Using Context to Resolve Ambiguity in Sketch Understanding

by

Sonya J. Cates

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2006

Author ....
Department of Electrical Engineering and Computer Science
January 10, 2006

Certified by ....
Randall Davis
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Using Context to Resolve Ambiguity in Sketch Understanding

by

Sonya J. Cates

Submitted to the Department of Electrical Engineering and Computer Science
on January 10, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis presents methods for improving sketch understanding, without knowledge of a domain or the particular symbols being used in the sketch, by recognizing common sketch primitives. We address two issues that complicate recognition in its early stages. The first is imprecision and inconsistencies within a single sketch or between sketches by the same person. This problem is addressed with a graphical model approach that incorporates limited knowledge of the surrounding area in the sketch to better decide the intended meaning of a small piece of the sketch. The second problem, that of variation among sketches with different authors, is addressed by forming groups from the authors in training set. We apply these methods to the problem of finding corners, a common sketch primitive, and describe how this can translate into better recognition of entire sketches. We also describe the collection of a data set of sketches.

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

Acknowledgments

I would like to thank my advisor, Randy Davis, for his wisdom, support, and inexhaustible patience. Thanks also to all of the members of the Design Rational Group for their conversations and friendship, to my family for their support, and to the members of the MIT Cycling Team for a life outside of my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Consider a collaborative meeting of engineers, designers, scientists, or artists. Take away the whiteboard, pens, and paper and you will be left with a roomful of unhappy and unproductive people making frantic and indecipherable gestures. Communication naturally occurs in different forms, and sketches communicate some information most effectively, as for example, a complex diagram or map. As computers become more powerful and more pervasive, we would like both to communicate with them in the same efficient ways that we communicate with people and to use them to facilitate interaction and collaboration among people. This desire for more natural human-computer interaction and for a more effective role for computers in the design process is one compelling motivation for the development of pen based input for computers. While tablet computers are currently useful for taking notes and recognizing hand-writing, much remains to be done before a computer can recognize and understand unrestricted sketching.

1.1 Sketch Understanding and Early Sketch Processing

Ultimate goals for sketch based interfaces presented by [4] and [5] include human-like understanding, reasoning, and participation. While these are distant goals, a clear intermediate step is recognizing what has been drawn, i.e. associating a meaningful label with a collection of pen strokes. However, recognition is not always straightforward and typically requires several layers of processing, with results from early processing used in later phases.

One approach to early sketch processing is to find and label features, basic elements of a sketch that are important in recognition of high level objects. These features occur at a variety of scales. Some, like the presence of a corner, describe a very small area. Some, like whether the shape is closed, describe an entire stroke or several stokes, and some, like whether a portion of a stroke is straight, arise at an intermediate scale. Once recognized, such features may then be used to recognize higher level objects like quadrilaterals, arrows, resistors in a circuit diagram, etc. For example, a quadrilateral can be detected by testing whether the shape is closed, has four corners, and the segments between the corners are straight. We take this approach of hierarchical recognition in order to add new shapes easily and to give the user the flexibility to use previously undefined shapes in a sketch.

1.2 Difficulties in Early Sketch Processing

Classifying features can be problematic because of ambiguity common in sketches. This work examines two sources of ambiguity in early sketch processing. First, because sketches are almost never drawn perfectly, they contain internal inconsistencies, i.e., the same feature may have a different meaning in different parts of a sketch, even when drawn by the same person. Second, sketching styles often vary considerably across users and across domains, making it difficult to construct a general system capable of correctly classifying all instances of corners, parallel lines, straight lines, etc., across users and domains.

As a result, no simple function can correctly classify all features, even for a single sketch. Yet humans usually agree on the intention of a local feature in spite of ambiguities and, significantly, without extensive domain knowledge or prior knowledge of an individual's drawing style. For example, in Figure 1-1, nearly identical partial

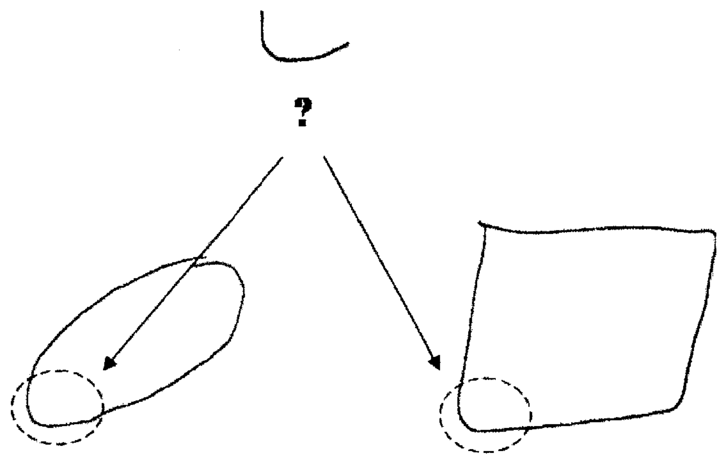


Figure 1-1: An example of local ambiguity. Considered out of context, the segment shown may equally well be an arc of the ellipse or a corner of the square.

strokes receive different interpretations. We form a different interpretation for the shapes at the bottom of the figure, even though it is impossible to distinguish between the side of the ellipse and the corner of the square with only local information, i.e., examining only the amount of the stroke shown at the top of the figure. This example illustrates the type of local ambiguity common in even relatively neat drawings. A template-based approach could easily distinguish between the ellipse and the square, but as we also aim to preserve the adaptability of a more general approach, a local unconstrained interpretation is preferred.

Figure 1-1 illustrates that identifying corners is important to a higher level understanding of shapes and, ultimately, entire sketches. Corners occur commonly and are often ambiguous, as this example demonstrates. As a result, this work focuses on corners. However, the methods presented here may be applied to other sketch elements as well.

1.3 Contributions

This work makes three main contributions: the collection of a data set and the introduction of two methods for improving early sketch recognition. These methods are applied to the problem of labeling corners.

Creating a system to recognize or interpret data often requires a data set for training and testing, and while some sketch data exists, current data sets do not yet approach the level of those in more established fields, such as speech recognition. Chapter 2 describes how a data set was created by collecting sketches from many authors, with each author given several sketching tasks. This data is intended to be generally useful for answering both those questions addressed in this work and future research questions.

Chapter 3 proposes using limited context, i.e. a small surrounding area of the sketch, to improve recognition of a localized feature. Results demonstrate that accuracy in recognizing a corner may be improved by incorporating knowledge about other possible corners that are nearby. We present a method based on belief propagation that has an accuracy of 86% in recognizing corners, improving over a baseline classifier that has an accuracy of 71%.

Chapter 4 discusses variation in drawing styles and presents results on improving corner recognition by considering differences and similarities among sketch authors. Results suggest that grouping similar authors in the training set can improve recognition for a previously unseen author. Applied independently of the method presented in Chapter 3, the method presented in this chapter achieves a corner classification rate of 85% with a sufficiently large training set divided into two groups, while a Support Vector Machine classifier applied to one group has a classification rate of 80%.

Chapter 2

Data Collection

This thesis describes methods to improve recognition of sketch primitives, which are often ambiguous, and applies these methods to the problem of recognizing corners. Creating a system to recognize or interpret noisy input often requires a large data set for training and testing. Although others have collected sketch data and made the data sets available, and there has been work to create a larger collection of sketch data from multiple sources by [7], sketch data sets have not yet reached the size and scope of standard data sets in other more established fields such as text understanding and speech recognition. Thus, there is still a need for the collection of data, both to investigate specific techniques, such as the ones in this thesis, and to form a general corpus. In this chapter we discuss the data collection that was conducted as part of this work. Training and test sets referred to in later chapters are drawn from the data set described here.

2.1 Objectives

This data was collected with several objectives. The first was to collect data from many authors in order to assemble a sample large enough to capture variations in drawing style and differences among authors. Our second objective was to collect data from multiple domains, since this work examines corners, a primitive that occurs in many domains. Existing sketch data includes several domains, but was collected from different authors under different conditions, i.e. with different pen input devices and with different instructions. We have collected sketches from many authors each sketching in several domains under the same conditions. Finally we sought to collect data that would be useful for answering future research questions as part of a larger corpus.

2.2 Sketch Collection Methods

Sketches were collected on an Acer Tablet PC and stored as collections of strokes. Strokes were represented as a series of points consisting of a screen position (x and y coordinates) and a time. The sketching interface was similar to a pen on paper, in the sense that the subjects could not erase but could scratch out mistakes or start over if they wished. To allow for a larger drawing area without distraction, instructions for each task were provided separately on paper, giving the subjects an essentially blank screen to draw on. Authors were allowed as much time as they wished to become familiar with the Tablet PC before beginning, and were given the option of completing the sketches in two sessions, since completing all the sketches took one and a half hours on average.

2.3 Sketching Tasks and Instructions

Each author was given four sets of instructions, corresponding to four domains: family trees, flow charts, floor plans, and maps. These domains were chosen because they are either familiar or easy to learn, and do not require any specialized knowledge. This allowed us to collect sketches from authors with diverse backgrounds rather than focus on one group, such as artists or engineers.

The four domains can be divided into two groups: symbolic (family trees and flow charts) and iconic (floor plans and maps). Figure 2-1 contains two examples of symbolic sketches and Figure 2-2 contains two examples of iconic sketches. In a symbolic domain, sketches are composed of a fixed number of predefined symbols. For

People

Male	
Female	

Relationships

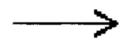
Marriage or Partnership	
Divorce	
Parent/Child	

Table 2.1: Symbols for the family tree domain.

example, the symbols used in flow chart and family tree sketches are given in Table 2.1 and Table 2.2. Authors were shown these symbols and instructed to use them in their sketches. We define an iconic domain as one where the things drawn resemble the objects they are intended to indicate. Iconic domains do not necessarily have a predefined set of symbols. For these domains, authors were asked to use any symbols or drawing style they felt were appropriate to complete the tasks. Authors were also asked to make some labeled and some unlabeled sketches. Examples of labeled sketches are in Figure 2-3. These sketches were collected for future investigations into combining handwriting recognition and sketch recognition.

Authors were asked to make three sketches in each domain:

- Family Trees
  - Given a textual description of a family, draw the corresponding family tree.
  - Draw the family tree for your family or one that you have made up.
  - Redraw the previous sketch, this time including labels for people.
- Floor Plans

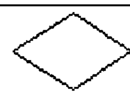
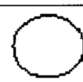
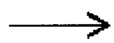
Start or End		This symbol is used to indicate the starting and ending points of the process.
Action or Step in a Process		This symbol represents a single step or action in the process.
Decision		This symbol occurs when there is a decision to be made. There will be two or more possible outcomes (branches).
Connector		This is an optional symbol used to indicate that the flowchart continues elsewhere (at the matching symbol). If necessary, use this symbol to avoid running out of room.
Flow Line		This symbol connects the steps in the process and indicates their order.

Table 2.2: Symbols for the flow chart domain.

  - Draw the floor plan for your house or apartment.
  - Draw the floor plan for another space you visit often.
  - Redraw the previous sketch, this time including labels for rooms and objects.

- Flow Charts
  - Given a textual description of a process, sketch the corresponding flow chart.
  - Draw a flow chart that describes your morning routine or any other process

you are familiar with.

- Redraw the previous sketch, this time including labels for steps in the process.

- Maps
  - Sketch a map of the route you take from home to work or school.
  - Sketch a map of another route you take frequently.
  - Redraw the previous sketch, this time including labels for roads and buildings.

Finally, authors were asked to make sketches that they might normally make as part of their work, hobbies, or school work. This task was optional and its purpose was twofold: to explore possible applications for a sketch interface and to collect sketches from domains with which the authors were very familiar. This will be useful for future exploration into the impact that the author's familiarity with the task has on a drawing.

2.4 Data Set

Sketches were collected from nineteen authors who were mostly unfamiliar with pen based computing. Only two authors reported that they had a lot of experience with pen input devices; most had no experience with such devices. We collected two hundred and fifty sketches divided equally among the four domains and used these to build the data set. (Including mistakes, warm-up sketches, and optional sketches, a total of nearly five hundred sketches were collected.) Examples referred to earlier in Figures 2-1, 2-2, and 2-3 are part of the data set.

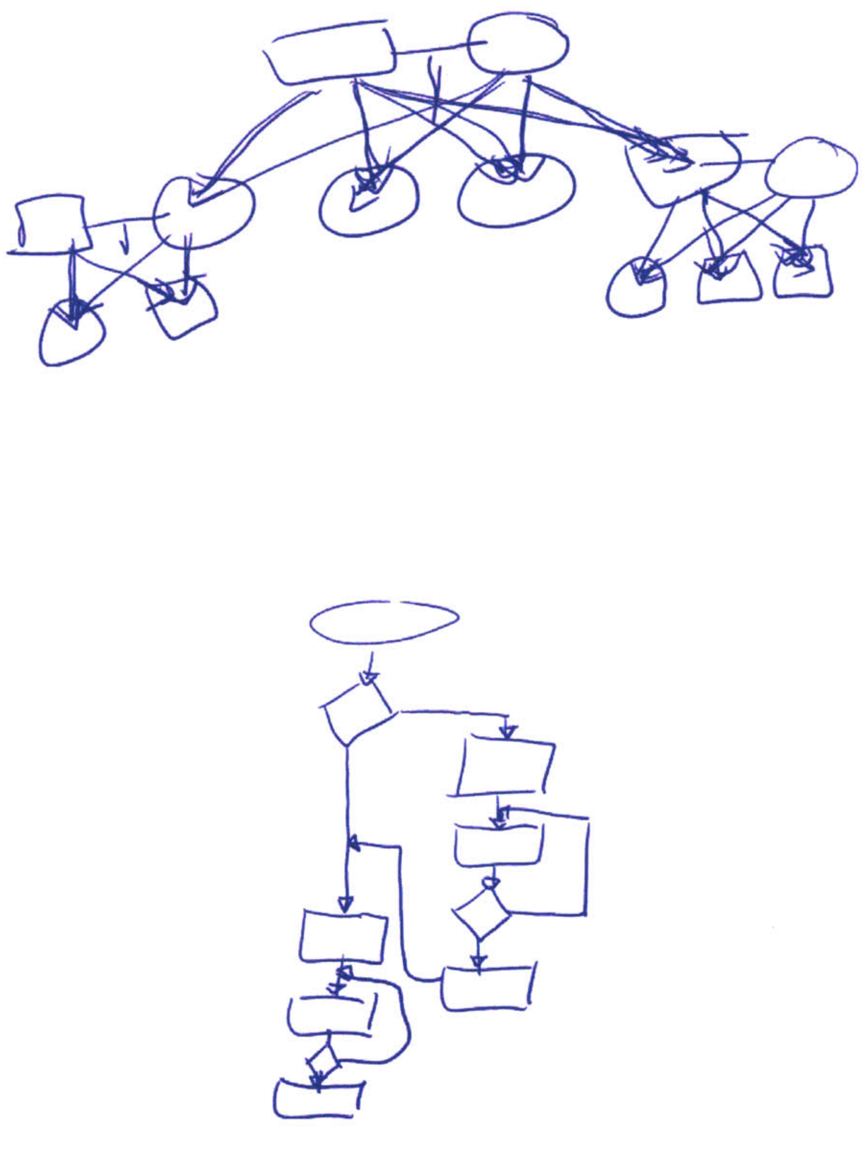


Figure 2-1: Examples of symbolic sketches: a family tree and a flow chart.

Figure 2-2: Examples of iconic sketches: a floor plan and a map.

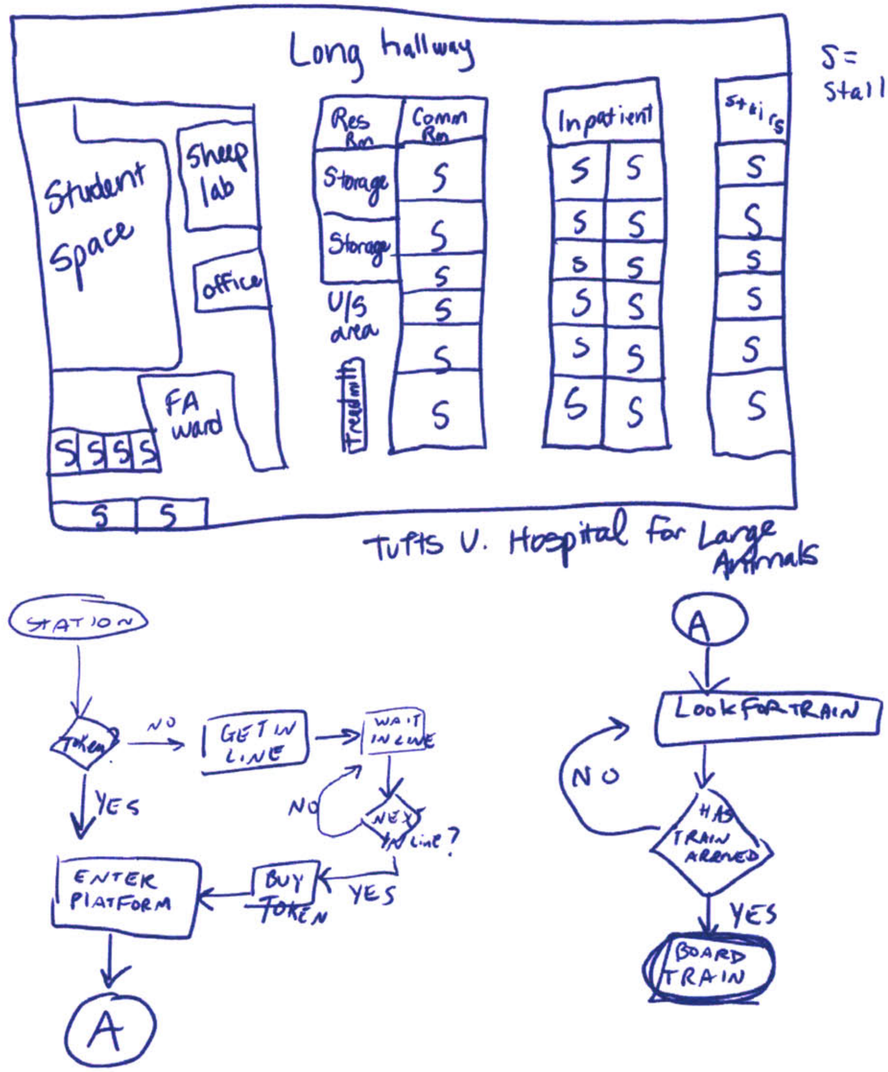


Figure 2-3: Examples of labeled sketches: a floor plan and a flow chart.

Chapter 3

Using Limited Spatial Context to Improve Recognition

The example presented earlier and reproduced in Figure 3-1 illustrates local ambiguity. While this ambiguity can sometimes be resolved by examining the entire shape, it is useful to be able to resolve these ambiguities without knowledge of the shape being drawn. This not only allows us to use the features to recognize the sketched object, it also enables processing of new symbols. This chapter presents a method for disambiguation by incorporating some spatial context, i.e. a part of the sketch surrounding the ambiguous area, without resorting to examining the entire shape.

Our method involves an application of Markov random fields (undirected graphical models) and belief propagation (an inference algorithm based on message passing), in which rough, initial interpretations of local features are improved with evidence from nearby areas. While we report here on the application of this approach to the problem of locating corners, we believe the approach may also be useful in resolving other ambiguous elements in sketches.

Markov random fields (MRFs) with belief propagation have been applied to problems in vision, such as locating text and shapes in images [12], [3], and for identifying relationships and entities in text understanding [8].

In our model, each stroke in a sketch is represented as an MRF, with each node in the MRF representing an area identified as a possible corner or an area in between

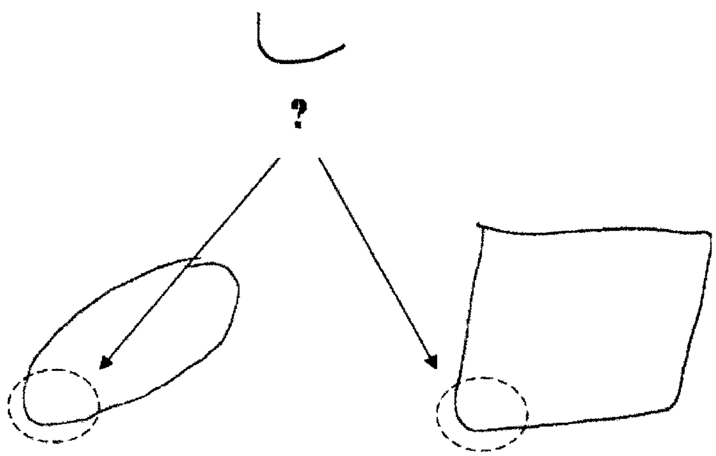


Figure 3-1: An example of local ambiguity. Considered out of context, the segment shown may equally well be the side of the ellipse and the corner of the square.

two possible corners. Connections between nodes establish the context for an area. Inference involves functions that represent the compatibility of the evidence associated with a single node or set of nodes and a particular labeling. We define these compatibilities based on qualitative principles. The results of this inference are beliefs for each node, which are used to determine a likely labeling for each possible corner.

3.1 Finding Possible Corners

As a first step, possible corners are located with a very weak classifier that finds all areas along the stroke with a direction change above a threshold. In the stroke in Figure 3-2, the possible corners are labeled c_1, c_2, and c_2. Given a low enough threshold, this method will find all corners, but generally also returns as many false positives as true corners.

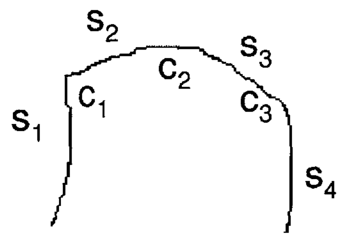


Figure 3-2: An example of a stroke with the possible corners labeled c_1, c_2, and c_2. Segments between possible corners are labeled by s_1, s_2, s_3, and s_4. Possible corners are found by selecting areas of the stroke with a direction change above a threshold.

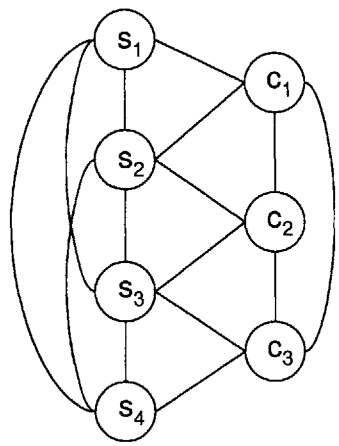


Figure 3-3: The graph associated with the stroke in Figure 3-2. Possible corners and the segments between them correspond to nodes. Edges connect possible corners and segments that are near each other.

3.2 Constructing the Graphical Representation of a Sketch

Each stroke is represented separately as an MRF. Figure 3-3 shows the graph associated with the stroke in Figure 3-2. Each possible corner is represented as a node with an associated random variable, which may have one of two labels, *corner* or *not corner*. The segments of the stroke between the possible corners are also represented as nodes.

Nodes are connected according to the following rules:

1. all n nodes of the same type are connected, forming a complete subgraph of order n

2. a node is connected to the closest nodes of different types

We define the closest nodes as those representing adjacent areas in the same stroke. Using this approach for the problem of detecting corners, all nodes representing possible corners are connected to each other, all nodes representing segments between possible corners are connected to each other, and a node representing a possible corner is connected to the nodes for the segments on either side of it. For example, in Figure 3-3, corner c_1 is connected to the other possible corners, c_2 and c_3, and to adjacent segments, s_1 and s_2. The neighbors of a node in the graph directly influence its interpretation and may be thought of as the context for the node.

Markov random fields are most commonly used as pairwise MRFs, meaning that the largest clique is of size two. In our representation, however, the graph will have cliques with size equal to the number of each type of feature. Inference on such a graph is more complicated and time consuming. We found, however, that due to the restricted size and structure of the graphs in our formulation, approximate inference may still be performed in real time. The benefit we find from using more connected (rather than pairwise) graphs is that we can define the compatibility functions more intuitively, defining functions that describe the relationship of a group of nodes, rather than only pairwise interactions. The work in [12] makes a similar argument for the use of higher order models.

3.3 Compatibility Functions

We define compatibility functions for each clique in the graph and for each individual node. The functions are a measure of compatibility between a particular labeling and the underlying data, and may be thought of as a measure of certainty for a particular labeling of a single node or a set of nodes. For a single node that has two possible labelings, e.g. *corner* or *not corner*, we compute a compatibility between each label and the observations made of the data: distance between the endpoints, arc length, direction change, and speed. For a group of nodes forming a clique, we compute a compatibility for all possible labelings of the group. (While there are an exponential number of such labelings, the set size is small in practice, so the computation is still feasible.) These multi-node functions convey context; nodes within a group may influence each other, but nodes that do not occur together in a group do not.

In our model for finding corners, we define four functions: one for single nodes, one for all the nodes of the same type, one for a corner and its surrounding segments, and one for a segment and its surrounding corners.[1]

3.3.1 Single node functions

The single node function is computed as a weighted combination of the measurements noted above, with the weights for each measurement determined by performing logistic regression on a small labeled data set. The single node function is defined by:

$$\psi_{c_i}(x_{c_i}, y_{c_i}) = (1 + exp(-(w_0 + \sum_j w_j f_j(x_{c_i}))(2 - y_{c_i})))^{-1}$$

where ψ_{c_i} is the compatibility between a label y_{c_i} and an observation x_{c_i}. The w_j's are weights, $f_j(x_{c_i})$ is the j^{th} measurement, and a label y_{c_i} is zero or one. This creates a soft step function that approaches zero and one asymptotically. Intuitively, labels, which may have values zero or one, are never assigned with complete certainty; this allows the labels and certainties to be updated based on other functions described below. For segments, $\psi_{s_i}(x_{s_i}, y_{s_i})$ is defined similarly and represents compatibility with the labels *straight* and *not straight*.

These functions can be used alone to determine a labeling for the features, independent of context; we use them later as a baseline for comparison.

3.3.2 Functions of the same node type

We next define a compatibility for a labeling of all the possible corners and a compatibility for a labeling of all the connecting segments. Intuitively, similar things should

[1]This is not defined for segments at the ends of a stroke.

be given the same label, and different things different labels. The group compatibility function conveys this heuristic. The group is split according to the labeling under consideration. For example, to find the compatibility of the possible corners in Figure 3-3 with the labeling $\{c_1 = \textit{corner}, c_2 = \textit{not corner}, c_3 = \textit{corner}\}$ subgroups $\{c_2\}$ and $\{c_1, c_3\}$ are formed. Then the distance between the subgroups and the average distance from the mean in each subgroup are compared. This is defined by:

$$\psi_{c_1 \ldots c_n}(x_{c_1}, \ldots, x_{c_n}, y_{c_1}, \ldots, y_{c_n}) =$$
$$\Big(1 + exp\Big(w_0 + w_1 d(m_0, m_1) + \frac{w_2}{|G_0|} \sum_{y_{c_j} \epsilon G_0} d(m_0, x_{c_j}) + \frac{w_3}{|G_1|} \sum_{y_{c_j} \epsilon G_1} d(m_1, x_{c_j})\Big)\Big)^{-1}$$

where $\psi_{c_1 \ldots c_n}$ is a compatibility between the labels y_{c_i} and observations x_{c_i}. G_0 is one subgroup with a common label, having mean m_0, and G_1 is the other subgroup with a common label, having mean m_1. d is a distance measure between observations, which we define as a simple Euclidean distance. The constants w_0, w_1, w_2,and w_3 are determined by hand. $\psi_{s_1 \ldots s_n}(x_{s_1}, \ldots, x_{s_n}, y_{s_1}, \ldots, y_{s_n})$ is defined similarly.

Note that this function is symmetric in that a labeling and the opposite labeling will have the same value, e.g. $\{\textit{corner}, \textit{not corner}, \textit{not corner}\}$ will have the same value as $\{\textit{not corner}, \textit{corner}, \textit{corner}\}$.

3.3.3 Functions of different node types

Finally, compatibility among adjacent nodes of different types is defined. These functions are specific to the types of features being considered. For detecting corners, two functions are defined, each with three arguments. The first function takes one possible corner and two surrounding segments; the other takes one segment and two surrounding possible corners. In both cases the function is based on the intuition that a corner should have a larger direction change than the surrounding area, and that a corner should be relatively short (a large direction change over a very long distance

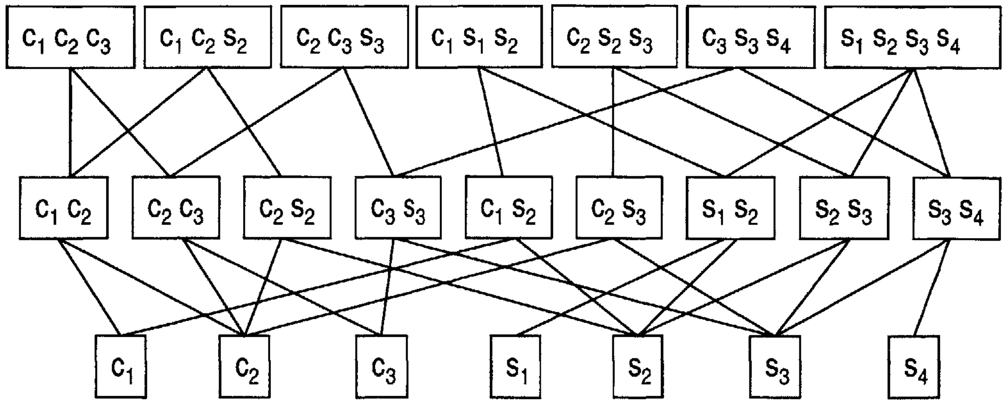


Figure 3-4: The region hierarchy for the graph shown in Figure 3-3. Messages are passed from a region to its subregions. Each line connecting two rectangles represents a message.

generally does not look like a corner). This function is defined by:

$$\psi_{s_i c_i s_{i+1}}(x_{s_i}, x_{c_i}, x_{s_{i+1}} y_{s_i} y_{c_i} y_{s_{i+1}}) = (1 + exp(w_0 + w_1 f_1(s_i, c_i, s_{i+1}) + w_2 f_2(s_i, c_i, s_{i+1})))^{-1}$$

where f_1 is a function comparing direction change of each piece, s_i, c_i, s_{i+1}, of the stroke and f_2 is a function comparing their arc length. $\psi_{c_i s_i c_{i+1}}$ is defined similarly.

3.4 Determining a Labeling

Given a graph as described in the previous section, we want to find an assignment of labels to random variables that has maximum likelihood. Because a distribution over an undirected graph may be factored as a product of the potential functions of the maximal cliques, this likelihood can be written as:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(x_C) \prod_j \psi_j(x_j)$$

where Z is a normalization constant, x_C are all the nodes in clique C, and ψ is as defined previously. Dependence on the observations is omitted from the equation for simplicity.

Belief propagation is a commonly used method for solving approximate inference problems on graphs; such algorithms work by passing local messages until the messages converge. Beliefs at each node (the certainty of a given label) are then computed as the product of the messages to the node and evidence at the node. A message from a to b maybe thought of as how certain a is that b has a particular labeling. Belief propagation is only well defined and guaranteed to converge for trees (acyclic graphs). However, it has been shown empirically that when applied to loopy graphs, belief propagation usually performs well [6].

In ordinary belief propagation, messages are exchanged between single nodes. Yedidia, Freeman, and Weiss outline generalized belief propagation, in which messages pass between groups of nodes, called regions [10], [11]. Messages pass from regions to their direct subregions, which are defined by the intersection of regions. Regions and subregions form a hierarchy; Figure 3-4 shows the hierarchy of regions of the graph in Figure 3-3. Rectangles represent regions and subregions; messages are passed from a region to one of its subregions where two rectangles are connected. In Figure 3-4, the rectangle containing c_1, c_2, c_3 is a region with direct subregions c_1, c_2 and c_2, c_3. The belief of a region is based on evidence from within the region and messages going into that region from outside.

We apply a variation of generalized belief propagation to the inference problem formulated above. The maximal cliques in the graphs are selected as the regions. We omit functions among nodes that form non-maximal cliques for simplicity and because this information is already expressed by the larger functions we have defined.

Consider the region containing c_2 and c_3 and its subregion containing only c_2. For each, the belief for that region is the product of the evidence within the region (the output of the functions described above) and the messages entering the region (but not messages passed within the region). The beliefs of these two regions are expressed as:

$$\begin{aligned} b_{c_2c_3}(x_{c_2}x_{c_3}) &= \psi_{c_2}(x_{c_2})\psi_{c_3}(x_{c_3})m_{c_1\rightarrow c_2c_3}m_{s_3\rightarrow c_2c_3}\cdots \\ &\quad m_{c_1\rightarrow c_2}m_{s_2\rightarrow c_2}m_{s_3\rightarrow c_2}m_{s_3\rightarrow c_3} \end{aligned}$$

and

$$b_{c_2}(x_{c_2}) = \psi_{c_2}(x_{c_2}) m_{c_1 \to c_2} m_{c_3 \to c_2} m_{s_2 \to c_2} m_{s_3 \to c_2}$$

where $m_{i \to j}$ represents a message from i to j. We may now find an expression for the message from c_3 to c_2, since

$$b_{c_2}(x_{c_2}) = \sum_{x_{c_3}} b_{c_2 c_3}(x_{c_2} x_{c_3})$$

Here we have summed over all possible labels for c_3. Solving for $m_{c_3 \to c_2}$ gives:

$$m_{c_3 \to c_2} = \sum_{x_{c_3}} \psi_{c_3}(x_{c_3}) m_{s_3 \to c_2 c_3} m_{c_1 \to c_2 c_3} m_{s_3 \to c_3}$$

Expressions for all the messages may be found similarly, and because the structure of the graphs does not vary, calculations of messages may be simplified because their compositions also do not change, i.e. we do not need to solve for every message as shown for $m_{c_3 \to c_2}$ above. First messages are initialized and updated iteratively, then the beliefs for the individual nodes representing possible corners are computed. These beliefs represent the certainty of a given label (*corner* or *not corner*) taking into account some context, indicated by edges in the graph.

3.5 Results

We have tested the approach presented above with the data set described in Chapter 2. Results are presented for family tree and flow chart sketches. These sketches were selected because these domains use symbols and can be labeled more easily by hand to obtain ground truth. The family tree sketches include 698 possible corners in 19 sketches, and the flow chart sketches include 485 possible corners in 18 sketches. Possible corners are areas of the sketch with a direction change above a threshold; they are classified as either *corner* or *not corner*.

We use the single node function described in Section 3.3.1 as a baseline. This function uses only local information and does not include information about the areas surrounding possible corners. Outputs from both the baseline function and the belief

	without context	with context
correct possible corner label (out of 1183)	837 (71%)	1014 (86%)
all possible corners in a shape labeled correctly (out of 610)	262 (43%)	418 (69%)

Table 3.1: Possible corner classification rates with and without context for family tree and flow chart sketches.

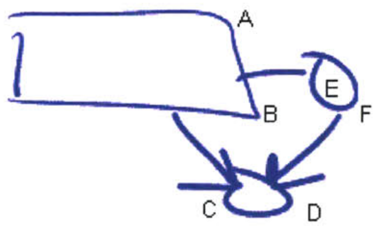
propagation are values between 0 and 1. A higher value indicates more certainty that the segment is a corner and a lower value indicates more certainty that the segment is not a corner. A possible corner with a value greater than .5 is regarded as a corner.

Table 3.1 presents classification rates for possible corners. "Correct possible corners" indicates the number of possible corners that were given the correct label (this includes those that were correctly labeled as *not corner*). "Correct shapes" indicates the number of shapes, e.g. ellipses and rectangles, in which all of the possible corners were labeled correctly. Results for the baseline function are listed in the column "without context" for comparison. Our results demonstrate that incorporating a small amount of context improves corner detection noticeably over a reasonable baseline.

Figure 3-5 contains two areas taken from larger sketches. These cropped areas show the output from our system more explicitly. Certainties without context (column 1) and with context (column 2) are listed in the tables beside each sketch.

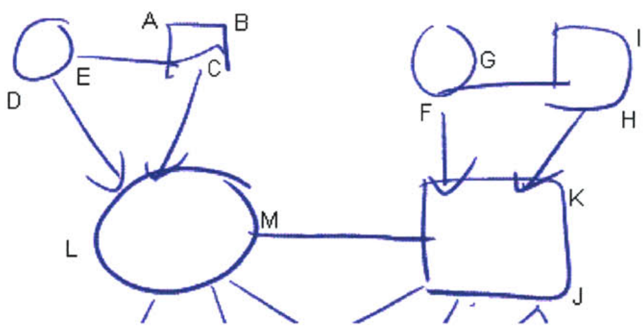
In Figure 3-5a, the corners in the rectangle (A and B) were correctly identified initially; however, their certainties were raised by the belief propagation, particularly in the case of A, which had a weak initial label. The upper ellipse containing possible corners E and F is similar: an initially correct labeling is strengthened. In nearly all cases where all of the possible corners were labeled correctly initially, belief propagation strengthens the initial guess.

In the lower ellipse of Figure 3-5a, both possible corners (C and D) were incorrectly assigned high values by the single node compatibility function. Although both values were lowered after belief propagation because of the influence of adjacent curved areas



	1	2
A	.56	.90
B	.98	.99
C	.84	.61
D	.76	.54
E	.20	.03
F	.31	.16

a



	1	2
A	.99	.99
B	.99	.99
C	.75	.96
D	.59	.24
E	.08	.05
F	.17	.10
G	.06	.04
H	.35	.48
I	.42	.75
J	.84	.96
K	.90	.97
L	.22	.07
M	.24	.11

b

Figure 3-5: Cropped areas of sketches from two different subjects. Tables to the right of each sketch give certainty values for each possible corner. The first column of numbers is before belief propagation; the second is after.

in the sketch, a shape with very strong but incorrect initial certainties can generally not be corrected.

In Figure 3-5b, the upper left ellipse has one possible corner, D, that was initially misclassified. However, the correct labeling of the other possible corner and influences from the areas between the possible corners, which are clearly curved, fix the initial errors.

Figure 3-5b also contains a case where belief propagation performed worse than the single node function by strengthening an incorrect label. Possible corner C was not intended as a corner. We assume this partly because we prefer to interpret the shape as a rectangle rather than as a concave pentagon with one very short side. In this case, higher level domain knowledge is needed to correct the mistake.

These examples demonstrate instances in which the method proposed in this chapter improves corner recognition and instances in which it does not. While the improvement in classification rate is significant, some ambiguities cannot be corrected with the limited context applied here and may require knowledge of the sketch lexicon or domain.

Chapter 4

Applying Author Groups to Recognition

Chapter 3 examined the application of spatial context to recognition; in this chapter we consider a different element of context, that of who drew the sketch. We have observed that some of the ambiguity present in sketches is due to variations in different authors' drawing styles. These individual variations make creating a system that accurately recognizes all sketching styles challenging. This chapter addresses the problem of accommodating many different drawing styles in a single recognition system.

Figure 4-1 presents four cropped areas of sketches from four different authors. These sketches were all drawn as part of the same task (constructing a family tree), yet differences due to the authorship of the sketches are evident among the sketches. Sketches 1 and 2 especially demonstrate the value of author context in recognition. In each of those two sketches two stroke segments are circled: a corner and a partial stroke that might be mistaken for a corner in the absence of shape or context knowledge. The mistaken slip of the pen in Segment C closely resembles the sharp corner in A that is typical of the corners in sketch 1 but much less common in sketch 2. The pointed end of the ellipse in B looks similar to segment D, one of the more rounded corners typical in sketch 2 but not in sketch 1. Even with very little information about the surroundings of these highlighted segments, knowledge of the drawing style

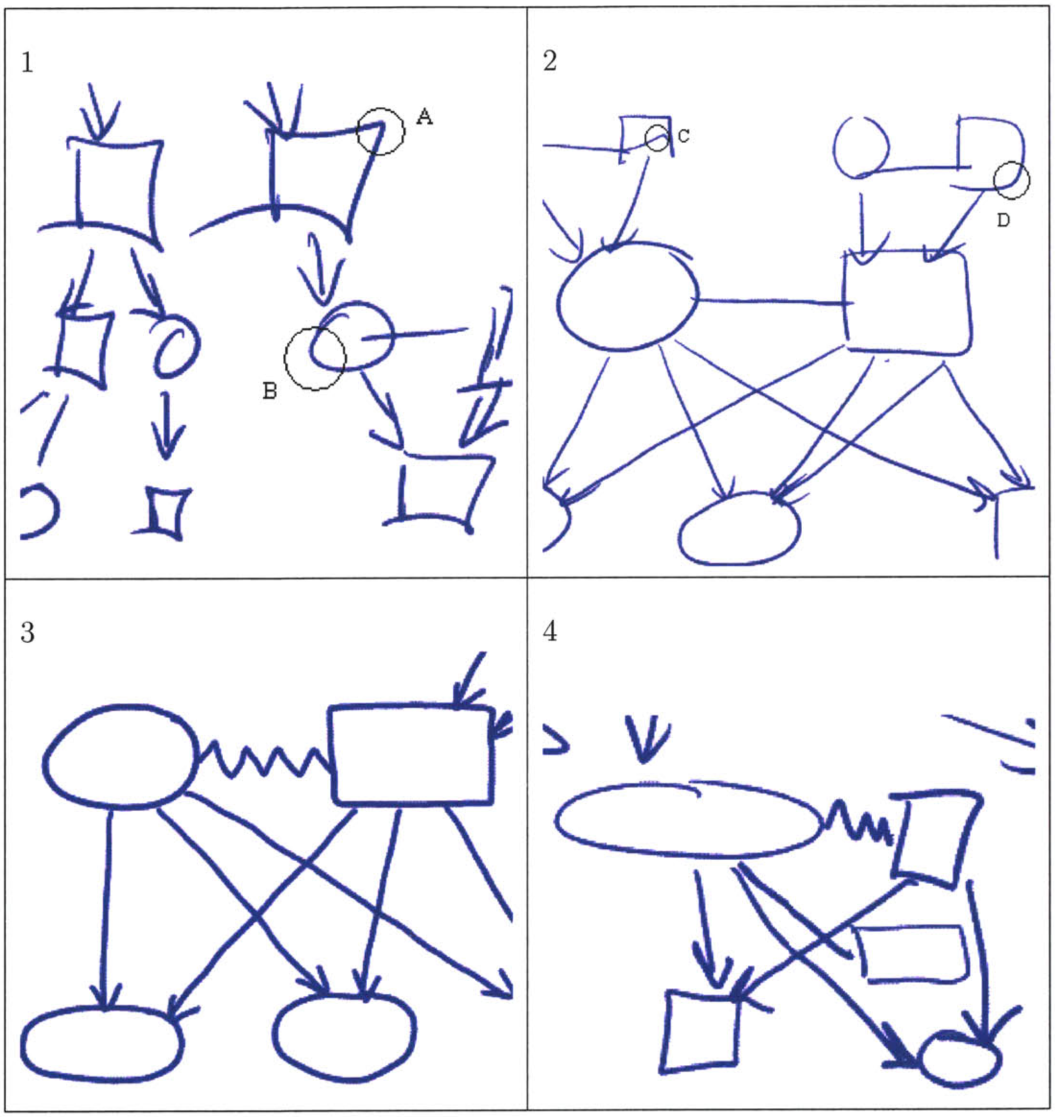


Figure 4-1: Partial sketches from four different authors with different drawing styles. Circled areas in sketches 1 and 2 highlight sources of ambiguity: similar looking areas in different drawings (segments A and C and segments B and D) require different interpretations. Knowledge about the authors' drawing style may help resolve these ambiguities.

of their authors may help disambiguate them.

One solution to the recognition problems created by differences in drawing styles among authors would be to create separate recognition systems for each user. This would require each new author to submit training data, and the amount of data required might be undesirable or even prohibitive for some applications. Ideally, we would like to take advantage of differences in drawing styles we have observed without requiring an extensive training phase from each new user. One solution is to create groups or clusters of authors with similar styles and base recognition on models trained separately for each group. This approach has been applied to unconstrained, writer independent handwriting recognition in [9].

This chapter presents a similar method for using differences in drawing style to improve recognition with very little data from each new author. We have created author groups, i.e. groups that consist of authors with similar drawing styles. These groups are determined using a scoring function that clusters authors in a training set. Classifiers are then constructed for each group. When a new author is encountered, the most similar group is selected based on a few examples from the new author, and the classifier trained with that training set group is used on the new author's sketches.

4.1 Determining Author Groups

For a given training set consisting of many sketches from each of n authors, we would like to find a division of the authors into m groups such that the similarity among authors within each group is maximized. This could be expressed as:

$$G' = \overset{argmax}{G} \left(\sum_{g \in G} (S(g)) \right)$$

Where G is a grouping which consists of groups $(g_1, g_2, ..., g_m)$ and S is a scoring function which returns a measure of similarity for a group of authors g.

4.1.1 Scoring Function

The formula above suggests one way to create a grouping, but even within that framework many scoring functions would be possible. Since our goal in constructing author groups is to make groups that are useful for classification, we tie the scoring function to the type of classifier that will be used later. We use Support Vector Machines (SVMs) in this chapter and describe a related scoring function.

SVMs are a method for supervised learning. Data is classified by finding the separating hyperplane which maximizes the margin (the distance between the hyperplane and the data points) while minimizing the training error, often by using a non-linear mapping into a feature space. The work in [2] provides a more detailed descriptions of SVMs. Here we are mostly interested in the output of training.

Given a training set X consisting of data points x with corresponding labels y, the result of training an SVM is a decision function of the form:

$$f(x) =< w, \phi(x) > +b = \sum_i \alpha_i y_i K(x_i, x) + b$$

Where w and b determine the hyperplane and ϕ is the mapping to the feature space. The kernel, K, is the inner product in the feature space (this avoids the calculation of the mapping, ϕ). We use a standard Radial Basis Function (RBF) in this chapter. The sum is over the support vectors, which are indexed by i, with weights α_i and labels y_i.

This decision function, f, is used for classification by taking the sign as a hypothesis, h, for what the label should be.

$$h = sign(f(x))$$

Using standard numerical methods, we find this decision function for each group g in a grouping G and then define a scoring function, S:

$$S(g) = \sum_{x \in X_g} f(x) \cdot y$$

The product $f(x) \cdot y$ will be negative if h (the hypothesized label) and y (the actual label) do not agree, and positive if they do agree. The magnitude of $f(x) \cdot y$

indicates the distance from the margin. X_g is the collection of data corresponding to the authors in group g.

The score for a grouping is the sum of the group scores:

$$S(G) = \sum_{g \in G} S(g)$$

The argmax from above is now:

$$\begin{aligned} G' &= \overset{argmax}{G} \left(\sum_{g \in G} (S(g)) \right) \\ &= \overset{argmax}{G} \left(\sum_{g \in G} \left(\sum_{x \in X_g} f(x) \cdot y \right) \right) \\ &= \overset{argmax}{G} \sum_{g \in G} \left(\sum_{x \in X_g} \left(\sum_i \alpha_i y_i K(x_i, x) + b \right) \cdot y \right) \end{aligned}$$

This determines an optimal grouping, G', for the authors in the training set. Our current work solves this exhaustively because this is part of a one time training phase, but other, faster approaches would also be possible.

4.1.2 Application to Corners

As in Chapter 3, we first find possible corners using a low threshold that also finds many false positives. We construct a vector for each possible corner, consisting of arc length, Euclidean distance between end points, maximum speed, minimum speed, average speed, direction change, direction change per unit length, total time, maximum curvature, minimum curvature, average curvature, and curvature variance. This vector corresponds to x, the data points, in Section 4.1.1 above. Each possible corner also has a label $y \in \{-1, 1\}$ where -1 indicates *not corner* and 1 indicates *corner*. We determine a grouping of the training set as described above and compute a classifier (SVM) for each group.

4.2 Encountering a New Author

When a new author is encountered, we need to select the best classifier for this author's sketches. Though this requires some labeled training data from each new author, a good choice may be made with very few examples. We have used five examples, including both positive and negative examples. This is the equivalent of drawing one square and one ellipse in most of the sketches in Figure 4-1.

We use the scoring function described in Section 4.1.1. For each new author a and group g we compute:

$$S(a,g) = \sum_{x \in X_a} (\sum_{g_i} \alpha_{g_i} y_{g_i} K(x_{g_i}, x) + b_g) \cdot y$$

Where X_a are the training examples from the new author. A score is computed for each group, g, and the classifier corresponding to the group with the largest score is selected.

4.3 Results

This section presents the results of applying this method to the family tree and flow chart data discussed earlier. We use data from 18 authors and randomly select a training set of n authors, leaving $18 - n$ authors to form a test set. We then divide the training set into groups to form a grouping, G', as described in Section 4.1.1. Since the results are dependent on the particular division of authors into the test and training sets, we perform multiple trials and find the average performance.

Figure 4-2 presents results for one, two, three, and n groups. Results for one group (line D) may be regarded as a baseline. In this case the training set is not divided, and data from authors in the test set is classified based on a model trained with all the available data. For the two group result (line C), training data is divided into two groups based on the scoring function described above. The three group case (line B) is similar. For the n group case (line A), each author in the training set is taken to be a separate group. For each division of the training set, authors in the test set are assigned to a group in the training set with the scoring function in Section 4.2. Data from each test set author is classified with an SVM trained only with data from that group. Results are averaged over 25 trials with training and test sets randomly selected for each trial.

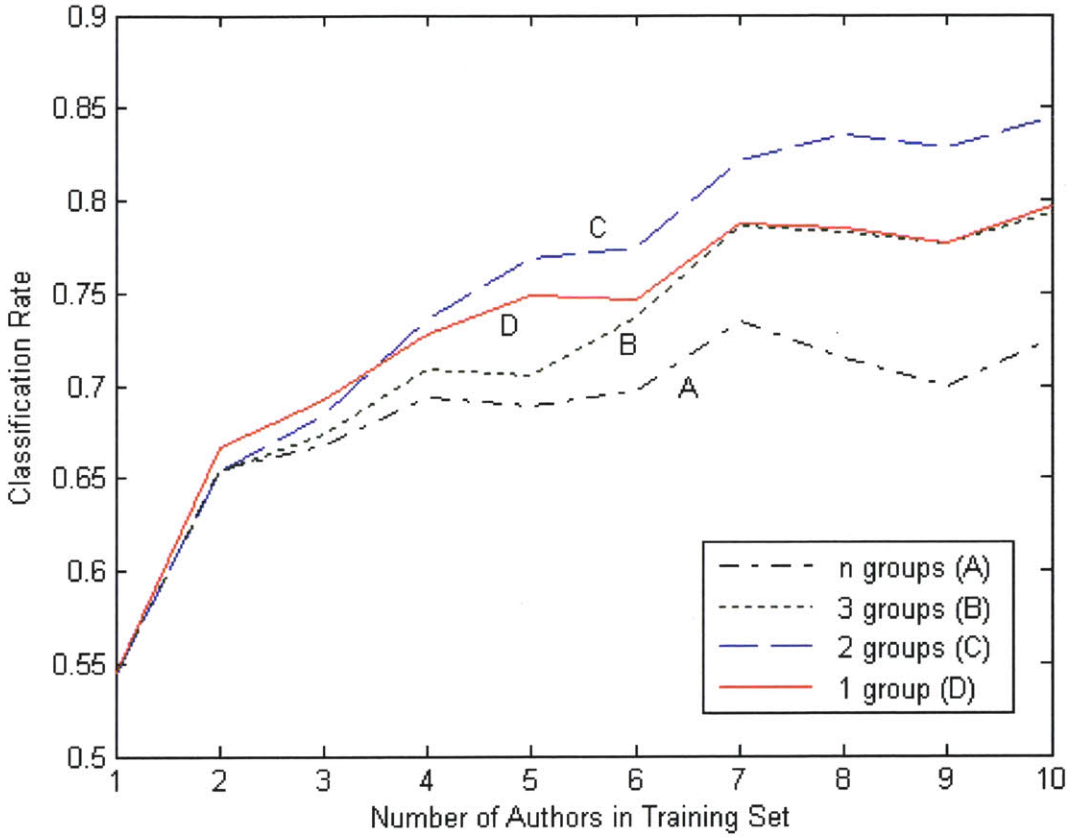


Figure 4-2: Classification rates for possible corners averaged over 25 trials for a training set divided into 1, 2, 3, and n groups. Training and test sets combined consist of 18 authors.

These results demonstrate that recognition of low level sketch elements may be improved by taking into account the drawing style of the authors if there are a sufficient number of distinct authors in the training set. With ten authors in the training set divided into two groups, our method had an accuracy of 85%, which improves over the baseline rate of 80%. We note that the slope of the lines decrease as more authors are added and we speculate that increasing the size of the training set or the number of authors in the training set will yield little additional improvement, particularly for the best cases of two groups and one group. Resolving the remaining error may require greater knowledge of what is being drawn.

Up to the point where domain knowledge is required for improvement, accuracy is limited by the size of the training set and the minimum number of people in a group. One possible extrapolation from these results is that the case of three groups (and possibly four or more groups) would approach and exceed the results for two groups. However, it is also possible that for the task of classifying corners, two groups is optimal and increasing the number of groups, even with a very large training set, will not yield better results. Further experiments and additional data would be required to resolve this question. Specifically, one would need to collect data from more authors in order to divide the training set into more groups while keeping the number of people in each group constant. More experiments would also be required to determine the optimal number of groups when sketch elements other than corners are included.

Chapter 5

Future Work

This chapter describes several directions for future work: combining the two approaches presented in this thesis to improve recognition further, expanding the methods presented here to include sketch primitives other than corners, and combining these methods with a higher level recognition system.

5.1 Combining Approaches

The two approaches presented here both seek to improve recognition of early sketch primitives. Both approaches use context, the sketch context and the author context, to disambiguate confusing parts of a sketch; however, we have applied the approaches separately. We speculate that recognition could be further improved by combining these methods and context sources. For example, multiple versions of the compatibility functions described in Section 3.3 could be used, similar to the way multiple Support Vector Machines are used to create group classifiers in Chapter 4.

5.2 Recognizing Other Primitives

This thesis focuses on the problem of finding and identifying corners. We have observed that corners present a particular problem in sketch recognition because they occur frequently and are often ambiguous. However, other basic elements of sketches

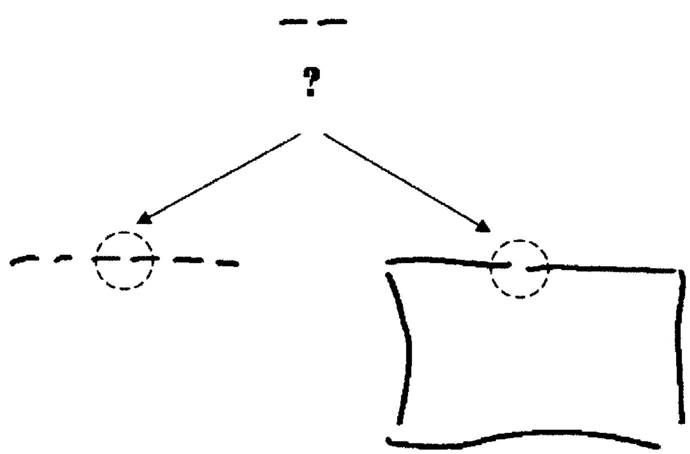


Figure 5-1: Another example of local ambiguity. Considered out of context, the two segments at the top could be separate lines or part of one longer line.

are also important for accurate recognition.

Figure 5-1 shows a different example of ambiguity. Depending on the surrounding area of the sketch the two partial strokes may be interpreted as either one continuous line or two adjacent lines. The method presented here for incorporating limited spatial context could also be applied to the recognition of this basic sketch element as well as others. For example, in the case of an ambiguous junction or gap, as in Figure 5-1 the graphical representation could have nodes corresponding to segments and to blank areas at the ends of segments and edges corresponding to distances within the sketch as before.

5.3 Combining with Higher Level Recognition

We have described how to improve recognition of corners; however, that is only a small part of recognizing an entire sketch. This work can be seen as part of the first stage of a complete recognition system. Its initial classifications could be used by later stages to produce more accurate interpretations for entire sketches. For example, systems such as [1] rely partly on the accuracy of initial recognition stages.

Chapter 6

Conclusion

This work focuses on improving the classification of primitive elements in sketches by resolving ambiguities. We present two methods for using context to improve recognition of sketch primitives and apply these methods to the problem of recognizing corners. The context includes information about the area surrounding a localized feature and information about the sketch's author. We chose to focus on corners because they occur frequently, are often critical to understanding, and are a common source of error. This thesis also describes a data set that was collected as part of the research.

The methods presented here focus on resolving ambiguities with limited contextual information, even though these ambiguities might be resolved using more knowledge, for example knowledge of the graphical lexicon used in the sketch domain. We have taken this approach because we believe that making local interpretations of small areas of a sketch is a fundamental capability and an important step towards accurate and unrestricted sketch understanding. The local classification we produce may be useful as a part of an early stage in a larger recognition system, where results from early stages are used as input to later stages that then incorporate shape and domain knowledge. These methods also provide a means to improve accuracy when no shape or domain knowledge is available, as for example when a new symbol or new domain is encountered as a result they are an appropriate focus of study for sketch understanding.

Bibliography

[1] Christine Alvarado. *Multi-Domain Sketch Understanding.* PhD thesis, Massachusetts Institute of Technology, August 2004.

[2] Christopher Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[3] James M. Coughlan and Sabino J. Ferreira. Finding deformable shapes using loopy belief propagation. In *Proceedings of the Seventh European Conference on Computer Vision*, 2002.

[4] Randall Davis. Sketch understanding in design: Overview of work at the MIT AI lab. *Sketch Understanding, Papers from the 2002 AAAI Spring Symposium*, pages 24–31, March 25-27 2002.

[5] Kenneth Forbus, R. Ferguson, and J. Usher. Towards a computational model of sketching. In *Proceedings of QR2000*, 2000.

[6] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in Artificial Intelligence*, 1999.

[7] Michael Oltmans, Christine Alvarado, and Randall Davis. Etcha sketches: Lessons learned from collecting sketch data. In *Making Pen-Based Interaction Intelligent and Natural*, pages 134–140, Menlo Park, California, October 21-24 2004. AAAI Fall Symposium.

[8] D. Roth and W. Yih. Probabilistic reasoning for entity & relation recognitions. In *Proceedings of International Conference on Computational Linguistics.*, 2002.

[9] J Subrahmonia, K Nathan, and M Perrone. Writer dependent recognition of on-line unconstrained handwriting. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, 1996.

[10] J Yedidia, W Freeman, and Y Weiss. Generalized belief propagation. Technical report, Mitsubishi Electric Research Laboratory, 2000.

[11] J Yedidia, W Freeman, and Y Weiss. Understanding belief propagation and its generalizations. Technical report, Mitsubishi Electric Research Laboratory, 2002.

[12] D.Q. Zhang and S.F. Chang. Learning to detect scene text using a higher-order MRF with belief propagation. In *IEEE Workshop on Learning in Computer Vision and Pattern Recognition.*, 2004.