Appeared in the Proceedings of the 13th International Conference on Pattern Recognition, Vienna, 1996

Algorithms for Pattern Rejection *

Simon Baker and Shree K. Nayar

Department of Computer Science, Columbia University, New York, USA

Abstract

The efficiency of pattern recognition is particularly crucial in two situations; whenever there are a large number of classes to discriminate, and, whenever recognition must be performed a large number of times. We develop a number of algorithms to cope with the demands of these difficult conditions. The algorithms achieve high efficiency by using pattern rejectors. A pattern rejector is a generalization of a classifier that quickly eliminates a large fraction of the candidate classes or inputs. After applying a rejector, the recognition algorithms can concentrate their computational efforts on verifying the small number of remaining possibilities. The generality of our algorithms is established through a close relationship with the Karhunen-Loéve expansion. We experimented on two representative applications, namely, object recognition and feature detection. The results demonstrate substantial efficiency improvements over existing approaches, most notably Fisher's discriminant analysis.

1 Introduction

The efficiency of a pattern recognition algorithm becomes increasingly important as the number of pattern classes grows. Object recognition using appearance matching [Murase and Nayar 95] is one example application where the computational dependence upon the number of classes (objects) is the key to a real time solution. High efficiency also proves critical whenever the recognition algorithm must be applied a large number of times. This is the case in local feature detection [Nayar et al. 96], where the feature detector must be applied at every pixel in an image.

We develop efficient pattern recognition algorithms to deal with both of the scenarios described above. The algorithms achieve high performance by using *pattern rejectors* [Baker and Nayar 96]. A rejector is an algorithm that very quickly eliminates a large fraction of the candidate classes (i.e. objects in recognition) or inputs (i.e. local image brightness values in feature detection). The theory of pattern rejection, as developed in [Baker and Nayar 96], first defines the notion of a rejector and then shows how a collection of simple rejectors can be combined to yield a much more effective *composite rejector*. By analyzing the performance of composite rejectors, a number of design criteria were derived for the individual component rejectors that go to form an effective composite rejector.

In this paper we propose a collection of general-purpose algorithms for the implementation of simple rejectors satisfying the design criteria derived in [Baker and Nayar 96]. Using our algorithms in the individual rejectors of a composite rejector then allows the construction of efficient pattern recognition algorithms. The derivation of our algorithms is based upon a single assumption about the underlying pattern classes, namely, the *class assumption*. The generality of the class assumption (and hence the algorithms) is established through a close connection with the Karhunen-Loéve (K-L) expansion [Fukunaga 90].

We demonstrate the performance of our algorithms by experimenting on two applications: object recognition using appearance matching [Murase and Nayar 95] and local feature detection [Nayar et al. 96]. We first construct a composite rejector for a widely-used image database of 20 objects. Each object appears in a large number of poses and constitutes a single pattern class. The final composite rejector is able to completely (and without error) discriminate between all 20 objects with an efficiency that is a significant improvement over currently used techniques. We compare the composite rejector with Fisher's discriminant analysis and show our algorithm to be both substantially more efficient as well as more accurate. Next, we constructed a composite rejector for the task of feature detection. The result is a very efficient method of preprocessing an image to identify pixels that truly deserve the application of a full-fledged feature detector, such as the one proposed in [Nayar et al. 96].

2 Background: Pattern Rejection

We begin by briefly summarizing pattern rejection as developed in [Baker and Nayar 96]. After stating the assumptions and definitions, we present the design criteria for the individual component rejectors of a composite rejector.

2.1 Assumptions and Definitions

A pattern recognition problem is based upon a finite set of measurements of an underlying physical process. Hence we assume the existence of a *classification space*, $S = \Re^d$, where d is the number of measurements. Elements, $x \in S$, will be referred to as *measurement vectors* or for convenience *vectors*. Next, we assume the existence of a finite collection, $W_1, W_2, \ldots, W_n \subseteq S$ of *(pattern) classes*. The classes themselves are defined by the application in question and so we assume that they are given to us *a priori*.

Definition 1 *A* classifier (or recognizer) *is an algorithm, ϕ, that given an input, $x \in S$, returns the class label, i, for which $x \in W_i$.*

A *rejector* is a generalization of a classifier in the sense that it returns a set of class labels. This set must contain the correct class label of the input, but it may also contain others:

*This research was supported in parts by ARPA Contract DACA-76-92-C-007, by DOD/ONR MURI Grant N00014-95-1-0601, and by an NSF National Young Investigator Award.

Definition 2 *A* rejector *is an algorithm, ψ, that given an input, $x \in S$, returns a set of class labels, $\psi(x)$, such that $x \in W_i \Rightarrow i \in \psi(x)$ (or equivalently $i \notin \psi(x) \Rightarrow x \notin W_i$).*

The name rejector is derived from the equivalent definition; if i is not in the output of the rejector, we can safely reject the possibility that $x \in W_i$. We then introduce the *rejection domain* of W_i as the set of all $x \in S$ for which $i \notin \psi(x)$. That is, the rejection domain is the set of all x for which we can reject the hypothesis that $x \in W_i$:

Definition 3 *If ψ is a rejector and W_i is a class, then the* rejection domain, R_i^ψ, *of rejector, ψ, for class W_i is the set of all $x \in S$ for which $i \notin \psi(x)$.*

In follows from Definitions 3 & 2 than ψ is a rejector if and only if $\forall i\ R_i^\psi \cap W_i = \emptyset$. The effectiveness of a rejector is defined to be the expected fraction of classes that are not rejected by it. Therefore, a small numeric value of the effectiveness corresponds to an "effective" rejector:

Definition 4 *If ψ is a rejector, the* effectiveness *of ψ is* $\mathrm{Eff}(\psi) = \frac{1}{n} E[|\psi(x)|] = \frac{1}{n} \sum_{i=1}^{n} P[x \in R_i^\psi]$

Applying a rejector does not guarantee that we will be able to solve the pattern recognition problem uniquely; there may be more than one class in the output of the rejector. Any ambiguity is dealt with by adding a verification stage:

Definition 5 *A* verifier *for a class, W_i, is a boolean algorithm that, given an input, $x \in S$, returns the result $True$ if $x \in W_i$ and $False$ otherwise.*

We form a *rejection-based classifier* by first applying a rejector and then applying a verifier for each class with a label in the output of the rejector. Combining the results we can immediately classify the input. The overall efficiency of a rejection-based classifier depends upon both the efficiency and effectiveness of the rejector.

The output of a rejector is a subset of classes and so a smaller instance of the original classification problem. Recursively applying another rejector, tuned to the reduced subset of classes, may eliminate more of the classes as candidates and so improve the effectiveness of the combined rejectors. This is the notion of a *composite rejector:*

Definition 6 *A* composite rejector, Ψ, *is a collection of rejectors, $\Psi = \{\psi_\imath : \imath \in \Im\}$, where $\Im$ is an index set for Ψ, and such that: (a) there is a rejector in Ψ designed for the complete set of classes, and (b) for any rejector, $\psi_\imath \in \Psi$, and any $x \in S$, either $\psi_\imath(x) = 1$ or there is a rejector in Ψ designed for $\psi_\imath(x)$.*

A composite rejector has the structure of a directed acyclic graph. Each rejector, $\psi_\imath \in \Psi$, together with the subset of classes for which it was designed, corresponds to a node in the graph. Then, the application of the composite rejector to a novel measurement vector corresponds to a path through the graph. At each node in the path, the corresponding rejector is applied and its output determines the next rejector to apply and hence the edge that should be taken to leave the node.

2.2 Rejector Design Criteria

1. For a rejection-based classifier to be efficient overall, we require the composite rejector to be both *efficient* and *effective.*
2. To maximize the effectiveness of a composite rejector, we should design each component rejector to be as effective as possible. This is achieved by choosing the rejection domains to be as large as possible. However, there is a trade-off between maximizing the size of the rejection domains, ensuring $R_i^\psi \cap W_i = \emptyset$ for correctness, and using simple decision boundaries for high efficiency.
3. To avoid an exponential explosion in the size of the composite rejector the following design criteria should be adhered to: (a) avoid rejectors with large number of outputs, (b) balance rejector output cardinalities, and (c) minimize the overlap between rejector outputs.

3 A General-Purpose Rejection Technique

Before presenting our algorithms in detail, we first describe the general principle upon which they are based. In what follows, we will write the Euclidean inner (dot) product of two vectors as $\langle x, y \rangle$. The induced Euclidean norm we denote by $\|x\|_2 = \langle x, x \rangle^{1/2}$. We also assume that the norm of a vector does not effect classification and so restrict attention to the surface of the unit ball, $B = \{x \in S : \|x\|_2 = 1\}$.

3.1 The Class Assumption

Designing a rejector is equivalent to deciding upon the rejection domains. Further, for correctness we require that $R_i^\psi \cap W_i = \emptyset$. Therefore, the choice of the rejection domains must depend heavily on the nature of the underlying classes. In order to make progress we need to assume something about the classes:

The Class Assumption *For each class W_i, there exists a vector, $c_i \in S$, a linear subspace, $L_i \subseteq S$, and a threshold, $\delta_i \geq 0$, such that $\forall x \in W_i$, $dist(x, c_i + L_i) \leq \delta_i$. Further we assume: (a) $dim(L_i) \ll d$, and (b) $\delta_i \ll 1$.*

The class assumption (see Figure 1) is very general and allows various "shapes" of classes including disconnected multi-cluster distributions. All that is required is that each class be roughly low dimensional. In fact, the class assumption is approximately equivalent to assuming that the Karhunen-Loéve K-L expansion results in a compact and accurate representation of the class. Suppose that M_i^k is the subspace spanned by the k most important K-L eigenvectors, and λ_i are the decaying K-L eigenvalues, then we have:

$$E_{x \in W_i}(\mathrm{dist}(x, E_{y \in W_i}(y) + M_i^k)^2) = \sum_{s=k+1}^{d} \lambda_s \approx 0. \quad (1)$$

Setting $c_i = E_{x \in W_i}(x)$, and $L_i = M_i^k$, we see that the difference between the class assumption and the K-L expansion is one of expected versus maximum value. Then, the widespread use of the K-L expansion allows us to argue that the class assumption can be expected to hold extensively.

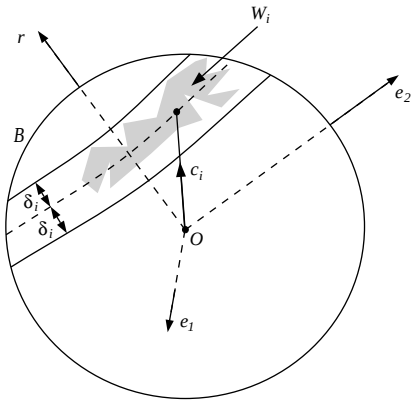

Figure 1: An illustration of the class assumption for a low dimensional example, $S = \Re^3$. The subspace, L_i, is the 2 dimensional subspace spanned by the vectors, $\{e_1, e_2\}$. Every vector in W_i can be approximated to within error, δ_i, by the linear combination of c_i and a vector in L_i.

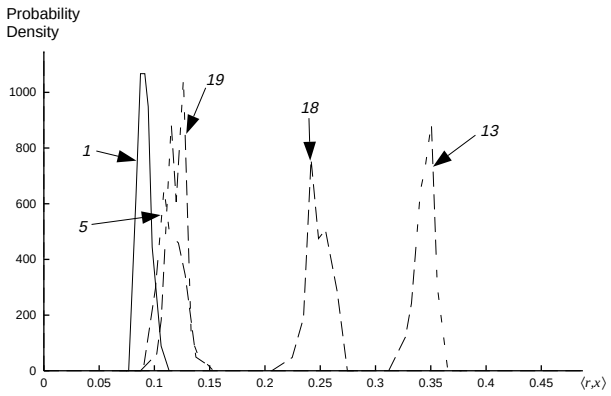

Figure 2: An illustration of equation (2) for 5 classes taken from the object recognition application in Section 5.1. On the abscissa we plot the projection of the measurement vector with the rejection vector. On the ordinate we plot an estimate of the probability that a vector from a particular class will yield that projection. We plot five lines, one for each of the classes. As can be seen, the distributions are peaked in accordance with the class assumption holding. We can use the projection with the rejection vector to discriminate between any pair of classes whose distributions do not overlap. For instance, if $\langle r, x\rangle = 0.25$ we can reject classes 1, 5, 13 & 19.

3.2 Rejection Vectors

Given that the class assumption holds, we now explain the basis of our algorithms. We begin by defining the notion of a *rejection vector*:

Definition 7 *Suppose the class assumption holds for the classes,* $W_1, W_2, \ldots, W_n$. *Then, a* rejection vector *is a unit vector,* $r \in B$, *for which* $r \perp \bigoplus_{i=1}^{n} L_i$.

If r is a rejection vector, it follows immediately from the class assumption, orthogonality, and the Cauchy-Schwarz inequality, that:

$$x \in W_i \;\Rightarrow\; |\langle r, x\rangle - \langle r, c_i\rangle| \leq \delta_i \qquad (2)$$

Equation (2) means that the rejection vector projects each class, W_i, onto approximately a single point, $\langle r, c_i\rangle$. So long as the points, $\langle r, c_i\rangle$, are well separated, not many of the intervals $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$ will overlap, and we can use equation (2) to discriminate between the classes.

In Figure 2 we illustrate equation (2) by plotting the projection, $\langle r, x\rangle$, against the likelihood of that projection occurring for a randomly selected measurement vector, x, of a fixed class. Equation (2) means that each class is projected onto almost a point, and so we expect to see very peaked distributions in the figure. Any pair of classes with distributions that do not overlap can be discriminated using this projection. In this particular example, we cannot discriminate between every pair of classes, but we can always reject at least 2 classes. For example, if $\langle r, x\rangle = 0.1$ we can only safely eliminate classes 13 and 18. In general there is no guarantee that we will be able to find a rejection vector that completely separates a given pair of classes. Note, however, that a rejector is only required to eliminate a large fraction of the classes, not necessarily every last one. In practice, we found that this technique allows us to reject sufficiently many classes to achieve a substantial efficiency improvement.

4 Algorithms for Pattern Rejection

To implement the technique just described, we must perform six tasks: (1) verify the class assumption, (2) select the rejection vector, (3) estimate the thresholds, (4) construct the component rejectors, (5) provide an algorithm with which to apply the component rejectors, and, (6) construct the composite rejector. We discuss each task in turn and present an algorithm to accomplish it. The algorithms assume that we have available a set of training samples: $\{y_i^1, y_i^2, \ldots, y_i^{m(i)} \in W_i : \; i = 1, \ldots, n\}$.

4.1 Verification of the Class Assumption

We use the K-L expansion to verify the class assumption and to find appropriate values for L_i and c_i. For each class, W_i, we put c_i to be the mean class vector, and set L_i to be the subspace spanned by the K-L eigenvectors with corresponding eigenvalues above a threshold, t:

Algorithm 1: Estimation of L_i and c_i

1. Set $c_i = \frac{1}{m(i)} \sum_{j=1}^{m(i)} y_i^j$
2. Compute the eigenvectors, e_i^j, and their corresponding eigenvalues, λ_i^j, of the covariance matrix, defined by: $M_i = \frac{1}{m(i)} \sum_{j=1}^{m(i)} y_i^j (y_i^j)^T$

   (If $m(i) \ll d$, the Singular Value Decomposition (SVD) should be used [Murakami and Kumar 82].)
3. Set L_i to be the subspace spanned by $\{e_i^j : \; \lambda_i^j > t\}$.

The choice of an appropriate value for the threshold, t, is application dependent. We suggest trying several different alternatives until an acceptable value is found. For many applications, a guideline figure would be one that results in the use of around 5-10 eigenvectors per class.

4.2 Choice of the Rejection Vector

The rejection vector is only constrained by Definition 7 to be a unit vector orthogonal to $\bigoplus_{i=1}^{n} L_i$. Therefore we

have a lot of freedom in its selection. Our aim should be to choose the rejection vector to maximize the effectiveness of the resulting rejector. Since an effective rejection vector will be one that widely distributes the centers of the projections of the classes, $\langle r, c_i \rangle$, we choose the rejection vector to maximize the spread of these points. If variance is used as the measure of spread, the optimal rejection vector is the first K-L eigenvector of the mean class vectors projected into the subspace $(\bigoplus_{i=1}^{n} L_i)^{\perp}$. This is the rejection vector that we use:

Algorithm 2: Choice of the Rejection Vector

1. Construct an orthonormal basis for $\bigoplus_{i=1}^{n} L_i$ by applying Gram-Schmidt orthonormalization to the basis vectors of the subspaces, L_i, computed in Algorithm 1. Suppose the result is $\{f_j : \ j = 1, \ldots, p\}$.
2. Project each mean class vector, c_i, into $(\bigoplus_{i=1}^{n} L_i)^{\perp}$ using: $c_i^{\perp} = c_i - \sum_{j=1}^{p} \langle c_i, f_j \rangle f_j$.
3. Apply the K-L expansion to $\{c_i^{\perp} : \ i = 1, \ldots, n\}$. Set the rejection vector, r, to be the (normalized) eigenvector with the largest eigenvalue.

4.3 Estimation of the Thresholds

The only property the thresholds must satisfy for the rejector to operate correctly is equation (2), which is redisplayed here:

$$x \in W_i \ \Rightarrow \ |\langle r, x \rangle - \langle r, c_i \rangle| \leq \delta_i \qquad (3)$$

However, the smaller the thresholds are the more effective the rejectors will be. Therefore we need to chose the thresholds carefully. There are various methods that could be used to do this. Here we present the method which we used in our object recognition example in Section 5.1. (Other methods, including the technique used for our feature detection experiments in Section 5.2, are provided in [Baker and Nayar 95].) It was found empirically (see Figure 2) that the projected class distributions for all the objects closely resemble normal distributions. We chose a confidence level of 99.9%, and set δ_i to be 3.5 times the standard deviation of the distribution:

Algorithm 3: Selection of the Thresholds

1. Set $\delta_i = 3.5 \times [\frac{1}{m(i)} \sum_{j=1}^{m(i)} |\langle r, y_i^j \rangle - \langle r, c_i \rangle|^2]^{1/2}$

4.4 Construction of the Rejector

One of the design criteria in Section 2.2 was that each component rejector should have a small number of outputs. We achieve this by partitioning $[-1, 1]$ into *buckets*, $b_1, \ldots, b_m$, where $\forall j, \ b_j = [\text{cut}_{j-1}, \text{cut}_j]$, $\text{cut}_0 = -1$, and $\text{cut}_m = 1$. (See Figure 3 for an illustration.) The bucket end-points, cut_j, are referred to as a *cut-point*. Once we have decided upon the buckets, we associate with each bucket a set of classes. The set of classes contains those for which the class projection intersects the bucket:

$$\text{classes}(b_j) = \{i : \ b_j \cap [\langle r, c_i \rangle - \delta_i, \langle r, c_i \rangle + \delta_i] \neq \emptyset\} \qquad (4)$$

It follows from equations (3) & (4) that:

$$x \in W_i \text{ and } \langle r, x \rangle \in b_j \ \Rightarrow \ i \in \text{classes}(b_j) \qquad (5)$$

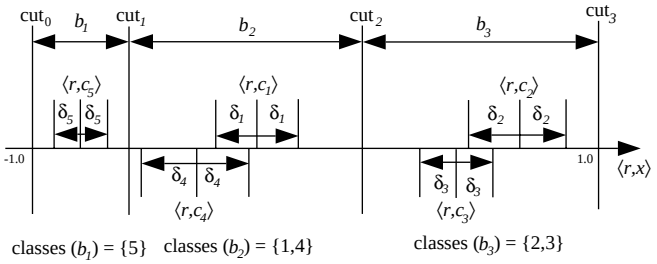

Figure 3: The interval $[-1, 1]$ is partitioned into buckets, separated by cut-points. Each bucket is associated with a set of classes; those whose projection intersects the bucket. The rejector is defined to return the set of classes of the bucket, into which the measurement vector is projected.

So long as equation (3) holds, defining $\psi(x) = \text{classes}(b_j)$, where $\langle r, x \rangle \in b_j$, is then a valid definition of a rejector; i.e. $x \in W_i \ \Rightarrow i \in \psi(x)$ is true.

The reason for introducing buckets is so that we may carefully select them to follow the design guidelines in Section 2.2. In the following algorithm, step 1(c) aims to minimize the intersection between the output subsets, and step 1(d) aims to maximize the balance between output subsets.

Algorithm 4: Construction of the Rejector

1. Select the set of cut-points, J:
   (a) Set, $J = \{-1, 1\}$, and, $M = \{\langle r, c_i \rangle - \delta_i : \ i = 1, 2, \ldots, n\} \cup \{\langle r, c_i \rangle + \delta_i : \ i = 1, 2, \ldots, n\}$.
   (b) Sort the set M, and for each consecutive pair of numbers in M, store their mean in the set M'.
   (c) For each point, $x \in M'$, in turn, insert x into J if and only if: $\forall i, \ \ x \notin [\langle r, c_i \rangle - \delta_i, \langle r, c_i \rangle + \delta_i]$
   (d) If $|J| = 2$, add to J the point in M' which maximizes: $\min(|\{i : \ y < \langle r, c_i \rangle - \delta_i\}|, |\{i : \ y > \langle r, c_i \rangle + \delta_i\}|)$
2. Create the buckets:
   (a) Sort the set of cut-points, J.
   (b) For each pair of neighboring points in J, $(\text{cut}_{j-1} < \text{cut}_j \in J)$, create a bucket, $b_j = [\text{cut}_{j-1}, \text{cut}_j]$.
   (c) For each b_j compute classes(b_j) using equation (4).
   (d) Store the buckets together with their associated classes, in an array in increasing cut-point order.

4.5 Application of the Rejector

The data stored by a rejector consists of two parts: the rejection vector, r, and the array computed in step 2(d) of Algorithm 4. Given a novel measurement vector, x, we apply the rejector using:

Algorithm 5: Application of the Rejector

1. Compute the projection, $\langle x, r \rangle$.
2. Perform a binary search on the array of buckets to find the bucket, b_j, containing $\langle x, r \rangle$.
3. Return classes(b_j).

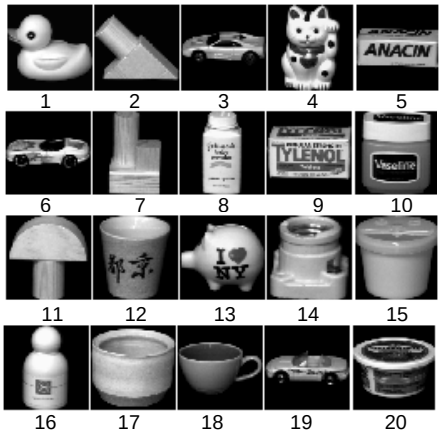

Figure 4: The 20 objects (classes) used in the recognition experiment. There are 72 images of each object with neighboring image pairs separated by 5° of pose. The data set is that used in [Murase and Nayar 95].

4.6 Construction of a Composite Rejector

Component rejectors may be reached by more than one path in the composite rejector. So that it can check for duplicates in step 2(b), we assume that Algorithm 6 maintains a hash table of the component rejectors it has constructed.

Algorithm 6: Construction of a Composite Rejector

1. Construct a simple rejector for the current set of nodes using Algorithms 1, 2, 3, & 4.
2. For each bucket, b_j, of the rejector just constructed:
   (a) If $|\text{classes}(b_j)| = 1$, or if no classes can be rejected (classes(b_j) equals the current set of classes) create a leaf node; i.e. a rejector which immediately returns classes(b_j).
   (b) If a component rejector has already been created for classes(b_j), just add a link to that rejector.
   (c) Otherwise recursively call Algorithm 6 with classes(b_j) as the current set of nodes. Then, add a link from the current rejector to the component rejector created in step 1 of the recursive call.

5 Example Applications

Our objective is to demonstrate the generality and efficiency of our rejection algorithms. As examples, we have chosen two problems in computational vision, namely, 3-D object recognition and feature detection. These problems were selected as they can, under certain assumptions, be cast as classical pattern recognition problems. Furthermore, both problems often need to be solved with high efficiency.

5.1 3D Object Recognition

We follow the appearance matching approach, first described in [Murase and Nayar 95]. Object recognition is reduced to pattern recognition by first segmenting the object and then scale normalizing it to an image of size 128×128 pixels. The image is then treated as a 16,384 dimensional measurement vector in the classification space, $S = \Re^{16,384}$, by reading the pixels in a raster scan fashion.

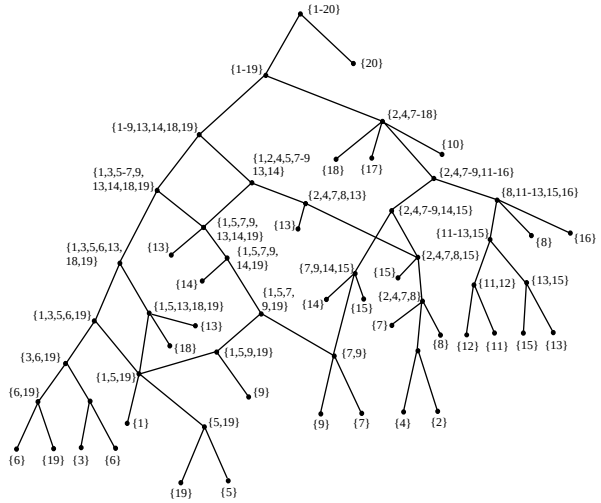

Figure 5: A representation of the composite rejector. Each interior node denotes a single rejector, and is labeled with the set of objects that it is designed to act on. At each node, only one dot product and a binary search need to be performed. (See Algorithm 5.) Each leaf denotes a possible output of the composite rejector.

Finally, the vector is intensity normalized to lie on the unit sphere, B.

The data set that we used (see Figure 4) consists of 20 objects (classes). It contains 72 images of each object separated by 5^o intervals of pose. The images were divided into a training set and a test set each comprising 36 images of every object. The training set is then treated as the samples of the classes and used to implement the composite rejector, a representation of which is presented in Figure 5. As it happens, every leaf of the composite rejector contains a single class, and hence the composite rejector can fully discriminate between the 20 objects. (We would have regarded the rejector as successful even if each leaf had contained 2-3 objects.)

We found that the composite rejector responded 100% correctly for both the training and test sets. Based on the assumption that each image in the data set is equally likely to appear, we calculated the average number of rejectors used in the composite rejector to be just 6.43. Since the time taken at each node is essentially the cost of one inner product (convolution), the efficiency compares very favorably with the results obtained by Murase and Nayar [Murase and Nayar 95]. Their implementation required 20 inner products, followed by a sophisticated search procedure.

Using the same image database, we compared the performance of the composite rejector against that of Fisher's discriminant analysis [Fisher 36]. Again, we followed the same test procedure, namely, setting aside half of the data, and using the other half to construct the classifier. We constructed Fisher spaces [Duda and Hart 73] of different dimensionality. In Fisher space the classes consist of tight clusters, which we modeled as multivariate normal distributions. We computed the mean and covariance matrix of each of these distributions. Then, each novel measurement vector was classified by finding its closest cluster, i.e. the

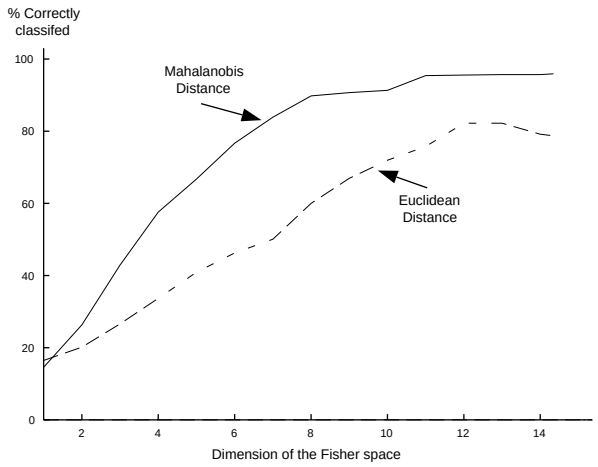

Figure 6: Results of applying Fisher's discriminant analysis to the data set in Figure 4. On the abscissa we plot the dimension of the Fisher space used, and on the ordinate the percentage of test images correctly classified. The peak performance is just under 97%, and to obtain this level of accuracy 11 discriminant vectors are needed. In contrast, the composite rejector gives perfect (100%) classification with just 6.43 rejection vectors.

cluster whose mean is closest to the vector. We used both the Mahalanobis and Euclidean distances.

Figure 6 shows the results for the combined performance on the training and test sets. Even for the Mahalanobis distance, the classification results are not perfect. In fact, after around dimension 11, the accuracy remains approximately constant and just below 97%. This compares poorly with the 100% classification obtained by the composite rejector, using an average of just 6.43 rejection vectors.

5.2 Local Feature Detection

We constructed a composite rejector for a feature detector of the type proposed in [Nayar et al. 96]. (The algorithms used in this case are slightly different to those presented in Section 4. The reason for the difference is that here there is only one pattern class corresponding to the one feature being detected. The details of the algorithms used may be found in [Baker and Nayar 95].) The output of the composite rejector is used as input to the feature detector, and consists of pixels at which further processing is deemed worthwhile. Although the technique is applicable to general parametric features, we only have space to display our results (see Figure 7) for edge detection.

6 Discussion

There is a relationship between our algorithm for choosing the rejection vector in Section 4.2 and Fisher's discriminant analysis [Duda and Hart 73]. In particular, Algorithm 2 tends to choose a vector that maximizes between-class scatter, while keeping within-class scatter fixed at a low level. The difference between our algorithm and discriminant analysis is that discriminant analysis is presented as a single level of processing. On the other hand, the composite rejector has a hierarchical structure, which leads to superior performance. In particular, the relative performance is accounted for by the fact that each rejector in the

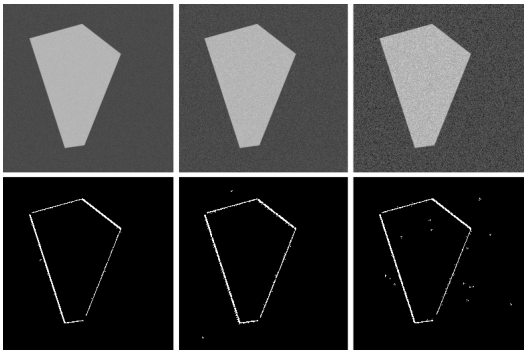

Figure 7: The edge rejector applied to 3 noisy synthetic images. The top row shows the noisy images whose pixels the rejector is applied to. The image on the left has added Gaussian noise of standard deviation 1 grey level, the middle image has noise of 2 grey levels, and the rightmost image has noise of 4 grey levels. The bottom row shows the output images produced by the edge rejector. Each output image consists of rejected pixels (marked black) and candidate pixels (marked white). In the least noisy image an average (computed over the whole image) of 1.61 rejectors were used. For the more noisy images, an average of 1.82 rejectors and 2.34 rejectors were used, respectively.

composite rejector is individually constructed for a subset of classes which is as small as possible. Since all the Fisher vectors are computed for the entire collection of classes, their discriminatory power is not as great.

References

[Baker and Nayar 96] S. Baker and S.K. Nayar, "Pattern Rejection," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* San Francisco, 1996.

[Baker and Nayar 95] S. Baker and S.K. Nayar, "A Theory of Pattern Rejection," *Columbia University Technical Report,* CUCS-013-95, 1995.

[Duda and Hart 73] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.

[Fisher 36] R.A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics,* 7:179–188, 1939.

[Fukunaga 90] K. Fukunaga, *Statistical Pattern Recognition*, Academic Press, 1990.

[Murakami and Kumar 82] H. Murakami and V. Kumar, "Efficient calculation of primary images from a set of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:511–515, 1982.

[Murase and Nayar 95] H. Murase and S.K. Nayar, "Visual Learning and Recognition of 3D Objects from Appearance," *International Journal of Computer Vision,* 14:5–24, 1995.

[Nayar et al. 96] S.K. Nayar, S. Baker, and H. Murase, "Parametric Feature Detection," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* San Francisco, 1996.