ELSEVIER

Computer-Aided Design 38 (2006) 210–223

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

A new modeling interface for the pen-input displays

Dae Hyun Kim [a,*], Myoung-Jun Kim [b]

[a] *Institute for Graphic Interfaces, 11-1, Daehyun-dong, Seodaemun-gu, Seoul 120-750, South Korea*
[b] *Division of Digital Media, Ewha Womens University, 11-1, Daehyun-dong, Seodaemun-gu, Seoul 120–750, South Korea*

Received 27 January 2005; received in revised form 12 October 2005; accepted 16 October 2005

Abstract

Sketch interactions based on interpreting multiple pen markings into a 3D shape is easy to design but not to use. First of all, it is difficult for the user to memorize a complete set of pen markings for a certain 3D shape. Secondly, the system will be waiting for the user to complete the sequence of the pen markings, often causing a certain mode error. To address these problems, we present a novel, interaction framework, suitable for interpretations based on single-stroke marking on pen-input display; within this framework 3D shape modeling operations are designed to create appropriate communication protocols.

Keywords: Pen-input displays; Direct manipulation; Sketch

1. Introduction

Most of the 3D modeling systems of WIMP[1] style do not match many important benefits of traditional tools such as pencil on paper to communicate design ideas at an early stage [8]. One of the main advantages of pencil-on-paper interface is that it allows ambiguous sketching, which is quite opposite to the existing sketch-based approaches having uniform binding similar to UNIX command line interpreter (i.e. one pen marking results in one command) and does not distract the focus of the attention from drawing task (e.g. for selecting menu buttons). Pen-input display implements most features of the pencil-on-paper paradigm although the pen-input display still have limited resolution and the display is not as handily manipulated as paper. Meanwhile, it is still questionable freehand drawings can be quickly and directly led to a feasible 3D model for engineering purposes. For this, many *sketch-based approaches* [2,4,9,12,15,19,25,33] have been suggested; user inputs pen markings and the system interprets them as a geometric primitive or as a modeling operation. However, providing interpretation to the pen markings, called *informal interface* [19,20], has posed the following problems:

- *Uniform binding:* Sketch-based systems interpret each pen marking and convert it into a primitive component (or a command). However, not all the pen markings carry a definite user intention, since they often come out of a vague idea.
- *Geometric ambiguity:* Geometries contained in a 2D pen marking may be insufficient to be used to infer 3D geometries, requiring further user interventions.
- *Memorability:* Usually a sequence of pen markings are required to constitute an interpretation into a 3D shape (e.g. a box, a cone, etc.). Therefore, it becomes even harder for user to remember the constituents of the pen sequence.

The problems listed above can now be rephrased: ‘inappropriate communication protocols between the user and the system’. To handle these problems, research needs to span topics from user interface design to 3D modeling. However, the earlier 3D sketch systems seem to put aside the issues of user interactions, while immensely emphasizing the importance of 3D modeling functionalities.

Contributions: We solve the problems by (1) reducing the number of pen markings that should be recognized for 3D modeling operations (mostly arrow-shaped pen markings), (2) further simplifying the multiple-stroke based pen markings of the other approaches [2,12,25,33] into single-strokes,

* Corresponding author. Fax: +82 2 3277 3893.
E-mail address: daek@acm.org (D.H. Kim).
[1] Windows, Icons, Menus, and Point-and-click.

(3) however, diversifying the number of shapes that can be made from the recognized pen markings through an *interaction framework*, but not by adding more recognizable pen markings burdening the users in remembering them, which, meanwhile, was often the case for other sketch-based 3D modeling approaches. Our design of the sketch-based 3D modeling system is based on the following key ideas:

- *Interaction framework:* We generalize an interaction framework of pen-based modeling system to both 2D and 3D using *immediate-* and *selective-manipulation*. All the input pen markings fall into one of the two techniques so that a certain feedback can be always given to the user (Section 4).
- *3D shape modeling:* We provide a novel 3D modeling environment that requires only a few single-stroke pen markings supporting the following diverse CAD operations:
  - A consistent style of model generation based on the *generative modeling* paradigm [29] such as extrusion, sweeping (translation, rotation, freeform), lofting, filling.
  - *Construction tools:* positioning the working axis and working plane and model transformations such as axis-aligned translation, rotation, and scaling.
- *Sketching gesture:* We devise most of recognizable pen markings for the generative modeling to have an *arrow* shape in common. Since this gesture is used in every corner of sketching practices to represent a certain behavior or relation between entities, the user can remember and try them out more easily and instantly.

The problem of uniform binding has not been completely resolved but relieved by introducing the selective-manipulation technique. Geometric ambiguity has been overcome by embedding the interaction framework into the modeling pipeline; A certain direct manipulator follows after user pen marking as a feedback with which user can refine her/his intention. Memorability issue has been improved by adopting the generative modeling paradigm and introducing a sketching gesture: most modeling operations are invoked from the 'arrow' marking that resembles the modeling behavior of the generative modeling.

Paper organization: The rest of this paper is organized as follows. We survey previous work on sketch-based modeling in Section 2, and give an overview of our system in Section 3. In Section 4, we describe our interaction framework for pen-based modeling generalized for both 2D and 3D modeling. Also, we build the 3D modeling environment consisting of various generative modeling commands based on the interaction framework. In Section 5, we discuss the implementation issues and present the user testing results of the modeling system. We conclude this paper and present future work in Section 6.

2. Previous work

We give a brief survey of interaction and modeling techniques used in other sketch systems, and discuss the pros and cons of using the techniques provided by these systems. For more extensive survey of the field, we refer the readers to [21,16]; for a comparison between many sketch-based systems in a wider technological view, see [21]. In Section 2.1, we discuss interaction techniques developed mainly for 2D planar drawings and other recognition techniques such as voice recognition, although they have not been considered for 3D sketching systems. These techniques serve on the issue how to properly feedback the user input and to let the user know what has been done by the system for the input.

Subsequently, we discuss the other sketch-based modeling systems for 3D shapes. Through this section, we find out what could have been improved to solve the problems of the informal interface, listed in Section 1.

2.1. Interaction techniques

Rubine suggested the *two-phase interaction* and *eager recognition*, in which a gesture recognition is immediately followed by a direct manipulator [27,28]. With the two-phase interaction, the user is forced to signal the end of the gesture that makes input behavior awkward. Moreover, physically the same action (e.g. dragging) is used for conceptually two different tasks (e.g. drawing and manipulation). An example can explain the eager recognition: the user performs a *create-line* marking by drawing a line, followed by a stopping motion for one-quarter second while continuing to press the mouse button. The system recognizes the gesture and creates a *straight-line*; remarkably, one endpoint of which is placed at the start of the gesture and the other at the current mouse position. The user may drag around the latter endpoint (rubber banding), while still pressing the mouse button.

A good reason behind this input constraint is to keep the physical tension of pressing the mouse button to the end of each primitive task, because such periods of tension are accompanied by heightened state of attentiveness and improved performance [3,28]. However, the two-phase interaction imparts a feeling to the user that only a special parameter is important. The eager recognition removes the awkward stopping motion from the two-phase interaction; the gesture is recognized as soon as enough of it has been drawn. However, this sharply reduces the number of recognizable gestures (e.g. line and rectangle cannot coexist within one gesture set; while drawing a rectangle, a line will be already recognized), and thus, the range of applications. In our approach, meanwhile, these problems caused by this awkward input pattern are solved by adopting the *hovering action* which can be found only in pen-input displays recently.

Pegasus in [13] is a line-drawing system, where the user sketches a line and the system decides on the position of two endpoints. A constraint solver produces candidates that satisfy the inferred constraints. All preferable candidates are presented to be chosen. Its drawback is that the candidate space is often too thickly populated, distracting the user from drawing. Suggestive interface [14] extends Pegasus into 3D space; it displays the possible 3D objects that can be generated from the current marking (again straight line) under the canvas.

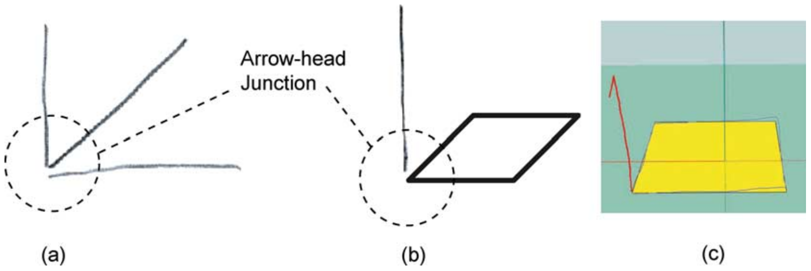


Fig. 1. Three different ways to define a box object from pen markings. (a) Bimber et al. and Zeleznik et al. [2,33]. (b) Hwang et al. and Qin et al. [12,25]. (c) Our approach.

This makes the user change the focus of attention into different places distracting the user. See [23] for the survey on what is called "mediation" techniques in many areas. In our approach, we introduce *selective manipulation* in which only two candidates are given for a few drawing primitives such as poly-line, circular arc, spline, or mixture of them in one stroke: inference results and further interactive drawing for the drawing primitive inferred.

2.2. Sketch-based 3D shape modeling

To represent diverse 3D shapes, current sketch-based modeling systems have implemented many gestures [2,33], which can be recognized and directly converted to 3D shapes. Research suggests, however, that limiting the number of gestures is needed considering the ability of the user to memorize them; for example, the number of the frequently used gestures for the Newton PDA system is reported to be no more than four [22]. Although very special effort can be exerted to design very memorable pen markings, there still exists a certain cognitive limit.

In [2,33] description of each target shape consists of a sequence of pen markings: for example, on the left of Fig. 1, a box is created from three strokes which should form arrow head junction; they do not need to have any ordering (e.g. left to right) but between them no other pen markings should come in. Between these pen markings, no feedback is given but only after recognizing them as a box. *Feedback* is an important criterion for a good interactive system [6]. Often, the user does not get any hint of what her input is bringing about: "Has the pen marking been correctly recognized?" or "Can I input the next pen marking?" This tends to cause unintended mode errors, when the user forgets what was input previously. For a detailed discussion on mode errors in interactive systems, see [26]. On the other hand, such lines are limited to represent only 3D objects, users are strongly prohibited from performing sketching (i.e. scratching, doodling, and so on): It is the problem of *uniform binding*.

Hwang et al. [12] suggested a 3D sketch system based on feature detection, i.e. box and cylinder features. For example, suppose that a new input line constitutes an arrow-head junction with the two existing lines that are already recognized as a rectangle (see on the right of Fig. 1) and they are perpendicular in a fixed isometric projection. Then, the system recognizes them as a box feature. Notice that unlike Zeleznik's approach no constraint such that the three lines should be drawn successively is imposed. However, it restricts the user's input behavior in that he/she cannot draw the arrow-head junction which may be flat.

Qin et al. [25] made the following improvements to Hwang's approach: (1) one pen marking can include several segments, each of which can be a primitive, (2) more features, i.e. revolution surface and sweeping, are added (see Fig. 2(c)). The arrow-head junction in Hwang's system is replaced by *a closed curve and an extrusion line emanating from the curve*. Eggli et al. [5] added more interactive features to Hwang's approach; before invoke extrusion operation, user picks a menu button to let the system know that the coming input marking is for extrusion.

Differently from the above approaches, in our *immediate manipulation*, after the arrow shaped pen marking, a direct manipulator comes for feedback while the pen is hovering over the screen surface, which adjusts the extrusion depth or the amount of the rotational sweep.

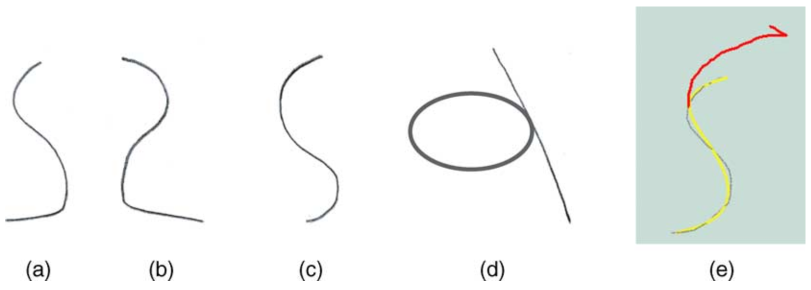


Fig. 2. (a) In Eggli's approach, two nearly symmetric curves (only open curves) are input to produce a rotational sweep. (b) In Bimber's and Zeleznik's approaches, one open curve (only open) just produces a rotational sweep. (c) In Qin's approach, a curve tangentially touching a circle produces a rotational sweep. (d) Our approach.

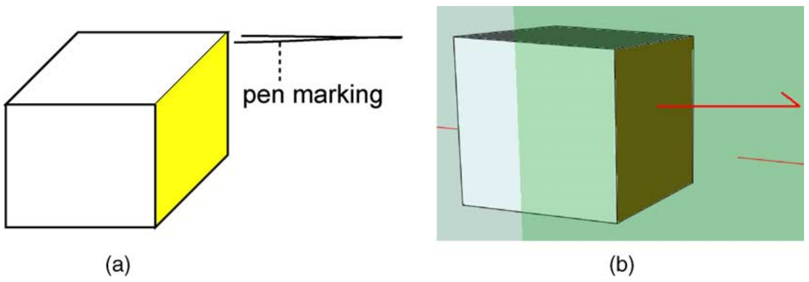


Fig. 3. (a) Zeleznik's approach to translate a face. (b) Our approach.

Lim et al. [21] made a rigorous survey on most sketch-based modeling systems published so far and proposed a method to represent vague geometry sketched. This allows the user to draw vague objects and the system keeps them in a hierarchy. For example, for an object, such as a cube, the hierarchy on its geometry consists of at least five levels. However, it is still not clear how to use such information in 3D modeling environment. It seems to us that all the decisions on a specific type of the curve objects at the leaf level are put to the later applications. In our approach, this problem of reaching a decision is provided with a user interaction technique, so called, selective manipulation.

We weigh the pros and cons of the aforementioned sketch-based modeling schemes. As far as feedback is concerned, they all have uncomfortable interaction frameworks since there exist only input pen marking and the resulting interpretation; no intermediate steps are taken. Nonetheless, object transformations in [33] are valuable in that they can be used for further diversifying base shapes or for recovering from recognition errors. However, they require another different pen marking to remember (see Fig. 3); nevertheless, they provide no explicit construction tools other than face transformation. Meanwhile, Qin's system is more consistent in modeling style than the others in that box and cylinder features are specified in a consistent way. However, as for the number of representable shapes, it is not comparable with [33].

3. System overview

In this section, we give an overview of our sketch-based modeling environment and, in the following sections, we explain two important issues relevant to devising such an environment in detail, i.e. interaction framework and modeling framework. We begin this section by defining terminologies that we use throughout the paper.

3.1. Terminologies

Definition 1 (Dragging and hovering action). Pen movement while the pen is touching a flat screen surface is called 'dragging'. Movement without touching but keeping within a certain distance to the surface is called 'hovering'.

Hovering is identified as a gestural element. This action is supported in most tablet PCs which use electromagnetic digitizer [31]. Moreover, this action lets the user move the cursor quickly and easily.

Since the pen movement is made by three fingers, wrist, and forearm in unison, kinematically the transition from the dragging action to the hovering action is continuous. By recognizing this gestural element, we separate, physically as well as conceptually, 'drawing on paper' and 'manipulating (editing) an object', on the continuously changing curve of muscular tension. Dragging is used for drawing an object and hovering action is used for manipulating an object.

Definition 2. (Picking action). Let p be a pen marking and $\|p\|$ its arc length. We call the pen marking 'picking' if $\|p\|$ is small (e.g. less than 5 pixel-lengths). In the context of using the mouse, it is also called 'clicking'.

Now, we define what interpretation means in the context of our application, since it is sometimes used with different meanings in different areas.

Definition 3 (Interpretation). Let C be a set of general commands, and T a mapping from the pen markings to C. If p is a pen marking and $T(p)=c$, $c\in C$, we call c the 'interpretation' of p. More precisely, c is the (T,C)-interpretation of p.

Notice that pen marking itself can be treated as an interpretation with further user interaction.

Definition 4 (Identical interpretation). When a pen marking is mapped to itself, the resulting interpretation is called 'identical interpretation', T^0, also known as 'personal touch', 'as-is' drawing, or 'content marking' [9,18].

In our approach, most basic result of pen marking is always identical interpretation as will be discussed in the later section for interaction framework; which lets the system stay like a sketch pad. None-identical interpretation needs more parameters to be set; for example, an interpretation, 'rectangle', will have two corner points to be presented. Thus, we need more to complete the interpretation.

Definition 5 (Command parameters and Inference). The variables that can affect the final presentation of an interpretation c are denoted by $\{\tau=\tau_i|i=0,\ldots,m\}$ and called 'command parameters'. The size of τ is denoted by $O(\tau)$. Thereby, the final presentation can be represented by $c(\tau)$. The process of determining τ by the system is called 'inference'.

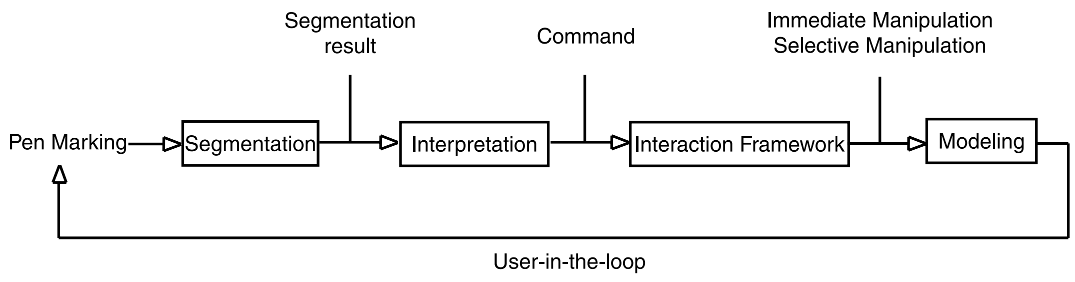


Fig. 4. The pipeline of our sketch-based modeling environment.

To help understanding what command parameters and the inference in this context are, we bring one simple example. To visualize what has been done by the system, the input pen marking *p* which looks like a rectangle, should be recognized as a rectangle first (i.e. *c*). Then the rectangle should have two diagonal corners (or one axis and two corners). This information can be directly given by the segmentation results. However, it is often the case that the rectangle can be aligned with neighbor objects. This additional information can be obtained by inference.

3.2. Pipeline

The pipeline of our sketch-based modeling environment consists of four major steps as follows (also shown in Fig. 4):

(1) *Segmentation:* Each pen marking is segmented to enhance the recognition, thus, eventually improving the overall user interaction by better recognition results. More shape features are extracted from the segmentation results and smoothing each segment; for example, primitive type of each segment (i.e. line, circle, etc.) and chord-length parametrization of each segment.
(2) *Interpretation:* Primitive shapes are inferred from the shape features and bound to the context of modeling environment to yield a complete command with the command parameters.
(3) *Interaction:* The command is combined with an appropriate interaction framework. For a fluent flow in the modeling process, we design a new interaction framework generalized for both 2D and 3D pen-based modeling: *immediate-manipulation* and *selective-manipulation*.
(4) *Modeling:* Most 3D modeling operations reminding us of a certain behavior (e.g. extrusion, transformations) are combined with immediate-manipulation, while drawing operations (e.g. line and curve primitives) are combined with selective-manipulations. Further interaction designed on the command parameters diversify shapes.

In the following sections, we will elaborate on interaction and modeling; for the segmentation and the interpretation step we refer the readers to the authors' another paper on pen marking segmentation. In particular, to recognize a primitive shape from the segmentation results, we adopted the approach of Hammond et al. [10]; they proposed a general language to describe shapes using the shape features.

4. Interaction framework

In this section, we present the interaction framework of our pen-input based 3D modeling environment supporting two different types of manipulation: immediate-manipulation combining the interpretation and the direct manipulator and selective-manipulation combining the two more selectively at the discretion of the user. The former improves the awkward input behavior of the two-phase interaction—which was discussed in Section 2—and the selective-manipulation permits *as-is* drawings, thus, allowing the user to perform vague drawings. Within each type of manipulation, 3D modeling operations are implemented.

For each pen marking, the system decides whether it should be directed to either immediate manipulation or selective manipulation. When the pen marking falls within the depiction shown in the table (Fig. 7), an immediate manipulation comes; as for its metaphor, the pen marking is categorized as the tasks of editing objects, such as transformation, or the tasks of 3D modeling such as extrusion, which emphasizes behavior rather than drawing. Meanwhile, all the other pen markings not connected to the immediate manipulation are combined with selective manipulation; that is, drawing behaviors (not editing) results in selective manipulation. Within the selective manipulation, user can just either perform vague drawings or selectively achieve hard-line drawings such as curve primitives, which tastes of computer-aided design.

4.1. Immediate-manipulation

The interaction scenario for immediate-manipulation is as follows:

(1) The user inputs a pen marking.
(2) The system interprets it and presents a direct manipulator corresponding to the interpretation.
(3) The user interacts with the manipulator, while hovering over the screen surface.

We call the technique immediate-manipulation, since the direct manipulator follows immediately after the pen marking. A possible drop of muscular tension in the transition from

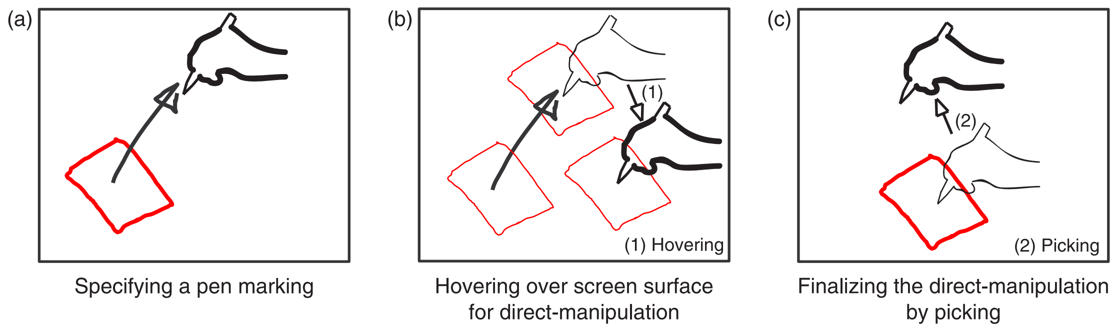

(a)
Specifying a pen marking
(b)
(1)
(1) Hovering
Hovering over screen surface for direct-manipulation
(c)
(2)
(2) Picking
Finalizing the direct-manipulation by picking

Fig. 5. Immediate-manipulation. Hovering over the screen surface right after the pen marking moves the object, which is, then, drawn with thin solid line. After the picking, which fixes the position of the object, hovering causes nothing.

the dragging action to the hovering action is, though slight, compensated by visualizing the direct manipulation providing visual feedback.

For example, assume that the user specified a *move-object* command by a pen marking as depicted in Fig. 5(a). First of all, the system interprets the pen marking as the move-object command, denoted by *c*. Along with the intermediate result $c(\tau=\{\tau\})$—τ is initialized using a simple inference routine because it is used to inform the user of what happened only—a manipulator immediately follows to let the user interactively decide on the command parameter τ. So, while the user hovering over the screen surface the target object follows the pen accordingly updating τ (see Fig. 5(b)). The user can fix command parameter τ by picking at the screen surface.

To finalize the manipulation explicitly, which corresponds to the *finalize manipulation* in the figure, the user pushes the barrel button on the pen. The manipulation will be terminated automatically when $O(\tau)$ is fixed and all τ_i in τ have been confirmed. When $O(\tau)$ is variable and at least τ_0 has been confirmed, pushing the barrel button will terminate the manipulation and leave $c(\tau)$ on the canvas. Pushing the button while τ is under-determined will discard $c(\tau)$.

Hovering action of the user is distinct enough, both conceptually and physically, from the action of pen making because during the hovering action pen does not touch the paper or leave ink on the paper. Sketching is more concerned about the act of creating marks on paper rather than editing and manipulating a drawing, which, on the other hand, are more emphasized in most CAD applications.

4.2. Modeling operations with immediate manipulation

We adopt *generative modeling* [29], which are all invoked by one class of pen markings, the *arrow*; moving a profile curve (or surface), the trace after the movement forms a final model. This consistent style of modeling automatically reduces the burden of remembering the recognizable pen markings; for example, generating different 3D shapes, such as cube, cylinder (cone), sphere, torus, can be obtained by the same style of pen marking. Moreover, tools are supported for the completeness as a modeling tool; for example, model transformations or construction tools such as a working plane and a working axis.

We illustrate an example of taking advantage of immediate-manipulation to model a 3D shape. Fig. 6 shows how the user creates a cylinder using our system.

(1) The user performs an arrow pen marking, emanating from a face.
(2) Once the system interprets it as an extrusion command, denoted by *c*, two direct manipulators immediately follow one by one to allow the user to interactively edit the command parameters, $\tau=\{\tau_0,\tau_1\}$: τ_0 is the extrusion amount and τ_1 the scaling factor of the top face.
(3) The first shape parameter is adjusted by the hovering-over action and the system displays the model changing the height of the extrusion.
(4) The extrusion height is determined by the picking operation.

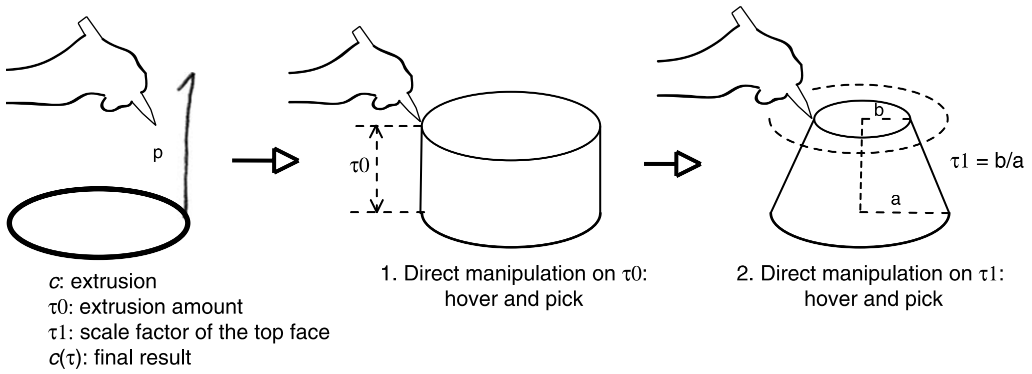

p
τ0
b
a
τ1 = b/a
c: extrusion
τ0: extrusion amount
τ1: scale factor of the top face
c(τ): final result
1. Direct manipulation on τ0: hover and pick
2. Direct manipulation on τ1: hover and pick

Fig. 6. Creating a cylinder, a cone, and a cut cone in one immediate-manipulation.

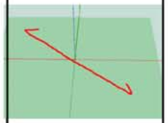
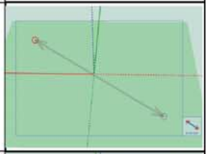
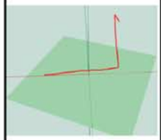
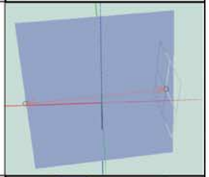
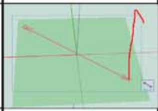
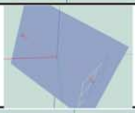
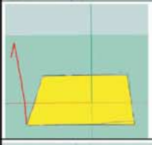
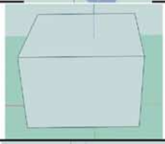
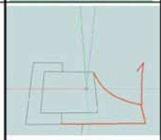
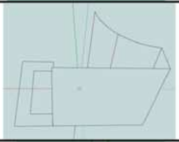
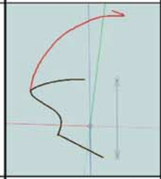
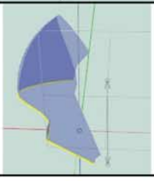
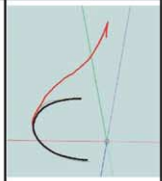
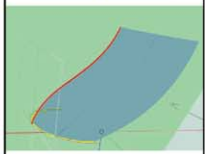
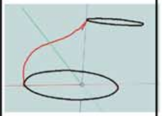
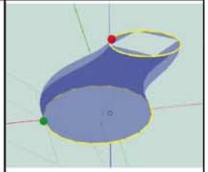
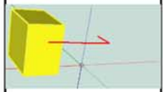
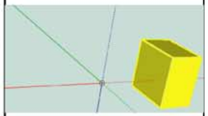
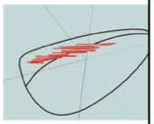
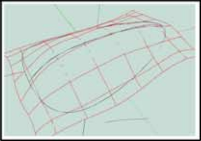
Commands	Command Parameters	Invoking Marking	Context	Result
Working Axis	τ_0: One end point of the working axis τ_1: The other end point of it		No context	
Working Plane 1	τ_0 and τ_1: Two end points of an axis lying on the working plane τ_2: A vector on the plane that passes through τ_1 and that is perpendicular to $\overline{\tau_0\tau_1}$.		No context	
Working Plane 2	τ_0: A vector on the plane that passes through one end point of an existing axis and that is perpendicular to the existing axis.		Straight arrow pen marking starting on an existing axis.	
Face Extrusion	τ_0: The extrusion amount; i.e., the height of the resulting object. τ_1: The scale factor with respect to the x-axis with the pivot located at the center of the base face.		Straight arrow pen marking starting at a corner of a face picked previously.	
Edge Extrusion	τ_0: The extrusion direction. τ_1: The extrusion amount; i.e., the height of the resulting object.		Straight arrow pen marking starting at a corner of the edges selected previously.	
Rotational Sweep	τ_0: Rotational amount in angle. τ_1 (only for the closed profile): the scale factor of the closed profile.		Rotational arrow pen marking starting at an edge (or edges) selected.	
General Sweep	τ_0: A 3D plane that is perpendicular to the tangent at the curve point of the profile curve nearest to the starting point of the arrow. τ_1: The parameter value of $C(t)$, such that the sub curve of $C(t)$, $0 \le t \le \tau_1$ is used for the trajectory. τ_2 (only for the closed profile): the scale factor of the		Differently from the rotational sweep, shaft of the pen marking is freeform. The shaft of the arrow pen marking is approximated with a B-Spline curve $C(t), 0 \le t \le 1$.	
Lofting	τ_0: The vector, which lies on a plane perpendicular to the line that connects two connection points on the existing two curves, where the pen marking starts and ends, and which determines the projection plane for the shaft. τ_i: The direction marks on the connection points, which can be toggled to opposite direction, to reverse the corresponding curve.		The shaft of the pen marking must start at a point on a curve selected and end at a point on the other curve selected.	
Transformations	The command parameter for translation, τ_0, is used to form the translation vector. The starting point of the vector is inferred from the starting point of the pen marking.		In order to be interpreted as a face transformation, the pen marking should begin inside the face.	
Filling Wires	No command parameters		Over the selected curves, draw scratchy marking.	

Fig. 7. Summarizing the immediate-manipulations. Shaft of the arrow pen marking means the remaining part of it after the arrow-head.

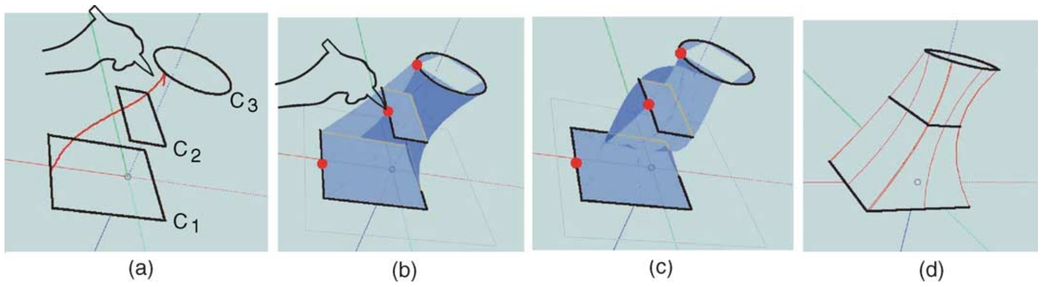


Fig. 8. Lofting when the number of the section curves are three. (a) The user performs a lofting pen marking, whose shaft emanates from a point on the first curve and ends at a point on the *N*th curve. (b) After the pen marking is interpreted as a lofting command, the user adjusts the seam of the closed curve. (c) The user changed the direction of $\mathbf{C}_2$. (d) Final result.

(5) Right after this, another direct manipulator comes to help the user adjust the scaling factor of the top face. Depending on the scaling factor, the resulting model can be a cylinder, a cone, or a cut cone.

Consequently, the user can produce different shapes such as a cylinder, a cone, or a truncated cone with only one pen marking. This has two advantages over earlier approaches such as [2,33] that require multiple pen markings: improvement in memorability and easy recovery from inference errors. Since user's sense of 3D space is not always correct, projecting 2D stroke into 3D space often conveys unwanted errors, not quantitatively but in user satisfaction.

Fig. 7 summarizes the pen markings that our system supports for 3D shape modeling, followed by immediate-manipulation. Fig. 8 generalizes the lofting operation in Fig. 7, so that N section curves, $\{C_k(t), k=1,\ldots,N\}$, are used for the final shape of the surface. Fig. 9 shows how transformation commands are issued by performing different pen markings. The recognition of the pen marking to issue the *scaling* command should be addressed; if the knob around the starting point of the pen marking is detected, the rest can be recognized as any other arrow marking. While user's inputting of the pen marking, current length of the pen marking (l) and length of the diagonal of the bounding box (d) are known. Setting the maximum knob size to k, if $l > d\pi$ and $d < k$, the pen marking is reported to have a knob.

Extracting command parameters: Command parameters which are inferred from 2D pen markings are not very important since they are just the initial feedback to the pen marking. They will be later on modified through further user interaction. Problem, anyway, is that this inference causes geometric ambiguity. Our methods heavily use projection method. For working axis command, for example, the two end points of the arrow in 2D are projected onto the current working plane along the viewing direction and the resulting 3D points are used for the two command parameters. For another example, let us consider general sweep. The starting point of the arrow projected onto the profile curve. From the curve point, a plane normal to the tangent direction at the point is computed. The second command parameter is then computed from the B-spline curve approximating the shaft of the arrow.

4.3. Selective-manipulation

Differently from the immediate-manipulation that emphasizes direct manipulation, selective-manipulation is used when *as-is* drawing is more emphasized and the application should be more sketch-based, thus letting the user draw ambiguous objects.

The difference from other approaches can be characterized by a question: Whether each pen marking results in a non-identical interpretation. If yes, this will force the user to draw like the left side of Fig. 11 for the same structure (or form) [4,9,11,19,25,5,2,12,13]. However, to allow more freedom for the user to draw like the right side of Fig. 11, a new interaction technique is necessary.

One metaphor for selective manipulation is this: "A user places a transparent paper on top of a Wacom LCD tablet, and sketches her idea on the paper. The system interprets each pen marking and presents to the user what it can do (but, not what it

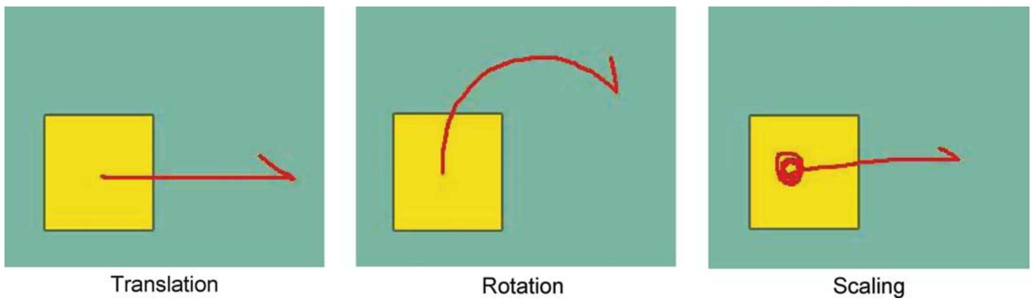


Fig. 9. Affine transformations of a face.

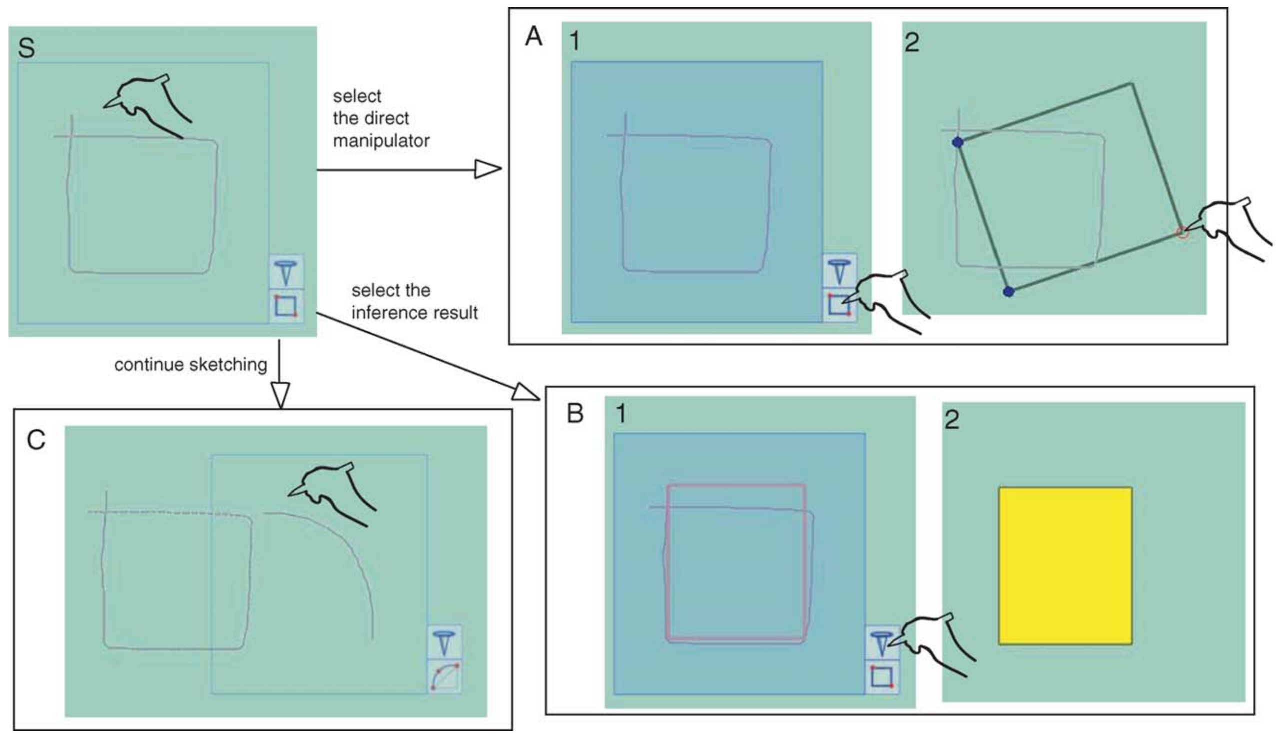


Fig. 10. Selective-manipulation (S). The user provides a rectangular pen marking. (A:1) The cursor is placed on a iconic label during hovering action and the system visualizes the selectable layer. (A:2) The user picks the icon and started to manipulate the object. (B:1) The cursor is placed on another iconic label during hovering action and the system visualizes the selectable layer with the inference result. (B:2) The user picks the icon and the system presents the inference result at once. (C) The user continues to perform another pen marking without any other action.

did) for the input, under the paper. Until the user finds his/her idea concrete on the paper, she is not forced to go to hard-line drawing which is always made available under the current pen marking by the system."

The interaction scenario for selective-manipulation is as follows:

(1) The user inputs a pen marking and the system interprets it (Fig. 10(S)).
(2) The user can choose three types of selective-manipulation:
- Continued manipulation by the user.
  (a) The system visualizes the tentative interpretation after inference under the pen marking on a selectable layer, only when the cursor is hovering over an iconic label attached to the bounding area of the pen marking (Fig. 10(A:1)).
  (b) The user may pick up the interpretation to be put into a modeling context (Fig. 10(A:2)).
- Inference by the system
  (a) The system visualizes only the selectable layer when the cursor is hovering over another iconic label (Fig. 10(B:1)).
  (b) By picking the icon, the user can invoke a direct manipulator assigned to the interpretation and manipulate it to his/her intention (Fig. 10(B:2)).
- As-is drawing
  The user can continue to sketch, resulting in automatically discarding the tentative interpretation but keeping the pen marking 'as is' (Fig. 10(C)).

The selectable layer is a transparent rectangular window with iconic labels (see Fig. 10), having similar usages to the transparent medium in [17,24,30]. The selectable layer is placed under the *as-is* drawing of the input pen marking. Since the transparent drawing can prioritize information within a complex display [30], the selectable layer suggests, to a certain degree, what can be obtained with further interaction, less distracting the user from his/her sketching process (i.e. the selectable layer is drawn with a low visual priority while the as-is drawing with a high visual priority). In our approach we do not visualize the layer at all unless the cursor hovers over the iconic labels (Fig. 11).

Fig. 10 illustrates an example of the selective-manipulation. In 2D space, right after each pen marking p (Fig. 10(S)), the system tries to interpret it; in this example, the user specified a rectangle. A corresponding selectable layer is laid

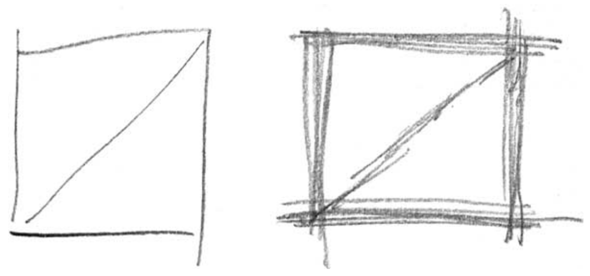


Fig. 11. *Left:* A drawing on the assumption that each pen marking carries a certain non-identical interpretation. *Right:* For the same form, the user added more markings to existing markings.

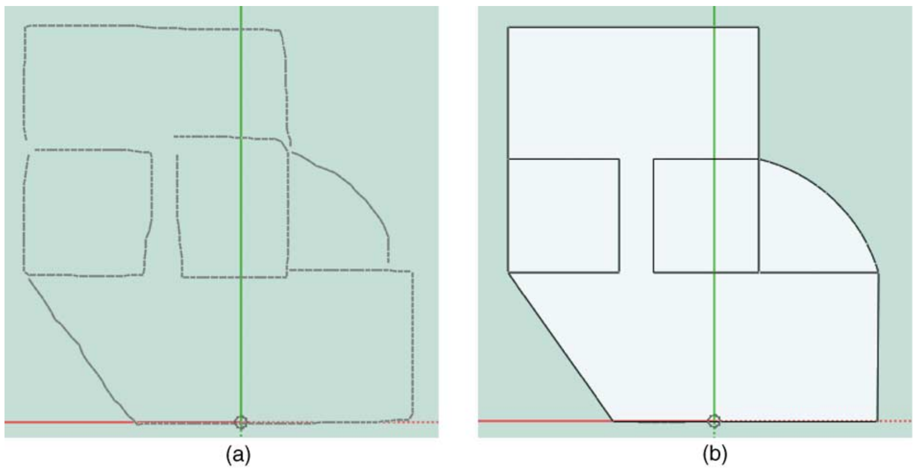


Fig. 12. (a) The user drew a planar map. (b) The system produced the interpretation and inference result by selecting the interpretations. Remarkably, the user could just leave the pen drawing (a) by just continuing sketching.

under the *as-is* drawing of the pen marking when the cursor moves to an iconic label. Similarly to the immediate-manipulation, the resulting interpretation $c(\tau)$ requires several parameters to be set, first tentatively by the inference routine. For example, the rectangle command needs two additional parameters, lower left corner τ_0 and upper right corner τ_1: $c(\tau=\{\tau_0,\tau_1\})$. The command parameters τ_0 and τ_1 are initialized using an inference rules and $c(\tau)$ is transparently rendered on the selectable layer (see Fig. 10(B:1)). For the inference and the direct manipulators we use Bier's snap-dragging [1]. Detected corner points are attracted to alignment objects; because his approach does not support automatic generation of alignment objects to which the cursor is attracted we established some rules for the generation of alignment

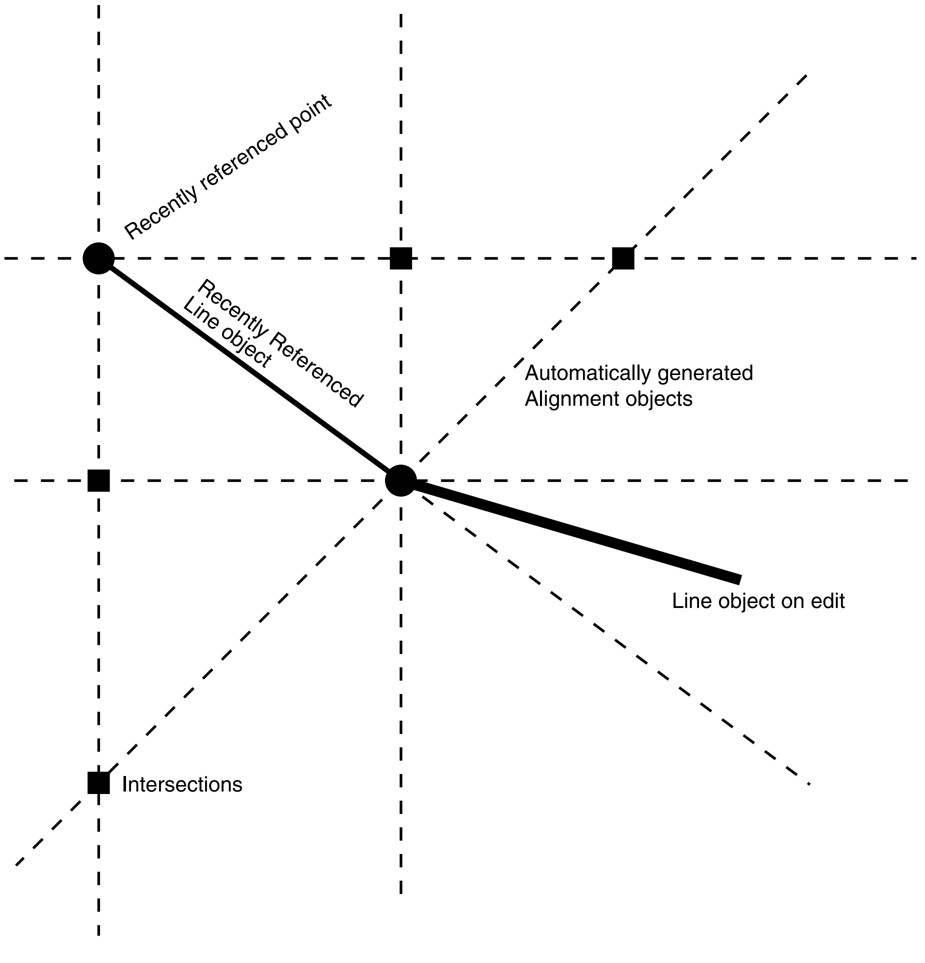

Recently referenced point
Recently Referenced Line object
Automatically generated Alignment objects
Line object on edit
Intersections

Fig. 13. Automatic generation of alignment objects (dashed lines). Black spots of circle shape denote point objects and rectangular spots intersection points.

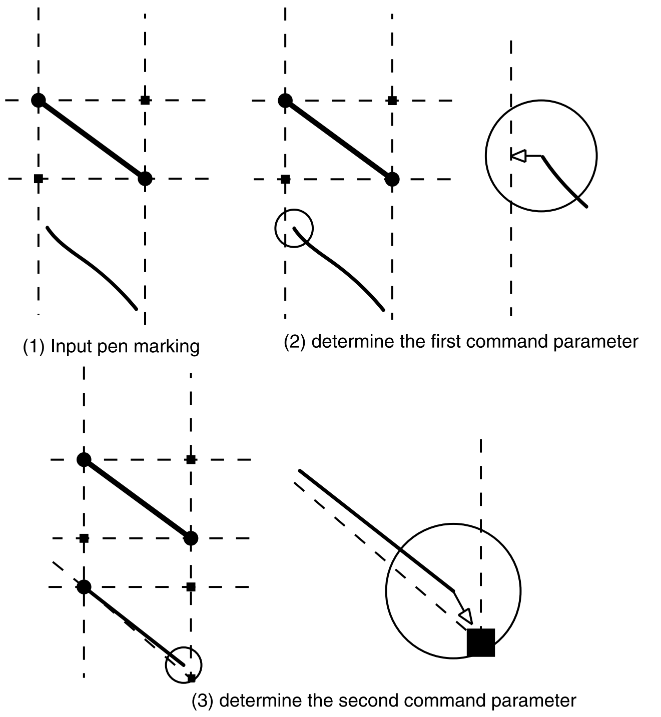

(1) Input pen marking
(2) determine the first command parameter
(3) determine the second command parameter

Fig. 14. Inferencing command parameters of a *straight-line* using snap-dragging. (1) User inputs a pen marking, which is interpreted as a *straight-line*. (2) One end point is attracted to an alignment line. (3) The other is attracted to an intersection point between two alignment lines.

objects. Fig. 12(a) shows several pen markings drawn and (b) shows results after the interpretation and inference without further user intervention.

In our approach, notably, the inference uses Bier's snap-dragging [1]. Because his approach does not support automatic generation of alignment objects to which the cursor is attracted we established some rules for the generation of alignment objects. The alignment objects can be one of the following: (1) axis-aligned lines emanating from the points, which are recently referenced during manipulation (e.g. touching the points), (2) lines perpendicular (parallel) to the recently referenced line objects, and (3) intersection points between the line objects or between the line objects and faces. We use a hash table to store the recently referenced objects. See Fig. 13.

With the aid of these alignment objects, the feature points detected from the segmentation are used as if they are the cursor; in snap-dragging [1], the cursor is attracted to objects (including the alignment objects). Fig. 14 illustrates the inference process. The resulting 3D points (snap results) replace the command parameters. For the curved objects, like splines, the curve points are attracted to the 3D points.

Since our interaction framework provides a safe means for many inference algorithms, other's approaches can be used as well, such as [13,14,5], if they can be made suitable for 3D space.

4.4. Drawing operations with selective-manipulation

We now consider how to draw curve primitives (e.g. circular arc, line, etc.), some of which can be connected to form a face. Fig. 15 shows what the system interprets from the user's pen marking. Six primitives are supported: a line, a rectangle (regular polygons), a circle, a circular arc, a polyline (polygon if closed), and a freeform curve.

Each pen marking that falls into selective-manipulation is interpreted as one of the above drawing primitives. For any non-identical interpretation, the system needs to figure out command parameters, τ (Fig. 16).

5. Implementation, analysis and evaluation

We now discuss the implementation issues of our sketch system and also present user testing results that we conducted to verify the effectiveness of the system.

5.1. Implementation

We implemented the sketch system described using the Visual C programming language and tested on a Fujitsu

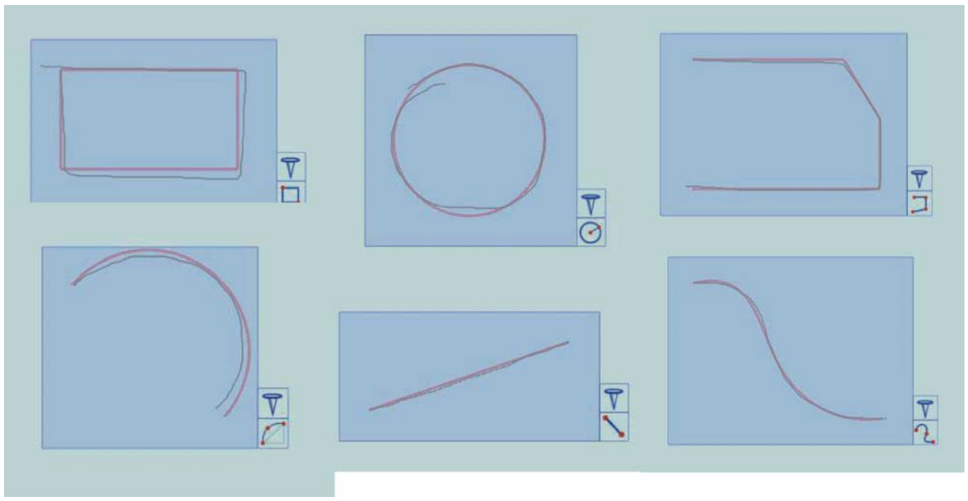


Fig. 15. Pen markings and their interpretations as primitive elements. From left to right and top to bottom, a rectangle, a circle, a polyline, a circular arc, a line, a freeform curve are inferred from the given pen markings, respectively.

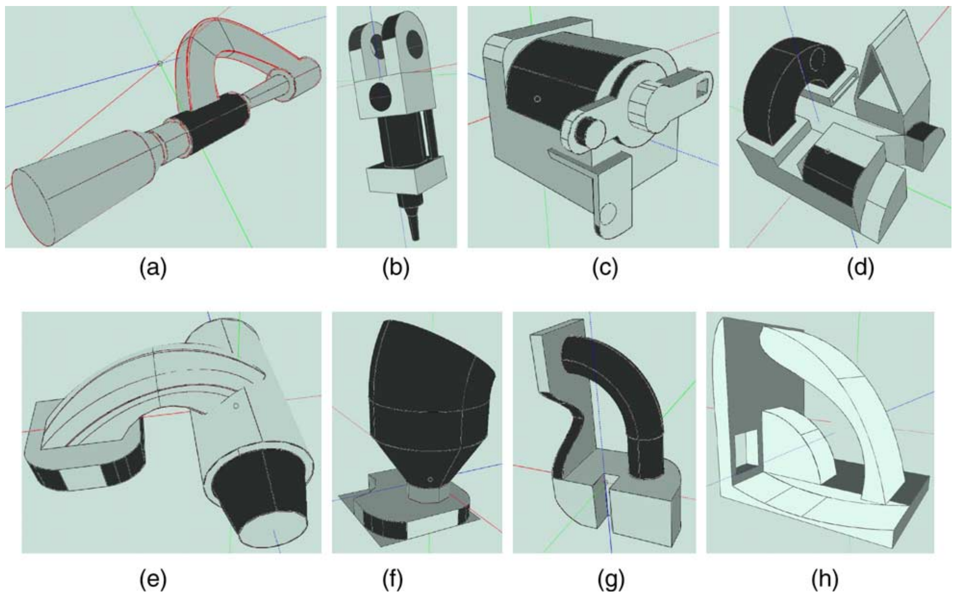


Fig. 16. Sketching results of various shapes using our system. All of these complex models are sketched in less than 5 min by novice users.

1.4 GHz Tablet PC running under the Windows XP Tablet PC Edition. We adopted Weiler's radial-edge representation [32] to handle a non-manifold geometric boundary that may appear during sketching. To automatically detect the filling and splitting of a polygon from the user's pen markings, we took advantage of the planar map algorithm, a well-known result presented by [7] (Fig. 12).

5.2. Usability tests for the interaction framework

We performed two usability tests: one to check the learning time and the other to check its usability for experienced users. For the tests to be more quantitative, we gave the users the target model they have to model; as for free modeling, the analysis of the final model and time to reach the level cannot show quantitative results but qualitative. The target model has been chosen such that the target model requires diverse modeling functionalities (i.e. basic primitives, boolean operations) and repetitive modeling operations that could result in frequent errors. Since fine freeform models such as the bicycle body can be achieved by neither using our system nor using other sketch based modeling systems, we did not undergo the test.

For the first test, we selected 20 subjects from industrial design department and divided them into two groups and gave them a task to build a LEGO block (Fig. 17). The first group used our modeling system and the other a commercial solid modeling tool; none of the subjects have used the tools before. A professional designer with much experiences with the commercial solid modeling tool was assigned to the second team for teaching and one author for the sketching system.

Prior to the test, the two tutors planned their tutorial, especially to define what modeling operations are to be taught to complete the task, and made user manuals designed for the task. To teach the first group, the teacher showed them how to use the needed operations on the tablet PC for some time and for further reference distributed the printed user manuals. To teach the second group, the teacher demonstrated

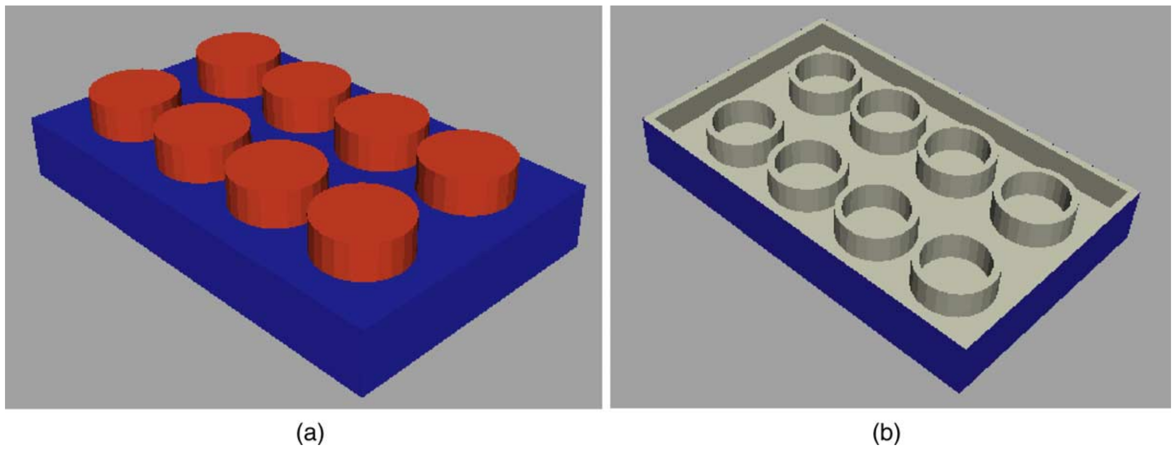


Fig. 17. An image from a LEGO block model given to the test subjects to draw: (a) an image for the top part and (b) an image for the bottom part.

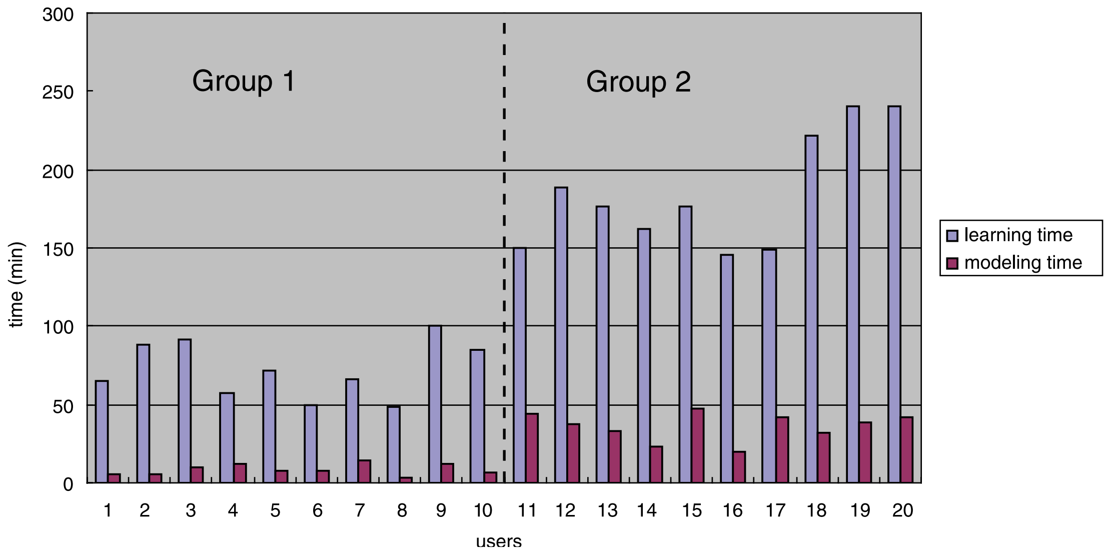


Fig. 18. Statistics for the first test.

the commercial tool on a Desktop PC with mouse and distributed the printed user manual too. After the tutorial, we let the subjects to practice until they feel they can build the model. During the practice, the tutors were sitting in the room to be available for their questions and possibly teach them more if needed. When a subject volunteer to take the test of building the model from the scratch, one of many assistants supervised the test: pass or fail. The subject who failed the test went back to practice and learn the tool more. For the evaluation, we measured two: (1) Time from the beginning of the tutorial to the point when each subject passes the test, which we call *learning time*, and (2) the time needed to build the model during the test for each subject, which we call *modeling time*.

About the learning time (shown with blue color bars in Fig. 18), our system showed better performance (in average 256%). About the modeling time (shown with red color bars), our system showed almost 400% better performance. One reason to this difference is that the second group was not taught how to make full use of the tool, e.g. using short-cut keys; again then, this will take additional time for teaching and remembrance. There was one thing that disturbed the first group; our system often crashed, some of the users had to restart the program, which also takes some time to get used to the system. Since the target shape was designed considering the limit of the modeling functionalities of our system as well as the amount of the time given for the testing (e.g. we could not simply hold the subjects for half a day or so in one place), we confess, this test was not very fairly set for the commercial system.

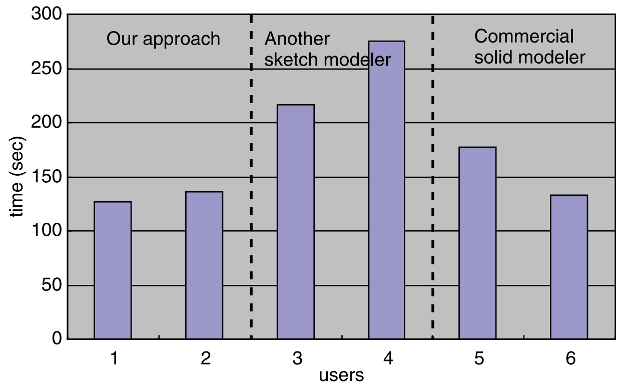


Fig. 19. Statistics for the second test.

For the second usability test, we invited two designers who have used the solid modeling tool for years and two students attended the test as the experienced users for sketch based modeling systems (ours and Zeleznik et al.'s system). For the very experienced users, there is no big difference in completing the modeling task, however, as we saw in the first user test, the learning time differs (Fig. 19).

6. Conclusions and future work

We have presented an interaction framework for sketch-based shape modeling on the pen-input displays. It encases two different interaction techniques, namely, immediate- and selective-manipulation. These interaction techniques alleviate the memorability problem of existing multiple-stroke based sketching systems, and can create diverse 3D shapes that might be otherwise limited by single-stroke pen markings of our approach. Although the idea inserting a further interaction step to sketching is not new, to all our knowledge, we were the first to elaborate upon the idea to be used in a 3D modeling environment.

Our major interaction metaphor for the single-stroke pen marking was an 'arrow' which has been extensively used in many sketching practices. The metaphor was suitable for the generative modeling paradigm. We also have carried out user experiments to benchmark our shape modeling framework, and the participated subjects have shown a steep learning curve to sketch complex 3D shapes using pen-input displays.

Nonetheless our approach still presents many limitations; our system is specially devised for rapidly sketching CAD-favored B-rep models, not other types of generic models (e.g. blobby objects and sophisticated freeform surface models). Moreover, although we targeted the system that can be out for real practice, the system is not quite stable yet. To be useful for practical purposes, first of all, this should support undo/redo function which is a fundamental one for modeling. While the user tests, this lack of useful functions did not make good impression for the users who expected them.

Supplementary Data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.cad.2005.10.007

References

[1] Bier Eric A. Snap-Dragging: Interactive Geometric Design in Two and Three Dimensions. PhD thesis, Department of Computer Science, University of California, Berkeley; 1988.
[2] Bimber Oliver, Encarnacao LMiguel, Stork A. A multi-layered architecture for sketch-based interaction within virtual environment. Comput Graph 2000;(24):851–67.
[3] Buxton William. Chunking and phrasing and the design of human-computer dialogues. Inf Process 1986.
[4] Davis R. Position statement and overview: sketch recognition at mit. In: American Association for Artificial Intelligence Spring Symposium on Sketch Recognition; 2002.
[5] Eggli Lynn, Hsu Chingyao, Bruederlin BeatD, Elber Gershon. Inferring 3d models from freehand sketches and constraints. Comput Aided Des 1997;29(2):101–12.
[6] Foley JD, van Dam A, Feiner SK, Hughes JF. Computer Graphics: Principles and Practice. Reading, MA: Addison-Wesley; 1996.
[7] Gangnet Michel, Herve Jean-Claude, Pudet Thierry, Van Thong Jean-Manuel. Incremental computation of planar maps. Technical Report 1, Digital Equipment Corporation; May 1989.
[8] Gribnau Maarten W. Two-handed interaction in computer supported 3D conceptual modeling. PhD thesis, Delft University of Technology, Netherland; 1999.
[9] Gross MD, Do EY. Ambiguous intentions: a paper-like interface for creative design. In: Proceedings of ACM Symposium on User Interface Software and Technology. ACM; 1996. p. 183–92.
[10] Tracy Hammond, Randall Davis. Ladder: a language to describe drawing, display, and editing in sketch recognition. In: Proceedings of IJCAI (International Joint Conference on Artificial Intelligence); August 2003.
[11] Hong Jason, Landay James, Long Chris, Mankoff Jennifer. Sketch recognizers from the end-user's, the designer's, and the programmer's perspective. In: In Proceedings of AAAI 2002 Spring Symposium (Sketch Understanding Workshop). American Association for Artificial Intelligence; 2002.
[12] Hwang T, Ullman D. Recognize features from freehand sketches. ASME Comput Eng 1994;1:67–78.
[13] Igarashi T. Interactive beautification: a technique for rapid geometric design. In: UIST'97; 1997. p. 105–14.
[14] Igarashi T, Hughes JF. A suggestive interface for 3d drawing. In: UIST; 2001.
[15] Igarashi T, Matsuoka S, Tanaka H. Teddy: a sketching interface for 3d freeform shapes. In: SIGGRAPH; 1999. p. 409–61.
[16] Kim Dae Hyun. A sketch-based modeling interface for pen-input displays. PhD thesis, University Bremen, Shaker Verlag, ISBN 3-832203121-8; June 2004.
[17] Axel Kramer. Translucent patches-dissolving windows-. In: UIST'94; November 1994. p. 121–30.
[18] Landay James A, Hong Jason I, Klemmer Scott R, Lin James, Newman Mark W. Informal puis: no recognition required. In: AAAI 2002 Spring Symposium (Sketch Understanding Workshop). American Association for Artificial Intelligence; 2002.
[19] Landay James A, Myers Brad A. Interactive sketching for the early stages of user interface design. In: Proceedings of CHI '95: Human Factors in Computing Systems; May 1995. p. 43–50.
[20] Landay James A, Myers Brad A. Sketching interfaces: toward more human interface design. Computer 2001;March:56–64.
[21] Lim SW, Lee BS, Duffy HB. Incremental modelling of ambiguous geometric ideas (i-magi): representation and maintenance of vague geometry. Artif Intell Eng 2001;15:93–108.
[22] Long Jr Allan Christian, Landay James A, Rowe Lawrence A. PDA and gesture uses in practice: Insights for designers of pen-based user interfaces. Technical report, Dept of EECS, U.C. Berkeley; 1997.
[23] Mankoff J, Hudson SE, Abowd GD. Interaction techniques for ambiguity resolution in recognition-based interfaces. In: Proceedings of UIST'00; 2000. p. 11–20.
[24] Mase Jitsuro. Moderato: 3d sketch cad with quick positioned working plane and texture modeling. In: Rhyne T-M, Chalmers A, editors. eCAADe2000, June. Germany: Bauhaus-Universitat Weimar; 2000.
[25] Qin SF, Wright DK, Jordanov N. From on-line sketching to 2d and 3d geometry: a system based on fuzzy knowledge. Comput Aided Des 2000; 32:851–66.
[26] Raskin Jef. The Humane Interface: New Directions for Designing Interactive Systems. Reading, MA: Addison Wesley; 2000.
[27] Rubine D. Specifying gestures by example. In: Computer Graphics (SIGGRAPH'91). ACM; March 1991. p. 329–37.
[28] Rubine D. Combining gestures and direct manipulation. In: CHI'92. ACM; May 1992. p. 659–60.
[29] Snyder JM, Kajiya JT. Generative modeling: a symbolic system for geometric modeling. In: SIGGRAPH 1992 Proceedings. ACM; July 1992. p. 369–78.
[30] Trinder M. The computer's role in sketch design: a transparent sketching medium. In: Computers and Building, Proceedings of CAAD futures 99, Atlanta; 1999. p. 227–44.
[31] Wacom. 2002. http://www.wacom-components.com/english/tablet_pc.asp.
[32] Weiler K. The radial-edge structure: a topological representation for non-manifold geometric boundary representations. In: Geometric Modelling for CAD Applications. New York: North Holland; 1988. p. 3–36.
[33] Zeleznik RC, Herndon KP, Hughes JF. Sketch: an interface for sketching 3d scenes. In: SIGGRAPH96. ACM; 1996. p. 163–70.