ILoveSketch: As-Natural-As-Possible Sketching System for Creating 3D Curve Models

Seok-Hyung Bae, Ravin Balakrishnan, Karan Singh
Department of Computer Science, University of Toronto
shbae | ravin | karan @dgp.toronto.edu

ABSTRACT
We present ILoveSketch, a 3D curve sketching system that captures some of the affordances of pen and paper for professional designers, allowing them to iterate directly on concept 3D curve models. The system coherently integrates existing techniques of sketch-based interaction with a number of novel and enhanced features. Novel contributions of the system include automatic view rotation to improve curve sketchability, an axis widget for sketch surface selection, and implicitly inferred changes between sketching techniques. We also improve on a number of existing ideas such as a virtual sketchbook, simplified 2D and 3D view navigation, multi-stroke NURBS curve creation, and a cohesive gesture vocabulary. An evaluation by a professional designer shows the potential of our system for deployment within a real design process.

ACM Classification: H.5.2 [Information Interfaces and Presentation]: User Interfaces–Interaction styles, User-centered design; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling–Geometric algorithms, languages, and systems, Modeling packages, Splines; I.3.8 [Computer Graphics]: Applications

General terms: Algorithms, Design, Human Factors

Keywords: Product design, Sketch-based modeling, 3D curve, Axis widget, Sketchability, Implicit mode change

INTRODUCTION

A significant portion of early conceptual design relies on traditional 2D drawing techniques. Modelers facilitate the leap from these 2D design visions to 3D digital models used in downstream manufacturing processes by using engineering-focused surfacing programs. This is a well-known bottleneck impeding frequent design iteration.

Given that product designers are trained to rapidly explore ideas on paper by sketching perspective line drawings [27, 35, 18], sketch-based 3D curve modeling is an active area of research [10, 30, 4, 42, 24, 21, 22, 23, 32, 5]. None of the existing approaches, however, have made much inroads in real conceptual product design. Reasons include the focus on specific curve interaction techniques rather than the overall user workflow, the poor control and fairness of the resulting 3D curves, and perhaps most importantly that they place the user within a virtual 3D scene without appropriate transitions from familiar freeform 2D sketch practices.

In this paper we present a system, ILoveSketch (Figure 1), which makes a more judicious leap from 2D to 3D. We first consider the designer's need to work strictly in 2D with a virtual sketchbook. We support multi-stroke curve sketching, with simple 2D control of virtual paper orientation and scale. We then introduce the designer to a 3D world with view navigation that complements the 2D controls. A suite of 3D curve creation tools based on sketched input provide designers with a rich palette of possible shapes. Our main contribution is a holistic system that enables professional product designers to create complicated 3D curve networks while maintaining a consistent sketch-like workflow. Component-wise, we make the following contributions:

- New features: axis widget, sketchability-based automatic view rotation, and implicit changing of curve creation method.
- Improved features: virtual sketchbook, multi-stroke curve inking, cohesive gesture vocabulary, and 2D/3D navigation for 3D curve sketching

RELATED WORK

Curve creation from digitized pen strokes is a quintessential aspect of any sketch-based modeling system. Baudel [6] proposed oversketching for local spline curve editing. Bae et al. [3] fit Bezier curves to the weighted average of multiple strokes. Kara et al. [23] used principal component analysis to reorder sample points from multi-strokes for cubic B-spline approximation. Schmidt et al. [37] suggested automatic gap-filling and smoothing from multiple disjoint strokes. Our approach is a variant of Bae et al. [3], where passing time both reduces the weight of old strokes and

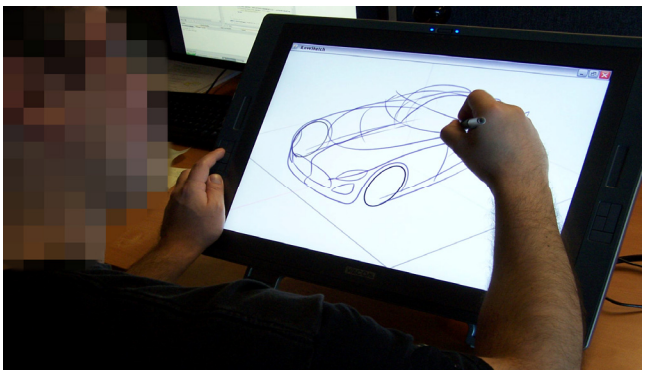

Figure 1: Professional product designer working with ILoveSketch.

UIST'08, October 19–22, 2008, Monterey, California, USA.

finalizes the curve in a manner similar to ink drying. We also seamlessly incorporate Baudel's oversketching for editing finished curves.

A 2D stroke sketched on the view plane sweeps a stroke-surface in 3D defined by rays from the eye-point through points on the 2D curve. Additional information is needed to uniquely determine a 3D curve on this stroke-surface. A common technique is to specify a sketch surface, on which this 2D stroke is projected [30, 22, 32, 16, 17, 42]. In other words, the 3D curve is the intersection curve of the sketch surface and the stroke-surface. Two 2D strokes drawn from different viewpoints can similarly define a unique 3D curve as the intersection of the two stroke-surfaces [24, 23]. Even from a single view, 2D strokes can define 3D curves with additional geometric constraints like shadows [10], symmetry [4, 5], and energy minimization between anchor points [22, 23]. We draw from this work a complementary set of techniques: sketching on orthographic planes or extruded surfaces, epipolar sketching using two views, and symmetric strokes in a single view.

Disambiguating gesture strokes from draw strokes is a challenge of gesture-based interfaces. Existing solutions include mode switching buttons [28], gesture delimiters such as a tap [44] or double tap [31], pressure [34, 42], and press-and-hold [9]. Saund and Lank [36] suggested inference-and-mediation. We minimize mode switching by defining a clear workflow and gesture stroke set whose shapes are distinct from the sketch strokes used for product design.

Selection-and-action phrasing techniques, such as lasso-and-action [25, 31], or lasso-and-menu [25, 19, 2], avoid going back and forth between the workspace and GUI components. We employ a similar phrasing to select a curve or a point-on-curve and choose a sketch surface from among seven options efficiently using an axis widget.

DESIGN GOALS

Our chief design goal is to wean designers from physical pen and paper with a similar 2D sketching interface and subsequently ease them into a 3D environment where their 2D sketches transition into 3D models used in product design. Based on years of on-site field experience in automotive design and developing real 3D modeling systems in industry, we indentified design goals that a successful system must meet from both a fluid workflow and design throughput perspectives:

- Visually smooth complex curves for design exploration
- Minimal interruption to sketching by GUI and gestures
- Minimal set of gestures with intrinsic affordances
- Immediate and easy access to 2D/3D navigation
- Dynamic information display to assist 2D/3D sketching
- Focus on geometric objects rather than UI components

SYSTEM OVERVIEW

The basic "feel" of the system borrows from that of a physical paper sketchbook. Designers working with real paper often flip and tear pages to get between sketches, and turn and move the sketchpad around for easier access to various parts of a particular sketch. Our system thus uses a virtual sketchbook metaphor with support for several page navigation interactions: tearing, peeling, panning, zooming, and rotation. We also support automatic dynamic rotation of the virtual sketchbook based on the users' input strokes to make further multi-stroke sketching biomechanically comfortable.

User input for curve creation is primarily by repeated sketching. Prior observations indicate that professionals want full control of the final shapes of 2D/3D curves while leveraging their sketching skills. We use a multi-stroke *pentimenti* style curve sketching approach with an ink drying visualization that allows users to sketch uninterrupted and watch curves continuously settle into their desired shapes. We provide five different 3D curve sketching methods along with the notion of *sketchability* – a view-dependent scalar measure that helps determine how good a given viewing angle is for a given 3D curve sketching method. Sketchability-based automatic 3D rotation increases a designer's throughput, by reducing the need for explicit 3D navigation to find a suitable view in which to sketch. In addition, an *axis widget* helps define sketching surfaces. We use transparency and other visual cues such as highlighting the intersection points of a sketch surface and existing 3D curves, to aid the users' understanding of the 3D scene. A minimal gesture set is used to provide command input, and audio feedback is used to support gesture confirmation. As a whole, these methods result in a coherent 3D curve sketching workflow that does not rely on menus, icons, or tool palettes that could clutter the screen. Thus, the user's focus of attention can stay on the artwork at all times.

GESTURE SET

We use a minimal set of pen gestures (Figure 2) with a strong semantic correspondence to the associated operation(s). The period (tap) gesture (Figure 2a) is a delimiter for immediately committing an action [44]. As in many sketch-based systems, the scratch-out gesture (Figure 2b) deletes geometric entities. The scratching extent defined by the number of stroke cusps is used to trim part of a curve or delete it entirely [5]. The roll gesture (Figure 2c) is used for undo/redo. The small lasso gesture (Figure 2d) is used for selecting a 3D curve and activating an axis widget. The span (perp) gesture (Figure 2e) and flick gesture (Figure 2f) crossing the axis widget are used to define the sketching surface onto which subsequent 2D strokes are projected to create 3D curves. These gestures are mostly distinct in shape and thus easily disambiguated from relatively large and smooth design sketch strokes [41]. The slightly less distinct span and flick gestures, which are orientation-sensitive, are inferred by context and applied only in the selection-and-action phrase following the lasso gesture.

Figure 2: Pen gestures: (a) period (tap), (b) scratch-out, (c) roll, (d) small lasso, (e) span (perp), (f) flick.

PEN-AND-PAPER LIKE INTERACTION

We now describe in detail the aspects of our system that support 2D sketching.

Virtual Sketchbook

Sketches can be saved by virtually flipping over sheets of "paper" (Figure 3a,b), or removed by tearing them (Figure 3a,c). Prior designs can also be recalled by similarly leafing through the sketchbook. The user can visually go through previous works and open one among them. Our implementation is an application of paper folding and peeling techniques [11, 7] that were developed originally for window management. The order of storing sketches is shown in Figure 3d.

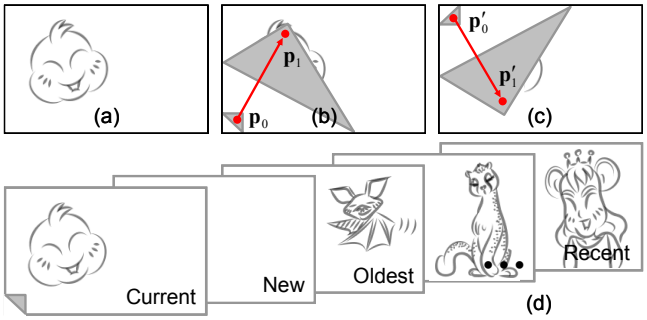

Figure 3: Virtual sketchbook: current sketch (a) is saved by turning it back (b) or removed by tearing it (c), order of pages (d).

Multi-Stroke 2D NURBS Curve Sketching

Product designers train to draw from the shoulder. They draw curves using simple smooth strokes, very lightly at first, darkening them as the curves form a satisfactory shape [41]. Bae et al. [3] capture this multi-stroke workflow digitally by fitting each stroke using a cubic Bezier curve (the lower curve in Figure 4a is drawn first; the upper curve second). Designers can correct subsequent strokes using the visual feedback of previous stroke curves and the estimated weighted average curve (the middle curve in Figure 4a), and finalize the curve by pressing a keyboard button. Intermediate strokes disappear after a curve is finalized. Repetitive stroking allows designers to control the shape of simple curves perfectly, but pressing the button every time can be cumbersome and easy to forget when their focus of attention moves to a new curve subconsciously. To provide a more fluid workflow, we introduce an inking metaphor for curve finalization. A lapse of time after a stroke is drawn (we use 2×time between the last two strokes) automatically finalizes the current curve (Figure 4b,c). We animate the estimated weighted average curve from pale-violet red to black, informing the user of passing time as the ink on the curve dries. Users can correct their curves before the 'ink' dries. This time-out based curve finalization can give the user a rhythm of drawing, and increase work speed by allowing consecutive curve drawing without pausing the pen because moving to a new curve usually takes enough time for the ink to dry. Alternatively, users can use a period gesture for immediate curve finalization.

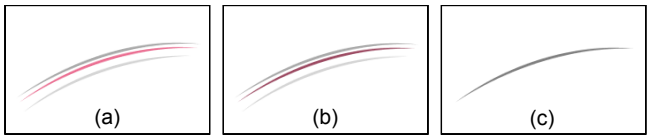

Figure 4: Simple spline curve creation based on multi-stroke sketching with a drying ink metaphor.

Complex curves with inflection points and sharp corners are created by connecting simple smooth curves. Strokes drawn over the overlapped part of two tangent curves make them connected smoothly, reflecting geometric constraints [14] as in physical inking and erasing [29] (Figure 5).

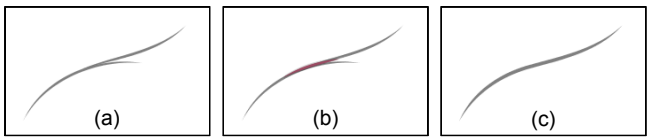

Figure 5: Tangential curves joined by repair strokes.

If a new settled curve starts tangential to an existing curve, the system proceeds one step further – the two curves are smoothly connected automatically [6] (Figure 6a-c). The user can go back to the intermediate result by applying the counterclockwise roll gesture once (Figure 6d,e), or the initial state by applying the gesture again (Figure 6e,f). This *one-step-further* and *roll-back* is arguably more usable than suggestive interfaces [20] or inference-and-mediation [36] since the user intent is explicit and unambiguous.

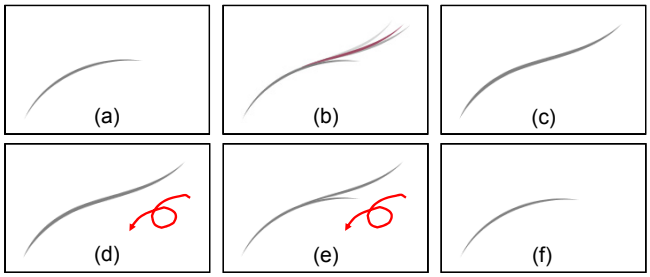

Figure 6: Curve connection (a)-(c), roll-back (d)-(f).

Paper Navigation

In pen-and-paper sketching, drawing and paper rotation are subconsciously integrated activities [13]. Good paper orientation not only provides designers with articulation comfort, but also enhances quality of sketched curves. A number of current painting programs support paper rotation, but designers rarely use it because they do not want to sacrifice workflow continuity for relatively unsatisfying paper rotation techniques. Such paper rotation seems more dispensable than explicit zoom and pan, which are needed for access to all parts of the sketch. Designers often prefer to rotate their head or twist their body to keep a comfortable kinematic chain for drawing.

To encourage users to rotate the 2D canvas so that they draw more comfortably and create better quality curves, we combine rotation with zoom into a single action (Figure 7d,e), borrowing from bimanual interaction techniques [26] (Figure 7a,b). The zoom-and-rotation center is set near the upper left corner of the screen, and marked with a cross-

hair (Figure 7d,e). A simple paper orientation indicator is shown as a gray triangle at the corner of the screen (Figure 7e). Following the experimental results of Li et al. [28], the combined zoom-and-rotation is performed in the quasi mode invoked by pressing a button on a graphics tablet with the non-preferred hand. With an additional modifier button pressed, the user can pan the virtual sketchpad with an action similar to that of moving a sheet of paper in the second-level quasi mode (Figure 7c,f).

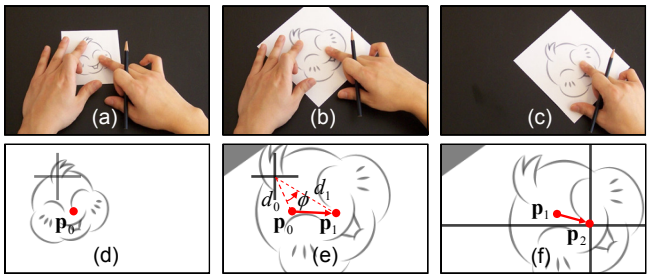

Figure 7: Paper object manipulation concept on table display (a)-(c); paper navigation: (d) initial state, (e) combined zoom-and-rotation around pre-defined pin location (d_1/d_0: zoom factor, ϕ: rotation angle), (f) pan.

Due to biomechanical constraints, there exist certain directions and sizes of curves that designers feel more comfortable drawing [43] (Figure 8a,d,g). We assume a pause in drawing after the first stroke of a multi-stroke curve as a cue of an uncomfortable paper orientation for sketching the curve, and consider an automatic rotation of the sketchpad towards the optimal orientation based on the length, curvature and orientation of the stroke (Figure 8).

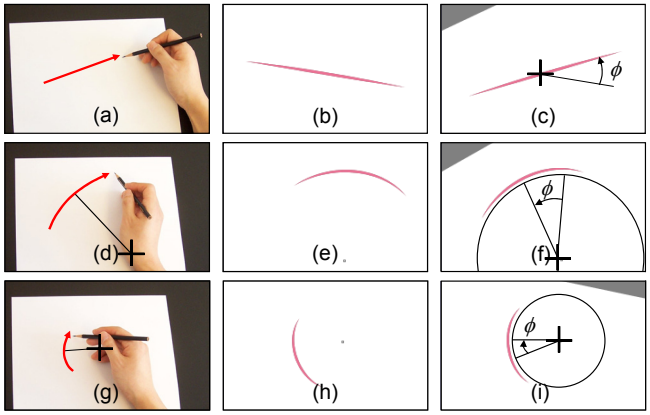

Figure 8: Comfortable drawing directions for strokes (a), medium arcs (d), small arcs (g); automatic paper rotation for strokes (b)-(c), medium arcs (e)-(f), small arcs (h)-(i) (ϕ: rotation angle).

3D CURVE SKETCHING

We now address the transition of our system from 2D sketching to the creation of 3D curves in a virtual 3D scene.

Space Navigation

We choose an object-centric paradigm for space navigation which is appropriate for modeling purposes. Observe that paper navigation already allows the user sketch access through panning and the combined zoom-and-rotation. The only remaining aspects of navigation necessary for design are 3D rotation out of the view plane (accomplished successfully by trackball manipulation) and a change in perspective to provide a better sketch understanding through foreshortening. Consistent with paper navigation a button on the graphics tablet activates the 3D quasi mode for a virtual trackball manipulation (Figure 9a,b). Perspective control is usually coupled with object magnification by dollying the camera towards or away from the object. The design intent here is simply providing alternate depth understanding independent of size. We thus implement a combined dolly-zoom (Figure 9c) operation as in [40] that alters perspective while preserving object size, activated in the another second-level quasi mode by pressing the same modifier button used for paper navigation. Note that only three buttons are used with the user's non-preferred hand for controlling all 2D/3D viewing parameters.

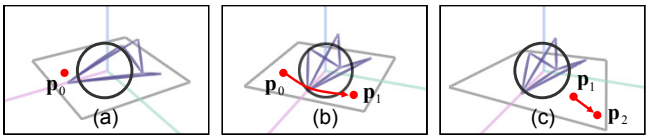

Figure 9: Space navigation: (a) initial state, (b) tumble (trackball), (c) dolly-zoom.

User Interaction of 3D Curve Sketching Methods

To design a coherent set of 3D curve sketching methods, we first classify them into two categories based on user input:

Two-curve methods: the user draws two curves to uniquely situate a 3D curve in space.

- *Single-view symmetric epipolar method*: the user sketches a pair of symmetric curves with respect to a given center plane from a single view point (Figure 10). This is fast and closest to traditional sketching as many pairs of 3D curves can be created from a single view point. This method is, however, targeted at expert designers adept at drawing accurately in perspective.
- *Two-view epipolar method*: the user sketches a curve from one view point, then, draws it from a different view point (Figure 11). Without the aid of epipolar lines of the first view and existing curves in 3D, this is difficult even for experts but gets easier as more curves are drawn to help situate the sketch in arbitrary views.

Sketch surface methods: once a 3D sketch surface is defined, 2D strokes project on it uniquely to form 3D curves. We look at three potentially useful sketching surfaces.

- *Orthographic plane method*: designers specify a 3D point and a principal orthographic plane through it (Figure 12).
- *Orthographic extruded surface method*: designers specify an extruded surface by selecting a 3D curve and a principal axis of extrusion (Figure 13).
- *Oblique extruded surface method*: designers specify an extruded surface by selecting a 3D curve and an arbitrary 3D axis of extrusion (Figure 14).

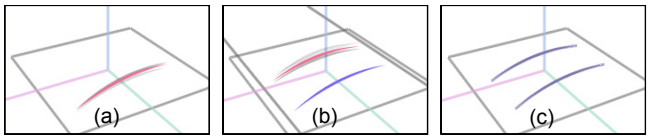

Figure 10: Single-view symmetric epipolar method: (a) 1st curve sketch, (b) 2nd curve sketch with the help of two receding lines to the vanishing point to which all the lines parallel to y-axis appear to converge (x, y, z-axes are in red, green, blue; zx-plane is the plane of symmetry), (c) a pair of resulting 3D symmetric curves.

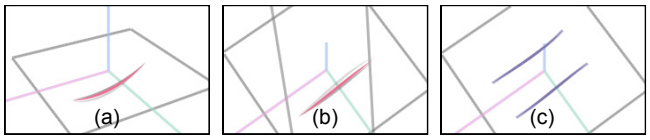

Figure 11: Two-view epipolar method: (a) curve sketch in one view, (b) curve sketch in other view with the help of two epipolar lines passing through the end points of the 1st curve from the 1st viewpoint, (c) the resulting 3D curve (lower right. c.f. the upper left 3D curve is created by simply reflecting the resulting 3D curve with respect to the plane of reflection).

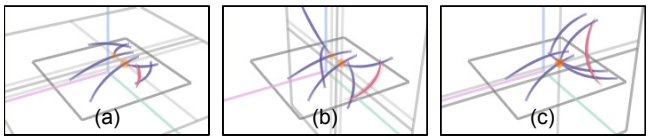

Figure 12: Orthographic plane method: sketch 3D curves on (a) xy-plane, (b) yz-plane, (c) zx-plane.

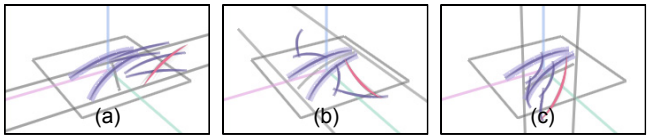

Figure 13: Orthographic extruded surface method: sketch 3D curves on (a) x-dir extruded surface, (b) y-dir extruded surface, (c) z-dir extruded surface (the generator curve of each surface is depicted as thickened in pale slate blue color).

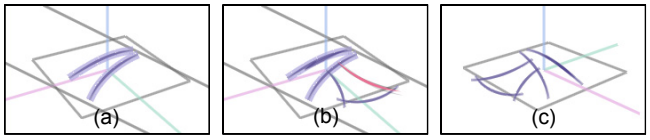

Figure 14: Oblique extruded surface method: (a) an extruded surface along an arbitrary direction, (b) sketch curves on the surface, (c) a scene viewed from on the surface.

There are a large number of 3D curve creation techniques based on a wide variety of geometric constraints not covered in the above list, as well a number of curve shapes that are difficult to achieve with the techniques above, such as helices. We base our choice on the observation that principal axes and symmetry are common guidelines to product design. The set of techniques described above is comprehensive enough to capture the smooth inflectionless 3D curve segments seen in product design, while minimal enough to provide the user with a simple toolset.

Axis Widget

We combine selection-and-action phrasing [8] with crossing [1] to define sketch surfaces using a context-sensitive 3D axes widget [39]. A small lasso gesture on a 3D curve (Figure 15a), selects both the curve and 3D point on it depicted visually by the axis widget (Figure 15b). Designers can now choose a sketch surface method with a single pen gesture crossing the axis widget: the span gesture crossing two axes of the axis widget selects the principal orthographic plane containing the two axes (Figure 16a); the flick gesture extrudes the selected curve along the principal axis most parallel to the gesture direction (Figure 16b); the flick gesture crossing the widget origin defines an oblique axis of extrusion in the view plane (Figure 16c). Users can also select the principal planes at the origin and/or global axes with a span gesture. This is commonly used to sketch on the center plane. The axis widget is simple and large enough to make crossing it easy. Furthermore, its context dependency on the selected object keeps the user focus on the artwork even during 3D sketch surface selection.

Erasing the axis widget is an easy way to escape from the selection-and-action phrase to the default single-view symmetric sketching (Figure 15c). The small lasso gesture can also be used to create curve networks. Performing the gesture over curves whose projections intersect in a given view causes those 3D curves to be tightened to a common average 3D intersection [5], where the axis widget is also positioned. In this case, only orthographic plane options are available due to multiple selected 3D curves.

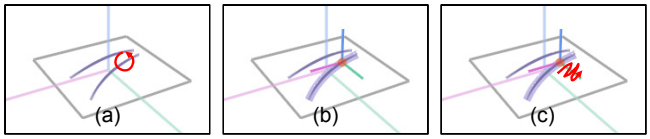

Figure 15: (a) Small lasso gesture applied on a 3D curve, (b) axis widget located on the curve, (c) scratch-out gesture to delete the axis widget.

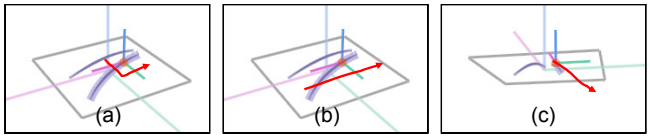

Figure 16: Sketch surface definition: (a) span gesture for orthographic planes, (b) axis-aligned flick gesture for orthographic extruded surfaces, (c) flick gesture starting at the origin of the axis widget to define a direction for an oblique extruded surface.

Sketchability

All our 3D curve sketch methods are based on *epipolar geometry*, where the stroke-surface swept by rays from the eye-point through a sketched 2D stroke intersects with another 3D surface (defined explicitly for sketch surface methods, implicitly by the stroke surfaces of the other stroke in two-view sketching, and the symmetric stroke from the mirrored view in symmetric sketching). Clearly a

good view for sketching the 2D strokes is one where the sketch surface has a large visible projection. Based on this, we define a simple normalized sketchability measure in terms of the shape of the sketch surface and a given view direction (see Appendix for a detailed calculation for each 3D curve creation method). Visualizing sketchability as the opacity of the selected sketch surface provides designers with persistent feedback to aid them in finding good views in which to sketch (compare Figure 17a and Figure 17b).

Implicit Changing of 3D Curve Creation Method

The sketchability of different curve creation methods can be quite different from each other and thus often complementary. Viewing along the y-axis for example, is good for sketching on the zx-plane but bad for the single-view symmetric epipolar method. We considered using this fact to always pick the curve creation method with maximum sketchability for any given view. We found the results to be somewhat overaggressive, unpredictably changing curve creation methods from ones that the designer may have intentionally selected. However, the idea is not without merit. Thus, rather than dynamically selecting the method with maximum sketchability, we do the following: after a view change we evaluate the sketchability of the current selected method and change to the default single-view symmetric method only if sketchability is lower than a specified threshold (Figure 17) – we use 0.2. Another implicit change is based on erasing a current sketch surface (Figure 18). Erasing the sketch surface is useful when the designer wishes to switch to a two-curve method in the current view. Yet another implicit change occurs within two-curve methods. The single-view symmetric epipolar and two-view epipolar methods share the first curve sketch step, but the choice of two-view or symmetric single-view method depends on whether a viewing change takes place prior to sketching the second curve. Figure 19 shows all mode change scenarios.

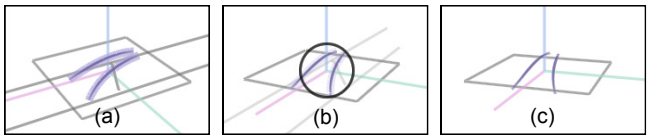

Figure 17: View-based implicit mode change.

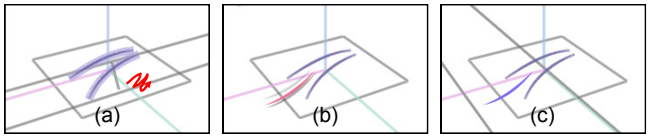

Figure 18: Erasing-based implicit mode change.

Auto-Tumble

Once a sketch surface method is chosen, the system smoothly animates to a new viewing direction to improve the sketchability of the selected surface. Grossman et al. [17] use a return-back animation from a perspective view to the previous orthographic view in digital tape drawing; Owada et al. [33] go from a view for sketching a contour on a 3D surface to a view for sketching a silhouette along which the contour extrudes/sweeps, whereas our auto-tumble follows the gradient of increasing sketchability until a given threshold is exceeded (we use 0.8). Auto-tumble has special importance in the oblique extruded surface method since good views for defining the direction of extrusion are orthogonal to the best view for sketching as in [33].

Audio Feedback

To reinforce the visual feedback of pen gestures, short audio clips are played: a short arrow sound (0.15 s) for the lasso gesture, a long arrow sound (0.3 s) for the span and flick gestures, a camera shutter sound (0.5 s) when 3D viewing motion stops.

IMPLEMENTATION

ILoveSketch is written in Java and runs on a desktop computer (Pentium®4, 2.99 GHz, 2.00 GB RAM) with a WACOM® Cintiq 21UX display tablet.

USER FEEDBACK AND DISCUSSION

It is important to note that ILoveSketch is intended for professional designers who are willing to invest some time learning a system in return for improved workflow later on, rather than for walk-up-and-use casual users. Thus, rather than doing a typical usability evaluation with many non-expert users, we instead invited one professional designer from a major automobile design studio, with an Industrial Design degree, to intensively evaluate ILoveSketch. He has 12 years experience in the automotive, toy, and film industries and has both a traditional design background and a familiarity with most 2D/3D graphics software.

After an one hour instruction session, the designer tried to use our system for 1.5 hours testing every feature while creating a 3D curve model (spider, Figure 20a,b). Once he was accustomed to using our system, he made the second 3D curve model in 0.5 hour (jet fighter, Figure 20c,d). Finally he *designed* a new roadster from scratch in 2.5 hours (Figure 20e,f). In this case, four circles were displayed in the 3D scene at his request because the dimensions of wheelbase, overall width, and outer radius of tires determine the car's size and proportion [27, 35].

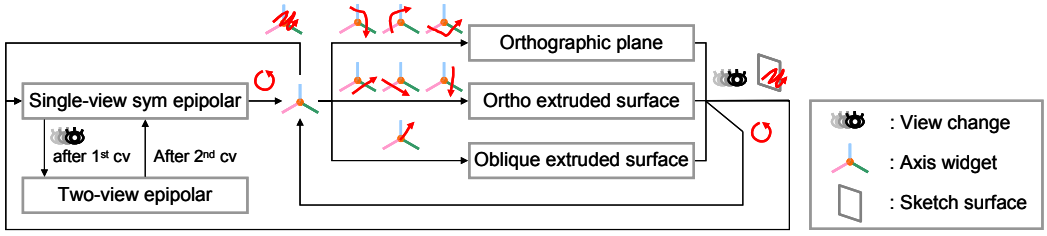

Figure 19: Mode change flow.

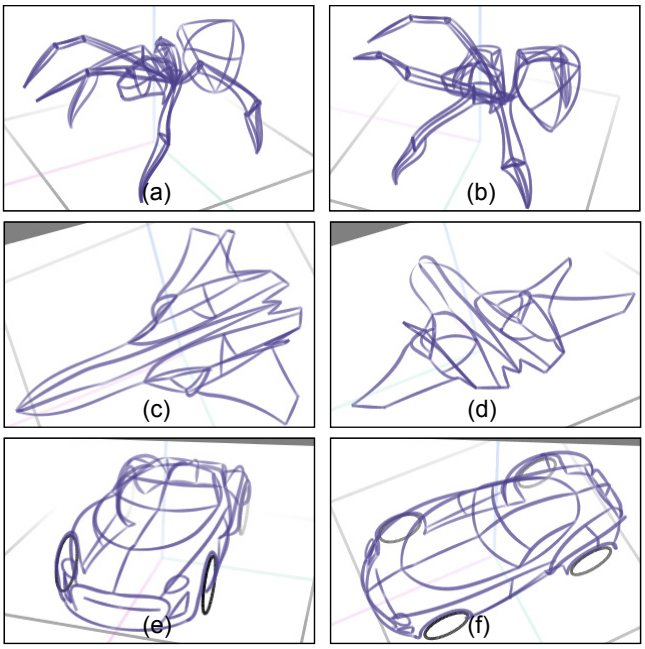

Figure 20: 3D curve models sketched by a professional car designer: spider (a)-(b), jet fighter (c)-(d), roadster (e)-(f).

Table 1: Evaluation by the test designer.

"How much do you like ... ?"	😍	😊	😐	☹	😩
Multi-stroke curve creation	√				
Color animation of drying ink	√				
Period gesture to finalize a curve	√				
Repair strokes for tangential curve connection		√			
Automatic curve connection		√			
Roll gestures for undo/redo	√				
Combined zoom-and-rotation	√				
Paper orientation indicator	√				
Automatic paper rotation					√
Interaction style with the axis widget	√				
Small lasso gesture to activate an axis widget	√				
Span gesture for an orthographic plane	√				
Flick gesture for an ortho extruded surface		√			
Way of selecting an oblique extruded surface			√		
Erasing the axis widget to return to sketching			√		
Sketchability-based implicit mode change	√				
Erasing-based implicit mode change			√		
Implicit tightening of curve intersection points		√			
Sketchability-based sketch surface opacity	√				
Display of {sketch surface ∩ 3D curves}	√				
Auto-tumble	√				
Audio feedback	√				

😍: strongly like, 😊: like, 😐: neutral, ☹: dislike, 😩: strongly dislike.

After all the sessions, he was asked to provide comments and complete a questionnaire (Table 1). His summary evaluation of our system is:

"The tool is effective in creating 3D curves for model building purposes easily and intuitively. Instead of creating and editing hundreds of control vertices to get a curve to look good, it takes much less time to develop a good 3D curve. My ideas were easy to implement, and I spent more time 'designing' and 'thinking' about the 3D shape, and a lot less time trying to guess the result of my idea in 3D space. If simple editing commands to adjust curves easily and real-time rendering function of surfaces interpolating 3D curves are added, this tool will be perfect for product and automotive design."

The designer liked working with ILoveSketch, but was indifferent about a few features (Table 1). He did not like automatic paper rotation because he likes to pause and contemplate his sketch from time to time and unexpected automatic paper rotations can be frustrating. Initially, the timeout for invoking automatic rotation was set to 2×time taken for the first stroke, but we changed it to 3 seconds at his request. Given his traditional sketching habits, it would take more time for him to comfortably adopt the automatic paper rotation protocol. He found the scratch-out gesture for erasing the axis widget and sketch plane intuitive, but noted that it was time-consuming in terms of design throughput. He also pointed out that it took time and effort to define the direction of extrusion of an oblique extruded surface because two steps were required: 1) change from an initial view to the view perpendicular to the extrusion direction, then, 2) change to a view good for sketching.

We analyzed the command execution logs of the designer's 3D curve sketching activity (Figure 21). We found the most frequent navigation command to be tumble. The next frequent ones in order were combined zoom-and-rotate, and pan. Dolly-zoom was used once in a while to check the proportion of designs in different perspectives. He used the period gesture more than automatic curve finalization as he considered work speed more important than convenience. One interesting observation was that the frequency of using each 3D curve sketch method differed according to the characteristics of design shapes (Table 2). In designing the spider whose shape is mostly organic, the single-view symmetric epipolar method was used frequently in the early stages (the dots in the fifth row from the bottom of Figure 21a). Note that this method is useful to situate 3D freeform curves in empty space. Once a number of 3D curves were created, the two-view epipolar method was used a lot while utilizing the spatial information of the intersection points of the first epipolar surface and existing 3D curves. Compared to the jet fighter and roadster models, the spider model was created with all five 3D curve sketch methods going around (Table 2). For the roadster model, the orthographic plane method played an important role in the beginning because

Table 2: Numbers of 3D curves created by the test designer.

Model	1-view sym epipolar	2-view epipolar	Ortho plane	Ortho extruded sf	Oblique extruded sf	Total
Spider	28 (20.00 %)	25 (17.86 %)	41 (29.29 %)	25 (17.86 %)	21 (15.00 %)	140 (100.00 %)
Jet fighter	11 (14.86 %)	1 (1.35 %)	32 (43.24 %)	23 (31.08 %)	7 (9.46 %)	74 (100.00 %)
Roadster	7 (3.14 %)	32 (14.35 %)	70 (31.39 %)	112 (50.22 %)	2 (0.90 %)	223 (100.00 %)

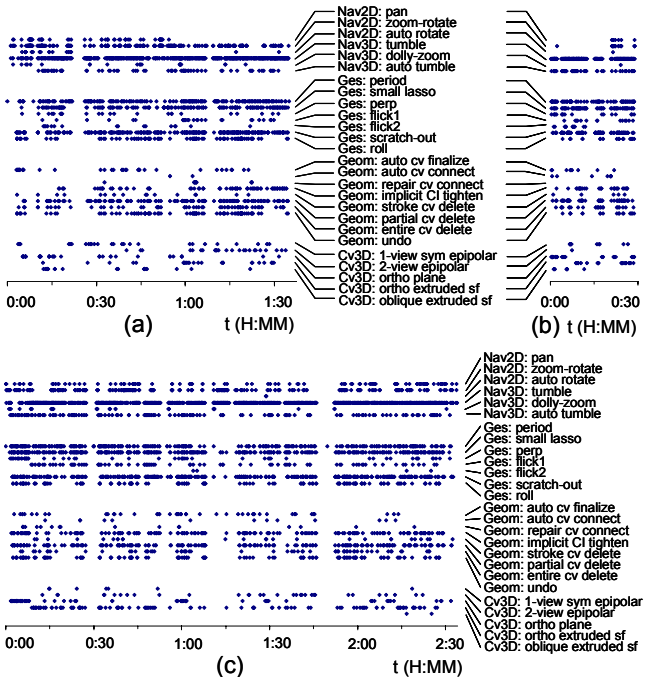

Figure 21: Command execution log plots: (a) spider, (b) jet fighter, (c) roadster.

of the traditional style of car design starting from a side view (the dots in the third row from the bottom of Figure 21c). To fine-tune key curves defining the characteristics of car styling, frequent re-sketching was performed by using the orthographic extruded surface method (Table 2). The various design explorations and deleted curves can be seen as scaffolding [38] and are shown in Figure 22a.

Sometimes, the designer found the sketched 3D curves had deviated from his imagination because of a view with poor sketchability resulted in confusion with neighboring 3D curves. These mistakes were sometimes serendipitous and also helpful in understanding the shapes in 3D space to sketch them better subsequently.

We used a fog effect in rendering 3D curves to provide a depth cue [12]. However, as the numbers of 3D curves increased, it was hard to differentiate between them in a busy scene. More effort is required to handle such visual clutter.

As 2D sketching is the framework for other drawing techniques, 3D curve sketching could be the framework for surface modeling techniques such as FiberMesh [32]. Figure 22b, for example, shows a rendering of the roadster surfaced quickly as a curve network in the modeling and animation system, *Maya*.

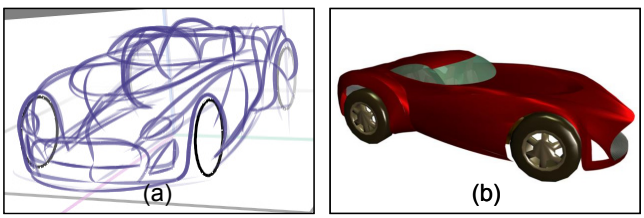

Figure 22: Design exploration: (a) scaffolding, (b) surface model.

CONCLUSIONS AND FUTURE WORK

We designed and implemented a 3D curve sketching system called ILoveSketch by effectively integrating many techniques: some novel ones, some existing ones that are used as it is or modified. An intensive evaluation by a professional product designer showed ILoveSketch to be potentially useful within a real product design process. While we have presented a fairly complete system, this research opens up many avenues for future work. We intend to undertake a formal study of each feature from a performance standpoint, extend the system to allow arbitrary 3D sketch surfaces, combine pen sketching with direct multi-finger touch technologies for camera navigation, and design a user interface for users having medium drawing skills.

ACKNOWLEDGEMENTS

We thank Calen Whitmore for evaluating our system, Eun-Jin Lee for processing execution log data, John Hancock and Noah Lockwood for video production, Tomer Moscovich, Ryan Schmidt, Tovi Grossman, Xiang Cao, and the members of dgp for valuable advices, and all the anonymous reviewers for helpful directions.

APPENDIX

Calculation of Sketchability

Sketchability is a simple ratio of the sum of the apparent areas of two faces of the bounding box of a reference curve to the maximum possible visible area.

Given a selected 3D curve (C^*), let us first consider the orthographic extruded surface along the z-axis (Figure 23a). The axis-aligned bounding box is used in this case. The areas of the two faces of the bounding box whose normal vector is perpendicular to the direction of extrusion are

$$A_u = \Delta V \cdot \Delta W\,,\ A_v = \Delta U \cdot \Delta W\,.$$

The areas of the two faces seen from a viewpoint are

$$A_{u,proj} = \Delta V \cdot \Delta W \left|\mathbf{l} \bullet \mathbf{v}\right|,\ A_{v,proj} = \Delta U \cdot \Delta W \left|\mathbf{m} \bullet \mathbf{v}\right|,$$

where **l**, **m** are the unit normal vectors of the two faces, respectively, and **v** is the view vector calculated as

$$\mathbf{v} = (\mathbf{e} - \mathbf{a}) / \left\|\mathbf{e} - \mathbf{a}\right\|,$$

where **e** is the eye-point, and **a** is the look-at point, which is the center of the bounding box in this case.

By the way, the maximum possible visible area is

$$A_{max} = \sqrt{(\Delta U)^2 + (\Delta V)^2} \cdot \Delta W\,.$$

Then, sketchability is calculated as

$$Sketchability = \frac{A_{u,proj} + A_{v,proj}}{A_{max}} = \frac{\Delta V \left|\mathbf{l} \bullet \mathbf{v}\right| + \Delta U \left|\mathbf{m} \bullet \mathbf{v}\right|}{\sqrt{(\Delta U)^2 + (\Delta V)^2}}\,.$$

Orthographic planes can be also thought of as extruded surfaces. For the zx-plane, choose the x-directional line passing through the selected point ($\mathbf{p}^*$) as the generator curve (Figure 23b). Then, select the extrusion direction to be along the z-axis. As the bounding box of the line has no volume ($\Delta V = 0$), the above equation becomes $\left|\mathbf{m} \bullet \mathbf{v}\right|$.

Table 3: Parameters for calculation of sketchability.

3D curve sketch method	Reference curve	Bounding box	**l**	**m**	ΔU	ΔV	**a**
Single-view symmetric epipolar	C_{Π_M}	Oriented	$\mathbf{i}_1$	$\mathbf{i}_2$	ΔE_1	ΔE_2	$L_{\mathbf{e}_M \to \mathbf{c}_{OBB}} \cap \Pi_C$
Two-view epipolar	C_{Π_1}	Oriented	$\mathbf{i}_1$	$\mathbf{i}_2$	ΔE_1	ΔE_2	$L_{\mathbf{e}_1 \to \mathbf{c}_{OBB}} \cap \Pi_C$
Orthographic plane (xy)	.	.	.	**k**	.	.	$\mathbf{p}^*$
Orthographic plane (yz)	.	.	.	**i**	.	.	$\mathbf{p}^*$
Orthographic plane (zx)	.	.	.	**j**	.	.	$\mathbf{p}^*$
Orthographic extruded surface (x-dir)	C^*	Axis-aligned	**j**	**k**	ΔY	ΔZ	$\mathbf{c}_{BB}$
Orthographic extruded surface (y-dir)	C^*	Axis-aligned	**k**	**i**	ΔZ	ΔX	$\mathbf{c}_{BB}$
Orthographic extruded surface (z-dir)	C^*	Axis-aligned	**i**	**j**	ΔX	ΔY	$\mathbf{c}_{BB}$
Oblique extruded surface	$C_{\Pi_\perp}$	Oriented	$\mathbf{i}_1$	$\mathbf{i}_2$	ΔE_1	ΔE_2	$\mathbf{c}_{OBB}$

C_{Π_M} : reflected sketch curve on the mirrored sketch plane; C_{Π_1} : sketch curve on the 1st sketch plane; $C_{\Pi_\perp}$: projected curve of C^* onto the plane passing $\mathbf{p}^*$ and perpendicular to the extrusion direction; $(\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3)$: axes of the oriented bounding box which correspond to the extents, $\Delta E_1 \geq \Delta E_2 \geq \Delta E_3$; $L_{\mathbf{e}_M \to \mathbf{c}_{OBB}}$: line from the mirrored eye-point $(\mathbf{e}_M)$ passing through $\mathbf{c}_{OBB}$; $L_{\mathbf{e}_1 \to \mathbf{c}_{OBB}}$: line from the 1st eye-point $(\mathbf{e}_1)$ passing through $\mathbf{c}_{OBB}$; $\mathbf{c}_{BB}$: center of the axis-aligned bounding box; $\mathbf{c}_{OBB}$: center of the oriented bounding box.

The single-view symmetric epipolar method is a special case of the two-view epipolar method [15]. Therefore, those two methods have the same geometric interpretation of sketchability calculation (Figure 23c). Refer to Table 3 to find all the parameters for calculation of sketchability.

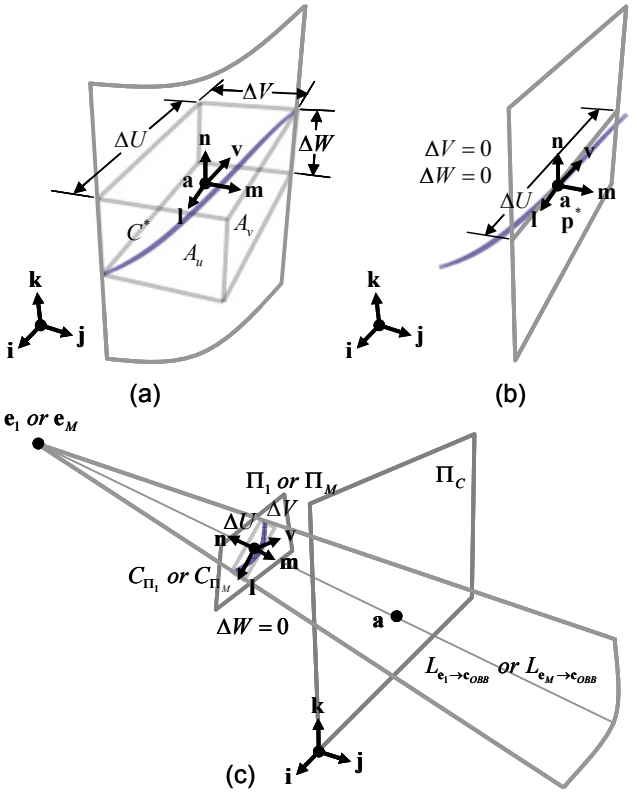

Figure 23: Calculation of sketchability: (a) orthographic extruded surface method, (b) orthographic plane method, (c) two-curve methods.

REFERENCES

1. Accot, J. & Zhai, S. (2002). More Than Dotting the i's – Foundations for Crossing-Based Interfaces. *CHI*, 73-80.
2. Agarawala, A. & Balakrishnan, R. (2006). Keepin' It Real: Pushing the Desktop Metaphor with Physics, Piles and the Pen. *CHI*, 1283-1292.
3. Bae, S.-H., Kim, W.-S., & Kwon, E.-S. (2003). Digital Styling for Designers: Sketch Emulation in Computer Environment. *ICCSA*, 690-700.
4. Bae, S.-H., Kijima, R., and Kim, W.-S. (2003). Digital Styling for Designers: 3D Plane-Symmetric Freeform Curve Creation Using Sketch Interface. *ICCSA*, 701-710.
5. Bae, S.-H. (2007). Pen-Based Drawing System. International Patent Application (Pending), Publication No. WO 2007/098243 A2.
6. Baudel, T. (1994). A Mark-Based Interaction Paradigm for Free-Hand Drawing. *UIST*, 185-192.
7. Beaudouin-Lafon, M. (2001). Novel Interaction Techniques for Overlapping Windows. *UIST*, 153-154.
8. Buxton, W. (1986). Chunking and Phrasing and the Design of Human-Computer Dialogues. *IFIP World Computer Congress*, 475-480.
9. Buxton, W. (1990). A Three-State Model of Graphical Input. *IFIP Tc13 International Conference on Human-Computer Interaction*, 449-456.
10. Cohen, J.M., Markosian, L., Zeleznik, R.C., Hughes, J.F., & Barzel, R. (1990). An Interface for Sketching 3D Curves. *I3D*, 17-22.
11. Dragicevic P. (2004). Combining Crossing-Based and Paper-Based Interaction Paradigms for Dragging and Dropping Between Overlapping Windows. *UIST*, 193-196.

12. Elber, G. (1995). Line Illustrations ∈ Computer Graphics. *The Visual Computer 11*, 6, 290-296.

13. Fitzmaurice, G., Balakrishnan, R., Kurtenbach, G., & Buxton, B. (1999). An Exploration into Supporting Artwork Orientation in the User Interface. *CHI*, 167-174.

14. Fowler, B. & Bartels, R. (1993). Constraint-Based Curve Manipulation. *IEEE Computer Graphics and Applications 13*, 5, 43-49.

15. François, A., Medioni, G., & Waupotitsch, R. (2003). Mirror Symmetry ⇒ 2-View Stereo Geometry. *Image Vision Computing 21*, 2, 137-143.

16. Grossman, T., Balakrishnan, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., & Buxton, W. (2001). Interaction Techniques for 3D Modeling on Large Displays. *I3D*, 17-23.

17. Grossman, T., Balakrishnan, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., & Buxton, W. (2002). Creating Principal 3D Curves with Digital Tape Drawing. *CHI*, 121-128.

18. Hanks, K. & Belliston, L. (2006). *Rapid Viz: A New Method for the Rapid Visualization of Ideas*, Thomson.

19. Hinckley, K., Baudisch, P., Ramos, G., & Guimbretière, F. (2005). Design and Analysis of Delimiters for Selection-Action Pen Gesture Phrases in Scriboli. *CHI*, 451-460.

20. Igarashi, T. & Hughes, J. (2001). A Suggestive Interface for 3D Drawing. *UIST*, 173-181.

21. Kara, L., D'Eramo, C., & Shimada, K. (2006). Pen-Based Styling Design of 3D Geometry Using Concept Sketches and Template Models. *SPM*, 149-160.

22. Kara, L. & Shimada, K. (2006). Construction and Modification of 3D Geometry Using a Sketch-Based Interface. *SBIM*, 59-66.

23. Kara, L. & Shimada. K. (2007). Sketch-based 3D Shape Creation for Industrial Styling Design. *IEEE Computer Graphics and Applications 27*, 1, 60-71.

24. Karpenko, O., Hughes, J., & Raskar, R. (2004). Epipolar Methods for Multi-View Sketching. *SBIM*, 167-173.

25. Kurtenbach, G. & Buxton, W. (1991). Issues in Combining Marking and Direct Manipulation Techniques. *UIST*, 137-144.

26. Kurtenbach, G., Fitzmaurice, G., Baudel, T. & Buxton, W. (1997). The Design of a GUI Paradigm Based on Tablets, Two-Hands, and Transparency. *CHI*, 35-42.

27. Lewin, T. & Borroff, R. (2003). *How To: Design Cars Like a Pro – A Comprehensive Guide to Car Design from the Top Professionals*, Motorbooks International.

28. Li, Y., Hinckley, K., Guan, Z., & Landay, J.A. (2005). Experimental Analysis of Mode Switching Techniques in Pen-Based User Interfaces. *CHI*, 461-470.

29. Martin, G. (1997). *The Art of Comic Book Inking*, Dark Horse Comics.

30. Mitani, J., Suzuki, H., & Kimura, F. (2000). 3D Sketch: Sketch-Based Model Reconstruction and Rendering. *Workshop on Geometric Modeling*, 85-98.

31. Moran, T., Chiu, P., & van Melle, W. (1997). Pen-Based Interaction Techniques for Organizing Material on an Electronic Whiteboard. *UIST*, 45-54.

32. Nealen, A., Igarashi, T., Sorkine, O., & Alexa, M. (2007). FiberMesh: Designing Freeform Surfaces with 3D Curves. *SIGGRAPH*.

33. Owada, S., Nielsen, F., Nakazawa, K., & Igarashi, T. (2003) A Sketching Interface for Modeling the Internal Structures of 3D Shapes. *Smart Graphics*, 49-57.

34. Ramos, G., Boulos, M., & Balakrishnan, R. (2004). Pressure Widgets. *CHI*, 487-494.

35. Robertson, S. (2004). *How to Draw Cars the Hot Wheels™ Way*, MBI.

36. Saund, E. & Lank, E. (2003). Stylus Input and Editing Without Prior Selection of Mode. *UIST*, 213-216.

37. Schmidt, R., Wyvill, B., Sousa, M., & Jorge J. (2005). ShapeShop: Sketch-Based Solid Modeling with BlobTrees. *SBIM*, 53-62.

38. Schmidt, R., Isenberg, T., Jepp, P., Singh, K., & Wyvill, B. (2007). Sketching, Scaffolding, and Inking: A Visual History for Interactive 3D Modeling. *NPAR*, 23-32.

39. Schmidt, R., Singh, K., & Balakrishnan, R. (2008). Sketching and Composing Widgets for 3D Manipulation. *Eurographics*, 301-310.

40. Singh, K., Grimm, C. & Sudarsanam, N. (2004). The IBar: A Perspective-Based Camera Widget. *UIST*, 95-98.

41. Taylor, T. & Hallett, L. (1996). *How to Draw Cars Like a Pro*, Motorbooks International.

42. Tsang, S., Balakrishnan, R., Singh, K., & Ranjan, A. (2004). A Suggestive Interface for Image Guided 3D Sketching. *CHI*, 591-598.

43. Wood, P. (1994). *Scientific Illustration: A Guide to Biological, Zoological, and Medical Rendering Techniques, Design, Printing, and Display*, John Wiley & Sons.

44. Zeleznik, R. & Miller, T. (2006). Fluid Inking: Augmenting the Medium of Free-Form Inking with Gestures. *GI*, 155-162.