A Visual Approach to Sketched Symbol Recognition

Tom Y. Ouyang and Randall Davis
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{ouyang,davis}@csail.mit.edu

Abstract

There is increasing interest in building systems that can automatically interpret hand-drawn sketches. However, many challenges remain in terms of recognition accuracy, robustness to different drawing styles, and ability to generalize across multiple domains. To address these challenges, we propose a new approach to sketched symbol recognition that focuses on the visual appearance of the symbols. This allows us to better handle the range of visual and stroke-level variations found in freehand drawings. We also present a new symbol classifier that is computationally efficient and invariant to rotation and local deformations. We show that our method exceeds state-of-the-art performance on all three domains we evaluated, including handwritten digits, PowerPoint shapes, and electrical circuit symbols.

1 Introduction

Diagrams are an essential means of capturing and communicating information in many different domains. They can also be 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 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 (all correctly identified using the method in this paper). These symbols clearly 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 may be drawn differently at the stroke level. For example, the strokes in the two symbols may differ in their order and direction. Finally, symbols may exhibit artifacts like overtracing (drawing over a previously drawn stroke) and pen-drag (failing to lift the pen between strokes). These types of variations present a major challenge for sketch recognition systems [Oltmans, 2007].

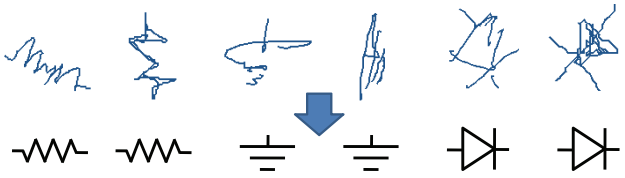

Figure 1: Six symbols from a dataset of electrical circuit diagrams that our method identified correctly. They illustrate the types of variations found in freehand sketches.

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 or geometric primitives and their temporal and spatial relationships. This emphasis on visual properties makes our method less sensitive to stroke level differences, improving robustness and accuracy. We also present a new symbol classifier that is invariant to rotation and local deformations, making it more tolerant to the types of visual variations we see 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.

1.1 Shape Recognition

One common approach to sketch recognition focuses on building structural shape descriptions. Here the base vocabulary is typically composed of simple geometric primitives such as lines, arcs, and ellipses. [Shilman *et al.*, 2002] used a hand coded visual grammar to describe shapes in the domain, treating recognition as an ambiguous parsing problem. [Alvarado and Davis, 2004] proposed using dynamically constructed Bayesian networks to parse a sketch, employing both top-down and bottom-up interpretation. [Hammond and Davis, 2006] developed a hierarchical language to describe how diagrams are drawn, displayed, and edited. They then used these descriptions to perform automatic symbol recognition.

An alternative approach, closer in spirit to what we do here,

looks at the visual appearance of shapes and symbols. [Kara and Stahovich, 2004] developed a trainable symbol recognizer that uses four image-based similarity metrics to perform template matching. [Shilman *et al.*, 2004] described a method for grouping and recognizing symbols in diagrams and equations, using a Viola-Jones-like detector to search for symbols among spatially connected strokes. [Oltmans, 2007] proposed a visual parts-based model that uses a library of shape contexts (oriented histograms of gradients) to describe and distinguish between the different symbols in their domain.

1.2 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 sketched diagrams, where they may appear either as annotations (e.g., in electrical circuits) or as part of the underlying structure (e.g., in chemical diagrams).

An early motivation for our approach came from the observation that current 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, 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 [Lecun *et al.*, 1998]. While a direct comparison between on-line and off-line handwriting recognition is difficult, a survey of past literature suggests that off-line methods [Lecun *et al.*, 1998; Keysers *et al.*, 2004] can perform as well as, or even better than, their on-line counterparts [Connell and Jain, 2001; Bahlmann *et al.*, 2002; Mitoma *et al.*, 2004]. This lead us to ask, can advances in off-line handwriting recognition be adapted to make better on-line sketch recognizers?

2 Our Approach

Following this intuition we designed our approach to focus primarily on the visual properties of the symbols. However, unlike purely off-line methods, we also try to exploit the extra information we have about the temporal nature of the strokes. An overview of the recognition process is shown in Figure 2.

The key contributions of our method are:

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

2.1 Symbol Normalization

In sketch interpretation it is often important to be able to recognize a symbol regardless of its size or position. Therefore, the first step in our approach is to eliminate differences due to sampling, scale, and translation. This improves the robustness of our recognizer, ensuring that the input symbols are all centered and scaled appropriately.

Since on-line strokes are typically sampled at a constant temporal frequency, the distance between neighboring points in the pen 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 feature extraction more reliable, we resample each stroke at a constant spatial frequency.

Next we remove differences due to 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 overly sensitive to artifacts like long tails at the ends of strokes or stray ink. In response, we normalize each symbol by translating its center of mass to the origin, and scaling it horizontally and vertically so it has unit standard deviation in both axes.

2.2 Feature Representation

A key part of our approach is how we convert the on-line stroke sequences into a set of low resolution feature images. We begin by computing five features for each sample 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 measure how nearly horizontal, vertical, or diagonal the stroke is at each point. 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 45 degrees). One major advantage of this representation is that it is independent to stroke direction. A stroke drawn left to right has the same orientation as one drawn right to left.
- The endpoint feature identifies stroke endpoints. It is equal to 1.0 if the point is at the beginning or end of a stroke and 0.0 otherwise. This feature helps us distinguish between symbols like “3” and “8”, which look similar but often differ in their endpoint locations.

The result is an ordered sequence of feature values, five for each point in the symbol. In order to preserve the spatial 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, in which 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 the stroke direction is nearly horizontal. This representation resembles the annotated images used by LeRec for handwriting recognition [Bengio *et al.*, 1995], but to our knowledge this is the first time it has been applied to sketched symbol recognition.

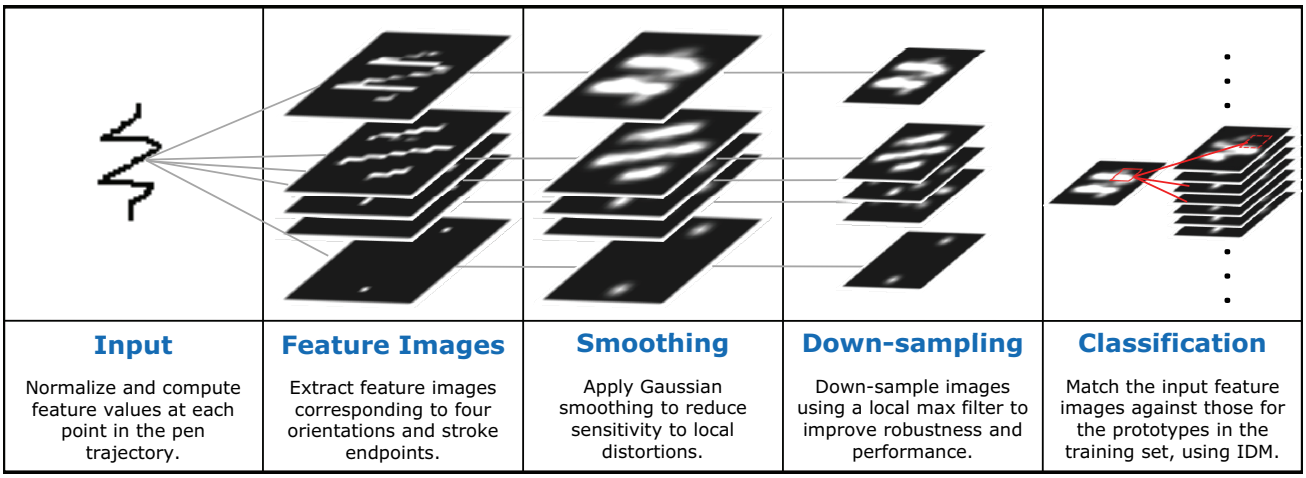

Figure 2: System Overview: First, a set of feature images representing the 4 orientations (top) and the endpoints (bottom) are extracted from the online stroke trajectory. Next, these images are smoothed and down-sampled to improve performance and increase tolerance to distortions. Finally, the images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

2.3 Smoothing and Downsampling

In the next stage we smooth and downsample the feature images to increase tolerance to local shifts and distortions. First we apply a Gaussian smoothing function to each image that "spreads" feature values to neighboring pixels. This ensures that small spatial variations in the symbol correspond to gradual changes in the feature values. We then downsample the images by a factor of 2 using a MAX filter, where 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.

2.4 Recognition

For the symbol recognition task we use a deformable template matching algorithm that is robust to local shifts and distortions. Our image deformation model (IDM) allows every point in the input image to shift within a 3x3 local window to form the best match to the prototype image. The individual shifts are independent, so computing this displacement mapping is computationally efficient. To avoid overfitting, we include the local context around each point, shifting 3x3 image patches instead of single pixels. The distance between two points is then calculated as the sum of squared differences between the five feature images at their respective patch locations. An illustration of this process is shown in Figure 3.

The IDM distance between two symbols I_1 (the input) and I_2 (the template) is defined as:

$$D^2 = \sum_{x,y} \min_{d_x,d_y} ||I_1(x+d_x, y+d_y) - I_2(x,y)||^2 \qquad (1)$$

where d_x and d_y represent pixel shifts and $I_i(x,y)$ represents the 3x3x5 feature values in I_i from the patch centered at x, y.

This image deformation model is similar to the one proposed by [Keysers *et al.*, 2004] for off-line character recognition. Here, we extend their approach to on-line symbols using the orientation and endpoint features described earlier.

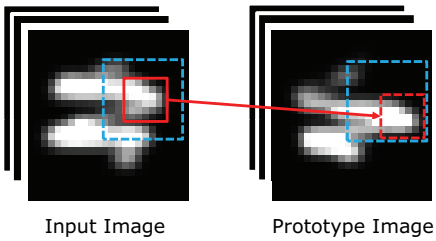

Figure 3: Image deformation model with local context matching. Each point in the input image can shift within a local window to form the best match to the prototype.

2.5 Performance Optimizations

One major limitation of the deformable template model is that it needs to match the input symbol against all of the training examples. As a result, computing the full set of IDM matches can take several seconds on a modestly sized training set of 7000 templates. This section describes two performance optimizations that, when combined, can reduce this runtime by over two orders of magnitude.

Coarse Candidate Pruning

Since applying IDM to the full training set is too slow, the first optimization is to use a fast "coarse" metric to prune the set of candidates before applying the more expensive "exact" metric. In our implementation, this is equivalent to passing only the first N nearest neighbors found by the coarse metric to the IDM matcher.

While simple Euclidean L_2 distance would work fairly well as the pruning metric, it would still involve comparing all 720 values in the feature images. We can improve performance even further by indexing these images using their first K principle components. Then, we use the distance between these reduced feature sets to find the nearest candidates, as shown in equation (2):

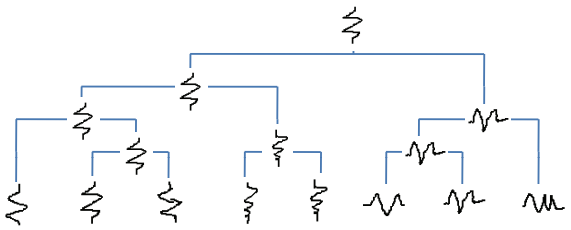

Figure 4: A hierarchical clustering tree for a set of resistors.

$$\hat{D}^2 = \sum_{k=1}^{K} (v_1(k) - v_2(k))^2 \quad (2)$$

where $v_i(k)$ is the k-th principle component of the i-th image.

Hierarchical Clustering
The second optimization is a branch and bound technique to speed up the coarse classifier even further. It begins by applying agglomerative hierarchical clustering to the training examples in each class, organizing them into groups based on complete-link distance. This process first initializes each symbol into its own cluster, then progressively merges the two nearest clusters until there is only one cluster per class. At each step, it records the two sub-clusters that were merged to form the parent.

The result is a hierarchical tree structure with the largest clusters at the top and progressively smaller sub-clusters below. For each cluster and sub-cluster, it selects a representative prototype, the cluster center. This is defined as the example that is maximally similar to all of the other examples in the cluster. Next, it computes the cluster radius as the maximum distance between the center and any of its members. Figure 4 shows the result of this process on a collection of resistors. The leaves in the tree represent individual templates and the nodes represent cluster centers.

During inference, the algorithm compares the input symbol to the set of cluster centers, starting at the top level of the hierarchy. It keeps track of the best match discovered so far, discarding clusters when it knows they cannot improve this match. Assuming our metric follows the triangle inequality, the lower-bound on the best 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 larger than the best distance discovered so far, we can safely ignore the entire cluster. If not, the algorithm expands the cluster and repeats the process for its children.

Since we want to find the N-nearest neighbors, we need to make a couple of modifications to the above algorithm. First, instead of keeping track of only the best match, we store a list of N-best matches. Second, we discard a cluster only if its lower bound is worse than the N-th best match discovered so far.[1]

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

2.6 Rotational Invariance

The recognition process described so far is robust to differences in translation, scale, and local deformation. The next step is to make our recognizer invariant to rotation. Our solution is to generate and match rotated versions of the input symbol to each of the training examples. In our implementation we use 32 evenly spaced orientations from 0 to 360 degrees. To improve performance, in the hierarchical clustering stage we perform rotation matching only for the top level clusters.[2] For the lower level clusters, we can assume that all of the members are similar enough that they share the same rotation as the parent. Similarly, in the IDM stage, we assume that the optimal rotation found by the coarse metric is correct and reuse it in the exact match.

3 Experimental Evaluation

We evaluate our approach on three datasets: handwritten digits, PowerPoint shapes, and circuit symbols. The following tables compare our performance (shown in **bold**) against four benchmark classifiers (described below) and previous results reported in literature (shown in *italics*). Note that in all three evaluations we use the same optimized version of the IDM recognizer, incorporating hierarchical clustering and PCA candidate pruning. The only exception is that we omit rotation invariance on the digits dataset because it would have made it impossible to distinguish between digits like "6" and "9".

Benchmarks
We include the following four benchmark classifiers:

- Pixel Nearest Neighbor (PIXEL NN): A baseline nearest neighbor classifier that uses the L_2 distance between the raw intensity images (no feature images).
- Feature Nearest Neighbor (FEATURE NN): A nearest neighbor classifier that uses the L_2 distance between the five feature images.
- Hausdorff Nearest Neighbor (HAUSDORFF): A nearest neighbor classifier that uses the Modified Hausdorff distance [Kara and Stahovich, 2004; Dubuisson and Jain, 1994]. This metric has been used previously for sketch and object recognition.
- SVM: A Support Vector Machine that uses a single 720-length feature vector to represent the five 12x12 feature images. We evaluated the performance using a LINEAR kernel and a RBF kernel.

3.1 Pen Digits

This dataset, first presented in [Alimoglu and Alpaydin, 1997], contains 10,992 isolated digits. It is divided into 30 writers for the training set and 14 writers for the test set, with no overlap between the two groups. Therefore, this evaluation is writer-independent and indicates how well our system is able to generalize to new users.

[2]We use rotation matching for the top 64 clusters in each class.

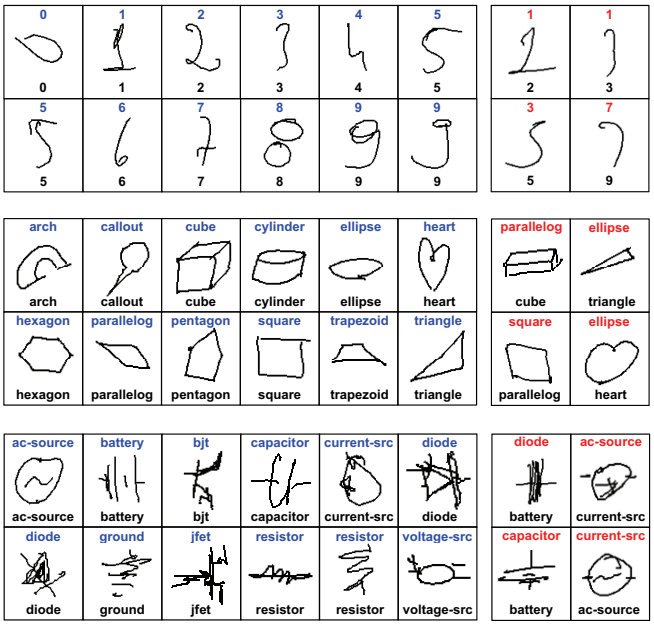

Figure 5: Examples of symbols that our recognizer classified **correctly** (left) and **incorrectly** (right). The predictions made by our system are shown above each symbol and the ground truth labels are shown below.

As we see in Table 1, our method correctly recognized 99.2% of the digits in this corpus, outperforming both external benchmarks. Compared to [Mitoma *et al.*, 2004], we were able to reduce the relative error rate by 56%, eliminating over half of the errors. The examples in Figure 5 (top) show that our method successfully handled a wide range of writing styles, and many of the symbols it missed could be considered difficult even for a human reader. Here the SVM-RBF model did slightly better than IDM, but at the cost of greater computational requirements (see Table 4, below).

3.2 HHReco PowerPoint Shapes

The HHReco dataset [Hse and Newton, 2004] includes 7,791 PowerPoint shapes like boxes, ellipses, cylinders, and callouts. The examples were collected from 19 different people, and each person drew at least 30 symbols per category. In our user independent cross-validation trials, we test on the examples from each user after training on the remaining 18 users.

As Table 2 shows, on this dataset we achieved an accuracy rate of 98.2%. Compared to the 96.7% result reported by [Hse and Newton, 2004], the best external benchmark, our method offers a 45% reduction in relative error. Figure 5 shows a number of examples that our system identified correctly and incorrectly.

3.3 Electrical Circuits

The Electrical Circuits dataset [Oltmans, 2007] contains 1,012 examples of circuit symbols like the ones in Figure 5 (bottom). Unlike the previous two domains, these symbols were extracted from complete sketches. As a result, they seem to exhibit a wider range of variations and drawing styles. We again evaluate using user-independent cross validation.

For this corpus our method achieved an accuracy rate of 96.2% (see Table 3). This represents a 64% relative error reduction over the best published benchmark of 89.5% [Oltmans, 2007]. As we can see in Figure 5, our method was able to correctly identify several very difficult examples, including ones that exhibit significant over-tracing and pen-drag.

3.4 Runtime Performance

Finally, we evaluate the runtime performance of the different classifiers presented in this paper, measuring the average time required to recognize one symbol in the Pen Digits dataset (running on a 2.4 GHz machine). As Table 4 shows, our IDM method is very computationally efficient, able to process over 100 symbols per second. Compared to the unoptimized ver-

Pen Digits Dataset	**Accuracy**
SVM-RBF	99.4%
IDM	**99.2%**
SVM-Linear	99.1%
FEATURE NN	98.9%
Eigen-Deformation [Mitoma et al.*, 2004]*	*98.2%*
Hausdorff	97.6%
PIXEL NN	97.1%
Combination [Alimoglu and Alpaydin, 1997]	*97.0%*

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

HHReco Dataset	**Accuracy**
IDM	**98.2%**
Zernike Moments [Hse and Newton, 2004]	*96.7%*
IDM (no rotation)	95.2%
FEATURE NN	95.1%
SVM-RBF	95.0%
Visual Parts [Oltmans, 2007]	*94.4%*
Hausdorff	93.0%
SVM-Linear	92.3%
PIXEL NN	92.2%

Table 2: Comparison of recognition results for the HHReco dataset.

Circuits Dataset	**Accuracy**
IDM	**96.2%**
IDM (no rotation)	93.9%
SVM-RBF	91.9%
FEATURE NN	91.2%
SVM-Linear	90.1%
Visual Parts [Oltmans, 2007]	*89.5%*
Hausdorff	79.9%
Zernike Moments [Hse and Newton, 2004]	*76.9%*
PIXEL NN	75.9%

Table 3: Comparison of recognition results for the Electrical Circuits dataset.

Runtime Performance	Time
SVM-Linear	2.4 ms
IDM	**8.1 ms**
FEATURE NN	40.8 ms
SVM-RBF	90.3 ms
Hausdorff	750 ms
IDM (unoptimized)	3952 ms

Table 4: The average time required to classify a symbol in the Pen Digits corpus.

sion, it improves classification speed by over two orders of magnitude. This speedup is essential if our eventual goal is to achieve real time recognition.

4 Discussion

This work focused on developing a fast, accurate, and robust sketched symbol recognizer that works for multiple domains. However, symbols in real sketches are rarely drawn in isolation; neighboring symbols may be close together or even touching, and multiple symbols may be drawn using a single pen stroke. A complete recognition system will need to address the problems of sketch segmentation and symbol detection, extracting valid symbols from messy sketches.

Although we did not look at these problems in this paper, previous works have successfully used the output of an isolated symbol recognizer to guide segmentation and detection [Ouyang and Davis, 2007; Oltmans, 2007; Shilman *et al.*, 2004]. Even though it is only one part of the solution, we believe that accurate and robust low level recognition is essential for high level understanding. In our future work we will explore how the ideas presented here can be extended to full sketch recognition.

5 Conclusion

In this paper we proposed a new visual approach to on-line symbol recognition. Unlike much of the previous work in this area, we represent symbols as visual feature images rather than as temporal sequences or geometric primitives. As a result, our method is less sensitive to the variations in drawing style that can pose major challenges for other sketch recognition systems. We also presented a classification technique that is computationally efficient and robust to rotation and local deformations. Finally, we showed that our method is able to exceed state-of-the-art performance for all the domains we evaluated, including handwritten digits, PowerPoint shapes, and electrical circuit symbols.

Acknowledgements

This research was supported in part by a DHS Graduate Research Fellowship and by Pfizer, Inc.

References

[Alimoglu and Alpaydin, 1997] F. Alimoglu and E. Alpaydin. Combining multiple representations and classifiers for pen-based handwritten digit recognition. In *Proceedings of ICDAR*, 1997.

[Alvarado and Davis, 2004] Christine Alvarado and Randall Davis. Sketchread: A multi-domain sketch recognition engine. In *Proceedings of UIST*, 2004.

[Bahlmann *et al.*, 2002] C. Bahlmann, B. Haasdonk, and H. Burkhardt. Online handwriting recognition with support vector machines - a kernel approach. *Proceedings of IWHFR*, pages 49–54, 2002.

[Bengio *et al.*, 1995] 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.

[Connell and Jain, 2001] S.D. Connell and A.K. Jain. Template-based online character recognition. *Pattern Recognition*, 34(1):1–14, 2001.

[Dubuisson and Jain, 1994] M.P. Dubuisson and AK Jain. A modified hausdorff distance for object matching. In *Proceedings of ICIP*, 1994.

[Hammond and Davis, 2006] 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.

[Hse and Newton, 2004] H. Hse and AR Newton. Sketched symbol recognition using zernike moments. In *Proceedings of ICPR*, 2004.

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

[Keysers *et al.*, 2004] 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.

[Lecun *et al.*, 1998] 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.

[Mitoma *et al.*, 2004] H. Mitoma, S. Uchida, and H. Sakoe. Online character recognition using eigen-deformations. *Proceedings of IWFHR*, pages 3–8, 2004.

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

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

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

[Shilman *et al.*, 2004] M. Shilman, P. Viola, and K. Chellapilla. Recognition and grouping of handwritten text in diagrams and equations. In *Proceedings of IWFHR*, 2004.