EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2009)
C.Grimm and J.J. LaViola Jr. (Editors)

Towards Beautification of Freehand Sketches using Suggestions

S. Murugappan[1], S. Sellamani[1], and K. Ramani[1, 2]

[1]School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA
[2]School of Electrical and Computer Engineering (by courtesy), Purdue University, IN 47907, USA

Abstract

Beautification of freehand sketches is integral for building robust sketch understanding systems and sketch-based interfaces for CAD. Many of the current methods for beautification do not consider some important information implied in the sketches such as spatial relationships (geometric constraints) between primitives. In addition, as the freehand input is ambiguous in nature, correctly interpreting the visual scene the user has in mind is a difficult problem. To this extent, we present our ongoing work, a suggestive interface for constraint-driven beautification of freehand sketches which provides multiple interpretations of the freehand input, from which the user can choose the intended result. A preliminary user study has been conducted to evaluate the effectiveness of the proposed method.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation - Line and Curve Generation

1. Introduction

Beautification of freehand sketches is the process of transforming informal and ambiguous freehand input to more formal and structured representations. Such a transformation process is one of the fundamental requirements in building robust sketch understanding systems, for assisting rapid creation and evaluation of new ideas (e.g. Sketch to 3D Translation) and also in reduction of the total time and effort spent in creating drawings on a computer. In this paper, we consider freehand sketches that represent CAD- like models only i.e., not of free-form and smooth shapes.

The initial steps in beautification include segmentation and recognition. The aim of segmentation is to identify the critical points on the strokes that split the stroke into its constituent primitives and the goal of recognition is to classify and fit the segments between the adjacent critical points as low level-geometric primitives (like lines and circular arcs).

As a result of segmentation and recognition, the freehand sketch is converted to a more simplified representation, where the different strokes are closely approximated by a set of parameterized geometric primitives. However, such a direct transformation does not take into account some important global information conveyed in the freehand sketches such as the spatial relationships between different primitives in a stroke and between strokes. These relationships are represented as geometric constraints, which are widely used and an integral part in many design related applications, such as drawing programs, CAD tools and graphical user interfaces [PR07a]. In current drawing systems, specifying geometric constraints is a difficult, time consuming and a tedious task. Also, the users need to undergo prior training before using the system. Novice users must be made aware of geometric constraints and also how these can be used to obtain what they want.

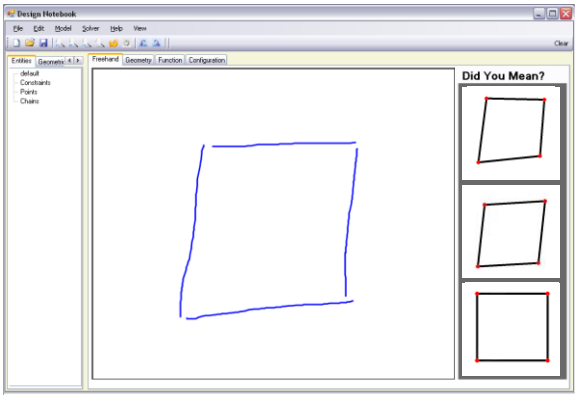


Figure 1: *Suggestive interface for beautification. The blue lines are the freehand strokes drawn by the user and the three small thumbnails on the right are the multiple interpretations generated by the system.*

To reduce this cognitive overload and to effectively support the use of geometric constraints in freehand sketching, the constraints have to be inferred automatically and solved simultaneously by the system without much intervention from

Sketch-Based Interfaces and Modeling 2009, New Orleans, LA, August 1–2, 2009.

user. [PR07a, IKTM98] are two such systems which demonstrate the potential advantages of using geometric constraints in driving beautification of freehand sketches. Due to the ambiguous nature of freehand strokes, correctly interpreting the visual scene the user has in mind becomes difficult as correct interpretation is best defined as what the user intends, the information the system does not have [MHA00]. For example, the user drawn strokes in Figure 1 can be interpreted either as a rectangle, parallelogram or as a quadrilateral depending on the recognizer used. Most of the current graphic recognizers provide either one of the results but not all. Each of the above interpretation has a different set of constraints that needs to be satisfied. The rectangle requires opposite sides to be parallel and the adjacent sides to be perpendicular whereas there is no perpendicularity constraints for parallelogram. Hence, in addition to inferring constraints, the system must be capable of determining what specific constraints are needed to beautify the sketch.

To this extent, we present our ongoing work, a suggestive interface for beautification of freehand sketches, which provides multiple interpretations of the freehand input (as visual snapshots), from which the user can choose the intended result. We refer to these multiple interpretations as "suggestions". The three small thumbnails inset in Figure 1 are the suggestions generated by the system for the input freehand stroke (blue lines). In addition to resolving ambiguity, suggestions help to speed up the sketching process, aiding the user to create precise diagrams very quickly. Our system is similar to Pegasus [IKTM98], but we support the use of arcs and circles in addition to just line segments and a stroke can represent any number of primitives. This makes the interface to be as natural as possible allowing the users to draw in an unrestricted fashion.

Cognitive studies show that users attend preferentially to certain geometric features while drawing and recognizing shapes [VD06]. We have used results from [Gol82, Arn82, Wer23, Sau03] to capture such features and relationships in the sketches. These captured perceptual biases are represented as geometric constraints and used to drive beautification.

A user study has been conducted to determine if the system correctly beautified the users' sketch by correctly capturing the specified intent. The remainder of the paper is organized as follows. In Section 2, some related work is introduced. Section 3 provides an overview of the system section 4 describes the methodology used behind the generation of suggestions in detail. Section 5 explains the user experiences with our system.

2. Related Work

2.1 Constraint based systems

A lot of work has been done in drawing diagrams on computers right from Sutherland's Sketchpad [Sut64] in 1964. We will overview some of the important techniques that have been developed. Interactive drawing applications employing direct manipulation techniques have been very successful [Bad00]. Sketchpad [Sut64] and snap-dragging systems [Bie88] permit constraints to be specified both explicitly and implicitly through the use of pseudo-pen locations and mouse movement respectively. Constraint based drawing systems like Briar [Gle92] and Rockit [KWL93] infer graphical constraints from the users' dragging operation and allow the user to select from several candidate constraints. Chimera [KF93] provides a constraint inference engine which works by comparing multiple snapshots. Explicitly selecting the constraints after each operation requires considerable effort from the user thereby reducing the usability of these systems. QuickSketch [LHBE97] is a 2D-based 3D modeling tool. It is a system for sketching with constraints that supports geometric recognition of simple strokes as well as a constraint maintenance tool. GIDeS++ [JVJ*04] is an incremental calligraphic drawing interface that naturally handles ambiguous interactions. ParSketch [NCAC07] is a sketch-based interface for editing 2D parametric geometry. Pegasus [IKTM98] is a rapid sketching tool for line drawings. It interactively infers seven kinds of constraints: connection, parallelism, perpendicularity, alignment, congruence, symmetry, and interval equality. Our system uses many of the concepts learned from Pegasus, but in addition infers other geometric constraints like tangency and concentricity. Pu and Ramani [PR07a] developed a statistical method - Relative Shape histogram to infer constraints directly from the freehand sketches instead of the primitives. This method works only for very limited types of constraints and in only certain specific configurations. For example, the method fails to capture the perpendicular constraint between the lines in a '+' symbol.

Langbein et al [LMM04] used a constraint-based approach to beautify boundary representation models reconstructed from 3D range data. They find geometric regularities approximately present in the model and impose a consistent subset of them to refine the model. Zou and Lee [ZL07] used a similar approach to beautify 3D polyhedral models reconstructed from 2D sketches composed of line segments. Both these methods use priorities to select a subset of constraints incase of inconsistencies. Our suggestions generation method also uses the notion of priorities to solve over-constrained sketches.

2.2 Suggestive Interfaces and Multiple Candidates

Multiple candidates are commonly used in recognition-based systems such as handwriting or speech recognition to solve the inherent ambiguity problem [MHA00]. Igarashi et al [IKTM98] adopted a strategy to beautify a single stroke (exactly one line segment) one after another to prevent accumulation of recognition errors. The system generates multiple candidates of possible intended geometry which are displayed next to the strokes. This often cluttered the scene making it complicated. Though a high efficiency was achieved compared to the traditional design systems, there were restrictions on the users drawing style as it supported only line segments. Lineogrammer [ZBLF08] is a diagram-drawing system, an extension of [IKTM98], but unlike [IKTM98] provides at most one alternate to the input stroke, thereby reducing the visual clutter. Like [IKTM98], Lineogrammer also does not support the use of curves like arcs and circles. Tsang et al [TBSR04] introduced a suggestive interface for image guided sketching where multiple candidates are generated; displayed spatially in place with the existing geometry. Chateau [IH01] extends Pegasus into 3D space. The user specifies hints about a desired operation to the system by highlighting related components. The system infers possible operations based on

the hints and presents the results as small thumbnails. We also use this idea of visual snapshots to communicate the feedback effectively, but in our system, the user neither provides hints [IH01] nor is forced to choose one of the multiple candidates [IKTM98] during sketching. Our system like [TBSR04] presents the top candidates from a discrete space of possibilities and unlike [MAB*97], which presents representative examples of a large continuous space.

2.3 Perceptions

Our work is greatly influenced by Saund et al's work [SM94, Sau03, SFLM04] which shares our motivation, to create a transparent and immediately accessible user interface where one should be able to walk up and draw without prior training, oblivious to whether the computer recognizes something correctly or not and that most work should be done directly on the drawing (without having to deal with menus). We take this work a step further and support the creation of drawing precise diagrams with different types of primitives. Veselova et al [VD06] have used results from perceptual studies to build a system capable of learning descriptions of hand-drawn symbols. We also have used such results in capturing geometric relationships in the freehand sketches to drive beautification.

3. System Description

Freehand sketches are usually composed of a series of strokes. Our sketch- based interface allows the user to draw in a natural way just as they would do on paper. The user can draw freely and there is no restriction on how a particular sketch is drawn. The stroke can represent any number of shape primitives connected together. The primitives recognized include lines, circles and circular arcs. The current implementation of the system uses Microsoft Tablet PC SDK to capture the user input strokes. The users are provided the option to view their sketch either as raw pen strokes or as primitives or both.

Our approach to transforming the freehand sketches to formalized representations (i.e. beautification) consists of two sequential stages - Initial processing module and Suggestions module. In the initial processing module, the input pen stroke(s) are decomposed into low-level geometric primitives with minimal error. There are four steps in the initial processing stage, namely, resampling, segmentation, recognition and merging. This is done using the algorithm described in [MR09]. Next, the system analyzes the output of the first stage to generate multiple interpretations. The various steps of the suggestion module shown in Figure 2 are explained in detail in the following sections. The suggestion module first identifies the spatial relationships between the primitives. These relationships are represented as geometric constraints, which are further grouped into different sets and solved by a geometric constraint solver. The solutions output from the solver are the different suggestions generated by the system. These candidates are then evaluated and ranked, returning only the top relevant choices. The user can either choose one of the suggestions or ignore them and continue sketching.

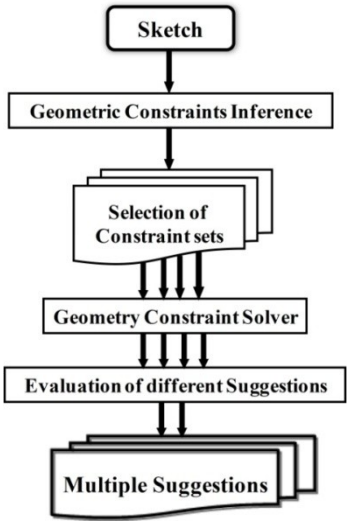


Figure 2: *The pipeline of Suggestions Module. The input is the `Sketch' with detected critical points and recognized segments and the output is the multiple suggestions generated.*

4. Generation of Suggestions

An example of the beautification process in our system is shown in Figure 3. The user starts by sketching the freehand stroke (Figure 3 (a)). When the user lifts the pen, the system starts to process the input and the results after initial processing and generation of suggestions are shown in Figure 3(b). The user can continue sketching or choose one of the multiple alternatives generated and update the sketch. The alternatives are displayed as snapshots. On hovering around each snapshot, the system provides a larger preview of the geometry in the sketching area on top of the input stroke. When the user selects a suggestion (Figure 3(b)), the system infers this selection as an early commitment the user wishes to make and remembers the associated constraints (Figure 3 (c)). Next, when new strokes are added to the system (Figure 3(d)), suggestions are generated for the new strokes with respect to the already existing sketch. In this case, the system infers the concentricity constraint between the newly drawn circle and already existing arc. On constraint solving, the system snaps the centers by moving the circle to the arc. Depending on the constraints inferred, the system may modify already existing geometry. In this case, as vertical constraint on line is enforced, along with coincident constraints on the end points, the system adjusts the geometry accordingly as shown in Figure 3(e). The following sections explain these steps in detail. The current implementation of the system does not visually communicate what constraints are satisfied in each snapshot as it would unnecessarily clutter the scene. For example, if two lines are parallel, the system does not show any symbol or gesture indicating it. In a way, this helps in decoupling the users from conscious use of constraints and let them focus solely on sketching process. Also, novice users who do not have relevant training in CAD and geometric constraints can easily use this system with limited learning.

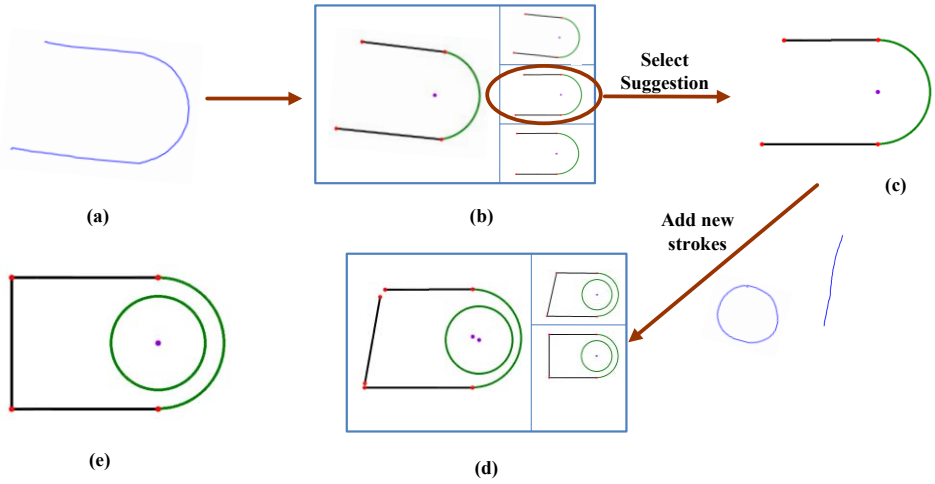


Figure 3: *An example of the beautification process with suggestions. (a) The user starts with a freehand stroke, (b) the result after initial processing (inset image) and generated suggestions (three thumbnails on the right), (c) The sketch is updated with the chosen alternative, (d) the status of the sketch after addition of new strokes*

The suggestion module has many benefits. It helps to speed up the beautification process aiding the users to create precise diagrams quickly. Also, it resolves ambiguities in segmentation process by suggesting removal of critical points in the sketch. For the sketch in Figure 4(d), the system inferred parallel constraints (and hence co-linearity) between line segments intersecting at the green circled red points and therefore removed them. It also effectively handles cases of over-sketching (Figure 4(c)) and under-sketching (Figure 4 (a)) through geometric constraints which typically require great involvement of user (manual editing operations).

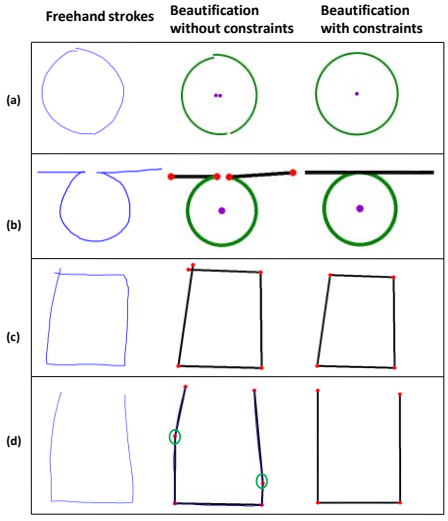


Figure 4: *Benefits of Suggestion module.*

4.1 Geometric Constraints Inference

The first step in generation of suggestions is inference of geometric constraints. They are spatial relationships between two entities which can be expressed quantitatively. They are usually classified as either (1) explicit constraints, which refer to the constraints that are explicitly specified by the user such as dimensions - distance between a point and a line or angle between two lines, (2) implicit constraints, which refer to the constraints that are inherently present in the sketch such as concentricity and tangency. It is natural for users to express geometric constraints implicitly when they are sketching. In our system, we detect the implicit constraints listed in Table 1 automatically. The entities considered include points, lines, arcs and circles.

	Point	**Line**	**Arc (Circle)**
Point	Coincidence, Horizontal and Vertical Alignment	Point on Line	Coincidence, Point on Arc (Circle)
Line	Point on Line	Parallel, Perpendicular, Collinear	Tangency
Arc (Circle)	Coincidence, Point on Arc (Circle)	Tangency	Tangency, Concentricity

Table 1: *Implicit Geometric Constraints inferred by our system.*

The implicit geometric constraints between entities are inferred as a ‘cost’ parameter, a measure of work that needs to be done by the system to satisfy that particular constraint. For two circles to be concentric, the cost incurred is equal to the distance between their centers. Similarly, the parameters for other constraints are computed as shown in Table 2. Some geometric properties can be captured using different sets of constraints and computing all of them result in redundancy. For example, a line can be made horizontal by either using the `horizontal' constraint or `vertical' alignment of the end points of the line. To avoid this redundancy, we infer only the ‘horizontal’ constraint, as line is higher in hierarchy to points. Similar such rules are used to avoid other redundancies. The

rationale behind computing the costs is to use them 1) in selection of constraints, i.e. if the cost is less than a certain threshold, then it would be included in the constraint set which would be later solved by the solver to yield suggestions (section 4.2) and 2) in evaluating the different suggestions generated as described in section 4.3. In addition to implicit constraints, explicit constraints can be added manually to the sketch.

Geometric Constraint	Parameters for Cost
Coincidence	Distance between two points
Concentricity	Distance between two centers
Parallel	Angle between two lines
Perpendicular	Angle between two lines
Horizontal	Angle between line and X-axis
Vertical	Angle between line and Y-axis
Horizontal alignment	Vert. distance between two points
Vertical alignment	Horz. distance between two points
Point on line	Shortest distance from point to line
Point on arc (circle)	|| Distance between point and center - radius ||
Tangency between line and arc (circle)	|| Shortest distance from center to line – radius ||
Tangency between arc (circle) and arc (circle)	|| Distance between centers – (sum of two radii) ||
Collinear lines	Distance between two parallel lines

Table 2: *`Cost' parameters for geometric constraints*

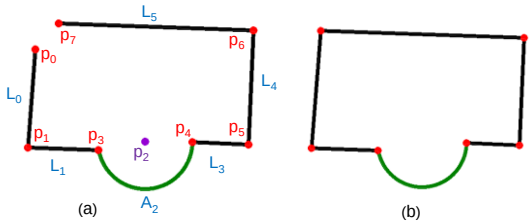
4.2 Selection of Constraint sets

The geometric constraints inference module infers all possible constraints implied in the freehand sketch and their associated costs. However, not all constraints are relevant and necessary for satisfying the sketch i.e. for constraint based beautification. Hence, a right subset of constraints needs to be determined which when satisfied by a geometric constraint solver yields a particular beautified version of the freehand sketch. Finding this subset becomes difficult as it is equivalent to correctly interpreting the intent. In order to solve this ambiguity problem, the system generates multiple sets of constraints which when satisfied provide different alternatives of the input sketch. Multiple sets can be created by considering all possible combinations of inferred constraints, but this will lead to combinatorial explosion.

To this extent, we take advantage of the cognitive studies done earlier to understand the specific geometric features that users attend to while drawing and recognizing shapes. We have used results from [Gol82, Arn82, Wer23, Sau03] to translate the perceptual cues into geometric constraints and also in selecting only those that are relevant. We explain these in the below paragraphs.

Goldmeier [Gol82] identifies the relevant and irrelevant features in a drawing by exploring their effect on perceived similarity. He found that people attend to specific properties called 'Singularities' i.e. special cases in the space of geometric configurations in the sense that small variations in them make a qualitative difference. The singularities he specifically mentions include parallelism, perpendicularity, horizontality, verticality and straightness.

Gestalt principles of primary concern in diagram analysis include smooth continuation, closure, spatial proximity and symmetry [Wer23]. These can be intuitively associated with formal geometrical properties. 'Smooth continuation' principle can be inferred as a tangency constraint at the junction where a straight line smoothly transitions to a curve. It is difficult to draw two separate strokes from the same starting point. Also, it is difficult to start and end a stroke at the same location. These situations are examples of 'closure' principle, which can be translated to coincidence constraints of the end points. Similarly, in cases where a line is drawn as broken line segments, like in Figure 4(d), they can be merged using coincidence and collinear constraints. For spatial proximity, entities that are spatially closer are more relevant than those that are far apart. Hence, two lines that are parallel and relatively close are more relevant than those that are far apart. Symmetry can be enforced through equidistant constraints about an axis. (Currently, our system does not use symmetry, as automatically determining the axis of symmetry in freehand sketches is difficult.)



Constraints Set - 1	Constraints Set - 2	Constraints Set - 3
Coincident, p7, p0	Coincident, p7,p0	Coincident, p7, p0
Fixation, L0	Parallel, L5,L1	Horizontal, L1
Fixation, L1	Parallel, L5,L3	Horizontal, L3
Fixation, A2	Parallel, L4,L0	Horizontal, L5
Fixation, L3	Perpendicular, L4,L3	Vertical, L0
Fixation, L4	Perpendicular, L4,L5	Vertical, L4
Fixation, L5	Perpendicular, L0,L1	Horizontal Alignment, p5,p3
		Horizontal Alignment, p5,p1

Constraints Set - 4	
Coincident, p7, p0	Parallel, L5,L3
Horizontal, L1	Parallel, L4,L0
Horizontal, L3	Perpendicular, L4,L3
Horizontal, L5	Perpendicular, L4,L5
Vertical, L0	Perpendicular, L0,L1
Vertical, L4	Horizontal Alignment, p4,p3
Parallel, L5,L1	Horizontal Alignment, p4,p2

(c)

Figure 5: *Selection of Constraint sets. (a) input sketch to the suggestion module, (b)output from the geometry constraint solver on satisfying constraints set-1, (c) the various constraints sets generated.*

Eggli et al [LHBE97] illustrate the difference in users' preferences in technical drawings versus symbols. Constraints like parallelism, right-angles, tangencies, point on entities and concentric circles are preferred in making technical drawings, whereas for symbols, constraints like horizontal, vertical, collinear, semi- and quarter circular arcs are preferred over other constraints. Semi- and quarter circular arcs are enforced through horizontal and vertical alignment constraints of the center and endpoints.

To reduce the combinatorial explosion, we use the above results in selecting the constraint sets. We identified four rationales for creating the sets namely, Goldmeier's singulari-

ties, Gestalt principles of closure and smooth continuation, technical drawings and symbols. The first set uses only closure and smooth continuation principles as these are inherent in creating closed curves. Only those constraints that are used in obtaining these properties are selected. The system uses dynamic tolerances to determine valid constraints among the selected to be used in creating the set i.e., the cost associated with each constraint is less than a threshold. These tolerances are based on the lengths of the line segments and radii of arcs. For example, the cost for the coincident constraint between p0 and p7 in Figure 5(a) is less than one-tenth the sum of lengths of L0 and L5. We used three different factors for thresholds, namely one- fifth, one- tenth and one-fifteenth across all the constraints. Accordingly, three constraint sets are selected. A set is discarded immediately if it is similar to the previously created sets. We observed that using a dynamic tolerance improved the results over using a static threshold. The freehand stroke in Figure 5(a) is converted to as in Figure 5(b), by using the closure and smooth continuation principles, which are inferred as a set of geometric constraints - 'fixation' of the line segments and 'coincidence' constraints of the end points (constraint set-1 in Figure 6c), and solved for by using a geometry constraint solver. Similarly, constraints sets are created corresponding to the other three rationales. In all, a total of twelve constraint sets are generated.

4.3 Geometry Constraint Solver

The different constraint sets when solved, represent different alternatives of the input sketch. We have integrated LGS2D [LED09] library with our system and used it for constraint solving purposes. The core technology of LGS2D is a combination of symbolic and numerical methods for solving systems of geometrical constraints. The main symbolic method used in LGS2D is a variation of constraint graph analysis, based on abstract degree-of-freedom approach. A brief description of LGS2D and its capabilities are described in [LED09]. In addition, LGS2D can effectively determine and solve well-defined, under-defined and over-defined geometric models which are very useful for systems that involve freehand sketching [PR07b]. For under-defined problems, it finds a solution that is close to the original sketch configuration without unnecessary rotations and movements of geometrical objects and also by preserving the initial radii of arcs and circles.

4.4 Evaluation of Suggestions

Different constraint sets lead to various suggestions to be generated. However, not all might be unique and some suggestions may be inconsistent. Also, displaying all the suggestions as visual snapshots to the user can be overwhelming. Hence, we evaluate each plausible candidate and display only the relevant suggestions to the user. We use two measures for evaluating and distinguishing between suggestions – 1) Perimeter of the sketch, which is equal to the sum of lengths of all lines and arc lengths of arcs and circles. 2) Total cost incurred by the system to transform from the original constraint-unsolved representation to constraint-beautified representation. This total cost for a suggestion is calculated from the geometric constraints used to satisfy the sketch. The costs are of two types namely distance measures and angular measures. For comparison purposes, we convert the angular measures to distances by using the arc lengths property i.e. for horizontal and vertical constraints, the cost is equal to (length of line segment * sine of angle). For perpendicular and parallel constraints, it is equal to 0.5 * (sum of lengths of line segments * sine of angle). Two suggestions are same if their corresponding perimeters are almost equal. If the perimeter is greater or lesser than 1.5 times that of the original sketch, we discard the suggestion. We rank the different suggestions based on the total costs and how close is their perimeter value to that of the original sketch. After evaluation, the top candidates are displayed to the user as small thumbnails; see Figure 3(b) and (d). When the user hovers around an image using the pen/mouse, a bigger version of the image is shown and on clicking one of them, the sketch gets updated accordingly in the sketching area.

5. Results

We conducted a preliminary user study to evaluate the effectiveness of our proposed method. Our prototype was ported on to a PC with Wacom Cintiq 21UX LCD monitor. Eight students from mechanical engineering participated in this study and all of them were familiar with sketching aspects of CAD programs (like AutoCAD and Pro/Engineer) and hence were well aware of use of geometric constraints in making diagrams. Also, all of them were familiar with digitizing media, and have used especially Tablet PCs and (or) PDAs before but not Wacom Cintiq. First, we demonstrated the capabilities of our system i.e., initial processing of the freehand strokes – segmentation and recognition of geometric primitives and the generation of suggestions with a few examples and also its limitations i.e., the system recognizes only lines, arcs and circles and does not handle over-tracing (making several overlapping strokes, such that the strokes are perceived as a single object collectively). In addition, interaction techniques (like pulling gesture with the stylus, clicking on a critical point) [MR09] for correcting errors due to segmentation and recognition were also demonstrated. The students were given 15 minutes to get acquainted with the system and to clarify any questions they had.

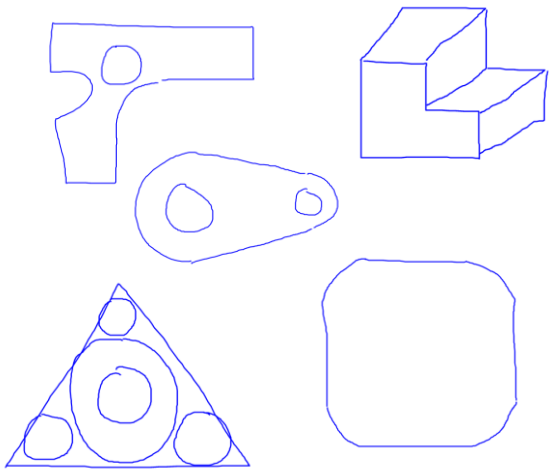


Figure 6: *A representative sample of freehand sketches collected from 40 sessions drawn by the participants.*

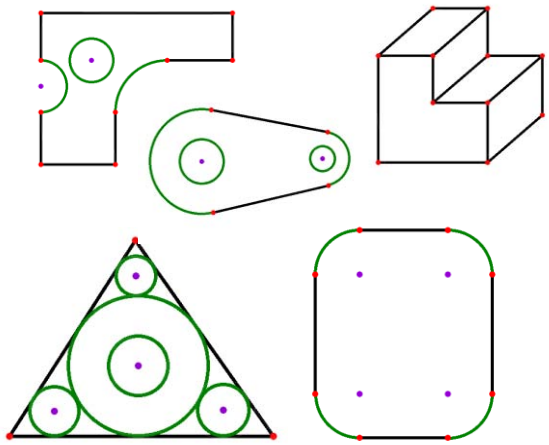


Figure 7: *Final beautified drawings of the freehand sketches.*

Next, the students were asked to sketch five objects. A representative sample of the freehand sketches drawn by the participants is shown in Figure 6 and of the final beautified versions in Figure 7. Through this study, we were mainly interested in finding out whether the system was able to generate relevant suggestions and if the suggestions matched the users intent. The participants were able to create all the drawings successfully. However, the number of interactions, steps needed and also the time taken by the users in achieving the final goal varied greatly. All of the participants reported that the system was easy to use and expressed a positive attitude towards drawing using freehand sketching. Majority of the comments came from the aspect of generation of suggestions. All the participants liked the idea of system generating multiple suggestions and found it useful and intriguing. The participants were very appreciative of the system inferring the constraints automatically and satisfying them simultaneously without the need for manually specifying them. However, there were a few instances where the system did not generate the intended result they had in mind. This was either due to 'Near misses' i.e., constraints being not inferred, (due to the cost of satisfying the constraint being over the threshold) or was not intended (wrongly inferred and satisfied). In such scenarios, the user had to delete the stroke and redraw it as there were no other editing techniques. We are investigating methods to address these problems. The current implementation of the system generates suggestions at a global level, i.e. the whole stroke/sketch. If a particular suggestion is not generated due to constraint non-inference, the user can use gestures to indicate where the geometry needs to be altered (locally) and the system can be made to provide only for that particular portion of the sketch. For example, if a coincident constraint is not inferred between a pair of points, the user can indicate it by drawing a closed loop gesture around the points. They can also use an over sketching gesture to snap the points [ZBLF08]. However, without explicitly switching modes, effectively disambiguating between strokes representing geometry and gesture is a research area in itself. The other improvements proposed by the participants was manipulation of already existing geometry i.e. capability for moving and resizing. This can be addressed as these aspects are related to the user interface elements of the system and not of the proposed method.

6. Conclusion

In this paper, we have developed a method to drive beautification of freehand sketches using geometric constraints. As freehand sketches are ambiguous in nature, correctly interpreting the visual scene the user had in mind becomes a difficult problem. To address this problem, we generated multiple alternatives (called 'suggestions') of the freehand input and presented them as visual snapshots from which the user can choose what he intended to draw. Further, suggestions help in speeding up the beautifying process, like by addressing over- and under- sketching cases through geometric constraints which typically require manual editing operations. Our system supports freehand strokes made up of multiple primitives like arcs and circles in addition to line segments. This allows the users to draw as natural as possible just as one would draw on paper. This suggestive system infers the geometric constraints implied in the sketches automatically and solves them simultaneously to generate multiple suggestions. We used results from psychological studies to determine the perceptually important constraints for beautification. A preliminary user study was conducted to determine the effectiveness and robustness of the proposed method. We are currently working to improve the usability of the interface by supporting various editing operations. We are also exploring the possibility of extending suggestions to three dimensional sketching.

7. Acknowledgements

This material is based upon work supported by the National Science Foundation Division of Information and Intelligent Systems (NSF IIS) under Grant No. 0535156. This work was done in collaboration with PARC (formerly Xerox PARC). We would like to acknowledge Dr. Eric Saund for his suggestions during this work. We would also like to acknowledge LEDAS Ltd., for providing us with LGS2D geometry constraint solver to create this application. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[Arn82] ARNHEIM R.: Art and Visual Perception, 1982.

[Bad00] BADROS G. J.: Extending interactive graphical applications with constraints (user interface). PhD thesis, 2000. Chair-Alan Borning.

[Bie88] BIER E. A.: Snap-Dragging: Interactive Geometric Design in Two and Three Dimensions. PhD thesis, EECS Department, University of California, Berkeley, May 1988.

[Gle92] GLEICHER M.: Briar: a constraint-based drawing program. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1992), ACM, pp. 661–662.

[Gol82] GOLDMEIER E.: The memory trace: its formation and its fate, 1982.

[IH01] IGARASHI T., HUGHES J. F.: A suggestive interface for 3d drawing. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2001), ACM, pp. 173–181.

[IKTM98] IGARASHI T., KAWACHIYA S., TANAKA H., MATSUOKA S.: Pegasus: a drawing system for rapid geometric design. In *CHI '98: CHI 98 conference summary on Human factors in computing systems* (New York, NY, USA, 1998), ACM, pp. 24–25.

[JVJ*04] JOÃO P. P., VASCO A.B., JOAQUIM A.J., NELSON F.S., TIAGO D.C., FERNANDO N.F.: Cascading recognizers for ambiguous calligraphic Interaction. In *Proceedings of EUROGRAPHICS workshop on Sketch-Based Interfaces and Modeling* (Grenoble, France, 2004), ACM, pp. 63-72.

[KF93] KURLANDER D., FEINER S.: Inferring constraints from multiple snapshots. *ACM Trans. Graph*. 12, 4 (1993), 277–304.

[KWL93] KARSENTY S., WEIKART C., LANDAY J. A.: Inferring graphical constraints with rockit. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems* (New York, NY, USA, 1993), ACM, p. 531.

[LED09] LEDAS LTD: LGS2D constraint solver - http://www.ledas.com/products/lgs2d/, accessed on feb 25th 2009

[LHBE97] L.EGGLI, H.CHINGYAO, B.BRUDERLIN, ELBER G.: Inferring 3d models from freehand sketches and constraints. *Computer*-Aided *Design 29* (February 1997), 101–112(12).

[LMM04] LANGBEIN F. C., MARSHALL A. D., MARTIN R. R.: Choosing consistent constraints for beautification of reverse engineered geometric models. *Computer-Aided Design 36*, 3 (2004), 261 – 278.

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P. A., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design galleries: a general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 389–400.

[MHA00] MANKOFF J., HUDSON S. E., ABOWD G. D.: Interaction techniques for ambiguity resolution in recognition-based interfaces. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2000), ACM, pp. 11–20.

[MR09] MURUGAPPAN S., RAMANI K.: FEAsy: A sketch-based interface integrating structural analysis in early design. To appear in *ASME Design Engineering Technical Conferences, Computers and Information Science in Engineering (ASME DETC CIE),* (2009), ASME.

[NCAC07] NAYA F., CONTERO M., ALEIXOS N., COMPANY P.: Parsketch: A sketch-based interface for a 2d parametric geometry editor. In *HCI (2) (2007)*, Jacko J. A., (Ed.), vol. 4551 of *Lecture Notes in Computer Science*, Springer, pp. 115–124.

[PR07a] PU J., RAMANI K.: Implicit geometric constraint detection in freehand sketches using relative shape histogram. In *SBIM'07: Proceedings of the 4th Eurograpics workshop on Sketch-based interfaces and modeling* (New York, NY, USA, 2007),ACM, pp. 107–113.

[PR07b] PU J., RAMANI K.: Priority-based geometric constraint satisfaction. *JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING* 7, 4 (July 2007), 322–329.

[Sau03] SAUND E.: Finding perceptually closed paths in sketches and drawings. *IEEE Trans. Pattern Anal. Mach. Intell*. 25, 4 (2003), 475–491.

[SFLM04] SAUND E., FLEET D., LARNER D., MAHONEY J.: Perceptually-supported image editing of text and graphics. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 728–728.

[SM94] SAUND E., MORAN T. P.: A perceptually-supported sketch editor. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1994), ACM, pp. 175–184.

[Sut64] SUTHERLAND I. E.: Sketch pad a man-machine graphical communication system. In *DAC '64: Proceedings of the SHARE design automation workshop* (New York, NY, USA,1964), ACM, pp. 6.329–6.346.

[TBSR04] TSANG S., BALAKRISHNAN R., SINGH K., RANJAN A.: A suggestive interface for image guided 3d sketching. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2004), ACM, pp. 591–598.

[VD06] VESELOVA O., DAVIS R.: Perceptually based learning of shape descriptions for sketch recognition. *In SIGGRAPH '06: ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), ACM, p. 28.

[Wer23] WERTHEIMER M.: A source book of gestalt psychology, 1923.

[ZBLF08] ZELEZNIK R. C., BRAGDON A., LIU C.-C., FORS BERG A.: Lineogrammer: creating diagrams by drawing. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology* (New York, NY, USA, 2008), ACM, pp. 161–170.

[ZL07] ZOU H., LEE Y.: Constraint-based beautification and dimensioning of 3d polyhedral models reconstructed from 2d sketches. *Computer-Aided Design* 39, 11 (2007), 1025 – 1036.