Appearance-Based Recognition Using Perceptual Components

XiuWen Liu
Department of Computer Science
P.O. Box 4530
Florida State University
Tallahassee, FL 32306-4530
liux@cs.fsu.edu

DeLiang Wang
Department of Computer & Information Science
Center for Cognitive Science
The Ohio State University
Columbus, OH 43210-1277
dwang@cis.ohio-state.edu

Abstract

A fundamental problem with appearance-based recognition is how to encode the perceptual similarity between images as images need to be grouped based on their perceptual similarity. In this paper, we employ a spectral histogram model for generic appearance-based recognition. A perceptual component is defined as the spectral histogram of a training image, which encodes all the images perceptually similar to the input image. The similarity between two perceptual components is measured as χ^2 distance between the corresponding spectral histograms, which has been shown to be perceptually meaningful. Building on this representation, we use the nearest neighbor classifier to classify an unseen input image, where each object class is represented by the perceptual components of the training images. A distinctive advantage of our representation is that it can be applied to many recognition problems, including texture classification, face recognition, and 3D object recognition.

1 Introduction

As the learning algorithms for neural network models have been the focus in the neural network literature, representation has recently been realized as the fundamental challenge for neural modeling [4, 16, 2]. Bishop stated that the choice of pre-processing and feature extraction is "one of the most significant factors in determining the performance of the final system"(p. 295, [2]). In this paper, we focus on problems in visual recognition. For visual recognition and classification, the essential problem is to have a perceptually meaningful representation. This meaningful representation, in theory, can be learned through a huge amount of data. In practice, a more effective way is to incorporate generic and perceptually meaningful prior knowledge. Konen and von der Malsburg[8] proposed a system that learns to classify mirror symmetric patterns from single examples based on a general *a priori* principle.

In this paper, we focus on a representation based on local geometrical and photometric features observing that a perceptually meaningful representation should be translation invariant. This leads to the histograms of filter responses. We demonstrate here that this representation is very effective for appearance-based classification and recognition, the goal of which is to classify images based on the perceptual similarity of their appearances.

In the texture modeling context, histograms of filter responses have been widely studied. Julesz [7] was among the first to propose that only the density of texture elements has perceptual significance and their positions are ignored. Computationally Heeger and Bergen [5] proposed a texture synthesis algorithm by matching the histograms of the observed image. Refer to Liu and Wang [11] for more complete references in texture modeling and discrimination.

Section 2 motivates the representation from a feature extraction perspective and introduces our perceptual component model. Section 3 describes the component pruning and filter selection with the perceptual component model. Section 4 presents our experimental results on texture classification and face recognition. 3D object recognition within this framework is addressed in a companion paper[10]. Section 5 concludes the paper with discussion on a number of issues.

2 Perceptual Components as Histograms

The starting point of our representation is based on the following observation. Suppose that we have a large number of different objects which may appear on a uniform background. We do not know the position of the object and we do not want to perform segmentation as the segmentation depends on the contrast between the object and the background. We further simplify that the objects are rigid and the sizes are fixed. We are interested in deriving a bottom-up and generic feature statistic that will be sufficient to discriminate a large number of objects.

This assumption leads to the use of histograms as the position must be integrated out as a random variable. However, the direct histogram of the images may not be sufficient as the different objects can have the same histogram. To improve the discriminative capability, one can use the statistics of local features to impose constraints on the images. This gives rise to a representation that uses the histograms of filter responses to characterize images and objects. We have named this representation the spectral histogram model [11]. A spectral histogram encodes all the images that are perceptually similar and thus we call it a perceptual component in the context of recognition and classification.

Natural images, however, can be distorted and deformed. Even for the images that belong to the same type, their appearance can be significantly different due to those distortions and deformations. To deal with the distortions and deformations, we represent each object as several perceptual components computed from training images with different distortions and deformations which bears some conceptual similarity to the mixture of experts[6].

Given this perceptual component representation, in this paper we use the nearest neighbor classifier to classify or recognize new input images. It is clear that other classifiers can also be used. The distance between components is defined as the χ^2 distance between their corresponding spectral histograms [11].

3 Component Pruning and Filter Selection

Within our perceptual component representation, it would be computationally desirable to keep only the components that are necessary for recognition as some of the training images may lead to the same perceptual components. We propose a component pruning algorithm based on the following observation. A training image is not critical if it is covered by a perceptual component of the same object. We implement this idea by computing the pair-wise distance between all the training components and we then prune the components which are much closer to other components of the same object than the components of different objects. To do this, we define relative inhomogeneity as

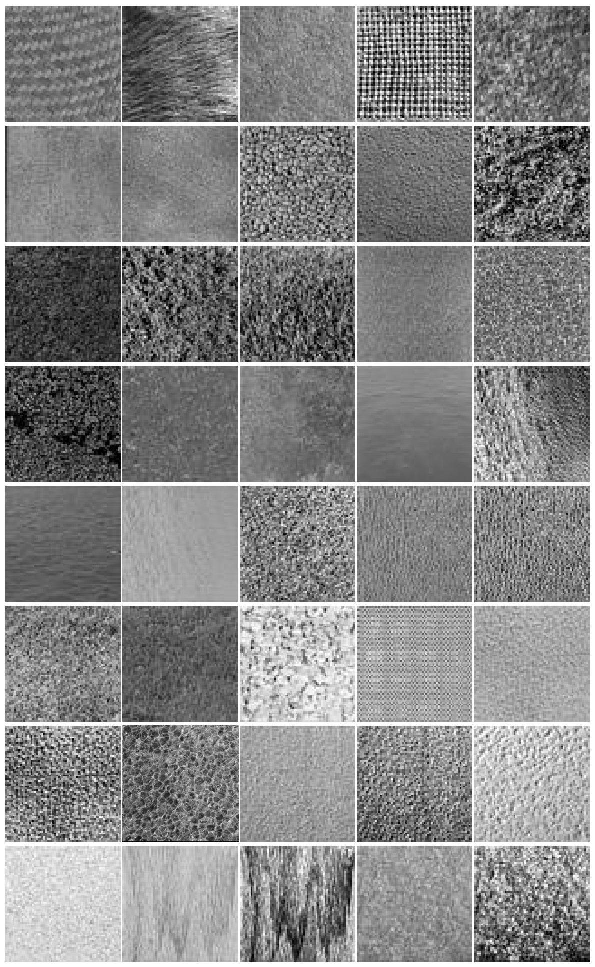


Figure 1: Forty terrain textures used in the classification experiments. The input image size is 256×256. These images are available at http://www-dbv.cs.uni-bonn.de/image/texture.tar.gz.

the ratio between the smallest distance within the same object to the smallest distance of different objects. The pruning algorithm stops when either the ratio is above a threshold or the number of components is acceptable.

Each dataset has its own prominent local characteristics and these local features can be best characterized by different filters. Note that the most discriminative filter is the one that gives the smallest average within-class and between-class ratio, as used in Fisher discriminant analysis[3]. Our algorithm chooses the best single first. Then we choose the next filter which works the best with the ones that have been chosen. The algorithm stops when the ratio does not improve much or a specified number of filters has been chosen.

Note also that both the component pruning and filter selection algorithms are greedy algorithms. While they are computationally efficient, the optimal solution is not guaranteed. Another point that is worth mentioning is that both algorithms only involve training images. This is based on the assumption that test images would be perceptually similar to the training images.

4 Experimental Results

Within our perceptual component representation framework, each object class is represented by the corresponding spectral histograms of the training images. Given a new input image, first its spectral histogram is calculated. Then the nearest neighbor classifier is used to do classification/recognition. However, other classifiers such as a neural network can also be used for classification. This paper focuses on texture and other 2D image classification and the results for 3D object recognition is presented in a companion paper[10].

4.1 Texture Classification

We apply our method to a dataset of 40 real texture images, shown in Fig. 1. This dataset contains different kinds of natural textures. Also some of textures are similar to others in the dataset, making the classification very challenging.

To do the texture classification, we partition each texture image into non-overlapping patches with size 32×32. We then choose a specified number of patches as the training set and the rest are used for testing. To avoid the bias of the training set choice on the classification performance, we randomly choose a given number of images as training and run our method many times. Table 1 shows the classification result for 100 trials with different number of training components per texture type. As one can see from this result, with as few as 4 training components, our method achieves a correct classification rate of 90%. Also the performance of our method does not depend much on the number of training examples. Note that a few training examples are needed for each texture type due to the inhomogeneity of the textures in the dataset and boundary condition for textures with large patterns compared to the patch size. Recently, Randen and Husoy[14] conducted a comprehensive comparative study on filter-based as well as other approaches for texture classification. Our result is significantly better than all the methods included in their study.

We also study filter selection on this dataset. To do that, we use 40 filters initially to calculate the spectral histograms. Then we apply our filter selection algorithm. Table 2 shows the performance of the first few

Table 1: Classification error for the 40-texture dataset shown in Fig. 1

# of Components	Average error rate	Best error rate	Worst error rate
1	21.01 %	15.60 %	27.74 %
2	13.49 %	10.08 %	18.02 %
4	8.32 %	6.08 %	10.79 %
6	6.23 %	4.47 %	8.71 %
8	5.11 %	3.97 %	6.83 %
12	4.03 %	3.08 %	5.77%
16	3.42 %	2.50 %	4.74 %
24	2.76 %	1.88 %	3.63 %
32	2.40 %	1.72 %	3.36%

Table 2: Classification error of different filters of the 40-texture dataset shown in Fig. 1

# of filters	First one correct error	First two correct error
1	22.73 %	8.36 %
2	11.17 %	8.36 %
3	5.31 %	1.18 %
5	2.97 %	0.78 %
11	2.81 %	0.39 %
40	6.80 %	1.48 %
8 (No intensity)	3.44 %	0.78 %

filters that our algorithm has chosen. Here we report the recognition error rate when the first one is correct and the error rate when one of the first two closest textures is correct. It is interesting to see that the performance with all 40 filters is worse than the one with only three filters chosen by our algorithm. The first filter chosen by our algorithm is the intensity filter, which gives the best performance among all the 40 filters as a single filter. However, in cases where the illumination changes significantly, derivative filters may be more appropriate. To do that, we exclude the intensity filter and our algorithm then chose 8 derivative filters and the performance of which is shown in the last row of Table 2. Another interesting point of our filter selection algorithm is that it chooses most of Gabor filters at a scale that is comparable with the structures in the images. Along the filter selection result for face recognition, it seems that our filter selection is able to choose the most effective filters specific to the dataset.

We apply our component pruning algorithm to this texture dataset to reduce the number of perceptual components for the nearest neighbor classifier. In our algo-

Table 3: Classification error of different number components of the 40-texture dataset shown in Fig. 1

# of components on average	First one correct error	First two correct error
1	29.53 %	15.70 %
2	17.27 %	7.11 %
3.2	8.67 %	2.42 %
6.4	3.67 %	1.41 %
32	3.44 %	0.78 %
4.3 (1.0)	7.19 %	1.72 %
5.9 (0.8)	3.90 %	0.78 %
10.98 (0.5)	3.44 %	1.10 %

rithm, one can specify how many components to keep or specify a threshold for the within-class and between-class ratio. The results of both methods are shown in Table 3. The first half of the table shows how the classification performance changes with the number of components. As we can see from the result, the performance does not change much from 32 components to only about 6 components. The second half, which is based on the within-class and between-class ratio, is more interesting, where the threshold is specified in the parenthesis. This gives a choice of balancing number of components and the performance. When the threshold is 0.5, the performance does not decrease at all even the number of components per texture type reduces to 11 from 32. Threshold 1.0 indicates how many components are actually needed to represent the given dataset. If the components are further reduced, the performance can decrease dramatically as shown in the first half of the table.

To show the effectiveness of the component pruning algorithm, Fig. 2 shows the number of components used for each texture type, where on average 5.9 components are used for each type. The algorithm chooses more components for texture types that are similar to other textures in the dataset or texture types that are inhomogeneous. Fig. 3 shows two enlarged textures from the textures shown in Fig. 1. The texture shown in Fig. 3(a) is homogeneous and distinctive, and thus only 1 component is needed for perfect classification. The texture shown in Fig. 3(b) is perceptually very similar to two other textures in the dataset and 19 components are used for this texture. Even with 19 components, some of the patches were still misclassified as the other two textures. This is essentially the intrinsic ambiguity of the dataset. Also the number of components needed indicates how difficult the texture is to be classified correctly.

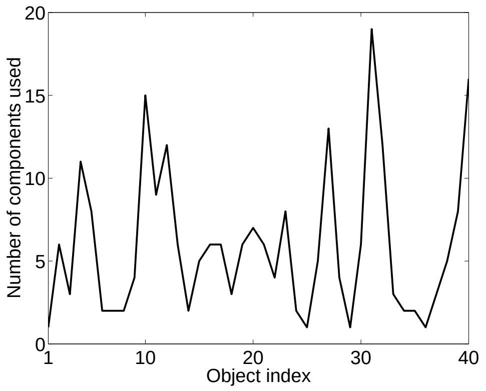


Figure 2: The number of components for each texture type of the 40 texture dataset.

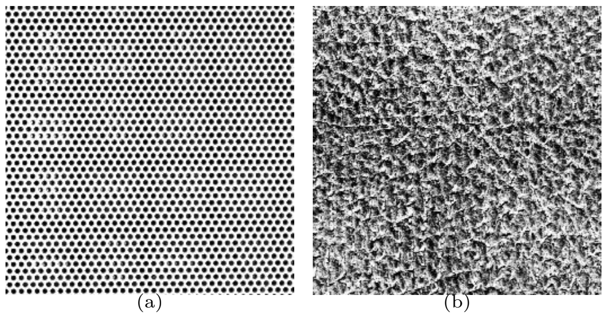


Figure 3: Textures with the least and most components. (a) 1 component is used for this texture. (b) 19 components are used for this texture.

A potential disadvantage of our representation is that it requires convolutions of many filters, which is computationally expensive, especially when the filters are large. While the computation can be reduced significantly through component pruning and filter selection and through dedicated hardware for convolution, there exists an interesting trade-off between performance and computation cost in our framework. We can estimate the spectral histogram from filter responses at a subset of pixels. For example, the computation time to classify all 320 patches was reduced from 30.5 seconds to 7.11 seconds while the performance was reduced from 96.56 % to 91.82 % for the first one to be correct and only from 98.91 % to 98.05 % for the first two to be correct.

4.2 Face Recognition

We have also applied our method to ORL[1], a standard face dataset. The dataset consists of faces of 40 different subjects with 10 images for each subject. The images were taken at different times with different lighting conditions on a dark background. While only limited side

[1]http://www.uk.research.att.com/facedatabase.html

movement and tilt were allowed, there was no restriction on facial expression. All the subjects are shown in Fig. 4(a) and all the 10 images of a particular subject are shown in Fig. 4(b) to demonstrate the variations of facial expression and lighting condition.

The procedure is the same for texture classification. Here we use 7 filters to calculate the spectral histograms. We vary the number of the training faces per subject and the remaining images are used for testing. As we did for texture classification, we randomly choose the training images from the dataset to avoid potential bias due to the choice of training faces. The result is shown in Table 4. Compared to the texture classification, the variation of the performance is more significant, especially the difference between the best and worst. This is because that all the faces are topographically similar and the filters we use are small compared to the size of faces. Our average performance is comparable with the best available result in the literature[13].

Table 4: Recognition error for ORL face database

# of training faces	Average error rate	Best error rate	Worst error rate
1	30.15 %	25.28%	36.67 %
2	16.01 %	7.81 %	22.50%
3	8.04 %	2.86 %	17.14%
4	3.88 %	0.42 %	9.58 %
5	2.06 %	0.00 %	5.50%

It is interesting to see the filters chosen by our filter selection algorithm for this face dataset. Again we use 40 filters initially to calculate the spectral histogram and then apply the filter selection algorithm to choose filters among the 40 filters. The algorithm chose first the Gabor filters with the largest scale, indicating that our filter selection chooses the most effective filters for this dataset. A recent psychophysical study [1] showed that human face recognition is different from general human object recognition in that the global configuration of faces is critical for human recognition performance. However, our spectral histogram representation provides a unified framework for face recognition and general human object recognition. While our representation can account for both cases, the difference between them is the choice of the most effective filters. As an extreme case, we can use the eigenfaces [15], the eigenvectors associated with the biggest eigenvalues as filters. Despite the demanding computation, our representation can include eigen approaches for face recognition as a special case.

4.3 Hand Written Digit Recognition

We have also applied our method to a subset of MNIST[2], a standard hand-written digit dataset. The dataset has been widely used to test and compare different methods for digit recognition. We have achieved a correct recognition of 93 %, which is not as good as the available performance[12]. We include this result to show a limitation of our representation. While the spectral histogram representation encodes local geometrical and photometric characteristics of images, this example suggests it is not very effective to encode global shapes as the histogram integrates out the position variable. This limitation could be overcome by using position-sensitive features within the filtering scheme. One solution would be to use the response vectors as features[9]. The integration of histograms with other features will be investigated in the future.

5 Discussion

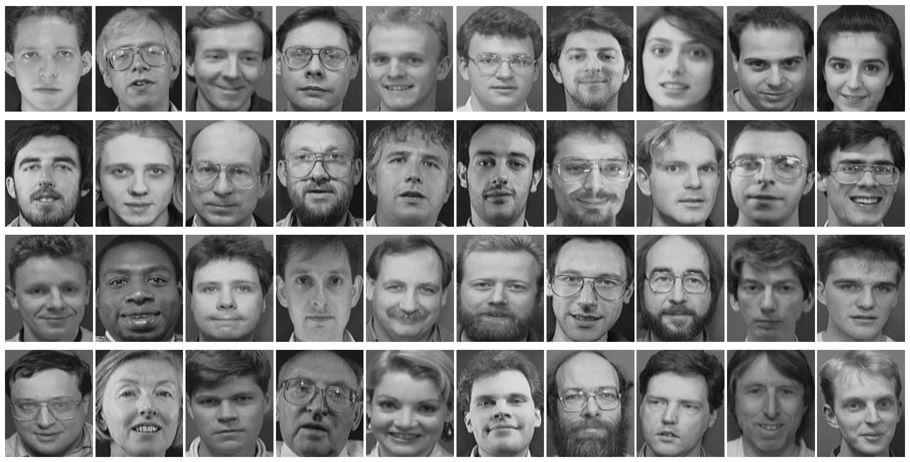
In this paper, we propose a general framework for appearance based classification and recognition. Our representation is based on the assumption that the positions of objects are not known and thus histograms of local features are most suitable. Within this framework, we have also studied filter selection and component pruning with the nearest neighbor classifier. We have applied our method to texture classification, face recognition, and 3D object recognition and have obtained very good results.

Besides the good performance on real image datasets, our method is also shown to be consistent with a systematic psychophysical data on texture perception[11]. Also our representation is biologically plausible in that the system could be implemented neurally. A more thorough discussion is given in [11].

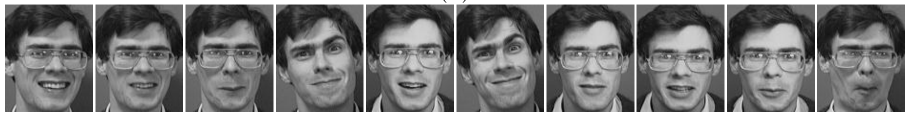
While our representation is very effective for many classification and recognition tasks, the lack of positional information makes it less effective for global shape representation. The integration of histograms with other positional features might provide a solution to the general recognition problem by combining local photometric, local geometric, and global shape information.

Acknowledgments XWL was supported in part by an NIMA grant NMA201-01-2010 and DLW was supported in part by an NSF grant (IIS-0081058) and an AFOSR grant (F49620-01-1-0027).

[2]http://www.research.att.com/~yann/ocr/mnist/index.html



(a)



(b)

Figure 4: ORL face database. The size of the images is 112×92. (a) 40 subjects in the database. (b) 10 face images of one subject taken at different facial expression and illumination conditions.

References

[1] I. Biederman and P. Kalocsai, “Neurocomputational bases of object and face recognition,” *Philosophical Transactions of the Royal Society London: Biological Sciences*, vol. 352, pp. 1203-1219, 1997.

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, UK, 1995.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd Ed., John Wiley & Sons, 2001.

[4] S. Geman and E. Bienenstock, “Neural networks and the bias/variance dilemma,” *Neural Computations*, vol. 4, pp. 1-58, 1992.

[5] D. J. Heeger and J. R. Bergen, “Pyramid-based texture analysis/synthesis,” in *Proceedings of SIGGRAPHS*, pp. 229–238, 1995.

[6] M. I. Jordan and R. A. Jacobs, “Hierarchical mixtures of experts and the EM algorithm,” *Neural Computation*, vol. 6, pp. 181–214, 1994.

[7] B. Julesz, “Textons, the elements of texture perception, and their interactions,” *Nature*, vol. 290, pp. 91–97, 1981.

[8] W. Konen and C. von der Malsburg, “Learning to generalize from single examples in the dynamic link architecture,” *Neural Computation*, vol. 5, pp. 719-735, 1993.

[9] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Wurtz, and W. Konen, “Distortion invariant object recognition in the dynamic link architecture,” *IEEE Transactions on Computers*, vol. 42, pp. 300-311, 1993.

[10] X. W. Liu and A. Srivastava, “3D object recognition using perceptual components,” in *Proceedings of the International Joint Conference on Neural Networks*, in press, 2001.

[11] X. W. Liu and D. L. Wang, “A spectral histogram model for textons and texture discrimination,” in *Proceedings of the International Joint Conference on Neural Networks*, in press, 2001.

[12] G. Mayraz and G. E. Hinton, “Recognizing handwritten digits using hierarchical products of experts,” in *Advances in Neural Information Processing Systems*, T. K. Leen, T. Dietterich, and V. Tresp, editors, vol. 13, pp. 953-959, 2001.

[13] “Special issue on face recognition,” *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 19(7), 1997.

[14] T. Randen and J. H. Husoy, “Filtering for texture classification: A comparative study,” *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 21(4), pp. 291–310, 1999.

[15] M. Turk and A. Pentland, “Eigenfaces for recognition,” *Journal of Cognitive Neuroscience*, vol. 3, pp. 71-86, 1991.

[16] D. L. Wang, “Pattern recognition: Neural networks in perspective,” *IEEE Expert*, vol. 8, pp. 52–60, 1993.