A-Sketch: A dynamic diagrammer. In *Proceedings of the IEEE Symposium on Visual Languages*. IEEE Computer Society Press, 232–238.

HALIMI, S. 2006. The concept map as a cognitive tool for specialized information recall. In *Proceedings of the 2nd International Conference on Concept Mapping*. 81–88.

HAMMOND, T. AND DAVIS, R. 2002. Tahuti: A geometrical sketch recognition system for UML class diagrams. In *Proceedings of AAAI Spring Symposium on Sketch Understanding*. 59–66.

HINCKLEY, K., ZHAO, S. D., SARIN, R., BAUDISCH, P., CUTRELL, E., SHILMAN, M., AND TAN, D. 2007. InkSeine: In situ search for active note taking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM, New York, 251–260.

JAIN, A. K. AND NAMBOODIRI, A. M. 2003. Indexing and retrieval of on-line handwritten documents. In *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'03)*. IEEE Computer Society Press, Los Alamitos, CA, 655–659.

JAWAHAR, C. V., BALASUBRAMANIAN, A., MESHESHA, M., AND NAMBOODIRI, A. M. 2009. Retrieval of online handwriting by synthesis and matching. *Patt. Recogn. 42*, 1445–1457.

JIANG, Y. Y., TIAN, F., WANG, X. G., ZHANG, X. L., DAI, G. Z., AND WANG, H. A. 2009. Structuring and manipulating hand-drawn concept maps. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI'09)*. ACM, New York, 457–462.

JIANG, Y. Y., TIAN, F., WANG, H. A., ZHANG, X. L., WANG, X. G., AND DAI, G. Z. 2010. Intelligent understanding of hand-drawn geometry theorem proving. In *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI'10)*. ACM, New York, 119–128.

KARA, L. B. AND STAHOVICH, T. F. 2004. Hierarchical parsing and recognition of hand-sketched diagrams. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST'04)*. ACM, New York, 13–22.

KARYPIS, G. AND KUMAR, V. 1998. hMETIS: A hypergraph partitioning package Version 1.5.3. http://glaros.dtc.umn.edu/gkhome/views/metis/hmetis/download.html.

LEUNG, W. H. 2003. Representations, feature extraction, matching and relevance feedback for sketch retrieval. Ph.D. thesis. Carnegie Mellon University Pittsburgh, PA.

LEVENSHTEIN, I. V. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernet. Control Theory 10*, 8, 707–710.

LI, Y., GUAN, Z. W., WANG, H. A., DAI, G. Z., AND REN, X. S. 2002. Structuralizing freeform notes by implicit sketch understanding, In *Proceedings of the AAAI Spring Symposium on Sketch Understanding*. AAAI Press, 91–98.

LIANG, S. AND SUN, Z. X. 2008. Sketch retrieval and relevance feedback with biased SVM classification. *Patt. Recogn. Lett. 29*, 1733–1741.

MORAN, T. P., CHIU, P., MELLE, W., AND KURTENBACH, G. 1995. Implicit structures for pen-based systems within a freeform interaction paradigm. In *Proceedings of the CHI Conference*. ACM, New York. 487–494.

NOVAK, J. D. AND GOWIN, D. B. 1984. *Learning How to Learn.* Cambridge University Press, Cambridge, UK.

NOVAK, J. D. 1998. *Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations.* Lawrence Erlbaum Associates, Mahwah, NJ.

NOVAK, J. D. AND CAÑAS, A. J. 2008. The theory underlying concept maps and how to construct them. Tech. rep. IHMC CmapTools 2006-01 Rev 01-2008. Florida Institute for Human and Machine Cognition.

OUYANG, T. Y. AND DAVIS, R. 2007. Recognition of hand-drawn chemical diagrams. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI'07)*. AAAI Press, 846–851.

SAUND, E., FLEET, D., LARNER, D., AND MAHONEY, J. 2003. Perceptually-supported image editing of text and graphics. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST'03)*. ACM, New York, 183–192.

SHILMAN, M., PASULA, H., RUSSELL, S., AND NEWTON, R. 2002. Statistical visual language models for ink parsing. In *Proceedings of the AAAI Spring Symposium on Sketch Understanding.* AAAI Press, 126–132.

SHILMAN, M., VIOLA, P., AND CHELLAPILLA, K. 2004. Recognition and grouping of handwritten text in diagrams and equations. In *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR'04)*. IEEE Computer Society Press, 569–574.

SZUMMER, M. AND COWANS, P. 2004. Incorporating context and user feedback in pen-based interfaces. In *Proceeding of the AAAI Fall Workshop on Making Pen-Based Interaction Intelligent and Natural.* AAAI Press, 159–166.

TERGAN, S. O. 2005. Digital concept maps for managing knowledge and information. In *Knowledge and Information Visualization*. Lecture Notes in Computer Science, vol. 3426, Springer, 185–204.

WANG, X. G., LI, J. F., AO, X., WANG, G., AND DAI, G. Z. 2006. Multimodal error correction for continuous handwriting recognition in pen-based user interfaces, In *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI'06)*. ACM, New York, 324–326.

WOBBROCK, J. O., WILSON, A. D., AND LI, Y. 2007. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST'07)*. ACM, New York, 159–168.

ZHOU, M. X., WEN, Z., AND AGGARWAL, V. 2005. A graph-matching approach to dynamic media allocation in intelligent multimedia interfaces. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI'05)*. ACM, New York, 114–121.

Received January 2011; accepted March 2011