



Free-Sketch Recognition: Putting the CHI in Sketching

**Tracy Hammond,
Brian Eoff,
Brandon Paulson,
Aaron Wolin,
Katie Dahmen,
Joshua Johnston,
Pankaj Rajan**

Sketch Recognition Lab
Department of Computer Science
Texas A&M University
College Station, TX 77840 USA
{hammond, bde, bpaulson, awolin,
katiedahmen, jbjohns, pankaj}
@cs.tamu.edu

Copyright is held by the author/owner(s).
CHI 2008, April 5 – April 10, 2008, Florence, Italy
ACM 978-1-60558-012-8/08/04.

Abstract

Sketch recognition techniques have generally fallen into two camps. Gesture-based techniques, such as those used by the Palm Pilot's Graffiti, can provide high-accuracy, but require the user to learn a particular drawing style in order for shapes to be recognized. Free-sketch recognition allows users to draw shapes as they would naturally, but most current techniques have low accuracies or require significant domain-level tweaking to make them usable. Our goal is to recognize free-hand sketches with high accuracy by developing generalized techniques that work for a variety of domains, including design and education. This is a work-in-progress, but we have made significant advancements toward our over-arching goal.

Keywords

Sketch recognition, pen input, tablet pc, multimodal interaction, free-sketch, LADDER, PaleoSketch, ShortStraw

ACM Classification Keywords

I.7.5 Document Capture: Graphics recognition and interpretation; I.4.6 Segmentation: edge and feature detection. I.2.10 Vision and Scene Understanding: Shape; Perceptual reasoning; H.5.2 User Interfaces: Input devices and strategies

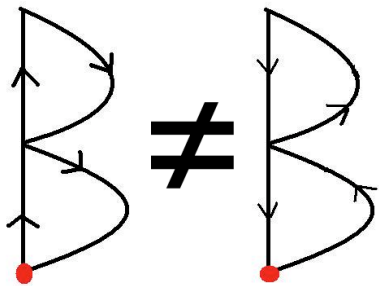


Figure 1: Two shapes that look similar, but would be recognized as two separate shapes by gesture-based systems.

Introduction

Sketch recognition is the automatic understanding of hand-sketched input via an electronic stylus. The recognition of hand-sketches has a variety of uses, including front ends for Computer-Aided Design systems, automatic correction or understanding of diagrams for immediate feedback in educational settings, alternative inputs for small keyboard-less devices (such as Palm Pilots), or gestural interfaces. Existing sketch recognition systems fall into one of two camps: gestural or free-sketch.

Gesture-based systems require each shape to be drawn in a particular style (making it a *gesture*, rather than a *shape*). Users have to learn how to draw each symbol since stroke order, stroke direction, and the number of strokes are determining factors for recognition. Recognition is performed based on a number of drawing-style features, such as the speed of the stroke, the start and end direction of the stroke, and the total rotation of the stroke [10]. The advantage of gesture-based systems is that if users draw shapes as defined, recognition accuracy can be quite high. The disadvantage is that users have to learn how to sketch the gestures, and the initial learning curve can hinder overall accuracy.

At the other end of the spectrum exist free-sketch systems that allow users to draw as they would

naturally. Such systems attempt to recognize the shape by vision- [7], geometric- [1, 2], and feature-based techniques [4, 9]. The advantage of free-sketch systems is that they allow users to sketch in a non-constrained manner. The disadvantage of these systems is that recognition accuracy is modest, unless tuned for a specific domain. Because of this limitation, most current systems attempting free-sketch recognition are somewhere in between the gesture-based and free-sketch system, relinquishing some constraints, while keeping others.

Our vision is to improve free-sketch recognition to the point that the accuracy is at least as good as that obtain by a human recognizer. In order to do this, we are developing and improving a framework for recognition.

Approach

Our approach for a free-sketch recognition platform relies on: 1) a variety of geometric, feature, temporal, contextual, multi-modal, and domain information that provides intuition about what shape is being drawn on the screen, 2) a way to describe the sketch information, and 3) the application of machine learning techniques that take advantage of, and learn from, the information provided.

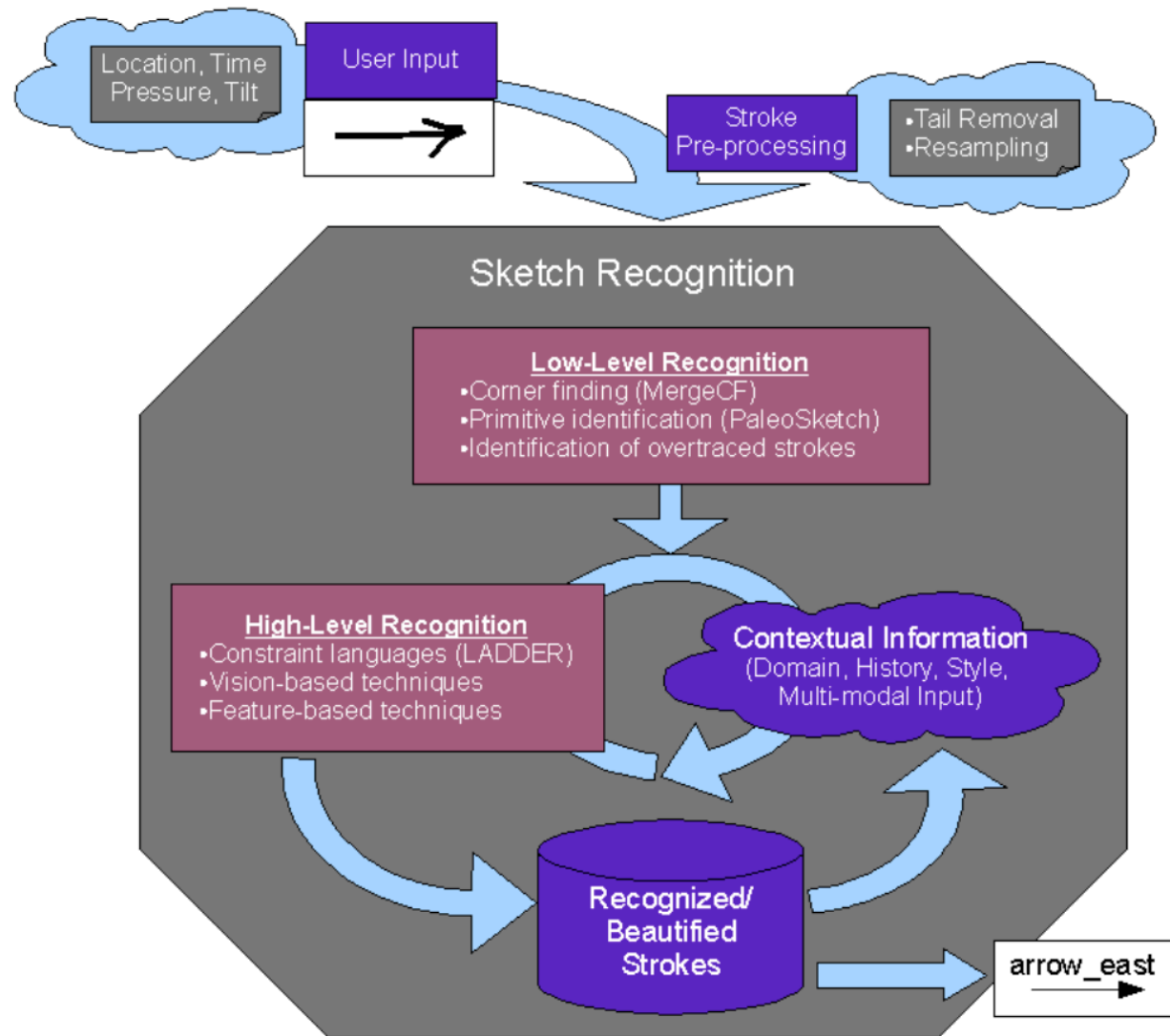


Figure 2. The variety of sketch information used to recognize free-hand sketched diagrams.

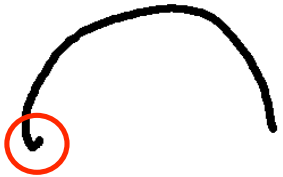


Figure 3: An unintentional 'tail' drawn by a user.

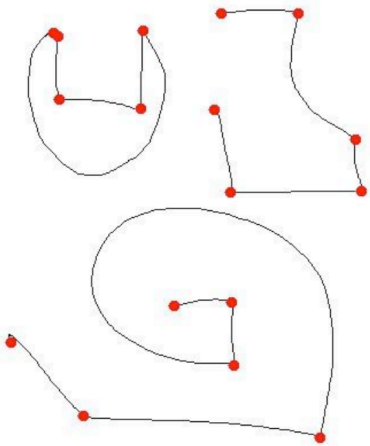


Figure 4: Corner finding results from our current corner finding algorithms.

Sketch Information

Figure 2 shows the variety of sketch information and processing steps useful in recognizing a free-sketch diagram. Sketch information and processing occurs on two levels: 1) low-level stroke processing, and 2) higher-level domain and user level processing.

Low-level stroke processing

Low-level processing breaks down strokes into primitives such as lines and circles, which can later be used to form higher-level shapes. Strokes are first pre-processed or cleaned by removing consecutive, duplicate points, as modern Tablet PC's will often sample points at the same location and/or time. Stroke tails, as in Figure 3, are removed since they are unintended by the user.

Cusp or corner detection is used to segment a stroke into its different primitives. Corners are detected using a variety of techniques including curvature and timing information [3, 11], our short straw method [in submission], our merging method [in submission], and recursion [5, 8, 13]. See Figure 4.

Segmented sub-strokes are then recognized as low-level primitives. Geometric and drawing-style information is used to match each segment to the primitive that a human would most likely perceive. These primitives form the basis for higher-level shapes. Because our domains are built from primitives, a larger primitive library will allow for a greater variety of domains.

Since humans draw imprecisely, it is important to return normalized confidence values to rank primitive interpretations. Context can then be used to

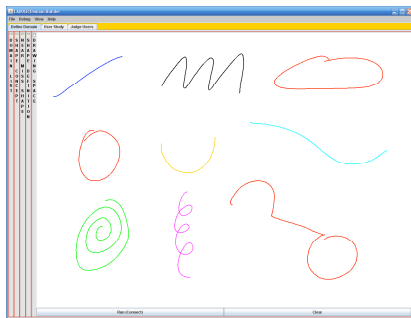
disambiguate these multiple interpretations. This is especially important when a domain differentiates between circles and ellipses or between arcs and curves, which are visually similar.

Thus far, we have built a low-level recognizer that recognizes eight different primitives, including line, ellipse, circle, arc, curve, polyline, spiral, and helix [8]. A limited ranked list of interpretations is returned, with an average of 2.6 interpretations returned per shape. Recognition rates for these shapes are currently at 99.8% for the correct interpretation appearing in the returned ranked list and at 98.6% for the correct interpretation appearing in the first slot of the ranked list. See Figure 5.

Higher-level information

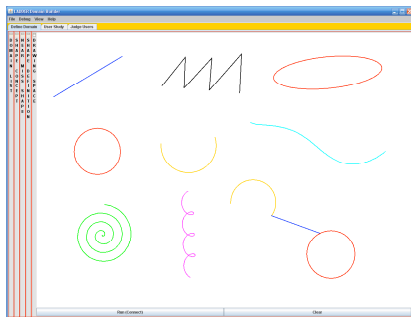
In order to combine primitives into higher-level shapes, a variety of geometric, user, and domain information is utilized. First, low-level primitive shapes may be joined together in case a primitive has been drawn with more than one stroke. Geometric information combines shapes together by applying a series of constraint rules. These constraints again need probabilities and rankings assigned to them because human sketching is imprecise. Vision techniques, such as those used by [7] could also be used.

Domain-specific information not only specifies which shapes are in the domain but also provides additional helpful context. To give an example, when provided two equally probable interpretations of an electrical circuit diagram, the system should favor the interpretation that allows current flow.



Different users draw using different styles. Certain users may favor certain shapes or drawing orders. This information can be used to reduce ambiguities and expedite recognition by checking for the most probable interpretation first. Additionally, multimodal information, in particular, accompanying speech, can be used to weight different interpretations for use in disambiguation.

Specifying Sketch Information



In order to specify sketch information, we have built and are improving the LADDER sketch language. This language allows users to specify the shapes in a domain and how each shape is drawn, such as the geometric, constraint, and contextual information associated with each shape. User interface information can also be included, which defines how shapes should be displayed and edited in a domain after they are recognized.

Figure 5: Original and beautified strokes recognized by our low-level primitive recognition system, PaleoSketch [8].

Because it is more natural to draw a shape than to describe it textually, a user can also draw a shape and have it automatically transformed into a LADDER text description. However, at this time, contextual information must be input by hand. We are looking into ways that this information can be learned automatically.

Combining Information

In order to recognize shapes while taking advantage of the myriad of sketching information available to us, machine-learning techniques must be used to recognize the shapes. We are building a probabilistic model to recognize shapes somewhat similar to a Bayesian

network, as in [1]. HMMs can be used separately to learn users' temporal preferences, similar to techniques used in [12].

Current Results

Thus far, over 30 different people have built sketch recognition systems, in such varied domains as mechanical engineering, electrical engineering circuit diagrams, UML class diagrams, Japanese Kanji, Chinese, Chemistry diagrams, and many others. These types of systems can be built in a matter of weeks using LADDER and achieve accuracies comparable to expert systems. Figures 6 and 7 show two systems automatically generated using LADDER descriptions. A video of a children's sketch-based game to teach shapes using our framework is attached.

Conclusion

The members of the Sketch Recognition Lab at Texas A&M University are working on a free-sketch recognition platform that will make building accurate sketch recognition systems easier for many domains. Currently, we have made much headway on the problem.

Acknowledgements

This work-in-progress is an effort by a large amount of folks. We thank Randall Davis for helping to shape our ideas, as well as other previous members of the MIT Design Rationale Group – specifically including Christine Alvarado, Metin Sezgin, Michael Oltmans, Aaron Adler, Sonya Cates, Tom Ouyang, Olya Veselova, and Jacob Eisenstein. We also wish to thank the members of the Sketch Recognition class of 2006 and 2007, whose insights and progress were infinitely helpful. This work funded in part by NSF IIS grant 0744150: Developing

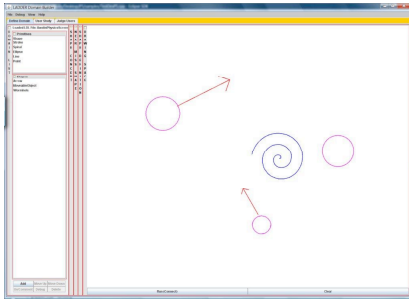


Figure 6: A planetary physics-based simulation implemented in LADDER.

Perception-based Geometric Primitive-shape and Constraint Recognizers to Empower Instructors to Build Sketch Systems in the Classroom.

References

- [1] Alvarado C., Davis R. Sketchread: A multi-domain sketch recognition engine. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (2004), Santa Fe, NM, 23–32.
- [2] Hammond, T. *LADDER: A Perceptually-based Language to Simplify Sketch Recognition User Interface Development* (2007). MIT PhD Thesis.
- [3] Herot, C. Graphical Input Through Machine Recognition of Sketches. In *Proceedings of SIGGRAPH '76* (1976), 97–102.
- [4] Kara, L.B, Stahovich, T.F. An image-based trainable symbol recognizer for sketch-based interfaces. *AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Natural* (2004).
- [5] Kim, D.H., Kim M.J. A curvature estimation for pen input segmentation in sketch-based modeling. *Computer-Aided Design* (2006), 38(3), 238-248.
- [6] Long A.C. *Quill: a gesture design tool for pen-based user interfaces* (2001), U.C. Berkeley PhD Thesis.

[7] Oltmans, M. *Envisioning Sketch Recognition: A Local Feature Based Approach to Recognizing Informal Sketches* (2007), MIT PhD Thesis.

[8] Paulson, B., Hammond, T. PaleoSketch: Accurate Primitive Sketch Recognition and Beautification. In *Proceedings of Intelligent User Interfaces (IUI'08)* (2008).

[9] Plimmer B. Ink Features. *Eurographics Sketch-based Interfaces and Modeling* (2007).

[10] Rubine D. Specifying gestures by example. *Computer Graphics* (1991), 25(4), 329–337

[11] Sezgin T.M., Stahovich T., Davis R. Sketch based interfaces: early processing for sketch understanding. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces* (2001), ACM Press, 1-8.

[12] Sezgin, T. M., Davis, R. HMM-based efficient sketch recognition. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI'05)* (2005), ACM Press, 281–283.

[13] Yu B., Cai S. A domain-independent system for sketch recognition. In *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (2003), ACM Press, 141-146.

Figure 7: A simple course-of-action diagram recognizer implemented in LADDER. The left shows the originally drawn strokes and the right shows the cleaned primitives with the automatically produced label.

