Sketch Recognition for Digital Circuit Design in the Classroom

Christine Alvarado
Harvey Mudd College
alvarado@cs.hmc.edu

1 Research Overview

Digital circuit simulation programs are powerful in education because, unlike paper, they allow students to explore and verify the behavior of a physical system. Unfortunately, learning to navigate the interfaces to these tools can be time consuming and useless from a pedagogical perspective. Our central aim is to build a tool that allows students to sketch circuit schematics on a Tablet PC and then simulate these circuits directly. Our hypothesis is that such a recognition tool will enable students to spend less time fighting with the interface and more time learning the material. We will test this hypothesis by completing a version of this tool for use in Harvey Mudd College's digital circuit design class, E85, starting in the fall of 2007 [1].

Our primary challenge is to construct a free-sketch recognition system that places few constraints on users' drawing styles and performs robustly enough to be used in the classroom. *Free-sketch recognition*—recognition without placing any constraints on the user's drawing style—is difficult because the symbols in a sketch may vary in position, rotation, and scale, and because sketches are messy. Free-sketch recognition is complicated by the added challenge of grouping strokes into individual objects. Simple temporal and spatial grouping techniques are not robust because symbols may overlap and because the user may draw two symbols in parallel. On the other hand, naively matching all templates to all parts of the user's sketch is computationally intractable.

To address these challenges we adapt, extend and combine a number of state-of-the-art approaches to solving more constrained sketch recognition problems. Our approach first labels individual strokes as wires or gates using a conditional random field (CRF) and then groups labeled strokes into individual shapes. Because strokes with the same label belonging to different symbols are separated in time and space, boundaries between individual symbols are easier to detect than in the unlabeled sketch. These grouped strokes will then be recognized using a graph-based isolated-shape recognition algorithm. Finally, the results of the recognition process will feed back into the system to help it correct stroke grouping errors.

Existing recognition technologies, including Tablet PC gesture recognition, either place significant constraints on users' drawing style (e.g., forcing the user to group strokes into symbols manually) [2, 5], recognize only a limited set of symbols [6, 8], or have not been shown to perform sufficiently robustly with end-users [1, 3]. A few systems, such as $\mathrm{MathPad}^2$ [4], have achieved real-world use for simple domains. Our goal is to achieve similar success in a more complex domain.

1.1 Stroke Classification

Unlike most existing systems that assume stroke grouping will occur before symbol recognition, our approach uses the results of single-stroke classification to inform the process of stroke grouping. Szummer and Qi [8] illustrated how conditional random fields effectively combine stroke and contextual information for single-stroke classification in organizational chart diagrams. We extend their approach to the more complex domain of circuit diagrams.

Briefly, a CRF is an undirected graphical model that represents the conditional probability distribution $P(\boldsymbol{y}|\boldsymbol{x})$ where $\boldsymbol{x}$ is a set of input data and $\boldsymbol{y}$ is a set of labels for that input data. The actual CRF consists of a graph $G = (V, E)$ and an associated set of potential functions that together define $P(\boldsymbol{y}|\boldsymbol{x})$. Each node in V corresponds to an element of the input, and each edge in E quantifies a probabilistic dependence between nodes.

As in [8], we first fragment the strokes in the sketch by finding corners using the algorithm presented in [7]. We then automatically construct the graph by creating a node for every fragment and linking nodes for fragments that are spatially or temporally proximal. We consider two types of potential functions: site potentials that measure the compatibility between a stroke and its associated label, and pairwise interaction potentials that measure the compatibility between neighboring labels. Both types of potentials measure compatibility by linearly combining parameters with a set of feature functions and passing the

[1] After successfully using the tool at HMC, we plan to make it available for others teaching similar courses.

result through a non-linearity (we use the exponential).

Unlike in [8], our graphs are dense because of our relatively complex domain. We must rely on approximate inference methods for both learning and classification; we use Loopy Belief Propagation. Loopy BP, while generally successful, is somewhat sensitive to initial conditions. To address this problem we precondition by training first on smaller sets of data to get parameters that are close to the optimal parameters.

1.2 Stroke Grouping

After labeling each stroke, we group strokes into individual objects. Even with perfect labels, stroke grouping is not trivial. For example, different wires may overlap in space and the same wire may be separated in time.

We use a graph theoretic method for stroke grouping that treats each labeled stroke as a node in a graph, with edges between adjacent strokes. The algorithm then finds the connected components in the graph.

Two strokes are adjacent if their minimum distance is lower than a given threshold. We designed specific distance metrics for the digital circuit domain. Given two strokes, if neither stroke is a wire, then the minium distance between the strokes is the minimum distance between any two points in the strokes. If either stroke is a wire, the minimum distance between the strokes is the distance from an endpoint to any other point on the other stroke. We use this modified distance because wires frequently overlap even when they are not meant to represent the same component. In both cases, the minimum distance is normalized by the sum of the diagonals of the smallest bounding box around each of the strokes. This normalization provides a unitless measure that is invariant under uniform scaling.

2 Demonstration

Our system is unique in that it recognizes freely-drawn, complete sketches, performing fragmentation, grouping and (eventually) symbol recognition. Our demonstration will illustrate each stage of this process.

We will recognize both real-world sketches collected from students in E85 in spring and fall 2006 and sketches produced by workshop participants. Given a raw sketch (Figure 1(a)), we fragment the individual strokes in the sketch at their corners (Figure 1(b)). Next, we classify individual strokes (Figure 1(c)), color coding them according to their label. Additionally, mousing over the strokes in the drawing shows the stroke's id, the system's interpretation of the stroke, and the system's belief in that interpretation. Finally, we group strokes into individual components, either connected wires or single gates (Figure 1(d)). Although not yet complete, the next step in this process will be to classify each gate according to its specific type, and to detect which wires connect to which gates in order to complete the circuit representation.

We will also demonstrate how our tool can be used to manually fragment, group and label strokes in a sketch, a crucial part of any sketch recognition effort.

3 Progress and Remaining Research Issues

We have some promising initial results, and we are actively working to complete our recognition tool chain, develop a user interface for the tool, and link our tool in with an existing simulator.

3.1 Initial Results

We tested our stroke labeling algorithm on a total of 51 circuit diagrams consisting of AND gates, OR gates, NOT gates, XOR gates and wires. We used 17 samples for full training and 4 or fewer samples for conditioning. We tested two-label (wire vs. gate) and 5-label classification. In the two-label case we achieved 96% accuracy, while in the 5-label case we achieved 39% accuracy.

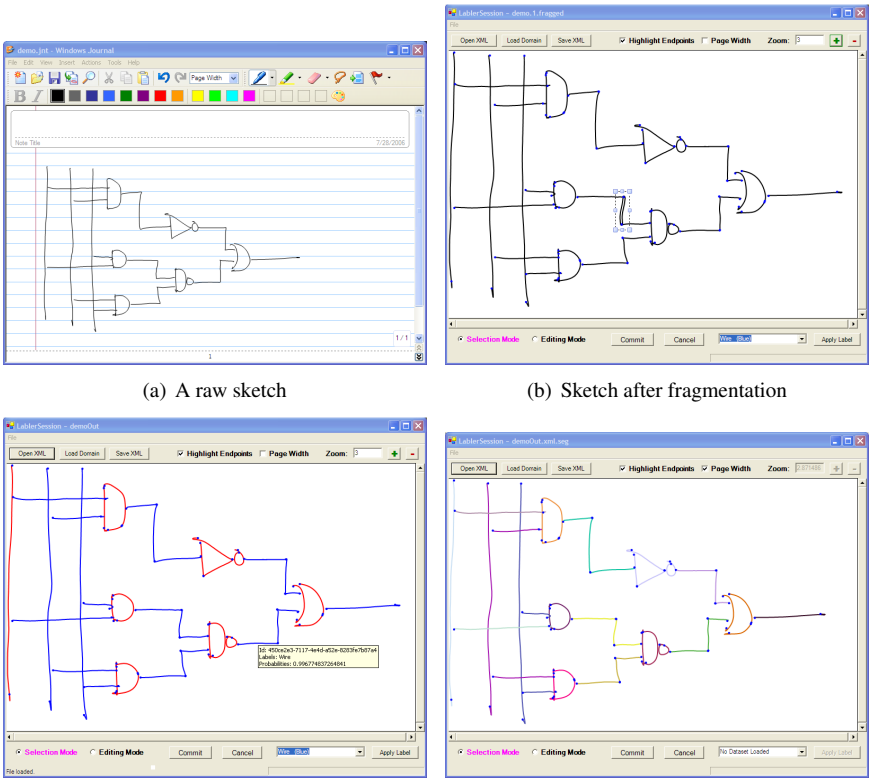
Two-label classification is likely sufficient at this stage of processing, as the specific type of gate may be determined by the symbol recognizer in later steps. However, we are working to improve multi-label classification by developing a two-pass classification scheme that first distinguishes between wires and gates and then applies a new CRF capable of distinguishing between only gates.

We tested our stroke grouping algorithms on the sketches classified by the CRF in the previous step. We achieved 77% accuracy in grouping. Below we discuss our efforts to incorporate an isolated-shape recognition algorithm capable not only of recognizing individual symbols, but also of detecting missing or superfluous strokes in the symbol. This information will be fed back into the grouping algorithm, leading to an improved sketch grouping.

3.2 Remaining Issues

First, to complete the recognition system, we must incorporate a symbol recognition algorithm that identifies groups of strokes. While many such algorithms exist, we require an algorithm that not only classifies the shape, but also detects superfluous or missing strokes to help correct for stroke grouping errors. We are currently working with Stahovich and his students to refine the symbol recognition algorithm presented in [5] for this purpose. In the process, we will develop a method for refining the initial stroke grouping produced by our algorithm.

Second, while improved recognition is essential in building sketch-based simulation tools, making full use its power in educational software will require addressing



(a) A raw sketch

(b) Sketch after fragmentation

(c) Sketch after stroke-level recognition

(d) Sketch after stroke grouping

Figure 1: Recognition Demonstration

new user interface challenges associated with this new interaction paradigm. We have designed an experiment to investigate the effect that recognition triggers, feedback mechanisms, error rates, and correction mechanisms have on user satisfaction and overall efficiency. We will carry out this study this spring, and the results of this study will inform the development of our user interface.

Finally, we plan to link our recognition interface into Xilinx, a commercial digital circuit simulator. We have already written a program to translate recognized sketches to Verilog, a format that Xilinx can import. We are currently working to make the interaction between the sketching interface and Xilinx as seamless as possible.

4 Acknowledgments

This work was done in conjunction with several HMC undergraduate students including Jason Fennell, Max Pfleuger, Devin Smith and Aaron Wolin. Paul Wais has assisted in designing the user interface experiment and will carry out this experiment in the spring.

References

[1] C. Alvarado and R. Davis. Sketchread: A multi-domain sketch recognition engine. In *Proc. UIST*, 2004.

[2] H. Hse and A. R. Newton. Sketched symbol recognition using zernike moments. In *Proc. ICPR*, 2004.

[3] L. B. Kara and T. F. Stahovich. Hierarchical parsing and recognition of hand-sketched diagrams. In *Proc. of UIST '04*, 2004.

[4] J. LaViola and R. Zeleznik. Mathpad2: A system for the creation and exploration of mathematical sketches. *ACM Trans. on Graphics (Proc. of SIGGRAPH)*, 23(3), 2004.

[5] W. Lee, L. B. Kara, and T. F. Stahovich. An efficient graph-based symbol recognizer. In *Eurographics Workshop on SBIM*, 2006.

[6] M. W. Newman, J. Lin, J. I. Hong, and J. A. Landay. DENIM: An informal Web site design tool inspired by observations of practice. *Human-Computer Interaction*, 18(3):259–324, 2003.

[7] T. M. Sezgin, T. Stahovich, and R. Davis. Sketch based interfaces: Early processing for sketch understanding. In *Proc. of PUI*, 2001.

[8] M. Szummer and Y. Qi. Contextual recognition of hand-drawn diagrams with conditional random fields. In *Proc. of IWFHR*, pages 32–37, 2004.