
Sketch Interpretation Using Multiscale Models of Temporal Patterns

Tevfik Metin Sezgin*

Randall Davis

MIT CSAIL, 32 Vassar St. 235, Cambridge, MA 02139 USA

MTSEZGIN@CSAIL.MIT.EDU

DAVIS@CSAIL.MIT.EDU

Abstract

With the emergence and widespread use of Tablet PCs, recognizing freehand sketches has gained importance as an enabling technology for pen-based intelligent human computer interfaces. This paper presents a sketch recognition framework based on multiscale statistical models of temporal patterns. Our previous work in sketch recognition has shown that in certain domains the stroke orderings used when drawing objects contain temporal patterns that can aid recognition. We build on this work to show how sketch recognition systems can use knowledge of both common stroke orderings and common object orderings. We describe a statistical framework based on Dynamic Bayesian Networks that can learn temporal models of object-level and stroke-level patterns for recognition. Our recognition framework supports multi-object strokes, multi-stroke objects, and it supports real-valued feature representations using a numerically stable recognition algorithm. We present recognition results for hand-drawn electronic circuit diagrams. The results show that modeling temporal patterns at multiple scales provides a significant increase in correct recognition rates, with no added computational penalties.

1. INTRODUCTION

Sketching is a natural input modality of increasing interest. The emergence of hardware such as Tablet PCs and hand-held PDAs provides easy means for capturing pen input. These devices combine a display with a computing device, making it possible to capture and process sketches online (i.e., as they are drawn). Online recognition has two main advantages. First, it enables interpreting and displaying pen input as it is entered (e.g., an engineer drawing a circuit diagram sees ink change color in real-time to indi-

cate which circuit components are recognized). Second, in an online sketching system we have access to stroke ordering information as well as the end product of the drawing process.

Our previous work showed that in certain domains, temporal stroke orderings used when sketching objects contain predictable patterns that can be used for object recognition (Sezgin & Davis, 2005). 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, followed by a vertical line, followed by two pairs of positively and negatively sloped lines, corresponding to the head, body, arms and legs of the stick-figure.

Another kind of temporal pattern present in online sketches is **object-level patterns** that capture the probability of seeing a certain sequence of objects being drawn. Consider the domain of UML class diagrams, drawn by software designers using rectangles to indicate classes and a variety of different kinds of arrows to indicate relations among classes, such as inheritance, generalization, association, etc. In this domain, when a designer draws a new class (indicated by a rectangle), it is 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. Note that the kind of arrow to expect can depend on the kind of class drawn: a *final* class, for example, cannot be extended, limiting the kind of arrow that we may expect next.

In domains like UML, where there is a graphical grammar of sorts, it is plausible to imagine writing down the grammar and using this to guide a UML diagram recognition system. But not every domain is likely to have patterns that are as well understood as they are 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 in which various objects may combine together. We argue that a better way of incorporating object-level temporal patterns in a recognition framework would be to learn such features from data

*Corresponding author.

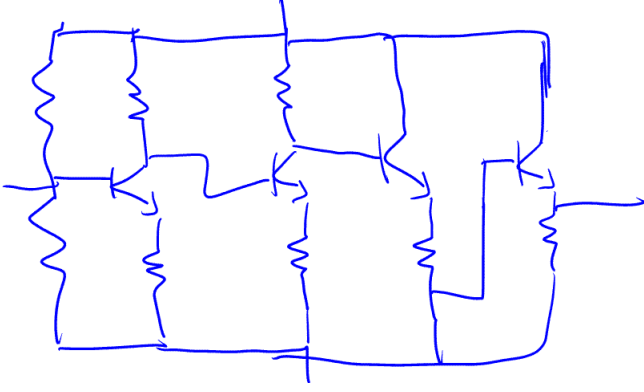


Figure 1. A hand-drawn circuit diagram showing what we mean by a sketch.

– along with stroke-level patterns – and use them in recognition.

In the rest of this paper, we present a sketch recognition framework where common object orderings that naturally appear in sketches can automatically be learned from data and used for sketch recognition. The key features that make our framework novel include the ability to learn object-level patterns from data, the ability to handle multi-stroke objects and multi-object strokes, and support for continuous observable features. We show how graphical models can be used to implement our approach efficiently and demonstrate that a specialized inference algorithm known as the Lauritzen-Jensen stable conditional Gaussian belief propagation should be used to avoid numerical instabilities in recognition.

We start with a formal description of the sketch recognition problem. Section 3 presents our approach to modeling common object-level and stroke-level patterns. Section 4 presents recognition results obtained for the electrical circuit diagrams domain. In section 5, we review related work, and conclude with a summary of our key contributions.

2. PROBLEM DESCRIPTION

2.1. Terminology

By a sketch, we mean messy, informal hand-done drawings (e.g., Fig. 1). Specifically we are interested in recognizing sketches with objects of a symbolic nature that can be represented using structural descriptions, that have been the focus of the sketch recognition community (Alvarado & Davis, 2004; Shilman et al., 2002; Mahoney & Fromherz, 2002; Gennari et al., 2004; Sezgin & Davis, 2005).

We formally define a *sketch* $\mathcal{S} = S_1, S_2, \dots, S_N$ as a sequence of strokes captured using a digitizer, preserving the

drawing order.¹ A *stroke* is defined as a set of time-stamped points sampled between pen-down and pen-up events during sketching.

Each stroke is preprocessed using the toolkit described in (Sezgin et al., 2001), converting the sketch $S_{1:N}$ to a time-ordered sequence of geometric *primitives* (line and arc segments) $P_{1:T}$. As a single stroke may result in more than one primitive, the total number of primitives is usually larger than the number of strokes.

We use *segmentation* to refer to the task of grouping together primitives constituting the same object, and by *classification* mean the task of determining which object each group of primitives represents (e.g., a stick-figure or a rectangle). Segmentation produces K groups $G = G_1, G_2, \dots, G_K$, and classification gives us the labels for the groups $L = L_1, L_2, \dots, L_K$. We define *sketch recognition* as the segmentation and classification of a sketch.²

By *observations* we refer to the sequence of features $\mathcal{O} = O_1, O_2, \dots, O_T$ obtained from the primitives. We use O_{G_i} to refer to the sequence of observations corresponding to a group of primitives G_i , and $L(G_i)$ refers to the class label assigned to that group. Finally we will use $\lambda_{L(G_i)}$ to refer to the stroke-level temporal model corresponding to the label $L(G_i)$, and λ_{obj} to refer to our model for object-level temporal patterns.

2.2. Problem Formulation

Given a sequence of time-ordered primitives obtained from a sketch, our goal is to find a segmentation and classification of the primitives such that the likelihood of stroke-level patterns and object-level patterns is maximized simultaneously. Over all possible ways in which the primitives can be grouped, \mathfrak{G} , and all possible ways in which these groups can be labeled, $\mathfrak{L}(G)$, we want the grouping $G \in \mathfrak{G}$ and the labeling $L \in \mathfrak{L}(G)$ that:

- maximizes the likelihood of the sequence of labels L_1, L_2, \dots, L_K given our model for object-level patterns (λ_{obj}), and
- maximizes the likelihood of the stroke-level observable features O_{G_i} for each group G_i given the stroke-level model corresponding to its label ($\lambda_{L(G_i)}$).

¹We will use the Matlab notation *begin : end* as a shorthand for a list of indices that begins with *start* and ends with *end* (i.e., 1:4 = 1 2 3 4). We will also use this notation in subscripts to denote a list of items (i.e., $S_{1:N} = S_1, S_2, \dots, S_N$).

²A simplifying assumption in most sketch recognition systems is that a stroke can be part of only one object. Our definition of segmentation in terms of primitive groupings is more general than a definition based on stroke groupings and allows a stroke to be part of multiple objects (e.g., drawing a box and an arrow, or a resistors and a pair of wires in a single stroke (Fig. 5)).

The terms corresponding to the stroke-level and object-level patterns can be written together to obtain the following maximization problem, the solution to which gives us the segmentation and labeling of the input sketch.

$$\operatorname{argmax}_{G \in \mathcal{G}, L \in \mathcal{L}(G)} P(L_1, \dots, L_K | \lambda_{obj}) \prod_{i=1}^K P(O_{G_i} | \lambda_{L(G_i)}) \quad (1)$$

3. APPROACH

The formula above is maximized over the set of all groupings and their labelings (\mathcal{G} and $\mathcal{L}(G)$). As it is well known from the vision and sketch recognition literature, exhaustive search of this space is computationally prohibitive (Mahoney & Fromherz, 2002).

Noting that we are modeling sequential patterns, we make the assumption that in our target domains, both stroke-level and object-level patterns can be modeled as products of first order Markov processes. This will allow us to efficiently compute maximum likelihood estimates to learn the parameters of our stroke-level and object-level models, and will also enable efficient recognition.

3.1. The Model

We will represent our model of temporal patterns using Dynamic Bayesian Networks (DBNs). Bayesian networks encode the joint probability of a set of variables $Z = \{Z_1, \dots, Z_n\}$ where the graphical structure of the network encodes the conditional dependencies among the variables. DBNs are extensions of Bayesian networks that model joint distribution of a set of variables over time by representing the conditional dependencies between the variables using a pair of Bayesian networks (B_1, B_{\rightarrow}). B_1 defines the prior for the Z_i values at time $t = 1$, and B_{\rightarrow} defines how variables at time $t + 1$ relate to each other and to those from time t . The rest of our discussion assumes that the reader is familiar with DBNs (an excellent review can be found in (Murphy, 2002)).

3.1.1. THE INPUT

The input to our model is the observation sequence $\mathcal{O} = O_{1:T}$ obtained by computing features from corresponding primitives $\mathcal{P} = P_{1:T}$. The specific features we use are described below.

Working with primitives (rather than strokes) allows us to handle multi-object strokes which is an essential feature in certain domains (Alvarado & Davis, 2004), but is sometimes unsupported for efficiency reasons or by architectural limitations. By breaking strokes into primitives and having a model that allows primitives obtained from a single stroke

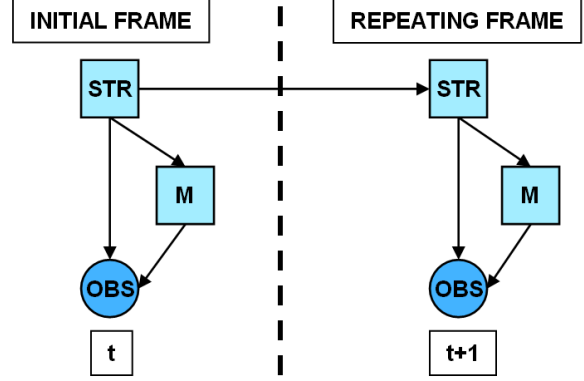


Figure 2. The dynamic Bayesian network representing the model for capturing stroke level patterns. This fragment has a dual representation as a hidden Markov model with continuous observations and $|\mathbf{M}|$ mixtures.

to be assigned to different objects, we support multiple objects to share a stroke (but not a primitive).

In sketch recognition, we deal with data that is most naturally described using geometric features such as the shape of a stroke segment, length and orientation of line segments, radii of circles etc. Some of these features are categorical (e.g., shape of a stroke segment can be arc, line etc.) and are best represented using discrete variables. Other features such as length and orientation are real-valued quantities and should be represented as such. Therefore we support both discrete and real valued (continuous) observations. In addition to allowing us to have a richer set of features, supporting real-valued features make it possible to avoid discretization issues such as choosing the optimal discretization parameters (e.g., number of bins, bin sizes).

3.1.2. THE STROKE-LEVEL MODEL

For the sake of clarity of presentation, we present our model bottom up, starting with the stroke-level model. For each object class to be recognized, we need a stroke-level model. The purpose of the stroke-level model is to compute the degree of agreement between a particular observation sequence and those sequences typically observed while drawing a particular class of objects. The stroke-level models answer questions of the sort “what is the likelihood of seeing the observation sequence O_{G_i} obtained from a group of primitives G_i under the rectangle model”, or more formally what is $P(O_{G_i} | \lambda_{rectangle})$. This corresponds to the innermost likelihood term $P(O_{G_i} | \lambda_{L(G_i)})$ in expression 1.

Fig. 2 shows the DBN fragment corresponding to our stroke-level model. The observable node **OBS** at time t represents the features extracted from primitive P_t . As noted

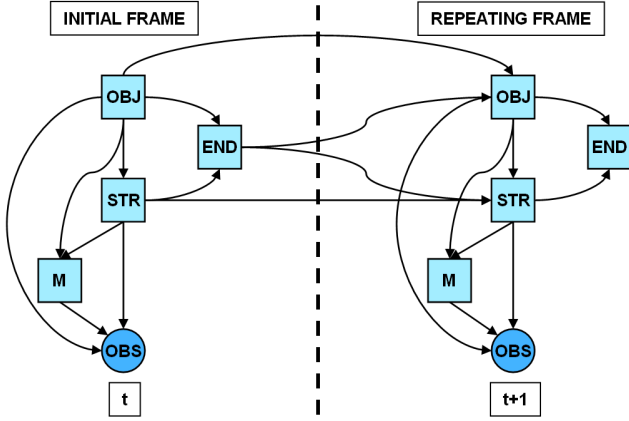


Figure 3. The dynamic Bayesian network representing our model.

before, we encode the stroke-level patterns using a first order Markov process. The discrete node **STR** in Fig. 2 captures the dynamics of this Markov process. During training, parameters of the **STR** node are estimated based on the patterns that exist in the training data provided through the **OBS** variable. The **M** variable is a discrete mixture variable and allows us to represent the observations using mixtures of Gaussians. Each **STR** node is connected to the one in the next time slice, reflecting our choice of modeling stroke-level patterns as products of a Markov process.

3.1.3. THE COMBINED MODEL

We next augment the stroke-level model by adding two nodes **OBJ**_t and **END**_t (Fig. 3).³ **OBJ**_t is a multi-valued discrete variable that holds our hypothesis of which object the primitive P_t is a part of. It can take as many values as the number of classes we can recognize. **END**_t reflects our belief about whether the user has just completed drawing object **OBJ**_t by drawing the primitive P_t .

The combined model defines a joint distribution over the variables from the stroke-level model, the sequence of object hypotheses **OBJ**_{1:T}, as well as hypotheses on where each object begins and ends (**END**_{1:T}). The joint probability over **OBJ**_{1:T} corresponds to $P(L_1, \dots, L_K | \lambda_{obj})$, the first term in expression 1. Combined with the stroke-level likelihood term $P(O_{G_i} | \lambda_{L(G_i)})$ from above, the combined model allows us to compute the most probable values of **OBJ**_{1:T} and **END**_{1:T}, which gives us the optimal segmentation and labeling of the input sketch.

The value of the **OBJ** node in each time slice is conditioned on the values of **END** and **OBJ** variables from the previous

³Our combined DBN model has a dual representation as a hierarchical HMM with continuous observables modeled using mixtures of Gaussians. We choose to present the DBN view because DBNs generalize HHMMs and the DBN representation is more efficient.

slice. This encodes the observation that our belief about the value of **OBJ**_{t+1} is based on the values of **OBJ**_t and **END**_t. The justification for this dependence is that we would expect the value of the **OBJ** node to change only if the user has just finished an object (i.e., **END**_t=true), in which case the next object we would expect to see would depend on what object was just completed. Also note that the two new inter-slice arcs introduced into our model **OBJ**_t→**OBJ**_{t+1} and **END**_t→**OBJ**_{t+1}, cross only a single slice boundary, thus our model remains first order Markovian, enabling training and recognition to be efficient.

In summary, the **OBJ** node keeps track of the class for the objects drawn over time and the joint distribution of the **OBJ** nodes over time gives us $P(L_1, \dots, L_K | \lambda_{obj})$, the first term in expression 1. In the combined model, the distribution of the observations is determined based on the value of the **OBJ** node given by $P(OBS_t | OBJ_t, STR_t, M_t)$. The choice of which stroke-level process is activated is determined by the **OBJ** node because **STR**_{t+1} is conditioned on **STR**_t, **END**_t and **OBJ**_{t+1}.

3.2. Implementation issues

Two issues need to be addressed in implementing the model described above, both related to the choice of the inference algorithm to be used.

As mentioned earlier, our model has a dual representation as a hierarchical hidden Markov model. Unfortunately, the original inference algorithm for HHMMs is not only complicated, but also has $O(T^3)$ time complexity (where T is the length of the observation sequence) (Murphy & Paskin, 2001). In our case T is the total number of primitives in a sketch, and this makes the model impractical for situations where real-time feedback is required. On the other hand, by using the DBN representation we can apply efficient graphical model techniques that take linear time (Murphy & Paskin, 2001). It is therefore essential to use the DBN representation, or convert any HHMM based representation to a DBN prior to inference and learning.

A major issue that arose during the implementation is the numerical instabilities that occur during training, due to the use of continuous observations. Bayesian networks that include continuous and discrete variables (mixed networks) usually represent the continuous variables as Gaussians or mixtures of Gaussians. The conventional belief propagation algorithm used for inference in these networks is the Lauritzen algorithm. Unfortunately this algorithm is susceptible to numerical underflow. Although this has been documented in the machine learning literature, it is not well known because it occurs rarely in reality. The problem appears in the form of singular matrices during inference and is hard to localize. In order to avoid this numerical instability, we need to use a specialized algorithm known as the

Lauritzen-Jensen stable conditional Gaussian belief propagation (Lauritzen & Jensen, 2001).

3.2.1. TRAINING

The goal of training is to estimate the parameters of the conditional probability distribution functions for each node in our DBN (B_1, B_{\rightarrow}) given a collection of labeled sketches as training data. We use parameter tying to ensure that the probability distribution $P(A|Parents(A))$ relating a node A and its parents is the same for each node A in B_1 and B_{\rightarrow} .⁴

During training we know what class each primitive belongs to and we also know when objects begin and end, so the values of the **OBJ** and **END** are observable and are supplied during training. In summary, for each labeled sketch, we use the values **OBJ**_{1:T}, **END**_{1:T} and **OBS**_{1:T} to estimate the model parameters.

3.2.2. RECOGNITION

The input to recognition is **OBS**_{1:T}. During recognition, the goal is to compute the values of **OBJ**_{1:T} and **END**_{1:T} that maximize the likelihood of the observations **OBS**_{1:T}. This is done using the stable conditional Gaussian belief propagation algorithm described earlier. Interpretation of complete sketches comparable to those in Fig.4 takes less than 2-3 seconds on a 2GHZ Pentium 4 machine.

4. RESULTS

In order to see whether modeling object-level patterns improves over modeling only stroke-level patterns, we tested our system using sketches from the domain of electronic circuit diagrams. This domain is particularly interesting because it is illustrative of a group of domains that can be characterized as *object/connector diagrams* (e.g., organizational charts, flow charts, UML diagrams). This domain also illustrates our model’s ability to deal with multi-object strokes (e.g., *stroke#7* in Fig. 5) and objects with varying number of components (e.g., resistors with varying number of humps).

4.1. Data collection

We collected circuit diagrams from undergraduate students who had recently taken the Microelectronic Circuits and Devices course at MIT and were familiar with the textbook used in the class. We asked the participants to draw circuits selected from their course textbook using a sufficiently ex-

⁴We add dummy parent nodes to the **OBJ** and **STR** nodes in B_1 to ensure that $P(\mathbf{OBJ}|Parents(\mathbf{OBJ}))$ and $P(\mathbf{STR}|Parents(\mathbf{STR}))$ can be represented using conditional probability tables with the same structure for B_1 and B_{\rightarrow} .

pressive set of electronic circuit components.⁵ A total of eight participants contributed circuit diagrams.

During data collection, we first showed participants diagrams of circuits that we wanted them to draw. We gave each participant some time to study each circuit until they understood how the circuit worked, then asked them to explain it verbally, to ensure that they really understood it. We next removed the textbook diagram and asked them to draw the circuit using a Tablet PC, allowing them to consult the original circuit diagram only if needed. Fig. 4 shows examples of circuits drawn by the participants.

To train our model we created labeled training data by manually annotating 10 sketches per participant.

4.2. Generating the Observation Sequence

Both training and classification require converting a sketch to an observation sequence $O_{1:T}$. Using the early sketch processing toolkit described in (Sezgin et al., 2001), we first preprocess each sketch to obtain a sequence of primitives $P_{1:T}$ (in our case simply using line segments is sufficient because our domain has no objects with curved strokes). For each primitive P_i , we obtain an observation vector O_t represented as a five-tuple $(l_t, \Delta l_t, \theta_t, \Delta \theta_t, sgn_t)$ where:

- l_t is the length of P_i ,
- Δl_t is relative length $(l_t/l_{t-1}, 1 \text{ for } t = 1)$,
- θ_t is the angle with respect to the horizontal axis,
- $\Delta \theta_t$ is the measure of relative angle between P_i and P_{i-1} ,
- sgn_t is the direction that the stroke turns when moving from P_{i-1} to P_i .

The value of $\Delta \theta_t$ is given by the magnitude of the cross product $\vec{u} \times \vec{v}$ of vectors \vec{u}, \vec{v} , which are length-normalized versions of P_i and P_{i-1} pointing in the direction of pen movement along each primitive. The turn direction (sgn_t) is the only discrete feature and is set to 0 for negative values of $\vec{u} \times \vec{v}$ and 1 for positive values or $t = 1$.

4.3. Recognition Results

For each user, we ran a series of hold-one-out experiments by getting the recognition rates for each sketch using a model trained on the remaining sketches. For each sketch, we also trained a baseline system that modeled only stroke-level patterns (i.e., no object-level pat-

⁵The components were NPN transistors, resistors, capacitors, batteries and wires, so $|\mathbf{OBJ}| = 5$.

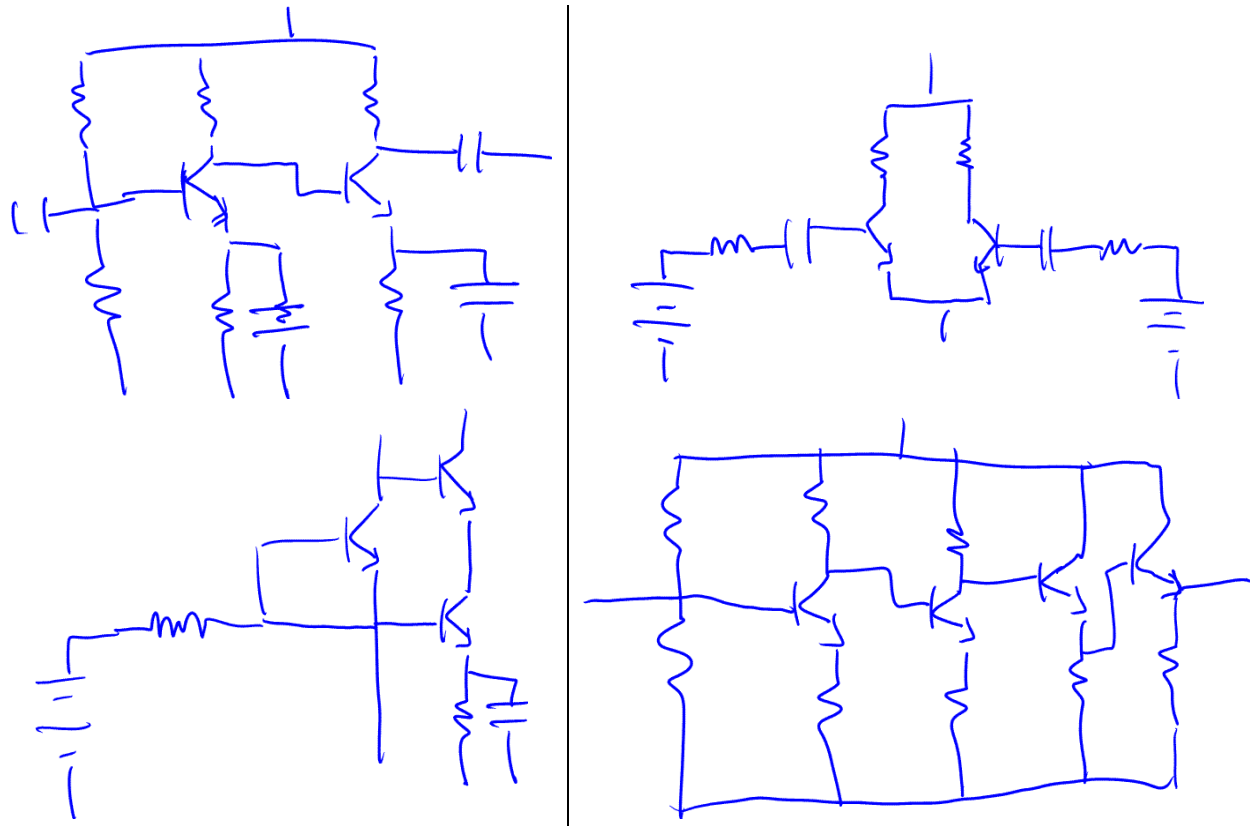


Figure 4. Examples of collected circuits.

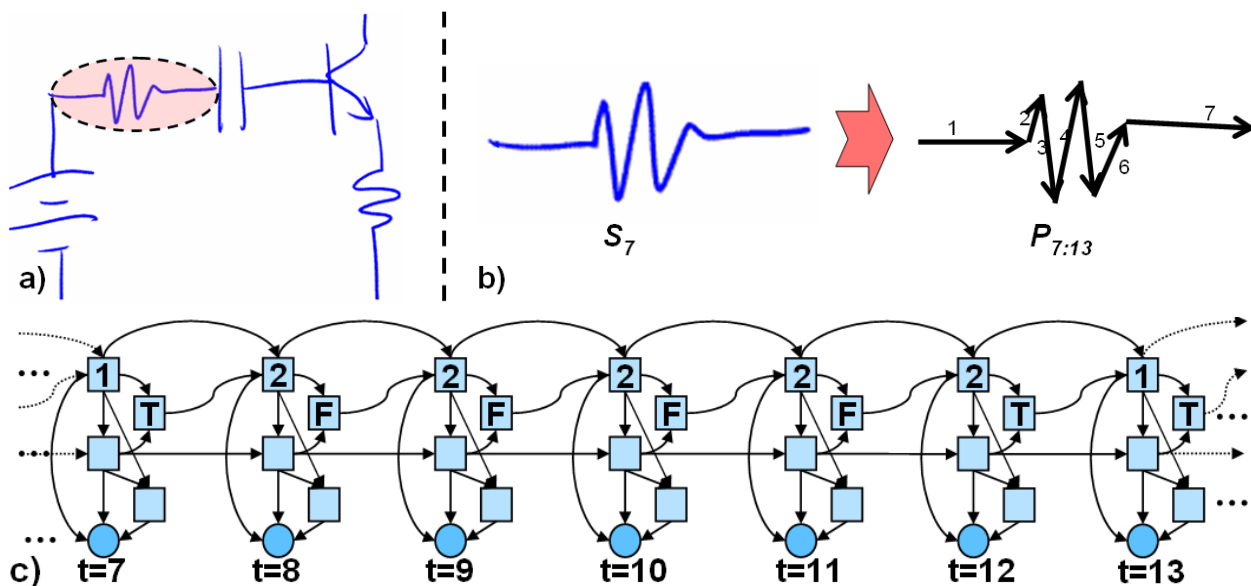


Figure 5. An example showing how a Bayesian network is created by unfolding (B_1, B_-) during recognition. In 5(a), we see a circuit fragment. During recognition, each stroke is broken into primitives. 5(b) shows how primitive extraction breaks *stroke#7*, which also happens to be a multi-object stroke, into primitives $P_{7:13}$. In 5(c) we show the corresponding fragment of the unfolded network. Expected values of $\mathbf{OBJ}_{7:13}$ and $\mathbf{END}_{7:13}$ are also shown. For $\mathbf{OBJ}_{7:13}$, the values $\{1, 2\}$ represent “wire” and “resistor”, for $\mathbf{END}_{7:13}$ $\{T, F\}$ represent *true* and *false*. Other details are omitted for brevity.

terns). The baseline system has the same computational complexity as our method and was obtained by setting $P(\mathbf{OBJ}_{t+1}|\mathbf{OBJ}_t, \mathbf{END}_t)$ to be uniform ($1/|\mathbf{OBJ}|$) if $\mathbf{END}_t = \text{true}$. For $\mathbf{END}_t = \text{false}$, it was set to 1 for $\mathbf{OBJ}_{t+1} = \mathbf{OBJ}_t$, and 0 otherwise. In both models we used the values $|\mathbf{SRC}| = 6$ and $|\mathbf{M}| = 3$.

4.3.1. QUANTITATIVE RESULTS

Table 4.3.2 shows the average correct recognition rates for each participant obtained using the baseline model and using our model. As seen in the table, the improvement in absolute terms ranges from 2.1% to 6.2%. The table also shows the average and maximum reduction values in the error rates in terms of percentages for each user. As seen here, on average, modeling object-level patterns always improves performance, allowing up to 65% of misrecognition errors to be corrected.

A paired t-test comparing the average correct recognition rates for each user finds the increase to be statistically significant for $p < 0.05$ and 7 degrees of freedom. We believe the reported decrease in the error rates would substantially improve users' satisfaction with the recognition system in an actual design setting.

4.3.2. QUALITATIVE RESULTS

It is instructive to explore what sort of object-level patterns were learned by our model. This can be done by inspecting the conditional probability table $P(\mathbf{OBJ}_{t+1}|\mathbf{OBJ}_t, \mathbf{END}_t)$. For the 8 participants in our study, all the models learned (the unsurprising fact) that most of the time circuit components are preceded and followed by wires.

A more interesting observation comes from looking at what happens when an object is not followed by a wire. In these cases, we see that components that frequently appear together in conceptually meaningful circuit fragments have a higher probability of being drawn consecutively. For example, resistors appear in series when drawing voltage dividers, hence when a resistor is not immediately followed by a wire temporally, it is more likely to be followed by a resistor than by anything else. It is conceivable that a circuit design expert may predict the existence of such patterns and try to incorporate them in the design of a circuit recognition system, but this would be a laborious task and require precision in balancing biases for different kinds of patterns. Our model learns and uses such patterns automatically in a mathematically sound framework.

5. RELATED WORK

There is a large body of work in sketch recognition that does not use temporal features (e.g., (Alvarado & Davis, 2004; Shilman et al., 2002; Mahoney & Fromherz, 2002;

	Participant ID							
	1	2	3	4	5	6	7	8
μ_b	83.2	86.5	85.2	73.8	87.1	90.7	79.1	85.0
μ_c	89.4	89.4	87.3	77.4	89.8	93.0	84.6	88.2
Δ_{err}	36.9	21.4	14.1	13.7	20.9	24.7	18.6	21.3
$max\Delta$	65.0	45	31.2	15.0	35.5	40.0	30.7	54.5

Table 1. Mean correct recognition rates for the baseline system (μ_b) and our model (μ_c) on sketches collected from 8 participants. The table also shows the percentage reductions in the error rates and maximum error reductions achieved for each user as percentages (Δ_{err} and $max\Delta$). On average, modeling object-level patterns always improves performance.

Gennari et al., 2004)). As this approach is complementary to ours, we limit our discussion here to systems that do use temporal features.

The handwriting recognition community has come up with a large number of methods for recognizing single stroke or segmented pen input using HMMs. These systems take advantage of a vast number of robust preprocessing segmentation methods to simplify the problem to an isolated recognition problem.

Several systems have used HMM or HHMM based frameworks to recognize pen input. In the sketch recognition community, work in (Anderson et al., 2004) describes an HMM-based symbol recognizer that uses chain-code-like features to recognize isolated symbols. Our work does not assume segmented input and builds a joint model for complete sketches.

Our previous (Sezgin & Davis, 2005) describes an HMM based sketch recognition system that does segmentation and recognition by combining outputs of individual HMMs using dynamic programming (DP). This method is a weaker version of our baseline method in that it does not support continuous features. Its two layer HMM/DP recognition method suffers from the $O(T^3)$ complexity, just like HHMMs.

In (Simhon & Dudek, 2004), the authors present a sketch interpretation and curve refinement system that uses an HHMM. This work shares the same motivation as ours. In their work, they assume that the parameters of the object-level process (which they refer to as the scene level) is supplied by the user using a semantic graph representation. We learn the parameters of the stroke-level and object-level patterns from data, and use the more efficient DBN representation to avoid the $O(T^3)$ complexity of the HHMMs. We also support multi-stroke objects and our use of real-valued continuous observations allow using a rich set of features.

6. CONTRIBUTIONS

Until very recently, temporal models were considered to be unsuitable for sketch recognition (Alvarado & Davis, 2004). We have presented a sketch recognition framework based on dynamic Bayesian networks that demonstrates we can recognize sketches by exploiting both stroke-level and object-level temporal orderings. Our framework allows learning stroke-level and object-level patterns from data, handles multi-stroke objects and multi-object strokes efficiently, and supports continuous observable features. We showed how graphical models can be used to implement our model efficiently, and showed that a specialized inference algorithm known as the Lauritzen-Jensen stable conditional Gaussian belief propagation should be used to avoid numerical instabilities in recognition. We have demonstrated that modeling object-level patterns improves classification performance for the electronic circuit diagrams domain. Finally we presented a framework that allows joint parameter estimation for the stroke-level and the object-level processes. This avoids the task of training the models separately and then stitching them together, as it is usually done in HHMM-based frameworks.

7. FUTURE DIRECTIONS

One limitation of our model is its assumption that the user completes drawing one object before moving on to the next. This is not always true of course. In the electronic circuit domain, for example, this assumption is sometimes violated when drawing transistors, primarily because transistors have three points where wires can connect, and sketchers at times intersperse drawing wires with drawing the transistor. We are developing ways of modifying our model to enable it to handle interspersed drawing. Early results that we have obtained that it is possible to extend our model to handle interspersing yet maintain its character as a Markovian process (and hence maintain the ability to do recognition efficiently) (Sezgin, 2006).

We are also working on ways of incorporating classification hypotheses supplied by external recognizers that use spatial or structural information into our model using a mechanism known as virtual evidence nodes in the graphical models machine learning community. This would be a first step toward combining temporal and spatial/structural information for sketch recognition.

8. ACKNOWLEDGEMENTS

This work was supported in part by the MIT/Microsoft iCampus Project.

References

- Alvarado, C., & Davis, R. (2004). Sketchread: A multi-domain sketch recognition engine. *Proceedings of UIST 2004*.
- Anderson, D., Bailey, C., & Skubic, M. (2004). Hidden markov model symbol recognition for sketch-based interfaces. *AAAI Fall Symposium Series Making Pen-Based Interaction Intelligent and Natural*.
- Gennari, L., Kara, L. B., & Stahovich, T. F. (2004). Combining geometry and domain knowledge to interpret hand-drawn diagrams. *AAAI Fall Symposium Series 2004*.
- Lauritzen, S., & Jensen, F. (2001). Stable local computation with conditional gaussian distributions. *Statistics and Computing*, 11:191–203.
- Mahoney, J. V., & Fromherz, M. P. J. (2002). Three main concerns in sketch recognition and an approach to addressing them. *AAAI Spring Symposium: Sketch Understanding 2002*.
- Murphy, K. (2002). Dynamic bayesian networks. *Chapter in Probabilistic Graphical Models by Michael Jordan*.
- Murphy, K., & Paskin, M. (2001). Linear time inference in hierarchical HMMs. *NIPS-01*.
- Sezgin, T. M. (2006). Online sketch recognition from a dynamic perspective. *PhD Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, June 2006*.
- Sezgin, T. M., & Davis, R. (2005). HMM-based efficient sketch recognition. *Proceedings of IUI-2005*.
- Sezgin, T. M., Stahovich, T., & Davis, R. (2001). Sketch based interfaces: Early processing for sketch understanding. *Proceedings of PUI-2001*.
- Shilman, M., Pasula, H., Russel, S., & Newton, R. (2002). Statistical visual language models for ink parsing. *AAAI Spring Symposium: Sketch Understanding, March 25-7, Stanford CA 2002*.
- Simhon, S., & Dudek, G. (2004). Sketch interpretation and refinement using statistical models. *Eurographics 2004*.