Modeling Sketching as a Dynamic Process

Tevfik Metin Sezgin MTSEZGIN@CSAIL.MIT.EDU
Randall Davis DAVIS@CSAIL.MIT.EDU
MIT Computer Science and Artificial Intelligence Laboratory, 32-235 Vassar st., Cambridge MA, 02139 USA

Abstract

Online sketching is an incremental and dynamic process; sketches are is drawn one stroke at a time and can be captured with devices such as Tablet PCs and pen based PDAs. We have shown that the dynamic properties of the sketching process contain valuable information that can aid recognition. We describe a framework that can handle complex user input. Specifically, we show how we can take advantage of the regularities in sketching even when users draw objects in an interspersed fashion.

1. Introduction

Online sketching is an incremental and dynamic process: sketches are drawn one stroke at a time and be captured in devices such as Tablet PCs and pen based PDAs. This is unlike scanned documents or pictures which only capture the finished product. The dynamic properties of the sketching process contain valuable information that can aid recognition (Sezgin & Davis, 2005). In particular, in a number of domains the order in which users lay out strokes during sketching contains patterns and is predictable. We have presented ways of taking advantage of these regularities to formulate sketch recognition strategies (Sezgin & Davis, 2005). Here, we describe a framework that can handle more complex user input. Specifically, we show how we can take advantage of the regularities in sketching even when users draw objects in an interspersed fashion (e.g., start drawing object A, draw B before fully completing A, come back and complete drawing A).

2. Sketching as a stochastic process

Previous work has shown that in certain domains stroke ordering follows predictable patterns and can be modeled as a Markovian stochastic process. Work in (Sezgin & Davis, 2005) shows how sketches of mechanical engineering drawings, course of action diagrams, emoticons and scenes with stick-figure can be modelled and recognized using Hidden Markov Models. In these domains, HMM-based modeling and recognition is possible because objects are usually drawn one after the other using consistent drawing orders. The HMM-based approach exploits these regularities to perform very efficient segmentation and recognition.

The HMM-based recognition algorithm scales linearly with the scene size, but requires each object to be completed before the next one is drawn. In certain domains, although there is a preferred stroke ordering, objects can be drawn in an interspersed fashion. For example, in the domain of circuit diagrams, people occasionally stop to draw wires connecting to the pins of a transistor before they complete the transistor. One way of thinking about such a drawing scenario is that, instead of a single Markov process, we have multiple processes that generate observations, and the task is to separate observations from these processes. [1] We model such drawing behavior as a multimodal stochastic process that can switch between different individual Markov processes, each of which captures drawing orders for individual objects. Although the new approach can also be described as a HMM, it is more easily described and understood using its dual representation as a dynamic Bayesian net (DBN).

Our approach to modeling interspersed drawing behavior is general enough to allow an arbitrary number of objects in a domain to be drawn in an interspersed fashion, but in practice people usually intersperse at most two objects. For example, in the circuit diagrams, unlike other circuit components, transistors have three connection points (emmiter, collector, base) sometimes people draw the wires connecting to these points when the transistor is only partially drawn, causing interspersing of transistor and wire strokes. We have created a model specialized to handle interspersing of wires with other components in circuit diagram sketches. [2]

[1]This is similar to the data association problem problem in signal processing and computer vision except at each time slice, the observation comes from a one of multiple potential models.

[2]Although it is also possible to have a model general enough to allow interspersing between any two objects, we use this spe-

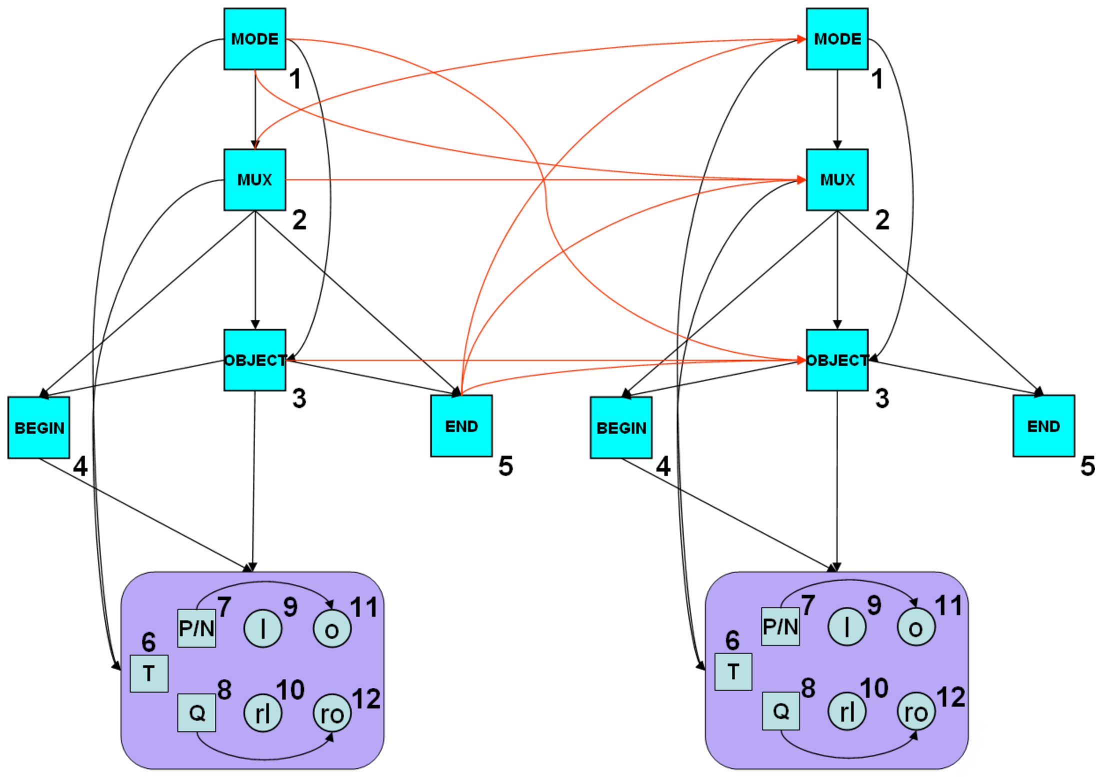

Figure 1. The network structure for two slices of the DBN for modeling electronic circuits.

3. The network structure

Next we introduce our DBN model for circuit diagrams which handles interspersed drawing orders while still allowing polynomial time inference in the number of strokes.

We model the circuit diagram sketching process using a DBN (Fig. 1). The square nodes are discrete and the circular nodes are continuous. Nodes 1-6 (dark blue) are hidden and 6-12 (light blue) are observed. The rounded rectangle groups the observable nodes, and is used primarily to indicate that the incoming arrows connect to all the nodes inside the rounded rectangle. Here is a brief description of each node in Fig. 1

- **MODE** is a binary variable that monitors whether the user is interspersing any object with wires.
- The **MUX** variable is a multi-valued discrete node that chooses one of the object processes.
- The **OBJECT** variable captures the drawing order for individual circuit symbols.
- **BEGIN** is a binary variable and is set to true if the user has started drawing a new object.
- **END** is a binary variable and is set to true if the user has finished drawing a new object.
- The remaining nodes are the observable variables of the model. They encode the observed ink in terms of primitive class of the ink (e.g., line, circle), and geometric features such as length, orientation, relative length and relative orientation.

cialized model due to the nature of interspersing in our domain.

3.1 The MODE variable

MODE is set to `off` when there is no interspersing of wires with other primitives. The node **MODE** at time slice $t+1$ ($\mathbf{MODE_{t+1}}$) is conditioned on the nodes $\mathbf{MUX_t}$ and $\mathbf{END_t}$. These arcs ensure that for objects that don't get interspersed with wires, $\mathbf{MODE_{t+1}}$ doesn't switch to the `on` mode unless the user has just finished drawing an object. **MODE** also controls the state transition behavior of the object HMMs through the arc to **OBJECT**. This ensures that the object HMMs preserve their current state when the user draws a wire.

3.2 The MUX variable

MUX keeps track of the main object the user is drawing (which can be interspersed with wires if it is a transistor). The actual observables (nodes [6:12]) are controlled by the value of this node as well as other nodes such as MODE, BEGIN and OBJECT. If there are N different objects that the user can draw (not including the wires), then MUX has N states. MUX$_{t+1}$ is conditioned on MODE$_t$, MUX$_t$ and END$_t$. The reasoning behind this conditioning is threefold: the user may start drawing a new object (other than a wire) only if the previous object is completed; the probability of drawing a particular class of object may depend on the type of the last object; and finally if MODE$_t$ is `on`, OBJECT$_{t+1}$ should maintain its state from the previous time slice.

3.3 The BEGIN and END variables

These discrete binary nodes are both conditioned on the OBJECT variable and the individual object processes (i.e., NPN, LAMP). When the user starts or ends an object, these nodes are set on and off accordingly. For example, the BEGIN node is set to `true` if the current object process selected by OBJECT is producing the first observation of a new object.

3.4 The OBJECT variable

OBJECT captures the drawing order of individual objects. MUX and OBJECT define the set of HMMs that capture the drawing order of individual objects. In addition to the Markovian dependency on MODE$_t$, OBJECT$_t$ and END$_t$, OBJECTt+1 is also conditioned on the current MODE and MUX variables.

3.5 The observable variables

The observable nodes of our model are in the rounded rectangle in Fig. 1 (T, P/N, Q, L, RL, O and RO). The discrete binary variable T encodes the type of the current primitive (circle, line). L and RL are Gaussian nodes that capture length and relative length. O and RO capture orientation of the current primitive and the relative orientation with respect to the previous primitive. O and RO are continuous variables modelled as mixtures of Gaussian, while P/N and Q are the mixture variables that respectively model positive/negative slope for O and the quadrant of the current primitive with respect to the previous primitive for RO.

The relative orientation and relative length features compute the change in these values for current and previous primitives. We expect these values to be reliable features for time slices that do not correspond to the beginning of a new object. This is because, for time slices that correspond to object beginnings, the previous primitive belongs to another object, hence the conditioning on the BEGIN.

4. Progress and implementation issues

We have collected circuit diagrams from electrical engineers. We annotated the data and are currently working on an initial implementation of the above model. The MODE and SRC state pair act as switching parent nodes for the OBJECT node. This poses some numerical problems in the standard junction tree algorithm used for inference. We are working on ways of avoiding this problem by using numerically stable learning and inference algorithms.

Research Support

This research is funded by the MIT iCampus project and Project Oxygen.

References

Sezgin, T. M., & Davis, R. (2005). HMM-based efficient sketch recognition. *International Conference on Intelligent User Interfaces, San Diego CA January 2005.*