A Novel Constraint-Based Approach to Online Graphics Recognition

Luo Yan, Guanglin Huang, Liu Yin, and Liu Wenyin

Department of Computer Science, City University of Hong Kong
83, Tat Chee Avenue, Kowloon Tong, Kowloon, Hong Kong SAR, China
{luoyan,hwanggl}@cs.cityu.edu.hk, {liuyin,csliuwy}@cityu.edu.hk

Abstract. Online graphics recognition has become the key problem for pen-based user interface on small screen devices, such as PDA and Tablet PC. In this paper, a novel constraint-based approach to online graphics recognition is proposed. The key idea of our approach is that when the user is drawing a graphic object, the system can extract the constraints between primitives and basic shapes from the object and use these constraints to retrieve similar graphic objects from the database at run time. The user can then choose the standard object from the ranked list of results to replace his sketches before he finishes drawing all strokes of the object. For this purpose, we summarize three types of primitives and several types of basic shapes as the basic components of a graphic object. We also define a set of constraints between primitives and basic shapes to represent their structural relations. The algorithms for online constraint extraction and graphics recognition are also presented. Experimental results show that our approach is efficient for online graphics recognition and effective for improving the user's productivity.

1 Introduction

Recently pen-based devices such as PDA and Tablet PC have become more and more common to the general public. In these devices, graphics is an important and useful means for users to store information, express thought, and sketch designs. Many systems were developed to facilitate users to draw graphics, such as Microsoft Visio, SmartDraw, and AutoCAD. In these systems, the user is asked to draw graphics by selecting the particular type of graphic object from lots of toolbar buttons or menu items. This task is very time-consuming and inconvenient, especially when the number of predefined graphic objects in the system is very large. The most convenient and natural way for human beings to draw graphics should be using a pen to draw sketches, just like drawing on a real sheet of paper. However, the sketches drawn in this way are not standard and clear in appearance, not compact in representation and storage, and not easy for machines to understand and process. It is necessary to recognize and convert the sketches to the regular and standard graphic objects that the user intends to draw. Moreover, it is even better if we can do recognition while the user is sketching since the recognized parts can provide immediate and useful feedback to the user so that he can realize errors or inappropriateness earlier and therefore draw the graphics more perfectly. In many cases, recognizing graphic objects early can also significantly save the user's input strokes and time. Hence, online graphics

A. Fred et al. (Eds.): SSPR&SPR 2004, LNCS 3138, pp. 104–113, 2004.

recognition has become the key problem for pen-based user interface on these small screen devices. Moreover, online graphics recognition can be also viewed as a query and retrieval problem. The user's input strokes can be viewed as a query and the system retrieves the similar graphic objects from a number of predefined standard graphic objects. Although the aims of retrieval and recognition are different, the underlying technology is common in that a matching procedure is needed to compare the input pattern with each known pattern. Therefore, the techniques for retrieving online graphics are also within the scope of online graphics recognition. In the following, we will not distinguish retrieval from recognition. The readers should bear in mind the common points and differences between them.

Compared with offline graphics recognition, online graphics recognition has some special characteristics. First, the input graphic object for online graphics recognition is usually incomplete, since our goal is to recognize the user's sketches before he finishes the whole graphic object, which can provide an immediate and useful feedback to the user. This characteristic implies online graphics recognition has to recognize the user intended object based on partial information in many cases. Second, the strokes in the same graphic object can be drawn in different orders by different users. Hence, the incomplete user's input of the same graphic object can be very different for online graphics recognition. That means it is not easy to apply the traditional matching methods for offline graphics recognition to online graphics recognition, since there can be many different kinds of incomplete graphic objects for the same complete one and it is difficult to match all of them to the complete one. Third, online graphics recognition needs more efficiency than offline graphics recognition. The system has to provide the immediate feedback to the user at run time; otherwise, it will be tedious and time-consuming instead of saving the user's input strokes and time. Hence, the efficiency of online graphics recognition is very important for a good user interface.

Many research works have been done on such online graphics recognition. Zeleznik et al. [1] have invented an interface to input 3D sketchy shapes by recognizing the predefined patterns of some 2D graphic objects. Jorge's group [2][3] have implemented an online graphics recognition tool that can recognize several classes of simple shapes based on global area calculation, which can hardly distinguish ambiguous shapes such as pentagon and hexagon and therefore cannot achieve high recognition precision generally. SILK [4] is an informal sketching tool that combines many of the benefits of paper-based sketching with the merits of current electronic tools. JavaSketchIt [5] is another system for this purpose, which can generate a Java interface from hand-drawn geometric shapes. SKETCHIT [6] is a system that can transform a single sketch of a mechanical device into multiple families of new designs. LADDER [7] is a language to describe how sketched diagrams in a domain are drawn, displayed, and edited, and used for online graphics recognition. The recognition approach is still not adequate for a real software tool that can be used for inputting most classes of diagrams. Hence, in order to provide the capability to input more complex diagrams, it is necessary to extend the online graphics recognition approach to handle more complex and composite shapes, as done in SmartSketchpad [8], which can efficiently and effectively input composite graphic objects by sketching only a few constituent strokes.

2 Our Approach and Contribution

In this paper, we propose a novel constraint-based approach to online graphics recognition. The key idea of our approach is that when the user is drawing a graphic object, the system can extract the constraints between primitives and basic shapes from the object and use these constraints to retrieve or recognize similar standard graphic objects from the database at run time. The user can then choose the standard object from the ranked list of results to replace his sketches before he finishes drawing all strokes of the object.

Our contribution includes, 1) we summarized three types of primitives and several types of basic shapes; 2) we defined a set of constraints between primitives and basic shapes to represent their structural relations; 3) we developed an algorithm for online constraint extraction from the user's input graphic object, which is incomplete in many cases; 4) we developed another algorithm for online graphics recognition based on the constraints of the user's input graphic object; 5) we proposed an algorithm for calculating the similarity between the user's input graphic object and the candidate graphic objects for displaying the recognized results in a ranked list.

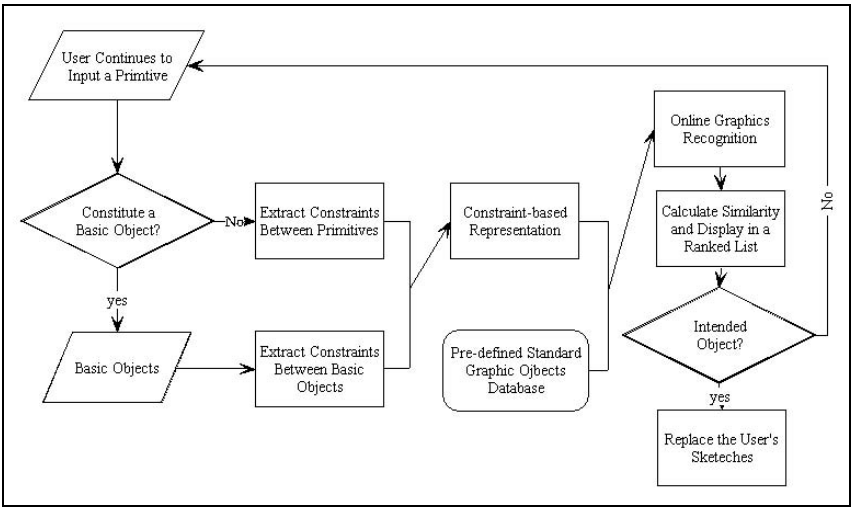

Fig. 1. The flowchart of our approach

Figure 1 is the flowchart of our approach. The user begins his sketches by drawing some basic strokes (or primitives). The system starts to extract the constraints between these primitives and uses the extracted constraints to recognize the similar standard graphic objects in the database. By using our proposed similarity calculation algorithm the system can then calculate the similarity between the user's input graphic objects and the candidate graphic objects, and display the recognized results in a ranked list. If the user's intended graphic object is displayed in the list, he can just choose this standard object to replace that incomplete sketches he has just drawn. The system applies these procedures, such as constraint extraction, graphics recognition, and similarity calculation, at the same time as the user is drawing the sketches. Hence

it can facilitate the user to draw graphics by significantly saving the user's input strokes and time.

In the following of this paper, we first propose our constraint-based approach to describe the user's input graphic object in Section 3. Then, algorithms for constraint extraction and graphics recognition are discussed in Section 4 and 5, respectively. Finally, experimental results and concluding remarks are presented.

3 Constraint-Based Representation of Graphic Objects

As we discussed above, our approach focuses on the relative spatial relations between primitives and basic shapes. Hence, we use constraints to represent the user's input graphic object in our approach. Constraint, or geometric constraint, is not a new concept, which has been widely used in CAD systems (e.g., [9]). However, in many CAD systems (e.g., [9][10][11]), the constraints are defined, extracted, and specified by professional and experienced users. In our approach, we defined a set of constraints to describe the spatial relations between primitives and basic shapes. The system can extract constraints while the user is drawing the sketches and uses these constraints to recognize similar standard graphic objects in the database at run time. Thus, our definition of constraints should be broad enough to support a wide range of graphic objects, while remaining narrow enough to be comprehensible.

First of all, we define three types of primitives: Line, Circle, and Arc. As shown in Figure 2, P_1 and P_2 are two endpoints of a Line. We can assume P_1 is the start-point and P_2 is the end-point such that we can define the direction of a Line is from P_1 to P_2. For a Circle primitive, it also has two attributes, C (center-point) and R (radius). In the definition of an Arc, we use P_1 and P_2 to represent the start-point and end-point of an Arc since the user usually pays more attention to the start-point and end-point than the center-point. That means the user does not care about the curving of an Arc but the position of an Arc. However, the direction of the bow of an Arc is very important for the user to distinguish different graphic objects. Hence, if we define a positive direction from P_1 to P_2, like X-axis, then we can define the Direction of the bow of an Arc.

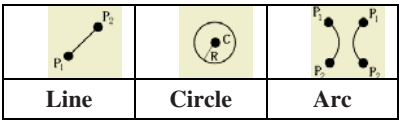

Fig. 2. Primitives

Then we define the constraints between the above primitives. We analyzed more than 300 types of graphic objects to summarize the constraints. Since we only use three parameters (i.e., P_1, P_2, and Direction) to define an Arc primitive, we can image an Arc as a Line plus a Direction. Hence, we can just analyze the constraints between Line primitives and apply these constraints to Arc primitives by simply adding a Direction parameter. Therefore, we first define four constraints between Line primitives and Arc Primitives, including Connection, Intersection, Parallelism, and Perpendicularity. For a Circle primitive, we regard it as a basic shape, which is discussed in the following section, and define the constraints between basic shapes and primitives to describe their spatial relations. Here, for easily understanding, we only use Line primitives to describe the four constraints between Line and Arc primitives. For the cases including Arc primitives, only one additional parameter, Direction, is required.

(1) Connection

Connection is a constraint to describe that two primitives share the same end-point, just like they are connected at one end. Figure 3 illustrates this constraint.

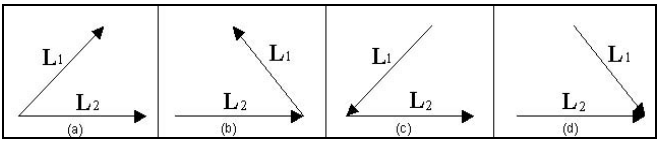

Fig. 3. Connection

From the above figure, we can see that there are only four cases between two primitives that are connected with each other, since one Line or one Arc has two end-points. We use a parameter *type* to represent this information and use another parameter *angle* to store the angle between the two primitives.

$$angle = \cos(\alpha) = L_1 \bullet L_2 / (| L_1 \| L_2 |)$$

In this definition, the parameter *angle* itself is not sufficient to fully specify the spatial relationship of two intersected lines since the angle has a direction. Thus, we use another parameter *direction* to describe this information. Consider $L_1(x_1,y_1,0)$ and $L_2(x_2,y_2,0)$, which are 2D vectors in 3D space, and their cross product

$$L = L_1 \times L_2 = \left(\begin{vmatrix} y_1 & 0 \\ y_2 & 0 \end{vmatrix}, \begin{vmatrix} 0 & x_1 \\ 0 & x_2 \end{vmatrix}, \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} \right) = (0,0,L_z) \quad L_z = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1$$

L is perpendicular to the plane formed by L_1 and L_2, and its direction complies with the *Right Hand Rule*. Thus we can determine the direction by the sign of L_z. In addition, we use the parameter *length* to describe the relative length of L_2 to L_1 (*length* = $|L_2|/|L_1|$).

(2) Intersection

Intersection is a constraint to describe that two primitives are intersected with each other, which means they share the common point on the primitives.

In Figure 4, two Line primitives are intersected with each other at iPoint. We define four parameters to describe this constraint. The first two parameters describe the relative position of iPoint on two Line primitives as follows.

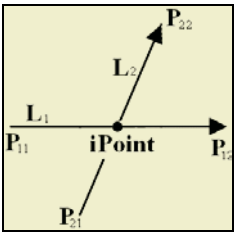

Fig. 4. Intersection

$$\mathrm{iP}_1 = \frac{\mathrm{iPoint} - P_{12}}{P_{11} - P_{12}} \qquad \mathrm{iP}_2 = \frac{\mathrm{iPoint} - P_{22}}{P_{21} - P_{22}}$$

We use other two parameters, *angle* and *length*, to describe the angle between two primitives and relative length of them just like Connection constraint.

(3) Parallelism

Similar to Intersection, we also use four parameters to describe Parallelism geometric constraint. The first one is *distance* = $D(L_1,L_2)/|L_1|$, in which $D(L_1,L_2)$ denotes the real distance between line L_1 and L_2. The second one, *direction*, is used to describe whether L_2 is on the left or right to L_1 and the computing method is

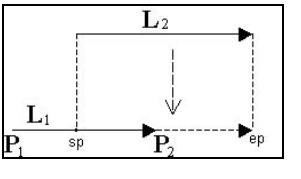

Fig. 5. Parallelism

similar to the definition in Connection constraint. Moreover, we use two other parameters to specify their relative position and length. In Figure 5, L_1 and L_2 are parallel to each other; *sp* and *ep* are the projections of the endpoints of L_2 on L_1. We set:

$$\text{start - point} = \frac{\text{sp} - L_1.P_2}{L_1.P_1 - L_1.P_2} \quad \text{end - point} = \frac{\text{ep} - L_1.P_2}{L_1.P_1 - L_1.P_2}$$

(4) Perpendicularity

For the Perpendicularity relationship in which two primitives are connected or intersected, we can use Connection or Intersection to represent it, respectively. Here, we only define the Perpendicularity between two primitives when they are not connected or intersected:

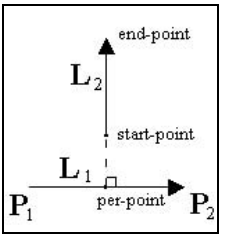

Fig. 6. Perpendicularity

- Length $= |L_2| / |L_1|$
- Per-point is the perpendicular point of L_2 on L_1
- Start-point $= |\text{per - point}, L_2.P_1| / |L_2|$
- End-point $= |\text{per - point}, L_2.P_2| / |L_2|$

When we calculate start-point and end-point, we set a sign to the value of them. We set it positive if the point is on the left-hand side of L_1 and negative on the right-hand side. The computing method is similar to computing direction in Connection constraint. In Figure 6, the values of start-point and end-point are both positive.

Some primitives can constitute a very common and basic shape, which is often used by users in many complex graphic objects. Especially, the user usually divides the whole sketch into some basic shapes when drawing a complex sketch. Therefore, we also summarized some basic shapes to represent the user's input graphic object at a higher level, as illustrated in Figure 7.

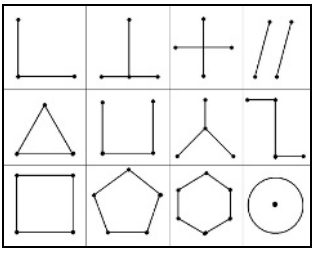

Fig. 7. Some basic shapes

For these basic shapes, we also define a set of constraints to describe the structural relations between them. For instance, to the closed shapes, such as Rectangle and Circle, we defined the Inner/Outer constraint to describe whether other primitives or basic shapes are inside or outside them, because, in many cases, the user pays more attention to the Inner/Outer relations between shapes than the precise position or orientation of these shapes. For other non-closed shapes, we also defined other constraints (e.g., relative position and orientation) to describe the structural relations between these basic shapes and other primitives.

4 Online Constraint Extraction

In this section, we discuss our developed algorithm for online constraint extraction, which means that our approach extracts the constraints between the primitives and basic shapes while the user is drawing sketches. This algorithm is developed based on

our previous work for offline graphics recognition [12]. We divide the procedure of recognizing user's drawing sketches into three stages.

1. The user begins his sketches with simple primitives, which do not constitute any basic shapes. However, the simple primitives do contain useful information about the user's intention, e.g., they can be a part of a standard graphic object. Hence, our algorithm extracts the constraints between the primitives as the representation of user's input at this stage and uses these constraints to retrieve the standard graphic objects that contain the similar part.
2. When the user continues to draw sketches, there are enough primitives to constitute a basic shape. At this stage, our approach uses the constraints between the primitives to recognize them as a basic shape and provides a useful and immediate feedback to the user. The user can accept the feedback or adjust his sketches at this stage. Once the user accepts his current sketches as a basic shape, his sketches are replaced by the standard basic shape and he can go on with his sketches. The system will then extract the constraints between the newly drawn primitives until a new basic shape is recognized.
3. As the user goes on with his sketches, the constraints between the basic shapes should also be extracted since they contain much useful information for recognition. Hence, at the third stage, the system extracts the constraints between basic shapes and constructs a hierarchical constraint-based structure for recognition.

For the detail of the online constraint extraction algorithm, see the Case-based Knowledge Acquisition Algorithm (CKAA) [12].

5 Online Graphics Recognition

The constraints extracted by the above algorithm are stored in a syntactical tree. We use this tree to retrieve or recognize the similar standard graphic objects. We search all the predefined graphic objects in the database for those that contain the similar constraints, i.e., contain the similar graphic object to user's input. However, we cannot use the matching method for recognition since the user's input is usually incomplete. Therefore, we propose a new scheme, which is like a reasoning method, for recognizing graphic objects based on the constraints. When we test one standard graphic object for whether it contains the similar graphic object to the user's input or not, we first hypothesize that one stroke of the standard graphic object is in the user's input. Using the constraints extracted from the user's input, we can calculate the specification of another primitive or basic shape based on the hypothesis stroke. Then we search the standard graphic object to see whether it contains this stroke. If the stroke is found, we continue tracing other constraints until all strokes are found in the standard graphic object, which means, this standard graphic object contains the similar graphic object to the user's input. Otherwise, we select another stroke to repeat this hypothesizing/testing procedure. The algorithm presented below deals with ideal situations. In practice, the tolerance should be considered and the matching measure should be defined, which are discussed in our previous work [12]. The detail of the online graphics recognition algorithm is shown in Algorithm 1.

When the result is output, the similarity between the user's input graphic object and the standard graphic object is calculated from two aspects. The first is the similarity between primitives, which is calculated according to the difference of length, angle and position between the two primitives. The second is the similarity of constraints, which is calculated by the percentage of exact matched primitives in the standard graphic object. According to the similarity of the standard graphic objects, we select top 10 objects in the database and return them in a ranked list to the user.

Algorithm 1: Online Graphics Recognition

Input: *SC*: the set of constraints from the user's input graphic object
DB: the database consists of standard graphic objects
TL: the tolerances, e.g., length and number tolerance

Variables: *CT*: the temporary constructed tree for reasoning procedure
SM: the set of marks to indicate primitives that have been tested

Output: *RR*: the recognition result, which type the graphic object is

1. Select a standard graphic object *SG* from *DB*. If all standard graphic objects have been searched, then stop (failure)
2. Set *CT* empty and initialize *SM*
3. Select the next primitive *P* from *SG*, which has not been marked in *SM*. Add it into *CT* as the root, and mark it in *SM* to indicate this primitive has been tested. If all primitives have been marked in SM, goto step 1.
4. Select the next constraint *C* from *SC*. If all constraints have been traced then stop (success) and output the current *SG* as *RR*
5. Calculate the new primitive or basic shape *P'* using *P* and *C*
6. Search for a *P''* in *SG*, which is similar to *P'* using the tolerances in *TL*.
7. If *P''* is found then set it as a child of *P* in *CT* and mark in *SM* to indicate *P''* has been used and goto Step 4
8. If *P''* is not found and the number of missing primitives exceeds the tolerance then goto Step 2. Otherwise, goto Step 4

6 Experimental Results

We have implemented a prototype system and done several experiments based on a database consisting of 345 standard graphic objects, some of which are illustrated in Figure 8. The user is asked to draw graphic objects and the system provides immediate recognition results, from which the user can select his intended standard graphic object. The average recognition accuracy is 90.5% since the user's input can be very different. We also record the number of strokes that have been saved for drawing an object. In our experiments, the number of one standard object's strokes ranges from 1 to 14 and the average is 10.32. The average number of saved strokes is 2.78, nearly 27%. We also evaluate the response time of our approach. The average response time to user's

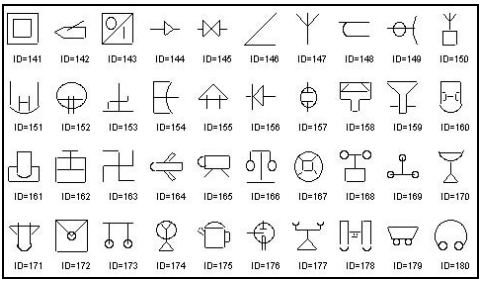

Fig. 8. Some standard graphic objects

input is within 100ms, which is efficient enough to give real-time response for a database consisting of several hundreds of graphic objects. From the experimental results, we can see that our approach is effective for online graphics recognition and saving the user's input strokes and time.

7 Conclusion and Future Work

In this paper, we proposed a novel constraint-based approach to online graphics recognition, with which the system can extract the constraints between primitives and basic shapes from the user's input and use these constraints to recognize similar standard graphic objects. Several constraints are defined and two algorithms are developed. Experimental results show that our approach is efficient for online graphics recognition and effective for saving the user's input strokes and time. However, some aspects of our approach can be improved. More types of primitives, basic shapes, and constraints can be added into our approach in the future to support more complex and various graphic objects. Two algorithms for online constraint extraction and graphics recognition can be also revised to improve the recognition accuracy and save the user's input stroke and time. We also plan to provide more graphic objects from various domains to do experiments to test our system.

Acknowledgement

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong SAR, China [Project No. CityU 1073/02E].

References

1. R.C. Zeleznik, K.P. Herndon, and J.F. Hughes, "KETCH: An Interface for Sketching 3D Scenes", *Proc. of SIGGRAPH*, New Orleans, pp.163-170, 1996.
2. M.J. Fonseca and J.A. Jorge, "Using Fuzzy Logic to Recognize Geometric Shapes Interactively", *Proc. of the 9th IEEE Conf. on Fuzzy Systems*, Vol.1, pp.291-296, 2000.
3. M.J. Fonseca, C. Pimentel, J.A. Jorge, "An Online Scribble Recognizer for Calligraphic Interfaces", *Proc. of AAAI Spring Symposium Series – Sketch Understanding*, 2002.
4. J.M. Landay and B.A. Myers, "Sketching Interfaces: Toward More Human Interface Design", *IEEE Computer*, Vol. 34, No. 3, pp. 56-64, 2001.
5. A. Caetano, N. Goulart, M.J. Fonseca, and J.A. Jorge, "JavaSketchIt: Issues in Sketching the Look of User Interfaces", *Proc. AAAI'02 Spring Symposium – Sketch Understanding*.
6. C. Calhoun, T.F. Stahovich, T. Kurtoglu, L.M. Kara, "Recognizing Multi-Stroke Symbols", *Proc. of AAAI Sprint Symposium Series – Sketch Understanding*, 2002.
7. T. Hammond and R. Davis, "Ladder: A Language to Describe Drawing, Display, and Editing in Sketch Recognition", *Proc. of IJCAI'03*, 2003.
8. W. Liu, X. Jin, and Z. Sun, "Sketch-Based User Interface for Inputting Graphic Objects on Small Screen Devices", *Lecture Notes in Computer Science* 2390, pp.67-80, 2002.

9. J.K. Lee and K. Kim, “Geometric Reasoning for Knowledge-based Parametric Design using Graph Representation'”, *Computer-Aided Design*, Vol.28, No.10, pp.831-841, 1996.
10. S. Ait-Aoudia, B. Hamid, A. Moussaoui, T. Saadi, “Solving Geometric Constraints by a Graph-Constructive Approach”, *Proc. of ICIV’1999*, pp.250-255, 1999.
11. I. Fudos, C.M. Hoffmann, “A Graph-Constructive Approach to Solving Systems of Geometric Constraints”, *ACM Trans. on Graphics*, Vol.16, No.2, pp.179-216, 1997.
12. Y. Luo and W. Liu, “A Case-based Interactive Approach to Graphics Recognition in Engineering Drawings”, *Proc. of GREC’2003,* pp.170-181, 2003.