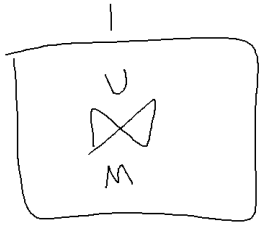
A Sketch Recognition Interface that Recognizes Hundreds of Shapes in Course-of-Action Diagrams

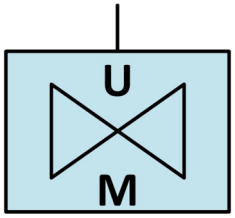
Tracy Hammond
Drew Logsdon
Josh Peschel
Josh Johnston
Paul Taele
Aaron Wolin
Brandon Paulson

Sketch Recognition Lab
Department of Computer Science
and Engineering
Texas A&M University
TAMU 3112
College Station, TX, 77843, USA
hammond@cse.tamu.edu

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.
ACM 978-1-60558-930-5/10/04.

Abstract
Sketch recognition is the automated recognition of hand drawn diagrams. Military course-of-action (COA) diagrams are used to depict battle scenarios. The domain of military course of action diagrams is particularly interesting because it includes tens of thousands of different geometric shapes, complete with many additional textual and designator modifiers. Existing sketch recognition systems recognize on the order of at most 20 different shapes. Our sketch recognition interface recognizes 485 different freely drawn military course-of-action diagram symbols in real time, with each shape containing its own elaborate set of text labels and other variations. We are able to do this by combining multiple recognition techniques in a single system. When the variations (not allowable by other systems) are factored in, our system is several orders of magnitude larger than the next biggest system. On 5,900 hand-drawn symbols drawn by 8 researchers, the system achieves an accuracy of 90% when considering the top 3 interpretations and requiring every aspect of the shape (variations, text, symbol, location, orientation) to be correct.





Keywords
Sketch recognition, pen-based input, course-of-action diagrams

ACM Classification Keywords
H5.2. Input devices and strategies; I.7.5. Graphics recognition and interpretation

General Terms
Design; Human Factors

Introduction

Drawing with pen and paper is one of the most natural and common methods for user interaction that exists. Sketch recognition is the automated understanding of freehand-sketched input on a screen. The recognition of hand drawings has a variety of uses, including functioning as front ends for Computer-Aided Design systems, providing automatic correction or understanding of diagrams for immediate feedback in educational settings, functioning as alternative inputs for small keyboard-less devices (such as Palm Pilots), or providing as a gestural interface.

Sketch recognition attempts to recognize the intent of the user while allowing the user to draw in an unconstrained manner. This permits the user not having to spend time being trained how to draw on the system, nor will the system need to be trained on how to recognize each user's particular drawing style. Our system is built on artificial intelligence techniques that aim to recognize diagrams as closely as possible to a human's understanding of said diagrams.

As pen-based input devices have become more common, sketch recognition systems are being developed for many domains such as mechanical engineering [1], UML class diagrams [4], architecture [3], GUI design [6], virtual reality [2], course-of-action diagrams [7], and many others. However, these systems tend to have on the order of 10-20 shapes in their domain (21 in the case of [7]), and they often require users to draw each shape with a specific gesture, and do not allow users to draw freely, as is the case in [7].

Military commanders use course-of-action (COA) diagrams to plan field operations. Currently commanders draw COAs by hand on a map for planning purposes, and then these diagrams are entered into a computer for purposes of simulation and plan communication. The purpose of this project is to develop an application that allows commanders to hand-draw directly on a map shown utilizing a high-end tablet display, and use sketch recognition so that the computer can have a human-like understanding of the content of the commander's drawings.

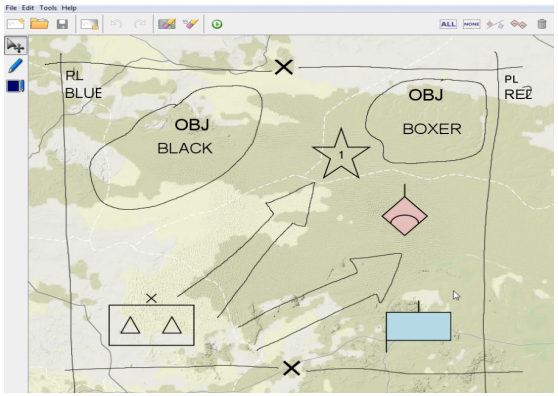


Figure 1: Course-of-Action Diagram Interface.

Current Recognition Methods

Sketch recognition techniques have generally fallen into one of three 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. Template-matching techniques allow drawing style flexibility, but it requires a different template to be created for every visual variation, an impractical task for many variable shapes, such as an arrow. Free-sketch recognition allows users to draw shapes as they would naturally, but many current techniques have very low accuracies and usually require an enormous amount of domain-level tweaking to make them usable.

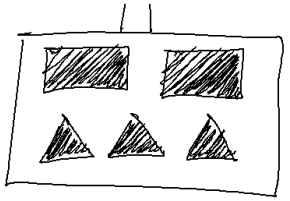
Current systems usually focus on one of three different techniques for recognition: gesture, vision, or geometrical. Gesture recognition technologies look at the path of the pen and order of the items drawn to help recognize diagrams; stroke path features, hidden Markov models (HMMs), and Bayesian Networks are some approaches used to recognize gestures. Vision recognition methods depend as Hausdorff template matching. Geometric recognition methods break down strokes into their primitive components and combine them into shapes by looking at perceptually important geometric properties and constraints based on Gestalt principles. In order to recognize a larger number of symbols while still achieving a high accuracy rate, we use a combination of gesture, vision, and geometrical recognition techniques.

Recognizing Course-of-Action Diagrams

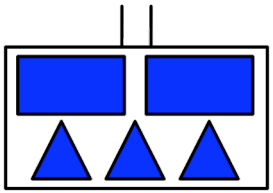
The automated recognition of course-of-action diagrams can be used to simulate battlefield scenarios using hand-drawn COA diagrams as input to inform commanders of the potential outcomes of various battle scenarios. We have developed a multi-platform system that works on Mac, Linux, and Windows to recognize 536 different hand-drawn COA symbols (although we only collected examples of 485 symbols for testing). On 5,900 testing examples hand-drawn by 8 different users, the system achieves an accuracy of over 84% when looking at only the top interpretation, and an accuracy of 90% when looking at the top 3 interpretations. Several examples in the dataset are impossible to distinguish without extra information. (For example, a phase line is represented by a stroke with or without a "PL NAME" label nearby, or just a stroke, and it may or may not have an echelon attached to it. Similarly, a boundary line is represented by a stroke with or without an echelon attached. Thus, a lone stroke could be either a phase line or a boundary line.)

In order to achieve this level of recognition, the system uses and combines a large number of gesture, vision, and geometric recognizers, of which many of the algorithms, and certainly the combination methods, were developed in-house. The overall system includes recognizers for 19 different primitives, combines results from 6 different corner recognizers, uses new methods based on neural networks, LADDER geometric principles [5], shape versus text recognizers, perception-based algorithms, and many other methods.

We now present a brief process overview of our recognition system for COA symbols. Users hand-sketch input into the drawing panel. The recognizer first looks for phase and boundary lines before sending the sketch to corner and low-level primitive recognition using PaleoSketch (examples of primitives include rectangles,



ellipses, and diamonds). The handwriting recognition and sub-unit stage employs a low-level primitive context, since handwriting occurs only in a few locations relative to the primitive shapes. The mid-level shape recognizer (i.e. shapes that are not low-level primitives, but whose complexity is minimal) is then called. Finally, the high-level recognizer finds shapes assembled from low-level primitives, text, and mid-level shapes. Shapes that do not get recognized immediately by the high-level algorithm may be recognized as arrows. After the sketch is recognized as a given symbol, a corresponding symbol identification code (SIDC) is assigned, which associates the recognized COA symbol with the semantic information it codifies. Figures 2, 3, and 4 present hierarchical recognition examples for a unit symbol, a tactical graphic with handwriting, and a decision graphic.



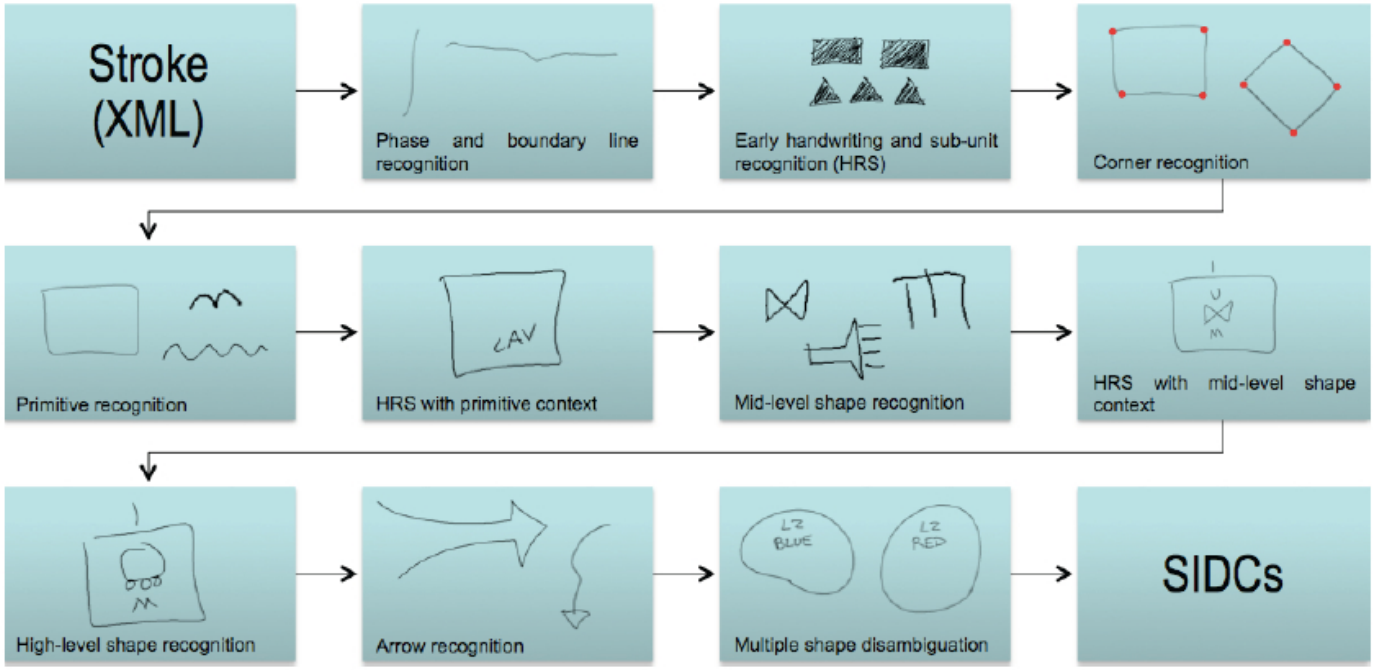
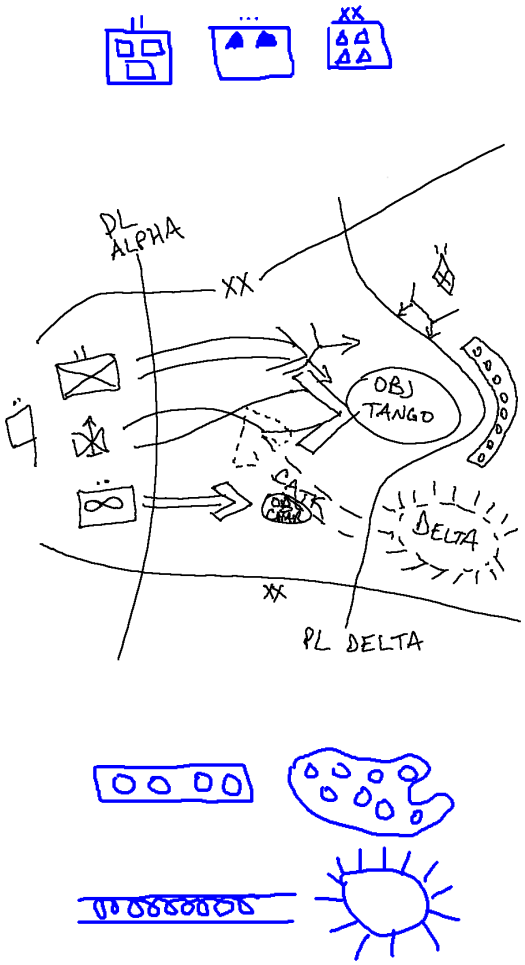


Figure 2. Graphical Overview of the Recognition Processes for a COA Symbol.



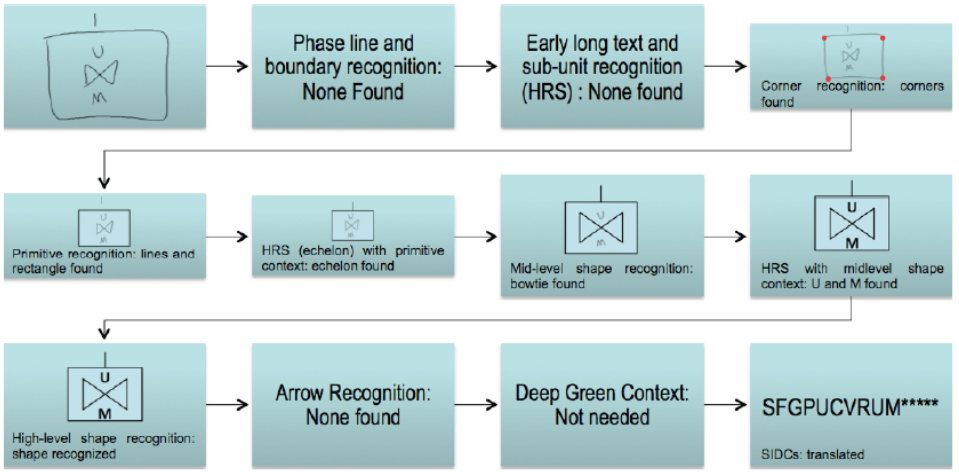


Figure 3. Graphical Overview of for the Recognition Process of a COA Unit Symbol.

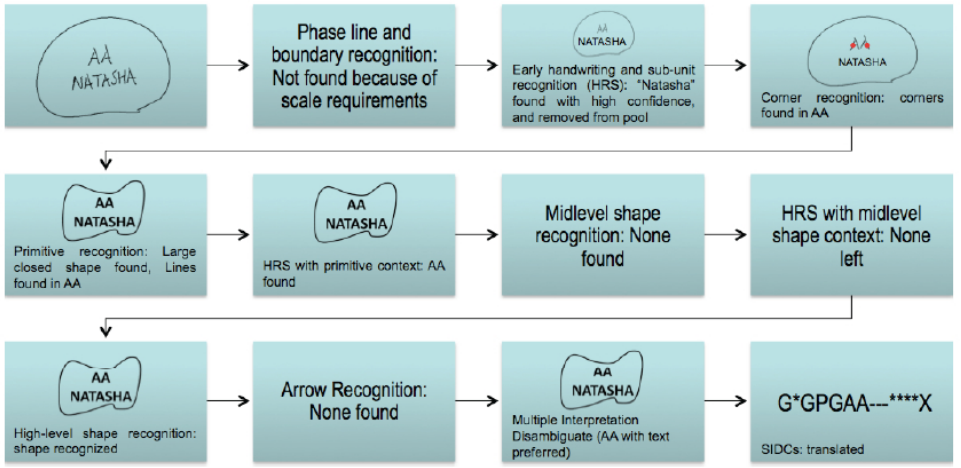


Figure 4. Graphical Overview of for the Recognition Process of a COA Tactical Graphic Symbol.

Interface

An interface for the sketch recognition system has been implemented. A blank drawing panel with a map background image is given to the user to provide the opportunity to input freehand sketches of COA diagrams. Users draw symbols one at a time, and by pausing, they signal that recognition should occur. Once the user draws a particular diagram, the system first processes the input, then computes a list of the top candidates of what was drawn, and finally displays the top candidates in a single-column list near the sketched input for the user to select. The interface can continue to allow the user to input COA symbols until the user has completed the diagram. Full editing functionality, including cut, copy, paste, and delete is provided to the user for both the original strokes and the recognized COA symbols.

Conclusion

This paper describes a sketch recognition system that recognizes military course of action diagrams. The sketch recognition system recognizes 485 different military course-of-action diagram symbols, with each shape containing its own elaborate set of text labels and other variations. Even without the variations and text this is well over an order of magnitude more symbols than the next largest system. When one factors in the variations (not allowable by other systems), this system is several orders of magnitude larger than the next biggest system. On 5,900 hand-drawn symbols drawn by 8 researchers, the system achieves an accuracy of 90% when considering the top 3 interpretations and requires every aspect of the shape (variations, text, symbol, location, orientation) to be correct.

Acknowledgements:

The authors would like to acknowledge Christine Alvarado, Paul Corey, Daniel Dixon, Brian Eoff, Marty Field, Robert Graham, Brandon Kaster, Walter Moriera, Manoj Prasad, Pat Robinson, Cassandra Rutherford, Metin Sezgin, and Yuxiang Zhu, in their participation in and discussion of the ideas of this work. This work is supports in part by NSF grants 0757557 and 0744150.

Works Cited

[1] Alvarado, C. 2000. A natural sketching environment: Bringing the computer into early stages of mechanical design, MIT Master's Thesis.

[2] Do, E.Y.L. 2001. VR Sketchpad - create instant 3d worlds by sketching on a transparent window. CAAD Futures 2001, Bauke de Vries, Jos P. van Leeuwen, Henri H. Achten (eds), 161–172.

[3] Gross, M., Zimring, C., and Do, E.Y.L. 1994. Using diagrams to access a case library of architectural designs. In J.S. Gero and F. Sudweeks, editors, Artificial Intelligence in Design '94, 129–144. Kluwer Academic Publishers.

[4] Hammond, T. and Davis, R. 2002. Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams. AAAI Symposium on Sketch Understanding.

[5] Hammond, T. 2007. LADDER: A Perceptually-Based Language to Simplify Sketch Recognition User Interfaces Development, MIT PhD Thesis.

[6] Lecolinet, E. 1998. Designing GUIs by sketch drawing and visual programming. In Proceedings of the International Conference on Advanced Visual Interfaces (AVI 1998), 274–276, ACM Press.

[7] Pittman, J., Smith, I., Cohen, P., Oviatt, S., and Yang, T. 1996. Quickset: A multimodal interface for military simulations. In Proceedings of the 6th Conference on Computer-Generated Forces and Behavioral Representation, 217–224.