SHAPE RECOGNITION USING FAST BOOSTED FILTERING

Yen-Yu Lin *Tyng-Luh Liu*

Institute of Information Science, Academia Sinica, Nankang, Taipei 115, Taiwan

ABSTRACT

We address the problem of recognizing 2-D shapes in images via multi-class classifications. Our approach has three key elements. First, a signed distance transform is introduced to represent a shape more informatively. Second, a filter bank is generated such that its filters can capture multiple-scale local and global features between two shapes of different classes. We then apply boosting to combine useful filters to construct discriminant classifiers. Third, in implementing our system, a new classification architecture is developed to accomplish multi-class recognition. To examine the claimed efficiencies, we consider an example of document recognition by pinpointing the strengths of our method through experimental results and comparisons.

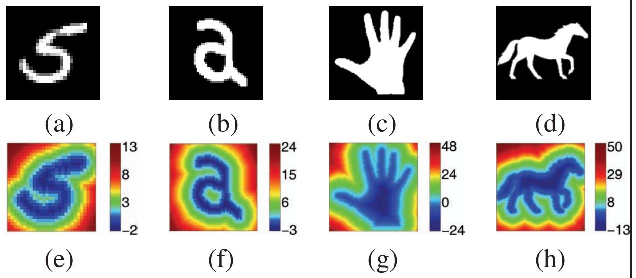


Fig. 1. (a)–(d) Examples of silhouettes/shapes of interest. (e)–(f) Their respective compact representations after applying signed distance transform, where the color bars show pixels' signed distance values to the boundaries.

1. INTRODUCTION

Our primary goal is to establish a general approach to recognize 2-D shapes/silhouettes through classifications. Specifically, the crux of the proposed method is a learning technique that employs boosted filtering over an effective shape representation model. With training data consisting of images of various shapes, we systematically identify important local and global features to characterize the dissimilarities among different classes. The resulting efficiency is illustrated with an extensive example of digit recognition. Indeed, compared with other existing methods, our approach achieves much higher efficiency with the accuracy rate approaching the state-of-the-art, e.g., [1], [2].

2. SHAPE REPRESENTATION

In many image processing and computer vision applications, e.g., OCR [1], gait recognition [3], and shape database management [4], to establish an effective and compact shape representation model often plays a critical role on the overall performance of such a system.

Perhaps the most explicit way to represent a 2-D shape is by a set of pixels contained within the shape. Alternatively one can describe a shape by its boundary contour(s), which is typically a more compact representation than the straightforward region representation. For more sophisticated shape descriptors, the work of Felzenszwalb [5] uses a triangulated polygon representation for shapes with possible deformations. In [6], based on the self-similarity of shapes, Geiger et al. introduce a variational framework that computes their *shape axes*, and establish a compact representation to account for deformations and articulations. The *shape context* [2] proposed by Belongie et al. can be used to compute the coarse distribution with respect to each point on a shape, and it naturally induces a shape matching algorithm via bipartite matching. However, while these systems are useful for various problems, they are often computationally too expensive for on-line applications.

In view of efficiency, we instead consider a new shape descriptor based on the following two practical issues.

- Most information about a shape/silhouette is given by the foreground pixels (e.g., those white pixels in Fig. 1a–Fig. 1d). However, the background (black) pixels often occupy most of the regions. They both should be taken into account in constructing an informative representation.
- To encode the background and the foreground information of a shape, we simply use the Euclidean distance to depict their topological relations to the shape boundaries, i.e., those foreground pixels having background pixels as their neighbors.

Given a shape image I (such as those plotted in Fig. 1a–Fig. 1d), let I_F be the set of its foreground pixels, and $\Gamma =$

∂I_F be the boundaries. The signed distance transform of I is denoted as $DT(I)$ and defined by

$$DT(I(x,y)) = \begin{cases} -d((x,y),\Gamma), & \text{if } (x,y) \in I_F - \Gamma, \\ 0, & \text{if } (x,y) \in \Gamma, \\ d((x,y),\Gamma), & \text{if } (x,y) \notin I_F, \end{cases} \tag{1}$$

where $d((x,y),\Gamma)$ is the Euclidean distance from pixel (x,y) to its nearest pixel in Γ. Clearly, the DT transform of a shape encodes both its local and global information in that foreground and background pixels are simultaneously evaluated. Examples of shape images after the signed distance transform are illustrated in Fig. 1e–Fig. 1h. Note that the formulation in (1) is different from the 2-D level-set representation, of which the level-set signed distance is defined with respect to a unique zero level set.

3. CLASSIFICATION BY BOOSTED FILTERING

To use the representation (1) for shape recognition, we first investigate a fundamental case of dealing with two-class data, denoted as $\mathcal{D} = D^A \cup D^B$. Intuitively, one could just try to experiment with most of the well-known similarity measures, e.g., shape context [2], *earth mover's distance* (EMD) [7], and *Chamfer distance* [8], and then use *nearest neighbor* criterion to accomplish the task of shape recognition. Nevertheless, evaluating such similarity measures are often slow, and the situation is further deteriorated by the search of the nearest neighbors. Our approach therefore focuses on efficiently performing shape recognitions, where its effectiveness is resulting from a boosted filtering framework using the DT transform.

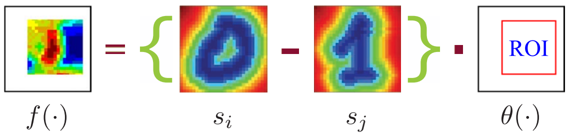


Fig. 2. (a) A filter f is specified by three parameters, including a pair of samples, say s_i and s_j, and their common ROI, θ. Thus, the filtering applying to a sample s is denoted as $f(s) = (s_i - s_j) \cdot \theta(s)$.

3.1. Filter Bank for Two-Class Data

Let $F_{\mathcal{D}} = \{f_1, f_2, ..., f_m\}$ be a filter bank of size m over the training data $\mathcal{D}$. And each filter $f \in F_{\mathcal{D}}$ is uniquely defined by three parameters, including a pair of shapes from different classes, and an area of *region of interest* (ROI). More precisely, let $s_i^A \in D^A$, $s_j^B \in D^B$ and θ be some ROI. Since we choose to limit the ROIs into square regions, we could simply write $\theta = (center, length)$ to specify the location and the size of an ROI. Consequently, we can represent a filter f by a *particle* of three parameters, and write out the correspondence as $f \Leftrightarrow (s_i^A, s_j^B, \theta)$. (See Fig. 2 for illustration.) In constructing $F_{\mathcal{D}}$, we randomly sample m particles from the parameter space. Notice that to ensure a more uniformly distributed sampling over principal modes of shapes in the same class, we have used *K-means* to cluster each class of shapes into mixtures. Finally, filtering an input sample, s, with f is simply the pixel-wise inner product between $s_i^A - s_j^B$ and $\theta(s)$, i.e.,

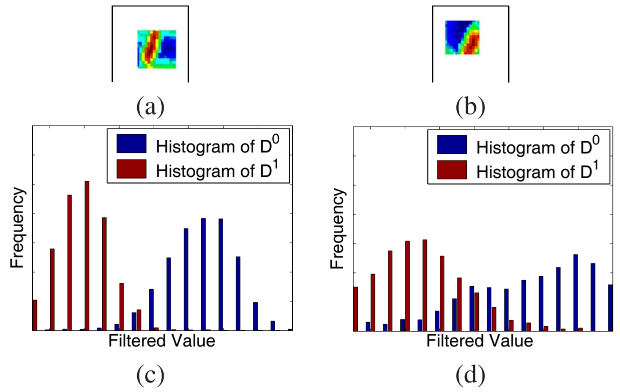


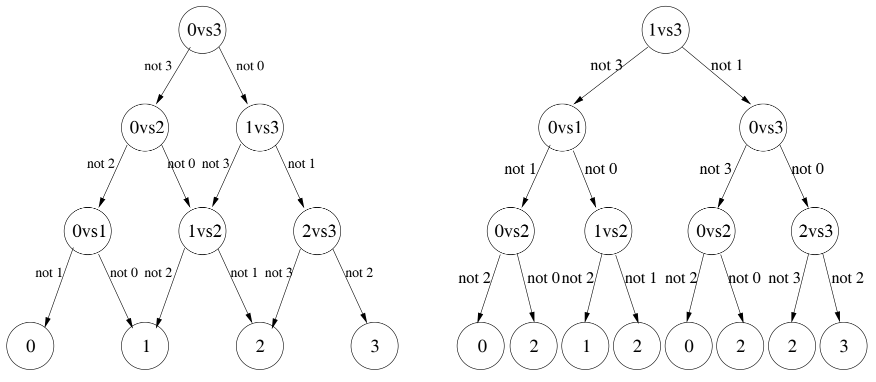
Fig. 3. $\mathcal{D} = D^0 \cup D^1$, where D^0 and D^1 contain various shapes of digits 0 and 1, respectively. (a), (b) The two filters selected in the first two iterations of BHBoost [9]. Since each filter (weak learner) yields a pair of weighted histograms for the two-class data, we also show the two corresponding pairs of weighted histograms in (c) and (d).

$$f(s; s_i^A, s_j^B, \theta) = (s_i^A - s_j^B) \cdot \theta(s), \tag{2}$$

where $\theta(s)(x,y) = s(x,y)$ if pixel (x,y) is in the ROI of θ, and 0, otherwise. Thus f is a real-valued filter that, depending on the location and the size of the ROI, the outcome of applying f to a sample s emphasizes either the local or the global correlations of s with the underlying parameters.

3.2. Discriminating Filter Selection via Boosting

In practice, the total number of filters in a typical $F_{\mathcal{D}}$ is quite large, and their discriminating power could vary significantly. This is indeed a critical issue if one intends to build an efficient classifier based on as few filters as possible. Take, for example, the illustrations in Fig. 3, the data $\mathcal{D} = D^0 \cup D^1$ have around $6,000$ images of digit 0 and another $6,000$ images of digit 1. (Each image is of size 28×28.) Even with random sampling, the filter bank $F_{\mathcal{D}}$ in this example still has around $30,000$ filters. We therefore adopt the BHBoost [9] for filter selection. Since each filter f projects a sample s into some real value, it will cause a pair of distributions for the two-class data D^0 and D^1.



(a) 4-class DAG (b) 4-class Low-Cost-First DAG

Digit	**0**	**1**	**2**	**3**
0	N/A	8	34	28
1	N/A	N/A	20	5
2	N/A	N/A	N/A	87
3	N/A	N/A	N/A	N/A

(c) Table of costs.

Fig. 4. Since the cost of each 2-class classifier can be computed in training, a DAG can be improved into an LCF-DAG.

Hence selecting a filter/weak learner to distinguish the two classes can be reduced to seeking an f that its pair of data distributions have a small Bhattacharyya coefficient. With BHBoost, we can conveniently obtain a classifier using only a few filters to achieve the required recognition accuracy. (Only 20 filters are needed for the example in Fig. 3.)

4. CLASSIFICATION ARCHITECTURE

Typical schemes to combine two-class classifiers for multi-class recognition include, e.g., one-against-one, one-against-all, *ECOC* (Error Correcting Output Code), and *DAG* (Directed Acyclic Graph). Among these classification architectures, DAG often yields higher efficiency with comparable accuracy, as reported in [10]. For an n-class recognition problem, a DAG is an n-level rooted tree, and contains i nodes in its ith level. Thus, a classification procedure for a test sample will be along some path from the root to one of the leaves. Moreover, at any of the $n-1$ internal nodes (a two-class classifier) of such a path, one class candidate is eliminated so that the leaf node would indicate the recognition result. An $n = 4$ example is given in Fig. 4a.

Often, the evaluation costs of these two-class classifiers may vary significantly. For boosting, the complexity of each classifier mainly depends on the number of weak learners. In our implementation for digit recognition, classifying any of the following pairs, $\{(2,7),(3,5),(3,8),(4,9),(7,9)\}$, is at least *five times* computationally more expensive than classifying each of $\{(0,1),(1,3),(1,8),(2,5),(6,9)\}$. On the other hand, classifications over a DAG inherit a property that the closer an internal node is to the root, the more frequently it will be visited. Taking account of these factors, we formulate a new classification architecture, called *LCF-DAG* (Lowest Cost First DAG), such that internal nodes of lower computational costs are positioned near the root. See Algorithm 1 for details about constructing an LCF-DAG.

Algorithm 1: LCF-DAG(C) for digit recognition

```
Input  : The set of digit classes, $C = \{0,1,...,9\}$;
         $cost(a,b),\ 0 \le a < b \le 9$;
Output : A $|C|$-level binary tree;
LCF-DAG( $C$ ) {
if $|C| = 1$ then
    Create a leaf node for this class;
else
    $(a^*,b^*) = argmin_{a,b \in C \wedge a<b}\, cost(a,b)$;
    Embed the classifier ($a^*$,$b^*$) in the node;
    Point to the right child: LCF-DAG($C - \{a^*\}$);
    Point to the left child: LCF-DAG($C - \{b^*\}$); }
```

Unlike the DAG having its nodes uniquely determined by the order of leaf nodes, the nodes in an LCF-DAG are arranged according to the costs of their corresponding two-class classifiers. In Fig. 4b, a four-class example of LCF-DAG is constructed with respect to the costs of classifiers listed in Fig. 4c. In both DAG and LCF-DAG, $n(n-1)/2$ classifiers are learned in training, and $n-1$ classifiers will be evaluated in testing an input sample. Clearly, the LCF-DAG implementation is more efficient in testing.

5. EXPERIMENTAL RESULTS

We test the proposed method with the application of handwritten digit recognition on a P4 3GHz PC. For the sake of comparison, the MNIST benchmark database is used (available at `http://yann.lecun.com/exdb/mnist/`) in evaluating our system. This database consists of $60,000$ training samples and $10,000$ testing samples. Each sample is centered in a 28×28 image by computing the mass of the pixels. In learning each two-class classifier, we randomly

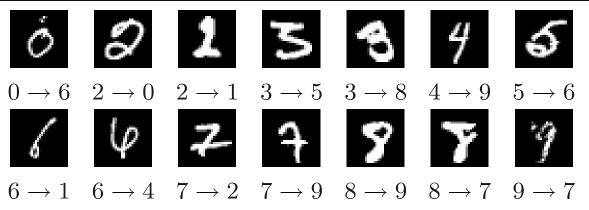


Fig. 5. Some of the misclassified test data.

sample $30,000$ filters into the filter bank. The ROIs of filters are squares with various scales, ranging from 4×4 to 28×28, and the number of bins used in each weak learner is set to 16. Then, the described boosting scheme over an LCF-DAG is used to construct the recognition system.

Table 1. Recognition results on MNIST data.

Class	0	1	2	3	4	5	6	7	8	9
0	**976**	0	0	*1*	0	0	*1*	*1*	*1*	0
1	0	**1131**	*2*	0	0	0	1	0	1	0
2	3	1	**1019**	*4*	1	0	0	2	2	0
3	0	0	1	**1002**	0	2	0	2	*3*	0
4	0	0	0	0	**974**	0	2	0	2	*4*
5	1	0	0	*4*	0	**883**	3	0	1	0
6	*4*	2	1	0	1	2	**947**	0	1	0
7	1	2	4	0	0	0	0	**1014**	2	*5*
8	2	0	2	1	1	3	0	1	**960**	*4*
9	2	1	1	2	*5*	1	0	*5*	2	**990**

Concerning the accuracy, the error rate yielded by our system is $1.04\% \,(104/10000)$. The result is superior to many well-known methods, such as linear classifier, RBF network, nearest neighbor classifier, and polynomial classifier, is compatible to the Reduced Set SVM [11], and falls slightly behind the boosted LeNet-4 [1] (0.7%) and the shape context [2] (0.63%). The detailed outcomes are given in Table 1, where each row gives the classification results of the corresponding digit class. To illustrate, some of the misclassified samples are displayed in Fig. 5.

The efficiency gain is even more impressive. Owing to the high convergence rate of BHBoost, the resulting classifiers often need fewer weak learners to achieve satisfactory recognition rates. For instance, the average numbers of weak learners used in the 45 two-class classifiers to attain error rates 2.37%, 1.53%, and 1.04% are 10.6, 34.3, and 87.3, respectively. In addition, to see the advantage of using an LCF-DAG, our system is compared with one-against-one and DAG (leaf nodes are arranged in an ascending order), given the same classifiers and testing data. The quantitative results are listed in Table 2. While delivering almost the same accuracy rate, LCF-DAG is 1.83 times faster than DAG, and 8.42 times faster than one-against-one.

Table 2. Comparison: *One-against-One*, *DAG*, *LCF-DAG*.

Architecture	Error Rate	CPU Time (10,000 testings)
One-against-One	1.02%	$8.42t$
DAG	1.08%	$1.83t$
LCF-DAG	1.04%	$t (= 7.7 secs)$

6. CONCLUSION

We have described a new approach for 2-D shape recognition, and demonstrated with an example of hand-written digit classification. The proposed image representation provides richer information, and is nicely coupled with the boosting algorithm to effectively capture useful features. With an LCF-DAG, our system can efficiently accomplish the task of shape recognition. This work is supported in part by grants 93-2213-E-001-018 and 93-EC-17-A-02-S1-032.

7. REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. PAMI*, vol. 24, no. 4, pp. 509–522, April 2002.

[3] L. Lee, G. Dalley, and K. Tieu, "Learning pedestrian models for silhouette refinement," in *Proc. Int'l Conf. on Computer Vision*, 2003, vol. 1, pp. 663–670.

[4] T.B. Sebastian, P.N. Klein, and B.B. Kimia, "Shock-based indexing into large shape databases," in *Proc. Euro. Conf. on Computer Vision*, 2002, vol. 3, pp. 731–746.

[5] P.F. Felzenszwalb, "Representation and detection of deformable shapes," in *Proc. Int'l Conf. on Computer Vision and Pattern Recognition*, 2003, pp. I: 102–108.

[6] D. Geiger, T.-L. Liu, and R.V. Kohn, "Representation and self-similarity of shapes," *IEEE Trans. PAMI*, vol. 25, no. 1, pp. 86–99, January 2003.

[7] Y. Rubner, C. Tomasi, and L.J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int'l J. Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[8] D.M. Gavrila and V. Philomin, "Real-time object detection for smart vehicles," in *Proc. Int'l Conf. on Computer Vision*, 1999, pp. 87–93.

[9] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh, "Fast object detection with occlusions," in *Proc. Euro. Conf. on Computer Vision*, 2004, vol. 1, pp. 402–413.

[10] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems*, 2000.

[11] C.J.C. Burges and B. Schölkopf, "Improving the accuracy and speed of support vector machines," in *Advances in Neural Information Processing Systems*, 1997.