# AN ON-LINE SYMBOLIC MATHEMATICS SYSTEM USING HAND-PRINTED TWO-DIMENSIONAL NOTATION*

Frederick W. Blackwell

Robert H. Anderson

The RAND Corporation
Santa Monica, California

## Summary

This paper describes a system that is being developed at The RAND Corporation for the on-line manipulation of symbolic mathematical expressions. The primary input consists of the user's expressions hand-printed on a RAND tablet in ordinary two-dimensional mathematical notation. The system recognizes the characters and interprets the whole expression from the spatial relationships present in accordance with a previously input syntax. The user at the console directs symbolic transformations upon his input expressions by instructing the computer to selectively attempt to apply various rules of mathematics; these rules have been previously entered into the system in the same manner as the expressions. A transformed expression resulting from the application of a rule or group of rules is displayed on the IBM 2250 graphic console. An experimental version of the system is in operation at the present time.

## Introduction

Most programming languages employ linear notation for algebraic formulas not only because it is much easier to implement, but also because the relatively few distinct types of two-dimensional configurations which normally occur can be readily represented in linear form. Familiar examples are the use of the slash for division and the use of the double asterisk or some special character for exponentiation. While the programmer and non-programmer alike adapt to this linear notation, we assume that both would usually prefer a language in which it is possible to write in ordinary two-dimensional mathematical notation. Examples of how this can be done for typewriter-like devices are provided by the work of Klerer and May[1] and Wells[2], and for more general input devices by the work of Anderson[3] and Bernstein and Williams[4]. The latter systems are complicated by the fact that the user prints his symbols on a RAND tablet or similar device; the characters must be individually recognized, and then the whole formula must be properly interpreted from the spatial relationships present. The development of formula-manipulating languages, in which the data and results are often inherently two-dimensional, has created additional demand (intensified in an on-line environment) for explicit two-dimensional representations in communicating with the computer.

This paper describes a system that is being developed at The RAND Corporation for the on-line manipulation of symbolic mathematical expressions. The primary input is the user's expressions hand-printed on a RAND tablet, and the primary output consists of transformed expressions which appear on the IBM 2250 graphical display console. Both input and output are in two-dimensional mathematical notation. The person at the console directs the transformations upon his expressions. Thus the system provides a kind of "sophisticated scratchpad" for the user.

## System Design

The goals of the system are several. First, it is hoped that it will be a useful mathematical tool for RAND scientists.

*The work described in this paper was supported under Air Force Project RAND and under contract to ARPA.

# GRAPHICAL INPUT THROUGH MACHINE RECOGNITION OF SKETCHES

Christopher F. Herot
Architecture Machine Group, Department of Architecture
Massachusetts Institute of Technology, Cambridge, Massachusetts

A family of programs has been developed to allow graphical input through continuous digitizing. Drawing data, sampled at a high and constant rate, is compressed and mapped into lines and splines, in two and three dimensions. This is achieved by inferring a particular user's intentions from measures of speed and pressure.

Recent experiments have shown that even the most basic inference making cannot rely solely upon knowledge of the user's drawing style, but needs additional knowledge of the subject being drawn, the protocols of its domain, and the stage of development of the user's design. This requirement implies a higher level of machine intelligence than currently exists. An alternate approach is to increase the user's involvement in the recognition process.

Contrary to previous efforts to move from sketch to mechanical drawing without human intervention, this paper reports on an interactive system for graphical input in which the user overtly partakes in training the machine and massaging the data at all levels of interpretation. The initial routines for data compression employ parallel functions for extracting such features as bentness, straightness, and endness. These are planned for implementation in microprocessors.

Results offer a system for rapid (and enjoyable) graphical input with real-time interpretation, the beginnings of an intelligent tablet.

## 1. INTRODUCTION

There are many areas of human endeavor which could benefit from the use of computer aids if there existed an effective means of communicating about these tasks with a machine. While the field of computer graphics arose to fill that need, it has too often added a new level of complexity. In computer-aided design, for instance, the process of "digitizing" is sufficiently cumbersome to delay its application until a relatively complete design has been produced by the human designer. The result is usually more akin to computer-aided evaluation or manipulation than to computer-aided design. The research described here is motivated by the desire to involve the computer in the early stages of the design process, where the feedback generated by the machine can be most useful. The medium chosen is free-hand sketching, as done with pencil and paper, as could be done at a data tablet. A machine is postulated to be looking on while the user is sketching. It could make inferences not only about the meaning of the sketch but also about the user's attitudes toward, and uncertainties about, his design.

This approach offers its own unique set of problems and solutions, since the data available to the machine are at once plentiful and incomplete. While the ultimate implementation assumes a near-human intelligence on the part of the machine, far off in time, there are many interesting things to be learned along the way.

Our previous experiments[1, 2, 3, 4] have been directed toward the creation of a "passive" input system which would make fairly complete inferences about a sketch while requiring a minimum of intervention from the user. The programs we used in these projects involved many levels of interpretation. From the 100 pen positions sampled each second, the machine would have to find lines, curves, and corners, building a description of a two-dimensional object. This description could be interpreted further, possibly as a three-dimensional description.

The following sections depict three experiments in computer processing of sketches. HUNCH, described in the next section, was directed toward answering the following question: Does there exist a syntax of sketching, something which could be processed independently of the embedding semantics? Could a machine make useful interpretations of a sketch without employing a knowledge of the subject domain? The mixed results of that experiment led to the investigation described in section 3, a rather ambitious effort to make use of architectural knowledge in recognizing a sketch. Finally, section 4 reports on current work which places more emphasis on user involvement in the input process.

The work reported herein has been sponsored by the Office of Naval Research, grant number N00014-67-A-0204-0074 and by the National Science Foundation, Division of Computer Research, grant number DCR74-20974-A01.

## 2. THE HUNCH SYSTEM

HUNCH is a name given to a set of FORTRAN programs which were designed to process freehand sketches drawn with a data tablet or light pen. Each program performs a different level of interpretation, storing its output in a file where it can be used as input by the other programs. Facilities exist to display and manipulate various stages of interpretation. The relationships among the programs are illustrated below.

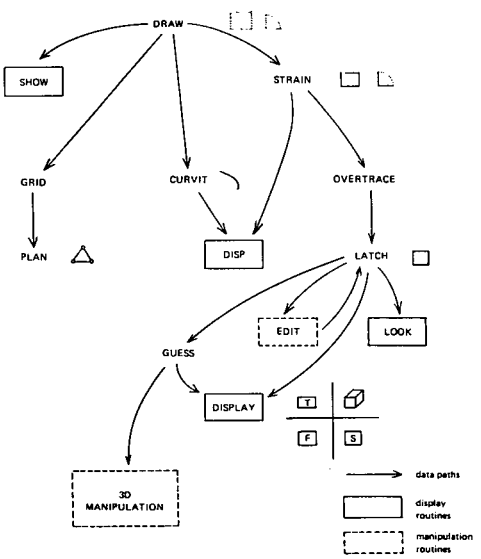



**Fig. 1 - The HUNCH System**

# SKETCHPAD
# A MAN-MACHINE GRAPHICAL COMMUNICATION SYSTEM*

*Ivan E. Sutherland*
*Consultant, Lincoln Laboratory***
*Massachusetts Institute of Technology*

## I. INTRODUCTION

The Sketchpad system makes it possible for a man and a computer to converse rapidly through the medium of line drawings. Heretofore, most interaction between man and computers has been slowed down by the need to reduce all communication to written statements that can be typed; in the past, we have been writing letters to rather than conferring with our computers. For many types of communication, such as describing the shape of a mechanical part or the connections of an electrical circuit, typed statements can prove cumbersome. The Sketchpad system, by eliminating typed statements (except for legends) in favor of line drawings, opens up a new area of man-machine communication.

## AN INTRODUCTORY EXAMPLE

To understand what is possible with the system at present let us consider using it to draw the hexagonal pattern in Figure 4. We will issue specific commands with a set of push buttons, turn functions on and off with switches, indicate position information and point to existing drawing parts with the light pen, rotate and magnify picture parts by turning knobs, and observe the drawing on the display system. This equipment as provided at Lincoln Laboratory's TX-2 computer[1] is shown in Figure 1. When our drawing is complete it may be inked on paper, as were all the drawings in this paper, by a PACE plotter.[15]

If we point the light pen at the display system and press a button called "draw," the computer will construct a straight line segment which stretches like a rubber band from the

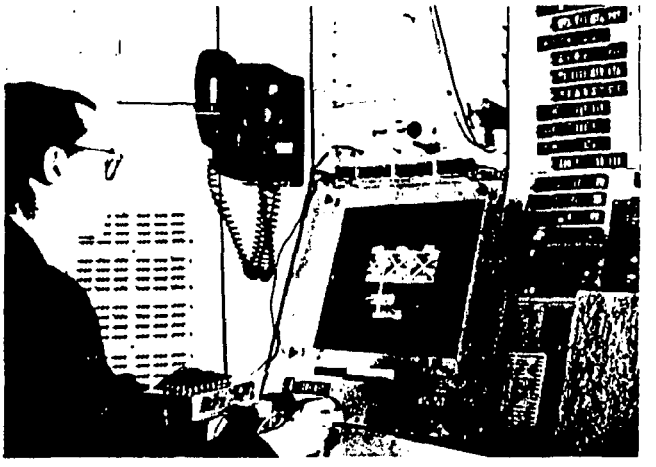


Figure 1. TX-2 operating area—Sketchpad in use. On the display can be seen part of a bridge similar to those of Figure 15. The Author is holding the light pen. The push buttons "draw," "move," etc., are on the box in front of the Author. Part of the bank of toggle switches can be seen behind the Author. The size and position of the part of tne total picture seen on the display are controlled by the four black knobs just above the tables.

* This paper is based in part on a thesis submitted tothe Department of Electrical Engineering, M.I.T., in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

** Operated with the support of the U.S. Army, Navy,and Air Force.

# Streaming Algorithms for Line Simplification

Mohammad Ali Abam*
Department of Computing Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, the Netherlands
m.a.abam@tue.nl

Mark de Berg†
Department of Computing Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, the Netherlands
mdberg@tue.nl

Peter Hachenberger†
Department of Computing Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, the Netherlands
phachenb@win.tue.nl

Alireza Zarei‡
Computer Engineering Department, Sharif University of Technology and IPM School of Computer Science, P.O. Box 11365-9517, Tehran, Iran
zarei@mehr.sharif.edu

## ABSTRACT

We study the following variant of the well-known line-simplification problem: we are getting a possibly infinite sequence of points $p_0, p_1, p_2, \ldots$ in the plane defining a polygonal path, and as we receive the points we wish to maintain a simplification of the path seen so far. We study this problem in a streaming setting, where we only have a limited amount of storage so that we cannot store all the points. We analyze the competitive ratio of our algorithms, allowing resource augmentation: we let our algorithm maintain a simplification with $2k$ (internal) points, and compare the error of our simplification to the error of the optimal simplification with $k$ points. We obtain the algorithms with $O(1)$ competitive ratio for three cases: convex paths where the error is measured using the Hausdorff distance (or Fréchet distance), $xy$-monotone paths where the error is measured using the Hausdorff distance (or Fréchet distance), and general paths where the error is measured using the Fréchet distance. In the first case the algorithm needs $O(k)$ additional storage, and in the latter two cases the algorithm needs $O(k^2)$ additional storage.

*MAA was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 612.065.307.

†MdB and PH were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301.

‡AZ was supported by Sharif University and a grant from IPM school of CS (no. CS1385-2-01)


*SCG'07,* June 6–8, 2007, Gyeongju, South Korea.


## Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and problem complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

## General Terms

Algorithms, Theory

## Keywords

line simplification, streaming algorithm, Frechet distance, Hausdorff distance

## 1. INTRODUCTION

*Motivation.* Suppose we are tracking one, or maybe many, moving objects. Each object is equipped with a device that is continuously transmitting its position. Thus we are receiving a stream of data points that describes the path along which the object moves. The goal is to maintain this path for each object. We are interested in the scenario where we are tracking the objects over a very long period of time, as happens for instance when studying the migratory patterns of animals. In this situation it may be undesirable or even impossible to store the complete stream of data points. Instead we have to maintain an approximation of the input path. This leads us to the following problem: we are receiving a (possibly infinite) stream $p_0, p_1, p_2, \ldots$ of points in the plane, and we wish to maintain a simplification (of the part of the path seen so far) that is as close to the original path as possible, while using not more than a given (fixed) amount of available storage.

*Related work.* The problem described above is a streaming version of line simplification, one of the basic problems in GIS. In a line simplification problem one is given a polygonal path $P := p_0, p_1, \ldots, p_n$ in the plane, and the goal is

EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2007)
M. van de Panne, E. Saund (Editors)

# Scribbles to Vectors: Preparation of Scribble Drawings for CAD Interpretation

A. Bartolo, K. P. Camilleri, S. G. Fabri, J. C. Borg and P. J. Farrugia

Faculty of Engineering, University of Malta, Malta

**Abstract**

*This paper describes the work carried out on off-line paper based scribbles such that they can be incorporated into a sketch-based interface without forcing designers to change their natural drawing habits. In this work, the scribbled drawings are converted into a vectorial format which can be recognized by a CAD system. This is achieved by using pattern analysis techniques, namely the Gabor filter to simplify the scribbled drawing. Vector line are then extracted from the resulting drawing by means of Kalman filtering.*

Categories and Subject Descriptors (according to ACM CCS): I.4.3 [Image Processing and Computer Vision]: Filtering I.4.7 [Image Processing and Computer Vision]: Texture

## 1. Introduction

Unconstrained sketching in early design is important. This fact is evident by the number of researchers that strive to create sketch-based interfaces that give the user the maximum freedom possible, aiming at obtaining a transparent, easy to use interface. The trend in these interfaces is to give the designer the impression that the drawing is made using traditional pen and paper, possibly through the use of digital pens and Tablet PCs. However, the freedom and flexibility available to designers when using traditional pen and paper remains unparalleled and this is evident in the lower adoption rates of Tablet PCs [NM04]. Using pen-and-paper, designers may benefit from quick, uninterrupted sketching, allowing the designer to focus on the design of the object rather than the interactions required with the interface system. This in turn makes the designers more likely to explore different design solutions, hence increasing the creativity of their designs [SL97].

This flexibility has a price. Paper is a passive medium and through paper alone, the designer cannot obtain the 3D rendered models that may be obtained through CAD systems. To obtain such models, the designer must redraw the sketch, either using the CAD tool directly, or by using an intermediary sketch based interface. Hence the need of an interpretation technique that may integrate the paper-based drawing with CAD systems. In our previous work [FBCG06], [BCFB06], we have demonstrated a possible solution, using a prescribed sketching language to facilitate the interpretation of paper-based drawings. This work further improves our previous interface by allowing the designer to sketch the object profile using multiple strokes or scribbles; thereby increasing the designer's drawing freedom. This paper focuses on the simplification of the scribbled profile such that the scribble may be represented by vector data which may be imported into a CAD system. The designer is limited to drawing the profile of the object since this paper does not tackle the issues involved in shading strokes.

The rest of the paper is organized as follows: Section 2 gives a brief review of the methods used to interpret scribbled drawings, Section 3 describes the use of Gabor filtering techniques for perceptual simplification, Section 4 describes how the Kalman filter framework may be used to extract a vectorial representation of the simplified scribble, Section 5 presents the results obtained and the evaluation of the proposed interpretation techniques while Section 6 concludes the paper with a discussion on the work.

## 2. Background

Given drawing freedom, designers would draw initial sketches using over-traced scribbles. Interpretation of these scribbles requires the grouping of the over-traced line strokes into a single line vector. Vectorization of neat sketches and technical drawings is a well researched

*Pattern Recognition*, Vol. 26, No. 3, pp. 419–431, 1993
Printed in Great Britain

0031-3203/93 $6.00 + .00
Pergamon Press Ltd


# NORMALIZING AND RESTORING ON-LINE HANDWRITING

WACEF GUERFALI and RÉJEAN PLAMONDON†
Laboratoire Scribens, Département de génie électrique et de génie informatique, École Polytechnique de Montréal, Campus de l'université de Montréal, 2900 Edouard-Montpetit, Case postale 6079, Succursale A, Montréal (Québec), Canada H3C 3A7

**Abstract**—Preprocessing and normalization techniques for on-line handwriting analysis are crucial steps that usually compromise the success of recognition algorithms. These steps are often neglected and presented as solved problems, but this is far from the truth. An overview is presented of the principal on-line techniques for handwriting preprocessing and word normalization, covering the major difficulties encountered and the various approaches usually used to resolve these problems. Some measurable definitions for handwriting characteristics are proposed, such as baseline orientation, character slant and handwriting zones. These definitions are used to measure and quantify the performance of the normalization algorithms. An approach to enhancing and restoring handwriting text is also presented, and an objective evaluation of all the processing results.

Feature extraction Normalization On-line Handwriting Preprocessing evaluation Restoration

## 1. INTRODUCTION

With the introduction of electronic tablets (during the 1960s), interest in handwriting recognition focused on research and development to produce interactive technologies like computer keyboard emulators, form filling applications and computer-aided design systems.

A wide range of techniques are used in handwriting recognition, however all the systems developed generally share a common processing sequence, which includes data acquisition, preprocessing, recognition and postprocessing. The nature of data acquisition determines the class of the system (on-line or off-line). It is important, at this stage, to distinguish between on-line and off-line systems. On-line (or dynamic) systems record sampled information about the state of the pentip. This information enables the construction of the discrete functions $X_s(t)$ and $Y_s(t)$, which are the coordinates of the pentip movement sampled at a fixed time ($t$). Off-line (or static) systems use only the component[15] images; information is not available on either the pen movement or on the order of the components.

The preprocessing step used in either on-line or off-line systems presents the first difficulty in handwriting recognition. Many characteristics used in this first stage of processing are quite subjective. Measurable definitions of characteristics such as slant, baseline, zones, etc. are simply not available. This problem makes the results of preprocessing algorithms neither quantifiable nor comparable.

This paper covers three major aspects of the preprocessing and normalizing of *on-line* handwriting. First, an overview of the preprocessing techniques used in on-line systems is presented. Techniques to reduce the amount of data, eliminate imperfections and normalize handwriting are described. Second, normalizing and restoring techniques of handwritten words are treated. Measurable definitions are provided, and new algorithms that detect and correct orientation, slant and handwriting zones are described. Third, an objective evaluation scheme for handwriting preprocessing is proposed. All the algorithms developed are also evaluated and results are presented.

## 2. PREPROCESSING

On-line systems use preprocessing techniques as a step to simplify the tasks of shape recognition algorithms. This step has a great influence on subsequent processing, and a real impact on the recognition rate.[1,2] Preprocessing techniques for on-line handwriting can be divided into three groups. They are used first of all to reduce the amount of information (filtering and dot reduction); second, to eliminate imperfections (smoothing, wild point correction, hook removal and component connection); and third, to normalize handwriting (deskewing, baseline drift correction, size and component-length normalization).

### 2.1. *Reduction of information*

Electronic tablets enable the sampling of information about pentip position and status at a fixed sampling frequency. The amount of data transmitted by electronic tablets is usually reduced by eliminating duplicate points and redundant information. This processing step is aimed principally at minimizing the amount of data and reducing recognition time.

*Filtering.* Filtering, as described by many authors,[3–8] consists in eliminating consecutive points spaced by an

† Author to whom all correspondence should be addressed.

# Robust Sketched Symbol Fragmentation using Templates

Heloise Hse
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720, U.S.A.
+1 510 642 2481
hwawen@eecs.berkeley.edu

Michael Shilman
Microsoft Research
One Microsoft Way
Redmond, WA 98052
+1 425 722 5853
shilman@microsoft.com

A. Richard Newton
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720, U.S.A.
+1 510 642 5771
newton@eecs.berkeley.edu

## ABSTRACT
Analysis of sketched digital ink is often aided by the division of stroke points into perceptually-salient fragments based on geometric features. Fragmentation has many applications in intelligent interfaces for digital ink capture and manipulation, as well as higher-level symbolic and structural analyses. It is our intuitive belief that the most robust fragmentations closely match a user's natural perception of the ink, thus leading to more effective recognition and useful user feedback. We present two optimal fragmentation algorithms that fragment common geometries into a basis set of line segments and elliptical arcs. The first algorithm uses an explicit template in which the order and types of bases are specified. The other only requires the number of fragments of each basis type. For the set of symbols under test, both algorithms achieved 100% fragmentation accuracy rate for symbols with line bases, >99% accuracy for symbols with elliptical bases, and >90% accuracy for symbols with mixed line and elliptical bases.

## Categories and Subject Descriptors
I.4.6 [**Image Processing and Computer Vision**]: Segmentation – *edge and feature detection.*

## General Terms
Algorithms, Human Factors.

## Keywords
Curve segmentation, perceptual grouping, shape templates, fitting, sketch-based user interface, HCI

## 1. INTRODUCTION
Sketching is a simple and natural mode of expression. It is especially desirable for conceptual design, both on an individual basis and in a collaborative environment. With a sketch-based user interface, one can have the freedom of sketching on paper and the benefit of an electronic design tool [9]. If a sketch system also includes a recognition capability, sketches can be interpreted and augmented with semantics so that they can be edited easily, efficiently searched, and neatened.

There has been a significant amount of research to date in various aspects of sketch-based user interfaces: interactive design tools [12, 13], studies of gestures [15], software toolkits [10], ink beautification [11], and sketch recognition [1, 24]. However, relatively little work has focused on the fragmentation of hand-sketched symbols (e.g. [4, 21, 25]). Fragmentation is a perceptual analysis of ink strokes in which stroke points are clustered into geometrically salient primitives, such as line segments and elliptical arcs. Figure 1 shows an example fragmentation of a sketched square and the ways that the resulting fragmentation can be utilized.

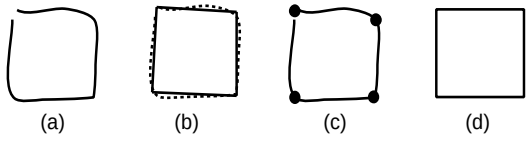



**Figure 1. (a) An initial stroke, (b) its natural fragmentation (overlaying the stroke points), (c) a direct manipulation user interface based on the fragmentation, (d) a beautification of the stroke.**

Fragmentation is a very basic problem, making it widely applicable to intelligent ink manipulation as well as other higher-level digital ink analyses. The structural information that it generates can be useful in the following situations:

- Generating structural descriptions for use in symbol recognition, especially by structural recognizers [4, 17].
- Locating functional points in a symbol, such as the tip of an arrowhead, the four corners in a square, etc. (Figure 1c) This is especially applicable to recognizers that do not perform structural analysis [2, 8, 14, 20, 25], such as statistical recognizers [8, 20, 25].
- Automatically 'neatening' a symbol using the geometric primitives which result from fragmentation (Figure 1b). Further beautification of the symbol can be based on the recognition result or geometric properties such as parallel lines, right angles, etc. (Figure 1d) [11]


*IUI'04*, January 13–16, 2004, Madeira, Funchal, Portugal.

# In Search for an Ideal Computer-Assisted Drawing System

**†Takeo Igarashi, ‡Sachiko Kawachiya, †Satoshi Matsuoka, †Hidehiko Tanaka**
†Dept. of Information Engineering, ‡Dept. of Information Science
The University of Tokyo
Hongo 7-3-1, Bunkyoku, Tokyo, Japan
+81-3-3812-2111 ext 7413
{takeo, tanaka}@mtl.t.u-tokyo.ac.jp, {sachiko, matsu}@is.s.u-tokyo.ac.jp

**ABSTRACT** Diagram drawing with conventional computer-assisted drawing(CAD) editors often tend to take considerable amount of time despite their seeming ease of use. We analyzed the problems of such systems focusing on the problem of cognitive overload, and observed that 1) the necessity of cognitive *planning* process in current CAD system causes the problems and that 2) reducing the overload can lead to fundamental improvement in overall drawing efficiency. We have conducted an experiment to verify these observations by comparing a typical drawing system and our prototype drawing system called *Interactive Beautification*, which combines the ease of freehand drawing and precision of traditional drawing editors by extracting various constraints in input strokes, and generating the desired diagrams automatically. Results show that significant amount of time is spent for cognitive *planning* process, and reduction of such planning time by *Interactive Beautification* can significantly improve the efficiency of CAD.

**KEYWORDS** CAD, Interaction technique, Cognitive workload, Constraint hierarchy, Beautification.

## 1 INTRODUCTION

Computer-assisted, Object-Oriented(OO) drawing editors are now used widely, allowing precise drawings not possible with traditional pen-on-paper drawing techniques. Furthermore considerable research and development have been conducted on creating new and innovative drawing systems resulting in those with abundance of functionalities. Despite such developments, drawing with such systems remains time-consuming and non-trivial, especially for novice or casual users.

The problem is that most research and development efforts on drawing systems have basically been 'feature wars', adding new drawing capabilities and interaction techniques, without deep, profound insights into their usability in practice. Instead we need studies to find "where does the fundamental problem exist" based on the close analysis of drawing processes on practical drawing systems.

Based on such a motivation, we propose an abstract model of drawing to explain the problems with computer-assisted drawing(CAD) systems focusing on the cognitive overload (Suchman, 87)
(Norman, 86), and verify it through several experiments. In the model, we assume that the fundamental problem with CAD lies in considerable requirements imposed on the user's cognitive *planning* process. By *planning* we mean the cognitive process during drawing, where a user plans on the combination of the physical operations available in drawing editors to obtain the desired image. In addition, we confirm through an experiment with our prototype system that the reduction of such cognitive *planning* process can improve the efficiency of CAD.

Our prototype system frees the users of the cognitive *planning* process by automatically generating the desired diagram from freestrokes, but detailed descriptions of the salient technical features of the system itself is not the purpose of this paper; rather, we generalize on the effectiveness of the approaches to remove user's cognitive *planning* process from drawing editors.

This paper is organized as follows. First, we analyze the problems with CAD and propose an abstract model that captures the problems of excessive cognitive workload. Second, we conduct an experiment to quantify the workload. Third, we introduce a new drawing system where the objective is to reduce the workload, and conduct an experiment to quantify its effectiveness versus traditional OO-based

ELSEVIER

Computer-Aided Design 38 (2006) 238–248

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

# A curvature estimation for pen input segmentation in sketch-based modeling

Dae Hyun Kim [a,*], Myoung-Jun Kim [b]

[a] *Institute for Graphic Interfaces, Ewha-SK Bldg. 11-1, DaeHyun, Seodaemun, Seoul Korea*
[b] *Division of Digital Media, Ewha Womans University, Ewha-SK Bldg. 11-1, DaeHyun-dong, Seodaemun, Seoul Korea*

Received 27 January 2005; received in revised form 12 October 2005; accepted 16 October 2005

**Abstract**

A proper segmentation of pen marking enhances shape recognition and enables a natural interface for sketch-based modeling from simple line drawing tools to 3D solid modeling applications; user input is otherwise restricted to draw only one segment per one stroke. In general, the pen marking segmentation is achieved by detecting the points of high curvature-called, segmenting points-and splitting the pen marking at those points. This paper presents a curvature estimation method, which considers only local shape information. The proposed method can therefore estimate curvature on-the-fly while user is drawing on a pen-input display, such as tablet PCs.


*Keywords:* Curvature; Local shape information; Pen-input displays; Segmentation

## 1. Introduction

Given a pen marking $p=\{p_i|i=0,\ldots,n\}$, drawn on a pen-input display, such as LCD tablets integrated into most Tablet PCs, segmenting points can be defined as the points that have high curvature (i.e. $|c|>\tau$, where $c$ is an estimated curvature and $\tau$ is a given threshold). Detected segmenting points lead to a preliminary segmentation, when the pen marking is split at each of them. Therefore, a proper segmentation owes its success, in most cases, to a proper curvature estimation. The inflection points, zeros of the second derivative, are also regarded as segmenting points. However, if needed, it is more advantageous to process them during curve fitting for each segment after the preliminary segmentation; otherwise small fluctuations as shown in Fig. 12 will produce many unnecessary inflection points [10].

Before developing a curvature estimation algorithm for pen-input displays, one needs to know input style; for example, user's drawing skill and the mechanical properties of the input devices. In this paper, the following input data assumption has been considered in the design of a new method to estimate curvature: *Input data is taken from a pen-based input device; therefore, it contains no noise occurring because the input device cannot follow user's drawing.* Therefore, all sampled points are regarded as genuine data, and their coordinates are taken as given. For example, wiggly lines, as those in Fig. 1, are caused by user intention, but not by the noise added performing the pen marking.

*Previous works*: We review two classes of approaches for detecting segmenting points which are directly related to our work; one specialized aspect of our work is that the method should be applicable for pen input devices—for a broader survey of segmenting point detection we refer the reader to [3].

The first class of approaches is to use scale space decomposition; it decomposes the pen marking into multi-level representations by progressively smoothing it with different smoothing functions and picks up one level among them, which can represent user's intention. There are two approaches available within this class: either curvature is evaluated in a selected scale after transforming the input data to the Gaussian scale space [7,8] or estimated curvature is transformed into scale space and one scale is selected [2,10]. One advantage of using scale space decomposition is that it is suitable for noisy data as shown in Fig. 1. However, a long computing time is needed to construct Gaussian scale space. Considering the input assumption that we have made above,

* Corresponding author. Fax: +82 2 3277 3893.
*E-mail addresses:* daek@acm.org (D.H. Kim), mjkim@ewha.ac.kr (M.-J. Kim).

0010-4485//$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2005.10.006

EUROGRAPHICS Workshop on Sketch-Based Interface and Modeling (2007)
M. van de Panne, E. Saund (Editors)

# Implicit Geometric Constraint Detection in Freehand Sketches Using Relative Shape Histogram

J. Pu and K. Ramani

Purdue Research and Education Center for Information Systems in Engineering (PRECISE), Purdue University, West Lafayette IN 47907-2024, USA

**Abstract**

*In order to take advantage of the sketch-based interaction, many methods have been proposed to beautify freehand sketches. Most of these efforts are dedicated to sketch segmentation and recognition, while some important information implied in the sketches, such as geometric constraints, are largely ignored. Thus, the final beautified results by these methods do not fully reflect the true intentions from users. In this paper, a statistical approach called Relative Shape Histogram (RSH) is introduced to detect the implied geometric constraint in sketches. The basic idea arises from such a discovery that the same geometric constraints between two geometric primitives have similar relative shape histograms. By computing the similarity between RSHs, the implicit geometric constraints between two segmented primitives are inferred. To evaluate the performance of the proposed algorithm, a user-based experiment is conducted and the results are presented in this paper.*

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Interaction techniques

## 1. Introduction

Freehand sketches have been widely recognized as an efficient and natural way to communicate ideas between human being and computer. However, correct interpretation of the intent of the users making freehand sketches, i.e., sketch understanding, is still a challenge because freehand sketches are informal, ambiguous, and implicit in comparison to traditional WIMP (Window, Icon, Menu, and Pointer) user interfaces in which each interaction is predefined and there is unique correspondence between an input and the internal interpretation of a computer [Li03]. To address this issue and enable a computer to interpret freehand sketches in a unified and robust way, many sketch beautification methods have been proposed to transform freehand sketches from an informal representation to a formal representation. Most of these efforts focused on two important issues: sketch segmentation [FLK*04, Stahovich04, LS05, NM04, GKS05] and primitive (or composite) recognition [SD05, HN04, KS04, AD04], which are regarded as the two main obstacles that have hindered the development of a system with superior sketch understanding [FLK*04]. Sketch segmentation is the process by which a continuous stream of pen strokes is parsed into a series of constituent geometric primitives that are atomic geometric entities in sketches and can not be further decomposed. Frequently, only lines, circles and arcs are considered as the basic primitives that constitute a user's sketches [FLK*04, Stahovich04, HN04] . Given a segmented portion of a pen stroke, the aim of primitive or composite recognition is to determine the type of geometric entity to which it belongs.

Often, once the primitives are recognized, the freehand sketches can be beautified by assigning primitives with proper parameters. However, such a direct beautification often misses some important information implied in sketches such as geometric constraints, which are widely used in many design related applications, such as drawing programs [KB90], computer aided design [Aldefeld98, BFH*95, FH97] and graphical user interfaces [BD86], to determine the relationship between two objects. To illustrate the disadvantage of direct beautification, two examples are shown in Figure 1, where (b) is the beautified result of (a) and (d) is the beautified result of (c). It is intuitive for users to conclude from the sketches that the two circles have the same center while the lines are parallel. However, due to the fact that the parameters (i.e., center and radius) of each circle are determined separately, their relative relationship such as geometric constraint might change after the beautification operation, thus leading to the conclusion that there is no relationship between the two circles and the lines. In a geometric constraint system, this direct beautification will lead to serious error propagation for which additional efforts [WSH05] are needed. Therefore, a more reliable method is to check their constraint type directly on the basis of the

# Handling Overtraced Strokes in Hand-Drawn Sketches

**Tevfik Metin Sezgin and Randall Davis**
MIT Computer Science and Artificial Intelligence Laboratory
The Stata Center 235
Cambridge MA, 02139
{mtsezgin,davis}@csail.mit.edu

### Abstract

Overtracing is the phenomenon in sketching of repeatedly drawing over previously drawn ink. It is a naturally appearing effect and is especially frequent in domains such as architectural drawings. Existing work in sketch recognition focuses on sketches with non-overtraced strokes. In this paper, we describe a method for generating geometric approximations of overtraced strokes in terms of primitives including lines, non-self-intersecting polylines, ellipses and arcs. Our system generates concise approximations for overtraced strokes at an early stage in sketch processing, making it possible to use these concise descriptions in higher level processing and sketch interpretation systems.

## Introduction

Freehand sketches are inherently informal and messy. Unlike clean, computer generated diagrams, manifestations of abstract shapes are imprecise and highly variable in sketches. One of the challenges in sketch recognition is the ability to support this imprecision and high variability, of which overtracing is an example (Fig. 1).

In a paper studying how architects sketch, Do and Gross describe overtracing as one of the drawing techniques heavily used in practice. They list the functions of overtracing as "selecting or drawing attention to an element; shape emergence, attending to one or another shape interpretation; and shape refinement or adding detail to an abstract or roughed out shape". We want to make it possible for users to employ this heavily used drawing technique in sketch-based interfaces. We do this by generating geometric descriptions of overtraced strokes in terms of common geometric primitives.

Our strategy for handling overtracing is to deal with it early in the recognition process – in the context of model based recognition – before high level interpretations are built. We focus here on generating geometric approximations for single stroke shapes that are overtraced (e.g., Fig. 1). We attempt to solve the problem at the geometric level by fitting geometric primitives to input strokes.



Figure 1: Example of an overtraced stroke.

## Shape approximation

We generate fits for lines, arcs and circles by computing model parameters[1] that minimize the least squares fitting error. Polylines usually require a feature point detection step, which is complicated by the overtraced nature of the strokes.

### Line approximation

We generate line approximations by finding a total least squares fit to the data, i.e., assuming noise in both coordinates $(x, y)$ of the data. Simpler regression techniques assume the noise is in only the $y$ component and measure only the vertical distance to the fitted line result in unreliable fits especially with lines near vertical, where a small error in the $x$ coordinate causes a large change in the error contributed by that point.

We compute the equation of the total least squares line fit in the form of $ax + by + c = 0$ by computing the first eigenvector $v$ of the covariance matrix of the $(x, y)$ positions. Then the parameters of the line are $a = v_1$, $b = v_2$ and $c = -a\bar{x} - b\bar{y}$.

### Generating arc and circle approximations

We have previously shown how an arc approximation can be obtained by fitting a circular arc to overtraced strokes (Sezgin & Davis 2004a). This is done by writing the equation for a circle as $(x_i - c_x)^2 + (y_i - c_y)^2 = r^2$ where $(c_x, c_y)$ is the center of the circular arc and $r$ is the radius of the arc. Then we find the least squares solution for:

[1] E.g., slope and position for lines; center, radius and extent for arcs.

# Scale-space Based Feature Point Detection for Digital Ink

**Tevfik Metin Sezgin and Randall Davis**
MIT Computer Science and Artificial Intelligence Laboratory
The Stata Center 235
Cambridge MA, 02139
{mtsezgin,davis}@csail.mit.edu

### Abstract

Feature point detection is generally the first step in model-based approaches to sketch recognition. Feature point detection in free-hand strokes is a hard problem because the input has noise from digitization, from natural hand tremor, and from lack of perfect motor control during drawing. Existing feature point detection methods for free-hand strokes require hand-tuned thresholds for filtering out the false positives. In this paper, we present a threshold-free feature point detection method using ideas from the scale-space theory.

## Introduction

There is increasing interest in building systems that can recognize and reason about sketches. Among different approaches to sketch recognition, model-based recognition techniques model objects in terms of their constituent geometric parts and how they fit together (e.g., a rectangle is formed by four lines, all of which intersect at right angles at four distinct corners). In order to be able to match scene elements to geometric model parts, it is necessary to convert the free-hand strokes in the scene into geometric primitives, which results in a more concise and meaningful description of the scene compared to a raw representation only in terms of sampled pen positions. As described in (Sezgin *et al.* November 2001), feature point (i.e., corner) detection is a major part of generating such geometric descriptions.

### Issues

The major issue in feature point detection is the noise in the data. We consider noise from two sources: imprecise motor control and digitization. We describe characteristics of each kind of noise with examples to make the distinction clear.

#### *Imperfect motor control*

Examples of noise due to imperfect motor control include line segments that were meant to be straight but are not, or corners that look round as opposed to having a precise turning point. This kind of noise gives sketches their "messy" appearance. Easiest way of characterizing this kind of noise is to ask if the noise would still be present if the user drew very carefully perhaps using a ruler. If the answer is negative, then the noise is due to imperfect motor control.



#### *Digitization noise*

Digitization noise is the kind of noise that cannot be removed even if one draws very carefully. Although visually less apparent, it hinders feature point detection because digitization corrupts curvature and speed data, which are primary sources of information in feature point detection. Digitization noise can be present in the $(x, y)$ positions and in their timestamps. Source of the spatial digitization noise is the conversion to screen coordinates. For example, in the Acer C110 Tablet PC, the pen positions are digitized into a 1024x768 grid. Spatial digitization can be so poor that the point stream returned by digitization may occasionally have points with repeating $(x, y)$ positions.

In the same platform, timestamps too have digitization noise. Because the concept of having digitization noise in timestamps is less intuitive, we illustrate the point with an example. Consider the stroke in Fig. 1 captured using an Acer c110 Tablet PC. In this platform, we know that the hardware samples points uniformly at a high resolution, digitizing the timestamps once. Then, the operating system digitizes the timestamps again at 100 Hz. Although the timestamps are good when read at the higher resolution directly using Microsoft's Tablet PC API, they get corrupted during digitization. For the stroke in Fig. 1, Fig. 2 shows the deviation of the digitized timestamps from their predicted ground truth values computed by a least squares linear regression line. The slope of the least squares regression gives us the hardware sampling rate (which is about 133 Hz). The difference in the sampling frequencies causes a skew to accumulate between the real timestamps of the points and those obtained after digitization. The timestamps are occasionally adjusted for the skew by repeating a timestamp, which occurs about every four points with a standard deviation of 0.53. Furthermore, although less frequent, the digitizer consistently returns a point which is 11ms apart from the previous one (as opposed to the more frequent 10ms time difference). This happens roughly once every 92 points ($\mu = 92.25$, $\sigma = 0.95$). If we consider that the time resolution at a sampling rate of 100Hz is 10 ms, the deviations in Fig. 2 which range between $(-8, 6)$ with $\sigma = 3.02ms$ is quite significant. Digitization noise of this nature causes

# Sketch Based Interfaces: Early Processing for Sketch Understanding

Tevfik Metin Sezgin
MIT AI Laboratory
Massachusetts Institute of Technology
Cambridge MA 02139, USA
mtsezgin@ai.mit.edu

Thomas Stahovich
CMU Deptarment of Mechanical Engineering
Pittsburgh, PA 15213
stahov@andrew.cmu.edu

Randall Davis
MIT AI Laboratory
Massachusetts Institute of Technology
Cambridge MA 02139, USA
davis@ai.mit.edu

## ABSTRACT

Freehand sketching is a natural and crucial part of everyday human interaction, yet is almost totally unsupported by current user interfaces. We are working to combine the flexibility and ease of use of paper and pencil with the processing power of a computer, to produce a user interface for design that feels as natural as paper, yet is considerably smarter. One of the most basic steps in accomplishing this is converting the original digitized pen strokes in a sketch into the intended geometric objects. In this paper we describe an implemented system that combines multiple sources of knowledge to provide robust early processing for freehand sketching.

## Categories and Subject Descriptors

D.2.2 [**Design Tools and Techniques**]: User interfaces; H.5.2 [**User Interfaces**]: Input Devices and strategies; J.6 [**Computer Aided Engineering**]: CAD

## General Terms

Design, Human Factors

## Keywords

freehand sketching, natural interaction, multiple sources of knowledge

## 1. INTRODUCTION

Freehand sketching is a familiar, efficient, and natural way of expressing certain kinds of ideas, particularly in the early phases of design. Yet this archetypal behavior is largely unsupported by user interfaces in general and by design software in particular, which has for the most part aimed at providing services in the later phases of design. As a result designers either forgo tool use at the early stage or end up having to sacrifice the utility of freehand sketching for the capabilities provided by the tools. When they move to a computer for detailed design, designers usually leave the sketch behind and the effort put into defining the rough geometry on paper is largely lost.

We are working to provide a system where users can sketch naturally and have the sketches understood. By "understood" we mean that sketches can be used to convey to the system the same sorts of information about structure and behavior as they communicate to a human engineer.

Such a system would allow users to interact with the computer without having to deal with icons, menus and tool selection, and would exploit direct manipulation (e.g., specifying curves by sketching them directly, rather than by specifying end points and control points). We also want users to be able to draw in an unrestricted fashion. It should, for example, be possible to draw a rectangle clockwise or counterclockwise, or with multiple strokes. Even more generally, the system, like people, should respond to how an object looks (e.g., like a rectangle), not how it was drawn. This will, we believe, produce a sketching interface that feels much more natural, unlike Graffiti and other gesture-based systems (e.g., [9], [14]), where pre-specified motions (e.g., an L-shaped stroke or a clockwise rectangular stroke) are required to specify a rectangular shape.

The work reported here is part of our larger effort aimed at providing natural interaction with software, and with design tools in particular. That larger effort seeks to enable user to interact with automated tools in much the same manner as they interact with each other: by informal, messy sketches, verbal descriptions, and gestures. Our overall system uses a blackboard-style architecture [6], combining multiple sources of knowledge to produce a hierarchy of successively more abstract interpretations of a sketch.

Our focus in this paper is on the very first step in the sketch understanding part of that larger undertaking: interpreting the pixels produced by the user's strokes and producing low level geometric descriptions such as lines, ovals, rectangles, arbitrary polylines, curves and their combinations. Conversion from pixels to geometric objects is the first step in interpreting the input sketch. It provides a more compact representation and sets the stage for further, more abstract interpretation (e.g., interpreting a jagged line as a symbol for a spring).

## 2. THE SKETCH UNDERSTANDING TASK


*PUI 2001* Orlando FL, USA

# A Domain-Independent System for Sketch Recognition

Bo Yu, Shijie Cai
State Key Laboratory of Software Novel Technology, Nanjing University, China
mikeyucn@hotmail.com

## Abstract

Freehand sketching is a natural and powerful means of interpersonal communication. But to date, it still cannot be supported effectively by human-computer interface. In this paper, we describe a domain-independent system for sketch recognition. Our system allows users to draw sketches as naturally as how they do on paper, and it recognizes the drawing through imprecise stroke approximation which is implemented in a unified and incremental procedure. This method can handle smooth curves and hybrid shapes as gracefully as it does to polylines. With a feature-area verification mechanism and the intelligent adjustment in the post-process, the system can produce user-intended results. Moreover, the output is organized in a hierarchical structure which includes syntactic and semantic information as well as raw data. Our system mainly utilizes low-level geometric features and does not rely on any domain-specific knowledge. Therefore, it will serve as a general and solid foundation for future high-level applications.

**Keywords:** Sketch Recognition, Graphics Recognition, HCI, Multimodal Interface

## 1. Introduction

Freehand sketching is a very natural and powerful means of interpersonal communication, and people frequently use it to convey information efficiently. It is also a very promising technology of human-computer interface which will release users from a maze of menus, toolbars and many complicated commands. Sketch recognition can be broadly used in the fields of engineering or software design, multimodal user interface, multimedia database retrieval, and so on.

The research on sketch recognition and interpretation has a long history. In [6], Rubine describes a trainable gesture recognizer which uses a linear evaluation function to discriminate gestures. But it can handle only unistroke symbols which must be drawn in a pre-specified manner.

The Design Rationale Group of MIT Artificial Intelligence Laboratory has done much work on sketch recognition [7] and some high-level applications based on it (e.g. [4], [5]). In [7], Sezgin described a scale space approach to analyze curvature and speed data of a stroke for vertex detection. The method avoids priori threshold settings and can achieve good results in some very noisy situations. However it treats vertex detection as a separate stage of process, which does not accommodate interior exchange of information between stages. Consequently, the procedure of vertex detection cannot utilize the result of shape approximation. In addition, his approach cannot handle smooth curves as gracefully as it does to polylines.



In [1], Arvo and Novins described a new sketching interface. Different from traditional ways of recognizing sketches, their method continuously morphs raw input strokes into ideal geometric shapes, which can apprise users the recognition progress and the appearance of final shapes. However, their method can handle only circles and rectangles drawn with one stroke. This is not adequate for a practical system.

Igarashi, Matsuoka, Kawachiya and Tanaka [8] created an interactive beautification technique which could turn imprecise sketches into a rigorous drawing just as being drafted with a CAD system. Their method infers any possible geometric constraints such as perpendicularity, congruence, symmetry etc, generates multiple beautification candidates, and then chooses the most plausible one. However, the restriction that each line must be drawn with a separate stroke may bother users.

Fonseca and Jorge [3] presented a method, based on fuzzy logic, for recognizing multi-stroke sketches. Their system processes one or more strokes, which are collected within a timeout value, as a whole and classifies it into one of several predefined shape kinds. However, their system cannot identify the constituent parts of a shape. And moreover, because it relies entirely on global geometric properties such as area and perimeter, it may have difficulties to distinguish between similar shapes.

From the above reviews, we can find two main deficiencies of the existing methods for sketch recognition:

- Strict restrictions on the drawing manner which do not allow users to sketch naturally.
- Lack of the ability to analyze hybrid or smooth curves and decompose them into primitive shapes.

In this paper, we will describe a domain-independent system for sketch recognition. It is designed to recognize freehand sketches and represent them with a hierarchy of primitive shapes and simple objects. In addition to the above two problems, it aims at generality, and all its operations are based on low-level geometric features. Therefore, it can be integrated conveniently with other high-level and domain-specific applications.

## 2. System Overview

We think that a practical sketch recognition system must have the following attributes:

- It must allow users to draw in a natural way just as how they do on paper.
- It should produce a consistent recognition result, i.e. if a user wants to sketch a line segment, no matter it is drawn with a single stroke, or more than one stroke, or a compound stroke which contains other primitive elements, the system should interpret it as a line segment.
- It should understand the hierarchical relations among the recognized graphic objects. For example, a stroke which is composed of several primitive shapes may serve as a constituent part of some more complex object.
- It should try to guess what a user intends to draw and turn the imprecise strokes into regular objects. Although domain-specific knowledge may be very helpful here, some simple decisions can still be made according to low-level geometric features only.

PERGAMON

Pattern Recognition 34 (2001) 1–14

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

# Template-based online character recognition

Scott D. Connell*, Anil K. Jain

*Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA*

Received 5 November 1998; received in revised form 16 August 1999; accepted 16 August 1999

**Abstract**

Handwriting is a common, natural form of communication for humans, and therefore it is useful to utilize this modality as a means of input to machines. One well-known method of classifying individual characters or words is template matching. We demonstrate a template-based system for online character recognition where the number of representative templates is determined automatically. These templates can be viewed as representing different styles of writing a particular character. The templates are then used as a reference for efficient classification using decision trees. Overall, our classifier achieves an 86.9% accuracy on a set of 17,928 alphanumeric characters (36 classes; 10 digits and 26 lowercase letters) with a throughput of over 8 characters per second on a 296 MHz Sun UltraSparc. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

*Keywords:* Clustering; String matching; Online handwriting; Prototypes; Decision trees

## 1. Introduction

The amount of information that can be stored and processed by desktop computers is increasing at a tremendous rate. Given this rate of increase, the ease at which information can be exchanged with a user becomes a serious bottleneck. In order to be effective, user interfaces should be (1) efficient and (2) natural to the user, thereby requiring little or no learning curve for the user. While there has been much progress on machine presentation of data to humans, such as data visualization tools, the primary mode of data input from a human to a computer is the keyboard. Speech recognition and handwriting recognition utilize other, more natural, forms of human communication, which have recently been integrated in many consumer products (e.g., Apple's Newton Messagepad, the CrossPad by A.T. Cross and IBM, Interactive Voice Response Units (IVRUs) used by many telephone companies, etc). However, for these input modalities to be economical and user-friendly, their recognition accuracy must be sufficiently high such that the user needs to make only a minimal number of corrections to the recognized text or speech.

Handwriting recognition can be broken into two fields which differ in the form in which the data is presented to the system. In *off-line* handwriting recognition, the user writes on paper which is later digitized by a scanner. The data is presented to the system as an image, requiring a segmentation of the writing from the image background before recognition can be done. In contrast, the field of *online* handwriting recognition requires that the user write on a digitizing tablet using a special stylus, so that the user's written strokes are captured as they are being formed by sampling the pen's $(x, y)$ coordinates at evenly spaced time intervals. The use of a pressure-sensitive switch on the pen tip indicates pen-up/pen-down status and disambiguates stroke segmentations.

To perform classification of handwritten characters, a recognition system must attempt to leverage between class variations, while accommodating potentially large within-class variations. If a recognition system is to work well for a large number of different writers (a writer-independent system), this within-class variation can be very large. The variance within a particular character

* Corresponding author. Tel.: + 1-517-355-9319; fax: + 1-517-432-1061.

*E-mail addresses:* connell@cse.msu.edu (S.D. Connell), jain@cse.msu.edu (A.K. Jain).


PII: S0031-3203(99)00197-1

# Automatically Transforming Symbolic Shape Descriptions for Use in Sketch Recognition

**Tracy Hammond and Randall Davis**
MIT Computer Science and Artificial Intelligence Laboratory (CSAIL)
MIT Building 32-(239,237), 32 Vassar St.
Cambridge, MA 02139
{hammond,davis} at csail.mit.edu

**Abstract**

Sketch recognition systems are currently being developed for many domains, but can be time consuming to build if they are to handle the intricacies of each domain. This paper presents the first translator that takes symbolic shape descriptions (written in the LADDER sketch language) and automatically transforms them into shape recognizers, editing recognizers, and shape exhibitors for use in conjunction with a domain independent sketch recognition system. This transformation allows us to build a single domain independent recognition system that can be customized for multiple domains. We have tested our framework by writing several domain descriptions and automatically created a domain specific sketch recognition system for each domain.

## Introduction

As pen-based input devices have become more common, sketch recognition systems are being developed for many domains such as mechanical engineering (Alvarado 2000), UML class diagrams (Hammond & Davis 2002), webpage design (Lin *et al.* 2000), architecture (Gross, Zimring, & Do 1994), GUI design (Caetano *et al.* 2002a; Lecolinet 1998), virtual reality (Do 2001), stick figures (Mahoney & Fromherz 2002), course of action diagrams (Pittman *et al.* 1996), and many others. These systems allow users to sketch a design, which is a more naturally interaction than a traditional mouse and palette tool (Hse *et al.* 1999). But sketch recognition systems can be quite time consuming to build if they are to handle the intricacies of each domain.

We propose that rather than build a separate recognition system for each domain, we instead build a single domain independent recognition system that can be customized for each domain. To build a sketch recognition system for a new domain, the developer would need only write a domain description, describing how shapes are drawn, displayed and edited. This description would then be transformed for use in the domain independent system. The inspiration for such a framework stems from work in speech recognition, which has been using this approach with some success (Zue & Glass 2000).



In our work, we transform a grammar into a domain recognizer of hand-drawn shapes. This is analogous to work done on compiler compilers, in particular visual language compiler compilers (Costagliola *et al.* 1995). A visual language compiler compiler allows a user to specify a grammar for a visual language, then compiles it into a recognizer which can indicate whether a arrangement of icons is syntactically valid. The main difference between this work and ours is that 1) ours handles hand-drawn images and 2) their primitives are the iconic shapes in the domain whereas our primitives are geometric.

In this paper we present the first translator that takes symbolic descriptions of how shapes are drawn, displayed, and edited in a domain and automatically transforms them into shape recognizers, editing recognizers, and shape exhibitors for use in a domain independent sketch recognition system. To succeed in our goal, we have created 1) LADDER(Hammond & Davis 2003), a symbolic language for describing how shapes are drawn, displayed, and edited in a domain, 2) a translator as described above, and 3) a simple domain independent recognition system that uses the newly translated components to recognize, display, and allow editing of the domain shapes. The implementation of this translator and domain independent sketch recognition system serves to show both that such a framework is feasible and that LADDER is an acceptable language for describing domain information.

The domain independent recognition system and transformer described here were designed to test whether we could in fact transform symbolic descriptions of a domain into active recognizers usable by a domain independent recognition system. Other work in our group is pursuing a more ambitious approach to both building a domain independent recognition system and studying the process of transforming descriptions into recognizers. Nevertheless, the work reported here does illustrate the plausibility of transforming descriptions into recognizers.

We have chosen a symbolic sketching language based on how shapes look rather than on features such as drawing speed, size of the bounding box, etc., (as in systems like (Rubine 1991; Long 2001)).We did this to ensure that symbols would be recognized if they looked the same, even if they weren't drawn in the same way (e.g., with a different number of strokes), allowing users to draw the shapes as they

# Sketched Symbol Recognition using Zernike Moments

Heloise Hse, A. Richard Newton
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
{hwawen, newton}@eecs.berkeley.edu

## Abstract

*In this paper, we present an on-line recognition method for hand-sketched symbols. The method is independent of stroke-order, -number, and -direction, as well as invariant to scaling, translation, rotation and reflection of symbols. Zernike moment descriptors are used to represent symbols and three different classification techniques are compared: Support Vector Machines (SVM), Minimum Mean Distance (MMD), and Nearest Neighbor (NN). We have obtained a 97% recognition accuracy rate on a dataset consisting of 7,410 sketched symbols using Zernike moment features and a SVM classifier.*

## 1. Introduction

Sketching is a simple and natural mode of expression. It is especially desirable for conceptual design, both on an individual basis and in a collaborative environment. There has been a significant amount of research to date in various aspects of sketch-based user interfaces: interactive design tools, studies of gestures, software toolkits, ink beautification, and sketch recognition. In this work, we focus on the recognition of graphic symbols used in common applications. Challenges in sketched symbol recognition lie in the variation and distortion of hand-sketched shapes. Different people may use different stroke-order, -number, and -direction to draw the same shape. Unlike printed symbols, hand-sketched symbols are imprecise in nature such that corners are not always sharp, lines are not perfectly straight, and curves are not necessarily smooth. Furthermore, symbols can be drawn in different sizes and orientation (e.g. the orientation of an arrow depends on its pointing direction), in contrast to handwriting which is often assumed to be written on a baseline in an upright position. A robust recognition system has to account for all of these factors.

The work in on-line sketched symbol recognition can be roughly categorized into statistical, structural, and rule-based approaches. The structural approach describes a symbol in terms of simpler geometric primitives and represents it with a semantic network. However, mistakes in the segmentation stage can lead to incorrect descriptions of symbols and inexact graph matching in the classification stage is very computationally expensive. Rule-based approaches are often ad-hoc, hard to extend, and not very robust. In this work, we consider a statistical approach to sketched symbol recognition using Zernike moments as features. Zernike moments have been used in the optical character recognition and image recognition communities with good results [1, 2], but as far as we know, this feature has not been explored for use in on-line symbol recognition. The rotation and reflection invariance properties of Zernike moments are especially desirable for symbol recognition.

In this work, we evaluate the effectiveness of Zernike moment features for hand-drawn symbol recognition along with three classification methods. Section 2 gives a description of our target symbol set and the test corpus. Symbol preprocessing, feature extraction, and classification methods are presented in Section 3, 4, and 5 respectively. Experiments and results are presented in Section 6.

## 2. Data acquisition

Since there is no publicly available benchmark for sketched symbols, we have created a test corpus by gathering data from different people. Our target class of application for this work is one that has a bounded set of target symbols from which to select (e.g. a UML diagram editor, a slide drawing program like Microsoft PowerPoint, or an electrical schematic editing tool). The shape set was chosen based upon the applications of interest, commonly used basic shapes, and the geometric properties of shapes (e.g. shapes with lines, shapes with curves, shapes with mixed lines and curves, and shapes with and without self intersections). Of course, other shapes can be added and learned by the system, if desired.

So far, we have gathered data from 19 users. Each user was asked to sketch 30 examples for each of the 13 symbols shown in Figure 1. The data set contains a total of 7,410 examples overall and 570 examples per symbol. The data was collected using the Wacom Graphire2 Pen and Tablet.

Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)
1051-4651/04 $ 20.00 IEEE

ARTICLE IN PRESS

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Computers & Graphics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

# An image-based, trainable symbol recognizer for hand-drawn sketches

Levent Burak Kara[a,*], Thomas F. Stahovich[b]

[a]Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[b]Mechanical Engineering Department, University of California, Riverside, Riverside, CA 92521, USA

**Abstract**

We describe a trainable, hand-drawn symbol recognizer based on a multi-layer recognition scheme. Symbols are internally represented as binary templates. An ensemble of four different classifiers compares and ranks definition symbols according to their similarity to the unknown symbol. The scores of the individual classifiers are aggregated to produce a combined score for each definition. The definition with the best combined score is assigned to the unknown symbol. All four classifiers use template-matching techniques to compute similarity (and dissimilarity) between symbols. Ordinarily, template-matching is sensitive to rotation, and existing solutions for rotation invariance are too expensive for interactive performance. We have developed a fast technique that uses a polar coordinate representation to achieve rotational invariance. This technique is applied prior to the multi-classifier recognition step to determine the best alignment of the unknown with each definition. One advantage of this technique is that it filters out the bulk of unlikely definitions, thereby reducing the number of definitions the multi-classifier recognition step must consider.


*Keywords:* Pen computing; Sketch understanding; Symbol recognition; Pattern recognition; Hausdorff distance; Tanimoto coefficient; Yule coefficient; Polar transform

## 1. Introduction

It is common for engineers, architects, and other designers to spend a considerable amount of time laying down their initial concepts using pencil and paper. Typically, it is only after the main ideas have sufficiently matured, that all of that work is transformed into electronic media in the form of technical drawings, flow charts and mathematical models. This obvious redundancy and inefficiency has propelled the idea of sketch-based user interfaces as a means of achieving more natural human–computer interaction.

Early attempts to create such systems were often limited by insufficient technology. New insights into human perception, as well as advances in pattern recognition, machine intelligence, computer graphics, and hardware technology, have now made it feasible to create usable systems. In fact, in many of today's mainstream computing devices, such as tablet PC's, electronic whiteboards, and personal digital assistants (PDA's), the pen is emerging as a standard tool for interaction.

Many of these new computing devices now come equipped with robust handwriting recognition utilities. However, an important aspect of sketch-based computer interaction that remains largely unsolved is the robust

*Corresponding author. Tel.: +1 412 268 8880; fax: +1412 268 3348.
*E-mail addresses:* lkara@andrew.cmu.edu (L.B. Kara), stahov@engr.ucr.edu (T.F. Stahovich).

0097-8493/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2005.05.004

# A Practical Approach for Writer-Dependent Symbol Recognition Using a Writer-Independent Symbol Recognizer

Joseph J. LaViola, Jr., *Member*, *IEEE*, and Robert C. Zeleznik

**Abstract**—We present a practical technique for using a writer-independent recognition engine to improve the accuracy and speed while reducing the training requirements of a writer-dependent symbol recognizer. Our writer-dependent recognizer uses a set of binary classifiers based on the AdaBoost learning algorithm, one for each possible pairwise symbol comparison. Each classifier consists of a set of weak learners, one of which is based on a writer-independent handwriting recognizer. During online recognition, we also use the $n$-best list of the writer-independent recognizer to prune the set of possible symbols and, thus, reduce the number of required binary classifications. In this paper, we describe the geometric and statistical features used in our recognizer and our all-pairs classification algorithm. We also present the results of experiments that quantify the effect incorporating a writer-independent recognition engine into a writer-dependent recognizer has on accuracy, speed, and user training time.

**Index Terms**—Handwriting recognition, AdaBoost, writer dependence, writer independence, pairwise classification, real-time systems.

## 1 Introduction

With the increasing popularity of pen-based computers, accurate and robust handwriting recognition is becoming a critical part of many pen-based applications. The choice of recognition engine—writer-independent or writer-dependent—has a significant impact beyond just recognition quality. Writer-independent systems provide the benefit that end users can simply step up to the application and start writing without any awareness of or special interaction with the underlying recognizer. Alternatively, writer-dependent systems maximize recognition accuracy by requiring that end users first provide some form of structured input in order to tailor the recognition engine to the user. The choice of recognition engine also has a significant impact on application design, since writer-independent systems are typically harder to customize to a specialized symbol set (for example, mathematical symbols or users who have alternate techniques for entering the same symbol). For example, the Microsoft Handwriting Recognizer [18] is designed to allow an application designer to choose a specific symbol set from a predefined list; however, the designer cannot make any modifications to a chosen set (for example, restricting the letter "Z" to Z or $\mathbb{Z}$), nor can additional symbols such as $\int$ and $\sum$ be added to the set.

Given the functional advantages of writer-dependent recognition, we were interested in the hypothesis that the primary deficiency of writer-dependent recognition, extensive training time on the part of the user, could largely be addressed by bootstrapping the training process with a writer-independent recognizer. In essence, we want to leverage the extensive a priori training typically done during the development of a writer-independent recognizer. Thus, our objective in this work is to quantify the effect incorporating a writer-independent recognizer would have on writer-dependent recognition accuracy as a function of the number of training samples used per symbol.

Consequently, we developed a representative symbol recognizer that, for simplicity of implementation, uses a set of binary classifiers, based on AdaBoost [34], as part of an all-pairs recognition algorithm. We use the Microsoft handwriting recognizer as weak learners in each of the pairwise classifiers. This naive all-pairs strategy was chosen specifically for its simplicity and accuracy. Runtime efficiency was not a primary focus of our work, although we did exploit an important speedup opportunity arising from the observation that the Microsoft handwriting recognizer has the correct symbol in its $n$-best list over 99 percent of the time, despite its overall accuracy of just over 91 percent (see Section 4). By using the $n$-best list during a preprocessing step to reduce the number of possible symbol candidates, we were able to prune the number of classifiers needed for each input symbol. Interestingly, this pruning step not only improves runtime speed but also improves recognition accuracy.

In the next section, we discuss related work on symbol recognition. Section 3 describes our recognition algorithm, including the features used in our weak learners and how we incorporate the Microsoft handwriting recognizer into our writer-dependent recognition engine. Section 4 presents the results of our recognition experiments and discusses their implications. Finally, Section 5 presents conclusions.

## 2 Related Work

There has been a significant amount of work in developing both writer-dependent and writer-independent symbol

- *J.J. LaViola Jr. is with the School of Electrical Engineering and Computer Science, University of Central Florida, Engineering 3—Harris Center, Orlando, FL 32816-2362. E-mail: jjl@cs.ucf.edu.*
- *R.C. Zeleznik is with the Department of Computer Science, Brown University, Box 1910, Providence, RI 02912. E-mail: bcz@cs.brown.edu.*

*Manuscript received 21 Oct. 2005; revised 7 June 2006; accepted 4 Jan. 2007; published online 8 Feb. 2007.*
*Recommended for acceptance by D. Lopresti.*
*For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0562-1005.*
*Digital Object Identifier no. 10.1109/TPAMI.2007.1109.*

EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2006)
Thomas Stahovich and Mario Costa Sousa (Editors)

# An Efficient Graph-Based Symbol Recognizer

WeeSan Lee,[1] Levent Burak Kara,[2] and Thomas F. Stahovich[3]

[1]Department of Computer Science, University of California, Riverside, CA 92521
[2]Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213
[3]Mechanical Engineering Department, University of California, Riverside, CA 92521

**Abstract**

*We describe a trainable symbol recognizer for pen-based user interfaces. Symbols are represented internally as attributed relational graphs that describe both the geometry and topology of the symbols. Symbol recognition reduces to the task of finding the definition symbol whose attributed relational graph best matches that of the unknown symbol. One challenge addressed in the current work is how to perform this graph matching in an efficient fashion so as to achieve interactive performance. We present four approximate graph matching techniques: Stochastic Matching, which is based on stochastic search; Error-driven Matching, which uses local matching errors to drive the solution to an optimal match; Greedy Matching, which uses greedy search; and Sort Matching, which relies on geometric information to accelerate the matching. Finally, we present promising results of initial user studies, and discuss the tradeoffs between the various matching techniques.*

Categories and Subject Descriptors (according to ACM CCS): I.5.2 [Pattern Recognition]: Classifier Design and Evaluation

## 1. Introduction

Researchers have developed a variety of approaches for recognizing hand-drawn shapes and symbols. However, many of the current approaches have important limitations. For example, some methods are limited to single-stroke shapes drawn in preferred orientations [Rub91]. Others consider only aggregate properties of a shape and can confuse dissimilar shapes that have similar aggregate properties [FPJ02]. Other approaches require shapes to be drawn with a consistent pen stroke order [SD05].

Our work is aimed at overcoming some of these limitations. Our goal is to create an efficient, trainable, multi-stroke symbol recognizer that is insensitive to orientation, scaling, and drawing order. This is achieved via a graphical representation. Specifically, a symbol is represented with an attributed relational graph (ARG) describing its geometry and topology. The nodes in the graph represent the geometric primitives, and the edges represent the geometric relationships between them. Representing a symbol in terms of its topology allows us to achieve invariance to rotation and scaling, including non-uniform scaling. Because of the later capability, our approach is particularly tolerant of large variations in the shape of a hand-drawn symbol.

With our approach, symbol recognition reduces to the task of graph matching. During recognition, the ARG of the unknown symbol is matched against the ARG of each definition symbol to find the best match. The unknown is classified by whichever definition matches best. Graph matching, or sub-graph isomorphism [Ull76, DPZ01], is known to be NP-complete [GJ79]. Here, the problem is made more difficult because of noise. Noise comes from variations in how the symbols are drawn as well as from processing errors. For example, it is not uncommon for a symbol to have extra or missing geometric primitives, and thus extra or missing nodes in its ARG.

There has been considerable research in developing efficient graph matching techniques for a variety of applications [CFSV04]. Here we present and evaluate four new techniques specifically designed for recognizing hand-drawn shapes. These techniques are designed to be efficient enough for interactive performance, and to be tolerant of the noise inherent in hand-drawn symbols.

Our recognizer assumes that the individual symbols in a sketch have been located prior to recognition. In other work, we have developed sketch parsers for locating the symbols in a sketch [KS04, GKSS05].

# Visual Similarity of Pen Gestures

**A. Chris Long, Jr., James A. Landay, Lawrence A. Rowe, and Joseph Michiels**
Department of Electrical Engineering and Computer Science
University of California at Berkeley
Berkeley, CA 94720-1776
{allanl,landay,rowe,cujoe}@cs.berkeley.edu
+1-510-643-7106
http://www.cs.berkeley.edu/~{allanl,landay,rowe}

## ABSTRACT

Pen-based user interfaces are becoming ever more popular. Gestures (i.e., marks made with a pen to invoke a command) are a valuable aspect of pen-based UIs, but they also have drawbacks. The challenge in designing good gestures is to make them easy for people to learn and remember. With the goal of better gesture design, we performed a pair of experiments to determine why users find gestures similar. From these experiments, we have derived a computational model for predicting perceived gesture similarity that correlates 0.56 with observation. We will incorporate the results of these experiments into a gesture design tool, which will aid the pen-based UI designer in creating gesture sets that are easier to learn and more memorable.

### Keywords

Pen-based user interfaces, pen gestures, multi-dimensional scaling, similarity, perception

## INTRODUCTION

Pen and paper is a versatile, powerful, and ubiquitous technology [17]. Pen-based user interfaces are becoming more widespread [9] and have great promise in power and versatility [4, 8, 17, 25]. Gestures, or commands issued with a pen, are one desirable feature of pen-based UIs. Because command and operand can be specified in one stroke, they are fast [5]. They also are commonly used and iconic,[1] which makes them easier to remember than textual commands [19]. Gestures are useful on displays ranging from the very small, where screen space is at a premium, to the very large, where controls can be more than arm's reach away [20].

A survey of PDA users [14] showed that users think gestures are powerful, efficient, and convenient. They want more gestures in applications and the ability to define their own gestures. However, the survey also revealed problems with gestures. Specifically, users often find gestures difficult to remember, and they become frustrated when the computer misrecognizes gestures. Users of other systems have also found gestures to be awkward [7].

1. By "iconic", we mean that the gesture shape suits or suggests its meaning.


CHI '2000 The Hague, Amsterdam


We believe gestures can be difficult to use because they are difficult to design. We are developing a tool to assist pen-based UI designers in creating and evaluating gestures for pen-based UIs [15]. The primary benefit of the tool will be to advise designers about how to improve their gesture set. For example, it will notify the designer of gestures that: 1) are likely to be perceived as similar by users, 2) may be difficult for users to learn and remember, and 3) may be misrecognized by the computer. This advice will enable the designer to improve their gestures early in the design process before investing in expensive user studies.

The current work is an investigation into gesture similarity. The goal of this work is to develop a computable, quantitative model of gesture similarity that can be incorporated into the gesture design tool. Perceived similarity is useful for designers to know because it affects how easily users can learn and remember the gestures. We contend that similar operations with a clear spatial mapping, such as scroll up and scroll down, should be assigned similar gestures. Conversely, gestures for more abstract operations that are similar, such as cut and paste, may be easily confused if they are visually similar.

To determine what features affect perceived gesture similarity, we ran a pair of experiments to measure the similarity of a variety of gestures. The data collected in these experiments enabled us to derive an algorithm for computing how similar novel gestures are. In conjunction with information about the impact of gesture similarity on their ease of learning and recall (from another experiment whose results are being analyzed), this will allow our gesture design tool to provide better advice.

The remainder of the paper is organized as follows. The first section discusses related work. The next section describes the gesture similarity experiments. A discussion of the results of both experiments follows. Finally, we present future work and conclusions.

## RELATED WORK

This section discusses relevant prior work. The first subsection gives some background on pen-based user interfaces, which is the context for this work. The second discusses prior work on perceptual similarity of gesture-like shapes. The last section introduces multi-dimensional scaling, a data analysis technique used in our experiments.

### Pen-based interfaces

The device that popularized pen input was the Apple Newton MessagePad. It was designed primarily for pen input. It minimized the use of overlapping windows and encouraged the user to focus on one document at a time. Its core applications were a notepad, to-do list, calendar/scheduler, and address book. By default it recognized text as it was entered, but the user could elect to enter ink and

*Published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2000*

# Learning from One Example Through Shared Densities on Transforms

Erik G. Miller and Nicholas E. Matsakis and Paul A. Viola
Massachusetts Institute of Technology Artificial Intelligence Laboratory
Cambridge MA 02139
{emiller, matsakis, viola}@ai.mit.edu

## Abstract

*We define a process called* congealing *in which elements of a dataset (images) are brought into correspondence with each other jointly, producing a data-defined model. It is based upon minimizing the summed component-wise (pixel-wise) entropies over a continuous set of transforms on the data. One of the biproducts of this minimization is a set of transforms, one associated with each original training sample. We then demonstrate a procedure for effectively bringing test data into correspondence with the data-defined model produced in the congealing process.*

*Subsequently, we develop a probability density over the set of transforms that arose from the congealing process. We suggest that this density over transforms may be shared by many classes, and demonstrate how using this density as "prior knowledge" can be used to develop a classifier based on only a single training example for each class.*

## 1 Introduction

There has been such great progress in the development of classifiers for problems such as handwritten digit recognition that the problem is for many researchers considered to be solved. Classifiers such as LeCun et al.'s convolutional networks [9, 12] achieve performance very close to that of human test subjects, a commonly assumed benchmark of "optimal" performance. It seems that performance cannot get much better. However, these methods require large training sets. For example, LeCun et al. use 6,000 samples of each character in training. This leaves open the question of whether these methods are appropriate when large amounts of training data are not available.

If we examine the performance of classifiers using a small amount of data, in the limit, one example per class, there still seems to be a large gap between the capabilities of machines and humans. Consider the symbol for the new European currency, the "Euro", shown in Figure 1. After seeing a single example of such a character, humans can recognize the character in a wide variety of contexts, styles, and positions.

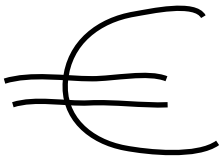


**Figure 1. The new symbol for the standard European currency. After seeing a single example, people typically have no difficulty recognizing the symbol in a wide variety of styles and variations.**

Clearly, this is due at least in part to generalization based on previously learned classes. That is, our knowledge about handwriting in general allows us to bring prior knowledge to the formation of our model for a new character based on a single example. A long-standing question in computer vision, and in AI in general, is what form this prior knowledge takes.

In this paper, we consider the value of knowledge about common deformations of images of objects. We show that such knowledge, in the form of a probability density on a space of deformations, can be used to greatly improve the behavior of a classifier on the handwritten digit recognition problem. Specifically, we discuss the performance of three classifiers. The first classifier is a variation on nearest neighbor using the Hausdorff distance between images [8]. It measures a distance directly to the original training samples. We use this classifier because it can be applied in very sparse data settings, i.e. with only a single training example. This classifier implicitly determines a probability measure on images for each class $c_j$, that is

$$P(I|c_j), \tag{1}$$

the probability of the observed image given the class.

For the second classifier, we give algorithms for finding transforms that place the training and test images in corre-

# Ink Features for Diagram Recognition

Rachel Patel[1], Beryl Plimmer[1], John Grundy[1, 2], Ross Ihaka[3]

[1]Department of Computer Science
[2]Department of Electrical and Computer Engineering
[3]Department of Statistics
University of Auckland
Private Bag 92019, Auckland, New Zealand

**Abstract**

*The ability to automatically recognize a sketch accurately is important to computer-based diagramming. Many recognition techniques have been proposed but few researchers have reported the use of formal methods to select the most appropriate ink features for recognition algorithms. We have used a statistical approach to identify the most important distinguishing features of ink for dividing text and shapes. We implemented these into an existing recognition engine and conducted a comparative evaluation. Our feature set more successfully classified a range of common diagram elements than two existing dividers.*

Categories and subject descriptors: I.4.7 [Image Processing and Computer Vision]: Feature Measurement - *feature representation*

## 1. Introduction

Computer-based sketch tools, particularly diagramming tools, show promise as an alternative to paper for capturing early-phase designs. They retain the advantages of paper - such as an unconstrained drawing space that allows ambiguity and quick construction. They also have the key advantages of computer tools - such as digital storage, transmission and archiving. However, despite suitable hardware being available for some time, diagramming sketch tools are yet to achieve general acceptance. One of the outstanding challenges is the need for far more accurate recognition.

Recognition of sketches is an important aspect of computer-based diagramming: it allows the software to support tasks such as intelligent editing, execution, conversion and animation of the sketches. The syntactic and semantic elements of a diagram are both important. Only with accurate recognition will the full potential of computerization of sketches be realized.

There are three main approaches to the recognition: bottom-up, top-down or a combination of both. Bottom-up attempts to recognize individual ink segments and then progressively join these into larger and more complex groups thus developing an overall semantic understanding of the diagram. Top-down starts with a high-level analysis of the structure and uses this information to aid recognition of the composite parts. Combinations work both the primitives and layout to try to resolve ambiguities.

Regardless of the approach, sketch recognition is comprised of capturing *ink features* and *algorithms* to combine these features. The features measure aspects of an ink stroke's curvature, size, time, intersections and use similar aspects to detect relationships between strokes. To date, a wide range of algorithms have been used to recognise hand drawn shapes and text. However, little research has so far been done into the relative effectiveness of different approaches and use of different ink features in recognition. Furthermore, considering that stroke features are such an important part of recognition, there is little evidence of the use of formal methods to identify the most significant ink features to use. Most reports suggest a reliance on ad-hoc heuristics and empirical trial and error.

Most previous work in sketched diagram recognition concentrates on basic shape and gesture recognition. There are few diagramming tools that allow for both shape and text recognition together. Many of the tools that do support both are either modal interfaces or limited in functionality. However, the flexibility of sketch tools must be increased if they are to equal the performance of designing with pen and paper [Bla90; Goe95; BK03; PA03]. Regardless of whether a top-down or bottom-up approach is taken, text and shapes are semantically different and need to be treated separately during the recognition process. Handwritten characters need to be clustered into words and phrases in preparation for character recognition while shape combinations need to be identified as components and the relationships between the components explored.

In the following section we review sketch recognition techniques currently being used. Following this we describe an experiment that we conducted to find distinguishing ink features of text and shape strokes for a text/shape divider. We discuss the use of these features to implement an improved divider algorithm and compare its performance to two existing algorithms. We then discuss the findings of our experiment in the wider context of diagram recognition and conclude with areas for further research.

# Specifying Gestures by Example

Dean Rubine
Information Technology Center
Carnegie Mellon University
Pittsburgh, PA
Dean.Rubine@cs.cmu.edu

## Abstract

Gesture-based interfaces offer an alternative to traditional keyboard, menu, and direct manipulation interfaces. The ability to specify objects, an operation, and additional parameters with a single intuitive gesture appeals to both novice and experienced users. Unfortunately, gesture-based interfaces have not been extensively researched, partly because they are difficult to create. This paper describes GRANDMA, a toolkit for rapidly adding gestures to direct manipulation interfaces. The trainable single-stroke gesture recognizer used by GRANDMA is also described.

**Keywords** — gesture, interaction techniques, user interface toolkits, statistical pattern recognition

## 1 Introduction

Gesture, as the term is used here, refers to hand markings, entered with a stylus or mouse, that indicate scope and commands [18]. Buxton gives the example of a proofreader's mark for moving text [1]. A single stroke indicates the operation (move text), the operand (the text to be moved), and additional parameters (the new location of the text). The intuitiveness and power of this gesture hints at the great potential of gestural interfaces for improving input from people to machines, historically the bottleneck in human-computer interaction. Additional motivation for gestural input is given by Rhyne [18] and Buxton [1].

A variety of gesture-based applications have been created. Coleman implemented a text editor based on proofreader's marks [3]. Minsky built a gestural interface to the LOGO programming language [13]. A group at IBM constructed a spreadsheet application that combines gesture and handwriting [18]. Buxton's group produced a musical score editor that uses gestures for entering notes [2] and more recently a graphical editor [9]. In these gesture-based applications (and many others) the module that distinguishes between the gestures expected by the system, known as the *gesture recognizer*, is hand coded. This code is usually complicated, making the systems (and the set of gestures accepted) difficult to create, maintain, and modify.

Creating hand-coded recognizers is difficult. This is one reason why gestural input has not received greater attention. This paper describes how gesture recognizers may be created automatically from example gestures, removing the need for hand coding. The recognition technology is incorporated into GRANDMA (Gesture Recognizers Automated in a Novel Direct Manipulation Architecture), a toolkit that enables an implementor to create gestural interfaces for applications with direct manipulation ("click-and-drag") interfaces. In the current work, such applications must themselves be built using GRANDMA. Hopefully, this paper will stimulate the integration of gesture recognition into other user interface construction tools.

Very few tools have been built to aid development of gesture-based applications. Artkit [7] provides architectural support for gestural interfaces, but no support for creating recognizers. Existing trainable character recognizers, such as those built from neural networks [6] or dictionary lookup [15], have significant shortcomings when applied to gestures, due to the different requirements gesture recognition places on a recognizer. In response, Lipscomb [11] has built a trainable recognizer specialized toward gestures, as has this author.

The recognition technology described here produces a small, fast, and accurate recognizers. Each recognizer is rapidly trained from a small number of examples of each gesture. Some gestures may vary in size and/or orientation while others depend on size and/or orientation for discrimination. Dynamic attributes (left-to-right or right-to-left, fast or slow) may be considered in classification. The gestural attributes used for classification are generally meaningful, and may be used as parameters to application routines.

The remainder of the paper describes various facets of GRANDMA. GDP, a gesture-based drawing program built using GRANDMA, is used as an example. First GDP's



 ACM-0-89791-436-8/91/007/0329 $00.75

EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2006)
Thomas Stahovich and Mario Costa Sousa (Editors)

# Constellation Models for Sketch Recognition

D. Sharon and M. van de Panne

University of British Columbia†

**Abstract**

*Sketch-based modeling shares many of the difficulties of the branch of computer vision that deals with single image interpretation. Most obviously, they must both identify the parts observed in a given 2D drawing or image. We draw on constellation models first proposed in the computer vision literature to develop probabilistic models for object sketches, based on multiple example drawings. These models are then applied to estimate the most-likely labels for a new sketch. A multi-pass branch-and-bound algorithm allows well-formed sketches to be quickly labelled, while still supporting the recognition of more ambiguous sketches. Results are presented for five classes of objects.*

## 1. Introduction

A large-class of sketch-based modeling systems, specifically those involving drawings of objects, diagrams, or maps, must solve a recognition problem. What did the user draw and what does each stroke correspond to? In many cases, this is solved with the help of domain knowledge, such as knowing that a sailboat has a mast and a hull. This recognition problem has a strong parallel with the goals of single-image interpretation in computer vision, an area which has seen significant progress over the past few years.

We apply a constellation or 'pictorial structure' model to the recognition of strokes in sketches of particular classes of objects. The model is designed to capture the structure of a particular class of object and is based on local features, such as the shape or size of a stroke, and pairwise features, such as distances to other known parts. We learn a probabilistic model from example sketches with known stroke labelings. The recognition algorithm determines a maximum-likelihood labeling for an unlabelled sketch by searching through the space of possible label assignments using a multi-pass branch and bound algorithm. Our technique supports flexible object structure by allowing for optional parts. By applying a recognition threshold, extraneous strokes can also be readily identified.

Figure 1 shows an example result for the recognition of parts in face sketches. A subset of the training examples are shown, along with a set of successfully labeled free-form sketches and trace-over sketches. A specific contribution of our method is to cope with objects that exhibit considerable variability in the way they are drawn and that allow a variable number of part instantiations.

The output of our algorithm is a set of labels assigned to the strokes. This can then be utilized by a variety of applications. Labelled strokes can be used to construct parameterized 3D models as in [YSvdP05]. Furthermore, they can help to instance models in a 2D or 3D scene, or serve as a partial interpretation of a larger sketched diagram. Sketches can also be used to retrieve images or 3D models from a database and can, in general, provide an intuitive alternative interface to models with complex internal parameterizations such as faces [fac].

Our system makes two particularly strong assumptions. First, it assumes that similar parts are drawn with similar strokes. For example, a flowerpot that is drawn with four separate strokes instead of one stroke is not easily modelled as part of the same object class. Second, object parts which are deemed mandatory in a sketch must have exactly one instance in the sketch. Optional parts may have multiple instances in a given sketch.

The remainder of the paper is organized as follows. Section 2 gives an overview of related work. Section 3 describes the details of the probabilistic constellation model. Section 4 then describes our algorithms for finding maximum-likelihood interpretations of images using the model. Results are presented and discussed in Section 5, including various

† email: dsharon,van@cs.ubc.ca

# Statistical Visual Language Models for Ink Parsing

**Michael Shilman, Hanna Pasula, Stuart Russell, Richard Newton**

Department of Computer Science
University of California at Berkeley
Berkeley, CA 94720
{michaels, pasula, russell, newton}@eecs.berkeley.edu

## Abstract

In this paper we motivate a new technique for automatic recognition of hand-sketched digital ink. By viewing sketched drawings as utterances in a visual language, sketch recognition can be posed as an ambiguous parsing problem. On this premise we have developed an algorithm for ink parsing that uses a statistical model to disambiguate. Under this formulation, writing a new recognizer for a visual language is as simple as writing a declarative grammar for the language, generating a model from the grammar, and training the model on drawing examples. We evaluate the speed and accuracy of this approach for the sample domain of the SILK visual language and report positive initial results.

## Introduction

Since Ivan Sutherland pioneered pen-based computing with his SketchPad system over three decades ago (Sutherland 1963), there has been a widely-held vision of unencumbered tablet computers that present the feel of interactive, smart paper. Over years, we have seen numerous prototype systems that allow users express themselves directly in an appropriate syntax for different application domains, ranging from as flow-charts (Gross 1994) to mathematics (Matsakis 1999) to music notation (Blostein and Haken 1999). Even less structured domains like user interface and web page design can have their own domain-specific visual notation (Lin et al. 2000). Researchers have shown that such sketch-based applications can combine many of the benefits of paper-based sketching with current electronic tools to enable important new creative and collaborative usage scenarios (Landay and Myers 1995).

Unfortunately, while we are on the verge of having suitable mass-market hardware devices to support the pen computing vision, we lack the software technology to adequately implement many of the most useful software applications that will run on these devices. This is not to say that researchers haven't built a variety of toolkits to support sketch-based application prototyping. Existing toolkits support digital ink capture and storage, facilities for interpreting and beautifying sketched ink (Hong and Landay 00), and even sophisticated reusable schemes for user correction of incorrect interpretations (Mankoff, Hudson, and Abowd 2000). However, we believe that the problem of robust sketch recognition has been largely ignored and is crucial to the ultimate success of sketch-based user interfaces in the real world. The goal of this research is to move beyond prototyping and push recognition accuracies to a point where these systems are useful and predictable to end users.

Our work is based on the intuition that considering multiple ambiguous interpretations in a well-characterized syntactic context will result in far more accurate interpretations. We formalize this intuition in a visual language parsing framework, and describe an initial solution based on a new statistical disambiguation technique. In our implementation of this approach, a user specifies a visual language syntax in a declarative grammar, automatically generates a parser from the specification, and trains the parser on a collection of drawing examples. For a representative visual language we compare a standard ambiguous parsing algorithm to our statistically-augmented version and begin to quantify the performance and accuracy benefits that come with a formal statistical model. In addition to basic improvements in recognition accuracy, the approach provides a simple way to trade-off accuracy for speed.

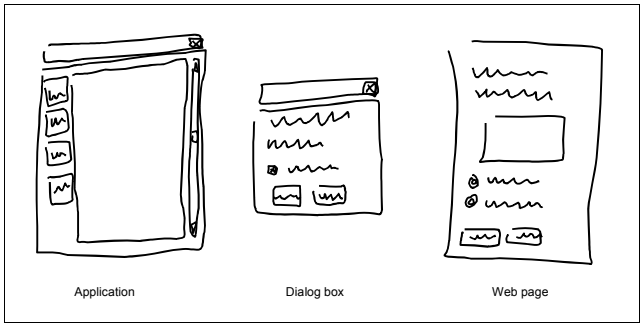



**Figure 1. Hand-sketched SILK diagrams.**

# Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes

*Jacob O. Wobbrock*
The Information School
University of Washington
Mary Gates Hall, Box 352840
Seattle, WA 98195-2840
wobbrock@u.washington.edu

*Andrew D. Wilson*
Microsoft Research
One Microsoft Way
Redmond, WA 98052
awilson@microsoft.com

*Yang Li*
Computer Science & Engineering
University of Washington
The Allen Center, Box 352350
Seattle, WA 98195-2350
yangli@cs.washington.edu

**ABSTRACT**
Although mobile, tablet, large display, and tabletop computers increasingly present opportunities for using pen, finger, and wand gestures in user interfaces, implementing gesture recognition largely has been the privilege of pattern matching experts, not user interface prototypers. Although some user interface libraries and toolkits offer gesture recognizers, such infrastructure is often unavailable in design-oriented environments like Flash, scripting environments like JavaScript, or brand new off-desktop prototyping environments. To enable novice programmers to incorporate gestures into their UI prototypes, we present a "$1 recognizer" that is easy, cheap, and usable almost anywhere in about 100 lines of code. In a study comparing our $1 recognizer, Dynamic Time Warping, and the Rubine classifier on user-supplied gestures, we found that $1 obtains over 97% accuracy with only 1 loaded template and 99% accuracy with 3+ loaded templates. These results were nearly identical to DTW and superior to Rubine. In addition, we found that medium-speed gestures, in which users balanced speed and accuracy, were recognized better than slow or fast gestures for all three recognizers. We also discuss the effect that the number of templates or training examples has on recognition, the score falloff along recognizers' *N*-best lists, and results for individual gestures. We include detailed pseudocode of the $1 recognizer to aid development, inspection, extension, and testing.

**ACM Categories & Subject Descriptors:** H5.2. [Information interfaces and presentation]: User interfaces – *Input devices and strategies*. I5.2. [Pattern recognition]: Design methodology – *Classifier design and evaluation*. I5.5. [Pattern recognition]: Implementation – *Interactive systems*.

**General Terms:** Algorithms, Design, Experimentation, Human Factors.

**Keywords:** Gesture recognition, unistrokes, strokes, marks, symbols, recognition rates, statistical classifiers, Rubine, Dynamic Time Warping, user interfaces, rapid prototyping.


*UIST'07*, October 7-10, 2007, Newport, Rhode Island, USA.

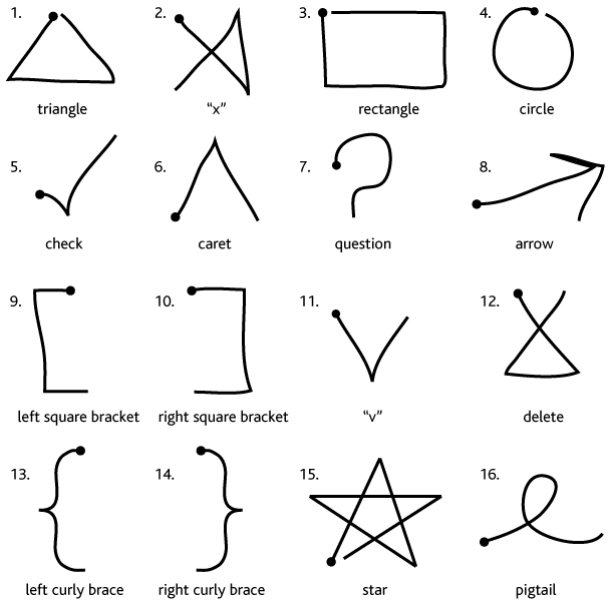




**Figure 1.** Unistroke gestures useful for making selections, executing commands, or entering symbols. This set of 16 was used in our study of $1, DTW [18,28], and Rubine [23].

## INTRODUCTION

Pen, finger, and wand gestures are increasingly relevant to many new user interfaces for mobile, tablet, large display, and tabletop computers [2,5,7,10,16,31]. Even some desktop applications support mouse gestures. The Opera Web Browser, for example, uses mouse gestures to navigate and manage windows.[1] As new computing platforms and new user interface concepts are explored, the opportunity for using gestures made by pens, fingers, wands, or other path-making instruments is likely to grow, and with it, interest from user interface designers and rapid prototypers in using gestures in their projects.

However, along with the naturalness of gestures comes inherent ambiguity, making gesture recognition a topic of interest to experts in artificial intelligence (AI) and pattern matching. To date, designing and implementing gesture recognition largely has been the privilege of experts in these fields, not experts in human-computer interaction

[1] http://www.opera.com/products/desktop/mouse/

# Dynamically Constructed Bayes Nets for Multi-Domain Sketch Understanding

**Christine Alvarado*** **and Randall Davis†**
MIT CSAIL
Cambridge, MA 02139
{calvarad,davis}@csail.mit.edu

## Abstract

This paper presents a novel form of dynamically constructed Bayes net, developed for multi-domain sketch recognition. Our sketch recognition engine integrates shape information and domain knowledge to improve recognition accuracy across a variety of domains using an extendible, hierarchical approach. Our Bayes net framework integrates the influence of stroke data and domain-specific context in recognition, enabling our recognition engine to handle noisy input. We illustrate this behavior with qualitative and quantitative results in two domains: hand-drawn family trees and circuits.

## 1 Introduction

There is a gap between how people naturally express ideas and the ability of computers to use that information. For example, while sketching provides a natural way to record design ideas in many domains, sketches are, unavoidably, static pictures. Computer aided design tools, on the other hand, offer powerful capabilities, but require designers to interact through buttons and menus. The hardware to draw on the computer exists; the missing element is the computer's ability to interpret hand-drawn symbols in a domain. To address this problem, we are constructing a general sketch recognition architecture applicable to a number of domains, and capable of parsing freely-drawn strokes in real time and interpreting them as depicting objects in the domain of interest.

Sketch recognition involves two related subproblems: stroke segmentation and symbol recognition. *Segmentation* determines which strokes should be grouped to form a single symbol. *Symbol recognition* determines what symbol a given set of strokes represents.

The difficulty of doing segmentation and recognition simultaneously has led previous approaches to place constraints on the user's drawing style, or focus on tasks where assumptions can greatly reduce segmentation complexity. For example, the multimodal approach in [Wu *et al.*, 1999] assumes that each symbol will be drawn independently, and that the user will often say the name of the symbol when drawing it. The approach in [Kara and Stahovich, 2004] assumes that the diagram (a feedback control system) consists of shapes linked by arrows, which is not true in other domains.

While these systems have proven useful for their respective tasks, we aimed to create a more general system, independent of drawing assumptions in any one domain. Our system is designed, instead, to be applied to a number of symbolic domains by giving it descriptions of shapes and commonly occurring combinations of shapes. We use these descriptions in a three-stage, constraint-based approach to recognition. Our system first relies on a rough processing of the user's strokes to generate a number of interpretation hypotheses. In the second stage, the system uses a novel form of dynamically constructed Bayes net to determine how well each hypothesis fits the data. In the third stage, it uses this evaluation to guide further hypothesis generation. This paper focuses specifically on the second stage, exploring hypothesis evaluation; the other two stages are described in [Alvarado, 2004].

Using Bayes nets for hypothesis evaluation offers two advantages over previous constraint-based recognition approaches (e.g., [Futrelle and Nikolakis, 1995; Hammond and Davis, 2004]). First, our system can interpret drawings as they develop, identifying shapes before they are complete. Second, the system's belief in a hypothesis can be influenced by both the strokes and the context in which the shape appears, allowing the system to cope with noise in the drawing.

We constructed a sketch recognition engine and used it in two domains: family tree diagrams and circuit diagrams. We show that the Bayes net successfully allows domain-specific context to help the system overcome noise in the stroke data, reducing interpretation errors compared to a baseline system.

## 2 Dynamically Constructed Graphical Models

While time-based graphical models (e.g., Dynamic Bayes Nets) have been widely used, they are not suitable for sketch understanding, for two reasons. First, we must model shapes based on two-dimensional constraints (e.g., touches) rather than on temporal constraints (i.e., follows). Second, our models cannot simply unroll in time as data arrives: we cannot necessarily predict the order in which the user will draw the strokes, and things drawn previously can be changed.

While it is not difficult to use Bayes nets to model spatial relationships, static Bayes nets are not suitable for our task

*Currently at Harvey Mudd College, Claremont, CA, 91711

†This work was funded by the MIT iCampus project supported by Microsoft and MIT Project Oxygen.

*In Proceedings of IJCAI-01*

# Resolving Ambiguities to Create a Natural Computer-Based Sketching Environment

**Christine Alvarado, Randall Davis**
MIT Artificial Intelligence Laboratory

## Abstract

Current computer-based design tools for mechanical engineers are not tailored to the early stages of design. Most designs start as pencil and paper sketches, and are entered into CAD systems only when nearly complete. Our goal is to create a kind of "magic paper" capable of bridging the gap between these two stages. We want to create a computer-based sketching environment that feels as natural as sketching on paper, but unlike paper, understands a mechanical engineer's sketch as it is drawn. One important step toward realizing this goal is resolving ambiguities in the sketch— determining, for example, whether a circle is intended to indicate a wheel or a pin joint—and doing this as the user draws, so that it doesn't interfere with the design process. We present a method and an implemented program that does this for freehand sketches of simple 2-D mechanical devices.

## 1 Sketching Conceptual Designs

Engineers typically make several drawings in the course of a design, ranging from informal sketches to the formal manufacturing drawings created with drafting tools. Drawing is far more than an artifact of the design process; it has been shown to be essential at all stages of the design process [Ullman *et al.*, 1990]. Yet almost all early drawings are still done using pencil and paper. Only after a design is relatively stable do engineers take the time to use computer aided design or drafting tools, typically because existing tools are too difficult to use for the meager payoff they provide at this early stage.

Our aim is to allow designers to sketch just as they would on paper, e.g., without specifying in advance what component they are drawing, yet have the system understand what has been sketched. We want to have the input be as unconstrained as possible, in order to make interaction easy and natural; our route to accomplishing this is to build a sufficiently powerful sketch recognizer.

It is not yet obvious that a freehand sketching interface will be more effective in real use than a carefully designed menu-based system. In order to do the comparison experiments, however, we must first build powerful sketch-based systems. It is the construction of such a system that is the focus of this paper.

The value of sketching as an interface and the utility of intelligent sketch understanding has gained increasing attention in recent years (e.g., [Hearst, 1998]). Some early research was concerned with single stroke classification ([Rubine, 1991]), while more recent work ([Gross, 1995; Landay and Myers, 2001]) puts groups of strokes together to form larger components. A number of efforts (e.g., [Gross and Do, 1996], [Mankoff *et al.*, 2000]) have acknowledged the necessity of representing ambiguities that arise in interpreting strokes, but have not substantially addressed how to resolve those ambiguities.

Given the frequency of ambiguities in a sketch, a tool that constantly interrupts the designer to ask for a choice between multiple alternatives would be cumbersome. Our work is thus focused, in part, on creating a framework in which to both represent and use contextual (top-down) knowledge to resolve the ambiguities. We built a program called ASSIST (A Shrewd Sketch Interpretation and Simulation Tool) that interprets and understands a user's sketch as it is being drawn, providing a natural-feeling environment for mechanical engineering sketches.

The program has a number of interesting capabilities.

- The basic input to the program is a sketch, i.e., a sequence of strokes drawn "while the system watches," not a finished drawing to be interpreted only after it is complete.
- Sketch interpretation happens in real time, as the sketch is being created.
- The program allows the user to draw mechanical components just as on paper, i.e., as informal sketches, without having to pre-select icons or explicitly identify the components.
- The program uses a general architecture for both representing ambiguities and adding contextual knowledge to resolve the ambiguities.
- The program employs a variety of knowledge sources to resolve ambiguity, including knowledge of drawing style and of mechanical engineering design.
- The program understands the sketch, in the sense that it recognizes patterns of strokes as depicting particular

# Distinguishing Text from Graphics in On-line Handwritten Ink*

Christopher M. Bishop, Markus Svensén
Microsoft Research
7 J J Thomson Avenue
Cambridge CB3 0FB, UK
{cmbishop,markussv}@microsoft.com

Geoffrey E. Hinton
Department of Computer Science
University of Toronto
King's College Road, Toronto
Ontario, M5S 3G4, CANADA
hinton@cs.toronto.edu

## Abstract

*We present a system that separates text from graphics strokes in handwritten digital ink. It utilizes not just the characteristics of the strokes, but also the information provided by the gaps between the strokes, as well as the temporal characteristics of the stroke sequence. It is built using machine learning techniques that infer the internal parameters of the system from real digital ink, collected using a Tablet PC.*

## 1 Introduction

Pen controlled computing devices such as PDAs and the Tablet PC are becoming increasingly widespread, and the digital ink captured by such devices offers many potential advantages compared to traditional pen and paper. However, in order to realize many of these advantages it is essential for the device to separate the ink strokes so that text strokes can be sent to a recognition engine, and graphics strokes can be grouped and recognized as higher level graphical entities. Such analysis is essential even if the ink is to be displayed to the user in its original form since search and intelligent editing require a high degree of interpretation of the ink.

In this paper, we consider the fundamental problem of classifying strokes of digital ink as either text or non-text (which we shall refer to as 'graphics'). An example of a page of ink is shown in Figure 1. Features extracted from an individual stroke provide some relevant information regarding the identity of the strokes, and can already give a reasonable separation of the two classes [2]. However,

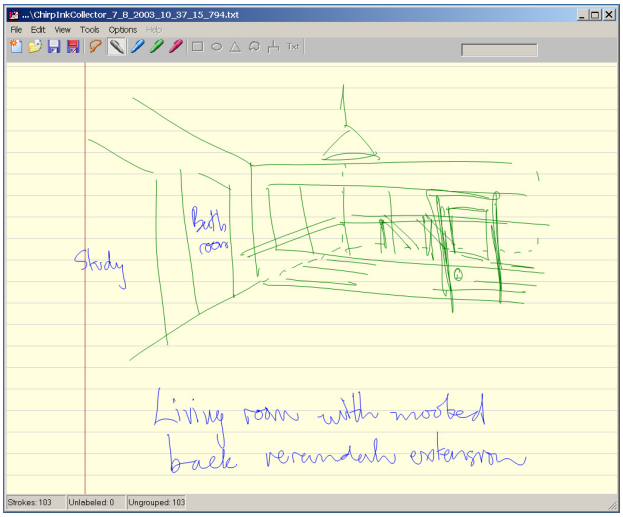



**Figure 1. A screen shot of the purpose built software used to collect and label data, together with a page of ink showing the text strokes (blue) and graphics strokes (green) resulting from manual labelling.**

to achieve improved performance we need to take into account the context of the stroke [5]. Dealing with spatial context leads to a two-dimensional problem which can easily become computationally intensive. Since low level parsing must be performed rapidly, we consider a different approach which exploits the temporal information associated with on-line ink, leading to a one-dimensional optimization problem.

Our approach is based on the use of discriminative machine learning techniques to infer the class of each stroke on the page (text or graphics) given the observed ink. By

*Accepted for oral prentation at the ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR), October 26–29, 2004, Tokyo, Japan. 

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Graphics 29 (2005) 518–532
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

# LADDER, a sketching language for user interface developers

Tracy Hammond[a,*], Randall Davis[b]

[a]MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., 32-239, Cambridge, MA 02141, USA
[b]MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., 32-237, Cambridge, MA 02141, USA

**Abstract**

Sketch recognition systems are currently being developed for many domains, but can be time consuming to build if they are to handle the intricacies of each domain. In order to aid sketch-based user interface developers, we have developed tools to simplify the development of a new sketch recognition interface. We created LADDER, a language to describe how sketched diagrams in a domain are drawn, displayed, and edited. We then automatically transform LADDER structural descriptions into domain specific shape recognizers, editing recognizers, and shape exhibitors for use in conjunction with a domain independent sketch recognition system, creating a sketch recognition system for that domain. We have tested our framework by writing several domain descriptions and automatically generating a domain specific sketch recognition system from each description.


*Keywords:* Object recognition; Object modeling; Shape; Hierarchical scene analysis; Knowledge representation; Representation languages; Representations; Vision and scene understanding; User-centered design

## 1. Introduction

As pen-based input devices have become more common, sketch recognition systems are being developed for many hand-drawn diagrammatic domains such as mechanical engineering [1–3], UML class diagrams [4–7], webpage design [8], GUI design [9,10], virtual reality [11], stick figures [12], course of action diagrams [13], and many others. These sketch interfaces (1) allow for more natural interaction than a traditional mouse and palette tool [14] by allowing users to hand sketch the diagram, (2) can automatically connect to a CAD system preventing the designer from having to enter the same information twice, (3) can offer real-time design advice from CAD systems, (4) allow more powerful editing since the shape is recognized as a whole, (5) provide diagram beautification to remove mess and clutter, and (6) use display as a trigger to inform the sketcher that the shapes have been correctly recognized. However, sketch recognition systems can be quite time consuming to build if they are to handle the intricacies of each domain. Also we would prefer that the builder of a sketch recognition system be an expert in the domain rather than an expert in sketch recognition at a signal level. Rather than build each recognition system separately, our group has been working on a multi-domain recognition system that can be customized for each domain.

Using our framework, in order to build a sketch recognition system for a new domain, a developer need only write a domain description which describes what the domain shapes look like, and how they should be displayed and edited after they are recognized. Thus, the writer of the domain description does not need to know how to program a system to perform sketch recognition. This domain description is then automatically translated

*Corresponding author.
*E-mail addresses:* hammond@csail.mit.edu (T. Hammond), davis@csail.mit.edu (R. Davis).

0097-8493/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2005.05.005

3B2v8.06a/w (Dec 5 2003).51c XML:ver:5.0.1 CAG : 1494 Prod.Type:FTP pp.1–16(col.fig.:NIL) ED:Asha PAGN:NMN SCAN:Global

ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Computers & Graphics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

# Combining geometry and domain knowledge to interpret hand-drawn diagrams

Leslie Gennari[a], Levent Burak Kara[a], Thomas F. Stahovich[b,*], Kenji Shimada[a]

[a]*Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[b]*Department of Mechanical Engineering, University of California, Riverside, CA 92521, USA*



**Abstract**

One main challenge in building interpreters for hand-drawn sketches is the task of parsing a sketch to locate the individual symbols. Many existing pen-based systems avoid this problem by requiring the user to explicitly indicate the partitioning of the sketch with button clicks or pauses in drawing. We have created a parser that automatically locates symbols by looking for areas of high ink density, and for points at which the characteristics of the pen strokes change. To demonstrate our techniques, we have developed AC-SPARC, a sketch-based interface for the SPICE electric circuit analysis program. An evaluation of our interface has indicated that, even for novice users, our system can successfully locate and identify most of the circuit components in hand-drawn circuit diagrams.


*Keywords:* Pen-based computing; Sketch understanding; Sketch parsing; Symbol recognition; Ink density; Electric circuits

## 1. Introduction

Sketching with pencil and paper has always been an important means of communication and problem-solving for designers and engineers. There are a variety of reasons for this. For example, sketches are a convenient tool for examining geometric, temporal, and other similar relationships, which cannot be described easily in words. Similarly, the simplicity and ease of creating a sketch allows one to focus on problem solving rather than the communication medium. Yet, despite the importance of sketches in engineering practice, traditional engineering software can do little with them. Engineers often find themselves recreating their sketches on the computer in order to take advantage of such software. We are working to change this by creating sketch-understanding techniques that enable software to work directly from the kinds of sketches engineers ordinarily draw.

To be natural, a sketch-based interface must place few constraints on the way a user draws, allowing the same freedom provided by pencil and paper. For example, the user should be able to sketch continuously, without being interrupted by the system, and without having to alter his or her drawing style to fit the constraints of the system. We have developed a technique for automatically parsing sketches as one means of achieving this kind of natural interface. Parsing is the task of grouping a user's pen strokes into clusters representing the intended symbols, without explicit indications from the user about where one symbol ends and the next one begins. This is a difficult problem since the number of possible stroke groups increases exponentially with the number of strokes. Additionally, in many types of sketches, more than one symbol can be drawn in a single pen stroke, while conversely there are many other

*Corresponding author. Tel.: +1 951 827 7719; fax: +1 951 827 2899.
*E-mail address:* stahov@engr.ucr.edu (T.F. Stahovich).

0097-8493/$ - see front matter © 2005 Published by Elsevier Ltd.
doi:10.1016/j.cag.2005.05.007

# Hierarchical Parsing and Recognition of Hand-Sketched Diagrams

*Levent Burak Kara*
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
lkara@andrew.cmu.edu

*Thomas F. Stahovich*
Mechanical Engineering Department
University of California, Riverside
Riverside, CA 92521
stahov@engr.ucr.edu

## ABSTRACT

A long standing challenge in pen-based computer interaction is the ability to make sense of informal sketches. A main difficulty lies in reliably extracting and recognizing the intended set of visual objects from a continuous stream of pen strokes. Existing pen-based systems either avoid these issues altogether, thus resulting in the equivalent of a drawing program, or rely on algorithms that place unnatural constraints on the way the user draws. As one step toward alleviating these difficulties, we present an integrated sketch parsing and recognition approach designed to enable natural, fluid, sketch-based computer interaction. The techniques presented in this paper are oriented toward the domain of network diagrams. In the first step of our approach, the stream of pen strokes is examined to identify the arrows in the sketch. The identified arrows then anchor a spatial analysis which groups the uninterpreted strokes into distinct clusters, each representing a single object. Finally, a trainable shape recognizer, which is informed by the spatial analysis, is used to find the best interpretations of the clusters. Based on these concepts, we have built SimuSketch, a sketch-based interface for Matlab's Simulink software package. An evaluation of SimuSketch has indicated that even novice users can effectively utilize our system to solve real engineering problems without having to know much about the underlying recognition techniques.

**Categories and Subject Descriptors:** H.5.2 [**User Interfaces**]: Graphical User Interfaces (GUI), Interaction Styles; I.5.5 [**Pattern Recognition**]: Implementation, Interactive systems

**Additional Keywords and Phrases:** Sketch understanding, pen computing, symbol recognition, visual parsing, sketch understanding, SimuSketch, Simulink

## INTRODUCTION

Pen-based computer interaction is becoming increasingly ubiquitous as evidenced by the growing interest in Tablet PC's, electronic whiteboards and PDA's. Many of these devices now come equipped with robust handwriting recognition, making them an attractive alternative to the keyboard and mouse for text entry. However, when it comes to *graphical input*, such as sketches and diagrams, such devices either leave the pen strokes uninterpreted, or offer only limited support in the form of stroke beautification or simple shape recognition.

We believe that among the many issues that remain to be solved, there are two particular challenges that hinder the development of robust sketch understanding systems. The first is *ink parsing*, the task of grouping a user's pen strokes into clusters representing intended symbols, without requiring the user to indicate when one symbol ends and the next one begins. This is a difficult problem as the strokes can be grouped in many different ways, and moreover, the number of stroke groups to consider increases exponentially with the number of strokes. The combinatorics thus clearly render approaches based on exhaustive search infeasible. To alleviate this difficulty, many of the current systems require the user to explicitly indicate the intended partitioning of the ink. This is often done by pressing a button on the stylus, or more commonly, by pausing between symbols [11, 25]. Alternatively, some systems avoid parsing by requiring each object to be drawn in a single pen stroke [20, 27, 17]. However, such constraints usually result in a less than natural drawing environment.

The second issue is *symbol recognition*, the task of recognizing individual hand drawn figures such as geometric shapes, glyphs and symbols. While there has been significant recent progress in symbol recognition [27, 11, 6, 24], many recognizers are either hand-coded or require large sets of training data to reliably learn new symbol definitions. Such issues make it difficult to extend these systems to new domains with novel shapes and symbols. Additionally most symbol recognizers have been built as stand alone applications without addressing the issue of integration into high-level sketch understanding systems.

In this paper, we address the issue of parsing and recognition of hand-drawn sketches in the domain of network diagrams. The types of sketches we consider can be broadly characterized as a set of symbols (nodes) connected by a set of arrows. The techniques we present are thus well-suited to a variety of diagrams such as signal flow diagrams, organizational charts and algorithmic flowcharts, and to various graphical models


*UIST '04*, October 24–27, 2004, Santa Fe, New Mexico, USA.

# MathPad$^2$: A System for the Creation and Exploration of Mathematical Sketches

Joseph J. LaViola Jr. Robert C. Zeleznik

Brown University *

## Abstract

We present *mathematical sketching*, a novel, pen-based, modeless gestural interaction paradigm for mathematics problem solving. Mathematical sketching derives from the familiar pencil-and-paper process of drawing supporting diagrams to facilitate the formulation of mathematical expressions; however, with a mathematical sketch, users can also leverage their physical intuition by watching their hand-drawn diagrams animate in response to continuous or discrete parameter changes in their written formulas. Diagram animation is driven by implicit associations that are inferred, either automatically or with gestural guidance, from mathematical expressions, diagram labels, and drawing elements. The modeless nature of mathematical sketching enables users to switch freely between modifying diagrams or expressions and viewing animations. Mathematical sketching can also support computational tools for graphing, manipulating and solving equations; initial feedback from a small user group of our mathematical sketching prototype application, MathPad$^2$, suggests that it has the potential to be a powerful tool for mathematical problem solving and visualization.

**CR Categories:** H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction Styles G.4 [Mathematics of Computing]: Mathematical Software—User Interfaces;

**Keywords:** pen-based interfaces, mathematical sketching, gestures

## 1 Introduction

Diagrams and illustrations are often used to help explain mathematical concepts. They are commonplace in math and physics textbooks and provide a form of physical intuition about abstract principles [Hecht 2000; Varberg and Purcell 1992; Young 1992]. Similarly, students often draw pencil-and-paper diagrams for math problems to help in visualizing relationships among variables, constants, and functions. With the drawing as a guide, they can write the appropriate math to solve the problem. However, such static diagrams generally assist only in the initial formulation of mathematical expressions, not in the "debugging" or analysis of those expressions. This can be a severe limitation, even for simple problems with natural mappings to the temporal dimension, or for problems with complex spatial relationships.

By animating sketched diagrams from changes in associated mathematical expressions, users can evaluate different formulations with their physical intuitions about motion. They can sense mismatches between animated and expected behaviors and can often see that a formulation is incorrect and also make better educated guesses as to why. Alternatively, correct formulations can be explored intuitively, perhaps to home in on an aspect of the problem to study with a more conventional numerical or graphing technique. It is beyond the scope of this paper to evaluate the educational merits of mathematical sketching; however, we are convinced that the rapid creation of mathematical sketches can unlock a range of insight, even for such simple formulations as the ballistic 2D motion of a spinning football, where correlations among position, rotation and their derivatives can be challenging to comprehend.

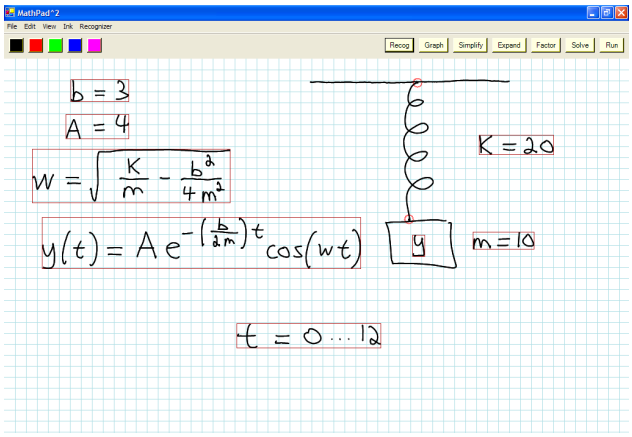



Figure 1: A mathematical sketch used to explore damped harmonic oscillation. It shows a spring and mass drawing and the necessary equations for animating the sketch. The label inside the mass associates the mathematics with the drawing.

This paper presents MathPad$^2$, a prototype application for creating mathematical sketches (see Figure 1). MathPad$^2$ incorporates a novel gestural interaction paradigm that lets users modelessly create handwritten mathematical expressions using familiar math notation and free-form diagrams, including associations between the two, with only a stylus. We posit that because users must write down both the math and the diagrams themselves, MathPad$^2$ will not only be general enough to apply to a full spectrum of problems, but may also support deeper mathematical understanding than alternative approaches including professionally created interactive illustrations. Thus, MathPad$^2$'s central design principle is to be broadly applicable by enforcing the notion that as much behavior as possible be specified by user-written mathematical expressions, not by rules or behaviors implicitly embedded in the system.

To present MathPad$^2$ from a system's perspective, we describe various user interface and recognition options and discuss why we chose our current design. The next section considers related work and is followed by sections on MathPad$^2$'s gestural user interface and on its visual parsing engine. We then present an example scenario of how an introductory physics student might use the system. Finally, we discuss informal feedback from a small user group and

*Email: {jjl,bcz}@cs.brown.edu

# Efficient Geometric Algorithms for Parsing in Two Dimensions

Percy Liang, Mukund Narasimhan, Michael Shilman, and Paul Viola

*Abstract* —

**Grammars are a powerful technique for modeling and extracting the structure of documents. One large challenge, however, is computational complexity. The computational cost of grammatical parsing is related to both the complexity of the input and the ambiguity of the grammar. For programming languages, where the terminals appear in a linear sequence and the grammar is unambiguous, parsing is $O(N)$. For natural languages, which are linear yet have an ambiguous grammar, parsing is $O(N^3)$. For documents, where the terminals are arranged in two dimensions and the grammar is ambiguous, parsing time can be exponential in the number of terminals. In this paper we introduce (and unify) several types of geometrical data structures which can be used to significantly accelerate parsing time. Each data structure embodies a different geometrical constraint on the set of possible valid parses. These data structures are very general, in that they can be used by any type of grammatical model, and a wide variety of document understanding tasks, to limit the set of hypotheses examined and tested. Assuming a clean design for the parsing software, the same parsing framework can be tested with various geometric constraints to determine the most effective combination.**

*Index Terms* —**Document Analysis Systems; Software Methodologies**

## I. Introduction

GRAMMATICAL approaches for document structure analysis have a long history. In comprehensive reviews of the field, a significant percentage of the reported papers have used grammatical approaches [1], [2], [3]. Grammatical document processing research relies on the related work in the field of general grammatical parsing. It is not surprising that these document papers adopt the state of the art in parsing technology at the time of publication. For example, the work of Krishnamoorthy et. al. uses the grammatical and parsing tools available from the programming language community [4] (see also [5], [6]). Similarly the work by Hull uses attributed probabilistic context free grammars [7] (see also [8], [9], [10]). In the last few years there has been a rapid progress in research on grammars in the natural language community . These advances have led to more powerful grammatical models that can be learned directly from data [11]. Such models are strictly more powerful than the probabilistic context free grammars used in previous document analysis research. Simultaneously there has been progress on accelerating the parsing process in the presence of ambiguity [12], [13]. Motivated by these results a new wave of research on grammatical parsing for documents is likely to result.

A grammar based approach selects a global description of the page from several competing descriptions based on a global figure of merit. The local interpretation which maximizes the global score is selected. This provides a principled technique for globally integrating local measurements and handling local ambiguity. The challenges of grammatical approaches include computational complexity, grammar design, and parameter estimation. The focus of this paper is computational complexity: we introduce a set of general purpose geometric constraints and data structures which can be used to accelerate the parsing of two dimensional documents.

Microsoft Research
Redmond, WA, 98033

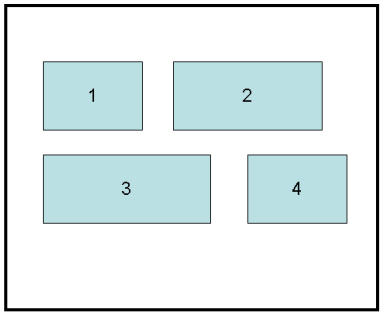



Fig. 1. A very simple page with four terminal objects.

In related work Miller and Viola describe a system for parsing equations which uses geometrical data structures to control the complexity of the parsing process [9]. We improve on their algorithms and present a number of new geometric algorithms as well. Other researchers have used geometric graph data structures to constrain the set of interpretations on the page [10]. While these papers do not discuss grammars directly, the segmentation and recognition problem that they solve can be rephrased as a parsing problem with a simplified grammar. Our work improves on these graph structures as well.

## II. Document Grammars

One simple example examined in detail may yield some intuitions regarding the algorithms presented below. Figure 1 shows a page with 4 terminal objects, which depending on the application could be connected components, pen stokes, text lines, etc. In this case, let us assume that the objects are words on a simple page and the task is to group the words into lines and lines into paragraphs. A simple grammar that expresses this process is:

```
(Page → ParList)
(ParList → Par ParList)
(ParList → Par)
(Par → LineList)
(LineList → Line LineList)
(LineList → Line)
(Line → WordList)
(WordList → Word WordList)
(WordList → Word)
```

The correct parse in this case is:

```
(Page (ParList
       (Par (LineList
```

# Envisioning Sketch Recognition: A Local Feature Based Approach to Recognizing Informal Sketches

by

Michael Oltmans

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2007, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

## Abstract

Hand drawn sketches are an important part of the early design process and are an important aspect of creative design. They are used in many fields including electrical engineering, software engineering and web design. Recognizing shapes in these sketches is a challenging task due to the imprecision with which they are drawn. We tackle this challenge with a visual approach to recognition. The approach is based on a representation of a sketched shape in terms of the visual parts it is made of. By taking this part-based visual approach we are able to recognize shapes that are extremely difficult to recognize with current sketch recognition systems that focus on the individual strokes.

Thesis Supervisor: Randall Davis
Title: Professor of Computer Science and Engineering

# An Evolutionary Algorithm for General Symbol Segmentation

Stephen Pearce
*University Of Guelph*
*Guelph Ontario Canada*
*spearce@uoguelph.ca*

Maher Ahmed
*Assistant Professor, Wilfrid Laurier University*
*Waterloo Ontario Canada*
*mahmed@wlu.ca*

## Abstract

*A new system is presented for general symbol segmentation, which is applicable for segmentation of any connected string of symbols, including characters and line diagrams. Using a powerful graph representation and an evolutionary algorithm framework, segmentation hypotheses are initialized and evolved towards a fully segmented and recognized string. The evolutionary segmentation was tested in many domains including connected digits, connected characters and simple circuit diagrams. The performance of the evolutionary algorithm depends heavily on the symbol recognition system used.*

## 1. Introduction

The problem of separating unknown connected symbols takes many forms. The unknown symbols could have a related meaning in any context imaginable. Most commonly, these symbols are adjacent characters in a handwritten string, but could also be components in a circuit diagram or glyphs in a mural.

The problem of recognizing digits, characters, and other symbols has been addressed by many researchers [1]. This system and others that accomplish the same goals, achieve high accuracy but each one relies on the same assumption, that the input symbol is isolated and free of noise.

In reality, most handwriting runs together, contains broken characters, or is slurred by other noise or lines on the paper. The effects of noise can be minimized through image processing techniques, but characters running together presents a paradox. The paradox is that the individual symbols can not be recognized until they are separated. Until they are recognized, however, there is no reliable information that would help to perform the segmentation. Information that would help includes what the symbols are, or even how many are connected.

## 2. Background and review

Previous segmentation algorithms have attempted to locate symbols based on properties of the domain. These properties usually define the domain and the places where the algorithm would be applicable and may include either the arrangement of the symbols or common features in the symbol set.

Character segmentation involves separating complicated characters arranged horizontally on a baseline. Connected characters are often placed so close that they are touching, overlapping, sharing a line in the image, or leaning over each other. In these cases there are no extra line segments to remove that will separate the characters.

Different character segmentation systems were developed by Strathy and Suen [2] and Parizeau and Plamondon [3] that at some levels are quite similar to each other. In both systems, segmentations are generated by determining an appropriate position to break the connected string.

In [2] the string is surveyed and a predetermined number of potential cuts are generated directly from features in the string. Sub-strings that exist between the cuts are treated as potential symbols. The method in [3] describes the connected string in terms of characteristic points and their surrounding symbol primitives. The primitives are grouped and regrouped until they can be recognized or accepted by some symbol classifier

The system in [2] would encounter difficulties with characters that are doubly connected. In this case, two cuts would have to be made on a single end of the segmentation hypothesis. Similarly, the primitives used in system [3] may not describe all possible symbols or features in a string, since a primitive set designed for the Latin alphabet may not be able

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03)
0-7695-1960-1/03 $17.00 © 2003 IEEE

# Diagram Structure Recognition by Bayesian Conditional Random Fields

Yuan Qi
MIT CSAIL
32 Vassar Street
Cambridge, MA, 02139, USA
alanqi@csail.mit.edu

Martin Szummer
Microsoft Research
7 J J Thomson Avenue
Cambridge, CB3 0FB, UK
szummer@microsoft.com

Thomas P. Minka
Microsoft Research
7 J J Thomson Avenue
Cambridge, CB3 0FB, UK
minka@microsoft.com

## Abstract

*Hand-drawn diagrams present a complex recognition problem. Elements of the diagram are often individually ambiguous, and require context to be interpreted.*

*We present a recognition method based on Bayesian conditional random fields (BCRFs) that jointly analyzes all drawing elements in order to incorporate contextual cues. The classification of each object affects the classification of its neighbors. BCRFs allow flexible and correlated features, and take both spatial and temporal information into account. BCRFs estimate the posterior distribution of parameters during training, and average predictions over the posterior for testing. As a result of model averaging, BCRFs avoid the overfitting problems associated with maximum likelihood training. We also incorporate Automatic Relevance Determination (ARD), a Bayesian feature selection technique, into BCRFs. The result is significantly lower error rates compared to ML- and MAP-trained CRFs.*

## 1. Introduction

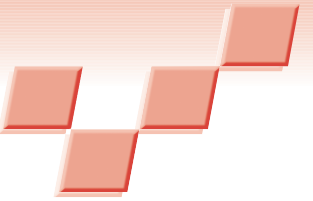
Many computer vision tasks can be effectively formalized as joint classification of multiple pixels or elements. Joint classification enables modeling of dependence between elements, allowing structure and context to be taken into account. For example, image segmentation requires joint labeling of multiple pixels such that related pixels have the same label but pixels across edge or color boundaries have different labels. Object recognition is similarly facilitated by contextual information, such as knowledge about the scene, which consists of multiple interacting elements. In this paper, we tackle another instance of the same problem, namely the recognition of hand-drawn diagram structure (Figure 2). Specifically, we try to identify which stroke fragments are parts of containers versus connectors. This task is challenging because individual pen-strokes look alike and only acquire meaning relative to neighboring strokes. Rather than classify strokes entirely based on local features, we desire to find a globally-consistent labeling of the diagram—an approach which turns out to considerably improve performance.

Joint modeling of structured data can be performed by generative graphical models, such as Bayesian networks or Markov random fields. For example, Tu et al. (2003) propose a generative model for globally parsing images into faces, text, shadows, etc. This sort of approach could be used for diagrams, but in our experience generative models take substantial effort to design and implement for each individual problem, and the result can easily require a prohibitive amount of computation.

Conditional random fields (CRF) are a conditional approach for classifying structured data, proposed by Lafferty et al. (2001). CRFs model only the label distribution conditioned on the observations. Unlike generative models, they do not need to explain the observations or features. This also allows CRFs to use flexible features such as complex functions of multiple observations. The same CRF architecture can be used on many different problems, simply by changing the features. The modeling power of CRFs has shown great benefit in several applications, including region classification (Kumar & Hebert, 2004), diagram labeling (Szummer & Qi, 2004), and CRFs have also been extended to perform simultaneous segmentation and recognition (Cowans & Szummer, 2005).

To summarize, CRFs provide a compelling model for structured data. Consequently, we need effective training, inference, and feature selection algorithms. The standard maximum likelihood (ML) criterion is prone to overfitting the data, especially when CRFs are trained with large numbers of features. Previous approaches to reduce overfitting include maximum a posteriori (MAP) and large margins (Taskar et al., 2004). Both of these approaches involve free parameters which are hard to tune.

Instead we use the method of Bayesian conditional random fields (BCRFs), proposed by (Qi et al., 2005). BCRFs can be viewed as a generalization of Bayes Point Ma-



# Sketch Interpretation Using Multiscale Models of Temporal Patterns

**Tevfik Metin Sezgin and Randall Davis**
*Massachusetts Institute of Technology*

**The growing popularity of tablet PCs and intelligent pen-based interfaces have increased the importance of freehand sketch recognition algorithms as enabling technology. A recognition framework based on multiscale statistical models of temporal patterns significantly increases correct recognition rates, with no added computational penalties.**

Sketching is a natural input modality that has received increased interest in the computer graphics and human–computer interaction communities. The emergence of hardware such as tablet PCs and handheld PDAs provides easy means for capturing pen input. These devices combine a display, pen tracker, and computing device, making it possible to capture and process sketches online, as they are drawn. Online recognition has two main advantages. First, it enables interpreting and displaying pen input as it is entered—for example, an engineer drawing a circuit diagram receives system feedback by observing the ink changing color in real time, indicating which circuit components are recognized. Second, an online sketching system provides access to stroke-ordering information as well as the end product of the drawing process. When we refer to temporal patterns in this article, we mean this stroke-ordering information.

In certain domains, temporal stroke orderings used when sketching objects contain predictable patterns that a system can use for object recognition.[1] We call these stroke-level patterns because they capture the probability of seeing a sequence of strokes with certain properties. For example, when people draw stick figures, one frequently seen stroke-level pattern is a sequence of a circular stroke, a vertical line, and two pairs of positively and negatively sloped lines, corresponding to the figure's head, body, arms, and legs.

Another temporal pattern in online sketches is an object-level pattern, which captures the probability of seeing a certain sequence of objects being drawn. Consider the domain of Unified Modeling Language class diagrams, drawn by software designers using rectangles to indicate classes and various arrows to indicate relations among classes, such as inheritance, generalization, and association. In this domain, when a designer draws a new class (indicated by a rectangle), it's natural to expect that the new object will soon be connected with an arrow to one or more of the objects drawn earlier. We describe this as an object-level pattern, indicating that drawing a class is followed by drawing some variety of arrow. The kind of arrow to expect can depend on the kind of class drawn. A final class, for example, can't be extended, which limits the kind of arrow that we can expect next.

In domains like the Unified Modeling Language, which has a graphical grammar of sorts, it's plausible to imagine writing down the grammar and using this to guide a UML-diagram recognition system. But few domains have patterns that are as well understood as those for UML diagrams. And even if experts could identify such patterns, incorporating them into a recognition system would be a laborious task at best, considering all the ways that various objects can combine. A better way of incorporating object-level temporal patterns in a recognition framework would be to learn them, along with stroke-level patterns, from data.

In this article, we present our sketch-recognition framework, which uses data to automatically learn the object orderings that commonly occur when people sketch and then use the orderings for sketch recognition. The key features that make this framework novel include learning object-level patterns from data, handling objects comprising multiple strokes (*multistroke objects*) and objects that share strokes (*multiobject strokes*), and supporting continuous observable features. We also present an efficient graphical model implementation of our approach and report that a specialized inference algorithm known as the Lauritzen-Jensen stable conditional Gaussian belief propagation

 Published by the IEEE Computer Society 

# Spatial Recognition and Grouping of Text and Graphics

Michael Shilman Paul Viola
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{shilman,viola}@microsoft.com

**ABSTRACT**

*We present a framework for simultaneous grouping and recognition of shapes and symbols in free-form ink diagrams. The approach is completely spatial, that is it does not require any ordering on the strokes. Initially each of the strokes on the page is linked in a proximity graph. A discriminative classifier is used to classify connected subgraphs as either making up one of the known symbols or perhaps as an invalid combination of strokes (e.g. including strokes from two different symbols). This classifier combines the rendered image of the strokes with stroke features such as curvature and endpoints. A small subset of very efficient features is selected, yielding an extremely fast classifier. An A-star search algorithm over connected subsets of the proximity graph is used to simultaneously find the optimal segmentation and recognition of all the strokes on the page. Experiments demonstrate that the system can achieve 97% segmentation/recognition accuracy on a cross-validated shape dataset from 19 different writers.*

*Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation).*

## 1. Introduction

Sketched shape recognition is a classic problem in pen user interfaces. Augmenting a sketched shape with its symbolic meaning can enable numerous features including smart editing, beautification, and interactive simulation of visual languages [Gross94, LM95, AD01, KS04].

In this paper we present an integrated, accurate, and efficient method for recognizing and grouping sketched symbols. Our approach applies to both hand-sketched shapes and printed handwritten text, and even heterogeneous mixtures of the two.

### 1.1 Previous Work

The problem of recognizing sketched drawings can be divided into two parts: grouping strokes into sets, and recognizing what symbol a set of stroke represents.

Previous research has proposed numerous shape recognition strategies including a wide variety of different features and classifiers. Some strategies emphasize invariance to changes in scale and rotation [HN04]. Others require few examples to train [Rub92, KS04, VD04]. Others are able to cope with dashed sketches and overstrikes [FPJ02].

There are also many approaches to grouping ink strokes for recognition. Some systems are designed with the constraint that the user must draw shapes with a single stroke [Rub92]. Some systems use a timeout: when the user does not sketch for a pre-specified time, the system will group the last set of strokes into a shape to be recognized. Some systems use hand-tuned heuristics to group shapes [KS04]. Many handwriting systems require the users to finish writing one shape before beginning on the next one, and then perform an optimization over the sequence of strokes to find the grouping that maximizes some heuristic or statistical score [TSW90].

In work that is most closely related to ours, Mahoney and Fromherz [MF01] have constructed a system that uses finds subgraphs of strokes that satisfy heuristically-specified constraints. They suggest that their approach should work well for sketches that are defined by the structural relationships between strokes, but may not be well-suited for sketches that are defined by the curve shape of the strokes.

We have presented an initial version of this work for the purpose of recognizing and grouping handwritten character strokes in mathematical equations and diagrams [SVC04]. This paper extends the work to flowcharts and mixtures of text and graphics. For this work we have developed a more powerful classification scheme and an improved search strategy for discovering the optimal grouping.

EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2006)
Thomas Stahovich and Mario Costa Sousa (Editors)

# Parsing Ink Annotations on Heterogeneous Documents

Xin Wang[1] and Michael Shilman[2] and Sashi Raghupathy[1]

[1]Ink Parsing Team, TabletPC, Microsoft Corp, One Microsoft Way, RedMond, WA 98052, USA
[2]Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

**Abstract**

*Annotation is an integral part of reading, comprehending, commenting, and authoring notes and documents. In this paper we present a system for recognizing annotations in a flexible digital notebook that may contain a variety of content ranging from text, to images, to handwritten notes. To accomplish the recognition task in real-time makes the complicated annotation parsing problem more difficult.*

*Our approach differs from previous approaches in several ways. First, our approach handles annotations on ink notes, which are significantly more ambiguous than annotations on printed documents and hence more difficult to recognize. Second, our approach is entirely learned from data, so it is easy to adapt to other scenarios. Third, our approach is more thoroughly evaluated than previous systems. On a test set of real user notes, the system has achieved an average recall of* 0.9258 *on all annotation types. Finally, the implementation of the approach will be commercially available as an API in the upcoming release of Windows® Vista® and Office 12®.*

Categories and Subject Descriptors (according to ACM CCS): I.7.m [Computing Methodologies]: Document and Text Processing; I.5.4 [Computing Technology]: Pattern Recognition

## 1. Introduction

A Holy Grail of personal information management is a digital notebook application that simplifies storage, sharing, retrieval, and manipulation of a user's notes, diagrams, web clippings, and so on. This application should be able to flexibly incorporate a wide variety of data types and deal with them reasonably. One approach, as exemplified by Microsoft OneNote®, is to explicitly represent different data types in a single application, and let users capture and fluidly manipulate text, digital ink, and images in data type-specific ways. The application becomes more powerful when ink is intelligently interpreted and given appropriate behaviors according to the type. For instance, hierarchical lists in digital ink notes should be able to expand and collapse just like hierarchical lists in text-based note-taking tools.

Annotations are an important part of a user's interaction with both paper and digital documents, and can be used in numerous ways within the digital notebook. Users annotate documents for comprehension, authoring, editing, note-taking, author feedback, and so on.

When annotations are recognized, they become a form of structured content that semantically decorates any of the other data types in a digital notebook. Recognized annotations can be anchored to document content, so that the annotations can be reflowed as the document layout changes. They can assist in information retrieval, marking places in

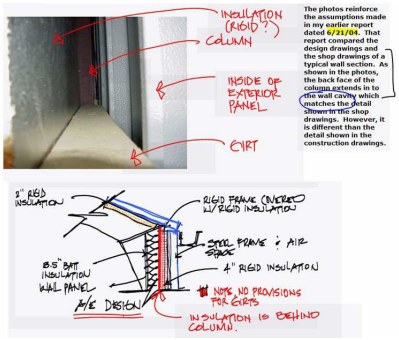



**Figure 1:** *A OneNote 12® file with a mixture of ink, text and images. The fist section of the file is a regular text region. The second section consists of two images with ink annotations. The last section is an ink-drawing with ink annotations surrounding it.*

# Teddy: A Sketching Interface for 3D Freeform Design

Takeo Igarashi[†], Satoshi Matsuoka[‡], Hidehiko Tanaka[†]

[†] University of Tokyo, [‡] Tokyo Institute of Technology

## Abstract

We present a sketching interface for quickly and easily designing freeform models such as stuffed animals and other rotund objects. The user draws several 2D freeform strokes interactively on the screen and the system automatically constructs plausible 3D polygonal surfaces. Our system supports several modeling operations, including the operation to construct a 3D polygonal surface from a 2D silhouette drawn by the user: it inflates the region surrounded by the silhouette making wide areas fat, and narrow areas thin. Teddy, our prototype system, is implemented as a Java™ program, and the mesh construction is done in real-time on a standard PC. Our informal user study showed that a first-time user typically masters the operations within 10 minutes, and can construct interesting 3D models within minutes.

**CR Categories and Subject Descriptions:** I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction Techniques; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Geometric algorithms

**Additional Keywords:** 3D modeling, sketching, pen-based systems, gestures, design, chordal axes, inflation

## 1 INTRODUCTION

Although much progress has been made over the years on 3D modeling systems, they are still difficult and tedious to use when creating freeform surfaces. Their emphasis has been the precise modeling of objects motivated by CAD and similar domains. Recently SKETCH [29] introduced a gesture-based interface for the rapid modeling of CSG-like models consisting of simple primitives.

This paper extends these ideas to create a sketching interface for designing 3D freeform objects. The essential idea is the use of freeform strokes as an expressive design tool. The user draws 2D freeform strokes *interactively* specifying the silhouette of an object, and the system automatically constructs a 3D polygonal surface model based on the strokes. The user does not have to manipulate control points or combine complicated editing operations. Using our technique, even first-time users can create simple, yet expressive 3D models within minutes. In addition, the resulting models have a hand-crafted feel (such as sculptures and stuffed animals) which is difficult to accomplish with most conventional modelers. Examples are shown in Figure 2.


Figure1: Teddy in use on a display-integrated tablet.

Figure 2: Painted models created using Teddy and painted using a commercial texture-map editor.

We describe here the sketching interface and the algorithms for constructing 3D shapes from 2D strokes. We also discuss the implementation of our prototype system, Teddy. The geometric representation we use is a standard polygonal mesh to allow the use of numerous software resources for post-manipulation and rendering. However, the interface itself can be used to create other representations such as volumes [25] or metaballs [17].

Like SKETCH [29], Teddy is designed for the rapid construction of approximate models, not for the careful editing of precise models. To emphasize this design goal and encourage creative exploration, we use the real-time pen-and-ink rendering described in [16], as shown in Figure 1. This also allows real-time interactive rendering using Java on mid-range PCs without

{takeo, tanaka}@mtl.t.u-tokyo.ac.jp, matsu@is.titech.ac.jp
http://mtl.t.u-tokyo.ac.jp/~takeo

# FiberMesh: Designing Freeform Surfaces with 3D Curves

Andrew Nealen
TU Berlin

Takeo Igarashi
The University of Tokyo / PRESTO JST

Olga Sorkine
TU Berlin

Marc Alexa
TU Berlin

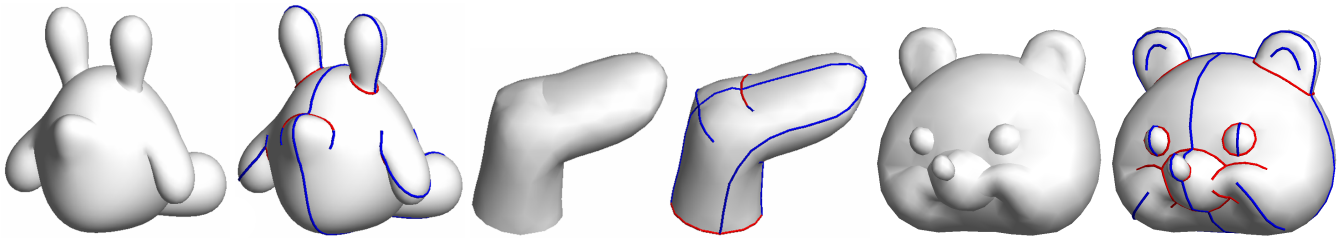


**Figure 1:** *Modeling results using* FIBERMESH. *The user interactively defines the control curves, combining sketching and direct manipulation, and the system continuously presents fair interpolative surfaces defined by these curves (blue = smooth curve, red = sharp curve).*

## Abstract

This paper presents a system for designing freeform surfaces with a collection of 3D curves. The user first creates a rough 3D model by using a sketching interface. Unlike previous sketching systems, the user-drawn strokes stay on the model surface and serve as handles for controlling the geometry. The user can add, remove, and deform these control curves easily, as if working with a 2D line drawing. The curves can have arbitrary topology; they need not be connected to each other. For a given set of curves, the system automatically constructs a smooth surface embedding by applying functional optimization. Our system provides real-time algorithms for both control curve deformation and the subsequent surface optimization. We show that one can create sophisticated models using this system, which have not yet been seen in previous sketching or functional optimization systems.

**Keywords:** Sketch Based Interfaces and Modeling, Differential Representations, Sketching, Deformations, Fair Surface Design

## 1 Introduction

Current tools for free-form design, and the resulting design process, can be roughly categorized into two groups. The group of professional modeling packages makes use of parametric patches or subdivision surfaces [Maya 2007; 3ds Max 2007], where the user has to lay out the coarsest level patches in an initial modeling stage, and then modify control points to generate details. Because it is difficult for inexperienced users to generate the control structure for an intended shape from scratch, a group of research tools [Igarashi et al. 1999; Igarashi and Hughes 2003; Schmidt et al. 2005; Karpenko and Hughes 2006; Kara and Shimada 2007] as well as in-game character editors [Maxis 2007; Gingold 2007] are built around intuitive modeling metaphors such as sketching, trying to hide the mathematical subtleties of surface description from the user. However, some of these tools lack a high-level control structure, making it difficult to iteratively refine the design, or re-use existing designs.

We try to bridge the gap by using curves, a universally accepted modeling metaphor, as an interface for designing a surface. Notice that curves appear in both tools mentioned above: they appear as parameter lines, or seams where locally parameterized patches meet; they are sketched to generate or modify shape, or they are extracted from the current shape and used as handles. Also note that traditional design is mostly based on drawing characteristic curves.

Yet, design is a process. We cannot expect a user to draw the control (or characteristic) curves of a shape into free space. Our first fundamental idea is to let the user **define control curves by drawing them onto the shape** in its current design stage. These curves can be used as handles for deformation right after their definition, as in other tools, or at any other time in the design process. Of course, the effect of control curves can be modified (i.e. smooth vs. sharp edge), they can be removed from the current design, and there are no restrictions on their placement and topological structure. Specifically, they may be connected to or intersect other curves, or not; this is more general than recent developments for parameterized surfaces [Sederberg et al. 2003; Schaefer et al. 2005]).

The second fundamental principle is that **the shape is defined by the control curves** at any stage of the design process. While we found it important to serve the process of construction, and this is also what defines the topology of the surface, the result should be independent of when a control curve was modified. We achieve this by defining the surface to minimize certain functions of its differentials [Moreton and Séquin 1992; Welch and Witkin 1994], while constraining it by the control curves.

It is crucial that both the modification of curves as well as the computation of surface geometry allow for an interactive and smoothly responding system. For this we build on the recent advances in discrete Laplacian [Sorkine et al. 2004; Yu et al. 2004; Botsch and Kobbelt 2004] and other higher order or non-linear functionals for surface processing [Huang et al. 2006; Botsch et al. 2006; Wardetzky et al. 2007].

Uniquely combining interface metaphors (Section 2) with geometry processing techniques (Section 3), our contributions are

- A fair surface definition based on curve constraints, and an accompanying functional optimization algorithm, which runs at interactive rates.
- A detail preserving, real-time 3D curve editing and peeling interface, and a curve deformation algorithm based on discrete co-rotational methods.
- The generation and smooth embedding of initial surface components by sketching a planar control curve on a canvas.
- An interface that enables the design of 3D models with 3D control curves. The user's 2D sketching operations turn into 3D curves, and they serve as handles for subsequent editing.

# ModelCraft: Capturing Freehand Annotations and Edits on Physical 3D Models

*Hyunyoung Song , François Guimbretière, Chang Hu*
Human-Computer Interaction Lab
Department of Computer Science,
University of Maryland,
College Park, MD 20742, U.S.A
{hsong, francois, changhu}@cs.umd.edu

*Hod Lipson*
216 Upson Hall
Cornell University
Ithaca, NY 14852-7501, USA
hod.lipson@cornell.edu

**ABSTRACT**
With the availability of affordable new desktop fabrication techniques such as 3D printing and laser cutting, physical models are used increasingly often during the architectural and industrial design cycle. Models can easily be annotated to capture comments, edits and other forms of feedback. Unfortunately, these annotations remain in the physical world and cannot be easily transferred back to the digital world. Here we present a simple solution to this problem based on a tracking pattern printed on the surface of each model. Our solution is inexpensive, requires no tracking infrastructure or per object calibration, and can be used in the field without a computer nearby. It lets users not only capture annotations, but also edit the model using a simple yet versatile command system. Once captured, annotations and edits are merged into the original CAD models. There they can be easily edited or further refined. We present the design of a SolidWorks plug-in implementing this concept, and report initial feedback from potential users using our prototype. We also present how this prototype could be extended seamlessly to a fully functional system using current 3D printing technology.

**ACM CLASSIFICATION:** H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

**GENERAL TERMS:** Design, Human Factors

**Keywords**: Pen based interactions, Tangible interactions, Rapid prototyping.

**INTRODUCTION**
In the process of designing artifacts, today's designers alternate between tangible, non-digital media such as paper or physical 3D models and intangible, digital media such as CAD models. An architect might start the design of a new building with sketches on paper, then, when her ideas solidify, create a rough model using cardboard, before finally creating the corresponding digital model. Once this model is finalized, it might be fabricated as a 3D object (either through rapid prototyping techniques or a modeling studio) so that her clients may have a better grasp of her vision. While a fully digital design process has long been advocated, it still seems a distant goal because tangible, non-digital media models present unique affordances often difficult to reproduce in digital media. Architectural models for example offer a unique presence that is difficult to reproduce on a screen. As a result, even projects that rely heavily on computer assisted design techniques (such as the recent Hearst building designed by Sir Foster) still employ

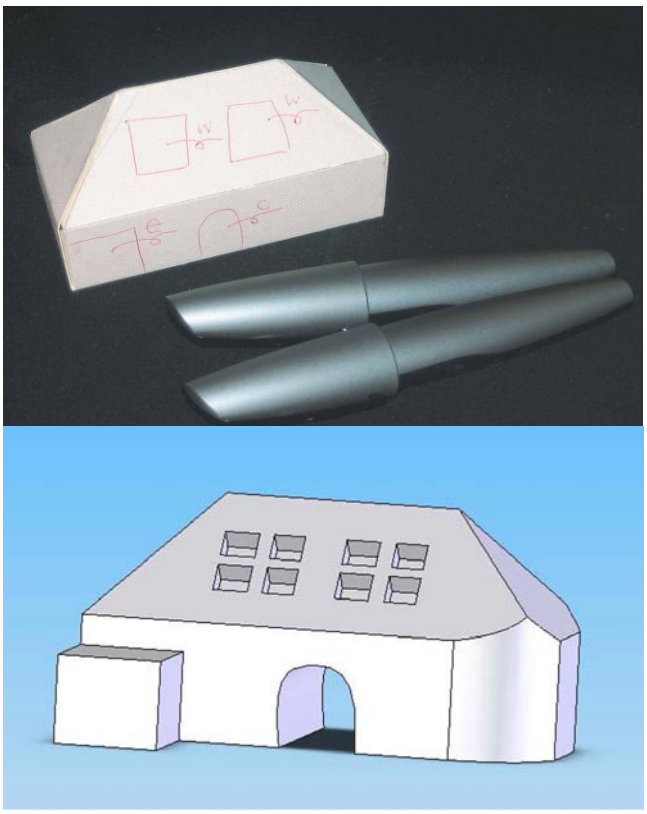


Figure 1: Our system in action. **Top**: paper model of a house with edits (a cut extrusion below the right front corner is not visible). **Bottom**: the same model in our rendering application showing the edits performed.


*UIST'06*, October 15-18, 2006, Montreux, Switzerland.

# SKETCH: An Interface for Sketching 3D Scenes

Robert C. Zeleznik
Kenneth P. Herndon John F. Hughes
{bcz,kph,jfh}@cs.brown.edu
Brown University site of
the NSF Science and Technology Center
for Computer Graphics and Scientific Visualization
PO Box 1910, Providence, RI 02912

## Abstract

Sketching communicates ideas rapidly through approximate visual images with low overhead (pencil and paper), no need for precision or specialized knowledge, and ease of low-level correction and revision. In contrast, most 3D computer modeling systems are good at generating arbitrary views of precise 3D models and support high-level editing and revision. The SKETCH application described in this paper attempts to combine the advantages of each in order to create an environment for *rapidly* conceptualizing and editing *approximate* 3D scenes. To achieve this, SKETCH uses simple non-photorealistic rendering and a purely gestural interface based on simplified line drawings of primitives that allows all operations to be specified *within* the 3D world.

**Keywords:** Interaction Techniques, 3D Modeling, Gestural Interface, Direct Manipulation, Sketching, Nonphotorealistic Rendering

**CR Categories:** I.3.8. [Computer Graphics]: Applications; I.3.6. [Computer Graphics]: Methodology and Techniques — Interaction Techniques

## 1 Introduction

SKETCH targets the exploration and communication of 3D geometric ideas. Traditionally, people have attacked conceptual design with paper and pencil, not with computers, even though computer models offer numerous advantages. The reasons for this include the low overhead of a single-tool interface (pencil), the lack of special knowledge needed to draw, the ease with which many kinds of changes can be made, and the fact that precision is not required to express an idea. Consider Ann sketching a table with an oval top for Joe. Joe gets an immediate sense of the object, without Ann having to indicate the precise locations of the legs, nor the exact shape of the top. By scribbling over what she has sketched, Ann can make the top round or square or freeform without affecting Joe's perception that the legs are attached to the top. (Imagine doing this in a typical CAD or drawing program.) Nevertheless, pencil and paper are still imperfect. After many changes, the paper can become cluttered. Drastic alterations such as showing the model from different viewpoints require new drawings, and collections of drawn objects cannot be transformed as a unit. While computer models do not have these disadvantages, they are typically considerably more difficult to create.

SKETCH is designed to bridge the gap between hand sketches and computer-based modeling programs, combining some of the features of pencil-and-paper sketching and some of the features of CAD systems to provide a lightweight, gesture-based interface to "approximate" 3D polyhedral modeling. Conceptually, our approach is very similar to Landay and Myers' use of sketching to support the early stages of conventional 2D interface design [16]. SKETCH uses a gestural mode of input in which all operations are available directly in the 3D scene through a three-button mouse. The user sketches the salient features of any of a variety of 3D primitives and, following four simple placement rules, SKETCH instantiates the corresponding 3D primitive in the 3D scene. SKETCH allows both geometry and the camera to be gesturally manipulated, and uses an automatic grouping mechanism, similar to that described by Bukowski and Sequin [6], to make it easier to transform aggregates of geometry. Since the set of geometric primitives is more restricted than those in most CAD systems, the user *approximates* complex shapes with aggregates of simpler primitives. Since we know these conceptual models are approximations (often to only partially formed mental images) SKETCH renders them with *non-photorealistic* rendering techniques designed to help viewers see what they want to see.

We also imagine that SKETCH might be used as part of a storyboarding system, for generating a series of scenes and camera views in planning a 3D animation.

The accompanying videotape[1] shows the features of SKETCH and indicates the utility of its simple approach in creating and editing 3D models.

## 2 Related work

A variety of efforts have been made to simplify the process of generating 3D models, including the "idea sketching" described by Akeo et al. [1]. Akeo allows users to scan real sketches into the computer where they are "marked-up" with perspective vanishing lines and 3D cross sections. The scanned data is then projected onto the 3D mark-up to complete the process.

Nearly all CAD applications employ some form of 2D sketching, although sketching is rarely used in 3D views. A notable exception is Artifice's Design Workshop [2], which allows cubes, walls, and constructive solid geometry (CSG) operations to be constructed directly in the 3D view. However, the overall style of interaction is still menu-oriented and the set of primitives is small.

The considerable work done in the area of drawing interpretation, surveyed by Wang and Grinstein [28], focuses solely on interpreting an entire line drawing at once. In contrast, we attempt to provide a complete interface for progressively conceptualizing 3D scenes using aspects of drawing interpretation to recognize primitives from

[1] The videotape can be obtained upon request from the authors.

# Properties of Real-World Digital Logic Diagrams

Christine Alvarado
Harvey Mudd College
Claremont, CA
alvarado@cs.hmc.edu

Michael Lazzareschi
Pomona College
Claremont, CA
michael.lazzareschi@gmail.com

## Abstract

*Despite the growing number of sketch recognition systems for education, little attention has been paid to how students actually draw in practice. We examine freely-drawn digital logic diagrams created by students in an electrical engineering class in order to inform the design of a sketch recognition digital circuit simulation tool. Our analysis reveals considerable drawing style variation between students and that standard drawing style restrictions made by sketch recognition systems to aid recognition generally do not match the way students draw. We identify drawing behaviors that can aid recognition while placing fewer unnatural constraints on students' drawing styles, and we describe specific recognition tasks whose solutions will lead to more robust free-sketch recognition systems.*

## 1. Introduction

Diagrams play a central role in education. Particularly in engineering, diagrams allow students to reason about a physical system, such as a circuit or a mechanical device. Simulation programs enhance the power of diagrams by allowing students to interactively explore the behavior of their design. Yet these programs are fundamentally limited by their mouse and keyboard interfaces. Menus and buttons prevent students from simply drawing their desired symbols, forcing them continually to consult menus to choose pieces of the diagram.

Tablet computers provide a pencil-and-paper-like interface, allowing users to sketch directly on the screen, alleviating many of the constraints of the traditional mouse and keyboard interface. A number of recent educational technologies attempt to combine the freedom of drawing on paper with the power of computer simulation tools in a number of domains including physics [6], chemistry [9], and electrical engineering [4]. The power of these systems comes from their ability to recognize a student's hand-drawn strokes as symbols in a particular domain.

One of the most difficult problems in creating a sketch recognition system is handling the trade-off between ease of recognition and drawing freedom. The more a system constrains the user's drawing style, the easier recognition becomes. Existing sketch recognition systems place a variety of restrictions on the way users draw in order to aid recognition. For example, some systems require users to draw each symbol with a single stroke while others require users to pause between symbols.

Although these restrictions aid recognition, researchers have paid little attention to how well they match the way students naturally draw. These restrictions may force students to change their drawing style so much that the burden of using the system outweighs the benefit. Of course, several factors influence how many and which restrictions users will accept: the utility of the tool, the domain and task, and how much of a burden these restrictions place on their drawing style. Nevertheless, we believe that the fewer restrictions we place on the user *without sacrificing recognition accuracy* the better. To leverage students' natural drawing behavior to improve recognition accuracy, we first must understand this behavior.

This paper examines how students naturally draw digital logic diagrams in order to inform the design of a recognition system for education. We focus on three aspects of students' drawing styles:

- **Stroke order:** Do students complete one symbol before moving to the next?
- **Stroke timing:** Do students pause between drawing different symbols?
- **Stroke number:** How many strokes do users draw per symbol? Do users draw more than one symbol with a single stroke?

We focus specifically on how *students* draw because how they draw may differ from how experienced designers draw. For example, students' symbols are probably messier, and their circuit construction may be unconventional. Educational software must cope with these idiosyncrasies.

We focus on a single domain because previous work has shown that domain-specific knowledge is essential in de-

# CrossY:
# A Crossing-Based Drawing Application

*Georg Apitz & François Guimbretière*

Department of Computer Science
Human-Computer Interaction Lab
University of Maryland,
College Park, MD, 20742
{apitz, francois}@cs.umd.edu

**ABSTRACT**

We introduce CrossY, a simple drawing application developed as a benchmark to demonstrate the feasibility of goal crossing as the basis for a graphical user interface. We show that crossing is not only as expressive as the current point-and-click interface, but also offers more flexibility in interaction design. In particular, crossing encourages the fluid composition of commands which supports the development of more fluid interfaces.

While crossing was previously identified as a potential substitute for the classic point-and-click interaction, this work is the first to report on the practical aspects of implementing an interface based on goal crossing as the fundamental building block.

**CATEGORIES AND SUBJECT DESCRIPTORS**

H.5.2 Graphical User Interfaces, Input Devices and Strategies; D.2.2 User Interfaces; I.3.6 Interaction Techniques

**ADDITIONAL KEYWORDS AND PHRASES**

Crossing based interfaces, command composition, fluid interaction, pen-computing

**INTRODUCTION**

The recent introduction of portable, pen-based computers has demonstrated that, while very powerful, the standard WIMP-interface (Windows, Icons, Menus, and Pointers) is not very well adapted to direct pen interaction. Many WIMP interactions that were originally developed for the mouse are difficult to perform with a pen on a tablet computer. A prime example is the double click: while easy to perform in a mouse environment (since the pointer is stable), it proves to be quite difficult in pen-based interfaces. Other difficulties that arise in pen-based interfaces include occlusions created by the user's hand due to the direct setting, difficulties in using modifier keys (such as pressing shift to extend the current selection), and reduced access to keyboard shortcuts which are crucial for expert performance.

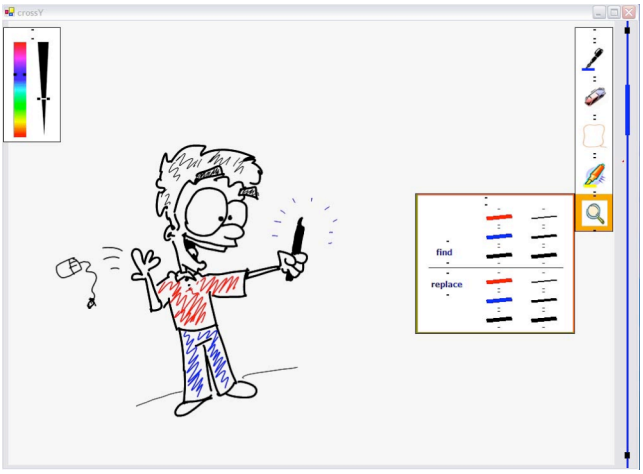



Figure 1 The CrossY interface showing the brush-palette (left) and the main palette with a find/replace dialogue box (right).

Several solutions have been proposed to address these problems. However, by its very nature, the design paradigm of current Graphical User Interfaces (GUI) is not well adapted to the pen's natural affordance of drawing strokes. Traditional point-and-click interfaces insist on segmenting user interactions in a sequence of point-and-click interactions. Using such interfaces with a pen may be frustrating, as users are forced to alternate between a very natural and fluid input mode for sketching or taking notes and a very rigid and segmented interaction while using the GUI elements.

At the same time, recent experimental results by Accot et al. [3] have suggested that steering through goals can be at least as efficient as pointing and clicking and could be a


*UIST '04*, October 24–27, 2004, Santa Fe, New Mexico, USA.

# SketchWizard: Wizard of Oz Prototyping of Pen-Based User Interfaces

*Richard C. Davis,*[1] *T. Scott Saponas,*[2] *Michael Shilman,*[4] *and James A. Landay*[2,3]

[1]CS Division
UC Berkeley
Berkeley, CA 94720
rcdavis@eecs.berkeley.edu

[2]DUB Group
UW, CSE
Box 352350
Seattle, WA 98195
{ssaponas, landay}@cs.washington.edu

[3]Intel Research, Seattle
1100 NE 45th Street,
6th Floor
Seattle, WA 98105

[4]ChatterPop, Inc.
2035 15th St. #3
San Francisco, CA 94114
michael@shilman.net

**ABSTRACT**

SketchWizard allows designers to create Wizard of Oz prototypes of pen-based user interfaces in the early stages of design. In the past, designers have been inhibited from participating in the design of pen-based interfaces because of the inadequacy of paper prototypes and the difficulty of developing functional prototypes. In SketchWizard, designers and end users share a drawing canvas between two computers, allowing the designer to simulate the behavior of recognition or other technologies. Special editing features are provided to help designers respond quickly to end-user input. This paper describes the SketchWizard system and presents two evaluations of our approach. The first is an early feasibility study in which Wizard of Oz was used to prototype a pen-based user interface. The second is a laboratory study in which designers used SketchWizard to simulate existing pen-based interfaces. Both showed that end users gave valuable feedback in spite of delays between end-user actions and wizard updates.

**ACM Classification:** H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

**General terms:** Design, Experimentation, Human Factors

**Keywords:** Wizard of Oz, pen-based user interfaces, informal interfaces, sketching, mark-based user interfaces

## INTRODUCTION

Interest in pen-based user interfaces has continued to grow over the past decade [2,12,13,17–20,24,28,32]. Unfortunately, no design standards have evolved, and adoption of such systems in the marketplace has been slow. The software shipped with pen computers is designed primarily for mouse input and usually ignores the expressive power of pen strokes and sketching. True pen-based user interfaces, such as informal interfaces [3,7,10,17,30], harness this expressive power, but these systems are notoriously difficult to design. We have addressed this situation by building SketchWizard, a tool that allows designers to build and test prototypes of pen-based user interfaces.


*UIST'07,* October 7–10, 2007, Newport, Rhode Island, USA.


Designers work by quickly generating and evaluating numerous design ideas [9,36], but this is hard to do with pen-based interfaces because of the tight coupling between the interface and the technology behind it. These systems provide rich interaction by recognizing, and at times transforming, user sketches and pen gestures—processes that are difficult to simulate with paper prototypes [31,34]. The only current alternative is to build a working system, but this takes time and a deep understanding of technology. With SketchWizard, designers can build Wizard of Oz prototypes, which have evolved as a solution to this type of problem for other hard-to-build interface styles [6,8,11,14,15,20,22,26].

A Wizard of Oz prototype is an incomplete system that a designer can simulate "behind a curtain" (usually by taking the place of a recognizer) while observing the reactions of real end users (see Figure 1). Existing tools make it possible for designers with no programming skill to build Wizard of Oz prototypes of speech [15], location-enhanced [20,22], augmented-reality [8], and desktop [27] applications. With these early-stage Wizard of Oz prototyping tools, designers have the freedom to explore possibilities before technology details are set in stone. This flexibility helps a design team make reasonable technology decisions as the design iterates. SketchWizard gives this ability to pen-based UI designers.

Whether it is possible to build effective early-stage Wizard of Oz prototypes of pen-based UIs is an open question. In other application domains where interactions can be modeled with a small number of input/output primitives, designers can mock up fairly complete interfaces. This way, simulations can be run with minimal input from the designer (the "wizard"). Pen-based interfaces, however, have a much broader input space, processing gestures and sketches in ways that cannot be easily defined. Consequently, the wizard must quickly execute complex transformations of pen input to create an acceptable simulation. SketchWizard assists designers with this process by offering special tools for capturing and modifying end-user input.

Our work makes the following research contributions:

# Hover Widgets: Using the Tracking State to Extend the Capabilities of Pen-Operated Devices

**Tovi Grossman[1,2] Ken Hinckley[1] Patrick Baudisch[1] Maneesh Agrawala[1,3] Ravin Balakrishnan[2]**

[1]Microsoft Research
Redmond, WA
research.microsoft.com
{kenh, baudisch}@microsoft.com

[2]University of Toronto
Toronto, ON
www.dgp.toronto.edu
{tovi, ravin}@dgp.toronto.edu

[3]UC Berkeley
Berkeley, CA
www.cs.berkeley.edu
maneesh@cs.berkeley.edu

## ABSTRACT

We present Hover Widgets, a new technique for increasing the capabilities of pen-based interfaces. Hover Widgets are implemented by using the pen movements above the display surface, in the tracking state. Short gestures while hovering, followed by a *pen down*, access the Hover Widgets, which can be used to activate localized interface widgets. By using the tracking state movements, Hover Widgets create a new command layer which is clearly distinct from the input layer of a pen interface. In a formal experiment Hover Widgets were found to be faster than a more traditional command activation technique, and also reduced errors due to divided attention.

### Author Keywords

Hover Widgets, pen input, gestures, tablets.

### ACM Classification Keywords

H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces, Graphical user interfaces.

## INTRODUCTION

Pen-based interfaces are effective tools for a variety of tasks, such as freeform note taking, and informal sketch design. However, these devices typically lack the keyboard keys, buttons, and scroll wheels that can provide shortcuts for common tasks on the desktop. As a result, the user must zigzag the pen back and forth between the work area and the system menus. This slows users down and diverts their visual attention from their actual task at hand.

Localized user interface elements attempt to solve this problem by bringing the interface to the locus of the user's attention, as indicated by the current pen location [5, 10, 12]. A significant challenge for localized interfaces is that the user must invoke them somehow, such that a pen stroke on the screen activates the interface rather than leaving behind ink. Even with the use of a well-crafted gesture recognition engine, unrecognized gestures can be misinterpreted as ink, and strokes intended as ink can be falsely recognized as gestures, causing unexpected results.

One approach to address this problem is to require the user to press a physical button to explicitly distinguish between command modes and an ink input mode [12, 14]. A button can provide an efficient and effective solution [11], but in some situations it is just not practical. Many mobile devices or electronic whiteboards lack a suitable button, and even if a button is available, it may be awkward to use [18].

We seek new strategies and techniques for supporting localized user interface interactions in pen interfaces. Many pen devices (such as Wacom Tablets and Tablet PC's) support a *tracking state*. The tracking state senses the pen location while the pen is proximal to the interaction surface. However, the literature offers few examples of uses for the tracking state other than cursor feedback [5, 7].

We propose *Hover Widgets*, a novel interaction technique that extends the capabilities of pen-operated devices by using the tracking state to access localized user interface elements. A Hover Widget is invisible to the user during typical pen use, but appears when the user starts moving the pen along a path in the tracking state, and then activates when the user reaches the end of the path and clicks the widget with the pen. For example, the user might form a backwards 'L' shape to activate a marking menu (Figure 1).

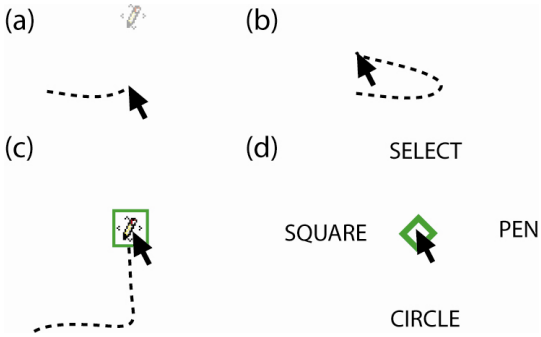



**Figure 1: (a) When the user starts a Hover Widget gesture (here a backwards 'L'), the widget fades in. (b) The user exits the gesture, so the widget fades out (c) Upon completing the gesture, the cursor is over the associated Hover Widget. (d) The user clicks the widget to activate it. The dashed line is for illustration only, showing the pen's path in the tracking state.**


*CHI 2006,* April 22–28, 2006, Montréal, Québec, Canada.

# Design and Analysis of Delimiters for Selection-Action Pen Gesture Phrases in *Scriboli*

*Ken Hinckley, Patrick Baudisch, Gonzalo Ramos, Francois Guimbretiere*
Microsoft Research, One Microsoft Way, Redmond, WA 98052
{kenh, baudisch}@microsoft.com, bonzo@dgp.toronto.edu, francois@cs.umd.edu

**ABSTRACT**
We present a quantitative analysis of delimiters for pen gestures. A delimiter is "something different" in the input stream that a computer can use to determine the structure of input phrases. We study four techniques for delimiting a *selection-action gesture phrase* consisting of lasso selection plus marking-menu-based command activation. *Pigtail* is a new technique that uses a small loop to delimit lasso selection from marking (Fig. 1). *Handle* adds a box to the end of the lasso, from which the user makes a second stroke for marking. *Timeout* uses dwelling with the pen to delimit the lasso from the mark. *Button* uses a button press to signal when to delimit the gesture. We describe the role of delimiters in our *Scriboli* pen interaction testbed, and show how Pigtail supports scope selection, command activation, and direct manipulation all in a single fluid pen gesture.

**Author Keywords**
Pen input, marking, delimiters, tablets, gestures

**ACM Classification Keywords**
H.5.2 Information Interfaces and Presentation: Input

**INTRODUCTION**
In graphical user interfaces, selecting a group of objects and immediately selecting the command to apply to them is a ubiquitous pattern of interaction. For example users might click and drag to sweep out a selection region, and then click on a tool palette or menu to choose a command. This typically requires a *round trip* [7] between the work area and the tool palette or menu bar. While experts sometimes can avoid round trips by learning keyboard shortcuts, this approach only works for desktop configurations. Tablet computers typically have no keyboard, making round trips with the pen unavoidable and tedious.

While the *selection-action* pattern occurs with high frequency in most pen interfaces, the literature lacks a careful study of how the *selection* and *action* subtasks can be combined efficiently. We contribute an experimental analysis and new hybrid approaches that allow designers of pen interfaces to support transitions between and effectively link together selection-action phrases [4]. Our goal is to research new building-blocks for pen interfaces that are (1) *rapid*, with no dwelling or repetitive prompting, but instead using fast, repeatable actions that, with expert use, make minimal demands on visual attention; (2) *unambiguous*, with no recognition unless the user explicitly calls for it; and (3) *expressive*, supporting a variety of commands and using general mechanisms that are not tailored to a specific application domain.


*CHI 2005*, April 2–7, 2005, Portland, Oregon, USA.

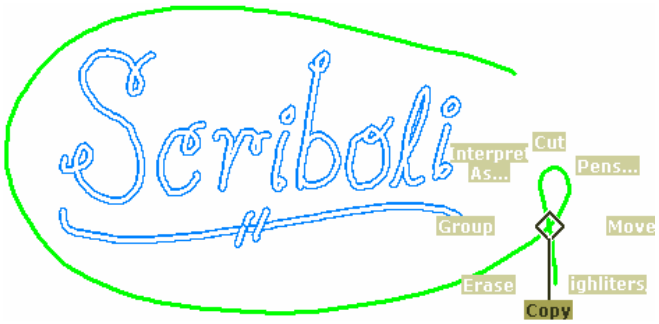




**Fig. 1. Pigtail splits the user's gesture into a lasso, which here selects the ink, and a mark, which here chooses the *Copy* command from 8 possibilities.**

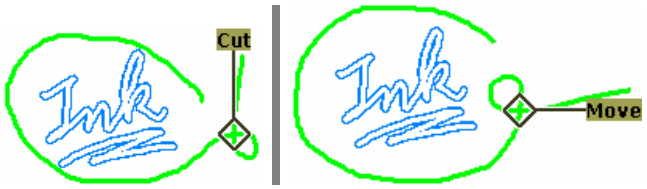



**Fig. 2. Drawing the tail in different directions chooses other commands. *Left*: Cut is north; *Right*: Move is east.**

One of these key building blocks is the delimiter. Delimiters allow interactive systems to determine the lexical structure of an input phrase [3]. They play a dual role of *connecting* tokens while also *separating* tokens from one another. We focus on delimiters in selection-action phrases, that is, how to merge object selection and command activation in a single fluid interaction. In this paper, we analyze four delimiters for selection-action tasks:

***Pigtail***: drawing a small loop at the end of the lasso, with the "tail" interpreted as the mark (see Fig. 1, Fig. 2).

***Timeout***: pausing with the pen at the end of the lasso;

***Button***: pressing a button to explicitly indicate when the computer should stop the lasso and start the mark;

***Handle***: lifting the pen after drawing the lasso, then making a mark starting from a handle the system adds to the end of the lasso (as proposed in [12]; see Fig. 3).

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Graphics 29 (2005) 490–500
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

# Sloppy selection: Providing an accurate interpretation of imprecise selection gestures

Edward Lank[a,*], Eric Saund[b]

[a]Computer Science Department, San Francisco State University, 1600 Holloway Avenue, San Francisco, CA 94132, USA
[b]Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA

**Abstract**

This paper describes on-going work in the analysis of motion dynamics in pen-based interaction. The overall goal is the creation of a model of user motion in pen gestures where constraint and curvature vary over the length of the path. In particular, speed/curvature models of motion are used to analyze pen trajectories and infer target constraints obeyed by a user performing selection gestures. We aim to use this information to calculate an effective local spatial selection tolerance associated with each gesture. This can be used to perform selection according to user intent instead of their literal stroke. Here, we describe our early analysis of constrained user selection gestures, and outline a prototype application that infers a tolerance for one type of selection gesture. The application selectively splits pen strokes based on an analysis of user motion.


*Keywords:* Intelligent scissors; Inferred mode; Steering law; Minimum jerk; Stylus; Motion; Pen interface

## 1. Introduction

This paper describes on-going work in the analysis of motion dynamics of pen-based interaction. The particular problem at hand is the analysis of selection gestures in pen computing. Pen-based selection strategies include two common selection options, tap-to-select and encircling. We focus on the latter, that is, selection by drawing a freehand closed shape around a target object.

Different applications treat selection by encircling according to domain-specific or application-specific criteria. Most paint programs, for example, view selection gestures as definitive and cut image material precisely on the gesture's path. Advanced selection techniques can adjust a selection stroke to fit the borders of salient visual objects.

In pen-based note taking applications, selection gestures are typically interpreted in light of underlying pen strokes. Digital ink strokes themselves, and sometimes groups of ink strokes forming words, are viewed as immutable objects, and the selection gesture selects among strokes or words. A problem faced by these programs is, which objects should become selected when the selection gesture in fact intersects immutable objects?

In our work, we seek to interpret selection strokes based on inference of user intention. We hypothesize that significant and useful aspects of intent can be estimated from measurable characteristics of the gesture.

This work is currently in its early stages. The purpose of this paper is to present initial work in selection gesture analysis under varying curvature and target constraints. In addition, we outline a prototype proof-of-concept application which uses this analysis to make

*Corresponding author.
*E-mail address:* lank@cs.sfsu.edu (E. Lank).

0097-8493/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2005.05.003

EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2006)
Thomas Stahovich and Mario Costa Sousa (Editors)

# An Initial Evaluation of a Pen-Based Tool for Creating Dynamic Mathematical Illustrations

Joseph J. LaViola Jr.

Brown University, Department of Computer Science, USA
Email: jjl@cs.brown.edu

**Abstract**

*MathPad$^2$ is a pen-based application prototype for creating mathematical sketches. Using a modeless gestural interface, it lets users make dynamic illustrations by associating handwritten mathematics with free-form drawings and provides a set of tools for graphing and evaluating mathematical expressions and solving equations. In this paper, we present the results of an initial evaluation of the MathPad$^2$ prototype, examining the user interface's intuitiveness and the application's perceived usefulness. Our evaluations are based on both performance and questionnaire results including first attempt gesture performance, interface recall tests, and surveys of user interface satisfaction and perceived usefulness. The results of our evaluation suggest that, although some test subjects had difficulty with our mathematical expression recognizer, they found the interface, in general, intuitive and easy to remember. More importantly, these results suggest the prototype has the potential to assist beginning physics and mathematics students in problem solving and understanding scientific concepts.*

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces — Interaction Styles, Evaluation/Methodology

## 1. Introduction

MathPad$^2$ (see Figure 1) is a pen-based, Tablet PC application prototype for creating dynamic illustrations used for exploring mathematics and physics concepts [LZ04]. The fundamental technology behind MathPad$^2$ is mathematical sketching, a pen-based gestural interaction paradigm for mathematics problem solving that derives from the familiar pencil-and-paper process of drawing supporting diagrams to facilitate the formulation of mathematical expressions; however, with mathematical sketching, users can also leverage their physical intuition by watching their hand-drawn diagrams animate in response to continuous or discrete parameter changes in their written formulas [LaV05]. Diagram animation is driven by associations that are inferred, either automatically or with gestural guidance, from handwritten mathematical expressions, diagram labels, and drawing elements.

The essential goal in developing the MathPad$^2$ user interface was that it be as similar and fluid as pencil and paper, since mathematics and physics problems are often solved using this medium. Thus, we did not want to use any additional hardware (e.g., a modifier key or stylus button) or software (e.g., buttons) modes. Instead, we wanted all interaction to be derived from using digital ink. We developed a gestural user interface for invoking different operations in MathPad$^2$ because we wanted users able to work as fluidly as possible with the mathematics and drawings they create. We wanted to explore whether our choice of gestures, which by themselves are not part of pencil-and-paper interaction, are thought of as intuitive or at least complimentary to pencil and paper.

Given the foundations for MathPad$^2$, we performed an initial usability evaluation to gauge users' performances and reactions to the prototype to validate its design and potential benefit and determine if further, more in-depth studies are needed. More specifically, we are interested in how easy it is for users to use MathPad$^2$ with only a visual demonstration of how to invoke gestural operations, and in how many mistakes they make in performing various MathPad$^2$ tasks. We are also interested in how well subjects remember various gestural commands, since this is a good indicator of intuitiveness. Using interface satisfaction [CDN88] and perceived usefulness [Dav89] questionnaires, we are addition-

# Perceptual Organization as a Foundation for Intelligent Sketch Editing

**Eric Saund, James Mahoney, David Fleet, Dan Larner, Edward Lank**
Palo Alto Research Center
3333 Coyote Hill Rd., Palo Alto, CA 94304
{saund, mahoney, fleet, larner, elank}@parc.com

**Abstract**

This paper discusses the design of intelligent sketch editing tools exploiting intermediate levels of visual interpretation known as *Perceptual Organization*. Sketches are much more than an accumulation of strokes: they convey information largely through spatial configurations and patterns *among* markings. We propose that the detection of this visual structure, based on the principles of Perceptual Organization, will make sketch editing tools more useful for a wide variety of hand-drawn and formatted diagrammatic material. We review our efforts which are embodied in a perceptually supported sketch editing tool, called ScanScribe.

## Introduction

One application of sketch understanding is to facilitate entering and editing of hand-drawn and formatted figures. Through recognition of image structure, intelligent editors can make it easier for users to select collections of markings that correspond to salient or semantically meaningful entities. Additionally, recognized image structure can be used to modulate the functioning of edit operations, such as by snapping or otherwise enforcing geometric constraints.

Most existing sketch-based systems rely heavily on prior constraints imposed from a targeted application domain. Domain-dependent strategies have been adopted to various degrees in, for example, the design of websites [Lin, et al 2000], the drawing of architecture sketches [Gross 1996], and the entry of mechanical diagrams [Alvarado 2001]. By reducing the space of possible interpretations, domain constraint serves to simplify the system's problem of recognizing structure in raw content and command data. While highly constrained systems do support editing and entry tasks in the domain for which they were devised, these constraints become barriers to sketching outside narrow limits. In a sketch system that interprets and re-renders all four-sided figures as rectangles, it can become impossible to draw a rhombus.

Conversely, unconstrained tools for drawing and editing sketches do not deal with sketch data in meaningful ways. Paint-style programs enable unlimited editing on a pixel-by-pixel basis, but offer little support for selecting salient collections of pixels in sketches beyond rectangle drag and perhaps lassoing selection commands. Digital ink editors (e.g., [Pedersen et al 1993]) sometimes manage to perform simple temporal grouping of strokes, but few of them break strokes into salient fragments or form more complex groups.

To build intelligent sketch editing tools for situations in which there is no *a priori* identified application domain, we believe that the early levels of Perceptual Organization in the human visual system provide inspiration for an approach to the extraction of salient visual structure. Classically, Perceptual Organization has been concerned with the Gestalt Laws of Grouping, such as *smooth continuation*, *common fate*, *symmetry*, *similarity*, *proximity*, and *closure* [Koffka 1922, Wertheimer 1923, Kanizsa 1979, Witkin and Tenenbaum 1983]. More recently, Computer Vision researchers have attempted to model these and related processes computationally (see e.g. [Stevens 1978, Lowe and Binford 1983, Zucker 1983, Boyer and Sarkar, 1999]). Although Perceptual Organization for general visual scenes will remain a daunting challenge for quite some time, intermediate visual structure in many sketches and diagrams is more readily accessible.

## Application and Test Platform: The ScanScribe Document Image Editor

Our group has been developing this idea in the context of a perceptually-supported document image editor, called *ScanScribe*, which builds on our earlier work in this area while it extends the range of image material that can be manipulated and the kinds of image structure recognized. Like PerSketch [Saund and Moran, 1994], ScanScribe is designed to manipulate primitive, or *prime* objects, as well as groupings or collections of these, called *composite* objects. These relations can occur as lattice structures and are not limited to tree hierarchies, as shown in Figure 1. While PerSketch was targeted at digital ink manipulation, ScanScribe is designed to work also with image regions as manipulable and groupable objects.

By virtue of placing users in the loop, in an interactive editing application, some burden of accuracy is removed from recognition technologies. Accordingly, UI techniques

# CueTIP: A Mixed-Initiative Interface for Correcting Handwriting Errors

*Michael Shilman, Desney S. Tan, Patrice Simard*
Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
{shilman, desney, patrice}@microsoft.com

**ABSTRACT**
With advances in pen-based computing devices, handwriting has become an increasingly popular input modality. Researchers have put considerable effort into building intelligent recognition systems that can translate handwriting to text with increasing accuracy. However, handwritten input is inherently ambiguous, and these systems will always make errors. Unfortunately, work on error recovery mechanisms has mainly focused on interface innovations that allow users to manually transform the erroneous recognition result into the intended one. In our work, we propose a mixed-initiative approach to error correction. We describe CueTIP, a novel correction interface that takes advantage of the recognizer to continually evolve its results using the additional information from user corrections. This significantly reduces the number of actions required to reach the intended result. We present a user study showing that CueTIP is more efficient and better preferred for correcting handwriting recognition errors. Grounded in the discussion of CueTIP, we also present design principles that may be applied to mixed-initiative correction interfaces in other domains.

**ACM Classification:** H.5.2 [Information Interfaces and Presentation]: User Interfaces - Graphical user interfaces, Input Devices and Strategies, Interaction styles, User-centered design; I.7.m [Document and Text Processing]: Graphics recognition and interpretation.

**General terms:** Design, Human Factors, Performance.

**Keywords:** Correction interface, mixed initiative, handwriting recognition, constraints, user study.

**INTRODUCTION**
The study of handwriting has been a topic of research in many fields, including psychology, neuroscience, physics, computer science, anthropology, education, forensic documentation, and others [11]. Much of this work has led to significant advances in recognition systems that automatically transform handwritten language into its symbolic representation. Because of the inherent ambiguities that exist in human handwriting, we believe that these systems will always make recognition errors. While many researchers continue to work on improving recognizer accuracy, there has been much less work exploring recovery mechanisms with which users can correct errors once they are made. Within this work, researchers have made heuristic and interface innovations that allow users to more efficiently manually correct text [e.g. 4,7]. Unfortunately, because correction has traditionally been viewed as an editing task, these systems often require laborious corrections that derail the flow of the text input task.

In this paper, we focus on exploring a mixed-initiative approach that reduces the cost of correction. Rather than treating correction as a manual editing task to be performed solely by the user, we construct a system that continues to assist the user even as they make corrections (see Figure 2 for the augmented correction flow). We characterize recognition as an optimization process and show that by treating user corrections as constraints to this optimization, the system can continue to 'steer' its guesses and attain the right answer in many fewer steps than would be possible with traditional alternatives (see Figure 1). This allows us not only to accelerate the correction process, but also to streamline the interface and interaction.

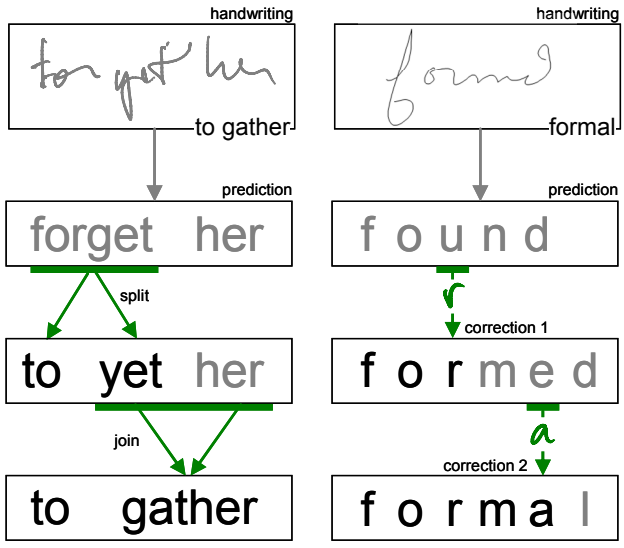



Figure 1: User handwrites some text and in order to attain the intended phrase, has to (left) split a word and then join two; (right) correct two letters. Using standard techniques to correct these same corrections would take many more operations.

EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2007)
M. van de Panne, E. Saund (Editors)

# Designing a Sketch Recognition Front-End: User Perception of Interface Elements

Paul Wais, Aaron Wolin, and Christine Alvarado

Department of Computer Science, Harvey Mudd College, Claremont, CA
{pwais,awolin,alvarado}@cs.hmc.edu

**Abstract**

*Programs that can recognize students' hand-drawn diagrams have the potential to revolutionize education by breaking down the barriers between diagram creation and simulation. Much recent work focuses on building robust recognition engines, but understanding how to support this new interaction paradigm from a user's perspective is an equally important and less well understood problem. We present a user study that investigates four critical sketch recognition user interface issues: how users integrate the process of triggering recognition into their work, when users prefer to indicate which portions of the diagram should be recognized, how users prefer to receive recognition feedback, and how users perceive recognition errors. We find that user preferences emphasize the importance of system reliability, the minimization of distractions, and the maximization of predictability.*

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces: *Evaluation/methodology, Interaction styles, Prototyping, User-centered design*

Categories and Subject Descriptors (according to ACM CCS): I.5.4 [Computing Methodologies]: Pattern Recognition: *Applications*

## 1. Introduction

Many engineering classes rely on simulation technologies to help students understand the systems they design. Unfortunately, mouse and keyboard interfaces to these programs are cumbersome. Students in these courses draw countless diagrams on paper (or on a Tablet PC) because sketch-based diagram creation is quicker and more natural. In fact, many instructors require students to draw diagrams on paper before entering designs into simulation software so that they focus on their design, not on the software interface.

Systems that can recognize and simulate students' hand-drawn sketches have the potential to lower the cognitive barrier between students and simulation software. However, these systems face their own interface challenges.

One challenge is how to allow users to trigger recognition and how to display recognition feedback. Feedback can be distracting in the early stages of design [HLLM02], but it can also aid recognition as it can help users adapt their drawing styles to match the system's expectations. Researchers have evaluated the usability of various recognition triggers and feedback mechanisms in isolation (e.g., [AD01, NLHL03, LaV06]), but have not compared different techniques directly.

A second challenge is how to allow users to indicate which pieces of the diagram the system should attempt to recognize. Students' homework often consists of a mix of text, equations, and diagrams. Despite advances in parsing heterogeneous notes [WSR06], a recognition system must receive only a single type of input to be practical. Most recognition systems allow the user to draw only one type of input (e.g., electrical circuits [GKS05]), while a few systems allow the user to manually select pieces of their drawing to be recognized after they have finished drawing [LaV06]. Again, little is known about which interface users prefer.

A third challenge is to minimize the impact of recognition errors on usability. Some systems reduce errors by placing constraints on users' drawing style. Others focus on intuitive error correction mechanisms as a way of reducing the impact of recognition errors [MHA00]. We believe that understand-