

A Conceptual Modeling Approach Actualizing the Pencil and Paper Metaphor

Markus Färber
Technische Universität Ilmenau
P.O. Box 100 565
D-98684 Ilmenau
Markus.Faerber@tu-ilmenau.de
www.tu-ilmenau.de/cg

Jens Weggemann
3Dinteractive GmbH
Ehrenbergstraße 11
D-98693 Ilmenau
jw@3dinteractive.de
www.3dinteractive.de

Beat Brüderlin
Technische Universität Ilmenau
P.O. Box 100 565
D-98684 Ilmenau
bdb@prakinf.tu-ilmenau.de
www.tu-ilmenau.de/cg

Alexander Blazic
Technische Universität Ilmenau
P.O. Box 100 565
D-98684 Ilmenau
Alexander.Blazic@tu-ilmenau.de
www.tu-ilmenau.de/fakei?2270

Abstract

This article presents a system for sketching on pen-based displays to create conceptual engineering designs. The system incorporates recognition of command gestures, stroke preprocessing, segmentation, classification, and approximation, furthermore regularization and snapping, embedding and post-processing.

Particular attention has been paid to the usability of the system. To achieve our goal, we first analyzed the design process and the role of pencil and paper sketching.

Unlike other systems, our system has been designed for permanent revision of the input, including erasing and crossing out incorrect details. The system tries to strike a balance between the creative freedom of pencil and paper sketching and the advantages of digital data processing.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input Devices and Strategies; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques

Keywords: Pen-based systems; Conceptual modeling; Interaction techniques; Usability

1 Introduction

To date, pen-based computers are widely available. In the design domain, they are mainly used for artistic drawing rather than for Computer-aided Design.

Conceptual hand sketching lives off imprecise or deferred decisions; and inaccuracies caused by a restive hand do not preponderate. Computers in turn have other strengths: numerical precision, logical consistency, and efficient access to known data.

In this article, we present a sketching system for conceptual design which gives the designer creative freedom, but still exploits the computer's capabilities. The system lets the user input sketch strokes; they are then translated into line segments, arcs, circles, or B-splines. Rather than relying on a precise and complete stroke interpretation, the system transforms only the minimum that can be determined safely. To correct false interpretations, the user can redraw features or cross out unwanted details. This way, he neither needs to predict the system's reaction nor to execute pen-less functions: the user is much less distracted.

Copyright © 2007 by the Association for Computing Machinery, Inc.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions Dept, ACM Inc., fax +1 (212) 869-0481 or e-mail permissions@acm.org.
SPM 2007, Beijing, China, June 04 – 06, 2007.
© 2007 ACM 978-1-59593-666-0/07/0006 \$5.00

2 Analysis

In this section, we define design as problem solving and describe sketching as a creative process. In conclusion, technical requirements are given.

2.1 Phases of Design

[Akin 1994] sees design as a three-step cyclic process which follows a “*Likely to Succeed*” strategy. Especially the early phase heavily relies on sketching. In this phase the objectives are set, strategies are carried out — first ideas and premature designs are the result.

2.2 Information Processing Theory

The *Information Processing Theory*, presented by [Newell and Simon 1972], sees creative design as a goal-oriented cognitive process.

The brain is presumed to be a computer-like state machine. Three basic assumptions are made: (1) Information processing consists of atomic state transitions. (2) All information is coded in symbols. The totality of currently active symbols forms the system's overall state. (3) Problem solving is a serial process.

However, conceptual design differs from problem solving in one crucial point: design is an *ill-structured* problem.

2.3 Ill-structured Problems

Following [Goel 1995], design tasks often lack precise starting conditions, well-defined objectives, or established procedures. Furthermore, physics, legislation, or costs can impose constraints.

Having only expectations on a design's future use, the designer faces difficulties confronting his design with reality. So, he can only gradually shape the desired solution.

2.4 Wallas's Model of Creative Processes

[Wallas 1926] identifies four stages of the creative process:

Preparation: Design always has a goal. To state it as clearly as possible requires a tightly focused analysis.

Incubation: The problem is “put a side” for a while to let the subconscious absorb, arrange, and structure the information.

Illumination: At the moment of inspiration, the individual concepts absorbed during preparation and incubation aggregate to a oneness of superior quality: the idea.

Verification: Idea finding concludes in a critical evaluation balancing conception and reality.

2.5 Whole Brain Creativity Model

[Herrmann 1989] has analyzed lots of activities using his *Herrmann Brain Dominance Instrument (HBDI)*. The profile

“Design Engineer” features a twofold dominant characteristic with respect to logical thinking and imagination, while planning and intuition are less present.

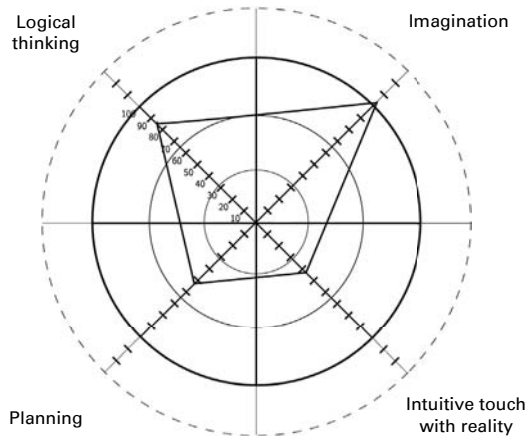


Figure 1: HBDI profile “Design Engineer”

2.6 Sketches

[Fish 2004] centers his analysis on the sketch as instrument of the creative process: Sketches serve to record and communicate ideas. Furthermore, they amplify visual thought and intention — a function that can be further diversified:

Translation of representational type: A designer continually switches between a descriptive view (focusing on function) and a depicting view (focusing on form).

Inventive perceptual retrieval: Sketches provide stimuli, sometimes of small magnitude, to unconsciously launch access of apparently blocked information.

Support for superimposed mental imagery: Sketches often appear diffuse or blurred. In spite of that, or maybe just therefore, the brain interprets a sketch by superimposing complex concepts.

Selective attention: A designer can deliberately focus attention to every single detail, without losing track of the overall design. This way, sketches provide an effective means to handle complexity.

Conscious monitoring of visual thought: Sketching a form directly by hand supports awareness about the own cognition, which is crucial for creative strength.

2.7 Technical Requirements for a Sketching System

The Pencil and Paper Metaphor

Seen as an interaction technique, pencil and paper allow for strictly modeless creation of geometry on a two-dimensional surface. The geometry consists of unconstrained strokes which can either be connected or unconnected. They will only implicitly be considered as line segments, arcs, circles, free-form curves etc.; and it is completely unnecessary to understand their potential mathematical representations. Miscarried details can easily be corrected by overstriking — at the cost of precision. Alternatively, erasing, redrawing, or crossing out work well to correct mistakes.

To take down a tentative draft, pencil and paper completely fulfill the five principles of usability given by [Shneiderman 2002] and [Nielsen 1994]: Sketching is easy to learn, memorable, quick, hardly error-prone, and convenient.

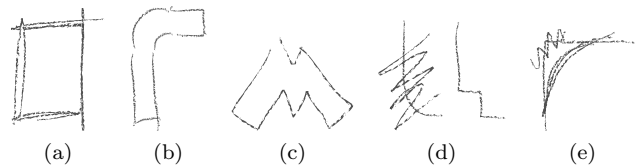


Figure 2: Examples of sketchy drawings

Only when it comes to precision, the amount of work grows considerably — both effectiveness and efficiency collapse.

Computer-aided Design

At that point, Computer-aided Design comes into play. In the later design phases, details matter, and exact geometry must be drafted. So, a CAD user must know the underlying mathematical representing the geometry.

Interaction techniques are often rigorous stepwise processes, far from natural sketching. In return, CAD provides powerful operations as well as easy means to connect new geometry to existing one. Furthermore, *in situ* modification techniques and undo/redo functionality greatly ease editing.

Technical Decisions

We set the focus clearly on the pencil and paper metaphor. Consequently, all interactions must strictly be modeless. Supported geometry are line segments, circles, arcs, and cubic B-splines; their mathematical equivalents remain hidden. Instead of modal *in situ* modifications as in [Baudel 1994] or [Fowler and Bartels 1993], equally powerful modeless redrawing as from [Michalik et al. 2002] is used.

To draw the system nearer to CAD, two views are defined: the uninterpreted view of the sketchy input and the internal exact view showing the corresponding geometrical primitives, including points where they connect.

Furthermore, the system features a modeless snapping paradigm to link and align new geometry with existing one. Eventually, undo, delete, and edge splitting operations are available — all performed by gestures.

3 Architecture

Separation of Views

Sketch strokes are visualized upon input using pencil-like rendering. When the pen is taken off, the stroke is immediately interpreted as a geometrical primitive, to be visualized by solid line style.

But, instant passage to the exact view would impede the awareness of cognition while manual switching would introduce modes. [Arvo and Novins 2000] suggests morphing, but our tests yielded, that a simple cross-fade, after about two seconds delay, performs best.

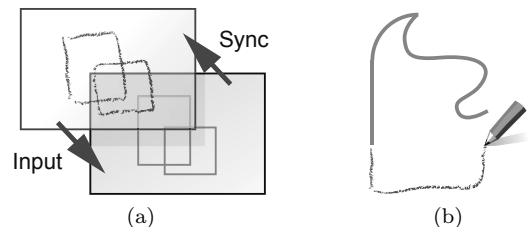


Figure 3: Separation of input view from exact view

Stages of Processing

The overall architecture to process a pen stroke follows a chain-like structure.

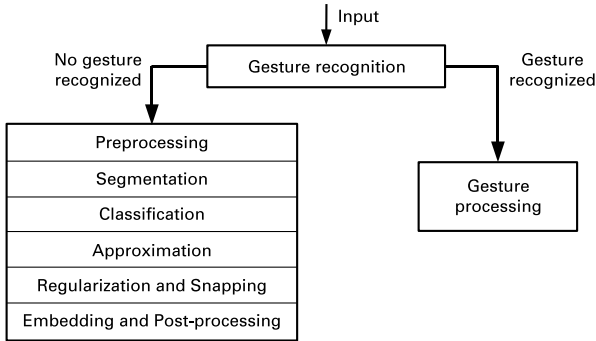


Figure 4: Stages of processing

3.1 Gesture Recognition and Processing

A gesture is a pen stroke that triggers a command, instead of being an ordinary input.

Figure 5a shows an example of the delete gesture, which features rigorous hatching. To recognize this gesture, a — slightly improved — heuristic from [Fonseca and Jorge 2000] is used: the stroke length must be much greater than the covered surface area. Elements which are at least 30 percent inside the region covered by the gesture will be deleted.

Making the gesture on empty space triggers the undo operation. Drawing a very short line on or closely aside an edge splits that edge. This gesture is recognized by finding the overall length of the stroke to be less than a threshold.

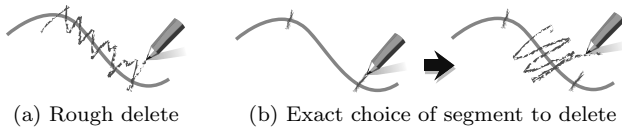


Figure 5: Deleting only a chosen part of an edge

3.2 Stroke Preprocessing

If a stroke is not a gesture, small hooks and squiggles are filter out using algorithms from [Wenyin et al. 2001].

3.3 Stroke Segmentation

Before converting a stroke into a geometrical primitive, the system tries to segment it into several primitives. In [Färber et al. 2007a] we describe our new segmentation algorithm at greater detail.

Direction estimation

Finding sharp corners relies on investigating the angle at a sample point p_i : $d_i = \angle(\overrightarrow{p_{i-1}p_i}, \overrightarrow{p_i p_{i+1}})$.

Simple direction change d_i is a heavily disturbed signal, as seen in Figure 6b. So, we apply orthogonal distance regression to calculate it over a window of k neighbor points, leading to much less disturbs [Sezgin et al. 2001].

Curvature estimation

[Kim and Kim 2006] observed, that corner points always have the following properties:

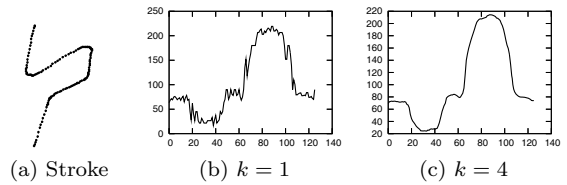


Figure 6: Approximated directions for window size k

- At a corner p_i , direction change d_i has a minimum.
- In a region before and after p_i , direction changes have the same arithmetic sign (local convexity), but decreasing absolute value (local monotonicity).

Consequently, Kim and Kim suggest a new curvature estimation algorithm. Figure 7 shows some exemplary results.

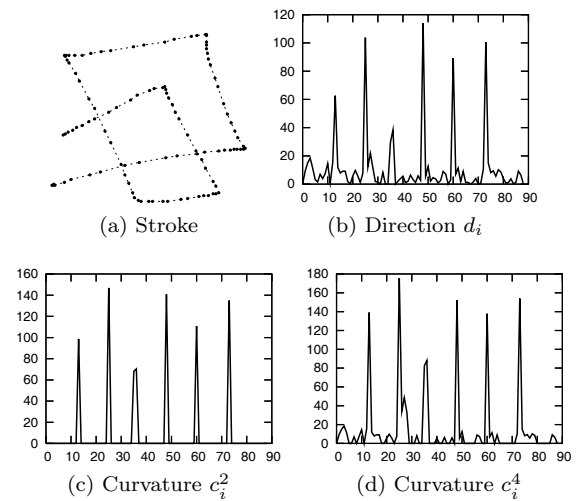


Figure 7: Kim and Kim's curvature estimation

Estimation of feature quantity

When an input stroke has only a few significant features, the known algorithms recognize too many false positives. To avoid this, the feature quantity $r = \rho/MD$ gives a good estimation. ρ is the range, i. e., the difference between maximum and minimum of all direction values, and MD is the mean deviation from their median.

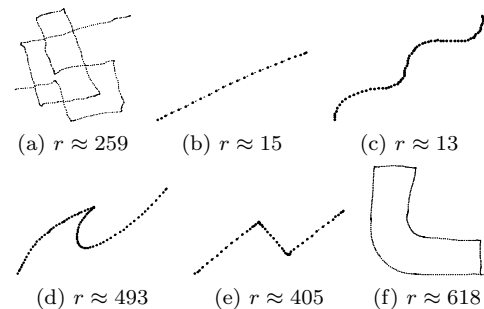


Figure 8: Feature quantity estimation for some strokes

Overall algorithm

Now, the overall algorithm can be formulated. It yields a more tolerant and reliable feature detection than reported in [Kim and Kim 2006]. The last two steps produce features at points where pen input is paused intentionally.

1. Compute the direction changes d_i with orthogonal distance regression ($k = 5$).
2. Compute curvatures c_i^2 using Kim and Kim's algorithm.
3. Compute delays (see [Färber et al. 2007a] why speed can be replaced by mere time difference). Cut delays greater than 300 milliseconds.
4. Compute feature quantity r . If $r < 100$, then ignore delay values in the forthcoming steps.
5. Threshold-based maximum search (i. e. find all local maxima that are above a threshold) on delays. Threshold is the sum of average and mean deviation.
6. Compute curvatures c_i^4 using Kim and Kim's algorithm for local maxima from step 3. Threshold is 40; if a local maximum delay exists at the same point, the threshold is set to 5.
7. Threshold-based maximum search on delays. Threshold is the arithmetic mean of average and maximum delay.
8. Add points with local maximum delay over 80 milliseconds from results of step 7 to the results of step 6.

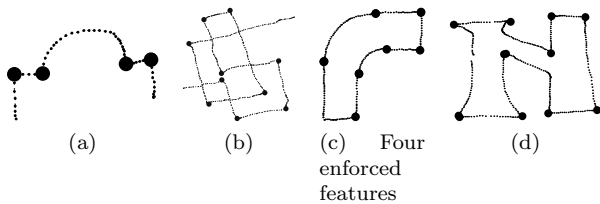


Figure 9: Example strokes, recognized features are marked

3.4 Classification and Approximation

Stroke segments will then be classified and approximated.

At first, a line is fitted to the points by least-squares fitting [Eberly 2001].

If the approximation error exceeds a threshold, the system fits an arc to the points. The circle's center and radius are determined using least-squares fitting from [Jin et al. 2002].

If again the approximation error is too large, the stroke will be approximated by a cubic B-spline. At first, the stroke points are parameterized using the centripetal method. Then, a knot vector is created having the parameters uniformly distributed. Finally, solving a linear equation system yields the position of the control points. The whole process is explained in [Piegl and Tiller 1997]. The approximation still has one degree of freedom: the number of control points. To find the optimum, we iterate the number of control points in five steps from four to one third the number of input samples until the approximation error falls below a threshold.

3.5 Regularization and Snapping

The resulting primitives are then snapped onto existing geometry or grid points. Furthermore, primitives are regularized to match the main axes. In contrast to previous snapping approaches like [Bier 1988] or the regularization

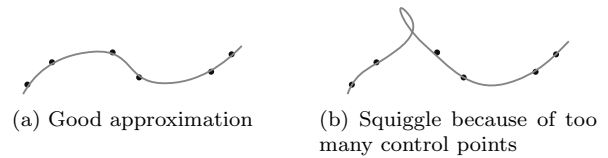


Figure 10: Degrees of freedom in B-spline approximation

scheme from [Wenyin et al. 2001], our system works without any modes [Färber et al. 2007b]. Furthermore, snapping onto points has been generalized to respect tangent and curvature constraints.

Continuity

Two curves are C^n -continuous at a point, if all i -th derivatives at that point are equal, with $0 \leq i \leq n$. Hence, C^2 -continuity implies C^1 - and C^0 -continuity.

In addition to parametric continuity C^n , geometric continuity G^n can be defined [Piegl and Tiller 1997]. Two types are relevant: G^0 -continuity: positions are equal, and G^1 -continuity: tangent vectors have the same direction but differ in length. Note, that G^0 -continuity equals C^0 -continuity and that C^1 -continuity enforces G^1 -continuity.

Continuity levels can be ordered with respect to the quality of smoothness: $G^0 \rightarrow G^1 \rightarrow C^1 \rightarrow C^2$.

Snapping Algorithm

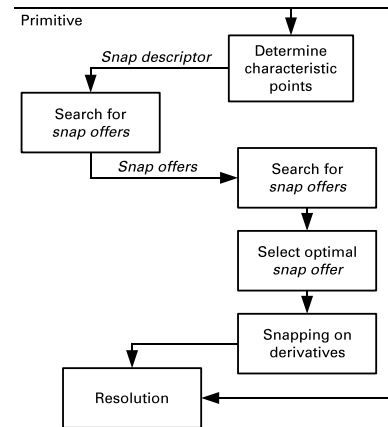


Figure 11: Modeless snapping algorithm

The snapping procedure consists of five steps:

Determine characteristic points: At first, *characteristic points* of the input primitive are determined: end points of line segments or arcs (G^0 and G^1), centers of arcs or circles (G^0), as well as end points of B-splines (G^0 up to C^2).

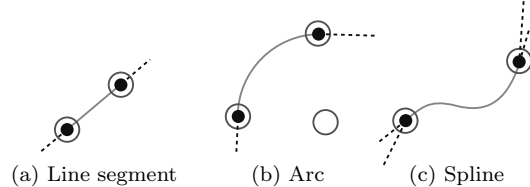


Figure 12: Characteristic points with continuity levels

Which level of continuity will actually be applied, is to be determined in the resolution step.

Each characteristic point is held in a *snap descriptor* containing snap position, tangent vector (with flag if tangent length has to be considered), and curvature vector with a similar flag.

Search for snap offers: Now, having snap descriptors for each characteristic point of the input, the scene is searched for possible points to snap the input to. These positions are held as a set of *snap offers* containing a snap descriptor and a category number for priority control. End points have highest priority, followed by points on edges, circle centers, and grid points.

To determine the relevant snap offers, the algorithm finds all primitives near the input's snap descriptors. For characteristic points of these primitives, snap offers are created. Then, the characteristic points of the input are projected onto the found primitives to create snap offers for the foot points. Finally, snap offers are produced for the nearest grid points.

Selection of the optimal snap offer: From the set of snap offers, all elements are removed which exceed given distance thresholds. From the remaining snap offers only the most prioritized ones will be kept; the closest will be chosen as the optimum.

In Figure 13a, P_2 will be excluded for being too far from the edge; in Figure 13b, A has higher priority; and in Figure 13c, A is chosen because it is closer to P .

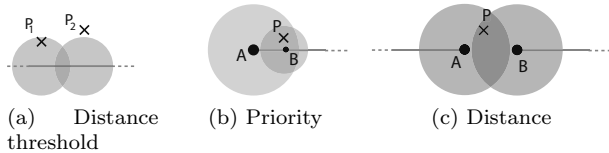


Figure 13: Selection of the optimal snap offer

Snapping on derivatives: Now, the input's snap descriptors are compared with the scene's snap offers to determine which continuity level can be reached. In a process, from $C^2 \rightarrow C^1 \rightarrow G^1 \rightarrow G^0$, the differences between the respective values are checked against thresholds. For example, to apply G^1 -continuous snapping, the angle between the tangents must not differ greatly. For all unsuitable continuities, the respective snap offer entries are cleared.

Resolution: At that point, actual snapping can be done: the characteristic points of the input primitive are moved onto the respective snap offers.

To snap the end points of a B-spline curve, the original stroke samples are newly approximated, this time with Lagrange multipliers expressing the side conditions [Piegl and Tiller 1997]. If only G^0 -continuity is desired, neighboring stroke samples are slightly moved to avoid ugly bends (see Figure 16).

3.6 Embedding and Post-processing

Eventually, the exact representation of the input can be stored as boundary representation. Connection information is preserved. For an example see Figure 17.

4 Conclusions and Future Work

In this article, we presented an analysis of the sketching process pointing out, that the pencil and paper metaphor is most suitable for the earlier phases of engineering design.

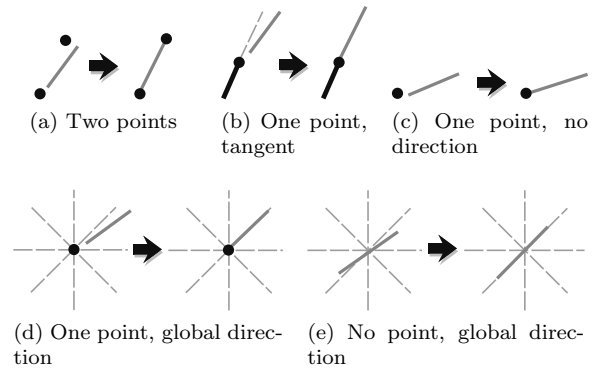


Figure 14: Snapping cases for line segments

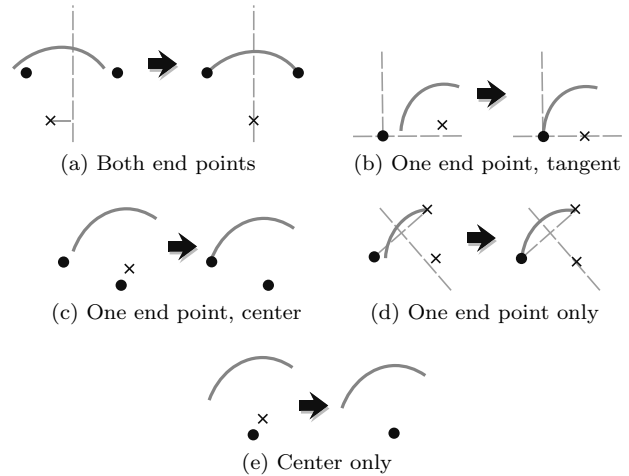


Figure 15: Snapping cases for line segments and arcs

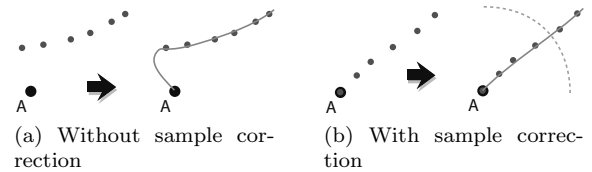


Figure 16: B-spline approximation with fixed end point

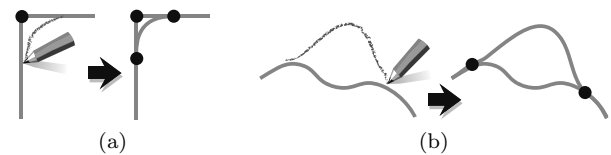


Figure 17: Example interactions

This result has been translated into concrete technical decisions on how to build a computer-assisted sketching system for conceptual modeling solely featuring modelless interaction techniques.

As hoped by the authors, the implemented system subjectively proved to be powerful and useful. Some of its effectiveness is due to improvements of sketch interpretations, in particular the segmentation and removal of artifacts. The largest improvement is due to the close adherence of the sys-

tems to the pencil and paper metaphor. Rather than relying on too much intelligence and complete accuracy, which is a hopeless endeavor to begin with, the system allows for easy and intuitive methods to correct wrong input by the user, and by the same means erroneous interpretations by the system. The user can make these corrections at any time during the design process, so not to interrupt the creative thought process, rather than let the system dictate the pace.

Nevertheless, a lot of future work still remains to be done. Dearly needed is a usability study which should quantify the postulated improvements in effectiveness and efficiency. This should reveal if the reduced set of interactions is really sufficient for more complex drawing tasks. At the same time, all the system parameters should be adjusted to match the user's expectations. Hopefully, some secret rule, governing these values, can be distilled, to finally yield an even more intuitive system behavior.

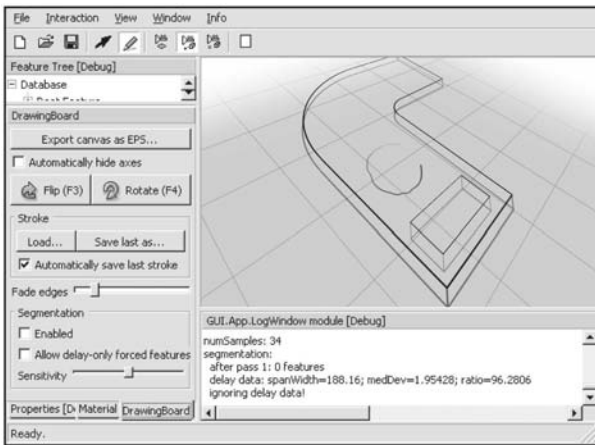


Figure 18: Sketching application

Acknowledgements

We thank Rike Brecht, Ulf Döring, Gabriele Janner, Heidi Krömker, Tibor Kunert, Peter Tail, Katharina Wild, as well as Ruby and Pearl.

References

- AKIN, Ö. 1994. Psychology of early design in architecture. In *Proceedings of the 2nd Conference on Design and Decision Support Systems in Architecture & Urban Planning. Vaals, The Netherlands, 15-19 August 1994*.
- ARVO, J., AND NOVINS, K. 2000. Fluid Sketches: Continuous recognition and morphing of simple hand-drawn shapes. In *Proceedings of the 13th ACM Symposium on User Interface and Software Technology (UIST'00). San Diego, CA, USA, 6-8 November 2000*, 73–80.
- BAUDEL, T. 1994. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of the Seventh ACM Symposium on User Interface and Software Technology (UIST'94). Marina del Rey, CA, USA, 2-4 November 1994*, 185–192.
- BIER, E. 1988. *Snap-dragging: Interactive Geometric Design in Two and Three Dimensions*. PhD thesis, University of California, Berkeley, CA, USA.
- EBERLY, D. 2001. Least squares fitting of data. Tech. rep., Geometric Tools Inc., Chapel Hill, NC, USA. <http://www.geometrictools.com> — Online Resource, 2 February 2006.

- FÄRBER, M., WEGGEMANN, J., AND BRÜDERLIN, B., 2007. Feature recognition in sketch-based input. Submitted for publication.
- FÄRBER, M., WEGGEMANN, J., AND BRÜDERLIN, B., 2007. A modeless snapping system for sketch-based conceptual design. In preparation.
- FISH, J. 2004. Cognitive catalysis: Sketches for a time-lagged brain. In *Design Representations*, G. Goldschmidt and W. L. Porter, Eds. Springer, London, UK, 151–184.
- FONSECA, M. J., AND JORGE, J. A. 2000. Using fuzzy logic to recognize geometric shapes interactively. In *Proceedings of the 9th IEEE International Conference on Fuzzy Systems (FUZZ IEEE 2000). San Antonio, TX, USA, 7-10 May 2000*, vol. 1, 291–296.
- FOWLER, B. M., AND BARTELS, R. H. 1993. Constraint-based curve manipulation. *IEEE Computer Graphics and Applications* 13, 5, 43–49.
- GOEL, V. 1995. *Sketches of Thought*. MIT Press, Cambridge, MA, USA.
- HERRMANN, N. 1989. *The Creative Brain*. Brain Books, Lake Lure, NC, USA.
- JIN, X., WENYIN, L., SUN, J., AND SUN, Z. 2002. On-line graphics recognition. In *Proceedings of the Tenth Pacific Conference on Computer Graphics and Applications 2002 (PG 2002). Beijing, China, 9-11 October 2002*, 256–264.
- KIM, D. H., AND KIM, M.-J. 2006. A curvature estimation for pen input segmentation in sketch-based modeling. *Computer-Aided Design* 38, 3, 238–248.
- MICHALIK, P., KIM, D.-H., AND BRÜDERLIN, B. 2002. Sketch- and constraint-based design of B-spline surfaces. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications (SMA'02). Saarbrücken, Germany, 17-21 June 2002*, 297–304.
- NEWELL, A., AND SIMON, H. A. 1972. *Human Problem Solving*. Prentice-Hall, Englewood, NJ, USA.
- NIELSEN, J. 1994. *Usability Engineering*. Morgan Kaufmann, San Francisco, CA, USA.
- PIEGL, L., AND TILLER, W. 1997. *The NURBS Book*, 2nd ed. Springer, Berlin, Germany.
- SEZGIN, T. M., STAHOVICH, T., AND DAVIS, R. 2001. Sketch based interfaces: Early processing for sketch understanding. In *Proceedings of the Workshop on Perceptive User Interfaces (PUI 2001). Orlando, FL, USA, 15-16 November 2001*, 1–8.
- SHNEIDERMAN, B. 2002. *User Interface Design*. mitp, Berlin, Germany.
- WALLAS, G. 1926. *The Art of Thought*. Harcourt Brace-Jovanovich, New York, NY, USA.
- WENYIN, L., QIAN, W., XIAO, R., AND JIN, X. 2001. Smart Sketchpad – an on-line graphics recognition system. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR 01). Seattle, WA, USA, 10-13 September 2001*.