View-Based 3D Object Recognition Using SNoW

Ming-Hsuan Yang Narendra Ahuja Dan Roth
Department of Computer Science and Beckman Institute
University of Illinois at Urbana-Champaign, Urbana, IL 61801
mhyang@vison.ai.uiuc.edu ahuja@vision.ai.uiuc.edu danr@cs.uiuc.edu

ABSTRACT

This paper describes a novel view-based algorithm for 3D object recognition using a network of linear units. The SNoW learning architecture is a sparse network of linear functions over a pre-defined or incrementally learned feature space and is specifically tailored for learning in the presence of a very large number of features. We use the pixel-level representation in the experiments and compare the performance of SNoW with Support Vector Machines and nearest neighbor methods on 3D object recognition using the 100 objects in the Columbia Image Object Database (COIL-100). Experimental results show that SNoW-based method outperform SVM-based system in terms of recognition rate and the computational cost involved in learning. The empirical results also provide insight for practical and theoretical considerations on view-based methods for 3D object recognition.

Keywords: View-Based Object Recognition, Support Vector Machine, SNoW, Winnow

1 Introduction

View-based object recognition has attracted much attention in recent years. In contrast to methods that rely on predefined geometric (shape) models for recognition, the problem is posed as one of matching appearance of an object in a two-dimensional image against the compact models of objects' appearance learned from two-dimensional images in which the objects appear in different poses and illuminations. The appearance of an object is the combined effects of its shape, reflectance properties, pose, and the illumination in the scene. Although shape and reflectance are intrinsic properties that do not change for any rigid object, pose and illumination vary from one scene to another. The visual learning problem is viewed as one to learn a compact model of the object's appearance under different poses and illumination conditions.

Among the methods on view-based object recognition, parametric eigenspace [9] [10] and support vector machine approaches [13] have demonstrated excellent recognition results on the COIL-20 and COIL-100 databases. Although these systems can recognize objects in almost real-time, the computational cost involved in learning is extremely high. Consequently the methods in the literature use only small, often unspecified, different subsets of objects from the whole database for experiments, which makes it difficult to compare the results. Furthermore, the training sets used in these methods consist of images taken in nearby poses (usually 10° apart). This particular experimental setup, as we will show, makes the learning problem less challenging. It is of great interest to compare the performance of these methods when only a limited number of views of the objects are presented during training.

In this work, we propose a method that applies the SNoW (Sparse Network of Winnows) learning algorithm [15] to 3D object recognition and compare the performance of our method with the SVM and nearest neighbor methods. SNoW is a sparse network of linear functions that utilizes the Winnow update rule [7]. SNoW is specifically tailored to learning in domains in which the potential number of features taking part in decisions is very large, but may be unknown *a priori*. Some of the characteristics of this learning architecture are its sparsely connected units, the allocation of features and links in a data driven way, the decision mechanism and the utilization of an feature-efficient update rule. SNoW has been used successfully on a variety of large scale learning tasks in natural language processing [15] [4] and face detection [16].

This paper is organized as follows. We review the related view-based methods that learn to recognize 3D objects in Section 2. The motivation and the SNoW learning architecture are discussed in Section 3. Section 4 presents experimental results and compares the performance of the proposed method with the SVM and nearest neighbor methods. We conclude this paper with comments on these learning methods and future work in Section 5.

2 Related Work

A number of view-based schemes have been developed to recognize 3D objects. Poggio and Edelman [12] show that 3D objects can be recognized from the raw intensity values in 2D images, which we will call here as the pixel-based representation, using a network of generalized radial basis functions. They argue and demonstrate that full 3D structure of an object can be estimated if enough 2D views of the object are provided. Turk and Pentland [19] demonstrate that human faces can be represented and recognized by "eigenfaces." Representing a sample face image as a vector of pixel values, the eigenfaces are the eigenvectors associated with the largest eigenvalues which are computed from a covariance matrix of the sample vectors. The attractive feature of this work is that the eigenfaces can be learned from the sample images in pixel representation without any feature selec-

tion. The eigenspace approach has since been adopted in different vision tasks from face recognition to object tracking. Murase and Nayar [9] [10] develop a parametric eigenspace method to recognize 3D objects directly from their appearance. For each object of interest, a set of images in which the object appears in different poses is obtained as training examples. Next, the eigenvectors are computed from the covariance matrix of the training set. The set of images is projected to a low dimensional subspace, spanned by a subset of eigenvectors, in which the object is represented as a manifold. A compact parametric model is constructed by interpolating the points in the subspace. In recognition, the image of a test object is projected to the subspace and the object is recognized based on the manifold it lies on. Using a subset of the Columbia Object Image Library (COIL-100), they show that 3D objects can be recognized accurately from their appearances in real-time.

Support vector machines (SVM) have also been applied to 3D object recognition. Scholkopf [18] applies SVM to recognize 3D objects in which 3D CAD models, such as animals and chairs, are used for experiments. The results show great promise of SVM in visual learning. Pontil and Verri [13] also adopt SVM for 3D object recognition. They use the COIL-100 dataset for experiments in which the training sets consist of 36 images (one for every 10°) for each of the 32 objects and the test sets consist of the remaining 36 images for each object. For 20 experiments where the objects are randomly selected from the COIL-100, the system achieves perfect recognition rate. Most recently, Roobaert and Van Hulle [14] also use COIL-100 to train and compare the performance of SVMs with different input representations. They also show the performance of the SVMs when only a limited number of views of the objects are available in training. Similar to [13], they only use a subset of the COIL-100 dataset in the experiments. Also, both methods use *linear* SVMs to recognize objects using pixel-based representation.

Note that the abovementioned methods learn to recognize 3D objects from sample images of varying poses in pixel-based representation. There exist numerous methods that learn models of 3D objects or features such as outlines [11], geometric moments [2], and local feature descriptors [17]. The focus of this paper is to introduce a novel learning method for 3D object recognition using the raw intensity values in the 2D images.

3 Motivation and Approach

Several learning methods have been developed to recognize 3D objects from their appearances in varying poses. In this section, we first review the Sparse Network of Winnows (SNoW) learning algorithm. The SNoW learning architecture is a sparse network of linear functions over a pre-defined or incrementally learned feature space and is specifically tailored for learning in the presence of a very large number of features. We use the pixel-level representation in the experiments and compare the performance of SNoW with Support Vector Machines and nearest neighbor methods on 3D object recognition using the 100 objects in the Columbia Image Object Database (COIL-100). We then describe how we apply SNoW learning algorithm to 3D object recognition, followed by a discussion and comparison with the SVM algorithm.

3.1 The SNoW Architecture

The SNoW (Sparse Network of Winnows) learning architecture is a sparse network of linear units over a common pre-defined or incrementally learned feature space. Nodes in the input layer of the network represent simple relations over the input and are used as the input features. Each linear unit is called a *target node* and represents relations which are of interest over the input examples. SNoW can be used for two-class or multiple-class pattern recognition problem. In this paper, the reason for using SNoW as two-class classifier is to have a fair comparison with SVM (which is designed for a two-class pattern recognition problem). For a two-class (A and B) problem, two target nodes are used, one as a representation for pattern A and the other for an pattern B. Similarly, n targets are used for a n-class problem. Given a set of relations (i.e., *types* of features) that may be of interest in the input image, each input image is mapped into a set of features which are *active* (present) in it; this representation is presented to the input layer of SNoW and propagates to the target nodes. Features may take either binary values, just indicating the fact that the feature is active (present), or real values, reflecting its strength; in the current application, all features are binary. Target nodes are linked via weighted edges to (some of the) input features. Let $\mathcal{A}_t = \{i_1, \ldots, i_m\}$ be the set of features that are active in an example and are linked to the target node t. Then the linear unit is *active* if and only if

$$\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t$$

where w_i^t is the weight on the edge connecting the ith feature to the target node t, and θ_t is its threshold.

In the current application a single SNoW *unit* which includes two subnetworks, one for each of the targets, is used. For a two-class (A and B) problem, a given example is treated autonomously by each target subnetwork; that is, an image labeled as a pattern A is used as a positive example for the target A and as a negative example for the B target, and vice-versa. Similarly for a n class problem, an image labeled as a i is used as a positive example for target i and as a negative example for target j, $j = 1, \ldots, n, \ , j \neq i$, and vice versa. The learning policy is on-line and mistake-driven; several update rules can be used within SNoW. The most successful update rule, and the only one used in this work is a variant of Littlestone's Winnow update rule [7]; this is a multiplicative update rule tailored to the situation in which the set of input features is not known a priori, as in the infinite attribute model [1]. This mechanism is implemented via the sparse architecture of SNoW. That is, (1) input features are allocated in a data driven way – an input node for the feature i is allocated only if the feature i is active in the input image and (2) a link (i.e., a non-zero weight) exists between a target node t and a feature i if and only if i has been active in an image labeled t. Thus, the architecture also supports augmenting the feature types from external sources in a flexible way, an option we do not use in the current work.

The Winnow update rule has, in addition to the threshold θ_t at the target t, two update parameters: a *promotion* parameter $\alpha > 1$ and a *demotion* parameter $0 < \beta < 1$. These are used to update the current representation of the target t (the set of weights w_i^t) only when a mistake in prediction is made. Let $\mathcal{A}_t = \{i_1, \ldots, i_m\}$ be the set of active features that are linked to the target node t. If the algorithm predicts 0 (that is, $\sum_{i \in \mathcal{A}_t} w_i^t \leq \theta_t$) and the received label is 1, the active weights in the current example are *promoted* in a multiplicative fashion:

$$\forall i \in \mathcal{A}_t, w_i^t \leftarrow \alpha \cdot w_i^t.$$

If the algorithm predicts 1 ($\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t$) and the received label is 0, the active weights in the current example are *demoted*:

$$\forall i \in \mathcal{A}_t, w_i^t \leftarrow \beta \cdot w_i^t.$$

All other weights are unchanged. The key feature of the Winnow update rule is that the number of examples[1] it requires to learn a linear function grows linearly with the number of *relevant* features and only logarithmically with the total number of features. This property seems crucial in domains in which the number of potential features is vast, but a relatively small number of them is relevant (this does not mean that only a small number of them will be active, or have non-zero weights). Winnow is known to learn efficiently any linear threshold function and to be robust in the presence of various kinds of noise and in cases where no linear-threshold function can make perfect classification, and still maintain its abovementioned dependence on the number of total and relevant attributes [8] [6].

Once target subnetworks have been learned and the network is evaluated, a decision support mechanism is employed, which selects the dominant active target node in the SNoW unit via a winner-take-all mechanism to produce a final prediction. In general, but not in this work, units' output may be cached and processed along with the output of other SNoW units to produce a coherent output.

3.2 Learning 3D Objects with SNoW

In this paper, we apply the SNoW approach to 3D object recognition. Our method makes use of Boolean features that encode the positions and intensity values of pixels. Let the pixel at (x, y) of an image with width w and height h have intensity value $I(x, y)$ ($0 \leq I(x, y) \leq 255$). This information is encoded as a feature whose index is $256 \times (y \times w + x) + I(x, y)$. This representation ensures that different points in the $\{\texttt{position} \times \texttt{intensity}\}$ space are mapped to different features. (That is, the feature indexed $256 \times (y \times w + x) + I(x, y)$ is *active* if and only if the intensity in position (x, y) is $I(x, y)$.) In our experiments, the values for w and h are 32 since each object sample has been normalized to an image of 32×32 pixels. Note that although the number of potential features in our representation is $262,144$ ($32 \times 32 \times 256$), only 1024 of those are active (present) in each example, and it is plausible that many features will never be active. Since the algorithm's complexity depends on the number of active features in an example, rather than the total number of features, the sparseness also ensures efficiency.

We compare the performance of the proposed SNoW-based method with the SVM-based method in 3D object recognition. Since SVM is a two-class classifier, one way to recognize objects from multiple classes is to build a hyperplane for each pair of the objects. In the one-against-one scheme, an n-class pattern recognition problem requires $\frac{n(n-1)}{2}$ binary classifiers to be trained using SVM. To classify test data, pair-wise competition between all the machines is performed which is similar to a tennis tournament. The final winner determines the class of the test data. Alternatively, the one-against-the-rest scheme requires only n classifiers in that each classifier is trained with the samples belong to one class (with label 1) and the rest of the samples as the another class (with label -1). The advantage of this scheme is that the number of classifiers to be trained and stored is linear in the number of classes involved, as opposed to the other scheme which requires exponential number of classifiers. But the downside of this scheme is that some test samples may not be classified into a single class because the training samples are not partitioned into two well-defined homogeneous classes. The recognition rates of SVM and SNoW based methods shown in Table 2 are performed using the one-against-one scheme. In other words, we train $\binom{100}{2} = 4950$ classifiers for each method. Furthermore, every test sample needs to go through 99 (50+25+12+6+3+2+1) classifiers to determine which class it belongs to.

3.3 Support Vector Machine

Both SNoW and SVM have been proposed as new machine learning algorithms with theoretical derivations using the VC (Vapnik-Chervonenkis) theorem. Vapnik [21] uses the VC theory to show that SVM has an upper bound of the error on the unseen test patterns as a function of error on training and the VC dimension of the classifier. Littlestone [7] also uses VC theory and shows that there is an upper bound for the number mistakes before the Winnows learns a concept (i.e., converges). SVM and SNoW have also been applied to various problems in natural language processing to hand digit recognition. Recently, SVM-based methods have demonstrated good performance in 3D object recognition [18] [13] [14]. We compare the performance of SNoW and SVM algorithms for 3D object recognition. In this section, we briefly review SVM approach.

Support Vector Machine (SVM), developed by Vapnik and colleagues, is a learning method for pattern recognition and regression problems [21] [3]. One characteristic of SVM is that it aims to find an optimal hyperplane such that the expected recognition error for the unseen data points is minimized. According to the structural risk minimization inductive principle, a function that describes the training data well and belongs to a set of functions with lowest VC dimension will generalize well regardless of the dimensionality of the input space [3]. Based on this principle, the SVM is a systematic approach to find a linear function that belongs to a

[1] In the on-line setting [7] this is usually phrased in terms of a mistake-bound but is known to imply convergence in the PAC [20] [5] (Probably Approximately Correct) sense.

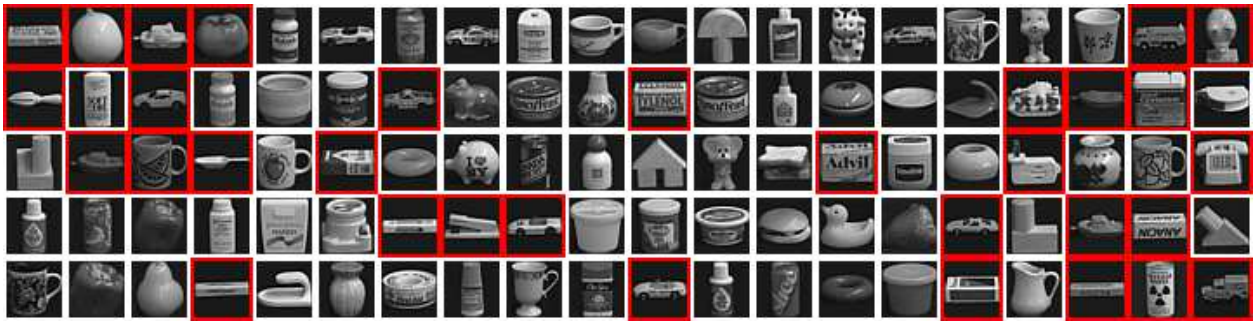


Figure 1. Columbia Object Image Library (COIL-100) consists of 100 objects of varying poses (5° apart). The objects are shown in row order where the highlighted ones are considered more difficult to recognize in [13].

set of functions with lowest VC dimension. The SVM also provides non-linear function approximations by mapping the input vectors into a high dimensional feature space where a linear hyperplane can be constructed. Although there is no guarantee that that a linear hyperplane will always exist in the high dimensional feature space, it is likely to find a linear separator (i.e., hyperplane in a SVM) in the projected space.

Given a set of samples $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_l, y_l)$ where $\mathbf{x}_i$ ($\mathbf{x}_i \in R^N$) is the input vector of N dimension and y_i is its label ($y_i \in \{-1, 1\}$) for regression problem, SVM aims to find a optimal hyperplane that leaves the largest possible fraction of data points of the same class on the same side while maximizes the distance of either class from the hyperplane (margin distance). Vapnik [21] shows that maximizing the margin distance is equivalent to minimizing the VC dimension in constructing an optimal hyperplane. The problem of finding the optimal hyperplane is thus posed as a constrained optimization problem in SVM and solved using the quadratic programming techniques. The optimal hyperplane, which determines the class of a data point $\mathbf{x}$, is in the form

$$f(\mathbf{x}) = sgn(\sum_{i=1}^{l} y_i \alpha_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b)$$

where $k(\cdot, \cdot)$ is a kernel function and sgn is a threshold function (with threshold value = 0) to determine the label of an input vector. Constructing an optimal hyperplane is equivalent to determining nonzero α_i. Any vector $\mathbf{x}_i$ that corresponds to a nonzero α_i is a *supported vector* (SV) of the optimal hyperplane. One feature of SVM is that the number of support vectors is usually small, thereby producing a compact classifier.

For a linear SVM, the kernel function is just the simple dot product of vectors in the input space while the kernel function in a nonlinear SVM projects the samples to a feature space of higher (possibly infinite) dimensions via a nonlinear mapping function:

$$\psi : R^N \rightarrow F^M, \ M \gg N$$

and construct a hyperplane in F^M. The motivation is that it is more likely to find a linear function, as done in linear SVM, in the high dimensional feature space. Using Mercer theorem, the expensive calculations in projecting samples into high dimensional feature space can be reduced significantly by using a suitable function k such that

$$k(\mathbf{x}, \mathbf{x}_i) = \psi(\mathbf{x}) \cdot \psi(\mathbf{x}_i)$$

where ψ is a nonlinear projection function. Several kernel functions, such as polynomial functions and radial basis functions, have been shown to satisfy Mercer theorem and been used in nonlinear SVM. By using different kernel functions, the SV algorithm can construct a variety of learning machines, some of which coincide with classical architectures. However, this also results in a drawback since one needs to find the "right" kernel function in using nonlinear SVM. SVM is also more prone to outliers in the data than mistake bound methods.

4 Experiments

We use the COIL-100 dataset to test our method and compare its performance with other view-based methods in the literature. In this section, we first describe the characteristics of the COIL-100 dataset and then show and compare the performance of several methods in numerous experiments, followed by comments on the empirical results.

4.1 Dataset and Experimental Setups

We use the Columbia Object Image Library (COIL-100) database for experiments (available at http: // www.cs.columbia.edu/ CAVE/ coil-100.html). The COIL-100 dataset consists contains color images of 100 objects where the images of the objects were taken at pose intervals of 5°, i.e., 72 poses per object. The images were also normalized such that the larger of the two object dimensions (height and width) fits the image size of 128×128 pixels. Figure 1 shows the images of the 100 objects taken in frontal view, i.e., zero pose angle. The 32 highlighted objects in Figure 1 are considered more difficult to recognize in [13] and are also used in our experiments. Each color image is converted to a gray-scale image of 32×32 pixels for our experiments.

We approach the recognition problem by considering each of the pixel values in a sample image as a coordinate in a high dimensional space (i.e., the image space). Consequently, each training and testing sample consists of 1,024 components. In this paper, we assume that the illumination conditions remain constant and hence object pose is the only

variable of interest (which is true for the objects in the COIL-100 dataset).

4.2 Ground Truth of the COIL-100 Dataset

At first glance, it seems difficult to recognize the objects in the COIL dataset because it consists of a large number of objects with varying pose, texture, shape and size. Since each object has 72 images of different poses (5° apart), many view-based recognition methods use 36 of them for training (10° apart) and the remaining images for testing. However, we conjecture that this particular recognition problem (36 poses for training and the rest 36 poses for testing) is not difficult because of dense sampling. In other words, the data points of the same object are close to each other in the image space (where each data point represents an image of an object in a certain pose). Consequently, it is easier to interpolate the data points in the subspace [9] and to construct an optimal hyperplane to separate the points of two classes [13] [14]. Our experiments with a simple nearest neighbor classifier (using Euclidean distance) on this problem show that the recognition rate is 98.50% (54 errors out of 3600 tests). Although it is well known that nearest neighbor method requires a lot of memory for storing templates and requires a lot of template matching in recognition, the recognition rate of this simple method is comparable to the complex SVM approaches [13] [14]. This experiment shows that the abovementioned recognition problem is not difficult and therefore not appropriate for comparison among different methods. Figure 2 shows some of the mismatched objects by nearest neighbor method.

It is interesting to see that the pair of the objects in the mistakes made by the nearest neighbor classifiers have similar geometric configurations and similar poses. A close inspection shows that most of the recognition errors are made between the three packs of chewing gums, bottles and cars. Consequently, the set of selected objects in an experiment has direct effects on the recognition rate. However, most methods in the literature use only a subset of the 100 objects (typically 20 to 30) from the COIL dataset for experiments. Table 1 shows the recognition rates of nearest neighbor classifiers in several experiments in which 36 poses of each object are used for templates and the remaining 36 poses are used for tests.

Table 1. Recognition rates of nearest neighbor classifier

	30 objects randomly selected from COIL	32 objects shown in Figure 1 selected by [13]	The whole 100 objects in COIL
Errors/Tests	14/1080	46/1152	54/3600
Recognition rate	98.70%	96.00%	98.50%

The experimental results using 30 and 32 objects reveal that the recognition problem when a large number of views of the objects are present during training is not as difficult as it seemed to be. Thus, we perform experiments in which the number of views of objects are limited to compare the performance of these learning methods.

4.3 Empirical Results

Table 2 shows the recognition rates of SNoW-based method, SVM-based method (using linear dot product for the kernel function), and nearest neighbor classifier using the COIL-100 dataset. We vary the number of views of an object (n) during training and use the rest of the views ($72 - n$) of an object for testing.

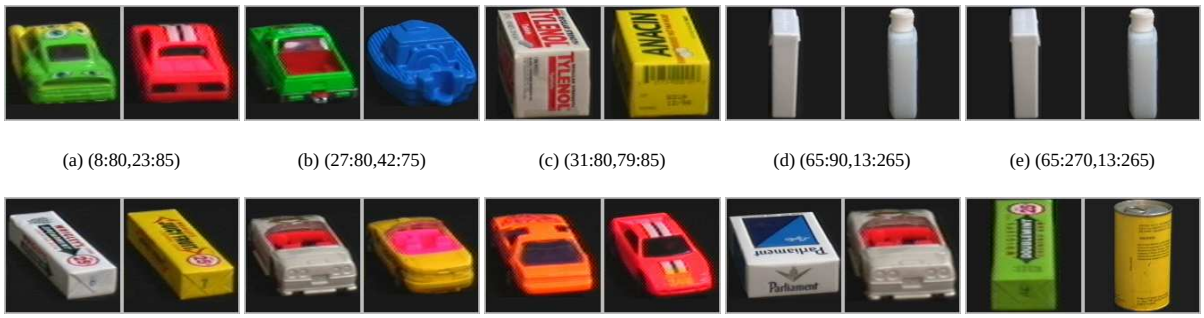
Table 2. Experimental results of three classifiers using the 100 objects in the COIL-100 dataset

	view/object				
	36	18	8	4	2
	3600 tests	5400 tests	6400 tests	6800 tests	7000 tests
SNoW	95.81%	92.31%	85.13%	81.46%	75.49%
Linear SVM	96.03%	91.30%	84.80%	78.50%	72.81%
Nearest Neighbor	98.50%	87.54%	79.52%	74.63%	65.31%

The experimental results show that SNoW-based method performs as well as SVM-based method when many views of the objects are present during training and outperforms SVM-based method when the numbers of views are limited. On the other hand, the time to train a classifier using SNoW is about 86% of that to train one using SVM. Although it is not surprising to see that the recognition rate decreases as the number of views available during training decreases, it is worth noticing that both SNoW and SVM algorithms are capable of recognizing 3D objects in the COIL-100 dataset with satisfactory performance if enough number of views (e.g., > 18) are provided. Also, even if only a limited number of views (e.g., 8 and 4) are used for training, the performance of both methods degrades gracefully. This indicates that both SNoW and SVM algorithms are robust and perform well even when class size increases.

The idea of finding support vectors for the optimal hyperplane in SVM-based method has an analogue to learning the "relevant" features when the "irrelevant" features abound. In other words, SVM aims to determine the necessary input vectors such that an optimal hyperplane can be constructed while reducing the risk to make a mistake in the future (i.e., structural risk minimization). The essence of the SNoW algorithm is to identify the relevant features to construct a network of linear units based only on the relevant attributes. Also, the number of mistakes made by this method grows only logarithmically with the number of irrelevant attributes in the examples. In the SVM-based methods, the ratio of the support vectors over the number of input vectors is roughly 27.78% (20 out of 72) while the ratio of the relevant features over the number of possible features is about 5.27% (13,805 out of 262,144). Caution should be observed in analyzing these numbers since the number of possible features is usually much higher than the number of input vectors. These ratios show that both methods are capable of determining what is necessary in recognizing the objects effectively.

Since the one-against-one scheme requires an exponential number of classifiers to be trained and stored, it is of great interest to compare the performance with the alternative one-against-the-rest scheme. Table 3 shows that al-



(a) (8:80,23:85) (b) (27:80,42:75) (c) (31:80,79:85) (d) (65:90,13:265) (e) (65:270,13:265)

(f) (67:70,84:75) (g) (69:80,91:75) (h) (76:80,23:255) (i) (96:260,69:85) (j) (98:270,99:195)

Figure 2. Mismatched objects by nearest neighbor method, where $(x : a, y : b)$ means that object x with view angle a is recognized as object y with view angle b. It shows some of the 54 errors (out of 3,600 test samples) made by the nearest neighbor classifier.

though the recognition rates of the SNoW-based classifiers trained with the alternative scheme are not as good as the counterpart, the recognition rates do not decrease drastically.

Table 3. Recognition rates of SNoW using two learning paradigms

SNoW	view/object				
	36	18	8	4	2
one-against-one	95.81%	92.31%	85.13%	81.46%	75.49%
one-against-the-rest	90.52%	84.50%	81.85%	76.00%	72.6%

5 Discussion and Conclusion

We have described a novel view-based method to recognize 3D objects using SNoW. Empirical results show that the SNoW-based method outperforms other methods in terms of recognition rates except for one case (i.e., 36 views). Furthermore, the computational cost to train a classifier using SNoW is less when using SVM.

For a fair comparison among different methods, this paper uses pixel-based presentation in the experiments. Although the current work achieves good recognition rates on the COIL-100 dataset when each image is represented as a vector of pixel values, it is of great interest to use local features as input vectors during training and testing. Feature extraction alleviates the burden of classifiers in that they do not need to learn models from raw inputs. This is particularly important in dealing with complicated objects in large datasets. Local features such as geometric moments at multiple scales seem feasible and attractive since they encode the objects in terms of more complex descriptors, yielding a more compact representation. Our future work will use this representation to recognize 3D objects.

References

[1] BLUM, A. Learning boolean functions in an infinite attribute space. *Machine Learning 9*, 4 (1992), 373–386.

[2] BREGLER, C., AND MALIK, J. Learning appearance based models: Mixtures of second moment experts. In *Advances in Neural Information Processing Systems 9* (1997), pp. 845–851.

[3] CORTES, C., AND VAPNIK, V. Support vector networks. *Machine Learning 20* (1995).

[4] GOLDING, A. R., AND ROTH, D. A winnow based approach to context-sensitive spelling correction. *Machine Learning 34* (1999), 107–130. Special Issue on Machine Learning and Natural Language.

[5] HELMBOLD, D., AND WARMUTH, M. K. On weak learning. *Journal of Computer and System Sciences 50*, 3 (June 1995), 551–573.

[6] KIVINEN, J., AND WARMUTH, M. K. Exponentiated gradient versus gradient descent for linear predictors. In *Proceedings of the Annual ACM Symposium on the Theory of Computing* (1995).

[7] LITTLESTONE, N. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning 2* (1988), 285–318.

[8] LITTLESTONE, N. Redundant noisy attributes, attribute errors, and linear threshold learning using winnow. In *Proceedings of the fourth Annual Workshop on Computational Learning Theory* (1991), pp. 147–156.

[9] MURASE, H., AND NAYAR, S. K. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision 14* (1995), 5–24.

[10] NAYAR, S. K., NENE, S. A., AND MURASE, H. Real-time 100 object recognition system. In *Proceedings of IEEE International Conference on Robotics and Automation* (1996).

[11] PENTLAND, A., MOGHADDAM, B., STARNER, T., OLIGIDE, O., AND TURK, M. View-based and modular eigenspaces for face recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1994), pp. 84–91.

[12] POGGIO, T., AND EDELMAN, S. A network that learns to recognize 3d objects. *Nature 343* (1990), 263–266.

[13] PONTIL, M., AND VERRI, A. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 6 (1998), 637–646.

[14] ROOBAERT, D., AND HULLE, M. V. View-based 3d object recognition with support vector machines. In *IEEE International Workshop on Neural Networks for Signal Processing* (1999).

[15] ROTH, D. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (1998), pp. 806–813.

[16] ROTH, D., YANG, M.-H., AND AHUJA, N. A snow-based face detector. In *Advances in Neural Information Processing Systems 12* (1999). To appear.

[17] SCHIELE, B., AND CROWLEY, J. Transinformation for active object recognition. In *Proceedings of the IEEE International Conference on Computer Vision* (1998).

[18] SCHÖLKOPF, B. *Support Vector Learning*. PhD thesis, Informatik der Technischen Universitat Berlin, 1997.

[19] TURK, M., AND PENTLAND, A. Eigenfaces for recognition. *Journal of Cognitive Neuroscience 3*, 1 (1991), 71–86.

[20] VALIANT, L. G. A theory of the learnable. *Commun. ACM 27*, 11 (Nov. 1984), 1134–1142.

[21] VAPNIK, V. *The Nature of Statistical Learning Theory*. Springer, 1995.