SKETCHBOARD: THE SIMPLE 3D MODELLING FROM ARCHITECTURAL SKETCH RECOGNITION

POJCHARA JATUPOJ
Department of Architecture, Faculty of Architecture, Chulalongkorn University, Bangkok, Thailand.
joel@mangobyte.com

Abstract. The main objective of this research was to study freehand architectural sketches and computer algorithms to develop a sketch recognition software and to apply software with basic 3D modelling. This research was conducted by collecting the data of sketch recognition and related software to analyst relevant information and to develop new software that would be practical for architects.

1. Introduction

Sketch is the most popular method that architects have always used to communicate their ideas. In the beginning of the architectural design process, architect mostly use a freehand sketch to develop a conceptual design and then elaborate it in the final stage on computer (Thidasiri, 2003). In the final stage of design, if there was an intention to change the design then architects had to go back to freehand sketch again and elaborate it again on computer. It is obviously a repeat process and should be considered as an obstacle to the architectural design process. If there were a software that could recognize a sketch, architects would be able to sketch a preliminary design directly on computer without going back and forth between paper and computer. Further, there would be no high learning curve like average CAD softwares.

Consequently, the idea resulted in the development of a sketch recognition software that should be helpful for architects to communicate their ideas and design process more efficiently. By the way, this research was not intended to replace a paper medium with this application but rather to create an option for architects and also to create more convenient ways for architects to use computer-aided design software.

2. Research and Development of SketchBoard

The research was conducted by collecting many types of relevant researches. The study of architectural sketches concluded that there were two basic types of line

sketching, namely geometric and freehand line sketching (Ching). Both types were varied by forms and styles of strokes but the most popular type used in architectural sketch was the geometric line sketch. Consequently, the development of sketch recognition software was based on this type of sketch. Further, the software development approach was referred on Filter-based Approaches (Jin et al., 2002) that focused on the development of filter to define a shape of sketch. The filter will define a shape by reference to x and y axis in a world coordinates.

The research and development of software was referred to many algorithms of sketch recognition such as, pattern recognition from Electronic Cocktail Napkin (Gross and Do), corner detection (Agar and Novins), parallel and length test of line (Fatos and BÜlent, 1990). This research used those algorithms to develop the specific sketch recognition software. The software was able to convert a sketch into a computer geometry and create a basic 3D modelling from a sketch at the end.

2.1. DEVELOPMENT OF INPUT SYSTEM FROM SKETCH

Referred to many researches, the electronic pen input system was more appropriate and convenient for sketch. The sketch recognition software development will use this type of input. Moreover, this input system has an important variable which is pen pressure.

Pen pressure was a software and hardware issue dependent on the specification of the manufacturer and software. This issue effected to the frequency of points. If the point density were too high it might too complicated to calculate and analyze. Meanwhile, this research were used the electronic pen that was able to adjust pen pressure at 70% level. This level was defined as an appropriate point density for analysis.

2.2. DEVELOPMENT OF ALGORITHM

The development of software started from algorithm development. It referred to many sketch recognition algorithms and related variables as follows.

2.21 continuing and overlapping line recognition
2.22 intersection line recognition
2.23 distant line recognition
2.24 extended line recognition
2.25 corner detection recognition
2.26 pattern recognition.

2.1.1. Continuing and overlapping line recognition

The program would receive coordinate of points from input system then it would calculate slope M with a slope equation.

M = (Y2 – Y1) / (X2 – X1)

M is slope of straight line that has start point (X1,Y1) and end point (X2,Y2).

Then comparing slope value with equation.

ABS(M2 - M1) <= PARALLEL

M1, M2 were slope of two selected straight lines for measuring parallel value.

The PARALLEL value was empirical defined as 5. If the difference of M2-M1 is more than PARALLEL then the algorithm will recognize a line as 2 different straight lines, as depicted in Figure 1.

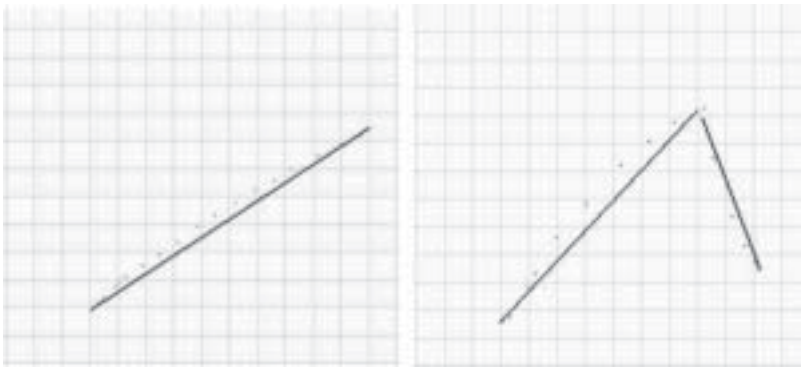

Figure 1. Pictures show straight line drawings that have proximity slope (left) software will recognize it as a same straight line. While a line sketch that has different slope value between two point (right) more than predefined variable software would recognize that they were not on the same straight lines.

2.2.2. intersection line recognition

The software would recognize intersection of line by using line coordinate to calculate intersection of Y axis or Y-intercept with intersection equation.

C = Y – MX

C is Y-Intercept value or intersection of Y axis value of straight line equation.

Then calculate slope of two lines with slope equation.

M = (Y2 – Y1) / (X2 – X1)

M is slope of straight line that has start point (X1,Y1) and end point (X2,Y2).

Then input value of C1 C2 M1 and M2 to the intersection equation. Though, a slope of both lines would not equal to 0 or none of line would parallel to X axis or Y axis.

X = (C2 – C1) / (M1 – M2);

Y = (M1 * X) + C1;

C1, C2 are Y- intercept value or Y- intersection value of straight line equation.

M1, M2 were slope of two straight lines.

X was intersection on X axis of two lines.

Y was intersection on X axis of two lines.

For each straight line that parallel on Y or X axis could input into Y1 variable in Y2=MX1+C2 equation to solve X1. Therefore, X or Y has to test to proof that it would be on the same segments as the intersection line to prove that both of lines were truly intersect

$t = (X - X1)/(X2 - X1)$

t is variable to proof x was on the same line segment (Figure 2).

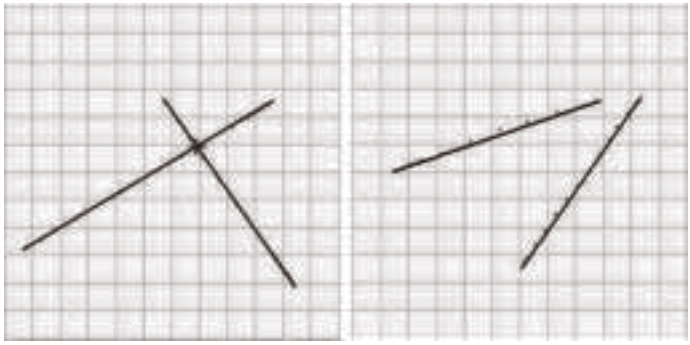

Figure 2. Accuracy of line intersection recognition algorithm (left) and also skip a process when the intersection was not a point on segment of any lines (right).

2.2.3. distant line recognition

The software would measure all end points between two lines and comparing them to other end points of lines to find if there was a distance between points that was not greater than EXTEND value (empirically predefined as 40) with the equation:

$t = (X - X1)/(X2 - X1)$

t is a variable to prove that X was on a line segment or not.

The algorithm was set to would work only if t is greater than 1, then the algorithm would move end points of lines to a closest intersection point.

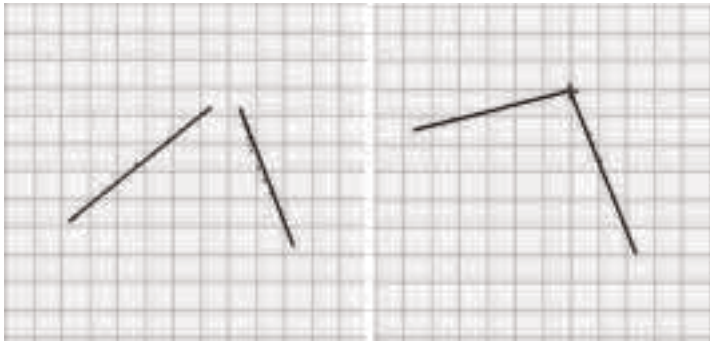

Figure 3. Distance between the end points that was greater than predefined value (left) and distance between the end points that was smaller than predefined value (right).

2.2.4. extended line recognition

The software would use all end points of both lines to recognize if there were end points of two lines that was nearer than 40 (empirical predefined) with the equation:

$$t = (X - X1)/(X2 - X1)$$

t is variable to determine that X was on a part of line segment or not,
t value would be between 0 and 1 (Figure 4).

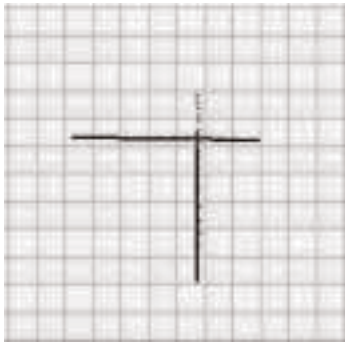

Figure 4. Result of the recognition of lines whose extend value were less than predefined value.

2.2.5. corner detection recognition

The software would count time of sketch between corners of drew shape. It would count and record time as 1/1000 second (millisecond) for each point, then it would start calculation to find the different values between each point and compared it with the value from the point that was drawn earlier.

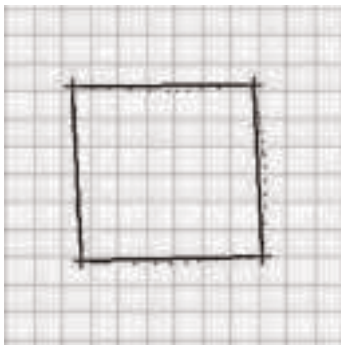

Figure 5. Software corner recognition.

2.2.6. pattern recognition

The algorithm was referred to a research of Electronic Cocktail Napkin (Mark and Ellen, 2002). The pattern recognition system, as shown in Figure 6, was developed by using grid as a filter system.

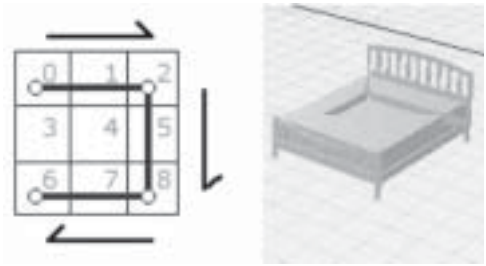

Figure 6. Sample of pattern recognition system.

TABLE 1. Example of Sketch Recognition Algorithms.

Algorithm	Sketch	Recognition
pattern recognition		
corner detection recognition		
extended line recognition, continuing and overlapping line recognition		
Bezier Curve Recognition (Experiment)		

2.3. DEVELOPMENT OF USER INTERFACE

The user interface of sketch recognition software was designed to support all earlier studied functions. It was also referred to a research, Free Form User Interface (Takeo, 1999), which pointed to an issue, an appropriate sketch user interface. Since the sketch should support quick creative idea such as critical thinking, user interface should be simple with not many working steps.

3. Conclusion

The result of this research revealed that the freehand sketch recognition software can help architects using a freehand sketch in their architectural designs as creating a basic 3D model more easily (Table 1). Furthermore, this algorithm can be applied to develop a freehand sketch recognition software for other complicated architectural 3D modelling in the future.

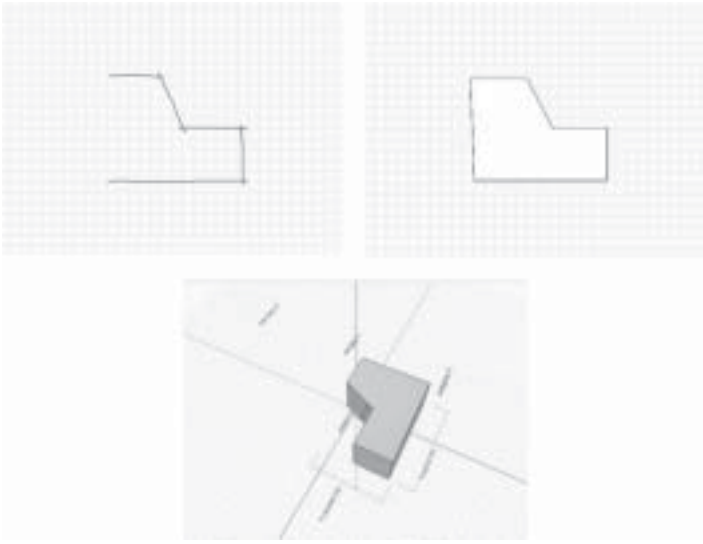

Figure 7. Sequent screen shots of recognition system.

Figure 8. Example of furniture design by SketchBoard.

Acknowledgements

This research was a masters degree thesis that presented to The Faculty of Architecture, Chulalongkorn University. I Thank to my Advisor, Asst. Prof. Kraweekrai Srihirun and co advisor Thidasiri Bhatrakarn for all support. Moreover, the research has been distributed through open source community, Sourceforge, at the website: http://sketchboard.sourceforge.net. It was freely distributed under GPL license.

References

Thidasiri, B. 2003. *Design plus Digital*, King Mongkut's Institute of Technlogy North Bangkok Press, Bangkok.