Visual Recognition of Sketched Symbols

Tom Y. Ouyang
MIT CSAIL
32 Vassar St, Cambridge MA, 02139, USA
ouyang@csail.mit.edu

Randall Davis
MIT CSAIL
32 Vassar St, Cambridge MA, 02139, USA
davis@csail.mit.edu

ABSTRACT
Diagrams are an essential means of capturing and communicating information in many different domains. They are also a valuable part of the early design process, helping us explore ideas and solutions in an informal environment. This paper presents a new approach to sketched symbol recognition that preserves as much of the visual nature of the symbol as possible. Our method is robust to differences in drawing style, computationally efficient, and achieves excellent performance for several different domains.

Author Keywords
Sketch Recognition, symbol recognition, vision recognition algorithms

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

INTRODUCTION
Diagrams are an essential means of capturing and communicating information in many different domains, as well as a valuable part of the early design process, helping us explore ideas and solutions in an informal environment. With the growing popularity of digital input devices like Tablet PCs and Smartboards, there is increasing interest in building systems that can automatically interpret freehand drawings. However, many challenges remain in terms of recognition accuracy, robustness to different drawing styles, and ability to generalize across multiple domains. The ideas we present here attempt to bridge part of the gap between how people naturally express diagrams and how computers interpret them today.

We will begin by looking at some of the challenges in recognizing freehand sketches. Figure 1 shows six symbols taken from a dataset of electrical circuit diagrams. As we can clearly see, these symbols exhibit a great deal of intra-class variation due to local shifts, rotations, and non-uniform scaling. In addition to these types of visible differences, two seemingly similar symbols can be drawn differently at the stroke level. For example, the strokes may differ in their order and direction and may exhibit artifacts like over-tracing (drawing over a previously drawn stroke) and pen drag (failing to lift the pen between strokes).

Besides these kinds of low-level variations, drawings can also differ at the conceptual level. This occurs when there are multiple acceptable ways of drawing the same symbol. Take the three resistors on the top row of Figure 1 for instance. They were all drawn with different numbers of ridges but all belong to the same resistor category. Low-level and conceptual variations present a major challenge for sketch recognition systems [17].

This paper presents a new approach to sketched symbol recognition based on visual appearance. This is in contrast to much of the work in the literature, which focuses on individual strokes and their temporal and spatial relationships. This emphasis on visual properties allows our method to be less sensitive to stroke order and direction, improving robustness and accuracy. We also present a new trainable symbol recognizer that is invariant to rotation and local deformations, making our approach more tolerant to the types of visual variations seen in Figure 1. The result is a more robust symbol recognizer that is better able to handle the range of drawing styles found in freehand sketches.

On-line Shape Recognition
Some of the earliest work in sketch recognition [19] focused on single stroke gestures, or glyphs. Symbols are recognized based on simple features such as the angle of the gesture or the area of the bounding box. Long et al. [15] developed an approach that analyzes the similarity between gestures to help developers design ones that will not be easily confused by the computer. While this early work has led to practical applications like the Graffiti handwriting system used by Palm, a major limitation of these approaches is the restrictions they place on how each symbol can be drawn.

Another common approach to sketch recognition focuses on structural shape descriptions. Here the base vocabulary is typically composed of geometric primitives such as lines, ellipses, and arcs. One work [6] modeled these primitives and the relationships between them as graphs, with recognition posed as a graph isomorphism problem. Shilman et. al. [20] used a hand coded visual grammar to describe shapes in the domain. Recognition is then treated as a visual parsing problem. Alvarado and Davis [2] proposed using dynamically

IUI 2009 Workshop on Sketch Recognition
February 8, 2009, Sanibel Island, Florida
Chair: Tracy Hammond

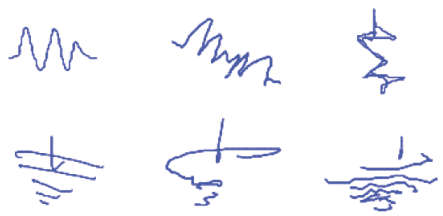

Figure 1. Six symbols from a dataset of electrical circuit diagrams. They illustrate the types of variations found in freehand sketches.

constructed Bayesian networks to parse a sketch, employing both top-down and bottom-up interpretation. Hammond and Davis [9] developed a language that described, among other things, the composition of a shape, and used it to do shape recognition.

A third approach, closer in spirit to what we do here, focuses on the visual appearance of shapes and symbols. Oltmans and Davis [17] proposed a visual parts-based approach to on-line sketch recognition. They build a library of parts (represented as oriented histograms of gradients) and use these parts to describe and distinguish between the different symbols in the domain. Shilman and Viola [21] presented a visual system for detecting and recognizing symbols in diagrams and equations. Their approach links individual strokes into a proximity graph, searching for symbols among spatially connected subgraphs using a Viola-Jones-like detector. Kara and Stahovich [11] developed a trainable, hand-drawn symbol recognizer based on pixel-based template matching and multiple similarity metrics.

Handwriting Recognition

Unlike most of the work in the preceding section, we designed our recognizer to handle handwritten characters as well as graphical shapes. This is important because letters and digits are often an essential part of many sketched diagrams. They may appear either in annotations (e.g., in electrical circuits) or as part of the underlying structure (e.g., in chemical diagrams).

An early motivation for this paper came from the observation that off-line handwriting recognizers, which operate on scanned images, perform very well despite the fact that they lack any information about pen trajectories. For example, current state-of-the-art techniques are able to achieve error rates in the range of 0.5% on a corpus of 70,000 scanned digits [14]. While a direct comparison between on-line and off-line handwriting recognition is difficult, a survey of past literature seems to suggest that off-line methods [13, 12] perform as well as, or even better than, on-line ones [4, 3, 7]. This raises an interesting question: can advances in off-line handwriting recognition, computer vision, and machine learning be adapted to make better on-line sketch recognizers?

OUR APPROACH

Following this intuition, we designed our approach to preserve as much of the visual properties of the symbol as possible. At the same time, we try to exploit the extra information we have about the temporal nature of the strokes.

The key contributions of our method are:

- It represents symbols as feature images rather than as temporally ordered points. This allows our approach to be more robust to differences in drawing style.
- It proposes a set of visual features for on-line symbol recognition that captures stroke properties like orientation and endpoint location.
- It introduces a classification technique that is computationally efficient and invariant to rotation and local deformations.
- It exceeds state-of-the-art performance on all of the datasets we evaluated. These include digits, common diagram shapes, and electrical circuit symbols.

Symbol Normalization

The first step in our approach is symbol normalization, which reduces the variation between symbols and eliminates differences in sampling, scale, and translation. This improves the robustness of our recognizer and makes the classification task easier.

Because on-line strokes are typically sampled at a constant temporal frequency, the distance between neighboring points in a trajectory varies based on the speed of the pen. This produces more samples in corners or regions of high curvature, where the pen is typically slower. In order to make calculation about stroke direction more reliable, we resample each stroke at a constant spatial frequency.

Next we try to remove differences in scale and translation. A traditional solution to this problem is to transform all of the symbols so that they have the same bounding box dimensions, but we found this technique to be sensitive to outliers caused by pen drag or stray ink. In response, we normalize each symbol by translating it so that its center of mass is at the origin, and scaling it horizontally and vertically so it has unit standard deviation in each axis.

Feature Representation

A key part of our approach is how we convert the on-line strokes into a set of low resolution feature images, seen in Figure 3. We begin by computing five features at each point in the pen trajectory, four concerned with stroke orientation and one concerned with stroke endpoints.

- The four orientation features correspond to four reference angles, at 0, 45, 90, and 135 degrees. They represent how nearly horizontal, vertical, or diagonal the pen stroke is as each point in the symbol. The feature values are calculated as the difference between the stroke angle and the reference angle, and vary linearly between 1.0 (if the two are equal) and 0.0 (if they differ by more than 22.5 degrees).

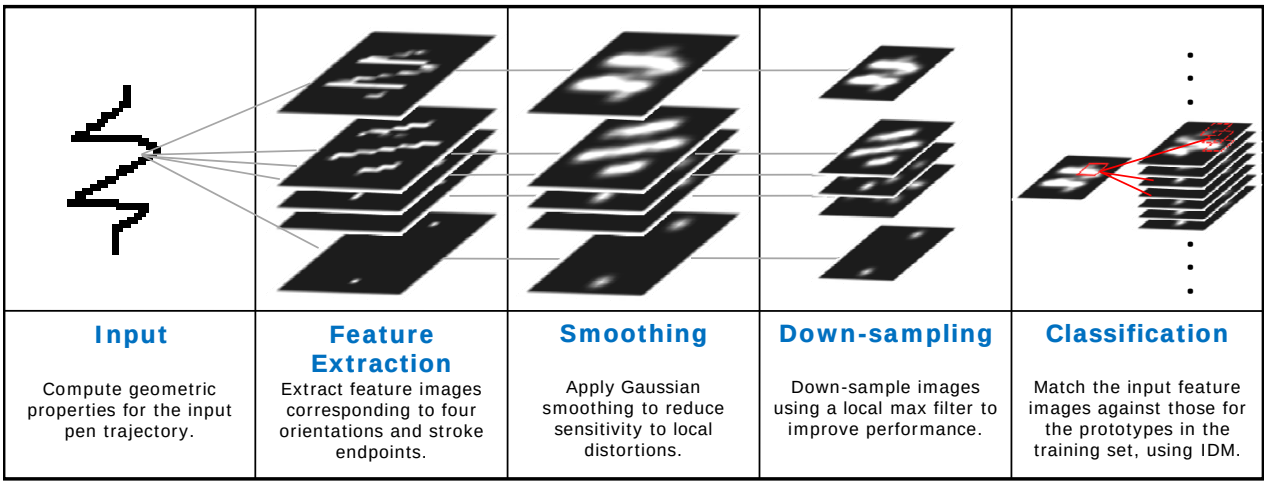

Figure 2. System Overview: First, a set of feature images representing the 4 orientations (top) and the endpoints (bottom) are extracted from the online stroke trajectory. Next, these images are smoothed and down-sampled to improve performance and increase tolerance to distortions. Finally, the images are compared against all of the prototypes in the training set using IDM.

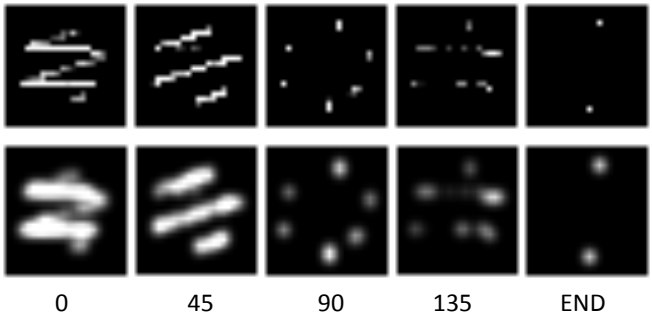

Figure 3. Feature Images: The set of five feature images generated from the input symbol. The top row shows the original image and the bottom row shows the smoothed image.

- The endpoint feature identifies the location of endpoints in the symbol. It is equal to 1.0 if the point is at the beginning or end of a stroke and 0.0 otherwise. While using endpoint information introduces more sensitivity to stroke-level variations, it helps us distinguish between symbols like "3" and "8", which look similar but usually differ in the location and number of endpoints.

The result is an ordered sequence of feature values, five for each sample point in the pen trajectory. In order to preserve the visual nature of the original input, we render these five features onto five 24 by 24 feature grids. The horizontal and vertical dimensions of the grid span 2.5 standard deviations of the original symbol's space in each direction. We can think of these grids as feature images, where the intensity of a pixel is determined by the maximum feature value of the sample points that fall within its cell. For example, the intensity of the 0-orientation image is high in regions where there are nearly horizontal segments. This representation resembles the annotated images used by LeRec [5] for handwriting recognition, but to our knowledge this is the first time it has been applied to sketched symbol recognition.

Smoothing and Downsampling

The next stage applies a smoothing and downsampling process to the feature images to increase tolerance to local shifts and distortions. First we apply a Gaussian smoothing function to each feature image that "spreads" feature values to neighboring pixels. This ensures that small spatial deformations in the symbol correspond to small changes in the features. We also downsample the images by a factor of 2 using a MAX filter (each pixel in the downsized image is the maximum of the four corresponding pixels in the original). This further reduces sensitivity to small shifts and improves runtime performance.

Recognition

For the symbol recognition task we combine nearest neighbor template matching with an image deformation model (IDM) that increases robustness to small translations and deformations. In this model, we allow every point in the input image to shift within a 3x3 local window while attempting to form the optimal Euclidean (L2) match to the prototype image. The individual shifts are independent, so computing this displacement mapping is computationally efficient. To avoid overfitting we use image patches instead of single pixels, shifting 3x3 sections of the input image at a time. An illustration of this process is shown in Figure 4.

The distance between two feature images I_1 (the input image) and I_2 (the template image) is thus given in equation (1):

$$D^2 = \sum_{x,y,c} \min_{D_x,D_y} (I_1(x+D_x, y+D_y, c) - I_2(x,y,c))^2 \quad (1)$$

where c indexes the 5 feature maps and D_x and D_y represent pixel shifts.

This image deformation model is similar to the one proposed by Keysers et al. [12] for off-line character recognition.

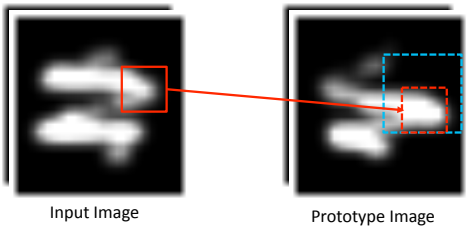

Figure 4. Image deformation model with local context matching. Each point in the input image can shift within a local window to form the best match to the prototype.

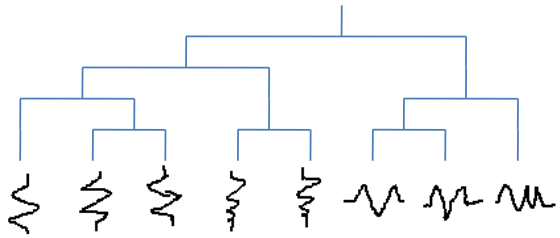

Figure 5. A hierarchical clustering dendrogram for a set of resistors.

Here, we extend their approach to on-line symbols using the orientation and endpoint features described above.

We compare our IDM against four benchmarks:

- Pixel Nearest Neighbor (PIXEL): A baseline nearest neighbor classifier that uses the Euclidean distance between the raw intensity images (no feature extraction).

- Feature Nearest Neighbor (NN): A nearest neighbor classifier that uses the sum of squared Euclidean distance from all five feature images.

$$D^2 = \sum_{x,y,c} (I_1(x,y,c) - I_2(x,y,c))^2 \tag{2}$$

- Hausdorff Nearest Neighbor: A nearest neighbor classifier that uses the Modified Hausdorff similarity metric. This technique has been applied previously to sketched symbol recognition and object matching [11, 8]:

$$H(A,B) = \max(h(A,B), h(B,A)) \tag{3}$$

where

$$h(A,B) = \frac{1}{N_a} \sum_{a \in A} (\min_{b \in B} ||a - b||) \tag{4}$$

- SVM: A Support Vector Machine trained on the 720 feature values from the five 12x12 feature images. Multi-class classification is implemented using a one-vs-one approach, combining the output from the full set of pairwise classifiers. We evaluate SVM performance using linear and Gaussian RBF kernels.

Feature Reduction using Principle Component Analysis

Even though the feature images we proposed are low resolution, they still produce 720 individual features values for each symbol. This is a large space in which to perform matching and classification, especially if we are computing similarities among thousands of examples. To improve runtime performance we first index the images using their first N principle components. We can then use this reduced feature set to compute approximate nearest neighbors based on the distance metric given in equation (4):

$$\hat{D}^2 = \sum_{n} (v_1(n) - v_2(n))^2 \tag{5}$$

where $v_i(n)$ represents the n-th principle component of the i-th image.

Hierarchical Clustering

One shortcoming of the nearest neighbor classification is the need to match against all of the training examples during inference. To reduce the computational requirements we introduce a hierarchical data structure that re-organizes and condenses the set of examples. First, we apply complete-link agglomerative hierarchical clustering to all of training examples in each class. This process starts with each example in its own cluster, then progressively merges the two nearest clusters based on Euclidean distance. This continues until there is only one cluster per symbol class. Figure 5 illustrates this process.

The result of this clustering process is a tree-like structure with the largest clusters at the top and progressively smaller sub-clusters below as we traverse down each branch. For each cluster and sub-cluster, we extract a representative prototype (or cluster center). This is defined as the instance that is the most similar (i.e., has the minimum average distance) to all of the other examples in the cluster. In addition to the cluster center, we also store the radius r of the cluster, the maximum distance between the center and any of its members.

During inference, we first compare the input symbol to the central prototype of each cluster, starting with clusters at the top level of the hierarchy. We then choose the best cluster center and recursively descend down that branch of the search tree. The algorithm keeps track of the best match distance encountered so far, while at the same time discarding clusters that cannot possibly improve on this best match. Assuming our metric follows the triangle inequality, the best possible match in cluster c is the distance to the cluster center d_c minus the cluster radius r_c. If $d_c - r_c$ is worse than the distance to the best match, we can safely discard the cluster. The process stops when there are no more clusters to expand.

If we are interested in finding the N-nearest neighbors we need to make a few modifications to the above method. First, instead of keeping track of only the best match, we store a list of N-best matches. Second, we discard clusters if $d_c - r_c$

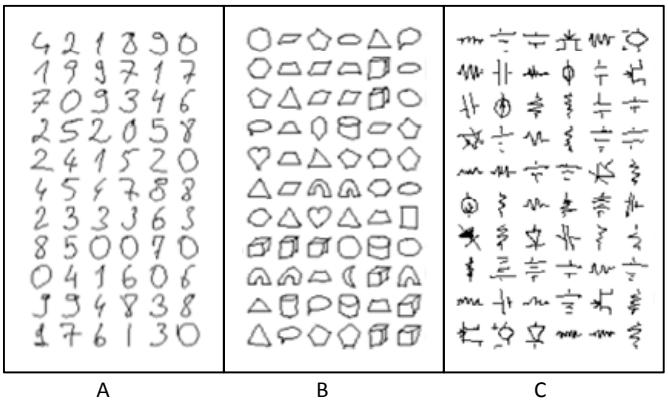

Figure 6. Example symbols from the three different evaluation datasets: A) pen digits, B) common diagram shapes, and C) electrical circuit symbols.

is greater than the worst of the N-best distances so far.

Coarse-to-fine Reranking

Even with the hierarchical search strategy described above, performing the IDM matches can require several seconds per symbol on a training set size of about 1000. To improve performance even further, we first apply a fast approximate recognizer to produce an N-best list of nearest neighbor candidates. Then, in the second stage, we rerank those candidates using the slower, more exact recognizer. In our implementation we use the hierarchical search strategy and PCA feature reduction to quickly find the best candidates for reranking[1].

Rotational Invariance

The recognition process described so far is robust to differences in translation, scale, and local deformation. To make our recognizer invariant to rotation, we generate and match rotated versions of the test symbol to each of the training examples. In our experiments we use 32 evenly spaced orientations from 0 to 360 degrees. For the hierarchical classifier, we perform rotation matching only at the top levels[2] to improve performance. For the lower level comparisons we reuse the best rotation from the parent cluster. Similarly, in the coarse-to-fine classifier, we reuse the optimal rotation from the approximate recognizer in the more expensive reranking stage.

EXPERIMENTAL RESULTS

We evaluated performance of our method on three diverse datasets: handwritten digits, diagram shapes, and engineering symbols, and given comparisons of our method (in **bold**) against the benchmarks described earlier, as well as other results reported in the literature (in *italics*). Note that for the IDM and NN based methods we use hierarchical clustering and coarse-to-fine reranking to improve runtime speed.

[1]In our implementation we use the first 128 principle components and rerank the first 10 nearest neighbor candidates. These parameters were chosen empirically; significantly lower values degrade accuracy while higher values do not seem to offer any improvement.

[2]We use the top 64 clusters for each category

Pen Digits

This dataset contains about 10,000 isolated on-line digits [1]; a sample is shown in Figure 6A. This corpus consists of 7,494 training examples and 3,498 test examples. The 30 writers of the training examples are independent of the 14 writers of the test examples, so the results also suggest how well our system would be able to generalize to new users. Here we compare our approach against two benchmarks from the on-line handwriting recognition community. Note that for this experiment we omit rotation invariant techniques since we do not expect rotations to be an issue.

Method	Accuracy
SVM-RBF	99.4%
IDM	**99.2**%
SVM-Linear	99.1%
NN	98.9%
Eigen-Deformation [16]	*98.2%*
Hausdorff	97.6%
PIXEL	97.1%
Alimoglu et. al. [1]	*97.0%*

Table 1. Comparison of recognition results for the Pen Digits dataset.

The results in Table 1 show that our approach is able to outperform both previous benchmarks. Compared to the approach in [16], our method is able to reduce the relative error rate by 56%. Here the SVM-RBF model did slightly better than the IDM model, but at the cost of much greater computational complexity (see table 4, below). The mistakes on this dataset are displayed in Figure 8.

HHReco

The HHReco dataset [10] consists of 7,791 diagram shapes (e.g., boxes, callouts, etc) like those shown in Figure 6B. The examples were collected from 19 different people, each of whom drew at least 30 examples per category. In each of our cross validation trials our system was tested on the examples from one user after it was trained on data from all of the other users.

Method	Accuracy
IDM+Rotate	**98.2**%
Zernike Moments [10]	*96.7%*
IDM	95.2%
NN	95.1%
SVM-RBF	95.0%
Visual Parts [17]	*94.4%*
Hausdorff	93.0%
SVM-Linear	92.3%
PIXEL	92.2%

Table 2. Comparison of recognition results for the HHReco common shapes dataset.

On this dataset the best method is IDM+Rotate, which achieves an accuracy of 98.2%. Compared to the 96.7% accuracy reported by Hse et al. [10], the next best external benchmark, our method provides a 45% reduction in relative error rate. Comparing the performance of the IDM and IDM+Rotate

classifiers, we see that rotation invariance is an important feature for this corpus.

Electrical Circuits

The Electrical Circuits dataset contains 1,012 examples of circuit components like the ones shown in Figure 6C. Unlike the previous two cases, these symbols were extracted from full sketches and exhibit a larger range of variations and styles. The symbols themselves are also more complex than those in the other two domains, on average containing a larger number of strokes and sub-components.

Method	Accuracy
IDM+Rotate	**96.2%**
IDM	93.9%
SVM-RBF	91.9%
NN	91.2%
SVM-Linear	90.1%
Visual Parts [17]	*89.5%*
Hausdorff	79.9%
Zernike Moments [10]	*76.9%*
PIXEL	75.9%

Table 3. Recognition accuracy for the electrical circuit dataset.

For this dataset our method achieves an accuracy rate of 96.2%. This represents a 64% relative error reduction over the best published benchmark of 89.5% [17]. The errors made by our method are displayed in Figure 9.

Runtime Performance

We have also evaluated the relative performance of the different classifiers presented in this paper and determined the average time required to classify one symbol in the Pen Digits dataset on a 2.4 Ghz machine. As Table 4 shows, the IDM method achieves very good runtime performance, able to process over 100 symbols per second on this corpus. We also see that hierarchical clustering and reranking provide over two orders of magnitude in speedup over the unoptimized version of IDM. This is essential if our eventual goal is to achieve real time recognition.

Method	Runtime
SVM-Linear	2.4 ms
IDM (Hierarchical+Rerank)	**8.1 ms**
NN (Full)	40.8 ms
SVM-RBF	90.3 ms
Hausdorff	750 ms
IDM (Full)	3952 ms

Table 4. The average time required to classify a symbol in the Pen Digits corpus using the different methods described in the paper.

DISCUSSION

This work focused on developing a fast, accurate, and robust sketched symbol recognizer that is designed to work in multiple domains. However, symbols in real sketches, like in Figure 7, are typically not drawn in isolation; neighboring symbols may be touching and multiple shapes may be drawn using the same stroke without lifting the pen. A complete recognition system will need to address the problems of symbol detection and segmentation, extracting valid symbols from messy sketches.

Although we did not look at these problems explicitly in this paper, previous works have successfully used the output of an isolated symbol recognizer to guide detection and segmentation [18, 17, 21]. We believe that accurate and robust low level recognition is essential for high level understanding. In future work we will explore how the techniques presented in this paper can be applied to the problem of full sketch recognition.

CONCLUSION

We have presented a new visual approach to on-line symbol recognition. Unlike much of the previous work in this area, we model each symbol using annotated feature images rather than temporal sequences or stroke primitives. As a result, our method is less sensitive to variations in drawing style that affect stroke order and direction. It also uses a classification technique that is robust to rotation and local deformations, further increasing accuracy. Finally, the method we proposed is computationally efficient and is able to exceed state-of-the-art performance for all the domains we evaluated, including handwritten digits, common diagram shapes, and electrical circuit symbols.

REFERENCES

1. F. Alimoglu and E. Alpaydin. Combining multiple representations and classifiers for pen-based handwritten digit recognition. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 1997.
2. C. Alvarado and R. Davis. Sketchread: A multi-domain sketch recognition engine. In *Proceedings of UIST*, 2004.
3. C. Bahlmann and H. Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 299–310, 2004.
4. C. Bahlmann, B. Haasdonk, and H. Burkhardt. Online handwriting recognition with support vector machines - a kernel approach. *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition, 2002.*, pages 49–54, 2002.
5. Y. Bengio, Y. LeCun, C. Nohl, and C. Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 7(6):1289–1303, 1995.
6. C. Calhoun, T. Stahovich, T. Kurtoglu, and L. Kara. Recognizing multi-stroke symbols. *AAAI Spring Symposium on Sketch Understanding*, pages 15–23, 2002.
7. S. Connell and A. Jain. Template-based online character recognition. *Pattern Recognition*, 34(1):1–14, 2001.

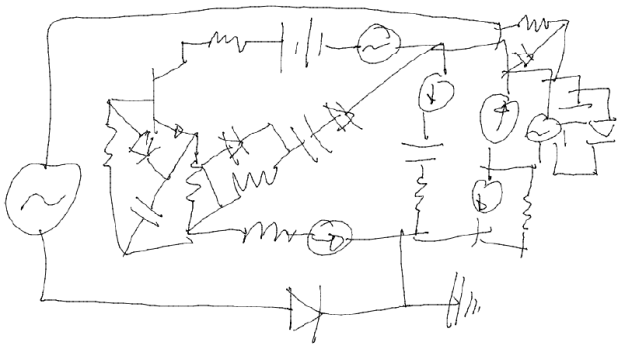

Figure 7. A complete sketch of an electrical circuit. Symbols from the electrical circuits dataset were extracted from diagrams like this one.

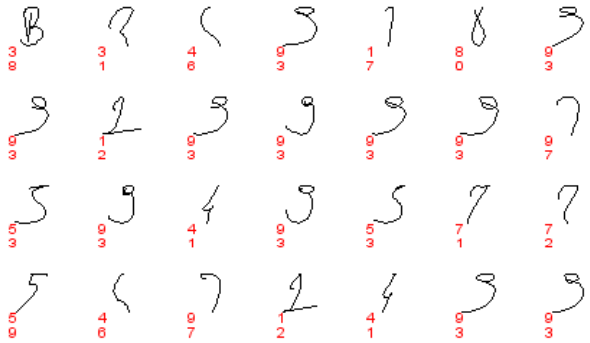

Figure 8. All errors made by our method on the Pen Digits dataset. The correct class is on the top and the predicted class is on the bottom.

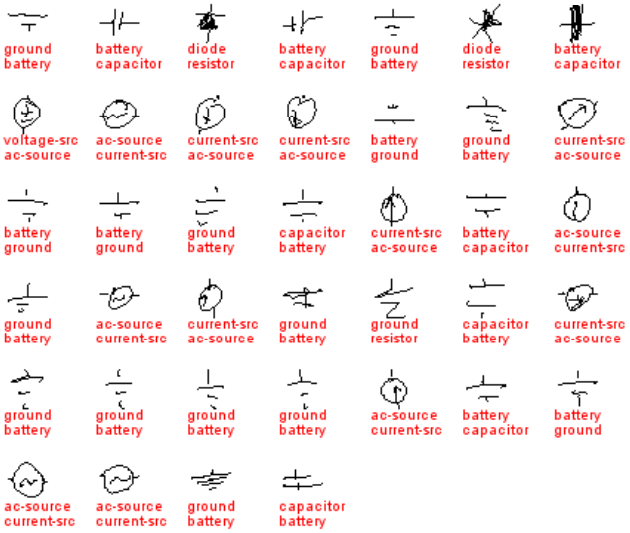

Figure 9. All errors made by our method on the Circuits dataset. The correct class is on the top and the predicted class is on the bottom.

8. M. Dubuisson and A. Jain. A modified hausdorff distance for object matching. In *Proceedings of the 12th International Conference on Image Processing*, 1994.

9. T. Hammond and R. Davis. LADDER: a language to describe drawing, display, and editing in sketch recognition. In *International Conference on Computer Graphics and Interactive Techniques*, 2006.

10. H. Hse and A. Newton. Sketched symbol recognition using Zernike moments. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2004.

11. L. Kara and T. Stahovich. An image-based trainable symbol recognizer for sketch-based interfaces. *AAAI Fall Symposium: Making Pen-Based Interaction Intelligent and Natural*, 2004.

12. D. Keysers, C. Gollan, and H. Ney. Local context in non-linear deformation models for handwritten character recognition. In *International Conference on Pattern Recognition*, 2004.

13. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

14. Y. LeCun and C. Cortes. The mnist database of handwritten digits. *NEC Research Institute, http://yann. lecun. com/exdb/mnist/index. html*.

15. A. Long, J. Landay, L. Rowe, and J. Michiels. Visual similarity of pen gestures. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000.

16. H. Mitoma, S. Uchida, and H. Sakoe. Online character recognition using eigen-deformations. *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 3–8, 2004.

17. M. Oltmans. *Envisioning Sketch Recognition: A Local Feature Based Approach to Recognizing Informal Sketches*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, May 2007.

18. T. Ouyang and R. Davis. Recognition of hand drawn chemical diagrams. In *Proceedings of AAAI*, 2007.

19. D. Rubine. Specifying gestures by example. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 329–337, 1991.

20. M. Shilman, H. Pasula, S. Russell, and R. Newton. Statistical visual language models for ink parsing. *AAAI Spring Symposium on Sketch Understanding*, 2002.

21. M. Shilman, P. Viola, and K. Chellapilla. Recognition and grouping of handwritten text in diagrams and equations. In *Frontiers in Handwriting Recognition*, 2004.