Informal User Interface for Graphical Computing

Zhengxing Sun and Jing Liu

State Key Lab for Novel Software Technology, Nanjing University, 210093, China
szx@nju.edu.cn

Abstract. This paper explores a concept of sketch-based informal user interface for graphic computing, which can be characterized by two properties: stroke-based input and perceptual processing of strokes. A sketch-based graphics input prototype system designed for creative brainstorming in conceptual design is introduced. Two core technologies for implementing such a system, adaptive sketch recognition and dynamic user modeling, are also outlined.

1 Introduction

For over three decades, the graphical user interfaces (GUI) and its associated desktop metaphors have dominated both the marketplace and HCI research. As computers changes in terms of physical size, capacity, usage and ubiquity, peoples interact with them in more informal ways than they used to. The obvious question arises, such as what is the next major generation in the evolution of user interfaces and if there is a paradigm (and its associated technology) that will displace GUI and become the dominant user interface model. There is no shortage of HCI researches collectively called as post-WIMP interfaces [1] or non-command interfaces [2], such as various flavors of immersive environments (virtual, augmented, and mixed reality), tangible interfaces, haptic interfaces and so on. The common goal of them is to make computers more intelligent, more convenient to use, more adaptable to the human-preferred communication mode, and to allow the user to concentrate on the task itself without worrying about commands.

In the domain of graphical computing, people are accustomed to write down their improvisatory ideas. For them, the ability to rapidly deliver their ideas using graphic objects with uncertain types, indefinite sizes, irregular shapes, and inaccurate positions is most important. However, most current drafting tools are formal and computer-oriented, which requires users to select graphic patterns from lots of toolbar buttons or menu items and does not work well for expressing arbitrary graphical ideas or geometric shapes in computers. Users frequently find it inconvenient via too many mouse-clicks. They also complain that they have to memorize the precise position of each toolbar button or menu item since they cannot focus on the design idea itself when utilizing these tools to deliver their bursting creative ideas. Therefore, they cannot finish the design fluently due to too many interruptions.

In this paper, we will explore an informal user interface (IUI) paradigm using sketching for graphical computing and illustrate our experiments on this topic. The remainder of this paper is organized as follows. In section 2, the concept of the sketch-based informal user interface is defined with two properties: stroke-based

J. Tao, T. Tan, and R.W. Picard (Eds.): ACII 2005, LNCS 3784, pp. 675 – 682, 2005.

input and perceptual processing, and some related works are outlined. Section 3 briefly introduces our prototype system, named as Magic-Sketch, embodying the idea of informal user interface. Section 4 discusses two core techniques for implementing informal user interface. Conclusions and future works are given in the final Section.

2 Sketch-Based Informal User Interfaces for Graphic Computing

People have been using pen and paper to express graphical ideas for centuries, which is a preferred choice for creative brainstorming [3]. Even in this high-tech computer era, paper and pencils have still to be designers' preferred choice to quickly sketch bursting ideas. It can help user convey ideas and guide our thought process both by aiding short-term memory and by helping to make abstract problems more concrete. Sketch-based user interface is an informal input modality of increasing interest for human-computer interaction. It embodies a non-command user interface for the graphical applications in that the user can transfer visual ideas into target computers without converting the ideas into a sequence of tedious command operations. The term "informal" is referred to that sketch-based interfaces are tolerant of the user's input and show variability in their output. The tolerance means allowable differences in input function mapping to an internal representation. The variability means that an internal representation can be mapped in a number of ways to an output mechanism without appearing to have a different meaning.

The sketchy shape in its rough state contains more information than the regularized one. But its ambiguity and uncertainty make the deduction of intents very difficult. The regularized shape is better for users to communicate and recall their original intention of this sketch in graphical applications. It will be more helpful for graphic computing if the sketchy shape can be recognized and converted into the user-intended regular shape. Therefore, the manner of sketch-based graphic input is that user can draw their approximate line shapes with pen-based stroking quickly and fluently, while computer must recognize/covert user's inputting strokes to the regular shapes immediately.

Pen-based stroking is usually recognized as a dragging operation in a standard programming environment: it is initiated by "button press" event, followed by a sequence of "mouse move" event, and terminated by "button release" event. The system's reaction is based on the entire trajectory of the pen's movement during the stroking, not just the pen's position at the end. In stroking, the user first imagines the desired stroke shape and then draws the shape on the screen at once, while the user constantly adjusts the cursor position observing the feedback objects during dragging. Through stroke seems to be a primitive unit in sketch user interface at first appearance, the stroke is not a structural and unique constitutive geometric primitive of a shape for human cognition. Therefore, stroke segmentation is the groundwork for realizing sketch-based IUI, which decomposes the inputting strokes into basic geometric primitives, such as lines and curves. As strokes can be segmented in many different ways, the challenge of stroke segmentation is to find out which bumps and bends are intended and which are accident. Sezgin [4] have used both curvature and speed information in a stroke to locate breakpoints, while Saund [5] used more perceptual context, including local features such as curvature and intersections, as

well as global features such as closed paths. All of them use empirical thresholds to test the validity of an approximation that ultimately leads to the problem of a threshold being too tight or too loose.

The another important property is its advanced processing of strokes inspired by human perception, which characterizes sketch-based IUI as a non-command user interface and makes sketch-based IUI different from plain pen-based scribbling systems that simply convert the user's pen movement into a painted stroke on the screen without any further processing. We call this advanced processing as "perceptual processing" or "sketch recognition". The idea behind sketch recognition is inspired by the observation that human beings perceive rich information in simple drawings, such as possible geometric relations among line primitives, three-dimensional shapes from two-dimensional silhouettes. Sketch recognition is an attempt to simulate human perception at least in limited domains. The goal of sketch recognition is to allow the user to perform complicated tasks with a minimum amount of explicit control. Sketch-based IUI must free users from detailed command operations by this perceptual processing of freeform strokes and reduces significantly the effort spent on learning commands. A variety of sketch recognition techniques have been proposed, which can be classified into three categories: feature-based methods [6][7], graph-based methods [8] and machine learning methods [9][10]. In addition, several experimental systems for supporting sketch-based informal user interface in limited domain, such as Sim-U-Sketch for mechanical design and simulation [11], DENIM for the early stages of web site design [12], and so on. In summary, while there has been significant progress in sketch recognition, the poor efficiency of the recognition engines is always frustrating, especially for complex sketchy shapes and newly added users. The main challenge in sketch recognition is that a recognizer should be adaptable to a particular user's sketching styles. More importantly, most symbol recognizers do stroke fragmentation and symbol recognition separately. This would apparently result in aimless segmentation of strokes and incorrect recognition of symbols deviating from users' intentions.

3 Magic-Sketch: A Sketch-Based Platform for Graphic Input

3.1 Overview of Magic-Sketch

We have being developed a prototype platform of sketch-based graphic input for conceptual design to support users' creativities, named Magic-Sketch. The framework of Magic-Sketch is outlined in Fig. 1. It is mainly consisted of following components: stroke pre-processing, stroke segmentation and sketch recognition, dynamic user modeling, database management, input and edit interface, and application interfaces.

As a basis of sketch-based graphic input, the *stroke pre-processing* is firstly adopted to eliminate the noise that may come from restriction of input condition or habits [13], such as redundant points reducing, agglomerate points filtering and end points refinement. The candidate breakpoints of strokes are also distinguished, where the pen speed is at a minimum; the ink exhibits high curvature, or the sign of the curvature changes besides the start and end point of each stroke. The *stroke segmentation* and *Sketch recognition* are then used to decompose each stroke into some kinds of primitives and recognize each of individual geometric shapes such

as glyphs and symbols and their relationships in the inputting pattern respectively. Several gesture commands are also recognized. To adapt for the arbitrariness and amphibology of inputting, we propose a novel method of sketch recognition, which integrates stroke segmentation with sketch recognition. This will be discussed in next subsection. *Dynamic user modeling* is designed to build user models for each specific user to capture users' habit of drawing styles and to facilitate the sketch recognition in an incremental manner. This will be discussed in the subsection 3.3. In addition, user model can also been updated by user mediation based on relevance feedback techniques, where user can refine/correct the recognition results by interactive feedback based on partial and overall structural similarity between inputting drawing and templates [13].

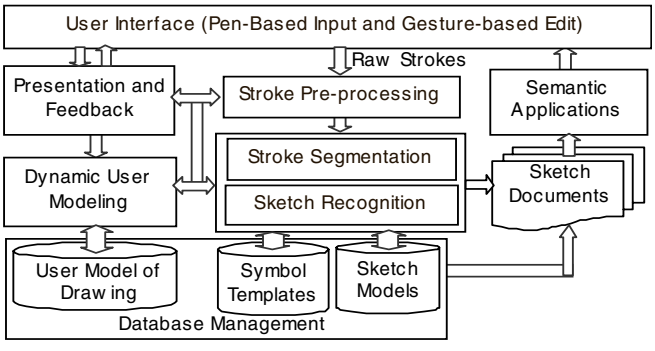

Fig. 1. Framework of Sketch-based Graphics Input Tool

In order to preserve and manage the information during user drawing, two types of data model are designed in our prototype system as shown in Fig. 2. Fig. 2(a) shows the hierarchical structure of Sketch model, which includes backboard, raw strokes, primitives, shapes and semantics from the bottom up. Backboard is a host structural format that sketches are located in. We can use HTML or XHTML as a backboard to define a *sketch document* for sketch model. This will make sketch model more powerful and portable for different domains. Raw stroke refers to ink points that are sampled by input equipment. Primitives and shapes are the geometric and relation information of the tokens extracted from raw strokes. Sketch semantics refers to recognized symbols related to applied domain. User model is also organized in a layered structure, as shown in Fig. 2(b). For a specific user, besides the identifier of user, some of his/her drawing properties of ink points, strokes, symbols and applied domain are defined, such as pen pressure and drawing speed at each of ink points, the temporal sequence of strokes, frequency of intended symbol and so on.

We also design a particular *user interface* to support the freedom and fluency of pen-based drawing and editing. We offer 9 gestures to users, including copying, deleting, dragging, pasting, undo, redo, cleaning panel, finishing and selecting. The interactive editor is provided with the manager of the document. It provides an interactive and visualized interface for document editing. If the input strokes are

identified as a visual token, then the new token will be added to the document, together with corresponding modifications of their spatial relations. If the stroke is recognized as a gesture commands, the editor processes them directly, just as common document editors. User interactions are saved in for feedback. The user feedback style indicates two styles. One is the time that the system submits its result. The other is the granularity that the result is presented in. This means that the form of result, which is shown to user, can be different, from regularized strokes to the whole sketch after recognition.

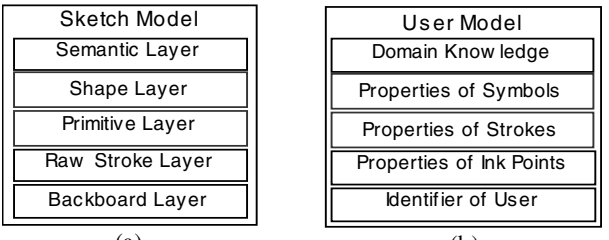

Fig. 2. Conceptual model of data structure in Magic-Sketch

3.2 Adaptive Sketch Recognition Based on Templates

Most of the existing methods treat stroke segmentation and sketch recognition separately. In fact, a user is purposeful with the intended symbol in head when expressing his/her ideas with the particular sketchy shape based on both the current observations and the past experiences, though he/she draws one stroke after another. This is the primary reason for the poor accuracy of recognizers no matter how robust they might. Therefore, it is necessary for sketch recognition to integrate the process of stroke segmentation and sketch recognition, in order to account for the variations inherent in hand-drawn sketches.

To achieve this goal, we propose a novel approach of adaptive sketch recognition by regarding both of stroke segmentation and sketch recognition as a problem of "fitting to a template" with a minimal fitting error between input patterns and the particular domain definition models of the symbol (templates). Fig. 3 shows the flowchart of our strategy, which is designed to work for single isolated symbols. Examples include symbols in conceptual design, analogy electric circuits design, data flow diagrams, algorithmic flowcharts and so on.

In our strategy, stroke segmentation optimizes the combination of breakpoints by calculating the similarity between the primitives of inputting pattern and that of the templates based on the selection of candidate breakpoints in stroke pre-processing. This makes stroke segmentation be well guided by the templates.

Given a sketchy symbol *SM* and a template *T*, *SM* is consisted of a sequence of strokes, each stroke contains a set of the ordered candidate breakpoints; a template *T* is represented as a set of ordered primitives $T\{t(i)\}$, the number of breakpoints needed to be identified is: $k=NT-NS$(in general, $NS \le NT \le NB-1$, where, *NT* is the number of primitives for defining a template of symbol, *NS* is the number of strokes and *NB* is the total numbers of ordered candidate breakpoints (NB_i is the number of ordered

candidate breakpoints for i^{th} stroke.) respectively for an inputting sketchy symbol. The problem of *stroke segmentation* using templates can then be defined as to select k numbers of breakpoints from the ordered candidate breakpoints to fragment the storke into some segments such that a sketchy shape represented by these segments is fit for some of shape definitions in template library with minimal fitting error.

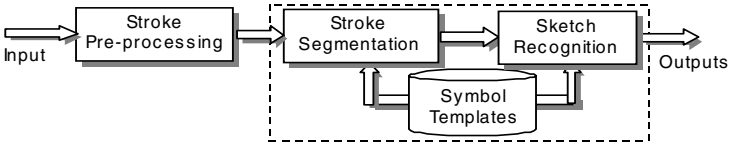

Fig. 3. Flowchart of strategy for adaptive sketch recognition

To find an optimal fragmentation of a set of strokes with template T, one assumes that the optimal solution for fragmenting everything up to the selected breakpoint with a template $T\{t(i)|i=1,2,\ldots,NT-1\}$ has been computed, and the piece from the choice breakpoint to the end is then fit with $T\{t(NT)\}$. A recursive solution is then defined based on above optimal substructure. Let $d(n,m,k,t)$ be a minimal fitting error to approximate every point up to the m^{th} point in the n^{th} stroke with the template t, and let $f(S_n,i,m,t(j))$ be the fitting error, resulting from fitting the segment from the i^{th} point up to the m^{th} point in the n^{th} stroke using $t(j)$. The best fragmentation for a set of strokes with NS strokes using K breakpoints and a template T would thus be $d(NS, NB,K,T)$. The recursive definition of $d(n,m,k,t)$ is expressed as follows:

$$d(n,m,k,t)=\begin{cases}\left[\sum_{i=1}^{n-1} f(S_i,1,NB_i,t(i))\right]+f(S_n,1,m,t(n)), & \text{if } k=0;\\ \min\limits_{k<i<m}\{f(S_n,i,m,t(NT))+d(n,i,k-1,t(j\mid j=1,\cdots,NT-1))\}, & \text{if } n=1,k>0;\\ \min\begin{cases} f(S_n,m,NB_{n-1},t(NT))+d(n-1,i,k,t(j\mid j=1,\cdots,NT-1)) \\ \min\limits_{k<i<m}\{f(S_n,i,m,t(NT))+d(n,i,k-1,t(j\mid j=1,\cdots,NT-1))\}\end{cases}, & \text{if } n>1,k>0.\end{cases} \quad (1)$$

Sketch recognition can then take the results of optimization of stroke segmentation directly as candidate symbols and prune the list of candidate symbols by matching each of primitives and their relationships of drawing shape with the templates. In practice, we design a nested recursive solution by adapting to the technology of dynamic programming. This brings on not only the integration of stroke segmentation and sketch recognition concurrently, but the acceleration of the optimization process also. The contexts of drawing or user model can also be used to reduce the computing complexity. Experiments prove adaptability of this method to both different drawing styles and various shapes with different complexities. Details can be seen in [14].

3.3 Dynamic User Modeling

It is quite difficult and impossible to ask computers to completely understand various sketches. To facilitate the processing of computers, it may be helpful for perceptual processing of strokes that computer can incrementally share the drawing habits and cognitive understanding of humans. Therefore, we propose a dynamic user modeling

method to collect and analyze the user's drawings incrementally and establish user model dynamically to assist sketch recognition.

In our work, the term 'user model' mainly means how a user draws a particular sketchy shape or reflects the user's drawing style. For a specific user, a user model is organized as an incremental decision tree, where the root records the user's id and each leaf node records the class label of the inputting graphics, and the branch nodes, that is one part of the integrate graph represented by the leaf node, record the drawing properties of each stroke. All drawing attributes are put together to identify one stroke and used to avoid over-branching of the tree. Each time he/she is drawing, user model is used as an assistance of sketch recognition to predict the "possible shapes" and updated incrementally based on statistical calculation of his/her historical drawing properties. Fig. 4 shows the principle of our strategy of dynamic user modeling.

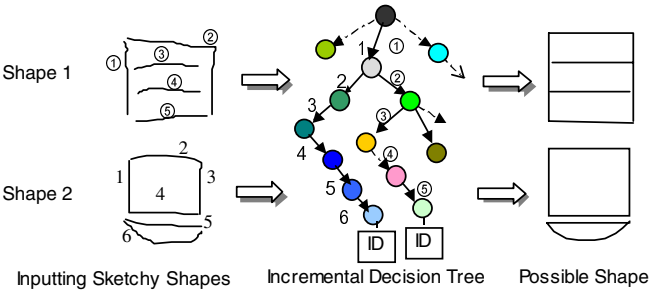

Fig. 4. Illustration of principle of dynamic user modeling

Along with the training process, the decision tree will grow and adjust to the user's styles in stroke sequence and construction of composite shapes. When a composite shape is being sketched in a sequence of strokes, each stroke may be tried to match along the branch. If the matching is successful, the possible composite shape can be predicted or recognized, and, at the same time, the weight of related nodes in the user models are adjusted. Let g' is a weight of the current searching node and *GList* is a list of the candidate objects of the current node, the weight of the candidate objects from this node is "$g' \cdot Glist$. All candidate objects from all surveyed nodes will be ranked by its weight and the objects with less weight will be deleted if there are the same objects classes. When another stroke is input, the direction of this stroke can be calculated as s_1 and s_2, the relation to the last stroke is r_1 and r_2, and the possible shape classes is R_{ij}. Consequently, the weight of the next searching node is: $g = g' s_m r_n R_{ij}$, where m, n=1 or 2, and i, j=1,…,14, which is based on our statistic analysis of our experiments. Otherwise, a new branch of the tree is created. Once a composite shape does not exist in the template, the strokes are collected and added to the system after shape regularization. Experimental results prove both effective and efficient of the proposed strategy. Details can be seen in [10] and [13].

4 Conclusion

This paper explores a novel concept of sketch-based informal user interface for graphic computing, Such an interaction mode can be characterized by two properties: pen-based stroking and perceptual processing of strokes, and makes user transfer visual ideas into computer without converting the ideas into a sequence of tedious command operations. This carries through the vision of human-centric computing. A prototype system has embodied some characteristics of informal user interface and two core technologies, adaptive sketch recognition and dynamic user modeling, make it more robust. Obviously, an important task for further researches is to identify an emerging application domain and find a paradigm of interface for that domain.

Acknowledgement

The work described in this paper was supported by grants from National Natural Science Foundation of China (Project No. 69903006 and 60373065) and the Program for New Century Excellent Talents in University of China (2004).

References

1. van Dam A. Post-WIMP user interfaces, Communications of ACM, Vol.40, No.2 (1997).
2. Nielsen J. Non-command user interfaces, Communications of ACM, Vol.36, No.4 (1993).
3. Fish J and S Scrivener, Amplifying the mind's eye: Sketching and visual cognition, Leonardo, Vol. 23, No. 1 (1990) 117-126.
4. Sezgin T. M., Stahovich T., Davis R., Sketch-based interface: early processing for sketch understanding, Proceedings of the 2001 Workshop on PUI**,** Orlando, Florida, (2001) 1-8.
5. Saund, E, Finding Perceptually Closed Paths in Sketches and Drawings, Transactions on Pattern Analysis and Machine Intelligence. Vol.25, No.4, (2003) 475-491.
6. Rubine Dean, Specifying gestures by example, Computer Graphics, Vol. 25, No. 1 (1991).
7. Fonseca M. J., Pimentel C., Jorge J. A., An online scribble recognizer for calligraphic interfaces. In: AAAI Symposium on Sketch Understanding, AAAI Press (2002) 51-58.
8. Xu X G, Sun Z X, Peng B B, et al, An online composite graphics recognition approach based on matching of spatial relation graphs, IJDAR, Vol. 7, No. 1 (2004) 44-55.
9. Sezgin T. M. and Davis R., HMM-Based Efficient Sketch Recognition, Proceedings of the international conference on Intelligence user interfaces, San Diego, USA, 2005.
10. Sun Z. X., Liu W. Y., et al, User Adaptation for Online Sketchy Shape Recognition. Lecture Notes in Computer Science, Vol. 3088. Springer-Veralg (2004) 303-314.
11. Levent Burak Kara, Thomas F Stahovich, Sim-U-Sketch: A Sketch-Based Interface for Simulink, Proceedings of AVI-2004 (2004) 354-357.
12. Newman M W, James L, Hong J I, et al: DENIM: An informal web site design tool inspired by observations of practice, HCI, Vol. 18 (2003) 259-324.
13. Sun Z X, Wang Q, Yin J F, et al, Incremental Online Sketchy Shape Recognition with Dynamic Modeling and Relevance Feedback, Proceedings of ICMLC2004, Shanghai, China, (2004) 3787-3792.
14. Sun Z X, Yin J F, Yuan B, A novel approach for sketchy shape recognition, Proceedings of GREC2005, Hong Kong, China, (2005).