EUROGRAPHICS Tutorial on Sketch Recognition

T. Hammond, B. Paulson, B. Eoff

Sketch Recognition Lab, Computer Science & Engineering, Texas A&M University, USA

Abstract

Sketch recognition is the automated understanding of hand-drawn diagrams. Despite the prevalence of keyboards and mice, hand-drawings still pervade in education, design, and other diagrams. This full day tutorial explains why sketch recognition is important, the underlying algorithms, how sketch recognition can be used in traditional interfaces, and the field's experiences with sketch recognition used in different domains.

Categories and Subject Descriptors (according to ACM CCS): Document Capture [I.7.5]: Graphics recognition and interpretation—User Interfaces [H.5.2]: Input devices and strategies, Interaction styles—Implementation [I.5.5]: Interactive Systems—

1. Introduction

Sketch recognition is a growing topic of interest to many people as graphical diagrams pervade education [AAS*04], business, design [DG96], and many other domains. Students draw diagrams in fields as varied as business, math, computer science, engineering, music, and many others. Hand-sketched student diagrams aid in active learning and creative processes. Designers use sketching throughout the development process, aiding in the brainstorming of ideas, visualizing organization, and understanding of requirements. Unfortunately when it comes to implementing the design, the drawings are left behind and a CAD tool is chosen because of its ability to understand and process the design. Furthermore, in education, correcting hand-sketch diagrams takes a significant amount of teacher time, and are, thus, often left out of the testing process. Sketch recognition provides a way to bridge that gap, automatically processing hand-sketched designs and correcting student-drawn diagrams to provide immediate student and instructor feedback, while significantly reducing instructor time.

This tutorial will provide participants with an introduction to sketch recognition. Participants will learn about a myriad of sketch recognition algorithms that rely on drawing style, geometry, context, timing, bitmap, and other features, and the techniques used to recognize sketches from these features including rule-based, linear and quadratic, HMM, Bayesian Network and other based classification algorithms. User interface techniques and issues will also be addressed, such as beautification and editing of diagrams. Participants will walk away with an overall understanding of the pitfalls and advantages of the different techniques and features, as well as an idea of how to implement them. Participants will be provided with a CD with videos and demos showing the different types of recognizers, to allow interactive learning throughout the tutorial.

1.1. Prerequisite Knowledge

Participants are expected to be proficient in coding and computer science in general. Participants should have taken some higher level computer science courses, such as artificial intelligence, user interfaces, or software engineering, in order to successfully grasp the conceptual material. No specific courses in machine learning or pattern recognition will be assumed.

1.2. Tutorial Goals

1. Provide an awareness of the ongoing work in sketch recognition.
2. Instruct participants on a myriad of sketch recognition techniques and algorithms including style-, vision-, geometry-, context-, HMM-, and Bayesian Network-based recognition algorithms, as well as beautification, editing, and general user interface techniques.
3. Educate participants of the drawbacks and advantages of the different techniques.
4. Provide participants with the tools to implement the different techniques.

5. Provide an interactive and engaging tutorial filled with videos and interactive demos to provide a firmer understanding of the topics.
6. Walk participants through the process of creating a sketch recognition system for themselves with the current technologies available to them.

The presentation of this tutorial will be particularly innovative not only in its topic, but also in its presentation. The presentation will be interactive filled with videos and demos both showing the techniques in action, but also that highlight the processes that the underlying technologies use. A CD will be provided for participants so that they may participate in the demos on their own tablet or laptop computers along with the lecturer (without worrying if the network will be available). The CD will include the videos and demos shown, as well as copies of the referenced published papers. While the demos were designed to work with a tablet PC and pen (by nature of the subject), users can also use a regular notebook (PC, Mac, or Linux) and mouse, and the demos will still work.

1.3. Sketch Recognition Algorithms

1.4. Gesture Recognition

Gesture recognition is one of the first attempts at recognizing hand-drawn glyphs. Gesture-based methods typically focus on how a particular stroke is drawn rather than on what the stroke actually looks like. Because of this, many drawing constraints are placed on users who interact with these systems (e.g. all circles must be drawn counter-clockwise). Gesture-based algorithms covered by this tutorial include Rubine [Rub91] , Long et al. [LLRM00], and Wobbrock et al.'s $1 [WWL07].

1.5. Corner Finding

Due to the constraints placed on users by gesture-based systems, recent approaches focus more on what the stroke looks like rather than on how it was drawn. These new breed of algoritms are geometric in nature and account for variances in scale, rotation, and orientation. Corner finding is the first step of these algorithms and has been a widely published sub-field of sketch recognition. Corner finding (also known as stroke segmentation or fragmentation) is the process of determining the perceptual "corners" of a stroke. Current algorithms focus primarily on pen speed and curvature and work either for polylines [DP73, WEH08] or complex combinations of lines and arcs [Her76, SSD01, YC03, KK06, Sau90].

1.6. Primitive Recognition

After corner finding is performed, low-level recognizers are responsible for interpreting single strokes as one of many different primitive shapes. Primitive shapes are those which are used by higher-level sketch recognition systems as building blocks to form more complex symbols. Common primitives include lines, arcs, polylines, and ellipses [SSD01, YC03]. More recent algorithms have also expanded their primitive lists to include other shapes like curves, helixes, spirals, etc. [PH08]. See Figure 1

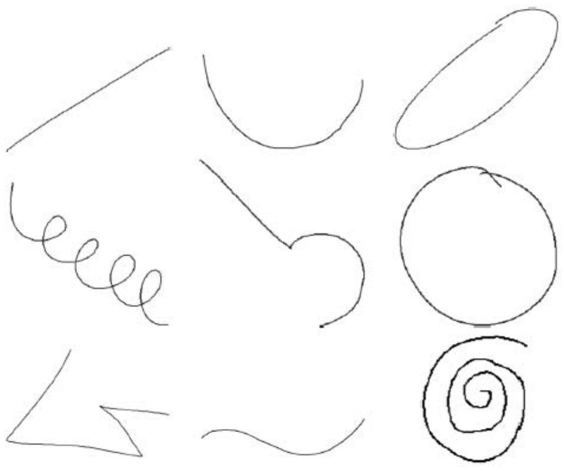
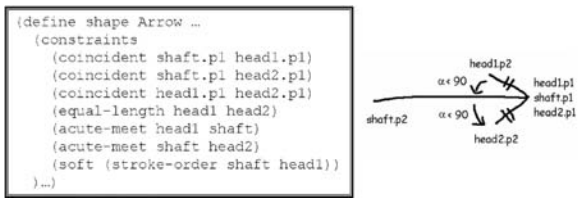


Figure 1: *Low-level primitive recognition.*

1.7. Geometric/High-Level Recognition

Once low-level primitives have been identified, they can then be combined by high-level systems such as [Gro96, HD04]. See Figure 2. Systems such as these typically allow for higher level symbols to be defined using a symbol description language. Symbols are defined using a set of geometric and spatial constraints between lower-level primitives. In addition, some languages (like LADDER [HD03]) allow for additional properties such as editing commands and display options to be defined within the shape description. At the higher-level, domain context can also be used to resolve ambiguity in the lower-level interpretations [AD04].



```
(define shape Arrow ...
  (constraints
    (coincident shaft.p1 head1.p1)
    (coincident shaft.p1 head2.p1)
    (coincident head1.p1 head2.p1)
    (equal-length head1 head2)
    (acute-meet head1 shaft)
    (acute-meet shaft head2)
    (soft (stroke-order shaft head1))
  )...)
```

Figure 2: *High-level geometric recognition.*

1.8. Vision Techniques

We will also discuss vision-based approaches to sketch recognition during this tutorial. Unlike many of the stroke-based methods mentioned previously, vision-based approaches focus solely on the pixel values of a sketch and do not rely on timing or speed data. While vision-based methods have their advantages, there are also many disadvantages to consider. For example, vision-based methods do not handle changes in symbol rotation without explicitely defining templates for all possible orientations. In this tutorial, we will discuss some common vision-based techniques [Olt07, OD09, SvdP06].

1.9. Combined Techniques

Gladder is a first step towards integrating different recognition techniques into a unified framework [CH08]. Specifically, Gladder combines recognition of single-stroke Rubine gestures and single-stroke Paleosketch primitives. This introductory work simply determines which algorithm should classify an example stroke by comparing a covariance-based distance metric with an empirically determined threshold.

2. Interfaces and Usage

During this tutorial we will discuss how and when sketch recognition can be used as the input layer in a variety of multimodal applications [Bol80, Kai05]. Certain domains such as education, design, and engineering can greatly benefit from sketch recognition. We will explain when a sketch based interface is beneficial compared to a more traditional keyboard/mouse interaction. We will discuss our experiences in creating sketch applications focusing on when we succeeded and also when an application failed to be embraced.

2.1. Domains

Primarily our experience in creating sketch recognition based application falls in the domains of education, engineering and early design [FUL*08, LM95, LZ04, CJM*97, HN04, LTC00, LNHL00]. We will discuss and demonstrate examples from each domain. In education, systems exist to teach a variety of language skills; in engineering we have worked in the civil and electrical engineering domains. Design has greatly benefited from sketch recognition's ability to take quickly drawn sketches and build working prototypes.

2.2. Paper-based UIs

Recently many have experimented with using techniques associated with sketch recognition - the low-level and high-level recognition algorithms - and applied those to sketches that were created with a traditional pen and paper. It needs to be pointed out that we are not discussing the digitizing pen systems of the like that are built by Anoto [Ano02]. Researchers have developed techniques that allow sketches that have been created using pen and paper to be recognized and have computation performed on them [RH08]. Advancements in this area will make sketch recognition available to a large repository of historical paper sketches that exist.

3. Conclusion

We look forward to giving a spirited tutorial which will focus on not only how to accomplish sketch recognition but focus on when sketch recognition makes sense for application developers. Sketch recognition was until recently unfeasible given the expensive nature of the input hardware needed and the computational resources that the algorithms demanded. With the rise in availability of tablet PC's and the increase in CPU performance and size of memory sketch recognition applications are becoming more feasible and have begun to tentatively started to appear in education, engineering, design and military environments. A tutorial on this subject was recently given at AAAI, which focused on the algorithms and low-level aspects of the technology. We want this tutorial to focus on the graphic nature of the technology and the role sketch recognition can play as a user interface in a variety of applications.

References

[AAS*04] ANDERSON R., ANDERSON R., SIMON B., WOLFMAN S., VANDEGRIFT T., YASUHARA K.: Experiences with a tablet pc based lecture presentation system in computer science courses. In *Proc. SIGCSE '04* (2004).

[AD04] ALVARADO C., DAVIS R.: Sketchread: A multi-domain sketch recognition engine. In *Proceedings of UIST '04* (2004), pp. 23–32.

[Ano02] ANOTO: *Development Guide for Service Enabled by Anoto Functionality*. Tech. rep., Anoto AB, 2002.

[Bol80] BOLT R. A.: Put-that-there: Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques* (1980), pp. 262–270.

[CH08] COREY P., HAMMOND T.: Gladder: Combining gesture and geometric sketch recognition. *23rd Annual AAAI Conference on Artificial Intelligence: Student Abstracts* (2008).

[CJM*97] COHEN P. R., JOHNSTON M., MCGEE D. R., OVIATT S. L., PITTMAN J., SMITH I., CHEN L., CLOWI J.: Quickset: Multimodal interaction for distributed applications. In *Proceedings of Mutimedia '97* (1997), ACM Press, pp. 31–40.

[DG96] DO E. Y.-L., GROSS M. D.: Drawing as a means to design reasoning. *AI and Design* (1996).

[DP73] DOUGLAS D., PEUCKER T.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization 10*, 2 (1973), 112–122.

[FUL*08] FORBUS K., USHER J., LOVETT A., LOCKWOOD K., WETZEL J.: Cogsketch: Open-domain sketch understanding for cognitive science research and for education. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2008).

[Gro96] GROSS M. D.: The electronic cocktail napkin - a computational environment for working with design diagrams. *Design Studies 17* (1996), 53–69.

[HD03] HAMMOND T., DAVIS R.: LADDER: A language to describe drawing, display, and editing in sketch recognition. In *Proceedings of the 2003 Internaltional Joint Conference on Artificial Intelligence (IJCAI-03)* (Acapulco, Mexico, 2003).

[HD04] HAMMOND T., DAVIS R.: Automatically transforming symbolic shape descriptions for use in sketch recognition. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)* (San Jose, CA, 2004), pp. 450–456.

[Her76] HEROT C.: Graphical input through machine recognition of sketches. In *Proceedings of the 3rd Annual Conference on Computer Graphics and Interactive Techniques* (1976), pp. 97 – 102.

[HN04] HSE H. H., NEWTON A. R.: Recognition and beautification of multi-stroke symbols in digital ink. In *Making Pen-Based Interaction Intelligent and Natural* (Menlo Park, California, October 21-24 2004), AAAI Fall Symposium, pp. 78–84.

[Kai05] KAISER E. C.: Multimodal new vocabulary recognition through speech and handwriting in a whiteboard scheduling application. In *IUI '05: Proceedings of the 10th international conference on intelligent user interfaces* (New York, NY, USA, January 2005), ACM Press, pp. 51–58.

[KK06] KIM D., KIM M.-J.: A curvature estimation for pen input segmentation in sketch-based modeling. In *Computer-Aided Design* (2006), pp. 238–248.

[LLRM00] LONG A. C., LANDAY J. A., ROWE L. A., MICHIELS J.: Visual similarities of pen gestures. In *Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems* (2000).

[LM95] LANDAY J. A., MYERS B. A.: Interactive sketching for the early stages of user interface design. In *Proceedings of CHI '95: Human Factors in Computing Systems* (May 1995), pp. 43–50.

[LNHL00] LIN J., NEWMAN M. W., HONG J. I., LANDAY J.: DENIM: Finding a tighter fit between tools and practice for web site design. In *CHI Letters: Human Factors in Computing Systems, CHI 2000* (2000), pp. 510–517.

[LTC00] LANK E., THORLEY J. S., CHEN S. J.-S.: An interactive system for recognizing hand drawn UML diagrams. In *Proceedings for CASCON 2000* (2000), p. 7.

[LZ04] LAVIOLA J., ZELEZNIK R.: Mathpad2: A system for the creation and exploration of mathematical sketches. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 23(3)* (2004).

[OD09] OUYANG T. Y., DAVIS R.: Visual recognition of sketched symbols. In *IUI Workshop on Sketch Recognition* (2009).

[Olt07] OLTMANS M.: *Envisioning Sketch Recognition: A Local Feature Based Approach to Recognizing Informal Sketches*. PhD thesis, Massachusetts Institute of Technology, 2007.

[PH08] PAULSON B., HAMMOND T.: Paleosketch: Accurate primitive sketch recognition and beautification. In *IUI (Intelligent User Interfaces)* (2008).

[RH08] RAJAN P., HAMMOND T.: From paper to machine: Extracting stokes from images for use in sketch recognition. In *Eurographics 5th Annual Workshop on Sketch-Based Interfaces and Modeling* (Annecy, France, June 2008).

[Rub91] RUBINE D.: Specifying gestures by example. In *Computer Graphics* (1991), vol. 25(4), pp. 329–337.

[Sau90] SAUND E.: Symbolic construction of a 2-d scale-space image. *Transactions on Pattern Analysis and Machine Intelligence 12*, 8 (Aug 1990), 817–830.

[SSD01] SEZGIN T. M., STAHOVICH T., DAVIS R.: Sketch based interfaces: Early processing for sketch understanding. In *The Proceedings of 2001 Perceptive User Interfaces Workshop (PUI'01)* (Orlando, FL, November 2001).

[SvdP06] SHARON D., VAN DE PANNE M.: Constellation models for sketch recognition. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2006).

[WEH08] WOLIN A., EOFF B., HAMMOND T.: Shortstraw: A simple and effective corner finder for polylines. In *Eurographics 2008 - Sketch-Based Interfaces and Modeling (SBIM)* (2008).

[WWL07] WOBBROCK J., WILSON A., LI Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of UIST* (2007).

[YC03] YU B., CAI S.: A domain-independent system for sketch recognition. In *Computer graphics and interactive techniques in Australasia and South East Asia* (2003).