

学位論文

Analyses and Reduction of Operational Overhead
in Computer-Assisted Drawing

計算機支援の描画における
操作負荷の分析と削減

平成 10 年 1 月博士 (理学) 申請

東京大学大学院理学系研究科
情報科学専攻
河内谷 幸子

Abstract

Presently, computer-assisted drawing editors have sufficient capabilities to draw pictures with the precision required by the user. However, it often takes too long to draw simple pictures. This thesis focuses on this problem and proposes several solutions from a human-interface perspective.

First, from the analyses of drawing operations with conventional drawing editors, it is shown that “wasted time” such as trial-and-error and mis-operations accounts for 20–50% of total operation time. To clarify the causes of this, this thesis proposes an operation model of computer-assisted drawings. This model is characterized to have two kinds of pictures: concrete pictures and abstract pictures, where the former are visible and the latter are expressed by the combination of the editor's functions such as copy, move, rotate, etc.

The abstract-picture layer is difficult to handle and causes “psychological load” for the user. One condition for an ideal drawing editor is to reduce the handling of abstract pictures. This thesis proposes such an ideal drawing editor which avoids the use of abstract pictures. Three approaches are presented and their implementations are examined. Among them, the best drawing editor is the “candidate-selection method.”

In the candidate-selection method, a user first inputs a sketch by free-hand. The system recognizes it and displays possible multiple results. The user then selects a preferable candidate. In the displayed candidates, the geometrical constraints which are important for humans such as adjustment to horizontal and vertical lines, connection, parallelism, perpendicularity, and symmetry are included. The sketch recognition satisfies such perceptual constraints, and thus editing operations such as copy, move, and rotate become unnecessary. Therefore, the system can reduce the abstract operation of the user.

The second half of this thesis analyzes the effectiveness of the candidate-selection method. The operations of the prototype system “GIGA” are compared with those of popular object-oriented drawing editor, Canvas, and sketch-beautification drawing editor, SmartSketch. Three experiments are performed on various pictures: drawing for the first time, drawing after practice, and drawing a line several times. The results are analyzed in the following ways:

- First, the variations of drawing strategies are counted. There exist multiple ways of drawing with conventional drawing editors. On the contrary with GIGA, there is only one way of drawing each picture and the user does not have to choose a drawing strategy.
- Next, total (actual) operation steps and the ideal number of operations are compared. Both the total and ideal numbers of GIGA are smaller than those of conventional drawing editors.
- The time spent for each operation step is examined. An operation that takes more than 10 seconds is regarded as an operation with thinking. With GIGA, the number of such operations is reduced to less than 0.5% of the total number of operation steps.
- The total operation time is analyzed by classifying it into four layers: time of wasted operations, overhead of immature drawing, indispensable overhead time, and physical minimum time. Among them, wasted time and immature drawing time are caused by the psychological load. With GIGA, these times are far lower than those of conventional editors.
- The relation between the complexity (number of line elements) of pictures and their drawing time is investigated. In the conventional editors, there is no clear relation between the number of lines and operation time. On the other hand, a proportional relation can be observed with GIGA. This fact proves that operations with GIGA involve concrete pictures mostly.
- Finally, problems unique to the candidate-selection method are examined. In the

experiments with GIGA, the primary candidate is selected with a very high ratio (94.8%), and the ratio of re-sketching is only 2.4% of the total number of drawing operations. Therefore, the re-sketching interface and small chunk size will not cause severe problems.

By combining these good features, the total drawing time with GIGA is reduced to less than half that of conventional drawing editors. In conclusion, the candidate-selection method, which reduces the use of abstract pictures from the drawing operation, is shown by the quantitative analyses to have high efficiency. It is also concluded that the psychological load in drawing tasks is greatly reduced by the candidate-selection method.

This thesis has established quantitative analysis methods for drawing tasks to clarify the psychological-load problem, proposed an operation model that explains the problem as well as a new operation model to reduce the psychological load, and evaluated the candidate-selection method based on the new operation model.

論文要旨

現在広く普及している描画エディタは、必要な精度を満たす描画を行うための機能は充実しているが、操作時間の短さは必ずしも満足のものではない。本論文ではこの問題をヒューマンインタフェースの観点から分析・検証し、その解決法を提案する。

まず、既存の描画エディタを用いて描画操作を分析すると、試行錯誤やミス操作の時間が 20～50% 近く存在することがわかった。このような問題点の理由を追求するために、本論文ではまず描画操作のモデルを提案する。このモデルの特徴は、描画エディタを使った描画操作で扱われる図形表現には「具体的図形」と「抽象的図形」の 2 種類があると考えている点である。具体的図形は目で見ることのできる形を持った図形である。一方、「抽象的図形」は複写・回転というような描画エディタの機能の組み合わせとしての図形表現である。この抽象的図形の扱いが直観的でなくユーザに心理的な負担となることが、描画に時間がかかる原因だと考えられる。

理想の描画エディタの一つの条件は抽象的図形の扱いを減らし、ユーザに対する心理的負担を下げることである。本論文では、この理想へのアプローチとして 3 つの描画方式の提案・実装を述べる。その中で最も理想的な方式と考えられるのが「候補選択法」である。

候補選択法では、ユーザはまず手書きスケッチを入力する。システムはこれを認識し、認識結果を何通りか提示する。ユーザは候補の中から好みの結果を選択する。この時、システムが提示する認識候補の中には水平・垂直や、すでに描き込まれた図形との接続・平行・直交・対称などの人間にとって重要な幾何制約（知覚的制約）が考慮されている。このような知覚的制約を満たす認識が行われるため、複写・移動・回転のような編集操作は必要でなくなり、ユーザの抽象的図形の扱いが軽減される。

本論文の後半では、評価実験により候補選択法の有効性を示す。実装システム「GIGA」を用いた操作を、広く普及しているオブジェクト指向型描画エディタのCanvas、およびスケッチ整形型エディタのSmartSketchの場合と比較する。実験は、初めて描く、よく練習した後でミスなく描く、線を数回描く、の3種類をいくつかの絵について行い、この結果に基づいて以下のような分析を行う。

- まず、被験者によって描画戦略が何種類あったかを調査したところ、従来エディタでは複数の描き方がみられた。GIGAでは1つの絵についての描画戦略は1通りであり、描画の戦略をたてるのに迷わなくてすむ。
- 次に、各エディタにおける操作ステップ数を分析した。描画操作全体にかかった操作数と、ミスなしの理想の操作数を比較する。GIGAは従来エディタと比べて全体の操作数が半分以下である上に、理想と実際の差が少なく無駄な操作が少ないことがわかった。
- 上記の1操作ステップごとにかかる時間を調べたところ、CanvasやSmartSketchでは1操作に10秒以上かかる操作がかなりみられるのに対し、GIGAでは全操作の0.5%以下とほとんどなく、描画戦略を考える時間が少なくてすむことがわかった。
- さらに、操作時間を無駄な操作に使われた時間、不慣れな操作による負荷の時間、位置合わせ等の必要不可欠な時間、線を描くために最低限必要な物理的時間の4種類に分ける分析を行った。このうち、「無駄な操作」と「不慣れな操作」は描画作業における心理的な負担に起因する時間と考えられる。GIGAではこの部分の時間が従来エディタより少なくてすんでいる。
- 続いて、絵を構成する線の数と操作時間の関係を調べた。従来エディタでは両者に相関性が見られず、描画作業の複雑さは見た目の複雑さである線分の数では決まらないことがわかる。一方GIGAでは線分の数と操作時間に比例関係がみられる。このことから、GIGAでの描画操作のほとんどが具体的図

形の扱いであると考えることができる。

- 最後に ,GIGA の操作インタフェースにおける 候補選択が頻繁で作業を妨げないか」目的の候補が出なくて描き直しになることが多くなかないか」という疑問について調査した .第一候補に決まる率が94.8%と非常に高く,描き直しは 2.4%程度ですんでいる .そのため ,上記の点が描画の大きな妨げとなることはない .

これらの特徴により,GIGA による描画時間は従来の描画エディタの半分以下にまで縮められた .以上の結果をまとめると,抽象的図形の扱いを排除した候補選択法により,描画効率を上げられることが定量的に証明されたといえる . このことにより,本論文における操作モデルや理想の描画エディタの提案が正しいことが示された .

本論文の主要な貢献としては ,描画作業の定量的分析手法を確立し心理的負担の問題を明らかにしたこと,その問題点を説明するための操作モデルを提案し改良を示したこと,問題点の解決のために候補選択法を提案し,実装 評価したことがあげられる .

Table of Contents

1	Introduction	1
2	Problems in Current Drawing Editors	6
2.1	Targeted Picture Class.....	6
2.2	Problems in Object-Oriented Drawing Editors	7
2.2.1	Experimental Method.....	7
2.2.2	Wasted Operations.....	8
2.2.3	Possible Reasons of the Overhead	9
2.2.4	Summary of Problems in Object-Oriented Drawing Editors	10
2.3	Problems in Sketch-Beautification Type Drawing Editors	10
2.3.1	Experimental Method.....	11
2.3.2	Wasted Operations.....	11
2.3.3	Possible Reasons of the Overhead	13
2.3.4	Summary of Problems in Sketch-Beautification Drawing Editors.....	14
2.4	Issues to be Addressed for Improving Drawing Editors.....	15
2.4.1	Issues in Input Stage	15
2.4.2	Issues in Editing Stage	16
2.4.3	Toward More Detailed Analysis of Drawing Tasks.....	16
3	Detailed Analyses of the Existing Drawing Editors	18
3.1	Experimental Method	19
3.1.1	Experiments A, B, and C.....	20
3.1.2	Comparison of the Experimental Method with Conventional Task Analyses	20
3.2	Variation of Drawing Strategies.....	21

3.3	Details of the Wasted Operations	22
3.3.1	The Number of Operations.....	22
3.3.2	Input-Device Specific Problems in SmartSketch	23
3.3.3	Protrusion Cutting in SmartSketch.....	25
3.4	M/O/N (MENU/OBJ/NON) Analysis.....	27
3.5	Four-Layer Analysis	28
3.5.1	Wasted time.....	28
3.5.2	Overhead Time by Immature Drawing.....	29
3.5.3	Physical Minimum Time and Indispensable Overhead Time	30
3.5.4	Four-Layer Classification of Drawing Time	31
3.6	The Case of Drawing Complex Pictures.....	33
3.7	Discussion of Revealed Problems.....	36
4	Operation Model of Conventional Drawing Editors.....	37
4.1	Concrete Picture and Abstract Picture	37
4.2	Two-Layer Operation Model.....	38
4.3	Conventional Work for Improving the Drawing Efficiency.....	39
4.4	Relationship to Norman's Seven-Stage Model.....	41
4.5	Ideal Improvement of the Operation Model	45
5	Toward an Ideal Drawing Editor	46
5.1	Requirements for an Ideal Drawing Editor	47
5.2	Approach 1: Sketch-Annotation Method.....	47
5.2.1	The Method that Annotates Inputted Sketches	47
5.2.2	Prototype Implementation.....	49
5.2.3	Comparison to Conventional Sketch-Beautification Editors.....	54
5.2.4	Evaluation as an Ideal Approach	55
5.3	Approach 2: Dot-Masked Revision Sheet Method	55
5.3.1	Problems in Scanner-Input Drawing	56
5.3.2	Dot-Masked Revision Sheet Method.....	58

5.3.3	Prototype System “HIDES”	60
5.3.4	Evaluation of Image Quality and Processing Speed	68
5.3.5	Psychological Effect Caused by Waiting Time of Scanner Input	70
5.3.6	Evaluation as an Ideal Approach	70
5.4	Approach 3: Candidate-Selection Method.....	71
5.4.1	A New Drawing Method Using Concrete Pictures.....	71
5.4.2	Explanation Using a New Operation Model.....	72
5.5	Summary of Approaches to an Ideal Drawing Editor	74
6	The Candidate-Selection Method	75
6.1	Basic Ideas of the Candidate-Selection Method.....	75
6.2	Advantage of Human Perceptual Constraints	76
6.3	Prototype System “GIGA”	77
6.3.1	Actual Operation	78
6.3.2	System Structure	80
6.3.3	Generating and Solving Geometrical Constraints.....	82
6.4	Possible Problems of the Unique Interface	83
7	Evaluation of GIGA	84
7.1	Variation of Drawing Strategies.....	85
7.2	The Number of Wasted Operations	85
7.3	M/O/N Analysis.....	87
7.4	Four-Layer Analysis	89
7.5	Relation of Picture's Complexity and Its Drawing Time.....	92
7.6	The Case of Drawing Complex Pictures.....	95
7.7	Effect of the Chunk Size.....	97
7.8	Summary of Analyses	98
8	Related Work: Survey of Computer-Assisted Drawing Systems.....	99
8.1	Painting Tools	99

8.2	Object-Oriented Drawing Tools	100
8.2.1	Commercial Drawing Editors	100
8.2.2	Object-Oriented CAD Systems.....	101
8.2.3	Programming by Example	102
8.2.4	Education of Drawing Tools	102
8.3	Sketch-Beautification Systems for an Electronic Panel.....	102
8.3.1	2-Dimensional Sketch Systems.....	103
8.3.2	3-Dimensional Sketch Systems.....	103
8.4	Recognition Systems for Drawing on a Paper	103
8.4.1	Recognition of Formatted Diagram or Text	104
8.4.2	Recognition of Hand-Written Sketch or Text	104
8.4.3	Recognition Techniques.....	104
8.5	Efficiency of Electronic Pen Input.....	105
8.6	Conclusion of the Investigation of the Related Studies	106
9	Conclusions.....	107
9.1	Results	107
9.2	Directions of Future Research.....	110
10	Bibliography.....	112
11	Appendix A. Detailed Description of Drawing Strategies.....	132
11.1	(a) Perpendicular V-Shape	133
11.2	(b) Slope.....	134
11.3	(c) Isosceles Triangle	135
11.4	(d) Rectangle	136
11.5	(e) Rhombus.....	137
11.6	(f) Elbow-Shape	139
11.7	(g) Rectangle Placed on a Slope.....	142

List of Figures

Figure 1.1 Rectangle placed on a slope.....	1
Figure 2.1 Time of drawing the “rectangle on a slope” with Canvas.....	8
Figure 2.2 Operation steps of drawing the “rectangle on a slope” with Canvas.....	8
Figure 2.3 Time of drawing the “rectangle on a slope” with SmartSketch	12
Figure 2.4 Operation steps of drawing the “rectangle on a slope” with SmartSketch....	12
Figure 2.5 Example of “bumpy” line recognized by SmartSketch.....	13
Figure 3.1 Six simple pictures used for the experiments	19
Figure 3.2 Operation steps of drawing the simple pictures	22
Figure 3.3 Example of “protrusion-cutting” operation in SmartSketch.....	26
Figure 3.4 M/O/N analysis of drawing the simple pictures	28
Figure 3.5 M/O/N analysis of drawing the “rectangle on a slope”.....	28
Figure 3.6 Four layers in the actual drawing time.....	32
Figure 3.7 Four-layer analysis of drawing the simple pictures.....	32
Figure 3.8 Four-layer analysis of drawing the “rectangle on a slope”.....	32
Figure 3.9 More complex pictures	33
Figure 3.10 Time of drawing complex pictures	34
Figure 3.11 Strategies for drawing the leg part of the “human”.....	35
Figure 4.1 Two-layer operation model for conventional drawing editors.....	38
Figure 4.2 Norman's seven-stage model.....	42
Figure 4.3 Relation of Norman's model and two-layer model.....	43
Figure 5.1 Sample drawing of the sketch-annotation method	48
Figure 5.2 Selection handles in the editing mode.....	49
Figure 5.3 Three menu types for specifying annotation	50
Figure 5.4 Processing flow chart of the prototype sketch-annotation method.....	51

Figure 5.5 Algorithm for ellipse recognition	53
Figure 5.6 Control points used for ellipse recognition.....	53
Figure 5.7 Conventional method of hand-written diagram recognition system.....	56
Figure 5.8 New method of hand-written diagram recognition system.....	57
Figure 5.9 Example of dot-masked revision sheet	58
Figure 5.10 Processing flow of the dot-masked revision sheet method	59
Figure 5.11 Thinning algorithm in HIDES	61
Figure 5.12 Sample operation result of HIDES.....	64
Figure 5.13 Problems in dot-mask pattern	65
Figure 5.14 The case that dot elimination is impossible	65
Figure 5.15 Dot-mask pattern adopted in HIDES	66
Figure 5.16 Pixel judgment algorithm	67
Figure 5.17 Reduction and expansion algorithm	67
Figure 5.18 An ideal operation model.....	72
Figure 6.1 Example drawings with GIGA.....	77
Figure 6.2 Candidate-selection method, GIGA	78
Figure 6.3 Interaction of selection from multiple candidates, using slider	79
Figure 6.4 Processing flow of GIGA	81
Figure 7.1 Operation steps of drawing the simple pictures with GIGA.....	86
Figure 7.2 Comparison of operation steps of drawing the simple pictures among editors	86
Figure 7.3 Operation steps of drawing the “rectangle on a slope” with GIGA	86
Figure 7.4 Comparison of operation steps of drawing the “rectangle on a slope”	86
Figure 7.5 M/O/N analysis of drawing the simple pictures with GIGA.....	88
Figure 7.6 M/O/N analysis of drawing the “rectangle on a slope” with GIGA	88
Figure 7.7 Four-layer analysis of drawing the simple pictures with GIGA	91
Figure 7.8 Comparison of operation times of drawing the simple pictures among editors	91
Figure 7.9 Four-layer analysis of drawing the “rectangle on a slope” with GIGA	91
Figure 7.10 Comparison of operation times of drawing the “rectangle on a slope”	91
Figure 7.11 Relation between the number of line elements and its drawing time	92

Figure 7.12 Relation in expertised GIGA (Exp. B)	94
Figure 7.13 Four-layer analysis of each picture	94
Figure 7.14 Time of drawing complex pictures with GIGA.....	96
Figure 7.15 Comparison of operation times of drawing complex pictures.....	96

List of Tables

Table 3.1 Variation of drawing strategies	21
Table 3.2 Line success ratio with SmartSketch in drawing the simple pictures	24
Table 3.3 Direct drawing ratio with SmartSketch in drawing the simple pictures.....	24
Table 3.4 Selection success ratio with SmartSketch in drawing the simple pictures	24
Table 3.5 Problems with SmartSketch in drawing the “rectangle on a slope”.....	25
Table 3.6 Protrusion cutting ratio with SmartSketch.....	26
Table 3.7 Protrusion cutting ratio in drawing the “rectangle on a slope”	26
Table 3.8 Average time of drawing an arbitrary line.....	30
Table 5.1 Drawing speed of four lines	54
Table 5.2 Editing marks supported in HIDES	60
Table 7.1 Ratio of selected candidates in GIGA.....	97

1 Introduction

Various computer-assisted drawing editors have been developed and are widely used. One reason for their popularity is that drawing editors enable the user to draw pictures more accurately and quickly than with manual drawing using rulers and compasses. However, with many of these drawing editors, users often find it difficult to draw figures that seem to have a simple relation, such as a rectangle placed on a slope as shown in Figure 1.1.

Various researches have been conducted on more efficient methods of assisting drawing. Among them are researches on increasing the drawing functions and researches on improving the interface to enable a greater variety of constraints and macros to be specified [17, 127, 14]. The researches have resulted in, for example, “programming by example” which uses a drawing operation history to assist drawing [105, 115, 94], and sketch-beautification systems [168, 143]. The sketch-beautification systems using an electronic pen have been reported to increase the efficiency of input and improve the simplicity of operation [4, 51, 157, 118, 89, 85, 194, 81]. However, these conventional researches have focused mainly on improving the drawing functions, and little work has

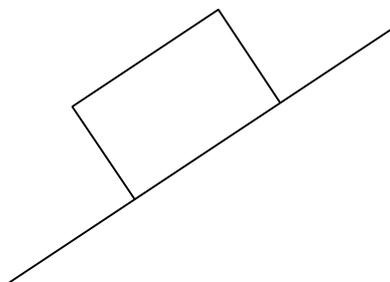


Figure 1.1 Rectangle placed on a slope

been done on cognitive considerations and evaluation of the problems inherent in drawing processes.

This thesis first attempts to clarify the basic cognitive problems of conventional drawing editors. For this purpose, an experiment, or a cognitive analysis [67, 165] of drawing processes, was first performed on a widely-used object-oriented drawing editor. The experiment revealed that the process of converting a picture to be drawn into the drawing editor functions (i.e., commands) is time-consuming. In this thesis, the process is called “command-planning.” The experiment also showed that much time is wasted by unnecessary operations (trial-and-error) based on incorrect command-planning. These “psychological overhead [165, 131]” are assumed to occur for the following reason. Drawing processes consist of the process of handling visible concrete pictures and the process of handling invisible abstract pictures such as the internal representation of a drawing editor. Users of drawing editors tend to feel that the conversion from the former concrete pictures to the latter abstract pictures and the latter picture itself are arduous.

The same experiment was carried out on a sketch-beautification type drawing editor. The experiment showed that this type of drawing editor, which seems to be convenient, actually has various overheads in its operation. The analysis revealed problems of the operations of the sketch-beautification type drawing editor such as that straight lines cannot be drawn easily with a mouse, objects cannot be selected easily with an electronic pen, and recognition based on certain geometrical constraints is required for input. In addition, the sketch-beautification type editor suffers the same problems in editing operations as the object-oriented editor because the two types use the same editor functions for editing drawn objects.

Based on the above experimental results, this thesis proposes an operation model for conventional drawing editors. This model consists of two functional layers: one layer handles concrete pictures and the other layer handles abstract pictures such as representations of geometrical constraints by combinations of tools and functions. Using this model, existing researches on drawing editors are analyzed and explained. The analysis concludes that it is important to focus on the concrete-picture layer in order to

reduce the time used for the command-planning and incorrect planning. This conclusion is unique but should be taken into account when improving drawing work and constructing an ideal drawing editor.

Next, this thesis explains three approaches to an ideal drawing editor. The first approach is a “sketch-annotation method [80, 81],” in which a picture is drawn by making a sketch and specifying its shape by additional annotation from a menu. Research is conducted on how to use the sketch-annotation method to solve problems of the conventional sketch-beautification type drawing editor such as “recognition based on certain geometrical constraints is required for input.”

The second approach, “dot-masked revision sheet method [75, 76],” uses a paper and pen (or pencil) which are the most natural tools used in manual drawing. Pictures manually drawn on paper are one typical example of “concrete pictures.” Research was conducted on how to assist the beautification and editing of those pictures. This method attempts to solve problems of existing computer-assisted drawing methods using paper.

The third approach is the proposal of a new drawing method that is fundamentally different from the conventional methods. Every existing approach to computer-assisted drawing editors has assumed that the human user performs the basic command-planning, and that the computer supports the user to perform the drawing operations based on this planning. However, eliminating this manual planning by the human user is can solve the fundamental problems of conventional drawing editors. The new method proposed in this thesis is a “candidate-selection method [83, 64, 65, 84].” In this method, the computer displays multiple pictures that can be created by a combination of general functions of an existing drawing editor as candidates based on a picture manually sketched by the user. Then, the user selects one of the displayed candidates. This method is innovative because it decreases the user's participation in the operations at the abstract-picture layer.

For each of the above three approaches, a prototype system was developed and evaluated. The conclusion was that the third approach was the most suitable for an ideal drawing editor that reduces the psychological load on the user.

The second half of this thesis describes the implementation and evaluation of the candidate-selection method, which is a new drawing method that enables the computer to handle all the abstract pictures and the user to handle concrete pictures in order to reduce the psychological overhead caused by command-planning. Because the drawing editor executes the command-planning, the user is freed from this time-consuming work. When the user makes a sketch of a desired picture, the computer displays actual candidate pictures for beautification by using a combination of existing drawing editor functions. The user simply selects a desired picture among the candidates displayed. The user need not be particularly aware of the drawing editor functions because the candidates are displayed in consideration of such constraints conforming to human perception such as the connection of intersecting points, parallelism, perpendicularity, symmetry, and so on. Using the implemented system named “GIGA,” experiments were carried out to evaluate the new drawing method and compare it with conventional drawing editors. The results of the experiment and evaluation proved that the new method reduces the time required for command-planning and incorrect planning, which are problems in conventional drawing editors.

The main contributions of this thesis are the proposal, implementation, and evaluation of a new drawing editor that reduces the psychological load on the user. The thesis also provides a detailed analysis of the drawing operations with existing drawing editors and identifies the problems. Such analysis and problem definition have not been addressed by existing researches. The proposed new drawing method is the successful product of the following research flow: analyze existing drawing editors, create drawing operation model, define the requirements of an ideal drawing editor, and propose a new approach for an ideal drawing editor. In other words, analyses of existing drawing editors and creation of a drawing operation model make it possible to propose an ideal drawing editor. Therefore, about half of this thesis describes the analyses of existing drawing editors and creation of the drawing operation model.

This thesis follows the order of the specific flow described above. Chapter 2 outlines the problems in existing drawing editors. Chapter 3 shows the result of detailed analysis.

Based on the analysis, Chapter 4 describes the creation of the drawing operation model. Chapter 5 describes the requirements of an ideal drawing editor and shows several approaches to achieving it. The candidate-selection method is shown to be best suited as the ideal drawing editor. Chapter 6 describes an implementation of the candidate-selection method named GIGA. Chapter 7 describes the analysis and evaluation of the implemented prototype system. Chapter 8 shows the result of an extensive investigation of related work to verify that the chosen problem and proposed drawing method are unique and have not been studied previously. Finally, Chapter 9 lists several conclusions and directions for future work.

2 Problems in Current Drawing Editors

This chapter describes a set of experiments that were performed to clarify problems in widely used drawing editors. Preparing figures and other explanatory illustrations in writing documents using existing drawing editors often takes much more time than expected, even for users who are familiar with the particular drawing editor. It is not exactly clear, however, why such extra time is needed. Therefore, drawing works using current drawing editors were examined to determine which part of the drawing process took up time, and to judge whether that time is really necessary.

2.1 Targeted Picture Class

The class of pictures targeted in this thesis is a type of pictures that can be expressed by the combination of drawing objects such as lines, circles, polygons, etc. This is the same class of pictures targeted by many object-oriented drawing editors and by conventional researches, and it is well suited for the type of explanatory pictures used in technical papers, presentations, etc. This class excludes unstructured sketches as well as CAD drawings and graphs requiring a high degree of precision.

Two well-known drawing editors for this class of pictures are Canvas and SmartSketch, which are typical object-oriented drawing editor and electronic-pen sketch-beautification drawing editor, respectively. This chapter examines the problems associated with these two representative drawing editors.

2.2 Problems in Object-Oriented Drawing Editors

One typical example of a computer-assisted drawing editor is an object-oriented type that treats drawing elements like lines, squares, and circles as units of operation (objects). In the actual use of this type of editor, however, it often happens that more time is needed to complete a drawing than originally expected. This section roughly determines the cause of this problem through a psychological experiment using the verbal protocol analysis [34].

2.2.1 Experimental Method

In the experiment, subjects were asked to draw the picture of the “rectangle placed on a slope” as shown in Figure 1.1 using Canvas 3.5, a widely used object-oriented drawing editor from Deneba Systems Inc. [29]. This picture was selected because it is a good example of an explanatory drawing and one typical picture that takes more time than expected. The subjects, which were 13 in total, were considered to be casual users [67] that had been using drawing editors¹ for at least three years.

Canvas functions were explained to the subjects before the experiment. While drawing, subjects were asked to announce their intended operation, and their drawing sessions were video recorded for later analysis. All operations were carried out using a mouse, and the functions used in drawing were limited to those generally used in object-oriented drawing editors and functions unique to Canvas were not used.

¹ The drawing editors that the subjects had actually been using in their work were Canvas or idraw. However, since explanation of basic Canvas operations was given beforehand, no difference in experimental results could be detected between Canvas users and idraw users.

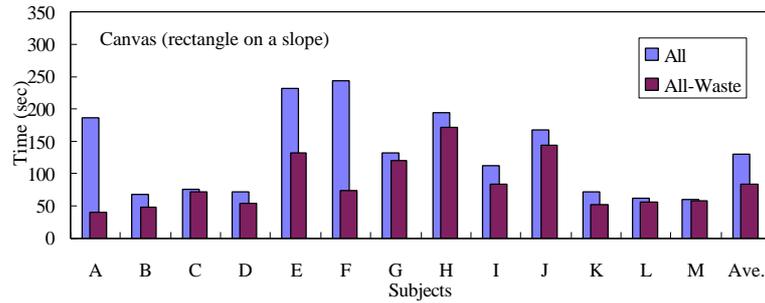


Figure 2.1 Time of drawing the “rectangle on a slope” with Canvas

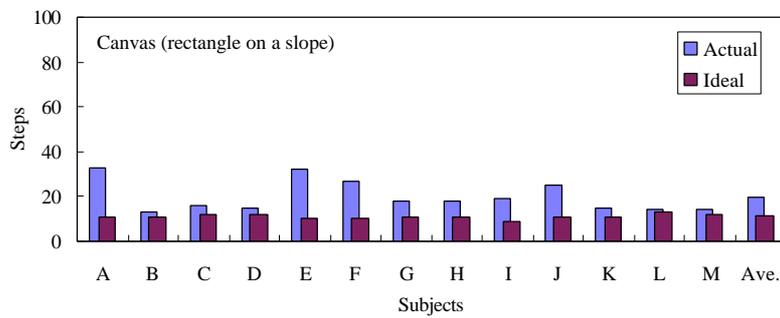


Figure 2.2 Operation steps of drawing the “rectangle on a slope” with Canvas

2.2.2 Wasted Operations

Using the video tapes of the drawing sessions, the ratio of wasted operation time in the total operation time was examined.

The wasted operation is defined as “unnecessary operation” to draw the picture and includes all trial-and-error and mistaken operations. Trial-and-error means to draw and delete some part of the picture that is eventually deleted and not used, and mistaken operation means unsuccessful editor operation such as mis-selection. It excludes those parts that the subject specifically announced as intending to delete as part of his/her drawing plan.

Results are shown in Figure 2.1, which shows the data of 13 subjects and their average. It was found that considerable amount (34% in average, 79% in worst subject) of total operation time was spent by unnecessary operations that do not contribute to the

result. Moreover, based on the speech made by the subjects, it was found that it takes a relatively long time to notice that one's drawing strategy was wrong. This extended the time of trial-and-error.

In addition to the operation time examination, the number of these wasted operation steps was also examined. Basically, the operation numbers were counted as follows. A menu operation is defined as pushing the mouse button in a menu area and releasing the button, and likewise, an object operation is defined as pushing the mouse button and releasing it for drawing or editing a specific object. For example, drawing one line with Canvas usually requires two operation steps: one menu operation to select the shape of object (line) and one object operation to draw the line by mouse dragging. Figure 2.2 compares actual number of operation steps with the number of ideal operation steps which excludes all wasted (unnecessary) operations. Note that the number of ideal operation steps may differ among subjects because various drawing strategies were used. As shown in the graph, many unnecessary operation steps (44% in average, 68% in worst subject) were included in drawing operation.

2.2.3 Possible Reasons of the Overhead

The above results show that in drawing a picture much time is wasted by many unnecessary operation steps such as trial-and-error and mistaken operations. The following describes a particular observed situation in which such wasted operation occurs. To draw the “rectangle on a slope” picture, one typical procedure is to first draw the slope portion as a combination of three line segments, draw a rectangle, and rotate and move the rectangle so that it makes contact with the slope. At this time, it may happen that the inclination of the slope and that of the rectangle do not match due to an inappropriate degree of rotation accuracy. There were many cases, however, in which the subjects could not find out the reason of the failure and simply continued the trial-and-error process for a long time.

Even if a user utilized a grid, time would also be used up in counting grid lines beforehand to successfully draw the intended lines. In the experiment, a variety of

drawing procedures were observed among the subjects; the method of drawing could be quite different depending on the person. This is one reason that the operation time differed very much among subjects.

By combining the above analysis with the speech made by the subjects, it was found that much time is taken up by using incorrect strategies, and that much time is needed to think up an appropriate strategy for using the functions of the drawing editor to draw the target picture. The drawing strategy was also found to differ among subjects, as mentioned above. Apparently, the existence of multiple drawing strategies means that time is taken up to select one of them.

2.2.4 Summary of Problems in Object-Oriented Drawing Editors

A drawing experiment was carried out using an object-oriented drawing editor, Canvas, to measure the wasted operation time and wasted operation steps. It was found that much time is spent using an incorrect strategy, and that much time is needed to determine a correct strategy for using the functions of the drawing editor to draw the desired picture. It was also found that multiple drawing strategies exist and time was spent in selecting a strategy.

2.3 Problems in Sketch-Beautification Type Drawing Editors

Sketch-beautification drawing editor is a type of drawing editor that accepts hand-written input from the user by an electronic pen and then beautifies that input. Though there are several commercial products, sketch-beautification editors are not so widely used as the usual object-oriented drawing editors. The question can be asked as to why they are not so popular with such an advanced input interface that naturally simulates drawing with pencil and paper. One reason offered for this is that there are not so many computers which equip electronic pens. Nevertheless, the mouse can be used for the operation, and this type of editor should find widespread use if it is really useful. This

section clarifies problems with the sketch-beautification type drawing editor through an experiment using the same verbal protocol method as in the previous section.

2.3.1 Experimental Method

The performed experiment was basically same as that of previous section. Subjects were asked to draw the “rectangle on a slope” picture (Figure 1.1) using SmartSketch 1.0, a well-known sketch-beautification type drawing editor from the FutureWave Software Inc. [42]. As with the Canvas experiment, total of 13 subjects were examined. In this experiment, however, no subjects had experience with this type of editor, and they were first given an explanation of SmartSketch and allowed to practice for 10 minutes. They were not, however, allowed to practice drawing the target picture at this time. In the experiment, subjects were asked to announce their intended operation, and their drawing sessions were video recorded for later analysis.

Although SmartSketch provides functions to directly draw rectangles, lines, and several other objects by selecting a menu as in object-oriented drawing editors, subjects were instructed to draw all such objects using the sketch-and-beautify function. This is because the purpose of the experiment was to examine the facility of sketch-beautification input. However, three editing functions of copy, move, rotate, and delete were allowed to be used.

Two sets of experiments were carried out, one with all operations being performed with the mouse, and the other with the electronic pen. The mouse-based experiment was carried out first and the pen-based experiment was performed three months later. No subject used SmartSketch during the three-month break, and many subjects had forgotten SmartSketch operations during this time. It was therefore considered that the experience in the mouse-based experiment had no influence on the pen-based experiment.

2.3.2 Wasted Operations

Based on the experiments, the same examinations of wasted operations have been done as the Canvas experiment.

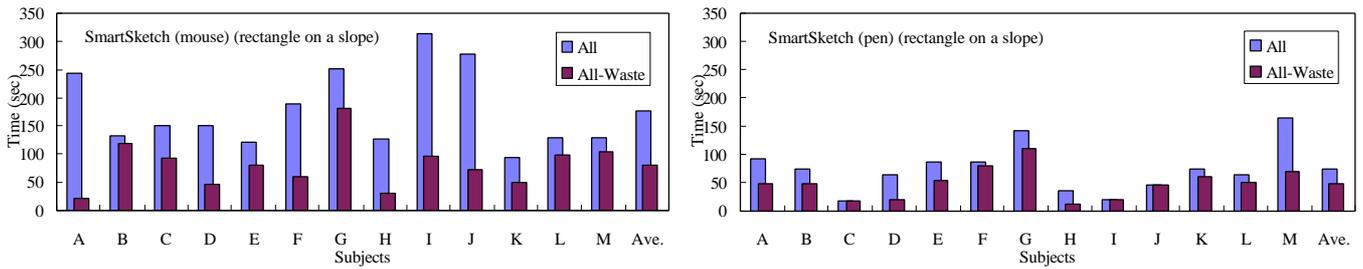


Figure 2.3 Time of drawing the “rectangle on a slope” with SmartSketch

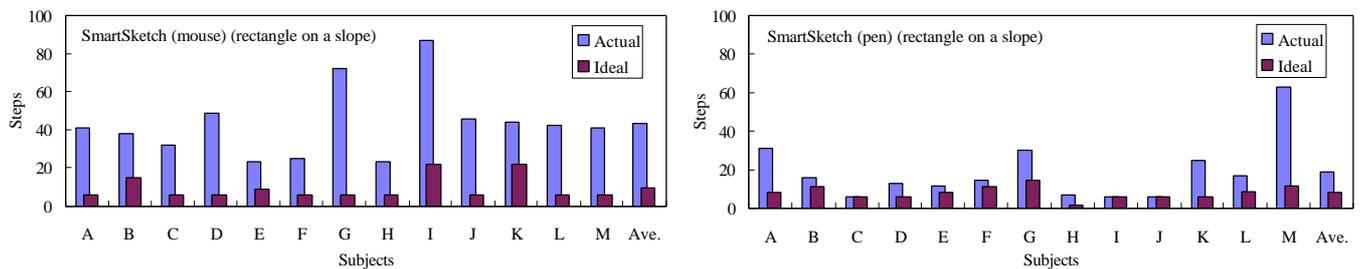


Figure 2.4 Operation steps of drawing the “rectangle on a slope” with SmartSketch

Figure 2.3 shows the ratio of wasted operation time of each subject in his total operation time. Here as well, considerable amount (54% with mouse and 34% with pen in average) of total operation time is wasted. Based on the speech made by the subjects, it takes a relatively long time to notice that he chose the wrong drawing strategy.

The number of wasted operation steps was also determined. The method used to count these operations was essentially the same as that in the Canvas experiment for both the mouse-based and pen-based experiments. In particular, for the electronic pen, a menu operation is defined as touching the screen with the pen in a menu area and then releasing the pen, and an object operation is defined as touching the screen with the pen in a drawing area and then releasing it. SmartSketch has two operation modes: input mode and editing mode. In input mode, the user can input objects by sketching with the electronic pen and the system automatically beautifies that input. In editing mode, the user can perform the copy, move, rotate, and delete functions to the drawn objects. One menu operation is needed to switch between these two operation modes. On the basis of

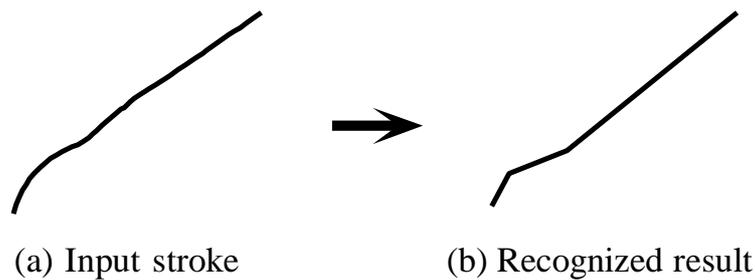


Figure 2.5 Example of “bumpy” line recognized by SmartSketch

the above, drawing and moving a line with SmartSketch, for example, would require a total of three operation steps: one object operation for drawing the line, one menu operation for switching the operation mode, and one object operation for actually moving the line.

Figure 2.4 compares actual number of operations recorded with that of ideal operation steps excluding all wasted (unnecessary) operations. It can be observed that many unnecessary operation steps (78% with mouse and 57% with pen in average) were included, especially with mouse.

2.3.3 Possible Reasons of the Overhead

On comparing the above analysis results with the speech made by subjects, the following problems were revealed.

First of all, it was found that several operations are difficult with SmartSketch. For example, when using the mouse, users had a hard time for drawing a straight line. Since the system faithfully beautifies input strokes, “bumpy” lines would frequently be recognized instead of perfectly straight lines as intended, as shown in Figure 2.5. Although this problem was relatively minor in the case of horizontal and vertical lines, it took about six or seven attempts to draw one diagonal line. On the other hand, it was easy to draw straight lines with the electronic pen and failed attempts were rare in the pen-based experiment. This is one reason that increased the wasted operations in the mouse-based experiment.

While it was easy to draw a straight line with the pen, it was revealed that object selection caused problems with a pen. In the pen-based experiment, the users may fail two or three times before successfully selecting a particular object for editing. This is mainly caused by the parallax between the pen point and the cursor. When using the mouse, failure of object selection was rare.

To support easily drawing vertex points, SmartSketch includes an “intersection processing function.” If a user inputs two line segments whose endpoints are very close to each other, SmartSketch shifts the endpoint of the second line so that it matches the endpoint of the first line (where “first” and “second” refers to the order of line input). However, in actually, it is very difficult to input two line segments so that their endpoints are “very close” to each other. As a consequence, it was often observed that users would draw two line segments so that they intersect and then delete unneeded parts to make endpoints match. This drawing style increased the unnecessary drawing operation. And it also increased the variation of drawing strategies.

To draw a picture having geometrical constraints like the one used in the experiment, the three editing functions of copy, move, rotate, and delete are almost indispensable. As a result, the same kind of problem that occurs with object-oriented drawing editors also occurs here. Specifically, much time is spent thinking up a strategy to effectively combine editor's functions like copy, move, and rotate to draw the target picture. Much time is also spent because of incorrect strategy.

2.3.4 Summary of Problems in Sketch-Beautification Drawing Editors

Several problems exist in the drawing operations associated with the SmartSketch sketch-beautification type drawing editor. With the mouse, it is difficult to draw diagonal lines, and with the electronic pen, it is difficult to select objects. Moreover, because it is difficult to draw two line segments having a common endpoint, users would rather draw two intersecting line segments and then delete unneeded protruding parts. All of these problems increase operation steps. In these respects, sketch-beautification drawing

editors are inferior to object-oriented drawing editors, and such problems are probably one major factor in the lack of popularity of the sketch-beautification editors.

Even assuming that straight lines can be drawn and that objects can be selected easily, the same editing functions as found in object-oriented drawing editors must be used to draw a picture with geometrical constraints. Sketch-beautification drawing editors therefore exhibit problems similar to those of object-oriented drawing editors. In short, much time is spent using the incorrect strategy, and much time is needed to think up a correct strategy for using editor functions to draw the target picture.

2.4 Issues to be Addressed for Improving Drawing Editors

Based on the discussions in previous sections, this section summarizes the issues in drawing editors by dividing the drawing operation into input stage and editing stage.

2.4.1 Issues in Input Stage

In object-oriented drawing editors, the shape of object desired is first selected from a menu and then drawn with a mouse. In this type of object-oriented drawing, an object is formed not by tracing its shape but rather by adjusting the selected object using a rubber band. Such operations are not natural compared to the drawing with pencil and paper.

On the other hand, the manner of inputting with sketch-beautification drawing editors is the same as that with pencil and paper. Menu selection is unnecessary and the trace of a target picture can directly be used for the input. Sketch-beautification drawing editors should therefore be far superior for input from an intuitive point of view.

However, it was found that this type of editor is inferior to the object-oriented type because of problems associated with the drawing of diagonal lines with a mouse, unnecessary delete operations, etc. Sketch-beautification drawing editors require beautification techniques that take into account frequently occurring geometric constraints like straight lines and endpoint matching.

2.4.2 Issues in Editing Stage

The operations involved in this stage are the same for either type of editor. In other words, the target picture must be eventually converted into some sequence of editing functions. If this conversion stage cannot be improved, overhead time in using a drawing editor might not be reduced. Moreover, with existing editors, various strategies exist for drawing one picture, which means that some confusion may occur in selecting a drawing strategy.

For sketch-beautification drawing editors, a more appropriate method of selecting objects by electronic pen is also needed.

2.4.3 Toward More Detailed Analysis of Drawing Tasks

In the above way, drawing operations using conventional drawing editors must proceed while keeping in mind two kinds of pictures: “concrete pictures” that are visible actual pictures, and invisible “abstract pictures” expressed by a symbolic sequence combining object types and editing functions like copy, move, and rotate.

In the case of object-oriented drawing editors, it is assumed that the frequent use of the abstract pictures in both input and editing make this kind of drawing difficult to understand and consequently time consuming.

In the case of sketch-beautification type drawing editors, pictures can be inputted as a concrete picture in the input stage. But unless beautification functions that take into account various geometric constraints are implemented, such input can hardly be used. Moreover, in the editing stage, this kind of editor is same as the object-oriented type, which means that editing must be performed in considering two types of pictures.

On the basis of the above discussions, following issues are raised up to pursuit the fundamental reason of the problems. To begin with, various types of operation time should be measured in more detail. For such detailed analysis, the “rectangle on a slope” picture examined in this chapter is considered to be too complex. Because too much deviations were observed in actual operation time and wasted operations among subjects and adopted drawing strategies. Experimental data should therefore be collected using a

picture a bit simpler than the one used in this chapter. In this manner, it becomes possible to construct an operation model of the drawing task and to improve it to reduce the various overheads found in this chapter. These issues will be discussed from the next chapter.

3 Detailed Analyses of the Existing Drawing Editors

In this chapter, drawing operations with existing drawing editors are analyzed more deeply and precisely, using six pictures that is simpler than the “rectangle on a slope” picture. The analysis was performed by adopting the following four methods in a quantitative manner.

1. The drawing strategies are examined for each picture, and the number of subjects who chose the strategy is counted.
2. Wasted operations are analyzed in detail. “The ratio of wasted operations” and “the problems with the mouse, pen, and protrusion cutting operations in the use of a sketch-beautification type drawing editor” which were described in Chapter 2 are ascertained using simple pictures.
3. For M/O/N (MENU/OBJ/NON) analysis, operation steps are classified into the “menu selection (MENU),” “object handling (OBJ),” and “others (NON)” for each drawing editor. Distribution of times for individual operation is examined.
4. By four-layer classification analysis, the operation time is separated into four layers: overhead time by wasted operations, overhead time by immature drawing, indispensable time for alignment, and physical minimum time.

To classify the operation time into four, two additional experiments are introduced. First, Each participant (subject) is requested to draw a given pictures once more after being skilled in the specific drawing. Further, the time to draw a simple straight line is measured.

Additionally, more complex pictures than the “rectangle on a slope” are analyzed globally as supplementary data.

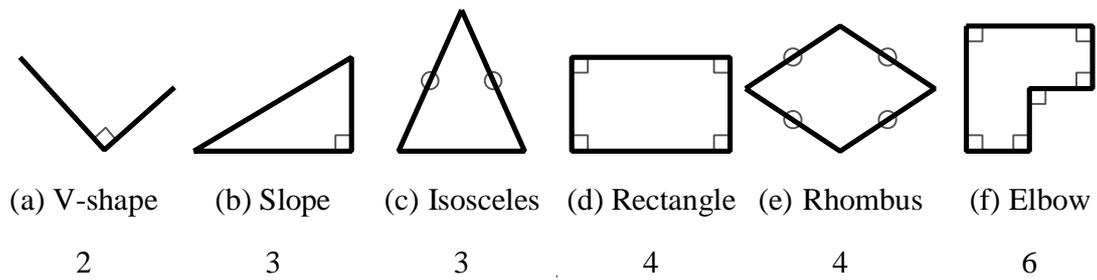


Figure 3.1 Six simple pictures used for the experiments

3.1 Experimental Method

The pictures used in the drawing experiment are total six as shown in Figure 3.1: (a) perpendicular V-shape, (b) slope, (c) isosceles triangle, (d) rectangle, (e) rhombus, and (f) elbow-shape. Their structures are all simpler than that of the “(g) rectangle placed on a slope” used in Chapter 2. In the figure, gray marks show the constraints that must be satisfied and number under each picture indicates its number of line elements. These need not be drawn in the experiments.

Experiments were carried out by the same 13 subjects and who participated in the experiment of the “rectangle on a slope” in Chapter 2. Canvas and SmartSketch were examined. For SmartSketch, both cases of using the mouse and the electronic pen were evaluated. Functions of Canvas and SmartSketch were explained to the subjects before the experiment. But functions used in the experiments were limited to basic functions that are common in conventional drawing editors, and functions unique to Canvas or SmartSketch were not used. SmartSketch supports the object-oriented input such as with selecting menu and drawing an object by rubber-band like Canvas. However, only sketching input was used to examine the facility of sketch-beautification. Note that editing functions such as copy, move, rotate, and delete were allowed to be used with SmartSketch.

3.1.1 Experiments A, B, and C

For each subject and editor, following experiments were carried out:

Exp. A First, subjects were asked to draw the six simple pictures with verbally announcing their intention. The drawing sessions were video recorded for later analysis.

Exp. B Next, they were asked to watch the video of Exp. A, understand the drawing procedure used by themselves, and re-draw those pictures *as fast as possible* with no wasted operation or drawing-strategy planning.

Exp. A is basically same as the experiment used in Chapter 2 except the target pictures are different. It can be considered that the drawing time in Exp. A contains the time for planning the drawing strategy and trial-and-error, while such kind of psychological overhead time is excluded in Exp. B. In addition, to investigate the basic drawing speed of each subject, following examination was also held:

Exp. C Subjects were asked to draw several arbitrary lines on the screen by both mouse and pen.

3.1.2 Comparison of the Experimental Method with Conventional Task Analyses

The appropriateness of the series of above experiments and analyses is taken into consideration. In conventional work-analysis models such as the KLM [24] and GOMS [25], their main target is to evaluate the result of work done by *expert* users. Therefore, the time taken for the above-mentioned psychological load, which can often be seen in general non-expert users, is defined as zero or a very small constant value, and the problems with the drawing editors pointed out in the previous chapter have not yet been clarified using these conventional work-analysis models.

Suchman's situation-dependent approach [165] is one famous task-analysis model which takes into account the existence of beginner users and operation errors, and discusses the psychological (cognitive) load. It can be considered that our experimental

method supports and analyzes the Suchman's approach, and specializes it for applying one specific task of picture drawing.

3.2 Variation of Drawing Strategies

First of all, on Exp. A, the variation of drawing strategies among subjects was counted. If the difference of two drawing procedures is only the drawing order, those procedures are regarded as a same strategy. For example, for drawing the perpendicular V-shape picture (a) with Canvas in the experiment, following three kinds of drawing strategies were observed:

1. Draw the picture as one multiple-line object (4 subjects)
2. Draw the two lines sequentially (2 subjects)
3. Draw an L-shape as a multiple-line object and rotate it (1 subject)
4. Draw a line, duplicate it, then rotate and move (6 subjects)

The number inside parentheses shows the subjects who used the strategy. In this case, the variation of drawing strategies is counted as four.

Table 3.1 shows the counting on each drawing editor. The detailed strategies actually used to draw each picture in Exp. A are shown in Appendix A. There exist multiple ways of drawing with Canvas. With SmartSketch, more numbers of variation are observed because of the “protrusion-cutting” operation in drawing vertex, which will be analyzed,

Table 3.1 Variation of drawing strategies

	Canvas	SmartSketch (mouse)	SmartSketch (pen)
(a) V-shape	4	6	6
(b) Slope	2	5	6
(c) Isosceles	3	3	3
(d) Rectangle	1	3	1
(e) Rhombus	5	5	4
(f) Elbow	4	10	9
Average	3.2	5.3	4.8

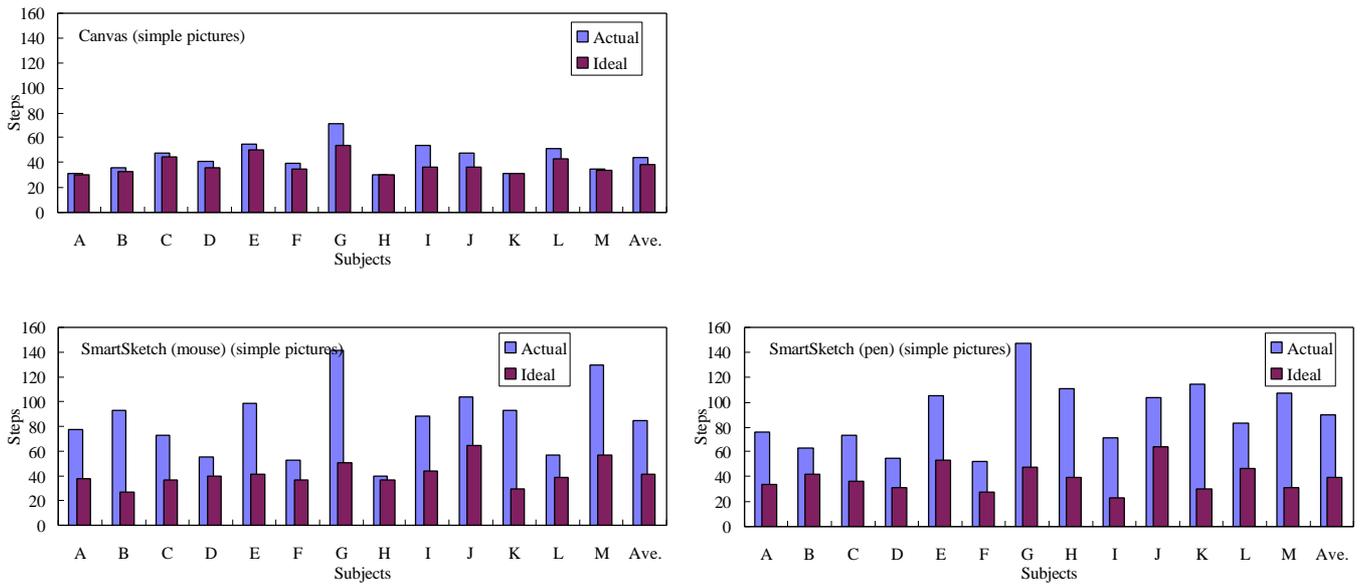


Figure 3.2 Operation steps of drawing the simple pictures

in Section 3.3.3. From the result, it can be concluded that there exists the load for considering the strategy when drawing with Canvas and SmartSketch.

3.3 Details of the Wasted Operations

In this section, the number of operations is analyzed in detail. The analysis of “the number of operation steps” shown in Chapter 2 is also applied to the six pictures. A number of problems unique with the sketch-beautification type drawing editor described in Chapter 2 are also further examined.

3.3.1 The Number of Operations

Figure 3.2 shows comparisons between the ideal number of operations and the number of actually executed operations by each subject to draw the six simple pictures.² One operation step is basically defined as one mouse-button push and release, which

² Results of the “rectangle on a slope” have already been shown in Figure 2.2 and Figure 2.4.

corresponds to one menu operation, one object drawing, one object selection. The method for counting the number of actual and ideal operations is the same as that explained in Chapter 2. The number of operations in Exp. A means the actual number of operations, and the ideal number of operations is gained by subtracting the wasted operations from actual number of operations. The ideal number of operations can also be counted from Exp. B because the drawing procedures used in Exp. B were performed with no wasted operation. Note that “the number of operations without wasted operations in Exp. A” is equal to “the number of operations in Exp. B,” while “the time of operations without wasted operations in Exp. A” is not equal to “the time of operations in Exp. B.”

Even when drawing a picture simpler than “the rectangle on a slope,” there existed much deviation of operation steps among subjects. Since there are a number of strategies for drawing a picture, a certain variation could be observed in the number of ideal operations.

Even for the simple pictures, there still remained wasted operation steps. Especially with SmartSketch, the ratio of wasted operation steps was very high. The reason of this phenomenon will be discussed in the following subsections.

3.3.2 Input-Device Specific Problems in SmartSketch

In Chapter 2, it was pointed out that with SmartSketch it is hard to draw a straight line with the mouse, and to select an object with the pen. In this subsection, these problems are analyzed in detail. The result of Exp. A for drawing six simple pictures with SmartSketch was used for the analysis.

Table 3.2 shows the success percentages in drawing straight lines using a mouse and a pen, respectively. The failure in drawing a straight line is such a case that despite the user's intention of drawing a straight line, the resultant sketch is beautified into the connection of a plurality of straight lines as shown in Figure 2.5. The success percentage is defined by the following expression:

Table 3.2 Line success ratio with SmartSketch in drawing the simple pictures

	A	B	C	D	E	F	G	H	I	J	K	L	M	Average
Mouse	58.8%	75.0%	80.0%	87.5%	61.5%	81.8%	54.5%	100 %	81.3%	48.3%	61.1%	78.3%	73.3%	71.7%
Pen	88.2%	100 %	100 %	100 %	92.3%	100 %	100 %	71.4%	100 %	100 %	100 %	81.8%	78.6%	97.2%

Table 3.3 Direct drawing ratio with SmartSketch in drawing the simple pictures

	A	B	C	D	E	F	G	H	I	J	K	L	M	Average
Mouse	81.8%	22.7%	77.3%	81.8%	45.5%	72.7%	31.8%	81.8%	68.2%	86.4%	81.8%	72.7%	59.1%	66.4%
Pen	72.7%	81.8%	77.3%	81.8%	50.0%	72.7%	54.5%	81.8%	77.3%	95.5%	50.0%	54.5%	72.7%	71.0%

Table 3.4 Selection success ratio with SmartSketch in drawing the simple pictures

	A	B	C	D	E	F	G	H	I	J	K	L	M	Average
Mouse	100 %	94.1%	100 %	100 %	100 %	91.7%	94.2%	100 %	96.3%	100 %	97.7%	100 %	97.9%	97.2%
Pen	68.4%	81.8%	no use	46.2%	63.6%	94.7%	66.0%	40.5%	45.5%	83.3%	62.1%	80.0%	47.5%	62.7%

$$(\text{Line success ratio}) = \frac{(\text{the number of successfully drawn straight lines})}{(\text{total number of line-drawing attempts})}$$

The average success percentage in drawing a straight line with the mouse was 71.7%, and that by the pen was 97.2%. This proves that the use of the SmartSketch drawing editor makes it more difficult to draw a straight line by using the mouse.

Therefore, most of the subjects drew lines by copying another well drawn line instead of drawing directly. Table 3.3 shows the direct drawing ratio defined by the following expression:

$$(\text{Direct drawing ratio}) = \frac{(\text{lines drawn by direct stroke input})}{(\text{total number of lines})}$$

where total number of lines are fixed in each picture as shown in Figure 3.1. For example, the total line number of isosceles V-shape (a) is 2, rectangle (d) is 4, and so on. More than 29% of lines were inputted by as the abstract pictures using copy and rotate function. It can be concluded that the advantage of sketching input, using concrete pictures, was not effectively used.

Table 3.5 Problems with SmartSketch in drawing the “rectangle on a slope”

	Line success ratio	Direct drawing ratio	Selection success ratio
Mouse	46.1%	71.8%	100.0%
Pen	95.5%	57.7%	59.4%

Next, the success percentage in the selection of an object by using the mouse and that by using the pen are respectively shown in Table 3.4. The failure in selection is such a case that despite the user's intention of selecting a particular object, the intended object could not be selected. The success percentage is defined with the following expression:

$$(\text{Selection success ratio}) = \frac{(\text{the number of successful selection operation})}{(\text{total number of selection attempts})}$$

The success percentage in selecting an object with the mouse was 97.2%, and that with the pen was 62.7%. This result proves that the use of the SmartSketch drawing editor with the pen makes it more difficult to select an object correctly.

It is just because the inadequate design of interface, which did not consider the “parallax” in pen-based systems well. That is, it is too narrow for pen interface to select a line which has normally 0.5–0.75 points width in general documents. In general, there are few troubles in the operation of selecting by pen in many other pen-based existing systems because the selection area such as icons or menus are large enough. Some improvements using the exiting techniques for parallax can solve this problem.

The accumulated result in more complex picture “rectangle on a slope” is presented in Table 3.5. The same tendency can be observed in drawing more complex picture.

3.3.3 Protrusion Cutting in SmartSketch

There was another problem with the SmartSketch drawing editor. That came from the “protrusion cutting” operation for drawing a vertex. The protrusion cutting operation is as follows.

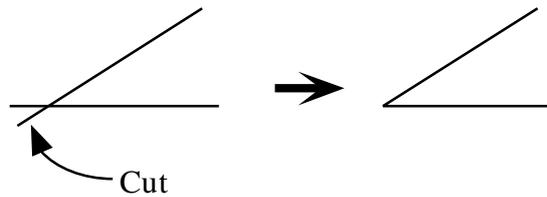


Figure 3.3 Example of “protrusion-cutting” operation in SmartSketch

With SmartSketch, two typical methods were observed for drawing two straight lines that share one endpoint (vertex). One method is to draw two lines by one stroke operation. The other is to draw two straight lines crossing each other, and then cut unnecessary segments (protrusion) of those lines as shown in Figure 3.3.

$$(\text{Protrusion cutting ratio}) = \frac{(\text{the number of vertices drawn by protrusion cutting})}{(\text{total number of vertices})}$$

where total numbers of vertices are fixed in each picture. For example, the total vertex number of V-shape (a) is 1, rectangle (d) is 4, and so on.

How often the protrusion cutting operations were executed with mouse and pen are shown in Table 3.6. The results prove that there are very many protrusion cutting operations in the entire drawing operation. The average percentage of the use of the protrusion-cutting operation reached 10–20%. Such a protrusion-cutting operation is

Table 3.6 Protrusion cutting ratio with SmartSketch

	A	B	C	D	E	F	G	H	I	J	K	L	M	Average
Mouse	9.5%	42.9%	19.0%	9.5%	9.5%	0.0%	38.1%	0.0%	19.0%	0.0%	42.9%	0.0%	76.2%	20.5%
Pen	38.1%	9.5%	0.0%	0.0%	9.5%	0.0%	42.9%	0.0%	14.3%	14.3%	14.3%	0.0%	33.3%	13.6%

Table 3.7 Protrusion cutting ratio in drawing the “rectangle on a slope”

	Protrusion cutting ratio
Mouse	41.8%
Pen	17.6%

essentially an unnecessary operation which increases the operation steps and time.

Such a tendency happened more frequently in drawing more complex picture “rectangle on a slope,” as shown in Table 3.7.

3.4 M/O/N (MENU/OBJ/NON) Analysis

Next, the length of each individual operation step was analyzed. The operation steps were classified into menu selection and object handling. The time between these operation steps was also examined. Therefore, the following three types of times were measured:

- MENU the time for selecting a command from the menu
- OBJ the time for directly handling an object (the time for object drawing, aligning, etc.)
- NON the time used between the above MENU time and OBJ time (the time for moving the mouse pointer, the time for doing nothing, etc.)

Figure 3.4 shows the result of analyzing the time required for each operation step with respect to MENU, OBJ, and NON (data are accumulated for all the subjects and six pictures). The result of more complex picture “rectangle on a slope” is also presented in Figure 3.5.

It is specifically clarified that the drawing editor functions are frequently used because MENU operations are highly used. It is also observed that even the same type of drawing operations take a widely varying periods of time ranging from one second to more than 10 seconds. The result of analyzing the verbal contents where each step takes a long time indicated that the subject was confused by how the function of the drawing editor should be best used to draw the target picture. Examples of such verbal contents actually observed were as follows — “What should be done next, ummm ...,” “The point is to decide whether the rotation should be done now or later,” Such tendencies happen more frequently in drawing more complex picture “rectangle on a slope.”

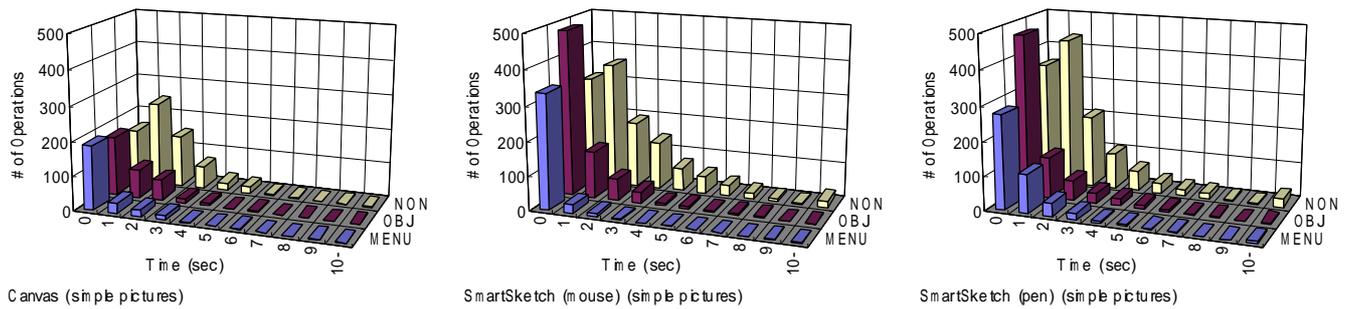


Figure 3.4 M/O/N analysis of drawing the simple pictures

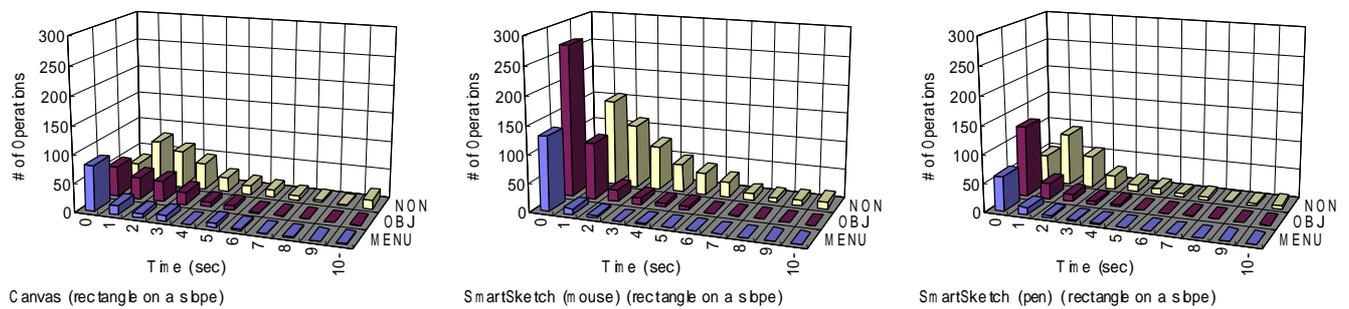


Figure 3.5 M/O/N analysis of drawing the “rectangle on a slope”

3.5 Four-Layer Analysis

In Chapter 2, only the “wasted operation time” was separated from the total drawing time. In this section, the drawing operation time is analyzed more deeply. It can be considered that total operation time in Exp. A (T_A) is the actual time to draw the pictures at the first time, which include all of trial-and-errors, psychological overhead for strategy planning, etc. This time was separated into four layers.

3.5.1 Wasted time

The drawing procedures in Exp. A were investigated and the time wasted by trial-and-error or mistaken operations (T_w) was extracted. The definition of wasted operation

is same as that used in Chapter 2. It includes all unnecessary operations to draw the target picture with minimal number of steps.

3.5.2 Overhead Time by Immature Drawing

It can be considered that psychological overhead time exists even in the effective object drawing operations. This inference is supported by the result of M/O/N analysis that there were considerably large NON times and much time was consumed by one operation. It becomes clear from the wasted-operation analysis that as long as an existing drawing editor is used, the overhead to make up a drawing strategy is considerably large. However, this overhead time is not necessarily equal to the NON time mentioned above. A certain period of overhead for considering a drawing strategy can also be contained in the MENU and OBJ time. In fact, the verbal contents issued during long MENU or OBJ operations indicate that a certain period of overhead time is present in these operations. In contrast, the NON time also contains the time required for physically moving the mouse pointer from one point to another.

Therefore, Exp. B shown in Section 3.1 was carried out to measure the drawing operation time by well-skilled subjects. The operation time of Exp. B (T_B) can be treated as an ideal time when a user can completely operate the editor without any psychological overhead such as planning, etc. In other words, the psychological overhead in Exp. A can be extracted by subtracting the time of Exp. B.

$$(\text{The psychological overhead time}) = T_A - T_B$$

Note that “the drawing time excluding wasted operations in Exp. A ($T_A - T_W$)” is not equal to “the ideal drawing time in Exp. B (T_B)” although the number of operation steps is same in these two cases. The difference of those two times can be considered as the overhead time by immature drawing. The strategy-planning time is included in this overhead.

$$(\text{Overhead time by immature drawing}) = (T_A - T_W) - T_B$$

In other words, the psychological overhead time is the sum of wasted time and immature drawing time.

Table 3.8 Average time of drawing an arbitrary line

	A	B	C	D	E	F	G	H	I	J	K	L	M	Average
Mouse	1.25	1.29	1.08	1.30	0.99	2.78	1.95	1.16	1.07	1.24	1.13	1.57	1.24	1.39
Pen	1.35	1.67	0.85	0.94	1.48	1.27	1.98	1.18	0.94	2.50	0.96	0.80	0.80	1.28

3.5.3 Physical Minimum Time and Indispensable Overhead Time

As described in the previous subsection, the time of Exp. B can be considered as the ideal operation time which removes psychological overhead. This time was further separated into two layers by using the result of Exp. C. The result of Exp. C is shown in Table 3.8, which shows average time for drawing an arbitrary line (T_C) with mouse and pen of each subject. It is clear that the line drawing accompanied with some alignment always takes more time than the line drawing with no alignment. Since in actual picture drawing the rotation and movement are accompanied by the alignment, more time is taken up by the alignment.

To estimate such an indispensable overhead for drawing, the “physical minimum time” to draw the picture (T_M) was defined by the following equation.

$$(\text{Physical minimum time to draw the picture, } T_M) = T_C \times 22$$

where 22 is the total number of lines contained in the examined six pictures. This time means the physical time to draw the number of lines contained in the pictures. Of course this time is not completely equal to the time which eliminates the alignment overhead, but it can be used as one index of ultimate drawing time.

The remaining time which removes the physical minimum time from the time of Exp. B can be considered as an indispensable physical overhead for drawing the picture, which includes alignment, etc.

$$(\text{Indispensable overhead time}) = T_B - T_M$$

3.5.4 Four-Layer Classification of Drawing Time

Arranging all of those operation times mentioned in the previous subsections, the time for drawing operations is broken down into the following four layers as shown in Figure 3.6.

T_1	Time of wasted operations (trial-and-errors and mis-operations)	$= T_W$
T_2	Overhead time by immature drawing (including planning)	$= (T_A - T_W) - T_B$
T_3	Indispensable overhead time for alignment, etc.	$= T_B - T_M$
T_4	Physical minimum time to draw the picture	$= T_M$

Figure 3.7 shows the graphs in which the time for drawing operation of each subject was classified into four layers.

First, it can be observed that considerably large portion is occupied by the psychological overhead time ($T_1 + T_2$). With Canvas, the time was about 33% of total drawing time, and it reached almost 60% with SmartSketch. The psychological time differed much among subjects. This can be considered to indicate the drawing skill of each subject with the editor. The time T_3 also differed among subjects, depending on the adopted drawing strategy. With SmartSketch, the wasted time (T_1) became long because of the editor's unique problems described in Section 3.3.2 and 3.3.3.

The same analysis for the “rectangle on a slope” picture is shown in Figure 3.8. Much more time (60–80%) was consumed by the psychological overhead. It is observed that Canvas became worse performance with such a complex picture because of increasing T_2 time. This means that strategy-planning takes long time for drawing a complex picture, although simple pictures may be drawn easily by using primitive objects.

It is understood from the above results that the main points of problem with existing drawing editors lie in the psychological overhead time ($T_1 + T_2$).

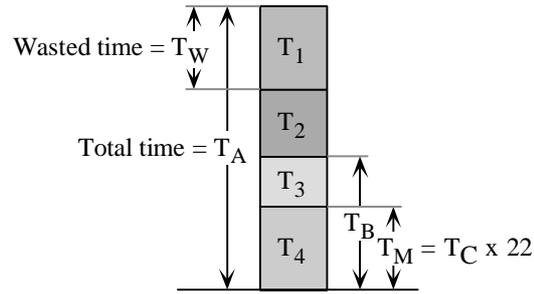


Figure 3.6 Four layers in the actual drawing time

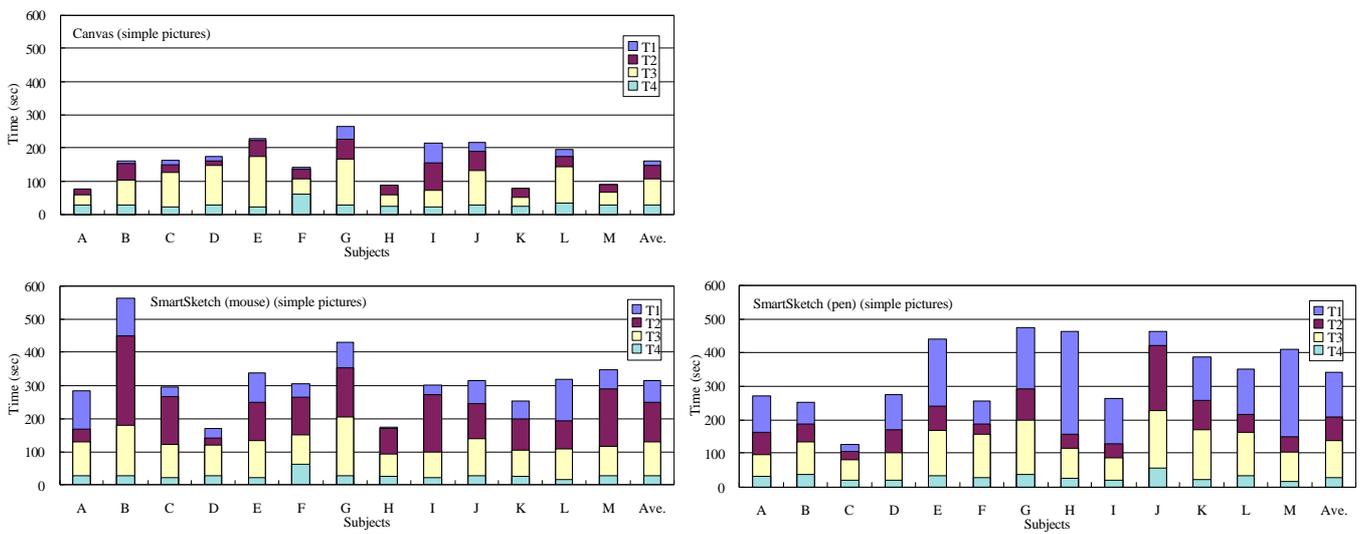


Figure 3.7 Four-layer analysis of drawing the simple pictures

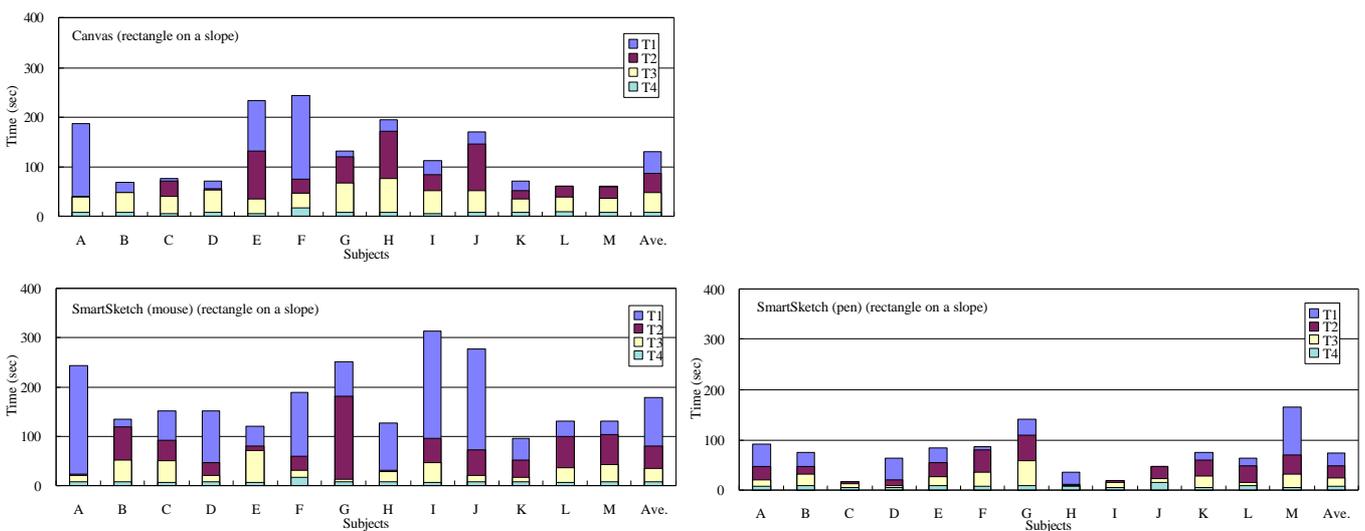


Figure 3.8 Four-layer analysis of drawing the “rectangle on a slope”

3.6 The Case of Drawing Complex Pictures

It was known from the analyses described so far that much time is required to design a drawing strategy for converting a concrete picture to the abstract picture as the function of drawing editor, or that trial-and-error is repeated without noticing strategic errors. Those results are lead from the case of simple six pictures and complex picture “rectangle placed on a slope.” Here, the word “complex” means that the picture has many geometric constraints. It can be easily predicted that such a tendency as above is also found when drawing much more complex pictures.

In this section, in order to prove this prediction, rough analysis of more complex pictures is introduced as supplemental information. It is already known that although SmartSketch adopts an object inputting procedure different from that of Canvas, at the editing stage SmartSketch uses functions similar to that of Canvas, and takes much more time than Canvas. Accordingly, this experiment was executed using only Canvas.

For the analysis, only the comparison of total and wasted operation time used in Chapter 2 was performed, because:

- It was very hard to draw complex pictures with no error even though after exercising well. Therefore Exp. B and four-layer analysis couldn't be performed.

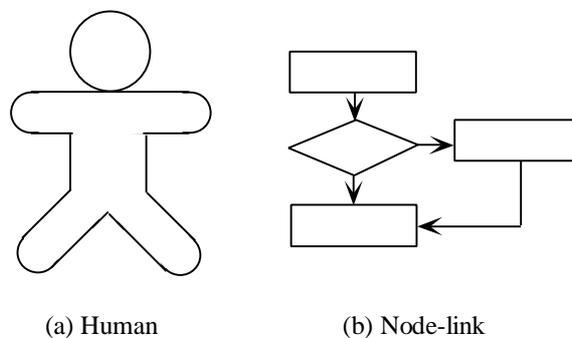


Figure 3.9 More complex pictures

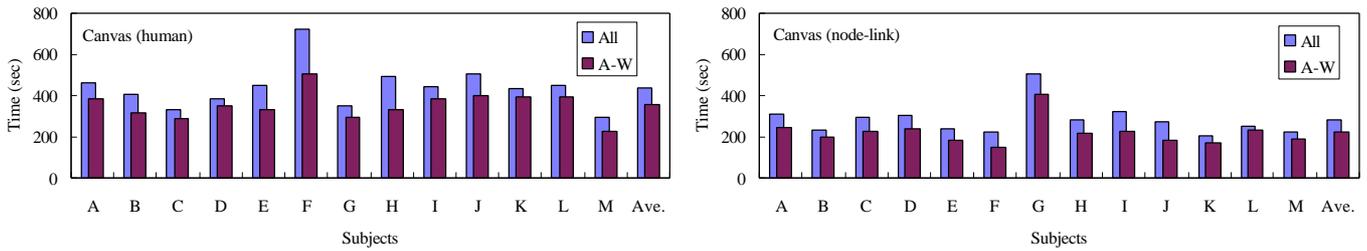


Figure 3.10 Time of drawing complex pictures

- Correlation between the number of the operation steps and operation time was observed.
- From the experience of the analyses mentioned above, it could be enough to know the global tendency of total and wasted time for complex pictures.

The experiment (Exp. A) was carried out on the following two complex pictures:

1. A human figure model as shown in Figure 3.9(a), which is much more complex picture with more geometrical constraints than those of “rectangle on a slope.”
2. A node-linked picture or flow chart (a picture consisting of rectangles, rhombuses, and circles connected with arrows) shown in Figure 3.9(b), which is frequently used in technical papers or presentations. This type of picture is a favorite for the object-oriented editors to draw.

First, the human figure model experiment is described. The human figure model took an unexpectedly long time to draw. The left graph of Figure 3.10 shows the relationship between the total time for drawing operation and the time which excludes wasted operations. There was a variation in drawing time. The drawing strategy also had wide variations. Most of the subjects spoke “Hard!” when drawing the legs of the human figure model.

Figure 3.11 shows a typical incorrect strategy for drawing the leg part:

1. Draw two parallel lines (Figure 3.11(a))
2. Draw a circle to contact both two lines (Figure 3.11(b))
3. Move and resize the circle to contact both two lines

The operation 3 is an incorrect strategy, while a correct strategy is:

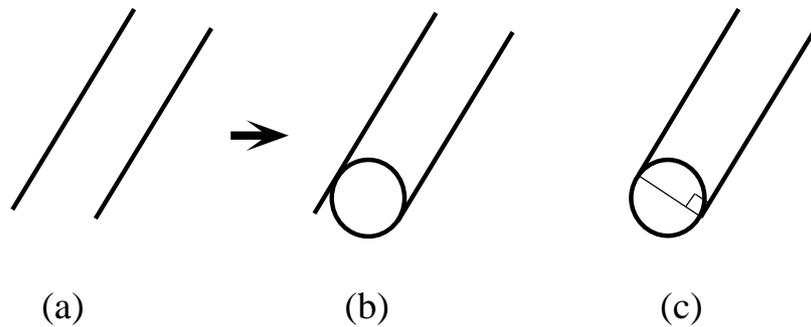


Figure 3.11 Strategies for drawing the leg part of the “human”

4. Resize one of the lines such as shown in Figure 3.11(c)

In the experiment, many subjects could not notice their error and tried the operation 3 for a long time. Some subjects, trying as hard as they might, but without being able to make the drawing well, gave up the drawing in an incomplete state in the middle of the operation speaking “I regard the drawing as completed.”

Next, an example of the node-linked picture is explained. The right graph of Figure 3.10 shows the time taken for drawing the picture and the time which excludes the wasted operations. Subjects did not take so much time to draw this example. However, since the number of elements constituting the node-linked picture is as many as three rectangles, one rhombus, and four arrows, certain variation in drawing procedure was observed among the subjects.

It was observed from those analyses that wasted operations caused from incorrect strategies increased. Moreover, rather than this tendency, the more serious problem is that in a picture with many geometrical constraints, the understanding of the geometrical constraints (given in the picture desired to be drawn) itself was difficult for subjects. Here, the following new problem with existing drawing editors arises. With existing drawing editor, the picture cannot be drawn unless the subject has the correct knowledge of geometrical constraints that resides in the picture desired to be drawn.

In the “node-linked picture” the percentage of the wasted operation time is nearly the same as that required for drawing the basic pictures. This is because the node-linked picture consists of several simple pictures located with simple geometrical constraints such as “two objects being in contact at one point with each other.” The object-oriented drawing editor is considerably well suited for drawing such picture as in the node-linked picture.

3.7 Discussion of Revealed Problems

When drawing a picture by using a pencil and a sheet of paper, the whole work is a directly visible treatment of the concrete picture. In contrast, the drawing operation using a drawing editor is a work in which the concrete picture treatment and the consideration for a combination of editor functions such as copy, move, and rotate are alternately repeated. In the drawing operation using a drawing editor, the problems are considered to be a mixture of two types of pictures, one being a visible concrete picture and the other being an invisible symbol sequence (abstract picture). In particular, the treatment of the abstract picture enforces a psychological load on the subject. The next chapter will propose an operation model for conventional drawing editors that can cope with the above mentioned problems.

While above results are the cases of using Canvas, they are regarded as general to object-oriented drawing editors because:

1. Canvas is widely spread object-oriented drawing editor.
2. The experiment was performed using common object-oriented drawing functions.
3. The figures used in the experiments were primitive and basic.

4 Operation Model of Conventional Drawing Editors

This chapter first proposes an operation model considering the concrete picture and abstract picture. Next, using this model, conventional studies on drawing editors that have been made up to now are analyzed and explained. Further, the relationship between the proposed model and Norman's seven-stage model which is famous in the area of cognitive engineering is examined.

4.1 Concrete Picture and Abstract Picture

As already mentioned several times, in pictures to be dealt with in drawing operations on computers, there exist “concrete picture” and “abstract picture.” The former is the picture that one can see directly with his/her eyes, and the latter is the picture as a symbol sequence, which is a combination of editor's various functions. In other words, abstract pictures have only geometrical constraints such as “rotation,” “copy,” and “move,” while concrete pictures also have actual shape with the constraints. In operations of conventional drawing editors, there are two categories of work; that is, the work that belongs to concrete-picture layer and the one that belongs to abstract-picture layer. The process of drawing operation can be analyzed as follows.

At first, a user images certain desirable picture, then converts it to editor's commands, and then draws the picture by executing the drawing operations in accordance with rules of the editor. Thus, comparing each appearing picture to the original image, the user repeats the same cycle until he/she gets the desired picture. In this operation, the “picture image to be drawn” is considered as a concrete picture, because it is still the “picture

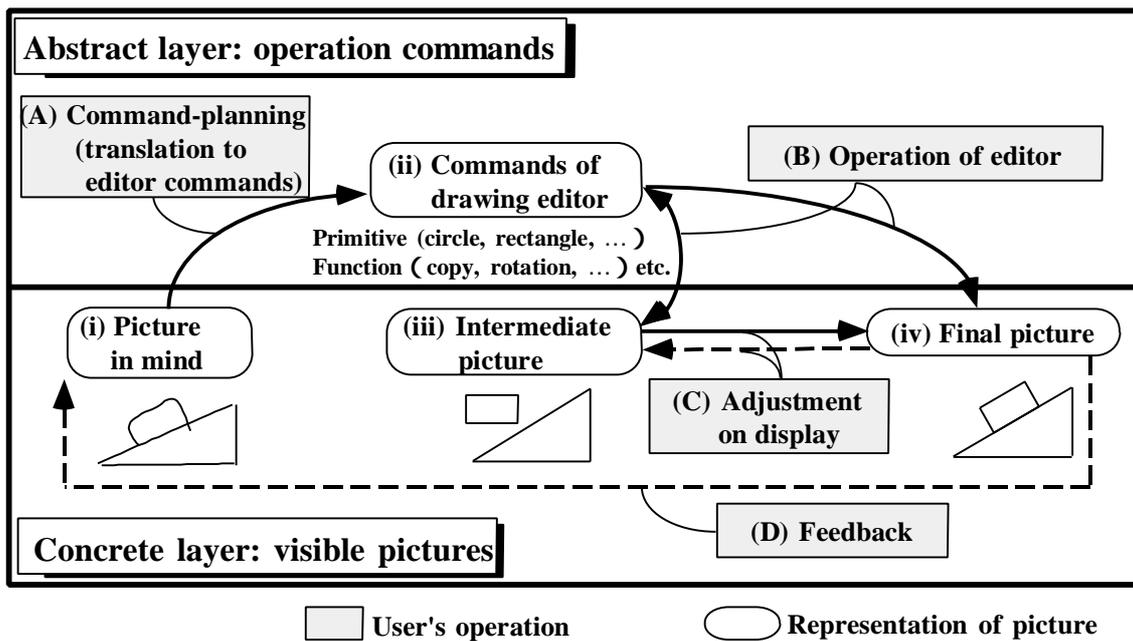


Figure 4.1 Two-layer operation model for conventional drawing editors

itself” although it exists only in the user's mind. On the other hand, “buttons or menus of the editor” are considered to be abstract pictures, since they are not “picture itself” although they are existent entities and visible.

Thus, operations of conventional drawing editors can be modeled on the basis of such concepts as “concrete-picture layer and abstract-picture layer” and “work cycle.”

4.2 Two-Layer Operation Model

A user first images a concrete picture to be drawn, and convert it to a sequence of editor commands. This process is named as “command-planning” in this thesis. Then, the user draws intermediate picture by executing the drawing operations in accordance with rules of the drawing editor. Comparing the displayed picture to the original desired image, fine tuning is repeated until final picture is drawn on the display.

This cyclic drawing task can be modeled as shown in Figure 4.1, which consists of two layers: concrete-picture layer and abstract-picture layer. In the model, command-planning (A), command sequence (ii), and editor operation (B) reside in the abstract-

picture layer, whereas the target picture (i), intermediate picture (iii), final picture (iv), fine tuning (C), and feedback (D) exist in the concrete-picture layer

Based on this model, the factor of psychological overhead time can be described as follows. First of all, in works carried out in the abstract-picture layer, psychological time is required because they are different from those drawing works done with paper and pencil. In particular, “command-planning,” which is the first step necessary for the user to enter from the concrete-picture layer into the abstract-picture layer, is the factor that requires a lot of psychological overhead. In the proposed model, the “immature drawing time (T_2)” discussed in Chapter 3 can be explained as the “planning overhead,” and the “wasted operation time (T_1)” can be explained by “planning errors.” In addition, the existence of various drawing strategies corresponds to that the arrow mark shown as (A) diverges and multiple command sequences (ii) exist.

4.3 Conventional Work for Improving the Drawing Efficiency

Traditional studies of drawing editors are explained using the model mentioned above. Most conventional researches for computer-assisted drawing aimed only one individual operation in the model. The total cycle of drawing process has scarcely been investigated.

There is an approach of increasing the functions of drawing editors, or drawing more constraints and macros [168, 17, 127, 143, 142]. Intelligent CAD systems [169, 170, 168, 70, 175, 172, 88, 122, 50, 161, 109, 72, 132, 145, 57, 177, 137, 148] are also included in this approach. It can be said that while this approach aims to eliminate the fine tuning operations (C) and to enhance the drawing accuracy, in actually it results in the lengthening of the command-planning time (A) by providing too many possible drawing strategies (ii). Further, this approach then has the problem that the inadequacy of constraints makes it difficult to obtain the intended diagram and much time may be taken for trial-and-error.

There is an auxiliary approach utilizing operation history, so-called “Programming by Example (PBE)” [115, 105, 94]. Although this approach attempts to reduce the frequency of the recurrence of the simple work in editor operations (B) or fine tuning operations (C), it cannot reduce the time of command-planning (A) in the first drawing operation.

There is another method for recognizing and beautifying a document or diagram drawn on paper by reading it from a scanner [20, 10, 101, 31, 36, 113, 124, 22, 123, 182, 95, 146, 77, 78, 71, 147]. This method is superior in that no abstract picture is handled at the first drawing step before the scanner is used to read the diagram. However, when an attempt is made to edit a diagram, the use of planning (A) becomes necessary. Furthermore, since the processes of feedback operation (D) and correction of a picture cannot be executed in the real time mode, this method is inferior in editing efficiency to the electronic drawing editor. The superiority of the first drawing step and the inferiority of the editing work are characteristics of this method.

Sketch-based drawing method has been improved according to the development of pen device, The computer recognizes and beautifies the sketched picture. 2D sketching [51, 52, 167, 111, 193, 194, 27, 90, 100] and 3D sketching [28, 192, 158] are available. The sketching operation is suitable for drawing pictures since the first drawing is carried out by hand-writing and no abstract picture is needed. This can be considered as a big advantage to the object-oriented type drawing editors. However, for editing the inputted picture, abstract picture must be handled even if editing commands by gesture can be used. Note that for drawing a picture which meets various geometrical constraints, this method of recognizing and beautifying a diagram still needs to use the same operations (A) and (B) as those in an ordinary drawing editor. Accordingly, it can be said that the superiority of this method to the object-oriented type drawing editor lies in the input efficiency. However, it cannot be utilized effectively unless some geometrical constraints are satisfied, as described in Chapter 3. In other words, sketch-beautification drawing need to use the function identical to what is used in object-oriented drawing editor, not only in editing stage but also in input stage.

There is a study that is conceptually close to the proposed operation model shown in Figure 4.1 which tries to use the concrete expression of a picture by classifying the expression into four degrees of abstractness [85]. This study also considers that the concrete expression is a very important subject. The drawing system proposed in this study copes with a concrete diagram by applying a stationary metaphoric expression as the input method. However, for editing it still requires the employment of operations similar to object-oriented type drawing editors. For the entire flow of operation it is necessary to use operations (A) and (B).

Moreover, in comparing the method for drawing a picture on electronic panel and the method for drawing a picture on paper, the panel method is superior to the paper method in interactively correcting the diagram. The use of an algorithm superior in recognition and beautifying can omit only the fine tuning process (C), and both operations (A) and (B) are still required. It can be said that the entire flow of operation is almost the same in both methods.

Additionally, an interesting study has recently been reported that the number of drawing operation (B) can be reduced by educating the user an efficient drawing procedure [12]. The efficient procedure is none other than the command-planning (A). In other words, it can be said that this study proves the existence and importance of the “command-planning.” Most conventional operation analysis compares only the number of steps in an operation using task analysis and does not consider the load of planning in the abstract-picture layer. This study simply considers the reduction of the number of steps. The planning load for pre-educated fixed work is exactly decreased, however, the burden of a new work cannot be reduced by the education.

4.4 Relationship to Norman's Seven-Stage Model

The two-layer operation model proposed in the preceding section can explain problems of conventional drawing editors and position of conventional studies on drawing editors. Accordingly, this model can be considered as useful enough to show

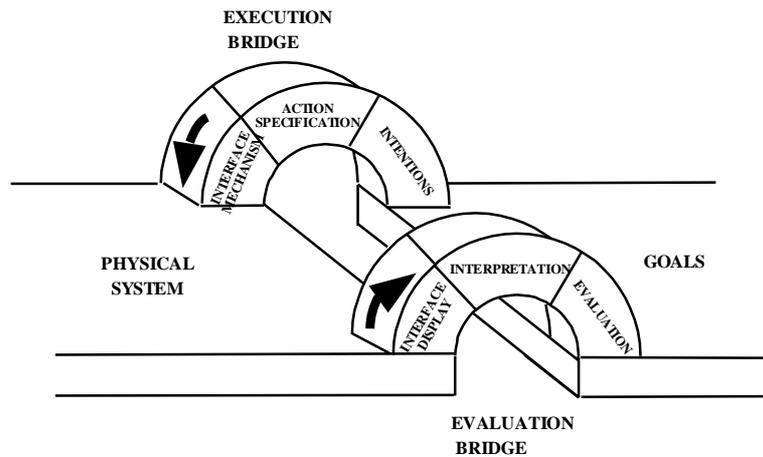


Figure 4.2 Norman's seven-stage model

operation methods of drawing editors. This section shows that the proposed model is not inconsistent with the results already established by studies in cognitive engineering area, by explaining the model with Norman's seven-stage model.

In the first place, let's look at Norman's seven-stage model [131]. Norman is called the father of cognitive engineering, and his model has been accepted as the base of cognitive engineering since he proposed it in 1986. He showed such a model as cited here in Figure 4.2. There, he defined that: we have to jump over the big “gulf” between physical world and psychological world (goal) at the point when a human starts to carry out some work to achieve his/her goal (execution bridge), and also at the point; when the human starts to evaluate whether or not the result obtained by the work has achieved the goal (evaluation bridge).

Thus, according to him, the core source of problems of the interface exists in this gulf, and major task of system design is how to make these bridges between the two worlds easy to be crossed. To make the study of such bridging more effective, he classified the cognitive actions of a human being into seven steps and used such seven steps cycle as the theory of actions. The Norman's seven stages are:

1. Establishing the goal
2. Forming the intention
3. Specifying the action sequence

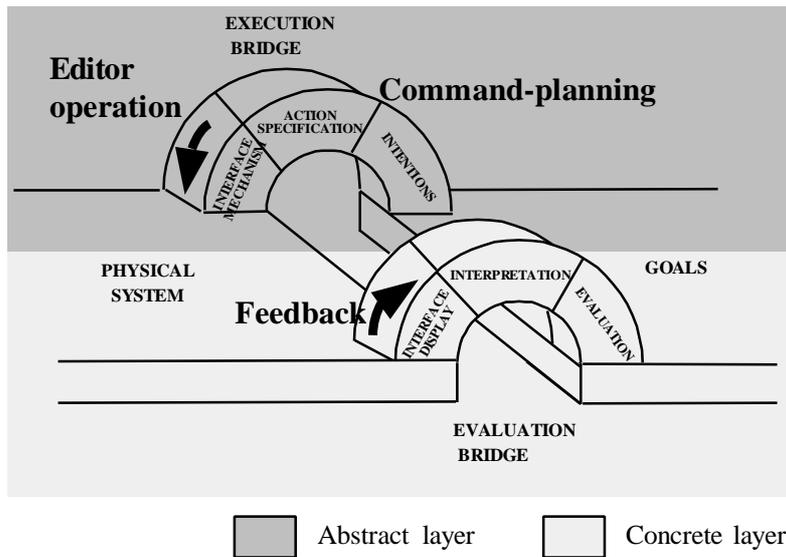


Figure 4.3 Relation of Norman's model and two-layer model

4. Executing the action
5. Perceiving the system state
6. Interpreting the state
7. Evaluating the system state with respect to the goals and intentions

Norman's "establishing the goal (1), forming the intention (2), and specifying the action sequence (3)" correspond to the "command-planning (A)" in the proposed two-layer model, his "executing the action (4)" corresponds to "editor operation (B)," and his "perceiving the system state (5), interpretation the state (6), and evaluating the system state with respect to the goals and intentions (7)" correspond to the "feedback (D)" of the proposed model. Though the two-layer model unifies Norman's three stages, it does not deny his theory. It merely replaces the stream of such three steps with one word of "command-planning" to clarify the problem in drawing operations.

Next, let's study the two-layer model (Figure 4.1). Norman defined the classification of two worlds by "system's physical world" and "user's psychological world." On the contrary, the two-layer classification in this thesis is made based on the different concept of "the structure (expression) of picture." In this thesis, a picture visible to the user is referred as "concrete picture," and a picture as a sequence of editor's functions is referred as "abstract picture." In short, the two layers defined in this thesis are the ones that are

created by horizontally dividing Norman's model at the both ends of the gulf, as shown in Figure 4.3.³

For example, a picture which the system outputs on the display is a concrete picture, and the data of picture that the system internally maintains is an abstract picture. Also, specific drawing that the user images in his mind in such way as “the picture made by combining a diagonally placed rectangle and a triangle,” for instance, is a concrete picture. But when he images it (the same picture) as a sequence of editor commands, such as “[rectangle] tool button => draw the rectangle by mouse dragging => [multi-line] tool button => draw a triangle => selection of a square => click [rotation] item in the menu = ...,” such image in his mind is an abstract picture.

From the preceding examination, it is evident that the two-layer model is not in any way inconsistent with Norman's model, although the approach and the defining method are different. The model can be considered as a variation of Norman's seven-stage model specialized to explaining the computer-assisted drawing task.

The unique concept of the two-layer model is that even in user's psychological world there exist both concrete picture and abstract picture, as well as in system's world. By grasping processes through such “concrete and abstract” approach, it helps to understand Norman's model more easily. In the case of drawing pictures, difficulties of the two bridges are different. Using the two-layer model, the problem in the drawing operation can be explained that the execution bridge, especially the command-planning, exists in the abstract-picture layer and is difficult to be crossed because of its psychological load to handle invisible abstract pictures.

³ Note that the left and right are reversed between the Norman's model and the two-layer model.

4.5 Ideal Improvement of the Operation Model

As Norman insists, the important thing is to make the two bridges — execution bridge and evaluation bridge — over the gulf smooth to be crossed. Saying this with the words of two-layer model, the important point is to make the command-planning (A) and feedback (D) smooth. Also, from the viewpoint of “concrete picture” and “abstract picture,” which is unique to the model, the important thing is to minimize tasks in “abstract-picture layer,” which are difficult to be handled. Therefore, the most important point to improve the drawing efficiency is to reduce the load of the command-planning (A).

5 Toward an Ideal Drawing Editor

In this chapter, methods to improve the operation model described in the previous chapter are pursued. The following three actual approaches to an ideal drawing editor are proposed and evaluated:

1. Sketch-annotation method

To solve the unique problems in the sketch-beautification type drawing editor mentioned in Chapter 3, an “annotation” is added to the inputted stroke to explicitly specify the desired recognition. The method aims to achieve 100% “line success ratio” and “direct drawing ratio” defined in Chapter 3.

2. Dot-masked revision sheet method

To confirm that paper utilization has great benefits as a user interface, a drawing system using real papers is proposed. Using the most “concrete” way for drawing is the main advantage of this method. In the system, unique “dot-masked revision sheet” is introduced for supporting easy and accurate editing.

3. Candidate-selection method

Aiming to reduce the user's operations in the abstract-picture layer, this method introduces the human-perceptual recognition of the inputted sketches and the indication of multiple recognition results. The user has only to sketch by free-hand and select a desired picture from the displayed recognition candidates.

Each proposal is evaluated from the following two viewpoints:

- Does it achieve its original aim?
- Does it approach to an ideal drawing editor?

5.1 Requirements for an Ideal Drawing Editor

From the aforementioned experiments and operation model, the conditions which are required to achieve essential improvement to a drawing work can be summarized as follows:

Reduce the operations in the abstract-picture layer.

Increase and improve the operations in the concrete-picture layer.

Possible actual directions for it are shown as follows:

- Use concrete-picture layer at least for the first input operation
- Improve the feedback interface
- Reduce the load of command-planning
- Reduce the editor operation steps in the abstract-picture layer

5.2 Approach 1: Sketch-Annotation Method

Sketch-beautification type drawing editors should be ideal in that they require users to use only concrete pictures for input. However, the analysis in Chapter 3 clarified that they cannot be used well without a recognition that satisfies some geometrical constraints. Without such recognition, editing becomes indispensable to satisfy the desired geometrical constraints, and it enforces the use of abstract pictures. By solving the problem, this section seeks a specific way to utilize the advantage of the sketch-based input.

5.2.1 The Method that Annotates Inputted Sketches

Conventional sketch-beautification type drawing editors have the following problems:

1. When a user draws a straight line with mouse, the system does not recognize the drawing as a straight line, even if the user thinks “I must have drawn the straight line.” “Electronic pen is better, but I want to draw well even with mouse.” “Even if my drawing is a little poor, I want the system to recognize it.”

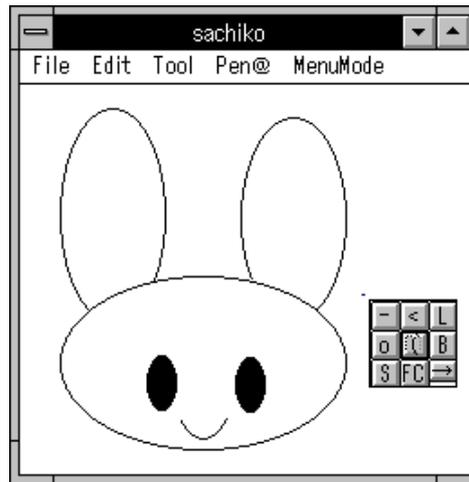


Figure 5.1 Sample drawing of the sketch-annotation method

2. Even with electronic pen, a user cannot skillfully draw an arc-shape at all. It is difficult to recognize the drawing as an arc by system, too. There exist pictures that the user cannot skillfully draw by hand. Also, there are some kind of pictures that computers cannot easily recognize.
3. The result of recognition seems to be a right angle, but the user cannot determine whether it is exactly right angle or not. The result of recognition seems to be a circle, but it may be an ellipse. So, it is better to clearly specify geometrical constraints with drawings.
4. When the user tries to move or rotate a specific object, it is somewhat difficult to select the object using pen. The user wants to make selection by pen easier.

Therefore, to solve these problems the “sketch-annotation method” was proposed [80, 81]. In this method, the user draws the shape by free-hand at first. Operations in this stage are the same as what are done in conventional sketch-beautification type editors. But after this input, in this method, the user selects the desired shape from a menu (i.e. adds an *annotation* to the input). By this way, the user can intentionally affect the system’s recognition process. In other words, the user can teach his/her intention to the computer.

In addition, when the system is switched to editing mode, “selection handle” is displayed for each object to make the selection by pen easier.

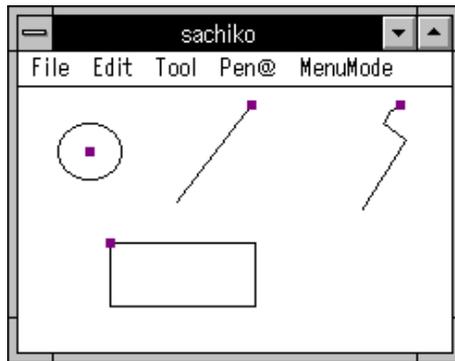


Figure 5.2 Selection handles in the editing mode

5.2.2 Prototype Implementation

Prototype system of the “sketch-annotation method” described above is developed on ThinkPad 360P, which is an IBM's pen-computer, using Microsoft Visual Basic. Figure 5.1 shows an example drawing. The box displayed at the right bottom is a “shape-menu” for sketch annotation..

Operation with the prototype is as follows. When a user draws a shape with pen, its locus is displayed in blue. The time span from the point when the pen touches to the screen to the point when it leaves is treated as one stroke. Then, the user selects the shape from the shape-menu. In the shape-menu, there are [straight line], [rectangle], [circle], [arc] and [exactly the same as the sketch]. Final determination of the shape is made when the user starts the next stroke input after the shape-menu selection. When the shape is determined, the color is turned to black. If the user starts the next stroke input without shape-menu selection, the previous stroke input is invalidated. It stands that the user is re-drawing. Also, as the shape has not been finally determined still at the stage of shape-menu selection, the user can change the selection any number of times if necessary. After the shape-menu selection, the user can invalidate the input by the special [undo] menu.

Copy, move, rotate, and delete are executed under editing mode which is basically same as that in conventional sketch-beautification editors. But in the editing mode, small square-shaped “selection handle” is displayed for each object as shown in Figure 5.2 to

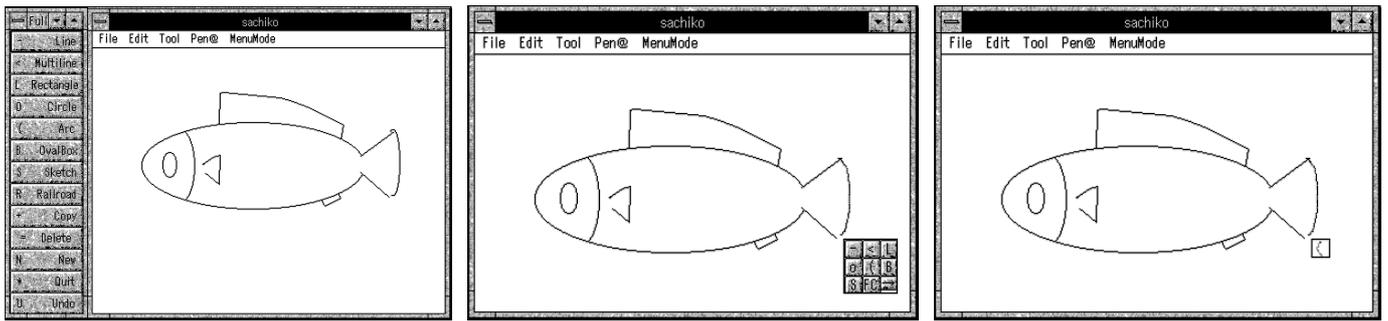


Figure 5.3 Three menu types for specifying annotation

support easy and accurate object selection. The input mode and editing mode are switched through a menu.

As methods for adding annotations to drawings, following three types were developed:

1. Menu bar (left of Figure 5.3)
2. Icon menu (middle of Figure 5.3), in which a set of small icons are displayed near the end of the locus per each stroke.
3. Gesture menu (right of Figure 5.3), in which a gesture for annotation is inputted into a small box opened near the end of the locus per each stroke.

Here presents the comparative examination for these three types. For type 1 and 2, it is considered that type 2 is more efficient because the amount of pen movement can be small. However, type 2 has a problem that it may occupy too much space of the screen when the number of menu items increases. Type 3 has an advantage that it requires only small size menu area. However, it may cause one problem; namely, erroneous recognition of the gesture itself. Although the problem can be reduced if the system uses a set of gestures that would cause minimum recognition error and are not confused with each other, type 3 has another problem that users have to learn and memorize gestures in advance. This should become a heavy psychological overload. Consequently, type 2 method is the most efficient, and enforces the least burden on users.

Figure 5.4 shows the processing flow of the prototype. The prototype has two operation modes — input mode and editing mode. In input mode, input is used for drawing the locus (15), selecting a shape (14), or changing the operation mode (10). In

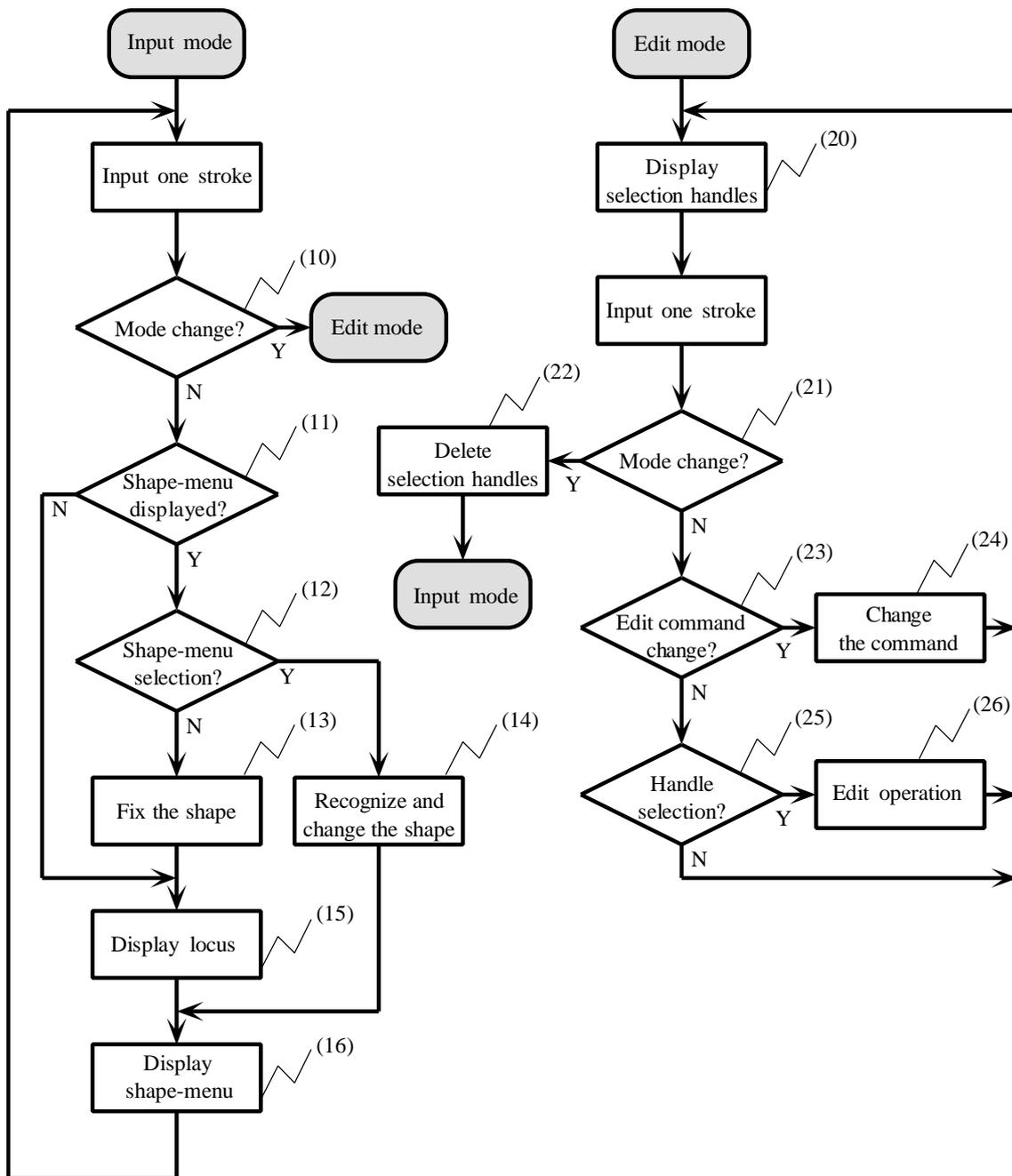


Figure 5.4 Processing flow chart of the prototype sketch-annotation method

editing mode, selection handles are first displayed (20). Then editing is performed (26) based on the selected editing command (copy, move, rotate, and delete) (24).

For converting the inputted stroke to a picture in a recognition module (14), there are no concepts particularly new. Therefore, the algorithms are described at a minimum. As for a straight line, reference is made only for the starting and ending point of the stroke.

For a rectangle, square is circumscribed about the stroke-input that has been drawn. To recognize an ellipse, an algorithm using Newton method is developed (Figure 5.5). The similar method is used for recognizing an arc.

Response time required to calculate a recognition result is less than 1 second for the pictures like Figure 5.1 and Figure 5.3. using an IBM PC compatible machine with i486 75MHz. So, interactive operation is possible.

Algorithm for ellipse recognition:

Among the inputted stroke, let the point that has minimum X-value be (x_0, y_0) , maximum X-value be (x_1, y_1) , minimum Y-value be (x_2, y_2) , and maximum Y-value be (x_3, y_3) , as shown in Figure 5.6.

Generally ellipse is given by the following equation.

$$(x - a)^2 / p^2 + (y - b)^2 / q^2 = 1$$

This equation can be transformed using $c = p^2$ and $d = (p / q)^2$.

$$(a - x_i)^2 + d (b - y_i)^2 - c = 0$$

Four simultaneous equations of four unknowns (a, b, c, d) can be created by substituting (x_i, y_i) to the above obtained coordinate values.

$$f_0(a, b, c, d) = (a - x_0)^2 + d (b - y_0)^2 - c = 0$$

$$f_1(a, b, c, d) = (a - x_1)^2 + d (b - y_1)^2 - c = 0$$

$$f_2(a, b, c, d) = (a - x_2)^2 + d (b - y_2)^2 - c = 0$$

$$f_3(a, b, c, d) = (a - x_3)^2 + d (b - y_3)^2 - c = 0$$

These equations are solved by Newton method. In general, Newton method calculates the following equation in an iterative manner.

$$a_{k+1} = a_k - f(a_k) / f'(a_k)$$

For the case of multiple unknowns, the Δa , Δb , Δc and Δd are obtained by solving the following equations.

$$\partial f_0 / \partial a \cdot \Delta a + \partial f_0 / \partial b \cdot \Delta b + \partial f_0 / \partial c \cdot \Delta c + \partial f_0 / \partial d \cdot \Delta d = -f_0$$

$$\partial f_1 / \partial a \cdot \Delta a + \partial f_1 / \partial b \cdot \Delta b + \partial f_1 / \partial c \cdot \Delta c + \partial f_1 / \partial d \cdot \Delta d = -f_1$$

$$\partial f_2 / \partial a \cdot \Delta a + \partial f_2 / \partial b \cdot \Delta b + \partial f_2 / \partial c \cdot \Delta c + \partial f_2 / \partial d \cdot \Delta d = -f_2$$

$$\partial f_3 / \partial a \cdot \Delta a + \partial f_3 / \partial b \cdot \Delta b + \partial f_3 / \partial c \cdot \Delta c + \partial f_3 / \partial d \cdot \Delta d = -f_3$$

The equations are solved by using Gauss-Jordan method, and the iteration is repeated until the Δa , Δb , Δc and Δd become small enough.

Figure 5.5 Algorithm for ellipse recognition

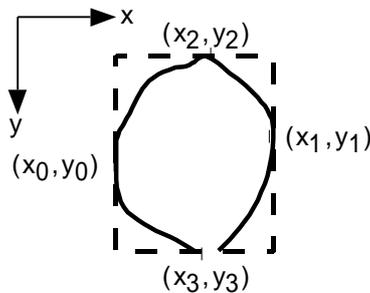


Figure 5.6 Control points used for ellipse recognition

5.2.3 Comparison to Conventional Sketch-Beautification Editors

The evaluation experiment for comparison was carried out using the prototype described in preceding subsection. Shape selection is executed using the type 2 method, i.e. a small icon menu opens near the end of the locus per each stroke. Same subjects and pictures in Chapter 3 were used for the evaluation. In the proposed method, both of the line success ratio and selection success ratio were 100%. Therefore, the input-device specific problems observed in SmartSketch (Section 3.3.2) have been resolved.

The operation time of proposed method is inferred to be shorter than conventional object-oriented type drawing editors because it uses concrete pictures during input operation. However, it was found that the actual operation time was nearly the same as that of Canvas. For example, Table 5.1 shows the average time for drawing four lines with each editor. The time with the prototype is not so much better than Canvas. This phenomenon can be examined as follows. In the case of the inputting one straight line, there is no essential difference of duration between the proposed method and conventional object-oriented method, because required operation steps are two in both methods, that is, [shape selection] and [line drawing]. In other words, the difference of operation between the two methods only exists in that “the sequential order of shape selection and picture drawing,” and that has resulted in the same duration.

As for editing work, operations are executed under the same editing mode as that of object-oriented editors. Therefore, it is hard for the proposed method to exceed object-

Table 5.1 Drawing speed of four lines

	Time (sec)
Canvas	23.5
SmartSketch (mouse)	47.0
SmartSketch (pen)	28.8
Sketch-annotation (mouse)	22.6
Sketch-annotation (pen)	20.2

oriented type editors in the editing performance.

As mentioned above, the proposed method was not able to exceed object-oriented type editors in performance. However, the method has succeeded in reducing the operation time of sketch-beautification editors to the level equal to that of object-oriented editors. This is a unique contribution of the sketch-annotation method.

5.2.4 Evaluation as an Ideal Approach

It has been proven that the proposed method has achieved such improvement that makes good use of the merits of sketch-beautification type drawing editors, namely the point of “using concrete pictures at input stage.” This should be a great contribution. However, could the sketch-annotation method be an approach to an “ideal drawing editor?” This method uses concrete pictures at the input stage, so, it had to surpass object-oriented type drawing editors in what relates to this part of processing. Actually, however, the improvement of input stage is very small and could not achieve so much effect. User's psychological overload still remains in heavy, especially in the editing stage.

5.3 Approach 2: Dot-Masked Revision Sheet Method

If the user can draw pictures on a paper by pencil, abstract operation in drawing editor can be avoided. Although the total operation time may increase because scanning of the paper is necessary, it is inferred that the user's psychological overhead time can be reduced. Therefore, in this section the paper utilization is focused on as an approach to an ideal drawing editor.

User interface based on physical paper had been studied since 1960s. and has continuously been studied even in recent years. Card et al. of XEROX released a report concerning an inquiry system of the database with paper [71, 147] and Sellen studied paper interface basically [45, 134, 159]. They emphasized the advantages of paper as “historical/social/educational background,” and “simplicity possessed by paper.” In

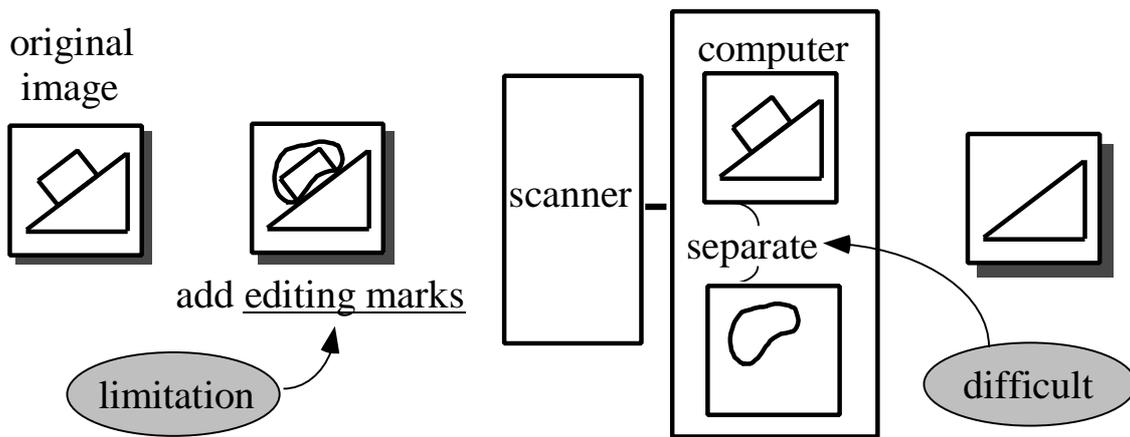


Figure 5.7 Conventional method of hand-written diagram recognition system

addition to those emphases, this section pays attention to the “less psychological overload by using paper.”

5.3.1 Problems in Scanner-Input Drawing

The important problems considered in paper input are “editing method” and “image degradation caused by multiple scanning.” First, those problems are explained.

1. Editing method

For editing on a paper, “editing marks” similar to “proof-reading symbols for documents” are required to be inserted. It is possible to use paper only in first stage and to edit on a display. However, for the consistent operation, editing should also be done on physical paper. Therefore, it is necessary to edit pictures by using “hand-written editing marks” on a paper.

2. Image degradation caused by multiple scanning

The revision marks are required to edit pictures. At that time higher version should be constructed by editing on the copied paper. However the problematic point in such an occasion is that repetition of “printing => revision => re-registration via a scanner and printer” many times produces degradation of image quality such as blurring, distortion, and increase of noise information.

In general, these problems are solved by the computer system as shown in Figure 5.7. When editing marks are written, they should be separated from the original image. Some

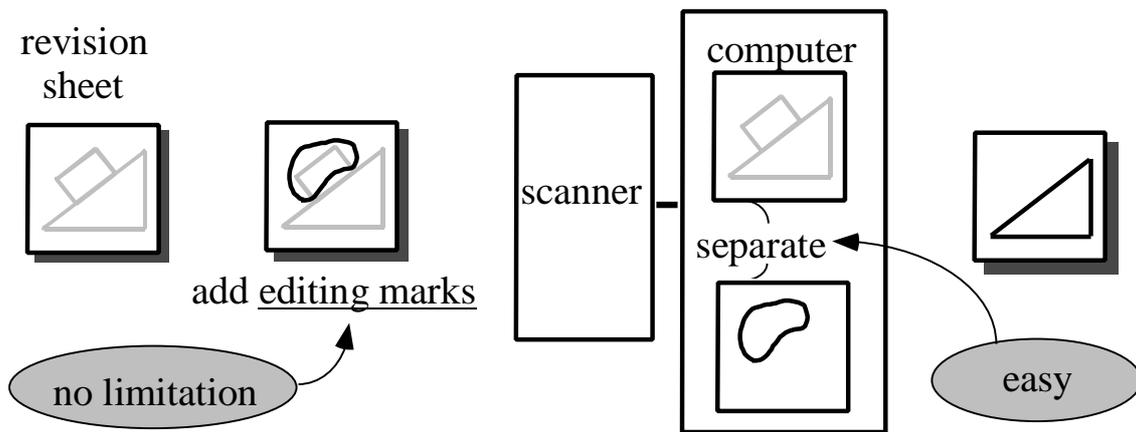


Figure 5.8 New method of hand-written diagram recognition system

extracting process of the editing marks (and correction parts) under the assistance of the computer is needed. Basically, it is sufficient to take matching of the original image and the scanned image in order to extract the editing marks. However, the extraction by matching is not so easy because the image once passed through the scanner is usually distorted non-linearly.

Conventionally, a method to edit by specific colors such as red has been proposed [107]. Another editing method to identify the editing marks by superimposing transparent paper [166] has been proposed. These methods can extract editing marks easily, however, impose some restrictions to users. In the former method, the color for editing cannot be used in the picture. In the later method, the transparent paper tends to slip down the original-image paper. At the same time, restrictions are found with both methods that diagrams and characters identical with the editing marks cannot be written and the editing marks are never permitted to come into contact with the editing data.

In addition, the following problems should be improved for the purpose of putting conventional methods to practical use — introduction of character recognition, macro-definition of the editing marks, application to color images, increase of alignment accuracy of the connected portions of the original images and corrected images, etc.

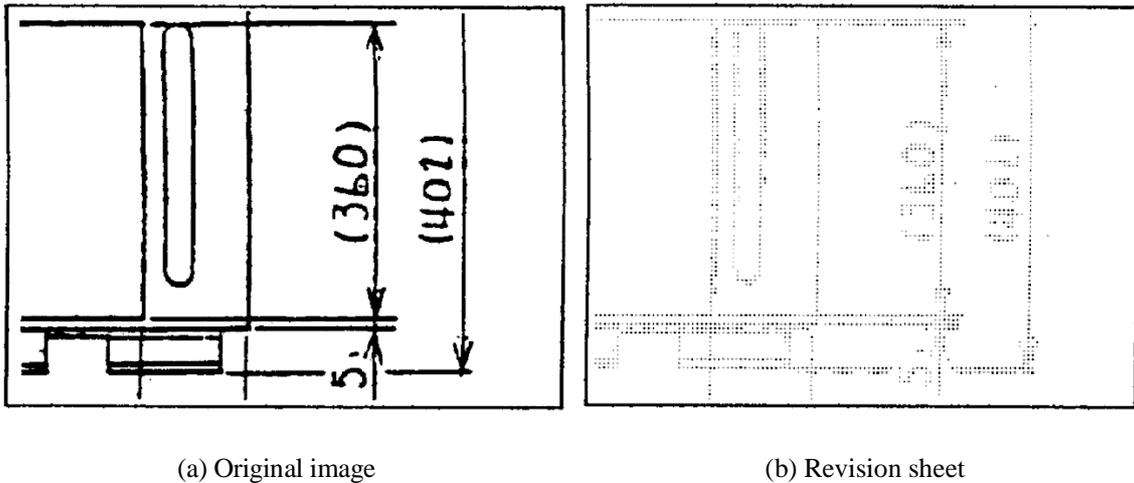


Figure 5.9 Example of dot-masked revision sheet

5.3.2 Dot-Masked Revision Sheet Method

To solve the above problems, a new method using a “dot-masked revision sheet” as shown in Figure 5.8 is proposed [75, 76]. In this method, a special “dot-image output” is used for adding editing marks. This dot-masked revision sheet dissolves limitation in editing marks and helps easy separation of editing marks.

In Figure 5.10, a processing flow of this method is described. The “dotted image” in the figure means an image which is composed by exclusively isolated points of pixel. This image is generated by eliminating the black pixels on the original image in accordance with the following algorithm. In this system, conversion into dots is made by taking logical product of the original image and dot-masked pattern. Illustrated in Figure 5.9 is the example of an original image and its dot-masked revision sheet.

This method takes the condition as a premise that no point with 1–2 pixels is written in general by a human hand. The resolution of printers and scanners used hereby is as high as 200 dpi (dots per inch) represented by G3, the standard of facsimile. In this resolution, 1 pixel (1 dot) becomes approximately 0.125 mm. In the meanwhile, the accuracy of writing tools is, in general, about 0.5 mm, and it is considered that writing with 1–2 pixels (less than 0.25 mm) is difficult. This hypothesis has been confirmed by experiments [75].

From this result, the editing marks can be exclusively extracted by eliminating the isolated black blocks with less than 2 pixels in the re-scanned images.

After the added portions (correction parts) are extracted, recognition of editing marks is performed on it. If needed, it is possible to use some parts-adjusting function to synthesis the corrected portions with the original image.

Color images can also be processed in accordance with the processing flow shown in Figure 5.10.

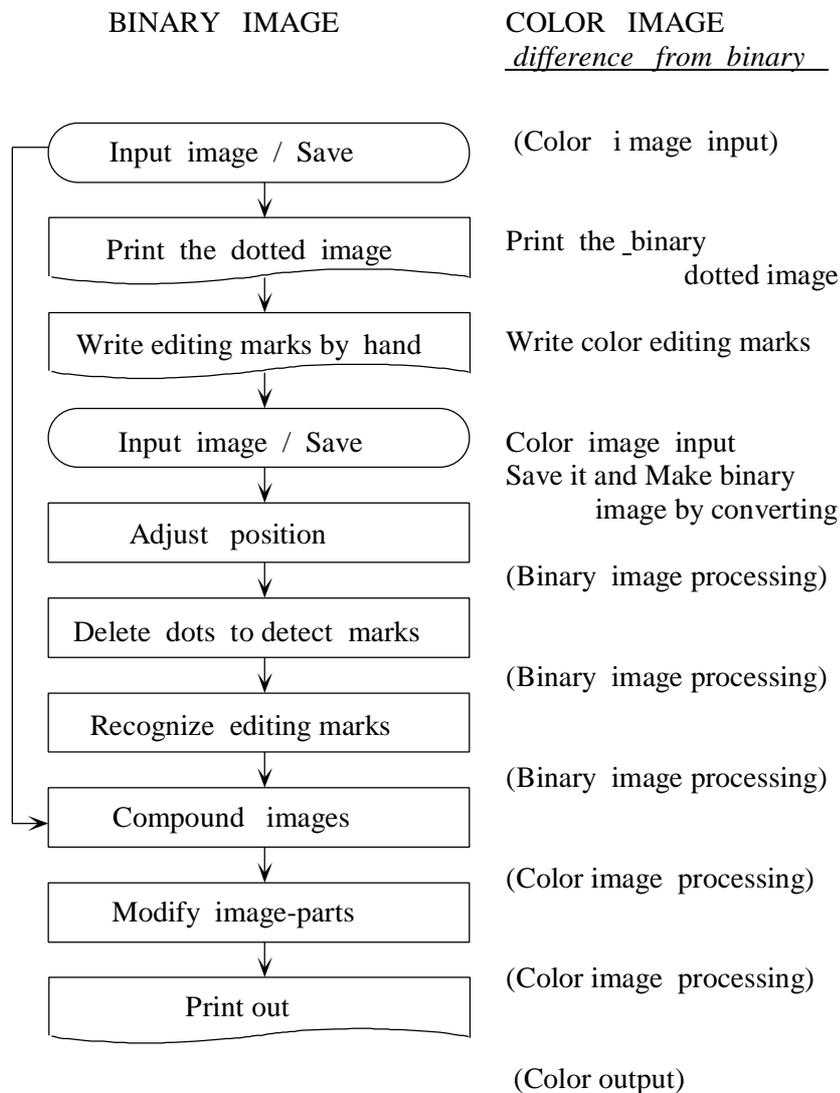


Figure 5.10 Processing flow of the dot-masked revision sheet method

5.3.3 Prototype System “HIDES”

Implementation of the dot-masked revision sheet method described above has been made as a system named “HIDES” (Hand-written Image Data Editing System) [76]. Figure 5.12 shows the sample operation result of HIDES for the binary image. This section describes the following two implementation details:

- Design and recognition of editing marks
- Generation and recognition of dot-masked revision sheets

Design and Recognition of Editing Marks

Table 5.2 shows the editing marks supported in HIDES. The editing marks are composed of two contents: an editing symbol such as “move,” copy,” or “ex” and an

Table 5.2 Editing marks supported in HIDES

operation		editing mark
ADD	delete	
	add	
CHANGE	delete and add	
MOVE	move	
	copy	
	ex-change	

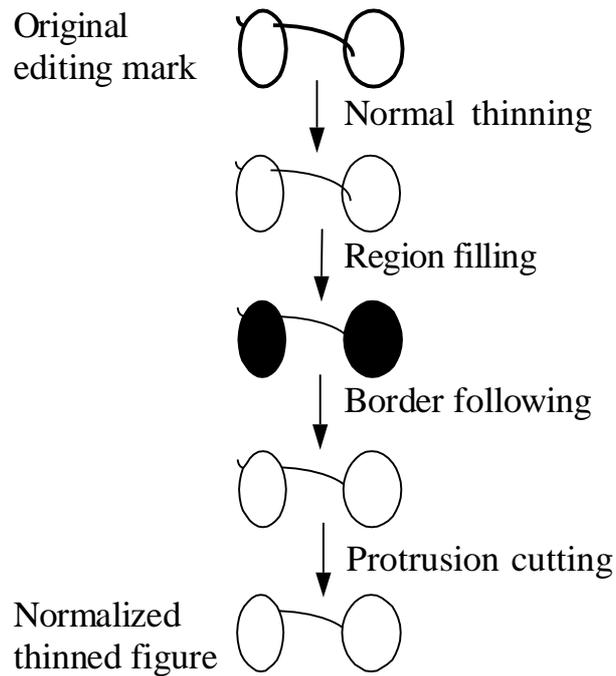


Figure 5.11 Thinning algorithm in HIDES

enclosing mark.

Enclosing marks are extracted by examining the connection number and branching point. Pre-processing considering the contact with the inside editing symbol is required for the extraction. With pre-processing, the algorithm to extract enclosing mark is executed in accordance with the flow shown in Figure 5.11.

First of all, editing mark is extracted by removing dots from the scanned image as shown in later. Then the extracted editing mark image A is copied into memory. For the copied image, line-thinning processing using traditional method [171] are performed. The inside of the thinned lines is filled using the existing algorithm [142]. Border following [171] is applied to the filled region. Through these steps, lines that are branched into the enclosing mark disappear. At the next stage, the short noisy protrusions around the enclosing mark are cut. The criterion of the cut is determined in advance of the procedure in accordance with the lengths of the protrusions.

The figure obtained in this way shall be called a “normalized thinned figure.” Then the connection number and branch-points of the normalized thinned figure are counted in

accordance with the existing method [171]. If the number of the branch-points is 0, then the editing mark is categorized to ADD-type shown in Table 5.1. Meanwhile if the number is 1, then it is treated as a CHANGE-type editing mark, and if the number is 2, it is treated as a MOVE-type editing mark.

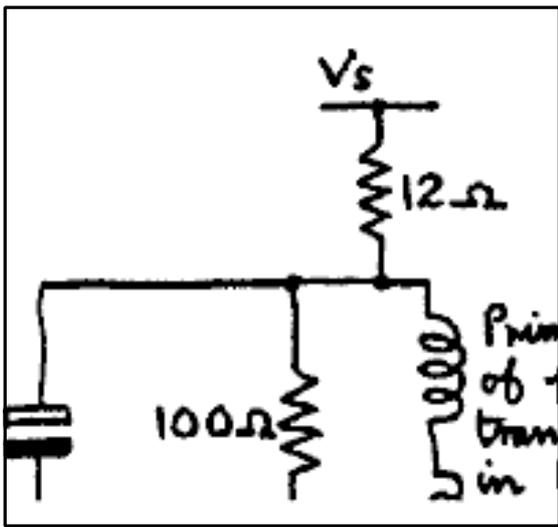
Next, the editing symbol inside the enclosing mark is extracted by removing the enclosing mark from the original image. The normalized thinned figure and un-thinned original image A are compared. If there exist black pixels on A that connect with the portions corresponding to the normalized thinned figure, such pixels are removed. Black pixels connected to those pixels that have been already removed.

Considering the contact of the enclosing marks with their inside images, the pixels in a range at a distance from the normalized thinned figure are regarded as the enclosing marks and removed. The threshold distance is decided as follows. First, edge detection is performed with the original image A, and then the distance between the result (the outermost portion of the enclosing marks) and the normalized thinned figure (the center of the enclosing marks) is calculated and used for the threshold. After the enclosing mark is removed, the inside region is memorized as a “parts image.”

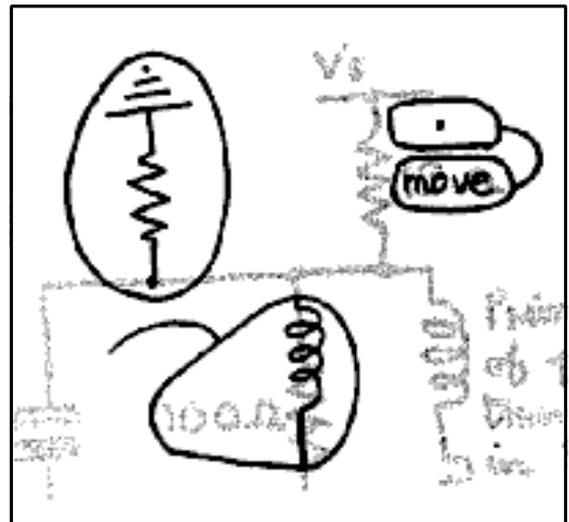
Since the shape of the enclosing marks can be judged from the peripheries of the closed curves, the figures identical with the editing marks in shape can be entered. For example, when a user intends to draw a shape identical to the CHANGE-type editing mark, i.e. a shape having a protrusion from the closed curve, the closed curve, he/she has only to draw another closed curve around the intended shape. The outer closed curve is recognized as the ADD-type editing mark and the inner shape is recognized as an additional drawing data.

Next, recognition of the editing symbol (“move,” “copy,” and “ex” in Table 5.2) is introduced. Hand-writing is sufficient for the editing symbols. The alpha-numerical characters are the specialty to be used for the characters. The conditions are that no contact is made among the individual characters and the characters are already separated from the enclosing marks by the processing referred to above. Perfect recognition of the editing symbols one by one is not necessarily required, and classification into three types,

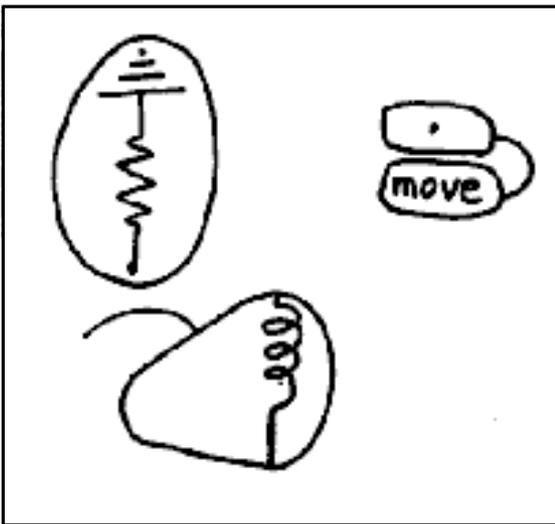
i.e. simple move, copy, and exchange might be sufficient. The alpha-numerical characters can be classified from the circumference distribution of the characters in a vertical direction [124]. This fact is improved and utilized for recognizing the editing symbols. By this recognition algorithm, following merits are generated. The pictures identical with the editing marks can be entered and the enclosing mark is permitted to contact with its inside. Those merits are not achieved in conventional studies.



Original

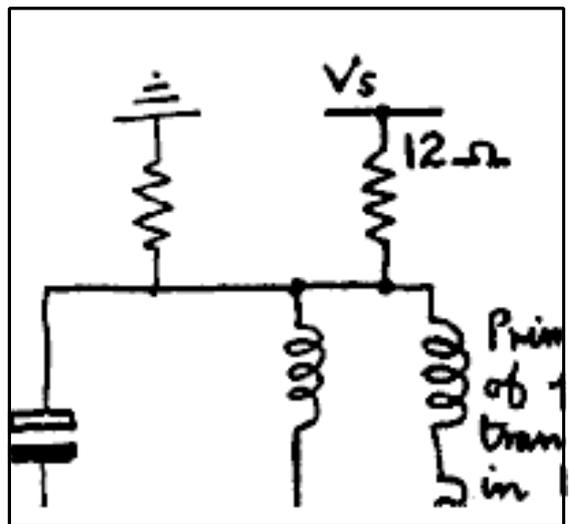


Revision sheet + marks



Detected marks

From CCITT
standard image



Result

Figure 5.12 Sample operation result of HIDES

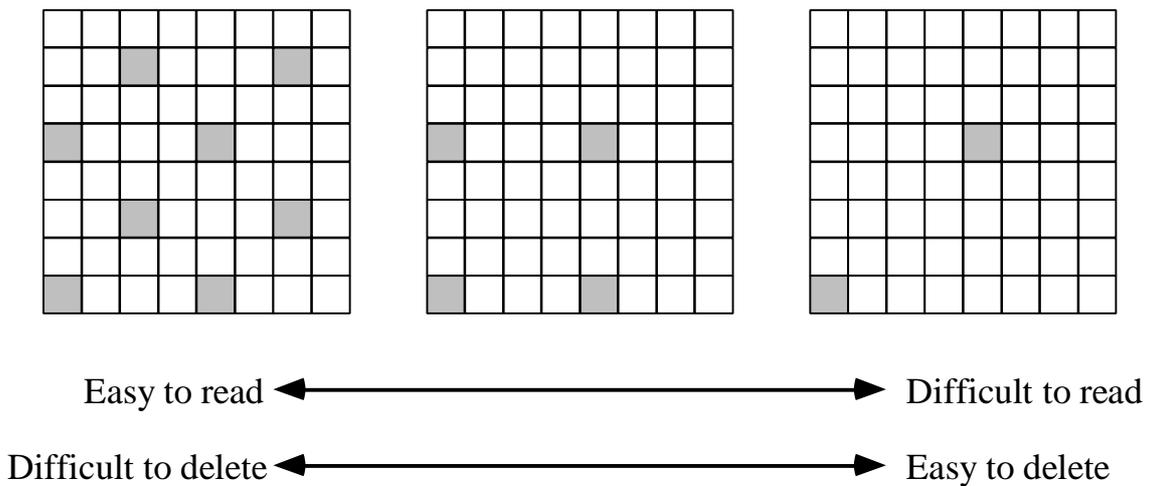


Figure 5.13 Problems in dot-mask pattern

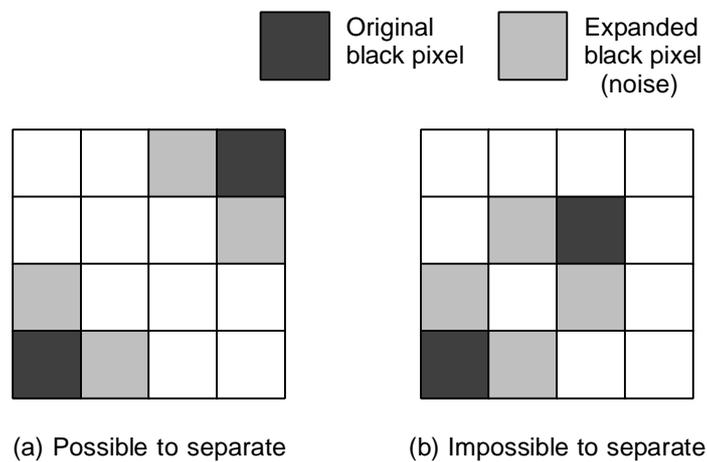


Figure 5.14 The case that dot elimination is impossible

Generation of the Dot-Masked Revision Sheet

For generating the dot-masked revision sheet, a logical product of the original image and dot-mask pattern is taken.

In determining the dot-mask pattern, the following two conditions must be taken into account. Here, the individual black pixel blocks composing the dot-mask pattern is called “isolated blocks.”

Condition 1: The picture converted into dots becomes to be thinner intensity than the original picture, and is poorly visible. Considering the user interface in editing, thicker and finely visible mask pattern is preferable, as shown in the left

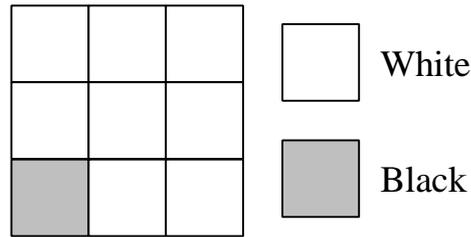


Figure 5.15 Dot-mask pattern adopted in HIDES

part of Figure 5.13.

Condition 2: When a revision sheet is printed and is inputted from the scanner, distinction of the dots from the added pixels becomes difficult if the isolated blocks are connected with each other owing to noise pixels. Broader isolated block interval is preferred for the perfect dot elimination, as shown in Figure 5.14.

In other words, the one with the narrowest isolated block interval is the most preferable in a range where none of isolated blocks are connected with each other via the noise of printer and scanner.

The best interval depends on the combination of printer and scanner. Therefore, it should be examined with the devices in advance. Several mask patterns ranging from the one with a narrower isolated block interval to the one with a broader interval are prepared. After that, the size and the number of the isolated blocks are examined by re-inputting the mask patterns from the scanner. For the prototype, the optimal mask pattern is determined as shown in Figure 5.15 [75, 76].

Dot Removal Algorithm

The experiment revealed that one a pixel is expanded to 2–3 pixels after re-inputted from the scanner. Considering this result, two types of algorithm have been developed to eliminate the dots from images of the revision sheet and to extract the added parts (correction parts).

1. Pixel judgment algorithm

The scanning starts from the upper left corner of the image. When a pixel “a” is

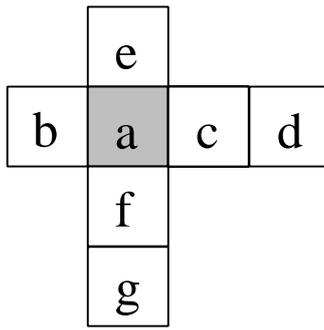


Figure 5.16 Pixel judgment algorithm

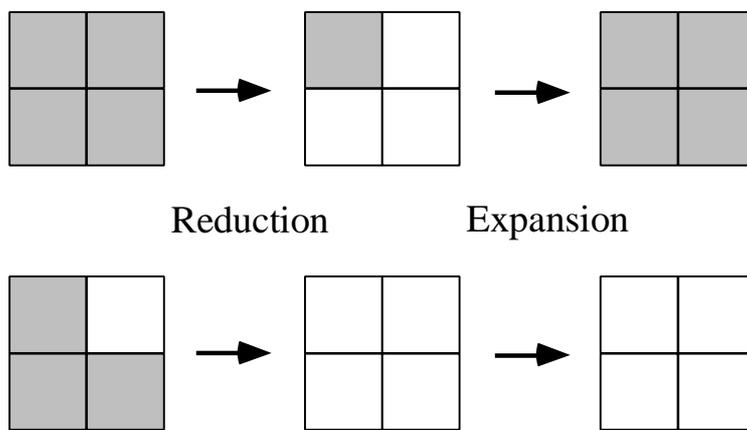


Figure 5.17 Reduction and expansion algorithm

black ($a = 1$), the six pixels “b” to “g” shown in Figure 5.16 are examined. The pixel a is converted to white if one of the following conditions are satisfied:

$$(i) \quad (b = 0) \wedge ((c = 0) \vee (d = 0))$$

$$(ii) \quad (e = 0) \wedge ((f = 0) \vee (g = 0))$$

1. Reduction and expansion algorithm

The dots are removed by the following two steps — reduction step and expansion step.

Reduction step: In Figure 5.17, a reduced image is obtained by leaving the black pixels untouched in case that the three pixels in the right, in the bottom, and in the lower right are all black, and by eliminating them in other cases. The same result can be obtained by the logical product of the three pixels, i.e. an image vertically shifted by one pixel, an image horizontally shifted by one pixel,

and the original image.

Expansion step: Since the black pixel block other than dots is thinned by the reduction step, the block is recovered to the original state by expanding it. The black pixels remaining after the reduction step are expanded in three directions to the right, to the bottom, and to the lower right. The two-direction shift technique can also be used for this step.

In the pixel judgment algorithm, neighbor six pixels must be examined per each black pixel. In the reduction and expansion algorithm, total number of pixel reference is also six per each black pixel since the surrounding three-pixel reference is made twice. But the two-direction shift technique can accelerate its processing.

5.3.4 Evaluation of Image Quality and Processing Speed

Using the prototype system HIDES, the dot-masked revision sheet method was evaluated to show the improvement to the conventional paper-based drawing systems. Three points — editing mark recognition ratio, processing speed, and image quality — were evaluated [76].

First, editing mark recognition ratio was evaluated. The ratio was 99.91%. The recognition ratio of general characters was less than 90%, however, the recognition ratio of the editing symbol with avoidance of the similar strings as a premise was 99.96%. These values are considered to be enough for practical use of the method.

Next, processing speed was measured. With 10MHz 68020, the processing speed of a 200 dpi A4-size sheet (1664 x 2352 dots) was approximately 80 seconds in average with a binary image having 5–10 portions to be corrected, excluding the scanning input. This time was not so attractive but can be reduced to less than 5 seconds by using recent high power PC.

Finally, image quality was evaluated with the 3 items shown below.

(a) Position divergence caused by automatic positioning

From the experiment conducted using A4-size sheets on which enclosed marks for positioning are arranged throughout the surface, following result was

obtained.

$$(\text{position divergence}) \leq 0.10 \text{ mm} / \text{A4}$$

(b) Dot remaining ratio

An experiment was conducted to count the un-removable pixel blocks in dot-masked sheet.

$$\begin{aligned} (\text{dot remaining ratio}) &= (\text{number of isolated blocks larger than } 2 \times 2) \\ &\quad / (\text{total number of isolated blocks}) \\ &= O(10^{-2}) \end{aligned}$$

(c) Degradation ratio of the additional drawing

Are the additional drawings degraded by the dot removal process? The following result denied such an anxiety.

$$\begin{aligned} (\text{degradation ratio of the additional drawing}) &= (\text{total number of black pixels smaller than } 2 \times 2) \\ &\quad / (\text{total number of black pixels}) \\ &= O(10^{-3}) \end{aligned}$$

The result of (a) is a good value because it is smaller than the standard accuracy 0.5 mm of the writing tools used in the business field. For the case that higher accuracy is required, the value was able to be reduced to almost 0 by the translation per pixel using an interactive “parts adjustment mode.” The values of (b) and (c) have been improved compared with the conventional method of repeating “printing => revision => re-registration.” The image quality degradation ratio of this copy repetition method was as follows.

$$\begin{aligned} (\text{image quality degradation ratio}) &= (\text{number of black pixels different from the original diagram}) \\ &\quad / (\text{number of black pixels of original diagram}) \\ &\geq 0.1 \end{aligned}$$

5.3.5 Psychological Effect Caused by Waiting Time of Scanner Input

In the dot-masked revision sheet method, the picture drawn on the sheet is entered through the scanner. Therefore, so a user must wait for a few minutes until the system completely recognizes the whole input image. While waiting, the user does not need to think of the picture, and there must be no psychological overload for the waiting. Actually, however, some users complained “I feel irritated for the too long waiting time.” So, psychological effect caused by the waiting time is discussed here. Psychological situation for the waiting time is subjective one, which cannot be measured by objective method mentioned in Chapter 3. For analyzing such subjective data, study through questionnaire is the general way.

A questionnaire for evaluating the usability of HIDES was carried out in accordance with existing method [86], which evaluates four items through questionnaire study; friendliness, efficiency, understanding, and strength of the system. The subject of the questionnaire study was a group of 100 persons that consists of technicians, engineers, and students in information processing area. The subjects used HIDES before answering the questionnaire.

By analyzing the questionnaire result, the followings were made clear. First of all, the system has the advantage that it is easy to understand the usage. But on the other hand it has weak points that; “To accurately specify positions of [move] or [copy], parts adjusting function should be implemented.” and “The system is not efficient, because waiting time for scanner input is too long.” 87 % persons pointed out the irritation caused by the waiting time.

5.3.6 Evaluation as an Ideal Approach

The proposed method has an advantage in the input operations. That is, in conventional drawing editors, [menu selection] and [object drawing] are needed for the input, but in the proposed method, they are unified into one [object drawing]. It can be concluded that input operation is executed only in the concrete-picture layer .

However, as for editing, the proposed method uses “editing mark,” so the user must choose possible combination of functions available in the system. Thus this method cannot completely banish the intervention of abstract pictures. Accordingly, the effect as an ideal approach is not substantial.

Also, because of batch processing, load on the feedback becomes heavy. Although the feedback operation belongs to the concrete-picture layer in the operation model, the study has shown that there exists new psychological overload of “too long waiting time.”

5.4 Approach 3: Candidate-Selection Method

Approach 1 (sketch-annotation method) and approach 2 (dot-masked revision sheet method) could partially reduce the user's abstract-picture handling, but could not eliminate it completely. Is it possible to leave the whole abstract-picture handling to computer?

5.4.1 A New Drawing Method Using Concrete Pictures

To avoid dealing with abstract pictures, this section proposes a new method. In the new method, the drawing editor manages the command-planning process that users conventionally have to handle. Its basic idea is as follows. When a user draws a hand-written picture, the system displays possible “candidates” by combining conversion patterns (commands) supported by a conventional drawing editor. The user simply has to “select” a desired picture from the displayed candidate pictures. In the conversion candidates, various geometrical constraints which are important for humans such as connection, parallelism, perpendicularity, and symmetry are taken into account. Therefore, the user's psychological load to think of the combination of the editor function (command-planning) is reduced.

Conventional sketch-beautification drawing editors try to improve the efficiency of superficial data input. However, to satisfy various geometrical constraints such as symmetry, editing functions for abstract picture representation are required. The

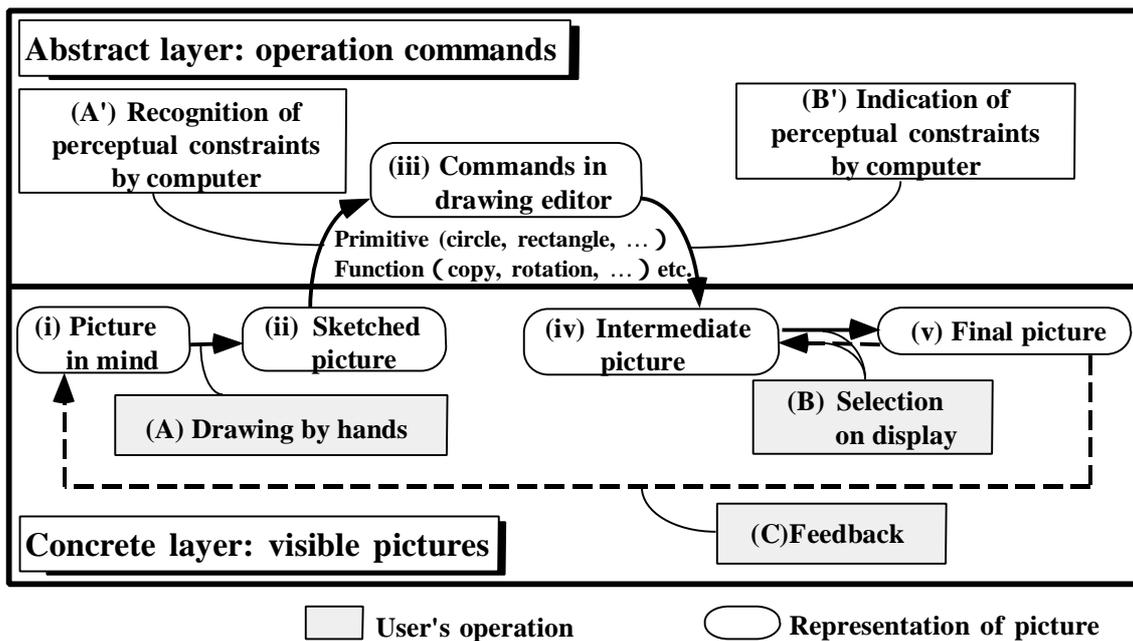


Figure 5.18 An ideal operation model

proposed “candidate-selection method” can reduce such editing stage and decrease users' operation with abstract pictures. The method can be considered as the improvement of the sketch-annotation method shown as the first approach. In the sketch-annotation method, only the shape of input is selected by the user. But in the candidate-selection method, geometrical constraints are also managed in the input stage.

5.4.2 Explanation Using a New Operation Model

The candidate-selection method provides essentially new drawing interface which is different from the conventional operation model. Figure 5.18 illustrates the operation model of the proposed method. In comparison with the operation model for the conventional drawing editors shown in Figure 4.1, the new model is characterized by that the user need not handle any abstract expression.

The abstract-picture layer is handled by the computer system. The user inputs a picture desired to be drawn (i) by hand-writing as a sketched picture (ii) (A). Then the computer system recognizes the sketched picture and builds up editor's commands (iii) internally with considering various geometrical constraints (A'), and displays a set of

possible pictures as recognition candidates as an intermediate picture (iv) by the drawing editor (B'). The user then selects (B) an appropriate intermediate picture to obtain a chunk of final picture (v). Visual feedback (C) from the final picture to the picture desired to be drawn is the same as that for the conventional model.

Hand-written sketch input operation is adopted so that a concrete picture can be directly inputted to the computer. However, as can be seen in conventional sketch-beautification editors, if the same method used by conventional object-oriented editors is used in the editing stage, it also will need abstract-layer operations. In the new operation model, most of operations⁴ performed in the conventional editing stage such as processing of contact point and symmetry have already been done in the recognition step (A'). However, since it is very difficult to completely judge what editing is intended by the user, multiple possible results are presented to the user so that the user can make the appropriate choice.

According to the human interface guidelines of Apple Computer, Inc., “re-recognition and selection” interfaces, like GUI, in which all usable instructions are displayed and selections can be made from them, are better than “memory and input” interfaces in which the user must remember the instructions to be entered on a command line [3]. It can be said that the proposed model incorporates the re-recognition and selection method into the drawing editor interface.

Comparing the conventional approaches on drawing editors, the effect of this method is expected to be high. The prototype system and the evaluation of the candidate-selection method will be described in the following chapters.

⁴ Note that if step (A) is done with guessing some recognition function, it may contain some command-planning in abstract layer.

5.5 Summary of Approaches to an Ideal Drawing Editor

This chapter has examined requirements for ideal drawing editor on the basis of the operation model proposed in the preceding chapter. Three possible approaches to ideal drawing editor have been shown. Among them, it was found that approaches 1 and 2 can achieve only partial improvement. But the third approach is expected to be the most suitable for the requirements because it can reduce user's abstract-picture handling.

Following chapters will describe the implementation of this candidate-selection method, which is the core part of this thesis, and evaluate its operations to prove the effectiveness.

6 The Candidate-Selection Method

The previous chapter proposed the “candidate-selection method” based on the ideal drawing model which reduces the user’s operation with abstract pictures and reduces the user’s psychological load. A prototype system based on the proposed method, named “GIGA” [82, 83, 64, 65, 84], will be presented in this chapter. First, the interface of the new system is discussed, and then the system structure and detail processing methods are described. Possible problems of the interface are also discussed.

6.1 Basic Ideas of the Candidate-Selection Method

Here, key consideration points of the candidate-selection method are summarized.

- The candidate-selection method aims to reduce user’s operation in the abstract-picture layer in the two-layer operation model.
- At the beginning of drawing, free-hand sketch is used to input the concrete picture directly.
- According to the analysis of conventional sketch-beautification drawing method, it is resulted that the recognition of the inputted picture should be performed considering some human perceptual geometrical constraints.
- Conventional sketch-beautification editor displays only one result of beautification. When a bending line is inputted, there is two possibilities: this is a straight line drawn in a bad manner, and this is really a bending line. However, the conventional system discards the possibility and decides only one result, which increases the re-drawing. To meet the various inputs and user’s preferences, multiple candidates of recognition results should be displayed.
- Conventional sketch-beautification editors require the editing operation similar

to that of object-oriented drawing editors — copy, move, rotate, etc., which resides in the abstract-picture layer. If the recognition result has already satisfied all the necessary geometrical constraints, most editing operations become unnecessary.

- To display all possible candidates to eliminate the editing operation, it is most important to choose the useful geometrical constraints. Human perceptual constraints should be accounted.

6.2 Advantage of Human Perceptual Constraints

In the candidate-selection method, “human perceptual constraints” such as connection, alignment, symmetry, parallelism perpendicularity, special angle (horizontal, vertical, 45-degree), etc. are used as the geometrical constraints to be satisfied.

What are the geometrical constraints used in conventional systems? Conventional sketch-beautification editors can perform the recognition (beautification) on limited primitives such as lines, rectangles, and ellipses. Therefore, when the recognition fails, re-drawing is usually needed. If the pictures that are not prepared as recognizable primitives have to be drawn, it takes a long time.

However, actually it is possible to draw many kinds of pictures using a few constraints. The shapes which should be recognized can be uniquely determined using *perceptual constraints* such as symmetry, parallelism, or perpendicularity. For example, a rectangle is obtained by making the corners right-angled or sides horizontal and vertical. For drawing a rhombus, constraints to keep the four lines symmetrically can be applied. Most pictures can be formed by dividing into segments and then using perceptual constraints. Perceptual constraints free the picture-drawing from pre-defined shapes.

Among various perceptual constraints, symmetry has a significant role in the proposed drawing method, although it is rarely used in usual recognition algorithms. The use of symmetry enables frequently-used primitives such as rectangles, rhombuses, ellipses, and

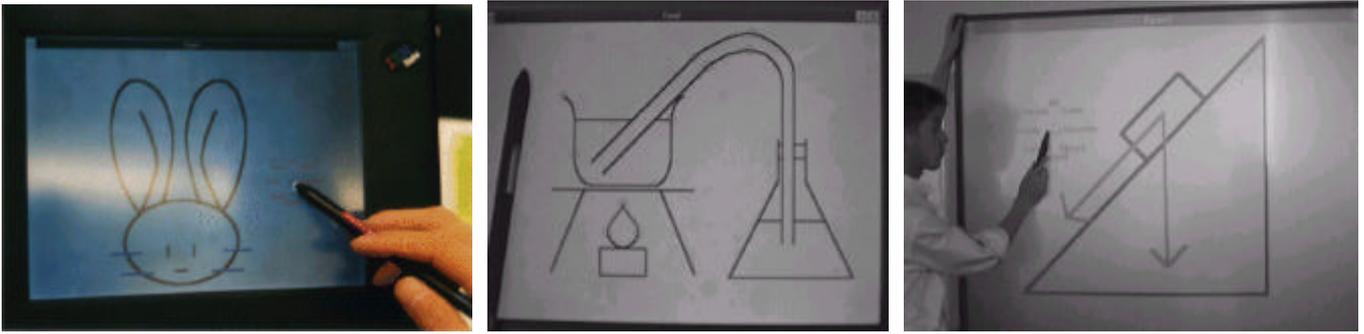


Figure 6.1 Example drawings with GIGA

concentric circles to be drawn without using special recognition rules. Symmetry is essentially superior to local similarities.

The interface of displaying multiple possible candidates not only frees the picture-drawing from fixed shapes but also provides other merits. In conventional constraints-based drawing systems, how to handle inconsistent constraints entered by users is a problem. The candidate-selection method solves the problem by displaying possible candidates according to the human perceptual constraints and having users to select a candidate. In conventional sketch-based drawing systems, increasing the recognition ratio is an important issue. However, determining whether the recognized result is good or bad is usually subjective, and 100% assured recognition is impossible. The method displaying possible multiple candidates, using human perceptual constraints, can be considered as one solution of the recognition ratio problem.

6.3 Prototype System “GIGA”

A prototype system named “GIGA” (Graphics with Interactive Geometrical Assistance) that employs the candidate-selection method is developed using Visual Basic and Visual C++ on Windows. The word “GIGA” is named after “Giga” in Japanese old artwork “Choujuu-Giga.” In GIGA, connections of endpoints, intersections with certain angles including parallelism and perpendicularity, horizontal or vertical, alignment of values of X or Y, symmetry, similarity, equalization of lengths, etc. are implemented as constraints used in the recognition step. Figure 6.1 shows example drawings using GIGA.

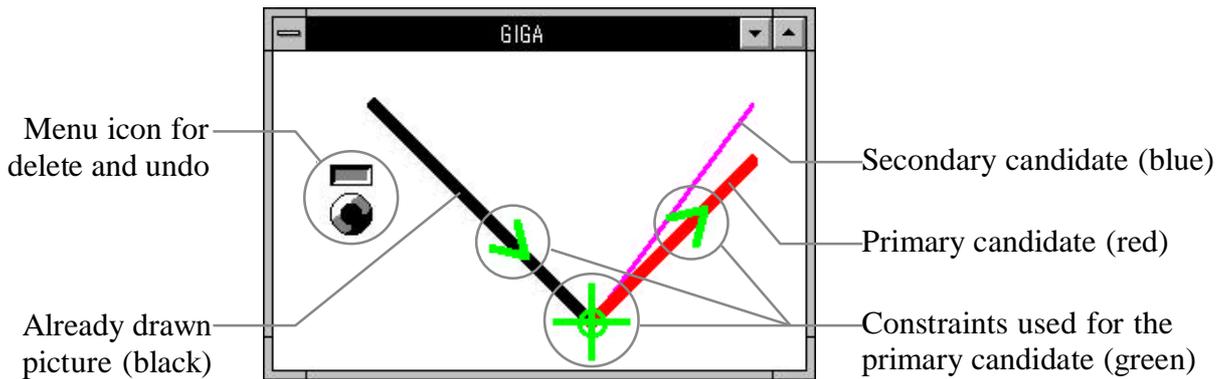


Figure 6.2 Candidate-selection method, GIGA

6.3.1 Actual Operation

Actual operation method of GIGA is as follows. First, a user directly sketches one line chunk of a desired picture on the editor screen with free strokes of mouse or pen. Then, the system recognizes the input and calculates possible candidates that satisfy some of the above mentioned constraints [65]. As a result, candidates are displayed with the constraints used for the candidate, such as which parts are vertically connected with which parts, to enable the user to make quick and assured selection. The primary candidate is displayed in red while other candidates are displayed in blue, and constraints used for the primary candidate are displayed in green, as shown in Figure 6.2. Thus, the screen does not become complicated and prevent operations.

If the primary candidate is the desired one, the user can proceed to draw the next line chunk. If the primary candidate is not the desired one, the user can select another candidate. If desired candidate is not displayed, it means that first sketching is too poor to be recognized correctly or that the intended input has some conflicts in its constraints. In the former case the user can get the desired candidate by re-sketching the input carefully.

The displayed candidates already incorporate various geometrical constraints, and there is few case that editing operation should be performed. Thus, GIGA only supports minimum editing functions, delete and undo.

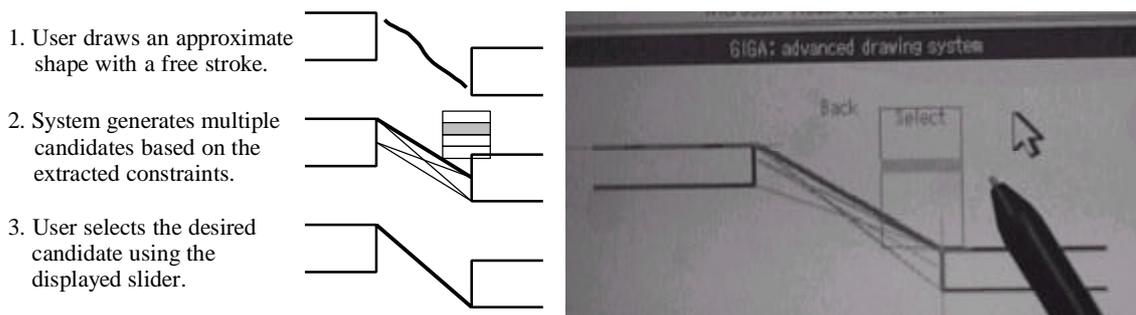


Figure 6.3 Interaction of selection from multiple candidates, using slider

Several interfaces can be considered on how to select a candidate and how to delete a segment in a picture.

As the selection interface, two methods are actually implemented. One is the direct selection in which a candidate is directly selected by clicking or tapping.⁵ The other is the selection using slider as shown in Figure 6.3. Using the slider, the selection area can be wide enough to avoid the problem of parallax in pen-computing. Moreover, the slider interface works effectively when the candidates are close with each other and hard to be selected directly. However, the slider disturbs the drawing because it occupies the drawing area on the screen. Furthermore, the most important fact is that the indirectly selection contains the abstract operation. Therefore, the direct selection is adopted in the prototype and evaluation experiments.

As the deletion interface, two gestures are implemented: trimming and stopping. The trimming gesture means to input a W-shape on the segment to be deleted. This gesture has already been implemented in several pen-based systems such as Newton from Apple Computer, Inc. The stopping gesture is newly proposed, which means that the cursor stays on the segment with holding the right mouse button or the pen button for a while. However, it is not preferable to use the pen button because the user must always be careful not to push the pen button and has to re-hold the pen when the pen button needs

⁵ Tapping means to knock the electronic panel by the electronic pen, corresponding to the clicking by mouse. The button on the electronic pen usually corresponds to the right button of the mouse.

to be pushed. It is time consuming to re-holding the pen. Thus the trimming gesture is adopted in the prototype and evaluation experiments.

6.3.2 System Structure

The above-mentioned operations are realized by the following system structure as shown in Figure 6.4. There are three possibilities in the inputted data: segment input, candidate selection, and deleting gesture. First the system classifies the input-data into those three kinds with considering the operation context. If the selection of a candidate is performed (10), the system changes the current candidate to newly selected one (11). If the input is not a candidate selection and the candidates are displayed, the system fixes the current candidate as the user's desired input (12) and continues the following input recognition. If the deletion gesture is inputted (20), the system deletes the specified segment (21). If the input is not selection or deletion, it is treated as a new segment input. The system generates the new geometrical constraints based on the inputted stroke (30), solves the geometrical constraints (31), displays the new candidates (32), and waits for the next input.

Throughout the processing, the internal information for generating and solving geometrical constraints is updated step-by-step. The details of the generation and solution of geometrical constraints are described in the next subsection.

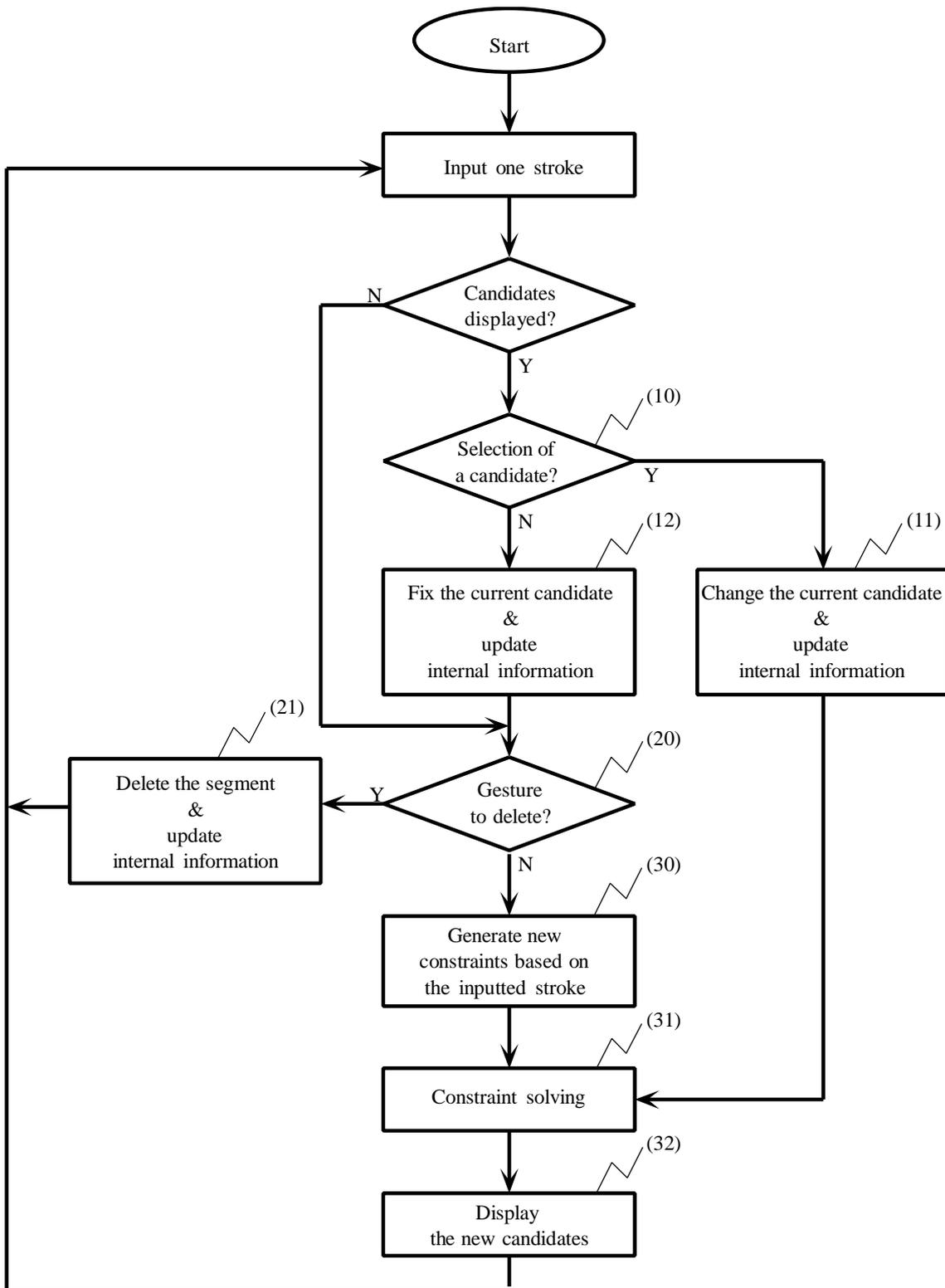


Figure 6.4 Processing flow of GIGA

6.3.3 Generating and Solving Geometrical Constraints

For generating new constraints based on the inputted stroke, first the relations between the inputted stroke and already drawn pictures are checked. Four parameters, X and Y coordinates of the start and end points, are compared between the new segment and each drawn segment. Then the slope of the stroke, calculated from those four parameters, is also examined.

Next, perceptual constraints are examined. Comparing the X and Y parameters of the inputted segment with those of existing segments, the constraints for alignment are generated as the equations such as “ $X_{\text{new}} = X_{\text{old}}$.” Then the slopes are compared between the new and existing segments, and constraints for special angles such as parallelism, perpendicularity, and symmetry are generated as the equations.

To create multiple candidates, a constraint solver which can calculate multiple solutions from excess amount of equations is developed [65]. Constraints are solved in the order of strength to create the primary candidate. After that, unsatisfied constraints are strengthened and solved repeatedly.

If the number of already drawn segments is n , the total number of geometrical constraints becomes $O(\log(n))$ because the parameters of already drawn segments are sorted. The time for creating multiple candidates depends on the time for searching the sorted parameter table. Thus the time taken to solve the constraints never becomes excessively long. Also, thresholds are set for the strength of constraints, and candidates with weak constraints are not displayed in order to reduce the number of candidates displayed on the screen. The response time to obtain the candidates is less than 1 second for the pictures shown in Figure 6.1 using an IBM PC compatible machine with i486 75MHz. Therefore, interactive operation is possible.

6.4 Possible Problems of the Unique Interface

In the next chapter, GIGA is widely evaluated and analyzed by comparing it with conventional drawing editors in quantitative manner. The following two problems specific to the interface of the prototype system GIGA must also be discussed:

1. Problem of chunk size

The unit of candidate display, or a specified unit of free strokes, is called the “chunk” here. The chunk size of the prototype system is a single free stroke; specifically, a line, polygon, or circle that is drawn by a single stroke. This chunk size almost corresponds to a primitive (or object) of the conventional object-oriented drawing editor. It must be examined whether the chunk size is too small and causes too frequent candidate selection or not.

The prototype system allows the user to input the next free stroke without selecting a candidate when the primary candidate is desired. For this reason, if the rate of selecting the primary candidate is high, the user is not disturbed by frequent candidate selection. Experiments must be conducted to find the rate of selection of the first candidate.

2. Problem of re-drawing

If the candidate for the picture intended by the user cannot be displayed, the user must re-sketch the picture many times. Using the prototype system, experiments must be carried out to find the rate of failure in displaying the desired candidate and re-sketching is performed.

7 Evaluation of GIGA

In this chapter, the new drawing editor GIGA based on the candidate-selection method described in the preceding chapters is examined to prove that it can reduce the overhead in drawing operation analyzed in Chapter 3. First of all, it is necessary to examine the effect of the design that lets the user participate only in the concrete-picture layer. For this purpose, the same experiments and analyses as those described in Chapter 3 have been carried out using GIGA. The following experiments were carried out with the same subjects and pictures:

Exp. A First, subjects were asked to draw the pictures with verbally announcing their intention. The drawing sessions were video recorded for later analysis.

Exp. B Next, they were asked to watch the video of Exp. A, understand the drawing procedure used by themselves, and re-draw those pictures *as fast as possible* with no wasted operation or drawing-strategy planning.

Exp. C Subjects were asked to draw several arbitrary lines on the screen by both mouse and pen. Actually, this experiment does not depend on GIGA, and the results in Chapter 3 (Table 3.8) can be used.

The experiments using GIGA and the previous experiments using Canvas do not affect each other because GIGA and Canvas are based on completely different drawing operation models. Actually, although half of 13 subjects performed this experiment first, the order of the experiments did not influence on the results.

An electronic pen and a mouse were used as the input device for this experiment to compare the result with the previous experiments.

The following seven items are analyzed and discussed. Items 1 to 4 and 6 are same analyses as shown in Chapter 3, and results of items 1 to 6 are compared to those of Canvas and SmartSketch.

1. Variation of drawing strategies
2. The number of actual and ideal operation steps
3. M/O/N (MENU/OBJ/NON) analysis
4. Four-layer analysis of actual drawing time
5. Relation of picture's complexity and its drawing time
6. Analysis using more complex pictures
7. Analysis of the effect of the chunk size in GIGA

7.1 Variation of Drawing Strategies

As shown in Chapter 3, there exist multiple drawing strategies with Canvas and SmartSketch. However in experiment with GIGA, only one strategy was observed for drawing each picture. That is, to draw the line elements of the picture one-by-one. Therefore, the user is never at a loss in choosing a drawing strategy. It can be concluded that with GIGA the load for considering which strategy should be used for the drawing is much smaller than other conventional drawing editors.

7.2 The Number of Wasted Operations

Figure 7.1 shows the number of drawing operations of each subject and Figure 7.2 compares the number of operations in GIGA and those in other conventional drawing editors by accumulating the results of 13 subjects. In drawing with Canvas or SmartSketch, extra menu-selection operations were needed because of the interface design. On the contrary, GIGA needs no such extra menu-selection step, and thus the number of ideal operation step shown in Exp. B is smaller than others. Moreover, it is important that the difference between the number of ideal operation steps (Exp. B) and that of actual operation steps in Exp. A is also smaller than that of other conventional drawing editors, which means that the number of unnecessary or mistaken operations is smaller in GIGA. These good tendencies with GIGA can be observed as regardless with subjects.

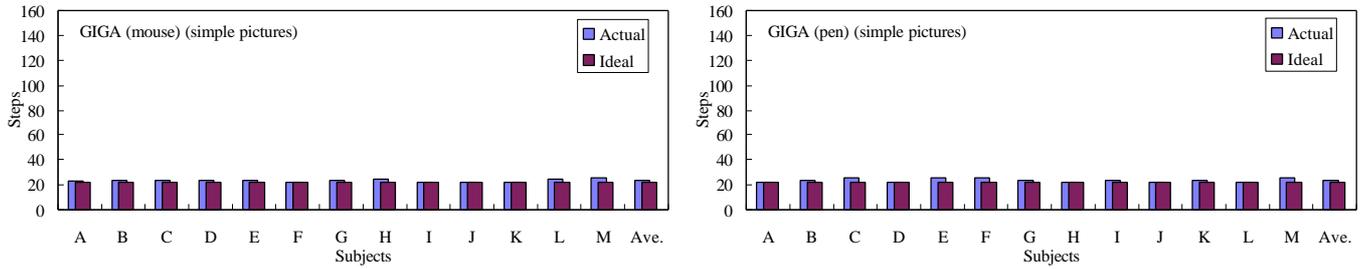


Figure 7.1 Operation steps of drawing the simple pictures with GIGA

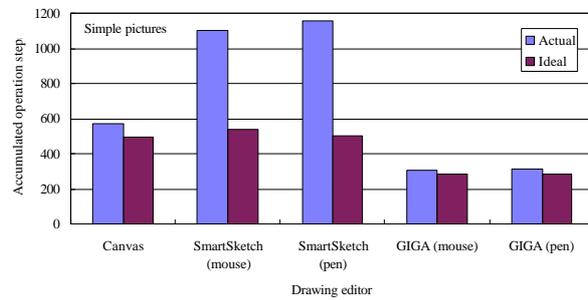


Figure 7.2 Comparison of operation steps of drawing the simple pictures among editors

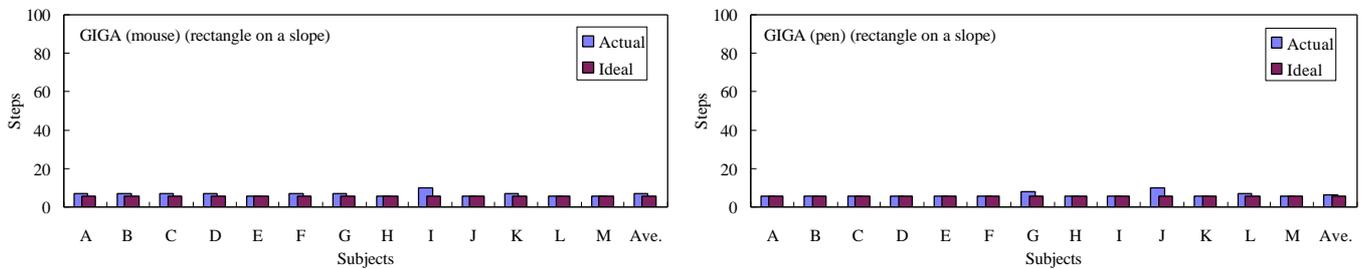


Figure 7.3 Operation steps of drawing the “rectangle on a slope” with GIGA

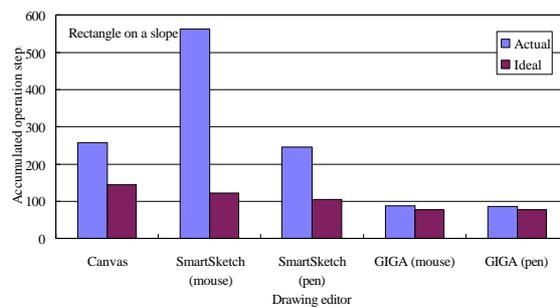


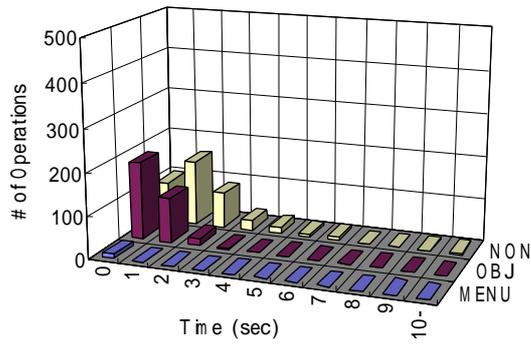
Figure 7.4 Comparison of operation steps of drawing the “rectangle on a slope”

Such tendency is more clearly observed in the results of more complex picture “rectangle placed on a slope” as shown in Figure 7.3 and Figure 7.4.

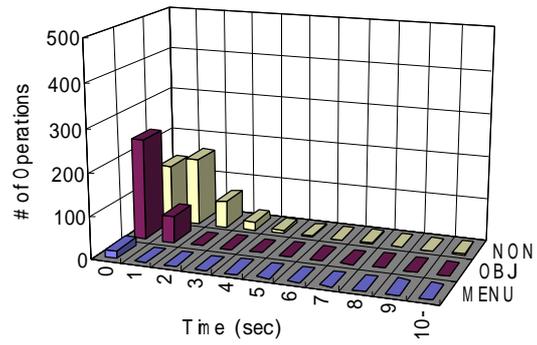
7.3 M/O/N Analysis

Next, the length of each operation step was analyzed using the MENU/OBJ/NON classification. The contents of drawing operation in Exp. A are classified into “time used for menu selection (MENU),” “time used for directly handling an object (OBJ),” and “time used for other operations (NON).” The time for candidate-selection with GIGA is regarded as menu selection (MENU). Figure 7.5 shows distribution of these operations taken by 13 subjects. Compared with Canvas and SmartSketch (Figure 3.4), the MENU time was considerably reduced. Moreover, the trouble such as thinking the drawing strategy is small, which can be resulted from the fact that almost all operations finished in less than 5 seconds. In GIGA the number of operation steps required more than 10 seconds was less than 0.5% of total operation steps.

Similar results were achieved on more complex picture “rectangular placed on a slope” as shown in Figure 7.6. With Canvas or SmartSketch, there exist lots of operations which take more than 10 seconds in drawing this picture, as shown in Figure 3.5. In the case with GIGA, such long time operations were eliminated.

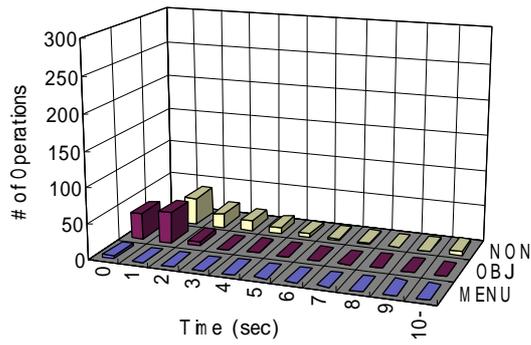


GIGA (mouse) (simple pictures)

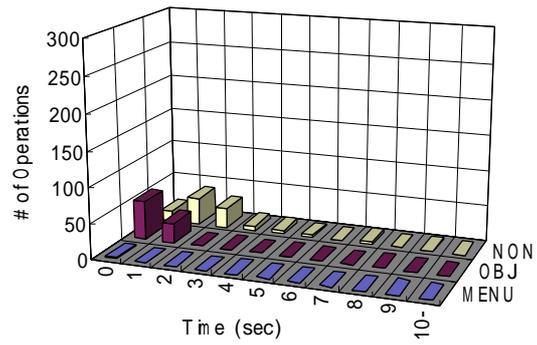


GIGA (pen) (simple picture)

Figure 7.5 M/O/N analysis of drawing the simple pictures with GIGA



GIGA (mouse) (rectangle on a slope)



GIGA (pen) (rectangle on a slope)

Figure 7.6 M/O/N analysis of drawing the “rectangle on a slope” with GIGA

7.4 Four-Layer Analysis

Next, using the result of Exp. A, B, and C, the operation time was divided into the following four layers as described in Section 3.5.

T₁ Time of wasted operations (trial-and-errors and mis-operations)

T₂ Overhead time by immature drawing (including the planning)

T₃ Indispensable overhead time for alignment, etc.

T₄ Physical minimum time to draw the picture

Figure 7.7 shows the results of each subject. On each drawing editor, the accumulated results are shown in Figure 7.8.

It can be observed that the psychological overhead time ($T_1 + T_2$) was much reduced in GIGA. The time of wasted operations (T_1) was reduced because trial-and-errors by wrong strategies is reduced, and mis-operations like in SmartSketch seldom occurs. The overhead time by immature drawing (T_2) was reduced to less than half of conventional drawing editors because the strategy planning is not needed in GIGA. These fact means that psychological load can be reduced by the candidate-selection method.

In GIGA, the indispensable overhead time (T_3) was also reduced. The reason can be considered that additional editing operations for satisfying geometrical constraints and alignment are not necessary in GIGA. By this, the expertised drawing time ($T_3 + T_4$) reached about 1.8 times of the physical minimum time (T_4), while the values were about 4 to 5 in conventional drawing editors.

As a combination of the time reduction in each layer, total drawing time ($T_1 + T_2 + T_3 + T_4$) of GIGA was much shorter than those of conventional drawing editors. Drawing time with GIGA (pen) was only 42% of Canvas and 20% of SmartSketch (pen).

It is also observed that the difference among subjects was not so large with GIGA. This means that the drawing time is not affected so much by the user's drawing skill. Even beginner users can draw fast with GIGA.

Similar results were achieved on more complex pictures. Figure 7.9 and Figure 7.10 show the four-layer classification of operation time in drawing the “rectangle placed on a slope” picture. In this figure, the advantage of GIGA, such that it needs a little time for the wasted operations (T_1) and strategy planning (T_2), is shown more clearly. It is important that in GIGA T_2 did not increase so much even for drawing the complex picture, unlike Canvas.

Here, the operation models mentioned in previous chapters are considered again. As described in Chapter 4, the load of command-planning ((A) in Figure 4.1) is considered to be large in traditional drawing editors. In Norman's model (Figure 4.2), the command-planning corresponds to the steps of “establishing the goal,” “forming the intention,” and “specifying the action sequence.”

The proposed operation model (Figure 5.18) is designed to reduce such an overload in the command-planning. As described in Section 3.5.4, the overhead corresponds to the time of $T_1 + T_2$. From the experiments, it is observed that the time is actually much reduced in GIGA. This shows that the command-planning is reduced in the candidate-selection method.

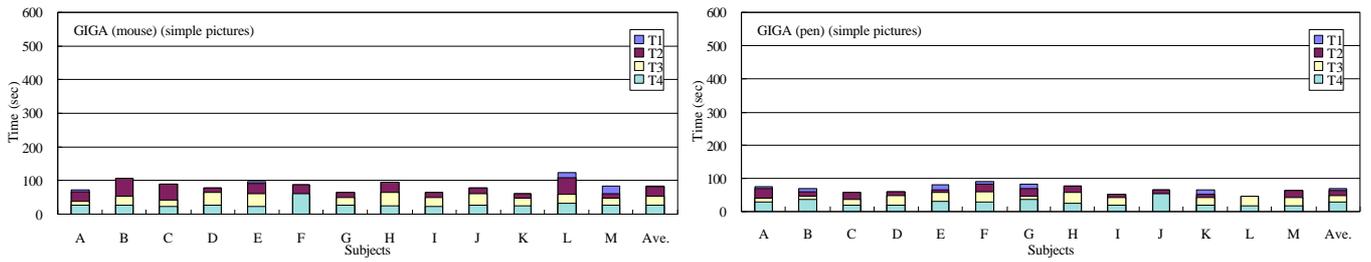


Figure 7.7 Four-layer analysis of drawing the simple pictures with GIGA

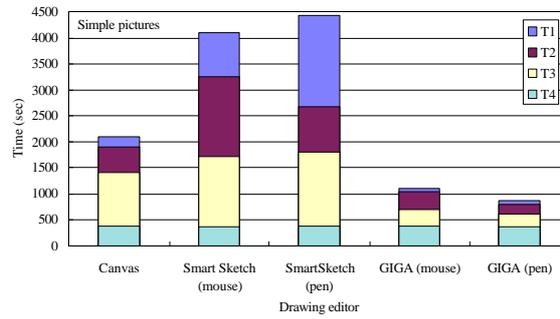


Figure 7.8 Comparison of operation times of drawing the simple pictures among editors

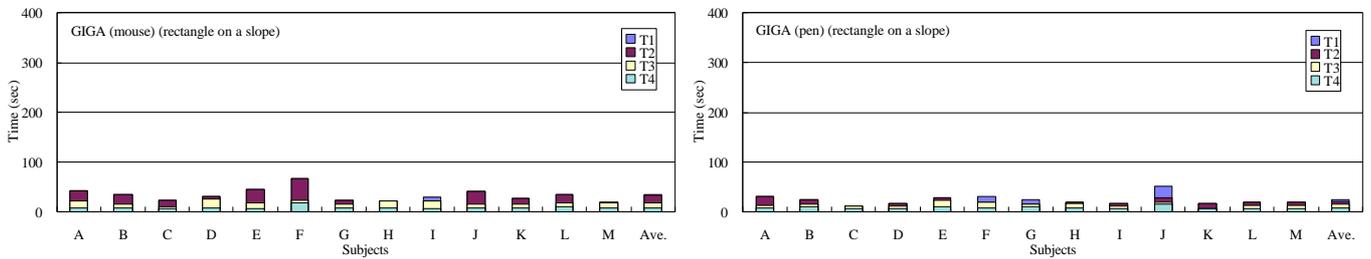


Figure 7.9 Four-layer analysis of drawing the “rectangle on a slope” with GIGA

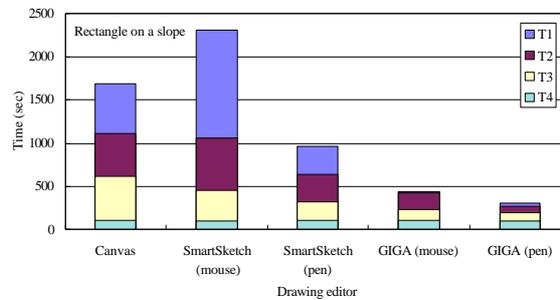


Figure 7.10 Comparison of operation times of drawing the “rectangle on a slope”

7.5 Relation of Picture's Complexity and Its Drawing Time

Next analysis has been done to show the characteristics of the candidate-selection method. Figure 7.11 shows a relation between the number of line elements of each picture (X-axis) and its operation time (Y-axis). Actual operation time in Exp. A (T_A) was used

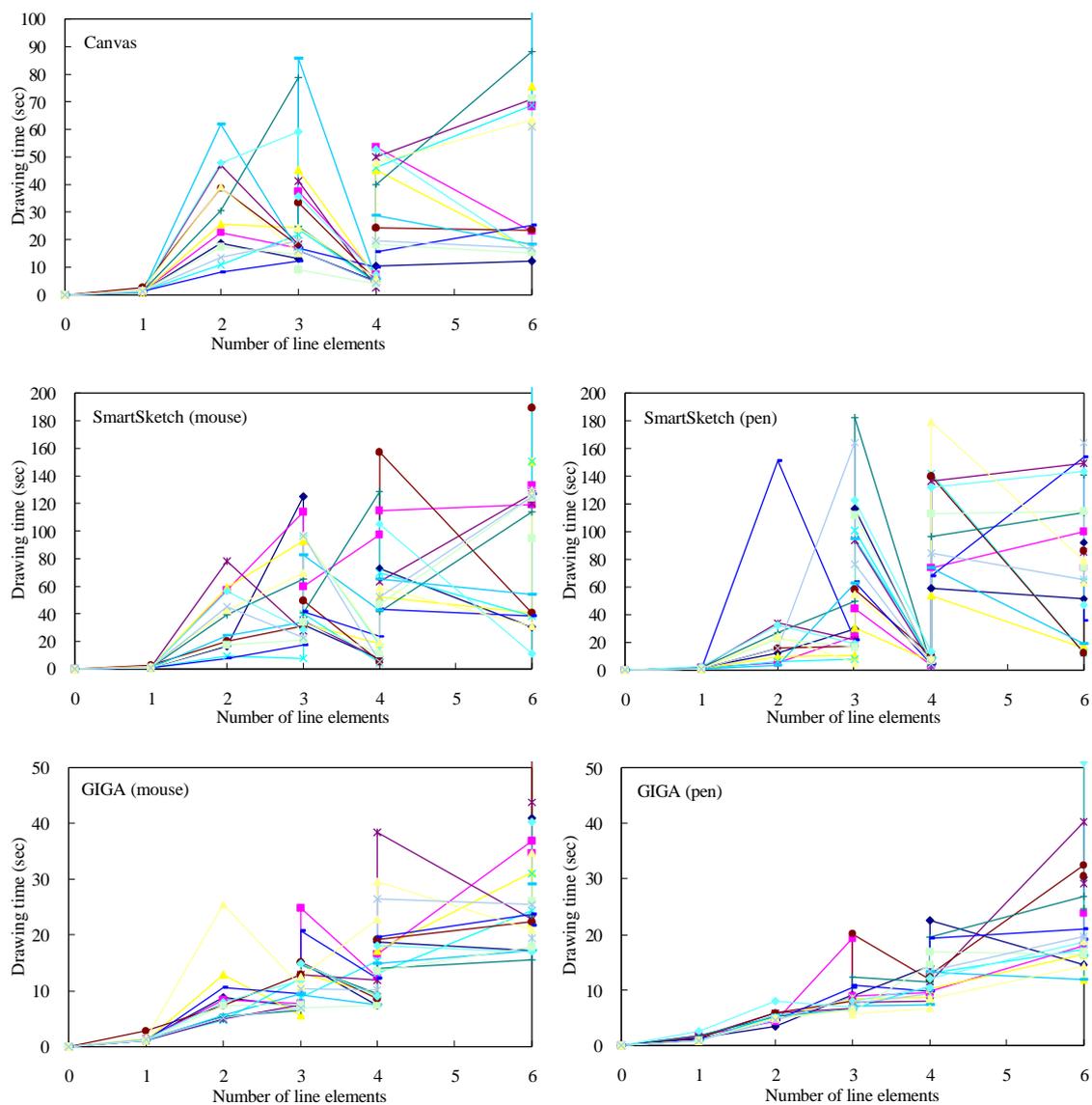


Figure 7.11 Relation between the number of line elements and its drawing time

for plotting except that Exp. C result (T_C) was used as the one-line ($X = 1$) time. Each line in the graph corresponds to each subject.

With Canvas and SmartSketch, operation time for rectangle and rhombus differs very much although both pictures have the same number of lines, 4 ($X = 4$). This is because rectangle can be drawn easily by rectangle tool with Canvas or by using strong recognition designed for rectangle with SmartSketch, while rhombus cannot be drawn by such special functions and needs strategy planning. In those conventional editors, there is no clear relation between the number of lines and operation time. Therefore, it is hard to estimate the workload of drawing from the visible complexity such as the number of lines. This is caused by that abstract pictures are handled in drawing with these editors.

On the other hand, clear proportional relation can be observed between the number of lines and operation time in drawing with GIGA. It is therefore concluded that the number of lines can be used as a hint how the picture is complicated and difficult to be drawn. This result endorses the fact that drawing operations in GIGA are performed in the concrete-picture layer just according to the model shown in Figure 5.18.

As supplementary data, Figure 7.12 shows the relation in the expertised experiment (Exp. B) with GIGA (pen). The relation becomes clearer when a user is well-practiced with GIGA. Figure 7.13 is a four-layer analysis of each picture accumulated by 13 subjects. With Canvas or SmartSketch, the drawing time became much short for the pre-defined primitive object (rectangle). But the time increased for a picture that cannot be drawn as a primitive (rhombus, etc.). And the time T_1 and T_2 became very large for drawing a complex picture (rectangle on a slope). On the contrary, the four layers increase almost equally with GIGA.

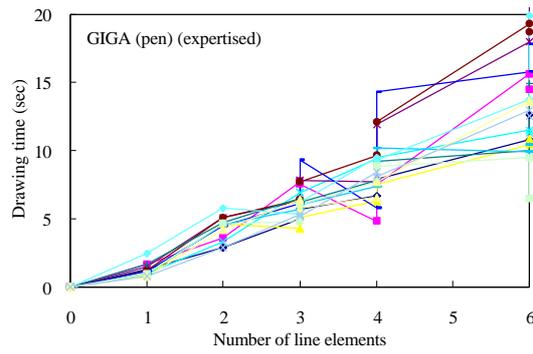


Figure 7.12 Relation in expertised GIGA (Exp. B)

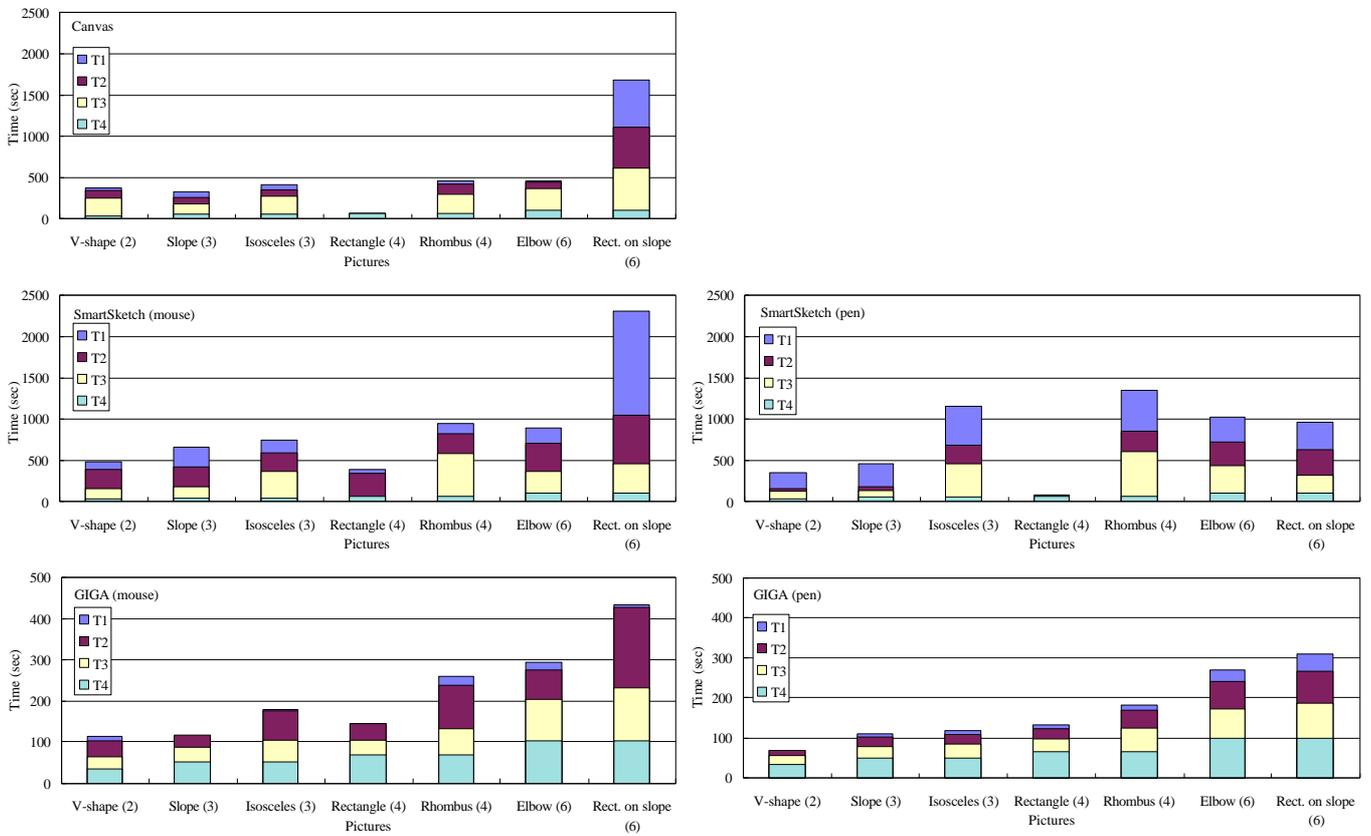


Figure 7.13 Four-layer analysis of each picture

7.6 The Case of Drawing Complex Pictures

The times for drawing complex pictures “human” and “node-link” shown in Figure 3.9 were also examined with GIGA. In the experiment, no subject gave up his drawing, unlike the case of SmartSketch. Figure 7.14 shows the relationship between the time for actual drawing operation and the time without wasted operation. Accumulated time of 13 subjects are shown in Figure 7.15.

Comparing with the result with Canvas shown in Figure 3.10, following tendencies are observed in GIGA:

- Total time for drawing is reduced to 40% for the human picture. Even for the node-link picture which Canvas is suited for, GIGA needs 60% time.
- The wasted time is smaller.
- The difference of the total operation time between the most slowly subject and the most speedy one is smaller.

These show that the advantage of GIGA is preserved even for drawing complex pictures.

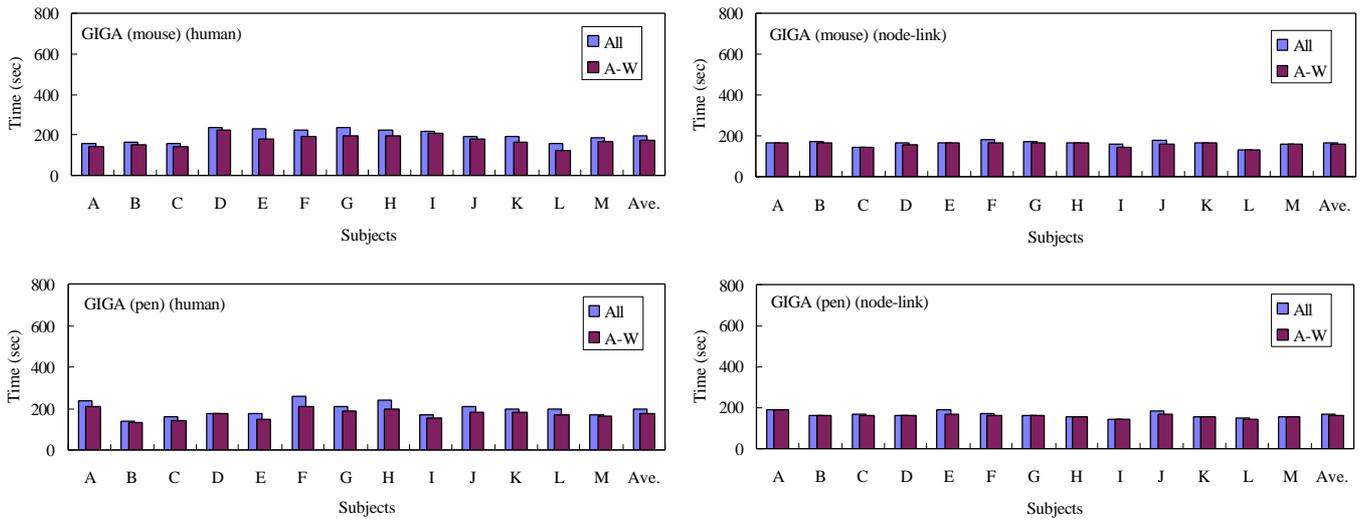


Figure 7.14 Time of drawing complex pictures with GIGA

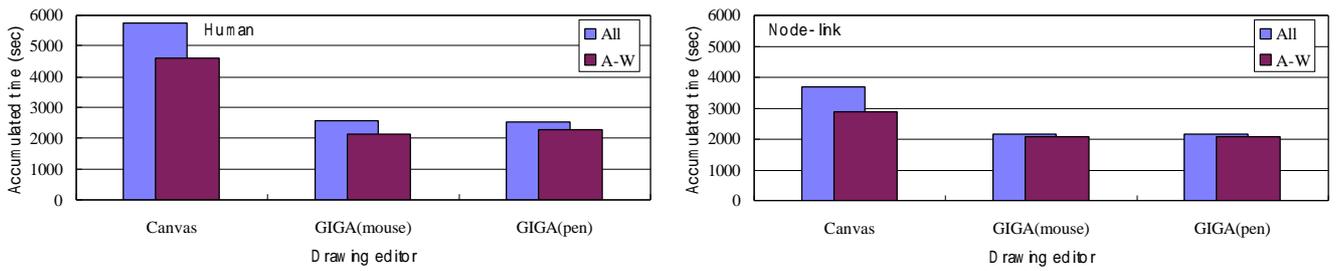


Figure 7.15 Comparison of operation times of drawing complex pictures

7.7 Effect of the Chunk Size

Finally, the problems unique to GIGA are examined. As described in Section 6.4, the following two problems can occur in the interface of the prototype system:

- The candidate picture desired by the user cannot be displayed, and the user must re-sketch the picture many times.
- The chunk size, which is a single free stroke in GIGA, is too small, and candidate selection happens too frequently.

The results of analysis are shown in Table 7.1. This table shows the ratio of which candidate was selected in all drawing operations of the simple pictures in Exp. A. The primary candidate was selected with very high ratio (94.8%), and the ratio of re-sketching was only 2.4% of total drawing operations. Even for re-sketching, only one re-sketching was enough for getting the desired picture. It was confirmed that these numbers do not become so worse even for more complex pictures [84].

These results show that the re-sketching interface does not cause serious problems. If desired candidate is not displayed although a user sketches carefully, there is a problem in the user's drawing strategy in many cases. Therefore, the user can find his/her strategy error in early time, and the wasteful trial-and-error time can be reduced.

As for the chunk size, current size corresponds to a primitive object of the

Table 7.1 Ratio of selected candidates in GIGA

	Primary candidate	Other candidates	One re-sketching	More re-sketching
Mouse	95.5%	1.7%	2.8%	0.0%
Pen	94.1%	3.8%	2.1%	0.0%
Average	94.8%	2.8%	2.4%	0.0%

conventional drawing editor, and the subjects in the experiment did not complain that candidate selection was required too often. Furthermore, in the prototype system the primary candidate is selected automatically when the next sketching is started. Therefore, the user can continue the drawing smoothly. Note that this automatic selection function becomes possible because the rate of selecting the primary candidate is high.

7.8 Summary of Analyses

From the above analyses, it can be concluded that the candidate-selection method in which user handles mainly concrete pictures reduces the psychological operation load of the user. From the above analyses, it can be concluded that the candidate-selection method in which user handles mainly concrete pictures reduces the psychological operation load of the user. The result is lead from the difference between the operation models described in Chapters 4 and 6.

GIGA's unique user interface was also evaluated. The primary candidate was selected with very high ratio (94.8%), and the ratio of re-sketching was only 2.4% of total drawing operations. As a combination of these features, drawing time with GIGA (pen) was reduced to 42% of Canvas and to 20% of SmartSketch (pen).

8 Related Work: Survey of Computer-Assisted Drawing Systems

This chapter thoroughly investigates related special studies in order to verify the novelty of claims presented in this thesis. Studies related to drawing methods are described and discussed by classifying into the following four categories: painting tools, object-oriented drawing tools, sketch-beautification systems, and recognition systems for drawing on paper. Finally, studies of the input efficiency of pens are summarized, which were highly related to implementing GIGA.

8.1 Painting Tools

The painting tool is a system in which the locus of the mouse pointer on the display itself becomes a picture, and the drawn diagram is preserved as a bitmap instead of its logical structure. In order to refine the drawn diagram, editing is performed not on each logical part of the diagram, but on each region. Operations such as copying, moving, or rotating each logical diagram cannot be carried out, but can instead be carried out on each region of pixels. Therefore, the paint system is suitable for drawing and correcting an unstructured picture. Furthermore, since this drawing tool uses the movement of the mouse pointer itself to form a picture as in drawing on paper with a pen or a brush, and does not have complicated underlying structure, this drawing tool is suitable to a beginner.

This type of drawing systems are very popular for drawing work using a personal computer. PaintBrush on Windows and PhotoShop for Windows and Macintosh are well known commercial products. One recent research topic of such paint system is how

closely a computer can simulate actual paint brushes. Since about 1980s, there have been several researches on shaded-off paint and brush-touch [183, 164, 156, 15, 181]. Haeberli coped with abstract image representation of the painting system in his paint-by-numbers system [53]. These kind of studies were paid much attention even by the top level academic society like SIGGRAPH held in 1994, in which one session was dedicated for the painting system. In that session, results of five studies on advanced painting systems were reported [11, 62, 108, 154, 185]. In the same conference, non-photorealistic technology, by which outlines can be drawn with multiple strokes like a “dessin,” was also reported [185]. StudioPaint was proposed as a painting tool for GUI building [99].

8.2 Object-Oriented Drawing Tools

The object-oriented drawing tool is a system in which a user first selects a shape from a menu when drawing a picture, and then operates the mouse to designate the control points of the shape. A drawn picture is stored as a logical structure of diagrams (objects), rather than an unstructured bitmap data. In this type of drawing tool, therefore, it is possible to copy, move, expand, shrink, and rotate a group of logical diagrams.

8.2.1 Commercial Drawing Editors

There are many commercial object-oriented drawing editors. Typical example is MacDraw for Macintosh. Idraw, tgif, and xfig are available for Unix systems. Object-oriented drawing functions are incorporated in many applications such as CorelDraw, Microsoft Word and PowerPoint on Windows. Hanako and Tsuru (crane) are available for MS-DOS, Windows, etc. Canvas, which is used in the experiments in this thesis, works both on Windows and Macintosh.

8.2.2 Object-Oriented CAD Systems

In this subsection, studies of object-oriented CAD systems considering how to apply geometrical constraints are investigated and classified as follows. Note that sketch-based CAD systems have already been described in Section 4.3.

From the early stage of computer graphics, geometrical constraints have been used in Sketchpad [168], ThingLab [17], and Juno [127]. Pavlidis et al. have investigated a mechanism to infer the geometrical constraints during editing diagrams, and developed “automatic beautifier” [143]. Bier's “snap-dragging” [14] incorporated a ruler compass metaphor and created new type of grid. In “Grace” [2], flexible inference was implemented.

There are many studies to incorporate the constraints into drawing systems. Unidraw [179] and CoDraw [49] are constraint-based drawing editors. Many systems are also proposed as constraint-based CAD [176, 187, 18, 155, 106, 1, 30, 59, 39, 92, 162, 188]. Garnet [119] is a constraint-based interface builder. In such constraint-based systems, constraint solver [120, 19, 21, 38] plays an important role.

Intelligent CAD [169, 170] already has a long history in the area of artificial intelligence in design [144]. In the Sketchpad system, Sutherland proposed to use graphics as a medium of communication and illustrates the “constraint-oriented” approach from the first rough sketch to its final completed drawing [168].

Constraint processing as a knowledge representation paradigm as well as an inference mechanism is used in a broad spectrum of applications. Recent result in this area is, for instance, an automatic configuration of user interfaces [172]. Klein also pointed out interesting issues in the area of configuration problem solving [88]. Gross et al. proposed the use of a “constraint manager” system in wider area of designing work [50]. Shimada et al. studied a constraint-based framework for intelligent CAD systems. Several sort of constraints were listed by Liu et al. [109]. In the constraint-based figure maker by Kalra et al., the system maintains the relations among objects [72]. But, Nourani et al. proposed that the user should check such global consistency [132]. Philips et al. discussed the problem of both under- and over-specification of constraints [145]. Hel-Or et al.

proposed the use of inexact satisfaction of constraints as a mean to express a general outline to avoid over-specification [57]. Veltkamp showed that an incremental technique (instead of using a constraint logic programming system) is better for interactive design purposes and dealt with under-constrained situations [177]. Olsen et al. also focused on interactive techniques [137]. Higher level interface when indicating the constraints among geometrical objects were proposed by Rappoport [148].

8.2.3 Programming by Example

PBE (Programming by Example) is a technique to forecast future operations on the basis of an operation history. This technique has already been widely used in drawing editors. Metamouse [115], Mondrian [105], and History-based macro [94] are drawing tools in which the theory of the PBE is applied.

8.2.4 Education of Drawing Tools

Drawing tools can be utilized to educate geometrical constraints to users. Several researches reported to use the system for learning geometry [135, 136, 58]. Nakayama described a CAI system of mathematics containing the geometry [125]. Oosterholt et al. proposed a drawing tool for children [139]. KidPad is a drawing system for children, which has special zooming function [32].

Recently, the education system of CAD itself was announced [12, 13]. The approach of educating the strategy in drawing with CAD has an effect to improve the drawing efficiency of educated objects. However, this approach is not useful for drawing a new picture which was not educated. In order to achieve a true education effect, the education would require much time.

8.3 Sketch-Beautification Systems for an Electronic Panel

There are two methods for drawing a sketch on an electronic panel and making the computer to recognize and beautify the input sketch; one is to recognize and beautify a

complete sketch in an aggregated manner as a batch processing, and the other is to successively recognize and beautify primitives of a sketch one by one each time a primitive is drawn up. Both methods of recognizing and beautifying a sketch are available for 2-dimensional drawing and 3-dimensional data inputting. Such systems are also found in CAD area.

8.3.1 2-Dimensional Sketch Systems

There are many 2-dimensional sketch systems such as sketch book [51, 52] and reference by picture for CAD [167, 111, 193, 194, 27]. Kazama et al. adopts the stationery metaphor for drawing [85]. Several systems have been proposed as online hand-written text editing systems [90, 100].

Since sketch operation has more humanity aspects, “perceptual constraints” play more important role than the geometrical constraints. Human-organized structure [63] and perceptually supported 2D sketch editors [157, 117] consider the human perceptual constraints.

8.3.2 3-Dimensional Sketch Systems

On the basis of several important works [149, 152], it has become possible to input 3D object through sketching. Perspective image [91] and solid model [40, 41, 60] can be also inputted by sketching. Several studies coped with 3D modeling [70, 175], and other studies proposed to incorporate the constraint-based modeling into engineering [191, 122]. There are several systems to enable interactive 3-dimensional input by a pen device [28, 192]. 3D graphics can be applied to non-photorealistic rendering like sketch [158].

8.4 Recognition Systems for Drawing on a Paper

Many studies in this field have been done during 1970s and 1980s, but new studies continue even to the present. These studies can be basically categorized into two types on the basis of what kind of diagram or text are scanned, recognized, and beautified.

Several researches conclude that paper is better than screen in the operational efficiency such as reading speed [45, 112, 159, 134].

8.4.1 Recognition of Formatted Diagram or Text

Many researches have been done on a recognition of formatted engineering diagram or typed text [20, 10, 101, 31]. In addition, lots of OCR systems can be found in the market. In these systems, pictures such as a mechanical drawing or typed text which was previously once prepared and printed by the computer are used as a reference for recognizing and reshaping. Text-graphics separation [36, 113] and text-text separation [124] are one of the basic technologies used in this type of recognition systems. PaperLink was proposed to link paper document and electronic world [6].

8.4.2 Recognition of Hand-Written Sketch or Text

The other type is a recognition of hand-written sketch or text. That is a system to use a computer to recognize and beautify a hand-written diagram. There are recognition systems of logical circuit diagrams written by freehand [22, 123]. There also exist hand-written memo system [182, 184], retrieving system of picture database by hand-written drawing [95], text recognition systems [146], and hand-written design system of cloths image [77, 78]. The interface aspects of drawing on paper are discussed in several papers [71, 147].

Revision of formatted engineering diagram and typed text by hand-written diagrams or texts is also included in this category. There are several examples of such recognition systems as is mentioned in Section 4.3 [107, 166].

8.4.3 Recognition Techniques

Basic algorithms of pattern recognition can be applied both of above two categories. There are many studies on such general and basic recognition algorithms [171, 142]. Hough-transformation [7] is also the basic algorithm to extract shapes. Ronse surveyed various techniques of the pattern recognition [150]. There also exist many studies on line

drawing [126, 73, 121] and curve drawing [180, 35, 87]. Saga et al. developed a fuzzy spline curve for a freehand drawing interface [153].

Since it can be applied to pen input operations, researches on character recognition is active even now. However, since this thesis does not deeply focus on character recognition, only typical examples are discussed here. Tappert has written a good survey paper of Chinese character recognition [174].

8.5 Efficiency of Electronic Pen Input

There are many active studies on input devices and their menu systems. This kind of studies were accelerated by the advent of mouse in the late 1980s and early 1990s. Studies of such input devices have been paid more attention as GUI becomes popular in commercial computer systems. Further, from the start of 1990s, studies of pen devices increased corresponding to the penetration of personal digital assistants (PDAs) and personal handy phone systems in the market. Studies that experimentally clarify the predominance of electronic pen are as follows. In the early stage of free-hand input, tablet [23] was used. In 1990s, pen device became to be widely used. Mackinlay et al. compared these two input devices by the experimental analysis [114].

Basic technology for pen such as pen gesture [151] and multiple-stroke [5] input are researched. T-cube [178] uses the orbital path of pen strokes. Pen gesture also plays an important role in multimodal input [141]. For efficient mobile computing pen input is indispensable [48]. Frankish et al. investigated the relation between the user acceptance of recognition and the accuracy in pen computing [37]. There also exist a number of studies on the experimentation of menu shapes that are suitable for the electronic pen [4, 173, 96, 97, 98, 54]. A mark-base interaction paradigm was proposed by Baudel [8].

There are many drawing systems other than those introduced in Section 4.3 that use electronic pen predominantly. Among them are Artist's Studio by Sengupta et al. [160], BrightBoard system by Stafford-Fraser et al. [163], marking on map with pen by Oviatt

[140], window system HITSUJI by Hayakawa et al. [56], and sketch-based user-interface builder by Landay et al. [102].

8.6 Conclusion of the Investigation of the Related Studies

Major journals and proceedings related to the computer are thoroughly investigated. As a result, it has been concluded that there do not exist approaches to computer-assisted drawing aimed at the same way of this thesis: sketch-annotation method, dot-masked revision sheet method, and candidate-selection method which tries to reduce the user's handling of the abstract-picture layer.

9 Conclusions

This chapter summarizes the results and contributions of this thesis and discusses the direction of future research on improving drawing editor.

9.1 Results

This thesis first analyzed the drawing processes of “Canvas,” which is a widely used object-oriented drawing editor, and clarified the psychological problems that have not been addressed by past researches. Two psychological problems were found, as described below, inherent in the process of deciding how to convert a picture to be drawn into a sequence of drawing editor commands (functions), which process is named the “command-planning.”

- Problem of wasted operations (trial-and-error, etc.) caused by incorrect planning
- Problem of needing much time for the command-planning itself

The experiments and analyses revealed that the time wasted by incorrect planning occupied 10 to 20% of the total operation time, and the time for planning accounted for 25 to 30%.

A similar overhead was found also in the “SmartSketch,” which is a sketch-beautification type drawing editor. This is because this type of drawing editor still uses the same functions as the object-oriented drawing editor when editing beautified pictures and command-planning is needed for the editing. However, the planning load is not the only reason why the sketch-beautification type drawing editor has not become popular. Other reasons were found through several experiments. In the sketch beautification process, the success ratio in drawing a straight line with the mouse was 35%, the success rate in selecting an object with the electronic pen was 63%, and the rate in using a special

drawing method for drawing a vertex point, in which intersecting lines are drawn intentionally then unnecessary portions are cut, was 45%. The problem is that the sketch-beautification type drawing editor ignores the perceptual constraints such that the user tends to use many straight lines and often connects corners to each other when drawing pictures. The sketch-beautification method does not take into account those perceptual constraints and results in wasted operations. Thus, the sketch-beautification type drawing editor suffered recognition problem of perceptual constraints and same psychological load as in the editing process of the object-oriented drawing editor. The sketch-beautification type was shown to be less convenient than the object-oriented type drawing editor.

Those results of Canvas and SmartSketch are regarded as general tendency of object-oriented and sketch-beautification editors respectively, because the basic figures were drawn by using only common functions of these editors in the experiments.

On the basis of the above findings, this thesis proposed a drawing operation model. This model has two functional layers: one layer handles concrete (actual) pictures and the other layer handles abstract pictures such as representations of geometrical constraints by combinations of editing tools and functions. This model does not conflict with the famous Norman's seven-stage model in the field of cognitive science. Using the proposed operation model, various previous researches conducted on drawing editors were analyzed and explained. The analysis concluded that the layer for handling concrete pictures of the drawing operation model is important in order to reduce the time required for the command-planning and incorrect planning. This conclusion is a requirement for ideal drawing editors.

Next, this thesis described three approaches to the ideal drawing editor that meets the above requirement. The first approach was the “sketch-annotation method.” This method improved the conventional sketch-beautification type drawing editor which was less convenient than the object-oriented drawing editor by solving the input problem. This improvement made the sketch-input type drawing editor as convenient as the object-oriented drawing editor. However, this method still adopted the conventional editing

scheme and could not solve the problem in the command-planning found in the drawing operation model. The second approach was the “dot-masked revision sheet method” to beautify the pictures manually drawn on paper, which are typical concrete pictures. This method has a merit that input can be done in the concrete-picture layer, using paper and pencil. However, this method suffered from problems of long time taken to input a picture by a scanner, which has an adverse psychological influence on the user. The third approach was the “candidate-selection method.” In this method, the computer displays multiple pictures that can be created by a combination of general functions of existing drawing editors as candidates based on a picture manually sketched by the user. The user then simply selects one of the displayed pictures. The drawing editor using this method is a completely new type of drawing editor because it reduces the user's participation in the operations in the abstract layer. A prototype system was implemented and evaluated for each of the above three approaches. The conclusion was that the third approach was the most suitable approach for an ideal drawing editor.

The second half of this thesis gave a detailed explanation of the candidate-selection method, which is the major contribution of this thesis, and verified the effectiveness of the method. The results of experiments using a prototype system “GIGA” showed that psychological overhead time is reduced to less than half of conventional drawing editors. The difference mainly came from the proposed drawing model which reduces the user's operation in the abstract-picture layer.

In addition to the evaluation of the overhead time with GIGA, analyses have been done to confirm that selecting from candidates is not troublesome and the size of the candidate display (chunk size) is not too small. On the prototype system, the rate that intended candidate was not displayed by the first sketch operation was less than 3%, and, even in such failure cases, the intended candidate was displayed with at most two re-sketching operations. The re-sketching interface would thus not cause serious problems. Regarding the problem of chunk size (candidates are displayed each time a single free stroke is drawn), the interface functions smoothly if the first candidate was selected automatically because the rate of selecting the primary candidate was as high as 95%.

As a combination of these good features, total drawing time with GIGA is reduced to less than half of conventional drawing editors.

9.2 Directions of Future Research

One possible future work is to extract and evaluate the psychological time more exactly. This can be achieved by changing the unit of analysis to finer size. For example, a unit of operation “draw a line” can be divided into four finer parts such as:

1. Move a pointer from a menu button to a start point of a line
2. Decide a start point and push the mouse button (align)
3. Move the pointer to the end point of a line
4. Decide a end point and release the mouse button (align)

Another improvement of the analysis is to divide the feedback (evaluation) and command-planning more precisely. This can be achieved by utilizing the user's speech information during the experiments.

The experiments reported in this thesis showed that GIGA requires less time to explain the usage to the user. Comparison and analysis of such learning processes will reveal new areas of study that are different from the problems indicated in this thesis. GIGA can also work with an electronic white-board. The electronic white board is generally used for conferencing and CAI systems. If the operations of GIGA using an electronic white-board and the psychological load caused by the operations are analyzed, specific problems different from those in the operation of desktop computers may be found. In short, analyses of learning processes and operation with other types of devices will reveal new subjects of future research.

The candidate-selection method proposed in this thesis can become a broader research area of drawing systems. One topic in this area is a research on larger chunk sizes. Increasing the chunk size (one stroke on GIGA) would be a new field of research. This is because, considering that this research proposed a new front-end processor (FEP) for drawing editors, increasing the chunk size corresponds to a multiple phrase/clause

conversion in Kana-to-Kanji conversion method used for Japanese text editors. There must be many issues to be studied such as those concerning the FEP for text editors. Future research may also address improvement in the recognition of perceptual constraints.

More complicated figures should be considered. In such a case, it will be necessary and important to widen chunk size. The new chunk size must be carefully decided not to increase the psychological load on the basis of the experiments using complicated figures.

This thesis focused on the psychological load problems inherent in conventional drawing editors and regarded trial-and-error as wasted operations. However, some existing researches focus on the new usage of drawing editor, that is, so-called “usage as an idea processor,” in which trial-and-error operations are utilized to improve picture quality and allow the user to rearrange ideas and processes of thinking. The combination of such research with the analysis of psychological load described in this thesis would also be a new field of research.

10 Bibliography

1. B. Aldefeld. Variation of geometries based on a geometric-reasoning method. *Computer-Aided Design*, 20(3):117–126, April 1988.
2. S. R. Alpert. Graceful interaction with graphical constraints. *IEEE Computer Graphics and Applications*, 13(2):82–91, 1993.
3. Apple Computer, Inc. *Apple Human Interface Guidelines: The Apple Desktop Interface*. Addison-Wesley, 1992.
4. Ajay Apte and Takayuki Dan Kimura. A comparison study of the pen and the mouse in editing graphic diagrams. In *Proceedings of the 1993 IEEE Workshop on Visual Languages*, pages 352–357, 1993.
5. Ajay Apte, Van Vo, and Takayuki Dan Kimura. Recognizing multistroke geometric shapes: an experimental evaluation. In *Proceedings of the 1993 ACM Symposium on User Interface Software and Technology*, pages 121–128, 1993.
6. Toshifumi Arai, Dietmar Aust, and Scott E. Hudson. PaperLink: A Technique for Hyperlinking from Real Paper to Electronic Content. In *Proceedings of ACM CHI'97 conference on Human Factors in Computing Systems*, pages 327–334, 1997.
7. M. Atiquzzaman. Multiresolution Hough transform — an efficient method of detecting patterns in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1090–1095, November 1992.
8. Thomas Baudel. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of the 1994 ACM Symposium on User Interface Software and Technology*, pages 185–193, 1994.
9. Malcolm I. Bauer and Bonnie E. John. Modeling time-constrained learning in a highly interactive task. In *Proceedings of ACM CHI'95 conference on Human*

Factors in Computing Systems, pages 19–26, 1995.

10. Robert Bergevin and Martin D. Levine. Generic object recognition: building and matching coarse descriptions from line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):19–36, January 1993.
11. Deborah F. Berman, Jason T. Bartell, and David H. Salesin. Multiresolution painting and compositing. *ACM Computer Graphics, Annual Conference Series*, pages 85–90, July 1994.
12. Suresh K. Bhavnani and Bonnie E. John. Exploring the unrealized potential of computer-aided drafting. In *Proceedings of ACM CHI'96 conference on Human Factors in Computing Systems*, pages 332–339, 1996.
13. Suresh K. Bhavnani and Bonnie E. John. From Sufficient to Efficient Usage: An Analysis of Strategic Knowledge. In *Proceedings of ACM CHI'97 conference on Human Factors in Computing Systems*, pages 91–98, 1997.
14. Eric Allen Bier and Maureen C. Stone. Snap-dragging. *ACM Computer Graphics*, 20(4):233–240, August 1986.
15. Teresa W. Bleser, John L. Sebert, and J. Patrick McGee. Charcoal sketching: returning control to the artist. *ACM Transactions on Graphics*, 7(1):76–81, January 1988.
16. D. Bolz. Some aspects of the user interface of a knowledge based beautifier for drawings. In *Proceedings of 1993 International Workshop on Intelligent User Interfaces*. ACM Press, 1993.
17. Alan Borning. The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4):353–387, 1981.
18. Alan Borning and Robert Duisberg. Constraint-based tools for building user interfaces. *ACM Transactions on Graphics*, 5(4):345–374, October 1986.
19. W. Bouma, I. Fudos, C. Hoffman, J. Cai, and R. Paige. Geometric constraint solver. *Computer-Aided Design*, 27(6):487–501, June 1995.
20. Christopher M. Brown. PADL-2: a technical summary. *IEEE Computer Graphics*

and Applications, pages 69–84, 1982.

21. A. Alasdair Buchanan and Alan de Pennington. Constraint definition system: a computer-algebra based approach to solving geometric-constraint problems. *Computer-Aided Design*, 25(12):741–750, December 1993.
22. Horst Bunke. Attributed programmed graph grammars and their application to schematic diagram interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(6):574–582, November 1982.
23. William Buxton, Ralph Hill, and Peter Rowley. Issues and techniques in touch-sensitive tablet input. *ACM Computer Graphics*, 19(3):215–224, 1985.
24. S. K. Card and T. P. Moran. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, 1980.
25. Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Hillsdale, Lawrence Erlbaum Associates, 1983.
26. Stuart K. Card, Peter Pirolli, and Jock D. Mackinlay. The cost-of-knowledge characteristic function: display evaluation for direct-walk dynamic information visualizations. In *Proceedings of ACM CHI'94 conference on Human Factors in Computing Systems*, pages 238–244, 1994.
27. C. L. P. Chen and S. Xie. Freehand drawing system using a fuzzy logic concept. *Computer-Aided Design*, 28(2):77–89, January 1996.
28. Michael F. Deering. Holosketch: the VR sketching system. *Communications of the ACM*, 39(5):54–61, 1996.
29. Deneba Software Home Page. <http://www.deneba.com/>
30. D. Dori and K. Tombre. From engineering drawings to 3D CAD models: Are we ready now? *Computer-Aided Design*, 27(4):243–254, April 1995.
31. Dov Dori. Vector-based arc segmentation in the machine drawing understanding system environment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1057–1068, November 1995.
32. Allison Druin, Jason Stewart, David Proft, Ben Bederson, and Jim Hollan. KidPad: A Design Collaboration Between Children, Technologists, and Educators. In

- Proceedings of ACM CHI'97 conference on Human Factors in Computing Systems*, pages 463–470, 1997.
33. Dennis E. Egan, Joel R. Remde, Louis M. Gomez, Thomas K. Landauer, Jennifer Eberhardt, and Carol C. Lochbaum. Formative design-evaluation of superbok. *ACM Transactions on Graphics*, 7(1):30–57, January 1989.
 34. K. A. Ericson and H. A. Simon. *Protocol Analysis: Verbal Reports as a Data*. MIT Press, 1986.
 35. Martin A. Fischler and Helen C. Wolf. Locating perceptually salient points on planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):113–129, February 1994.
 36. Lloyd Alan Fletcher and Rangachar Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):910–918, November 1988.
 37. Clive Frankish, Richasrd Hull, and Pam Morgan. Recognition accuracy and user acceptance of pen interfaces. In *Proceedings of ACM CHI'95 conference on Human Factors in Computing Systems*, pages 503–510, 1995.
 38. Bjorn N. Freeman-Benson, John Maloney, and Alan Borning. An incremental constraint solver. *Communications of the ACM*, 33(1):54–63, January 1990.
 39. Ioannis Fudos and Chiristph M. Hofmann. Constraint-based parametric conics for CAD. *Computer-Aided Design*, 28(2):91–100, February 1996.
 40. Yukio Fukui, Toyotoshi Hirotsani, Tomohiro Ohira, and Yoshiki Kishi. An input method for solid models by sketches on projection planes. *Journal of the Information Processing Society of Japan*, 26(6):1113–1120, November 1985.
 41. Y. Fukui. Input method of boundary solid by sketching. *Computer-Aided Design*, 20(8):434–441, October 1988.
 42. FutureWave Software. SmartSketch Home Page.
<http://www.futurewave.com.au/smarts sketch.htm>
 43. Richard Gong and Jay Elkerton. Designing minimal documentation using a GOMS model: a usability evaluation of an engineering approach. In *Proceedings of ACM*

- CHI'90 conference on Human Factors in Computing Systems*, pages 99–106, 1990.
44. Richard Gong and David Kieras. A validation of the GOMS model methodology in the development of a specialized, commercial software application. In *Proceedings of ACM CHI'94 conference on Human Factors in Computing Systems*, pages 351–357, 1994.
 45. J. J. Gould, L. Alfaro, V. Barnes, R. Finn, N. Grischowsky, A. Minuto, and J. Salaun. Reading is slower from CRT displays than from paper: Attempts to isolate a single variable explanation. *Human Factors*, 29:269–299, 1987.
 46. Wayne D. Gray, Bonnie E. John, and Michael E. Atwood. The precis of project Ernestine or an overview of a validation of GOMS. In *Proceedings of ACM CHI'92 conference on Human Factors in Computing Systems*, pages 307–312, 1992.
 47. Wayne D. Gray, Bonnie E. John, and Michael E. Atwood. Project Ernestine: validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction*, 8:237–309, 1993.
 48. Sally Grisedale, Mike Graves, and Alexander Grünsteidl. Designing a Graphical User Interface for Healthcare Workers in Rural India. In *Proceedings of ACM CHI'97 conference on Human Factors in Computing Systems*, pages 471–478, 1997.
 49. M. D. Gross. Graphical constraints in CoDraw. In *Proceedings of 1992 IEEE Workshop on Visual Languages*, pages 81–87, 1992.
 50. Mark D. Gross, Stephen M. Ervin, James A. Anderson, and Aaron Fleisher. Constraints: knowledge representation in design. *Design Studies*, 9(3):133–143, July 1988.
 51. Mark D. Gross. Stretch-a-sketch: a dynamic diagrammer. In *Proceedings of the 1994 IEEE Workshop on Visual Languages*, pages 232–238, 1994.
 52. Mark D. Gross and Ellen Yi-Luen Do. Demonstrating the electronic cocktail napkin: a paper-like interface for early design. In *Proceedings of ACM CHI'96 conference on Human Factors in Computing Systems*, pages 5–6, 1996.

53. P. Haeberli. Paint by numbers: abstract image representations. *ACM Computer Graphics*, 24(4):207–214, August 1992.
54. Gary Hardock, Gordon Kurtenbach, and William Buxton. A marking based interface for collaborative writing. In *Proceedings of the 1993 ACM Symposium on User Interface Software and Technology*, pages 259–266, 1993.
55. Peter Haunold and Werner Kuhn. A keystroke level analysis of a graphics application: manual map digitizing. In *Proceedings of ACM CHI'94 conference on Human Factors in Computing Systems*, pages 337–343, 1994.
56. Eiichi Hayakawa, Tsunehisa Kawamata, Yasushi Miyajima, Naoki Kato, Mitarou Namiki, and Nobumasa Takahashi. Design and implementation of 'HITSUJI' window system for pen interface research and development. *Journal of the Information Processing Society of Japan*, 36(4):932–943, April 1995.
57. Yaacov Hel-Or, Ari Rappoport, and Micheal Werman. Relaxed parametric design with probabilistic constraints. *Computer-Aided Design*, 26(6):426–434, June 1994.
58. Kazuyoshi Hidaka. System for learning geometry based on the manipulation of constrained geometric figures. *Journal of the Institute of Electronics, Information and Communication Engineers*, J76-A(11):1612–1619, November 1993.
59. Yasuo Hidaka, Hidetoshi Ando, Hiromasa Suzuki, and Fumihiko Kimura. Construction and manipulation of a geometric model with geometric constraints. *Journal of the Information Processing Society of Japan*, 31(11):1677–1687, November 1990.
60. Tsutomu Horikoshi, Satoshi Suzuki, and Kazunari Nakane. 3D modeling using rough sketches and 3D shape retrieval system. *Journal of the Information Processing Society of Japan*, 35(9):1750–1758, September 1994.
61. T. L. J. Howard. Evaluating PHIGS for CAD and general graphics applications. *Computer-Aided Design*, 23(4):244–251, May 1991.
62. Siu Chi Hsu and Irene H. H. Lee. Drawing and animation using skeletal strokes. *ACM Computer Graphics, Annual Conference Series*, pages 109–122, July 1994.
63. Takeo Igarashi, Satoshi Matsuoka, and Toshiyuki Masui. Adaptive recognition of

- human-organized implicit structures. In *Proceedings of the 1995 IEEE Workshop on Visual Languages*, pages 258–266, 1995.
64. Takeo Igarashi, Sachiko Kawachiya, Satoshi Matsuoka, and Hidehiko Tanaka. In Search for an Ideal Computer-Assisted Drawing System. In *Proceedings of the 6th IFIP Conference on Human-Computer Interaction (INTERACT '97)*, pages 104–111, 1997.
 65. Takeo Igarashi, Sachiko Kawachiya, Satoshi Matsuoka, and Hidehiko Tanaka. Interactive Beautification: A Technique for Rapid Geometric Design. In *Proceedings of the 1997 ACM Symposium on User Interface Software and Technology*, pages 105–114, 1997.
 66. Sharon Irving, Peter Polson, and J. E. Irving. A GOMS analysis of the advanced automated cockpit. In *Proceedings of ACM CHI'94 conference on Human Factors in Computing Systems*, pages 344–350, 1994.
 67. N. Jacob. *Usability Engineering*. Academic Press, 1993.
 68. Bonnie E. John and Allen Newell. Cumulating the science of HCI: from S-R compatibility to transcription typing. In *Proceedings of ACM CHI'89 conference on Human Factors in Computing Systems*, pages 109–114, 1989.
 69. Bonnie E. John. Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In *Proceedings of ACM CHI'90 conference on Human Factors in Computing Systems*, pages 107–115, 1990.
 70. Imothy E. Johnson. Sketchpad III, a computer program for drawing in three dimensions. In *Spring Joint Computer Conference*, 1963.
 71. W. Johnson, H. Jellinek, R. Rao, and S. Card. Bridgin the paper and electronic worlds: the paper user interface. In *Proceedings of ACM InterCHI'93*, pages 507–512, 1993.
 72. Devendra Kalra and Alan H. Barr. A constraint-based figure-maker. In *the European Computer Graphics Conference and Exhibition*, pages 413–424, 1990.
 73. Rangachar Kasturi, Sing T. Bow, Wassim El-Masri, Jayesh Shah, James R. Gattiker,

- and Umesh B. Mokate. A system for interpretation of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):978–991, October 1990.
74. O. Kato, H. Iwase, M. Yoshida, and J. Tanahashi. Interactive hand-drawn diagram input system. In *Proceedings of PRIP 82. IEEE Computer Society Conference on Pattern Recognition and Image Processing*, pages 544–549, 1982.
 75. Sachiko Kawachiya and Haruo Takeda. Image data revising method using pattern extraction. In *Proceedings of IEICE SIGPRU JAPAN*, pages 17–24, 1990.
 76. Sachiko Kawachiya. Image data editing method using dotted paper. *Journal of the Institute of Electronics, Information and Communication Engineers*, J75-D-II(4):728–736, April 1992.
 77. Sachiko Kawachiya. Recognition of clothes from shading images using neural networks. In *Proceedings of IEICE SIGPRU JAPAN*, pages 71–78, 1992.
 78. Sachiko Kawachiya and Tsutomu Tabe. Feature extraction of hand-written images by fuzzy theory. In *Proceedings of Annual Convention of IEICE JAPAN*, number 7, page 295, 1992.
 79. Sachiko Kawachiya and Akinori Yonezawa. An integrated drawing and beautifying system which has freedom of selecting input/output devices. In *Proceedings of Information Processing Society of Japan SIGHI*, pages 7–12, October 1995.
 80. Sachiko Kawachiya. Guide-map drawing system using GIS. In *Proceedings of Computer Society of India '95 (CSI'95)*, pages 205–214. Tata McGraw-Hill, November 1995.
 81. Sachiko Kawachiya. Hand-written picture language for effective pattern recognition. In *Proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation (ISSIAI'96)*, pages 60–65, April 1996.
 82. Sachiko Kawachiya, Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Reduction of cognitive workload on drawing editor. In *Proceedings of the 1996 Workshop on Interactive System and Software*, pages 71–80, 1996.
 83. Sachiko Kawachiya, Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka.

- GIGA: A Pen-Based Constraint Drawing System. In *Proceedings of 6th Australian Conference on Computer-Human Interaction (OZCHI '96)*, pages 314–315, 1996.
84. Sachiko Kawachiya, Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. An Interactive Drawing Editor with Low Cognitive Overload. *JSSST Computer Software*, to appear, 1998.
 85. Shinya Kazama, Naoki Kato, and Masaki Nakagawa. A hand-drawing system with 'stationery metaphors'. *Journal of the Information Processing Society of Japan*, 35(7):1457–1468, July 1994.
 86. Yasuyo Kikuta. The quantitative evaluation of ease-of-use. In *Proceedings of IPS of Japan SIGSW*, pages 1–6, 1984.
 87. Myung-Soo Kim, Eun-Joo Park, and Soon-Bum Lim. Approximation of variable-radius offset curves and its application to bezier brush-stroke design. *Computer-Aided Design*, 25(11):684–698, 1993.
 88. Rudiger Klein. How to use constraints — exemplified in configuration problem solving. In *Third International Conference on Artificial Intelligence in Design*, pages 14–19, 1994.
 89. Haruhiko Kojima and Tohru Toida. Online Handsketched Line Figure Recognition by Adjacent Strokes Structure Analysis Method. *Journal of the Information Processing Society of Japan*, 28(8):863–869, August 1987.
 90. Haruhiko Kojima and Tohru Toida. Online text and line figure editing system using hand-written editing marks. *Journal of the Information Processing Society of Japan*, 29(3):242–248, March 1988.
 91. Kunio Kondo, Fumihiko Kimura, and Taro Tajima. Estimation of a point of view with perspective drawing and the application. *Journal of the Information Processing Society of Japan*, 29(7):686–693, July 1988.
 92. K. Kondo. Algebraic method for manipulation of dimensional relationships in geometric models. *Computer-Aided Design*, 24(3):141–147, March 1992.
 93. Takashi Kondo, Akio Okazaki, Mitsuo Tabata, Kazuhiro Mori, Sho Tsunekawa, and Eiji Kawamoto. Development of a reader for logic circuit diagrams equipped

- with high-speed image processing hardware. *Journal of the Information Processing Society of Japan*, 28(4):384–394, April 1987.
94. D. Kurlander and S. Geiner. A history-based macro by example system. In *Proceedings of the 1993 ACM Symposium on User Interface Software and Technology*, pages 99–106, 1992.
 95. David Kurlander and Steven Feiner. Interactive constraint-based search and replace. In *Proceedings of ACM CHI'95 conference on Human Factors in Computing Systems*, pages 609–618, 1992.
 96. Gordon Kurtenbach and William Buxton. Issues in combining marking and direct manipulation techniques. In *Proceedings of the 1991 ACM Symposium on User Interface Software and Technology*, pages 137–144, 1991.
 97. Gordon P. Kurtenbach, Abigail J. Sellen, and William A. S. Buxton. An empirical evaluation of some articulatory and cognitive aspects of marking menus. *Human-Computer Interaction*, 8:1–23, 1993.
 98. Gordon Kurtenbach and William Buxton. User learning and performance with marking menus. In *Proceedings of ACM CHI'94 conference on Human Factors in Computing Systems*, pages 258–264, 1994.
 99. Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency. In *Proceedings of ACM CHI'97 conference on Human Factors in Computing Systems*, pages 35–42, 1997.
 100. Soshiro Kuzunuki, Takanori Yokoyama, and Hiroshi Syojima and Yasushi Fukunaga. Online editing system using handdrawn proof-reading marks. *Journal of the Information Processing Society of Japan*, 27(10):1027–1034, October 1986.
 101. Chan Pyng Lai and Rangachar Kasturi. Detection of dimension sets in engineering drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):848–861, August 1994.
 102. James A. Landay and Brad A. Myers. Interactive sketching for the early stages of user interface design. In *Proceedings of ACM CHI'95 conference on Human*

- Factors in Computing Systems*, pages 43–50, 1995.
103. Adrienne Y. Lee and Peter G. Polson. Learning and transfer of measurement tasks. In *Proceedings of ACM CHI'89 conference on Human Factors in Computing Systems*, pages 115–120, 1989.
 104. F. Javier Lerch, Marilyn M. Mantei, and Judith R. Olson. Skilled financial planning: the cost of translating ideas into action. In *Proceedings of ACM CHI'90 conference on Human Factors in Computing Systems*, pages 121–126, 1989.
 105. H. Lieverman. Dominoes and Storyboards: beyond 'icons on strings'. In *Proceedings of the 1993 IEEE Workshop on Visual Languages*, pages 65–71, 1992.
 106. Robert Light and David Gossard. Modification of geometric models through variational geometry. *Computer-Aided Design*, 17(4):209–214, July 1982.
 107. Xinggang Lin, Michihiko Minoh, and Toshiyuki Sakai. Automatic diagram editing by free-hand color marking. *Journal of the Information Processing Society of Japan*, 26(4):740–747, July 1990.
 108. Peter Litwinowicz and Gavin Miller. Efficient techniques for interactive texture placement. *ACM Computer Graphics, Annual Conference Series*, pages 119–108, July 1994.
 109. Yanxi Liu and Robin J. Popplestone. Symmetry constraint inference in assembly planning — automatic assembly configuration specification. In *Eighth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, pages 1038–1044, 1990.
 110. Jerry Lohse. A cognitive model for the perception and understanding of graphs. In *Proceedings of ACM CHI'91 conference on Human Factors in Computing Systems*, pages 137–144, 1991.
 111. D. Lopresti and A. Tomkins. Computing in the ink domain. *Symbiosis of Human and Artifact*, pages 543–548, 1995.
 112. P. Luff, C. Heath, and D. Greatbatch. Tasks-in-interaction: Paper and screen based documentation in collaborative activity. In *Proceedings of CSCW '92*, pages 163–

- 170, 1992.
113. Kimiyoshi Machii and Masaki Nakagawa. A method for on-line text/drawing segmentation of stroke patterns. *Journal of the Information Processing Society of Japan*, 37(4):490–499, 1996.
 114. Jock Mackinlay, Stuart K. Card, and George G. Robertson. A semantic analysis of the design space of input devices. *Human-Computer Interaction*, 5:145–190, 1990.
 115. D. L. Maulsby, I. H. Hitten, and K. A. Kittlitz. Metamouse: specifying graphical procedures by example. *ACM Computer Graphics*, 23(3):127–136, 1989.
 116. S. Meeran and M. J. Pratt. Automated feature recognition from 2D drawings. *Computer-Aided Design*, 25(1):7–17, January 1993.
 117. Thomas P. Moran, Patrick Chiu, William van Melle, and Gordon Kurtenbach. Implicit structures for pen-based systems within a freeform interaction paradigm. In *Proceedings of ACM CHI'95 conference on Human Factors in Computing Systems*, pages 487–494, 1995.
 118. Hiroshi Murase, Toru, Wakahara, and Michio Umeda. Online Hand-Sketched Line Figure Recognition by Candidate Lattice Method with Connection Rules. *Journal of the Institute of Electronics, Information and Communication Engineers*, J67-D(3):273–280, March 1984.
 119. Brad A. Myers, Dario A. Giuse, Roger B. Dannenberg, Brad Vander Zanden, David S. Kosbie, Edward Pervin, Andrew Mickish, and Philippe Marchal. Garnet: comprehensive support for graphical highly interactive user interfaces. *IEEE Computer*, pages 71–85, November 1990.
 120. Brad A. Myers, R. Wolf, K. Potosnak, and C. Graham. Heuristics in real user interfaces. In *Proceedings of ACM InterCHI'93*, pages 302–307, 1993.
 121. Peter F. M. Nacken. A metric for line segments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1312–1318, December 1993.
 122. Uasuo Nagai and Aatoshi Terasaki. Constraining-based knowledge compiler for parametric design problem in mechanical engineering. Technical Report TM-1270, ICOT Technical Memorandum, 1990.

123. Masayuki Nakajima, Takeshi Agui, and Hisato Iitsuka. Pattern recognition for logical circuit diagrams written by freehand. *Journal of the Institute of Electronics, Information and Communication Engineers*, J68-D(11):1870–1880, November 1985.
124. Osamu Nakamura, Tsunehisa Isawa, and Toshi Minami. A separation for contact characters. *Journal of the Institute of Electronics, Information and Communication Engineers*, J68-D(7):1425–1426, July 1985.
125. Yasutomo Nakayama. Mathematical formula editor for CAI. In *Proceedings of ACM CHI'89 conference on Human Factors in Computing Systems*, pages 387–392, 1989.
126. Vishvit S. Nalwa. Line-drawing interpretation: bilateral symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1117–1120, October 1989.
127. Greg Nelson. Juno, a constraint-based graphics system. In *Proceedings of the ACM SIGGRAPH '85*, pages 235–243, San Francisco, July 1985.
128. Jakob Nielsen and Victoria L. Phillips. Estimating the relative usability of two interfaces: heuristic, formal, and empirical methods compared. In *Proceedings of ACM CHI'93 conference on Human Factors in Computing Systems*, pages 214–221, 1993.
129. Jakob Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of ACM CHI'94 conference on Human Factors in Computing Systems*, pages 152–158, 1994.
130. Erik Nilsen, Heesen Jong, Judith S. Olson, Henry Rueter Kevin Biolsi, and Sharon Mutter. The growth of software skill: a longitudinal look at learning and performance. In *Proceedings of ACM CHI'93 conference on Human Factors in Computing Systems*, pages 149–156, 1993.
131. D. A. Norman and S. W. Draper. *User Centered System Design*. Lawrence Erlbaum Associates, Inc., 1986.
132. Farid Nourani and Leo Pini Magalhaes. Management of consistency constraints in a

- CAD database system. In *International Conference on CAD and Computer Graphics*, pages 23–26, 1993.
133. K. Ogawa and K. Kato. Task analysis method using the GOMS model with grouping. In *Proceedings of 5th International Conference on Human-Computer Interaction*, pages 891–896, 1993.
 134. Kenton O’Hara and Abigail Sellen. A Comparison of Reading Paper and On-Line Documents. In *Proceedings of ACM CHI’97 conference on Human Factors in Computing Systems*, pages 335–342, 1997.
 135. Toshio Okamoto and Noboru Matsuda. An intelligent CAI for geometry proof. *Journal of the Information Processing Society of Japan*, 29(3):311–324, 1988.
 136. Toshio Okamoto and Noboru Matsuda. On recognition of student's plan in geometry proof and system's learning function for intelligent CAI. *Journal of the Information Processing Society of Japan*, 30(8):1046–1057, August 1989.
 137. Dan R. Olsen Jr. and Kirk Allan. Creating interactive techniques by symbolically solving geometric constraints. In *Proceedings of the 1990 ACM Symposium on User Interface Software and Technology*, pages 102–107, 1990.
 138. Judith Reitman Olson and Gary M. Olson. The growth of cognitive modeling in human-computer interaction since GOMS. *Human-Computer Interaction*, 5:221–265, 1990.
 139. Ron Oosterholt, Mieko Kusano, and Gobert de Vries. Interaction design and human factors support in the development of a personal communicator for children. In *Proceedings of ACM CHI’96 conference on Human Factors in Computing Systems*, pages 450–457, 1996.
 140. Sharon Oviatt. Multimodal interfaces for dynamic interactive maps. In *Proceedings of ACM CHI’96 conference on Human Factors in Computing Systems*, pages 95–102, 1996.
 141. Sharon Oviatt, Antonella DeAngeli, and Karen Kuhn. Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction. In *Proceedings of ACM CHI’97 conference on Human Factors in Computing*

- Systems*, pages 415–422, 1997.
142. Theo Pavlidis. *Algorithm for Graphics and Image Processing*. Computer Science Press, 1982.
 143. Theo Pavlidis and Christopher J. Van Wyk. An automatic beautifier for drawings and illustrations. In *Proceedings of the ACM SIGGRAPH '85*, pages 225–234, San Francisco, July 1985.
 144. Charles E. Pfefferkorn. A heuristic problem solving design system for equipment or furniture layouts. *Communications of the ACM*, 18(5):286–297, May 1975.
 145. Cary B. Philips, Jianmin Zhao, and Norman I. Badler. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *ACM Computer Graphics, Annual Conference Series*, 24(2):245–250, March 1990.
 146. James A. Pittman. Recognizing handwritten text. In *Proceedings of the 1995 ACM Symposium on User Interface Software and Technology*, pages 271–275, 1991.
 147. R. Rao, S. Card, W. Johnson, and H. Trigg. Protofoil: storing and finding the information worker's paper documents in an electronic file cabinet. In *Proceedings of ACM CHI'94 conference on Human Factors in Computing Systems*, pages 180–185, 1994.
 148. Ari Rappoport. Direct manipulation devices for the design of geometric constraint networks. In *Computer Graphics International'93*, pages 294–305, June 1993.
 149. T. H. Richards and G. C. Onwubolu. Automatic interpretation of engineering drawings for 3D surface representation in cad. *Computer-Aided Design*, 18(3):156–160, April 1986.
 150. Christian Ronse. A bibliography on digital and computational convexity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):181–190, February 1989.
 151. Dean Rubine. Specifying gestures by example. *ACM Computer Graphics*, 25(4):329–337, July 1991.
 152. Emanuel Sachs, Andrew Roberts, and David Stoops. 3-Draw: a tool for designing 3D shapes. *IEEE Computer Graphics and Applications*, pages 18–26, November

- 1991.
153. Sato Saga and Jun ichi Sasaki. A freehand CAD drawing interface based on the fuzzy spline curve identifier. *Journal of the Information Processing Society of Japan*, 36(2):336–350, February 1995.
 154. Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive pen-and-ink illustration. *ACM Computer Graphics, Annual Conference Series*, pages 101–118, July 1994.
 155. Michael Sannella, John Maloney, Bjorn Freeman-Benson, and Alan Borning. Multi-way versus one-way constraints in user interfaces: experience with the deltablue algorithm. *Software-Practice and Experience*, 23(5):529–566, May 1993.
 156. Tsuyoshi Tee Sasada. Drawing natural scenery by computer graphics. *Computer-Aided Design*, 19(4):212–218, May 1987.
 157. Eric Saund and Thomas P. Moran. A perceptually supported sketch editor. In *Proceedings of the 1994 ACM Symposium on User Interface Software and Technology*, pages 175–184, 1994.
 158. Jutta Schumann, Thomas Strothotte, Andreas Raab, and Stefan Laser. Assessing the effect of non-photorealistic rendered images in CAD. In *Proceedings of ACM CHI'96 conference on Human Factors in Computing Systems*, pages 35–41, 1996.
 159. Abigail Sellen and Richard Harper. Paper as an Analytic Resource for the Design of New Technologies. In *Proceedings of ACM CHI'97 conference on Human Factors in Computing Systems*, pages 319–326, 1997.
 160. Samudra Sengupta, Takayuki Dan Kimura, and Ajay Apte. An Artist's Studio: A metaphor for modularity and abstraction in a graphical diagramming environment. In *Proceedings of the 1994 IEEE Workshop on Visual Languages*, pages 128–136, 1994.
 161. Kenji Shimada, Masayuki Numao, Hiroshi Masuda, and Shinji Kawabe. Constraint-based object description for product modeling. In *Computer Applications in Production and Engineering (CAPE'89)*, pages 95–106, October 1989.
 162. Lluís Solano and Pere Brunet. Constructive constraint-based model for parametric

- CAD system. *Computer-Aided Design*, 26(8):614–621, August 1994.
163. Quentin Stafford-Fraser and Peter Robinson. BrightBoard: A video-augmented environment. In *Proceedings of ACM CHI'96 conference on Human Factors in Computing Systems*, pages 134–141, 1996.
 164. S. Strassmann. Hairy brushes. *ACM Computer Graphics*, 20(4):225–232, August 1986.
 165. S. A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge Univ. Press, 1987.
 166. Yasuhito Suenaga. Some techniques for document and image preparation. *Journal of the Institute of Electronics, Information and Communication Engineers*, J68-D(4):456–465, April 1985.
 167. S. Sugishita, K. Kondo, H. Sato, S. Shimada, and F. Kimura. Interactive freehand sketch interpreter for geometric modeling. *Symbiosis of Human and Artifact*, pages 56–548, 1995.
 168. I. E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the AFIPS Spring Joint Conferences*, volume 23, pages 329–346, 1963.
 169. Gerd Szwillus. Solving linear graphical constraint expressions. In *International Conference of Human-Computer Interaction jointly with 9th Symposium on Human Interface (JAPAN)*, August 1993.
 170. Gerd Szwillus. Graphical constraints. In *Tutorial of ACM CHI'94 conference on Human Factors in Computing Systems*, 1994.
 171. Hideyuki Tamura. *Introduction of computer image processing*. Souken-Shuppan, 1985.
 172. Toshikazu Tanimoto. A constraint decomposition method for spatio-temporal configuration problems. In *Eighth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, pages 145–151, July 1993.
 173. Mark A. Tapia and Gordon Kurtenbach. Some design refinements and principles on the appearance and behavior of marking menus. In *Proceedings of the 1995 ACM*

- Symposium on User Interface Software and Technology*, pages 189–195, 1995.
174. Charles C. Tappert and Ching Y. Suen. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, August 1990.
 175. Suresh Thennaragam and Gurminder Singh. Inferring 3-dimensional constraints with DEVI. In *International Workshop of Principles and Practice of Constraint Programming*, pages 78–85, May 1994.
 176. T. Tonouchi, K. Nakayama, S. Matsuoka, and S. Kawai. Creating visual objects by direct manipulation. In *Proceedings of the 1992 IEEE Workshop on Visual Languages*, pages 95–101, 1992.
 177. Remco C. Veltkamp. A quantum approach to geometric constraint satisfaction. In *the Second EUROGRAPHICS Workshop on Object-Oriented Graphics*, pages 54–70, June 1995.
 178. Dan Venolia and Forrest Neiberg. T-cube: A fast, self-disclosing pen-based alphabet. In *Proceedings of ACM CHI'94 conference on Human Factors in Computing Systems*, pages 265–270, 1994.
 179. John M. Vlissides and Mark A. Linton. Unidraw: A framework for building domain-specific graphical editors. In *Proceedings of the 1989 ACM Symposium on User Interface Software and Technology*, pages 158–167, 1989.
 180. D. J. Walton and R. Xu. Turning point preserving planar interpolation. *ACM Transactions on Graphics*, 10(3):297–311, July 1991.
 181. C. Ware. Bat brushes: On the uses of six position and orientation parameters in a paint program. In *Proceedings of ACM CHI'95 conference on Human Factors in Computing Systems*, pages 289–298, 1989.
 182. Steve Whittaker, Patrick Hyland, and Myrtle Wiley. FILOCHAT: Handwritten notes provide access to recorded conversations. In *Proceedings of ACM CHI'95 conference on Human Factors in Computing Systems*, pages 271–277, 1994.
 183. T. Whitted. Anti-aliased line drawing using brush extrusion. *ACM Computer Graphics*, 17(3):151–156, 1983.

184. Lynn D. Wilcox, Bill N. Schilit, and Nitin “Nick” Sawhney. DYNAMITE: A Dynamically Organized Ink and Audio Notebook. In *Proceedings of ACM CHI'97 conference on Human Factors in Computing Systems*, pages 186–193, 1997.
185. Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. *ACM Computer Graphics, Annual Conference Series*, pages 91–100, July 1994.
186. K. Y. Wong, R. G. Casey, and F. M. Wahl. Document analysis system. *IBM Journal of Research and Development*, 26(6):647–656, 1982.
187. Christopher J. Van Wyk. A high-level language for specifying pictures. *ACM Transactions on Graphics*, 1(2):163–182, April 1992.
188. Yasushi Yamaguchi and Fumihiko Kimura. Constraint modeling system for CAD. *Journal of the Information Processing Society of Japan*, 30(11):1512–1521, November 1989.
189. Shun-ichi Yonemura and Katsuhiro Ogawa. Human interface design guideline database using a visualized book-metaphor. *Journal of the Information Processing Society of Japan*, 35(10):2159–2169, October 1994.
190. Richard M. Young and Joyce Whittington. Using a knowledge analysis to predict conceptual errors in text-editor usage. In *Proceedings of ACM CHI'90 conference on Human Factors in Computing Systems*, pages 91–97, 1990.
191. R. E. Young, A. Greef, and P. O'Grady. Spark: An artificial intelligence constraint network system for concurrent engineering. In *Artificial Intelligence in Design '91*, pages 79–94, 1991.
192. R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: An interface for sketching 3D scenes. In *Proceedings of the ACM SIGGRAPH '96*, pages 163–170, 1996.
193. Rui Zhao. Gesture specification and structure recognition in handsketch-based diagram editors. In *Proceedings of the Fifth International Conference on Human-Computer Interaction*, volume 2 of *V. Hardware Interfaces*, pages 1052–1057, 1993.
194. Rui Zhao. Incremental recognition in gesture-based and syntax-directed diagram

editors. In *Proceedings of ACM InterCHI'93 Conference on Human Factors in Computing Systems, Typing, Writing and Gesture*, pages 95–100, 1993.

11 Appendix A. Detailed Description of Drawing Strategies

In this appendix, the drawing strategies taken by the 13 subjects for each picture in Exp. A in Chapter 3 are described in detail.

Following are supplemental representations in the tables:

E Used drawing editor and input device

Number of subjects using the drawing strategy

Smt SmartSketch

Mos Mouse

Grid Drawing a picture by utilizing grid lines for using Canvas

Multiple-line

Canvas function of drawing a polygon using a notched line, by which the vertices of a picture are clicked successively, and the last point is double clicked.

Protrusion cutting

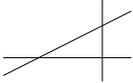
For use in the SmartSketch, a method of drawing the two lines which are in contact with each other at one endpoint by intentionally drawing the two lines once so as to cross each other, and then cutting off the unnecessary sections of the lines (Figure 3.3).

Note that in GIGA, only one strategy was observed for drawing each picture, that is, to draw the line elements one-by-one.

11.1 (a) Perpendicular V-Shape

E	Strategy	#
C a n v a s	Draw the picture as one multiple-line object using grids	4
	Draw the two lines sequentially	2
	Draw an L-shape as a multiple-line object and rotate it	1
	Draw a line, duplicate it, then rotate and move	6
S m t P e n	Draw the two lines in a single stroke	4
	Draw the two lines sequentially	3
	Draw an L-shape in a single stroke and rotate it	3
	Draw an L-shape as two lines sequentially and rotate it	1
	Draw a horizontal line, copy and rotate it to make a vertical line, and rotate the two lines together	1
	Draw a rectangle, cut out an L-shape from it, and rotate the L shape	1
		
S m t M o s	Draw the two lines in a single stroke	1
	Draw the two lines sequentially	3
	Draw an L-shape in a single stroke and rotate it	1
	Draw an L-shape as two lines sequentially and rotate it	4
	Draw a horizontal line, copy and rotate it to make an L-shape, and rotate it	1
	Draw a horizontal line, copy and rotate it, and make an L-shape using protrusion cutting, then rotate it	3

11.2 (b) Slope

E	Strategy	#
C a n v a s	Draw the picture as one multiple-line object using grids	7
	Draw the three lines sequentially	6
S m t P e n	Draw in a single stroke	2
	Draw the three lines sequentially	5
	Draw the three lines sequentially and use protrusion cutting	2
	Draw a line, copy and rotate it to make other two lines	1
	Draw a line, copy and rotate it to make other two lines, and use protrusion cutting	2
		
	Draw a rectangle and an intersecting diagonal line, and erase unnecessary lines	1
S m t M o s	Draw in a single stroke	0
	Draw the three lines sequentially	4
	Draw the three lines sequentially using protrusion cutting	5
	Draw a line, copy and rotate it to make other two lines	1
	Draw a line, copy and rotate it to make other two lines using protrusion cutting	2
	Draw a rectangle and delete two sides of it	1

11.3 (c) Isosceles Triangle

E	Strategy	#
C a n v a s	Draw the picture as one multiple-line object using grids	4
	Draw three lines using grids	5
	Draw a line, copy and rotate it to make the second line, and draw the third line	4
S m t P e n	Draw a line, copy and rotate it to make the second line, and draw the third line	9
	Draw a line, copy and rotate it to make the second line, and draw the third line using protrusion cutting	3
	Draw two rectangles, contact them, and draw two sides of the triangle, and erase unnecessary parts 	1
S m t M o s	Draw a line, copy and rotate it to make the second line, and draw the third line	9
	Draw a line, copy and rotate it to make the second line, and draw the third line using protrusion cutting	3
	Draw two rectangles, contact them, and draw two sides of the triangle, and erase unnecessary parts	1

11.4 (d) Rectangle

E	Strategy	#
C a n v a s	Draw by rectangle tool	13
S m t P e n	Draw in a simple stroke	13
S m t M o s	Draw in a simple stroke	7
	Draw a line, copy and rotate it three times	3
	Draw the four lines sequentially	3

11.5 (e) Rhombus

E	Strategy	#
C a n v a s	Draw the picture as one multiple-line using girds	5
	Draw the four lines sequentially	4
	Draw a line, copy it to make a V-shape, copy and rotate the V-shape	1
		
	Draw a line, copy it three times, rotate two copies and connect the four lines	2
		
	Draw a square, rotate it, and horizontally expand it	1
S m t P e n	Draw a line, copy it to make a V-shape, copy and rotate the V-shape	7
	Draw a line, copy it to make a V-shape, copy and rotate the V-shape, and use protrusion cutting	2
		
	Draw a line, copy it three times, rotate two copies and connect four lines	3
	Draw four rectangles, contact them with each other, draw four sides of rhombus, and erase unnecessary parts	1
		

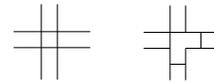
S	Draw a line, copy it to make a V-shape, copy and rotate the V-shape	4
m	Draw a line, copy it to make a V-shape, copy and rotate the V-shape, and use	1
t	protrusion cutting	
M	Draw a line, copy it three times, rotate two copies and connect four lines	6
o	Draw four rectangles, contact them with each other, draw four sides of rhombus, and	1
s	erase unnecessary parts	
	Draw a square, rotate it, and horizontally expand it	1

11.6 (f) Elbow-Shape

E	Strategy	#
C	Draw the picture as one multiple-line object using grids	10
a n v	Draw a line, copy and rotate it to make remaining five lines 	1
a s	Draw large and small rectangles, pile them up, and draw white lines to erase unnecessary lines 	1
	Draw an L-shape, copy it three times, rotate and move them 	1

S m t P e n	Draw the six lines sequentially	1
	Draw a line, copy and rotate it to make remaining five lines	1
	Draw a line, copy and rotate it to make remaining five lines using protrusion cutting	1
	Draw large and small rectangles, pile them up, and erase unnecessary lines	2
	Draw a rectangle and two lines, and erase unnecessary lines	1
	Draw an L-shape, copy it three times, rotate and move them	1
	Draw three rectangles, contact them with each other, and erase unnecessary lines	2
		
	Draw in a single stroke	3
	Draw in four strokes	1

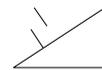
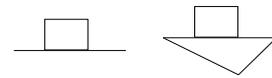
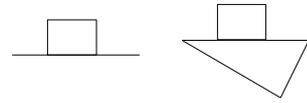
S m t M o s	Draw the six lines sequentially	2
	Draw the six lines sequentially using protrusion cutting	2
	Draw a line, copy and rotate it to make remaining five lines	1
	Draw a line, copy and rotate it to make remaining three lines, and draw other two lines	1
	Draw large and small rectangles, pile them up, and erase unnecessary lines	1
	Draw a rectangle and two lines, and erase unnecessary lines	1
	Draw an L-shape, copy it three times, rotate and move them	1
	Draw L-shape with right angle, copy it three times, rotate and move them, and cut unnecessary lines	1
	Draw in a single stroke	2
	Draw a line, copy and rotate it to make a #-shape, move two lines, and erase unnecessary lines	1

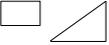
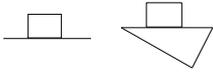


11.7 (g) Rectangle Placed on a Slope

E	Strategy	#
C a n v	Draw a rectangle object, draw a slope sequentially or by multiple-line function, rotate the rectangle changing rotation accuracy, and connect the two objects	2
a s	Draw a rectangle object, draw a slope sequentially or by multiple-line function, rotate the rectangle without changing the rotation accuracy, and connect two objects, then final picture actually does not satisfy the correct geometrical constraints	2
	Draw a rectangle object, draw a slope as a right isosceles triangle, rotate the rectangle with angle 45 degree (so there is no need to change the rotation accuracy), and connect the two objects	1
	Draw a rectangle by rectangle tool, draw a horizontal line connecting the rectangle, rotate them and draw other two lines changing the grid accuracy	1
	Draw a rectangle object, draw a horizontal line connecting the rectangle, rotate them and draw other two lines without changing the grid accuracy, then final picture actually does not satisfy the correct geometrical constraints	1
	Draw a rectangle object, draw a horizontal line connecting the rectangle, rotate them with angle 45 degree, and draw other two lines	2

<p>Draw a rectangle object, draw a horizontal line connecting the rectangle, draw other two lines changing the grid accuracy, and rotate them together</p>	0
<p>Draw a rectangle object, draw a horizontal line connecting the rectangle, draw other two lines without changing the grid accuracy, and rotate them together, then final picture actually does not satisfy the correct geometrical constraints</p>	1
<p>Draw a slope sequentially or by multiple-line function and add remaining three lines as a multiple-line object</p>	2
<p>Draw a slope sequentially or by multiple-line function, draw a line for the rectangle, and copy the line for remaining two lines</p>	1



S m t P e n	Draw the six lines sequentially	3
	Draw the six lines using protrusion cutting	3
	Draw a slope, copy a line in slope to make remaining lines	1
	Draw a slope in a single stroke and draw remaining three lines in a single stroke	1
	Draw a rectangle in a single stroke, draw a slope sequentially or by multiple-line function using protrusion cutting, rotate and move the rectangle	1
		
Draw a rectangle in a single stroke, draw a horizontal line connecting the rectangle, rotate them and draw other two lines	3	
		
Draw a rectangle in a single stroke, draw a horizontal line connecting the rectangle, draw other two lines, and rotate them together	1	
		
S m t M o s	Draw the six lines sequentially	2
	Draw the six lines using protrusion cutting	8
	Draw a slope, copy a line in slope to make remaining lines	1
	Draw a rectangle in a single stroke, draw a slope sequentially, rotate and move the rectangle	1
	Draw a rectangle in a single stroke, draw a horizontal line connecting the rectangle, draw other two lines, and rotate them together	1

Acknowledgments

I would like to express my sincere gratitude to Professor Akinori Yonezawa and Professor Satoshi Matsuoka for their advises.

I also express my thanks to Takeo Igarashi for his cooperation in implementing the candidate-selection method.

I would like to thank to Fumio Wakamori, Haruo Takeda, Gen Uchiyama, Toshihiro Hananoi, Bin Ito, Hidefumi Masuzaki, and Etsuo Horikawa, who were my superiors at Hitachi Ltd., for their various suggestions on the dot-masked revision sheet method.

I also would like to offer my thanks to all people who cooperated as subjects in the evaluation experiments: the employees of Hitachi Ltd. and the students at the University of Tokyo and those at Aoyama Gakuin University.

Finally, I would like to thank to my husband and two children for their continuous encouragement.