A Theory of Pattern Rejection

Simon Baker and Shree K. Nayar

Columbia Univeristy Technical Report
CUCS-013-95

Department of Computer Science
Columbia University
New York, NY 10027

Email: {simonb,nayar}@cs.columbia.edu
Tel. +1 (212) 939-7000, Fax +1 (212) 666-0140

Contents

Abstract

The efficiency of pattern recognition is critical when there are a large number of classes to be discriminated, or when the recognition algorithm must be applied a large number of times. We propose and analyze a general technique, namely pattern rejection, that leads to great efficiency improvements in both cases. Rejectors are introduced as algorithms that very quickly eliminate from further consideration, most of the classes or inputs (depending on the setting). Importantly, a number of rejectors may be combined to form a composite rejector, which performs far more effectively than any of its component rejectors. Composite rejectors are analyzed, and conditions derived which guarantee both efficiency and practicality. A general technique is proposed for the construction of rejectors, based on a single assumption about the pattern classes. The generality is shown through a close relationship with the Karhunen-Loéve expansion. Further, a comparison with Fisher's discriminant analysis is included to illustrate the benefits of pattern rejection. Composite rejectors were constructed for two applications, namely, object recognition and local feature detection. In both cases, a substantial improvement in efficiency over existing techniques is demonstrated.

Index Terms: Pattern recognition, computational efficiency, pattern rejection, composite rejector, object recognition, feature detection, edge detection, Fisher's discriminant analysis, Karhunen-Loéve expansion.

1 Introduction

We address the efficiency of pattern recognition, which is known to be vital when the number of classes involved is large. An example application in computational vision is object recognition, which in many cases can be reduced to a classical pattern recognition problem [Murase and Nayar 95]. Of particular importance in this context, is the growth rate of the recognition time as a function of the number of classes (objects). High efficiency also proves critical when the recognition algorithm must be applied a large number of times. This is the case in local feature detection [Nalwa 93] [Nayar et al. 95], where the detector needs to be applied at every pixel in an image.

We propose a general theory that results in substantial efficiency improvements in both of the above scenarios. The theory is based upon the central notion of a *rejector*. A rejector is an algorithm that efficiently eliminates from further consideration, most of the large number of classes (e.g. objects in recognition) or inputs (e.g. local image windows in feature detection). While the intuitive concept of a rejector is simple, its formalization is significant since it leads immediately to the following important observations and results that constitute the proposed theory:

1. The definition of correctness for a rejector is much less constraining than that for a classifier (recognizer). In particular, a rejector is only required to eliminate most of the classes or inputs most of the time, which is substantially less demanding than requiring perfect classification all of the time. As a result, rejectors can be constructed that are far more efficient than corresponding classifiers.

2. Although, in general, a rejector does not provide the final solution to the pattern recognition problem, it significantly reduces the number of possible classes or inputs to consider. Consequently, the recognizer can dedicate its computational resources to a much smaller number of candidates. In doing so, pattern rejection is taking advantage of the fact that the average case complexity of the recognition problem is generally far less than the worst case complexity. In both example applications mentioned above, namely object recognition and feature detection, this is the case.

3. Perhaps the most crucial aspect of pattern rejection is that, since a rejector eliminates a large number of classes (inputs), the task remaining after applying a rejector is a smaller instance of the original recognition problem. Hence, a collection of rejectors may be combined recursively in a directed acyclic graph structure to form a *composite rejector*. At each node of the composite rejector is a simple (as opposed to composite) rejector. Significantly, each such simple rejector may be individually designed for the set of classes (inputs) not eliminated by the parent rejector in the graph.

   Each application of the composite rejector corresponds to a path through the directed acyclic graph. At each node in the path, the associated simple rejector is applied thereby eliminating more of the classes (inputs). Since each subsequent rejector is constructed for a smaller subset of the classes (inputs), child rejectors are able to eliminate classes (inputs) that their predecessors where not able to. Overall, the recursive structure results in the composite rejector having much more discriminatory power than any of its component rejectors.

4. Another very important property of composite rejectors, is that it is possible to analyze their performance in terms of the performance of their components. For instance, we derive conditions that guarantee logarithmic time complexity of recognition, in terms of the total number of classes involved. We also analyze the preprocessing and space requirements of a composite rejector, in particular providing conditions that ensure practicality.

5. We propose a simple general purpose technique for constructing the component rejectors of a composite rejector. The technique is based on a single assumption about the nature of the pattern classes, namely, the *class assumption.* The generality of the class assumption is establishing by exhibiting a close relationship with the Karhunen-Loéve (K-L) expansion [Fukunaga 90] [Oja 83]. Hence, we expect the proposed rejection technique to be applied successfully in any application for which the K-L expansion is beneficial.

We demonstrate the significance of pattern rejection via experiments on applications in appearance matching based object recognition [Murase and Nayar 95] and feature detection [Nayar et al. 95]. First, we constructed a composite rejector for a widely used image database of 20 objects, each of which constitutes a pattern class. The appearance of each object changes considerably as the pose of the object varies. However, the composite rejector was able to completely, and without error, discriminate between all 20 objects. The efficiency is shown to be a substantial improvement over the technique used in [Murase and Nayar 95], which similarly achieved perfect recognition. We also empirically illustrate logarithmic growth in the time complexity of the composite rejector. Further, when compared with Fisher's discriminant analysis [Duda and Hart 73], the composite rejector is seen to be both significantly more efficient as well as more accurate. Discriminant analysis, even at its peak performance, has an error rate of slightly over 3%, in contrast to the error-free performance of the composite rejector. Finally, we constructed a composite rejector for the task of feature detection. This results in 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. 95].

The remainder of this paper is organized as follows. In Section 2 we discuss the relationship of pattern rejection to previous work. We proceed in Section 3 to introduce the notions of a rejector and of a composite rejector. We also analyze the time and space complexities of composite rejectors. In Section 4, we describe the construction of the individual rejectors that go to form a composite rejector. Section 5 presents our experimental results, and Section 6 concludes the paper with a brief discussion of this and future work.

2 Related Work

The recursive structure of the composite rejector constitutes a decision tree, or more generally a directed acyclic graph. A complete survey of work that use such a structure is well beyond the scope of this paper, but a small selection[1] is [Henrichon and Fu 69] [Payne and Meisel 77] [Weng 94]. In general, a composite rejector has a directed acyclic graph structure, as opposed to simply a tree structure, because there are many different orders in which classes may be rejected. Hence there may be a large number of different possible paths leading to any one node in the composite rejector. Connections can also be drawn between our results and the large body of work on computationally motivated nearest neighbor classifiers [Friedman et al. 77] [Bentley 80] [Yianilos 93]. Though the problem we address is somewhat similar, namely, efficient classification, our setting is more general.

The major novelty of our approach is the central role played by the pattern classes. It is this which leads to the composite rejectors having a directed acyclic graph structure, rather than a tree structure. Existing work which is concerned with complexity, either models the classes as collections of points, or studies partitions of space. Hence, our efficiency results are in terms of the number of classes, rather than the number of sample points, or the extent to which space is partitioned. We regard this class-centered approach a more natural model of the problem. Importantly, it also focuses attention on what we believe to be the key question: What properties must the pattern classes possess for recognition to be performed efficiently? The introduction of the class assumption is an attempt to answer this question, and to characterize what it means for a pattern class to have a "simple" rather than a "complex" decision boundary.

A relationship can be established between our technique for rejector construction and Fisher's discriminant analysis [Fisher 36] [Duda and Hart 73]. In particular, our rejection vector will be seen intuitively to maximize between-class scatter, while keeping within-class scatter fixed at a low level. The major differences between rejection theory and discriminant analysis are the following:

1. Discriminant analysis is presented as a single level of processing. On the other hand, a composite rejector has a hierarchical structure, which leads to superior performance. In particular, the relative performance is accounted for by the fact that child rejectors are individually constructed for reduced subsets of classes. Further, the second and subsequent Fisher vectors can be regarded as suboptimal when compared to the rejection vectors of the children in the composite rejector. Weng [Weng 94] uses a similar hierarchical structure which also takes advantage of this.

2. Whereas rejection is geared towards the computational efficiency of recognition, discriminant analysis is concerned with representational compactness. This paper, in part, illustrates the relationship between the two. Pattern rejection can be regarded as an attempt to bring together ideas from the nearest neighbor literature, which is

[1] A brief discussion on decision trees can also be found in [Duda and Hart 73].

primarily concerned with complexity issues, and the pattern recognition literature, which is more concerned with representational issues.

3. A weakness of discriminant analysis is that there is little known about when it can be expected to work. In contrast, for rejectors, our results provide much insight into this issue. Central in this respect is the class assumption.

3 Theory

In this section, we begin by defining both classifiers and rejectors as algorithms. The notion of a rejection-based classifier is introduced and its efficiency is discussed. Next, the general concept of a composite rejector is put forth and its time and space requirements are analyzed.

3.1 The Setting

A pattern recognition task is always based on a finite set of measurements of an underlying physical process. In this paper, we restrict attention to the case where the measurements consist of real numbers. However, even if the measurements are discrete valued, it is often both simple and desirable to convert them into reals. Hence, we assume the existence of a *classification space,* $S = \Re^d$, where the integer, d, is the number of measurements taken. Elements, $x \in S$, will be referred to as *measurement vectors*, or for convenience, *vectors.*

For each pattern class that is to be recognized, we can, at least conceptually, consider the set of measurement vectors that should ideally[2] be classified as belonging to that class. So, we assume the existence of a finite collection, $W_1, W_2, \ldots, W_n \subseteq S$, of *pattern classes,* or simply *classes.* The classes themselves are defined by the application in question and we will therefore assume that they are given to us *a priori*[3].

[2]Our model of pattern recognition is deterministic in the sense that for every measurement vector and every class, the vector is either a member of the class, or not a member of the class. Probabilistic (Bayesian) models, where a measurement vector is assigned a probability of being a member of a given class, are subsumed by this model. Since probabilistic models are meaningless in isolation and without a decision theory, we can regard the classes, W_i, as being defined by, say, the *Bayes decision rule* [Duda and Hart 73].

[3]There are a number of ways in which the classes can be obtained [Fukunaga 90]. One possibility is that the classes are derived using an analytical model of the underlying physical process, such as is the case in parametric feature detection [Nalwa 93] [Nayar et al. 95]. In many applications, however, modeling the underlying physical process proves extremely difficult. Then, the classes are often empirically estimated using sample measurement vectors of known classification. This procedure, which relies on some form of interpolation between sample points, has been the most widely studied problem in pattern recognition. For our purposes, we assume that that an appropriate model of interpolation has been decided upon, which then defines the classes W_i. We proceed to address efficient classification.

3.2 Basic Definitions

A classifier is simply an algorithm that returns the class label (if any) of the class in which the input measurement vector lies:

Definition 1 *A* classifier *is an algorithm, ϕ, that given an input, $x \in S$, returns the class label, i, of the class*[4] *for which $x \in W_i$. If $\forall i$, $x \notin W_i$, the classifier, ϕ, returns nothing.*

We now introduce a *rejector* as a generalization of a classifier. It is a generalization in two senses: (a) a rejector returns a set of class labels rather than a single label, and (b) although the set of labels must contain the label which a correctly functioning classifier would return, it is also allowed to contain more:

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 comes from the equivalent definition: $i \notin \psi(x) \Rightarrow x \notin W_i$. That is, if i is not in the output of the rejector, we can safely eliminate[5] the class, W_i, from further consideration. On the other hand, if $i \in \psi(x)$, we cannot be sure whether $x \in W_i$ or not. For notational convenience, we now introduce the term, *rejection domain,* for the set of all $x \in S$ for which the class, W_i, can be rejected:

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

Then, the following three important properties hold:

1. For any valid rejector, ψ, each class, W_i, and its corresponding rejection domain, R_i^ψ, are disjoint ($R_i^\psi \cap W_i = \emptyset$). This follows immediately from the above definitions since:

$$x \in W_i \quad \text{(Def'n 2)} \Rightarrow \quad i \in \psi(x) \quad \text{(Def'n 3)} \Rightarrow \quad x \notin R_i^\psi \tag{1}$$

2. Subject to the one constraint that, $R_i^\psi \cap W_i = \emptyset$, we are completely free to choose the rejection domains and still conform with the correct definition of a rejector:

$$x \in W_i \quad (R_i^\psi \cap W_i = \emptyset) \Rightarrow \quad x \notin R_i^\psi \quad \text{(Def'n 3)} \Rightarrow \quad i \in \psi(x) \tag{2}$$

   The resulting freedom to choose rejection domains with "simple" decision boundaries, is what allows rejectors to be efficient.

[4]This definition of a classifier implicitly assumes that the classes, W_i, are disjoint, which is often the case. Generalization to the non-disjoint case is straightforward.

[5]Although this is phrased as the class, W_i, being eliminated, more generally, we can think of the pair of input and class, (x, W_i), as being rejected. Hence, depending on the setting, we can view either the input, x, or the class, W_i, as being ruled out.

3. As argued above, the rejector, ψ, can be used to eliminate W_i from further consideration if and only if $i \notin \psi(x)$, and by Definition 3, that is if and only if $x \in R_i^{\psi}$. Hence, to reject as many classes (inputs) as possible, we should aim to choose the rejection domains, R_i^{ψ}, to be as large as possible. However, there is a trade-off between maximizing R_i^{ψ}, ensuring $R_i^{\psi} \cap W_i = \emptyset$, and using simple decision boundaries for efficiency.

3.3 Rejection Based Classifiers

Applying a rejector does not guarantee that we will always be able to uniquely classify an input, since there may be more than one class in the output of the rejector. We deal with this potential ambiguity by adding a verification stage:

Definition 4 *A* verifier *for a class W_i is a boolean algorithm which, given an input, $x \in S$, returns the result,* 1*, if x is a member of W_i, and* 0 *otherwise.*

We form a *rejection-based classifier,* ϕ^{rb}, by first applying a rejector, ψ, and then applying a verifier for each class, W_i, where $i \in \psi(x)$, the output of the rejector. From the outputs of the verifiers, we can immediately classify the input, $x \in S$. The efficiency of the rejection-based classifier is given by:

$$T_{av}(\phi^{rb}) = T_{av}(\psi) + E_{x \in S}(|\psi(x)|) \cdot T_{ver} \tag{3}$$

where, $T_{av}(\phi^{rb})$ is the average run time of the rejection-based classifier, $T_{av}(\psi)$ is the average run time of the rejector, $E_{x \in S}(|\psi(x)|)$ is the expected cardinality of the rejector output, and T_{ver} is the run time of each of the verifiers (assumed to be the same for all verifiers). Equation (3) is derived by noting that we must always apply the rejector (which contributes the term, $T_{av}(\psi)$) and that on average we must apply $E_{x \in S}(|\psi(x)|)$ verifiers.

The reason for introducing a rejection-based classifier is that we aim to be able to construct very efficient rejectors which are also very good at eliminating most of the classes. Hence, both $T_{av}(\psi)$ and $E_{x \in S}(|\psi(x)|)$ will be small quantities, leading to efficient classification. With $T_{av}(\psi)$ as the measure of the efficiency of a rejector, we now introduce *effectiveness* as a measure of how well a rejector eliminates classes:

Definition 5 *If ψ is a rejector designed for the n classes, $W_1, \ldots, W_n$, we define the* effectiveness *of ψ by:*

$$\mathrm{Eff}(\psi) = \frac{E_{x \in S}(|\psi(x)|)}{n} \tag{4}$$

Note that a small numeric value of $\mathrm{Eff}(\psi)$ corresponds to an "effective" rejector. Then, equation (3) shows that a rejection-based classifier will be efficient when: (a) rejection is *efficient,* and (b) rejection is *effective.*

3.4 Composite Rejectors

As we will see, constructing very efficient rejectors is straightforward. However, in some applications, these rejectors tend to be less effective than might be hoped for. Although a rejector may eliminate a large percentage of the classes, on average a substantial number may also be left as possibilities. However, since the output of a rejector is a subset of classes (which is simply a smaller instance of the original classification problem), we may recursively apply another rejector. If the new rejector is specifically designed for the reduced subset of classes, it may well be able to eliminate some classes which the original rejector was unable to. The result of a such a combination of rejectors is a significant improvement in the overall effectiveness. 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, *such that:*

(a) *Each rejector in* Ψ *is designed to be applied to some subset of* $\{W_1, \ldots, W_n\}$

(b) *There is a rejector in* Ψ *designed for the complete set of classes,* $\{W_1, \ldots, W_n\}$

(c) *For any rejector,* $\psi_\imath \in \Psi$, *and any* $x \in S$, *either* $|\psi_\imath(x)| \leq 1$ *or there is a rejector in* Ψ *designed for the subset of classes,* $\{W_i : \ i \in \psi_\imath(x)\}$

As indicated above, a composite rejector is applied by first applying the rejector designed for the complete set of classes. This yields a subset of the class labels and a reduced instance of the classification problem. By requirement (c) in the above definition, the composite rejector contains a rejector designed for the reduced set of classes. Hence, we can repeatedly apply rejectors in this manner, systematically reducing the number of classes at each step, until we are left with only one class. Alternatively, we may terminate if the rejector fails to result in a reduction in the number of remaining classes (and there are no other[6] unapplied rejectors in Ψ designed for the current set of classes).

The composite rejector is laid out in the form of a directed acyclic graph. Each rejector, $\psi_\imath \in \Psi$, and the subset of classes for which it was designed, corresponds to a node in the graph. There is a directed edge from the node corresponding to $\psi_\imath$ to that corresponding to $\psi_\jmath$, if and only if there is a measurement vector, $x \in S$, such that $\psi_\jmath$ was designed for the subset of classes, $\{W_i : \ i \in \psi_\imath(x)\}$. (If $\psi_\imath$ and $\psi_\jmath$ are designed for the same set of classes, to preserve acyclicity, we only include this edge if $\imath < \jmath$.) Hence, the application of the composite rejector to any measurement vector corresponds to a path through the directed acyclic graph. At each node in the path, the associated rejector is applied and its output determines the edge that should be taken to leave the node. Before we detail the construction of composite rejectors, we analyze their time and space requirements.

[6]It is entirely possible within our definition that a composite rejector may contain several rejectors designed for the same set of classes. This allows the opportunity to try a number of alternative rejectors, increasing the chance that one of them will be successful in rejecting some of the classes. Using multiple rejectors in this way is especially useful in cases where there is just one class to be recognized, for example in feature detection.

3.5 Time Analysis

Intuitively, the motivation for introducing a composite rejector is that designing a rejector for a reduced set of classes should be easier and enable the rejection of classes not previously eliminated. Hence, we expect that the composite rejector will be far more effective than any of its individual constituent rejectors, at the cost of a slight reduction in efficiency. The recursive structure of the composite rejector leads us to expect the complexity of the resulting rejection-based classifier to be logarithmic in the number of classes. Sufficient conditions to prove such a result are as follows:

1. For all $\psi_\imath \in \Psi$ and $x \in S$, either $|\psi_\imath(x)| \leq 1$, or at least one class is eliminated by $\psi_\imath$.
2. With respect to the underlying *a priori* probability density function from which the measurement vectors are drawn, the events, $R_j^{\psi_\imath}$, are mutually independent.
3. The effectiveness of all the component rejectors is the same: $\forall \imath \in \Im$, $\mathrm{Eff}(\psi_\imath) = \mathrm{E}$, say.

Then, a composite rejector truncated to apply at most k simple rejectors, has an effectiveness bounded above by $\frac{1}{n} + \mathrm{E}^k$. This follows from the fact that condition 1 given above implies that either we have at most one class left, or we have another rejector to apply. The first case is covered by the term $\frac{1}{n}$, and so we can assume the second case applies. Conditions 2 and 3 ensure that, for each subsequent rejector, the effectiveness is reduced by a factor of E, hence the term E^k. Setting $k = \lceil \log_{\mathrm{E}^{-1}} n \rceil$ and then using the truncated composite rejector in a rejection-based classifier, ϕ^{rb}, we have logarithmic time complexity:

$$T_{av}(\phi^{rb}) \leq \lceil \log_{\mathrm{E}^{-1}} n \rceil \cdot T_{rej} + 2 \cdot T_{ver} \tag{5}$$

where, T_{rej} is the run time of each of the rejectors in Ψ (assumed constant), and n is the number of classes.

3.6 Space Analysis

A potential problem with the composite rejector is that the number of rejectors within Ψ may become very large, possibly as many as 2^n, the number[7] of subsets of $\{W_1, \ldots, W_n\}$. To avoid such exponential growth in the space and preprocessing requirements of the composite rejector, we must impose further constraints on each rejector, $\psi_\imath \in \Psi$. We require that:

1. For each simple rejector, $\psi_\imath$, the number of different possible output subsets is two.
2. The two possible output subsets of classes are of equal cardinality.

[7]There may possibly be more if we have several rejectors per subset of classes. So, in what follows, we will assume that we construct at most one rejector for any given subset of classes. For similar reasons, we also assume that we only construct rejectors if they can actually be reached from the initial rejector, that is, the one for the complete set of classes.

3. The intersection between the two outputs consists of at most a fraction, $\epsilon \in [0, 1)$, of the number of classes for which the rejector was constructed.

Then, if we denote by $M(n)$ the maximum number of rejectors in Ψ that may be reached after, and including, a rejector constructed for a collection of n classes, we have:

$$M(n) \leq 1 + 2 \cdot M((1+\epsilon) \cdot n/2) \tag{6}$$

By induction on n, it can be proven that $M(n)$ is polynomial in n:

$$M(n) \leq n^{1/(1-\log_2(1+\epsilon))} - 1 \tag{7}$$

For $\epsilon = 0$, the bound is $M(n) \leq n - 1$, and for $\epsilon = \sqrt{2} - 1 \approx 0.41$, it is $M(n) \leq n^2 - 1$.

In practice, it may not be possible to completely satisfy the three requirements stated above. However, the following three design criteria may be used as guidelines while implementing each rejector in the composite rejector, Ψ: (a) avoid rejectors that produce a large number of outputs, (b) attempt to balance the output cardinalities, and (c) minimize the overlap between the outputs.

4 Construction of Composite Rejectors

As explained in the previous section, a composite rejector has a hierarchical structure that includes a number of simple rejectors as components. We now describe the general purpose technique used to construct each of the component rejectors. The composite rejector is then formed by recursively building the component rejectors, starting with one for the complete set of classes. Depending upon the application, alternative methods of rejector construction may be possible. If so, they can easily be combined with the following in the composite rejector.

4.1 Notation

We write the Euclidean inner (dot) product of two vectors, $x, y \in S$, as, $\langle x, y \rangle = \sum_{i=1}^{d} x_i \cdot y_i$, where x_i, y_i are the coordinates of the vectors $x, y \in S$ respectively. The induced Euclidean (l^2) norm we denote by $\|x\| = \langle x, x \rangle^{1/2}$. We assume that the norm of a vector is unimportant for classification purposes. It is only the direction of the vector that matters. Hence, we restrict attention[8] to the surface of the unit ball, $B = \{x \in S : \|x\| = 1\}$. We will assume that both the measurement vectors, and the classes, $W_1, W_2, \ldots, W_n$, have been normalized and thus lie in B. Normalization can be achieved by replacing $x \in S$, with $\frac{x}{\|x\|} \in B$.

[8]The assumption that all the vectors lie in B is not restrictive in the following sense. It is possible to code the magnitude of a vector, $x \in S = \Re^d$, in a vector of unit norm in $\Re^{d+1}$. The vector, $x = (x_1, x_2, \ldots, x_d)^T$, is replaced with, $x' = (x_1, x_2, \ldots, x_d, 1)^T$, and then x' is normalized. The magnitude of x is then encoded in the last coordinate of $x'/\|x'\|$.

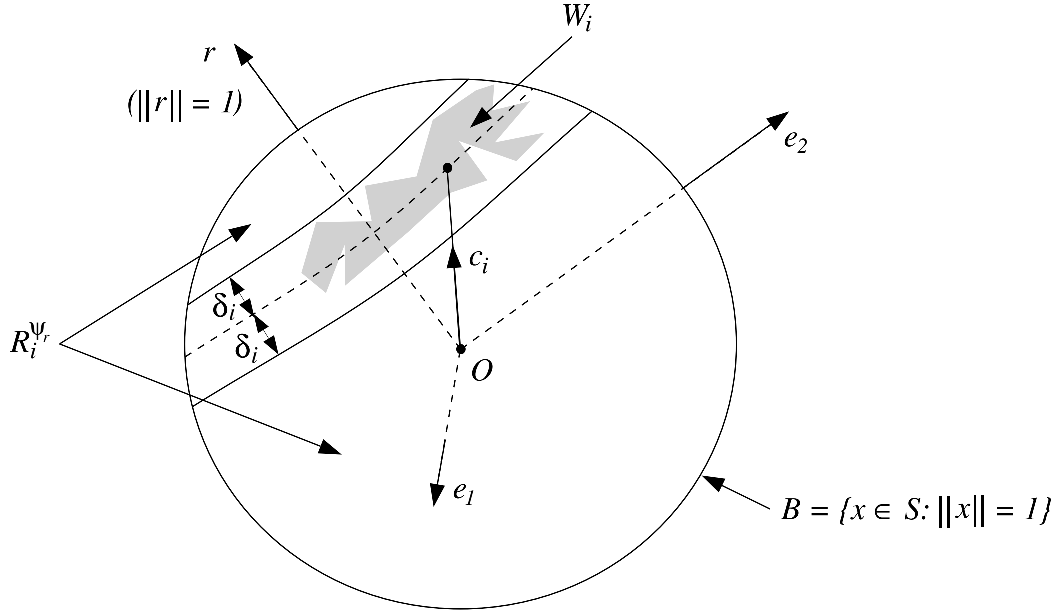

Figure 1: An illustration of the class assumption for a low dimensional example, $S = \Re^3$. The subspace, L_i, is the 2 dimensional subspace spanned by the vectors, $\{e_1, e_2\}$. Every vector in W_i can be approximated to within an error, δ_i, by the linear combination of c_i and a vector in L_i. The rejection vector, r, is a unit vector orthogonal to the subspace L_i. The rejection domain, $R_i^{\psi_r}$, of the derived rejector, ψ_r, consists of all points, $x \in B$, which have a projection in the direction of the rejection vector, r, at least δ_i away from the projection of c_i.

4.2 The Class Assumption

Designing a rejector is equivalent to deciding on the rejection domains associated with each of the classes. Since for correctness we require that $R_i^{\psi} \cap W_i = \emptyset$, the choice of rejection domains depends heavily on the nature of the underlying classes. Hence, we make the following assumption, which is illustrated in Figure 1:

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 $\forall x \in W_i$, $\mathrm{dist}(x, c_i + L_i) \leq \delta_i$. Further we assume, that:* (a) $\dim(L_i) \ll d$, *and* (b) $\delta_i \ll 1$.

It is therefore assumed that any vector in the class can be approximated to within a small error, δ_i, by a linear combination of a fixed vector, c_i, and a vector in the subspace, L_i. For the class assumption to be useful, it must be: (a) general enough to apply in a large number of applications, and (b) restrictive enough to facilitate the construction of rejectors which are both efficient and effective. In the following sections, we first demonstrate the generality of the class assumption by showing its relationship with the Karhunen-Loéve (K-L) expansion. Then, we proceed to show how the class assumption leads to a efficient and effective general form for a rejector.

4.3 Relationship with the K-L Expansion

The class assumption can be seen to be very general and allows many different forms for the classes, W_i, including, for instance, disconnected multi-cluster distributions. Its true generality can be demonstrated by noting that it is approximately equivalent to assuming that the application of the K-L expansion [Fukunaga 90] [Oja 83] results in a compact and accurate representation of the class, W_i. Suppose that M_i^k is the subspace spanned by the k most important K-L eigenvectors, and $\{\lambda_i : i = 1, \ldots, d\}$ are the decaying K-L eigenvalues. Then, we have:

$$E_{x \in W_i}(\text{dist}(x, E_{y \in W_i}(y) + M_i^k)^2) = \sum_{s=k+1}^{d} \lambda_s \ (\approx 0) \tag{8}$$

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

4.4 Verifying the Class Assumption

Since the K-L expansion may be computed efficiently (see for example [Chittineni 81] or [Murakami and Kumar 82]), we use it to validate the class assumption and moreover to find L_i and c_i. For each class, W_i, we put $c_i = E_{x \in W_i}(x)$, and take L_i to be the subspace spanned by the k most important K-L eigenvectors. With these estimates in place, it is straightforward to check if the maximum representation error, δ_i, is sufficiently small. (A better method of selecting the thresholds is discussed in Appendix A.)

Inherent in the class assumption is a trade-off. If we are prepared to accept the use of a subspace with higher dimensionality, we can expect to be able to reduce δ_i. Similarly, if we reduce $k = \dim(L_i)$, we will generally need to increase δ_i. There is an equivalent trade-off in the Karhunen-Loéve expansion between the compactness and accuracy of the representation. In our implementation, the value of k is dependent on the particular class, and is chosen by thresholding the sum of the discarded eigenvalues.

4.5 Derivation of a General Purpose Rejector

Given that the class assumption holds, we are now in a position to derive a general form for a rejector. We begin by defining the notion of a *rejection vector,* which is illustrated in Figure 1:

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

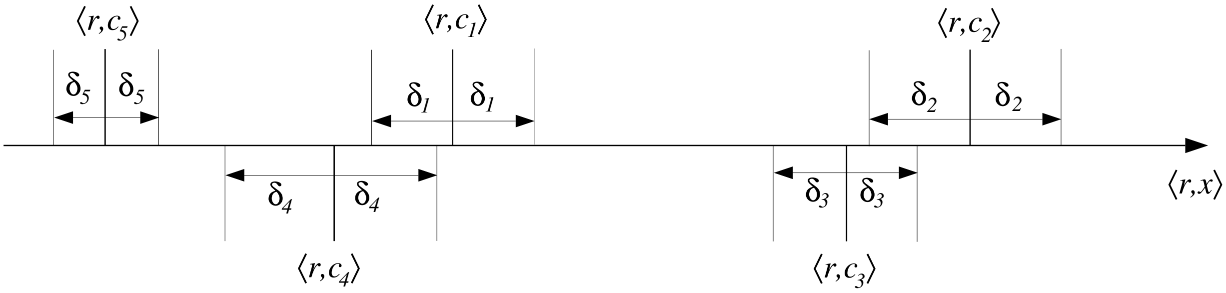

Figure 2: The effect of applying a rejection vector, r. The function $x \to \langle r, x\rangle$ maps each class, W_i, onto the interval, $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$. So long as the δ_i are small and the centers of the intervals, $\langle r, c_i\rangle$, are well separated, most pairs of intervals will not intersect. This makes the rejection vector a highly effective one. Given a measurement vector, $x \in B$, the output of the derived rejector, $\psi_r(x)$, consists of all i for which $\langle r, x\rangle$ lies in the interval $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$.

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

$$x \in W_i \;\Rightarrow\; \exists l_i \in L_i : \; \|c_i + l_i - x\| \le \delta_i \;\Rightarrow\; |\langle r, c_i + l_i - x\rangle| \le \delta_i \;\Rightarrow\; |\langle r, c_i\rangle - \langle r, x\rangle| \le \delta_i \quad (9)$$

Equation (9) means that the rejection vector, r, projects every measurement vector of an entire class, W_i, onto the subinterval, $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$, of the real line. Since $\delta_i \ll 1$, this interval is almost a point, and so a compact characteristic of the class. More importantly, if a measurement vector is not projected into the short interval, $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$, the class, W_i, can be safely rejected.

So long as the thresholds, δ_i, are small, and the centers of the intervals, $\langle r, c_i\rangle$, are well spread out, the intervals themselves will not overlap significantly, as illustrated in Figure 2. Then, we can easily discriminate[9] between the classes based on their projections, and so define the *derived rejector* as follows:

Definition 8 *Given that the class assumption holds for the classes $W_1, W_2, \ldots, W_n$, and that $r \in B$ is a rejection vector, we define the* derived rejector, *ψ_r, by:*

$$i \in \psi_r(x) \;\Leftrightarrow\; |\langle r, x\rangle - \langle r, c_i\rangle| \le \delta_i \quad (10)$$

Hence, the derived rejector returns the class labels of any classes, W_i, for which the point, $\langle r, x\rangle$, lies in the interval, $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$, that is the class labels of any class from

[9]There is no guarantee that we will be able to find a rejection vector that will completely distinguish between a given pair of classes. For example, if the convex hulls of the classes overlap, their projections with any rejection vector will intersect. Note, however, that this occurrence need not effect the usefulness of a derived rejector since the goal of a rejector is to eliminate most of the classes most of the time, as opposed to complete discrimination all of the time. We are implicitly assuming that in a large collection of classes, pairs of classes which are difficult to discriminate occur relatively rarely, not that they do not occur at all. In our object recognition application, this is indeed a very natural assumption.

which the measurement vector might have come. The rejection domain of ψ_r for the class, W_i, is $R_i^{\psi_r} = \{x \in B : |\langle r, x\rangle - \langle r, c_i\rangle| > \delta_i\}$, as illustrated in Figure 1. Equation (9) then shows that $W_i \cap R_i^{\psi_r} = \emptyset$, and hence ψ_r is well defined as a rejector.

The derived rejector may be implemented very efficiently as follows. (A slightly modified method more appropriate for use in a composite rejector is described in Appendix B.) First, we compute the projection of the measurement vector, $x \in B$, with the rejection vector, to give $\langle r, x\rangle$. Then, the set of class labels, i, for which $\langle r, x\rangle$ lies in the interval, $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$, can be computed with $\lceil \log_2(2n+1)\rceil$ comparisons and a lookup table. This is possible because the derived rejector is a piecewise constant function. It only changes its value at the $2n$ points, $\langle r, c_i\rangle \pm \delta_i$. The constant values on the intervening segments can easily be precomputed and stored in the lookup table. Finding the segment in which $\langle r, x\rangle$ lies takes $\lceil \log_2(2n+1)\rceil$ comparisons using a binary search.

4.6 Choice of the Rejection Vector

We have seen that the derived rejector can be applied efficiently. The reason we can expect it to be effective is because we have quite some freedom in choosing the direction of the rejection vector, r. Thus far, r has only been constrained to lie orthogonally to $\bigoplus_{i=1}^n L_i$. We enforce this constraint immediately by taking each vector used from now on, and subtracting the component in the space, $\bigoplus_{i=1}^n L_i$.

As Figure 2 shows, we should choose the rejection vector to be the one that spreads out the centers of the intervals, $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$, as much as possible. This will reduce the size of $|\psi_r(x)|$, and so tend to optimize the effectiveness of the derived rejector. If variance is used to measure the spread of the centers, the best rejection vector to choose is the first Karhunen-Loéve eigenvector[10], that is the one with the largest eigenvalue.

If there is just one class, as is in the feature detection application, the K-L expansion cannot be applied because there is only one vector, c_i. In this situation, we select the rejection vector uniformly at random in the space orthogonal to $\bigoplus_{i=1}^n L_i$. As described in [Knuth 81], this can be performed by drawing the d coordinates from a normal distribution, projecting out the component in $\bigoplus_{i=1}^n L_i$, and then normalizing to obtain a rejection vector that lies on the unit sphere, B.

[10] This choice of rejection vector may be seen to be closely related to Fisher's discriminant analysis [Fisher 36] [Duda and Hart 73]. By working in a space orthogonal to $\bigoplus_{i=1}^n L_i$, we are limiting the within-class scatter of each class, W_i. Spreading out the points $\langle r, c_i\rangle$, maximizes the between-class scatter. The important difference, however, is the inherent conservative nature of the derived rejector, which ensures we never make a wrong choice, and defers difficult decisions to subsequent rejectors which are in a better position to discriminate between the difficult cases.

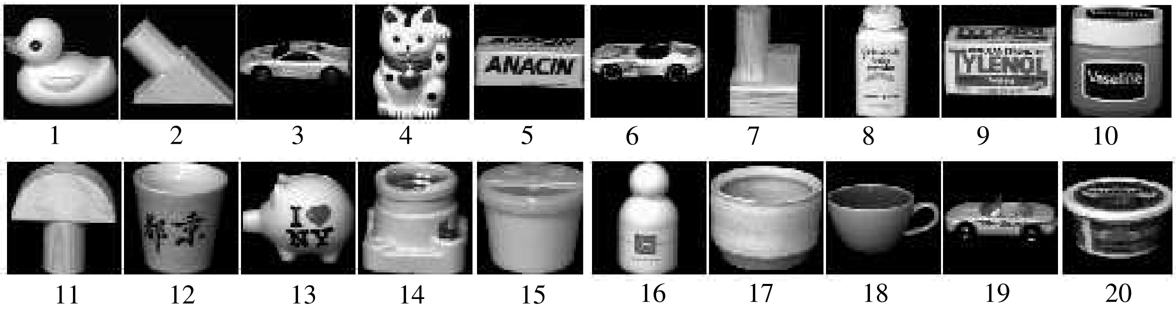

Figure 3: The 20 objects used for recognition. We used 72 images of each object, with consecutive images separated by 5° of pose. The data set is the same as used in [Murase and Nayar 95].

5 Experiments

The theory of pattern rejection is general and hence should find use in a variety of applications. Here, our objective is to demonstrate the generality, efficiency, and effectiveness of composite rejectors. 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 3-D Object Recognition

There are several approaches to 3-D object recognition, most of which attempt to match features in images to 3-D object models [Besl and Jain 85] [Chin and Dyer 86]. Recently, an alternative approach called appearance matching has gained popularity, where objects are modeled as collections of 2-D views [Edelman and Weinshall 91] [Poggio and Edelman 90] [Murase and Nayar 95]. The main advantages and limitations of appearance matching are described in [Murase and Nayar 95]. Similar view-based recognition techniques have also been applied to the problem of face recognition [Pentland et al. 94] [Brunelli and Poggio 93] [Sirovich and Kirby 87] [Turk and Pentland 91].

In our experiments, we use appearance matching simply as an example of the large class of problems for which efficient rejectors can be constructed. For simplicity we assume a constrained environment. We require that the object can be segmented from the background, is not occluded substantially, and appears in unknown pose but in one of a small number of stable configurations. Also, we assume that the illumination of the environment remains more or less unchanged. Under these conditions, appearance matching, as described in [Murase and Nayar 95], reduces object recognition to a classical pattern recognition problem. We first segment the object, and then scale by resampling the image so that the larger of the two object dimensions fits a preselected image size. In our implementation, the image size was 128×128 pixels. The scale normalized image is then treated

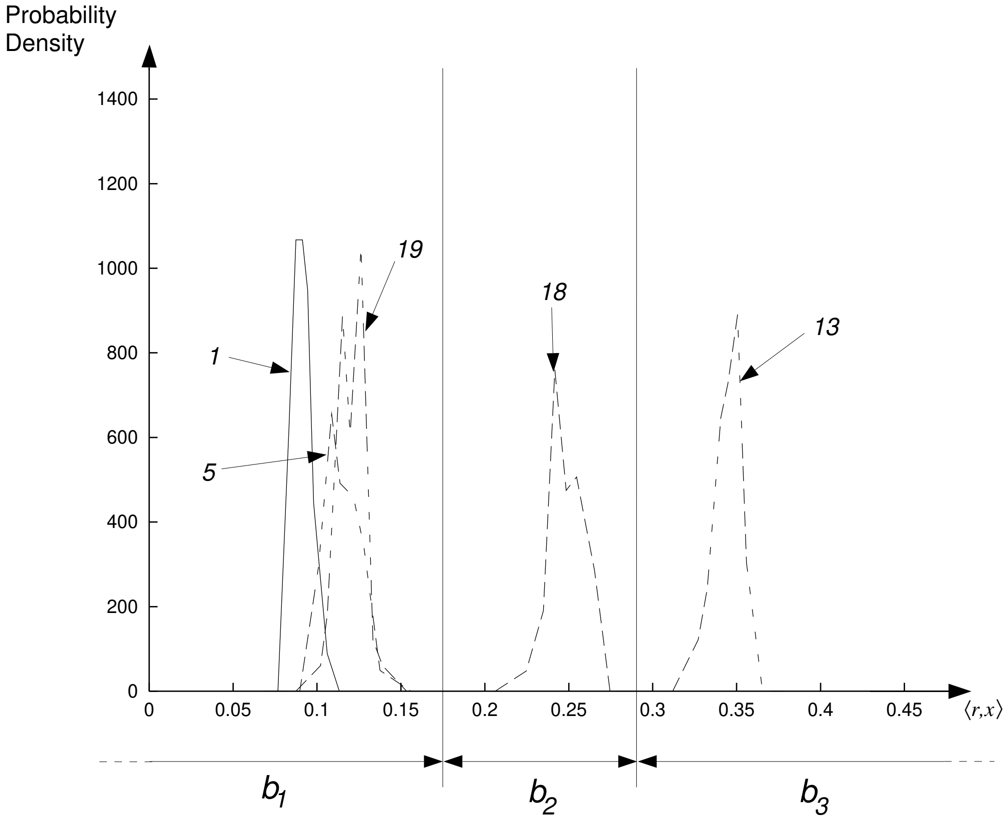

Figure 4: An example rejector for the set of objects, $\{1, 5, 13, 18, 19\}$. Following the procedure in Appendix B, we select 3 buckets, $b_1 = [-1.0, 0.18]$, $b_2 = [0.18, 0.29]$ and $b_3 = [0.29, 1.0]$. If $\langle r, x \rangle$ falls in bucket b_1, the rejector returns the set of class labels, $\{1, 5, 19\}$, if it falls in b_2 the rejector returns $\{18\}$, and if it falls in b_3 the rejector returns $\{13\}$. Since the use of such a rejector involves no more than a single dot product with the measurement vector followed by bin assignment, rejection proves both efficient and effective.

as a 16,384 dimensional vector by reading pixel values in a raster scan fashion. Finally, the image vector is intensity normalized to yield a unit vector which is fed as the input measurement vector to our rejector.

We used 20 objects in our experiments, each corresponding to a class. A single image of each object is displayed in Figure 3. We assume that each of the objects can appear in just one stable configuration. Thus, the pose of the object with respect to the viewer is given by a single rotation parameter. We used 72 images of each object taken at 5° intervals of pose. The images were divided into two sets, each set consisting of 36 images separated by 10° of pose. One set of images was used as training samples that define the classes, and the other set was reserved exclusively for testing the composite rejector.

We implemented a composite rejector for the 20 objects using the procedure outlined in Section 4. As an example, Figure 4 shows one of the constituent rejectors. A representation of the entire composite rejector is illustrated in Figure 5, part of which is expanded in

Figure 6. As can be seen in Figure 5, every leaf of the composite rejector contains a single class. Hence, the composite rejector is capable of discriminating between the 20 objects without ambiguity. It is guaranteed to assign a unique class to any input vector. This may be regarded as fortunate. The aim of the rejector is simply to eliminate most of the objects, and we would have regarded the rejector as successful even if each leaf had contained up to 2-3 objects that needed to be disambiguated using verifiers. We applied the rejector to all 72 images of each object, both the training images used for rejector construction as well as the test images that we set aside. We found that the rejector gave 100% correct response in both cases. It is worth noting that the composite rejector contains just 30 simple rejectors. This should be compared with the number of subsets of 20 objects, which turns out to be over 10^6.

As seen in Figure 5, the longest path in the composite rejector consists of 10 steps. Hence, the maximum number of simple rejectors needed to eliminate all but one of the classes is 10. By assuming that each image in the data set is equally likely to appear, we calculated the average number of rejectors needed to be just 6.43. In other words, the average run time of the composite rejector is the time it takes to compute 6.43 inner products plus the small overhead of walking the path in the directed graph. Since at each node there are at most 4 possible paths to take, making the decision consists of only two comparisons. This efficiency compares very favorably with the results obtained by Murase and Nayar [Murase and Nayar 95] on the same data. Their implementation based on the Karhunen-Loéve expansion required 20 inner products, followed by a sophisticated search procedure. If the time to calculate the inner products is the most important component in the overall time cost, the composite rejector is approximately 3 times more efficient. Further, given dedicated hardware to compute inner products, the rejector will yield even better improvement in performance since we require no complex search procedure. In cases where the composite rejector has leaves with multiple objects, tuned verifiers of the type used by Murase and Nayar [Murase and Nayar 95] can be used at the leaves to complete classification.

We investigated the growth rate of the number of rejectors required as a function of the number of classes by considering subgraphs of the composite rejector. The set of all vertices that can be reached from a given node in the composite rejector, can itself be regarded as a composite rejector, but for a reduced subset of the 20 objects. So, for each vertex in the graph, we approximated the average number of simple rejectors required for the composite rejector rooted at that vertex. In Figure 7, the logarithm of the number of classes for which the rejector is designed is plotted against the average number of rejectors required. Where there are several composite rejectors for a similar number of classes, we plot the average over all such cases. We calculated a least squares fit of a straight line (shown as a solid line) to the data. It is evident that the data validates our previous theoretical results and in particular equation (5). This equation predicted that the required number of rejectors would be a logarithmic function of the number of objects.

Using the same image database, we now compare the performance of the composite rejector against that of Fisher's discriminant analysis [Fisher 36]. Again, we followed the

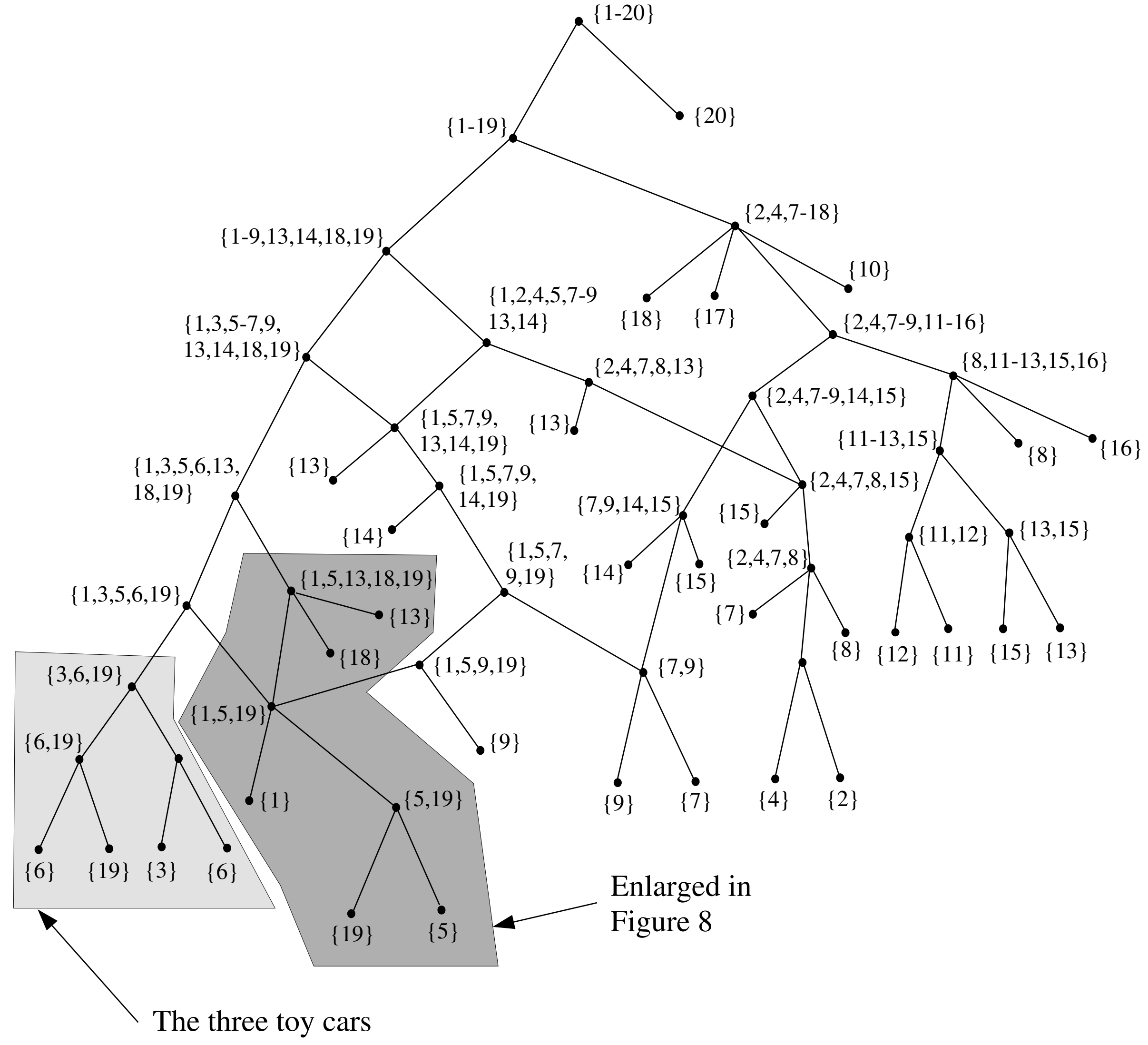

Figure 5: A representation of the composite rejector. Each interior node denotes a single rejector and is labeled with the set of objects that it is designed to act on. At each node, only one inner product and a couple of comparisons need to be performed. Each leaf denotes a possible output of the composite rejector. It is interesting to note that objects with similar "gross shape" tend to group together at higher levels of the rejector, and are only separated closer to the leaves. For example, the three toy cars (objects 3, 6, & 19) are almost indistinguishable until all other objects are eliminated.

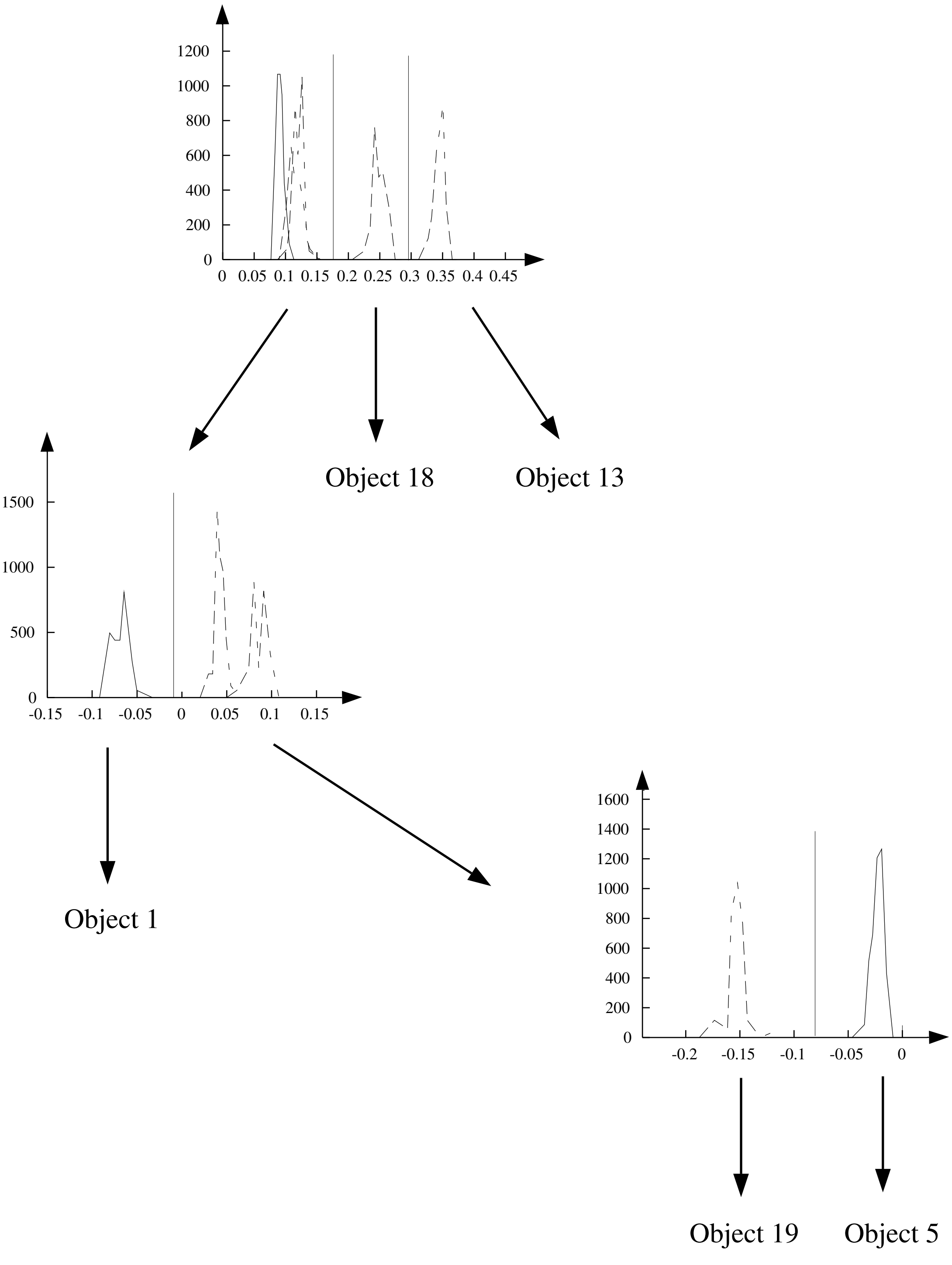

Figure 6: A small part of the composite rejector. As the number of classes is reduced by each successive simple rejector, subsequent rejectors become more tuned to the set of remaining classes. The five objects are first reduced to three, then two, and finally just one.

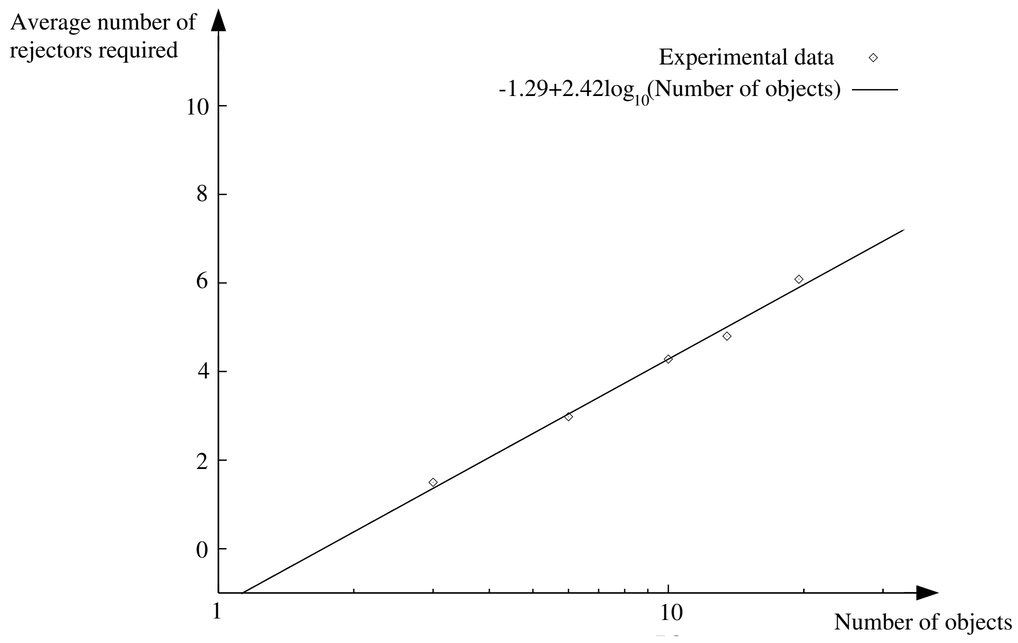

Figure 7: A graph of the number of objects against the average number of simple rejectors required to completely discriminate between objects. The graph is plotted using a log scale on the abscissa, implying a logarithmic growth rate in the time complexity.

same test procedure, namely, setting aside half of the data, and using the other half to construct the classifier. Then, we constructed the Fisher spaces [Duda and Hart 73] of different dimensions. In Fisher space the classes consist of tight clusters, which we model as multivariate normal distributions. We computed the mean and covariance matrix of each of these distributions. Then, each measurement vector was classified by finding its closest cluster, i.e. the cluster whose mean is closest to the vector. We used both the Mahalanobis and Euclidean distances. Figure 8 shows the results plotted as a graph of the percentage of test images correctly classified, against the dimension of the Fisher space used. The results shown are for the combined performance on the training and test sets. The classifier performs slightly better on the test set and slightly worse on the set aside data. However, the difference between the two is always less than 1%.

The Mahalanobis distance gave consistently better results than the Euclidean distance. However, even for the Mahalanobis measure, classification results are not perfect. In fact, the highest correct classification rate of 96.6% was attained for dimension 19. This compares poorly with the perfect classification obtained by the composite rejector, that uses an average of just 6.43 rejection vectors. The main reason for the rejector's superior performance is that its hierarchical structure eliminates classes step by step, while the rejector used at each step is optimal for the classes the step seeks to distinguish between. As is seen in Figure 6, rejectors closer to the leaves are tuned to a reduced set of classes,

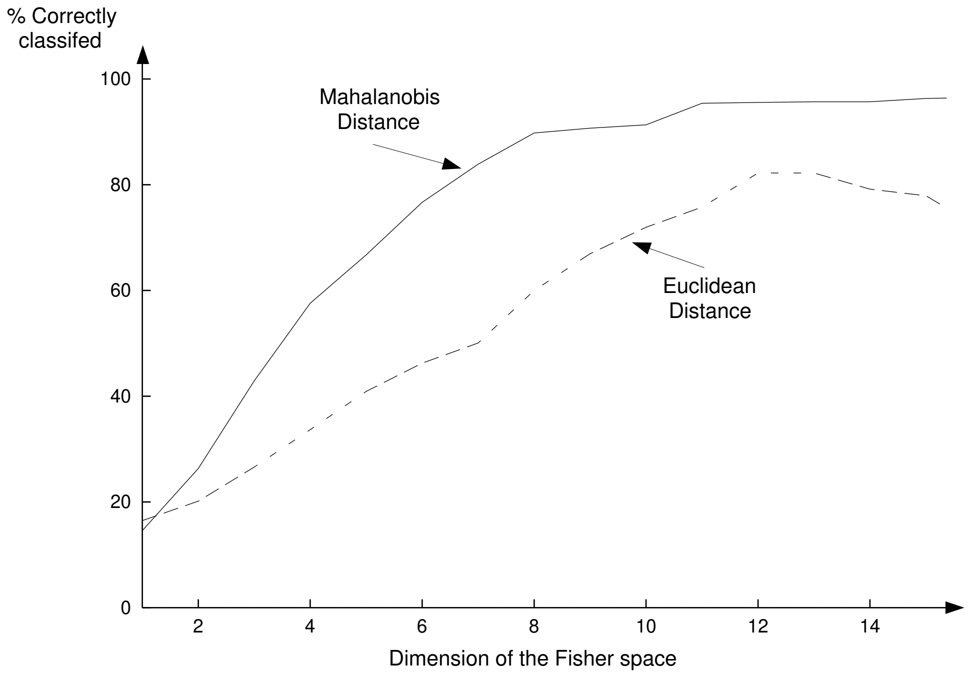

Figure 8: Results of applying Fisher's discriminant analysis to the data set in Figure 3. On the abscissa we plot the dimension of the Fisher space used, and on the ordinate the percentage of test images correctly classified. The peak performance is 96.6% correct, and to reach this 19 discriminant vectors are needed. In contrast, the composite rejector gives perfect (100%) classification with just 6.43 rejection vectors. Hence, by both measures, efficiency and robustness, the composite rejector outperforms Fisher's discriminant analysis.

and so are less "distracted" by other classes. In contrast, all dimensions of the Fisher space simultaneously seek to classify the entire set of classes. As a result, the second and subsequent dimensions turn out to be suboptimal when compared with the second and subsequent layers of a composite rejector.

5.2 Local Feature Detection

Another important problem in computational vision which can be reduced to pattern recognition is the detection of local features (edges, lines, corners, etc.) in an image. The decision of whether the local feature appears at a given pixel in an image, is based entirely on the intensity values in a surrounding window of d pixels. Treating these intensity values as real numbers, we have the classification space, $S = \Re^d$. If we can characterize the class, say, W_f, of intensity vectors which represent the feature, the problem of feature detection is reduced to deciding whether a measurement vector, $x \in W_f$.

For lack of space, we will concentrate solely on the step edge as the example fea-

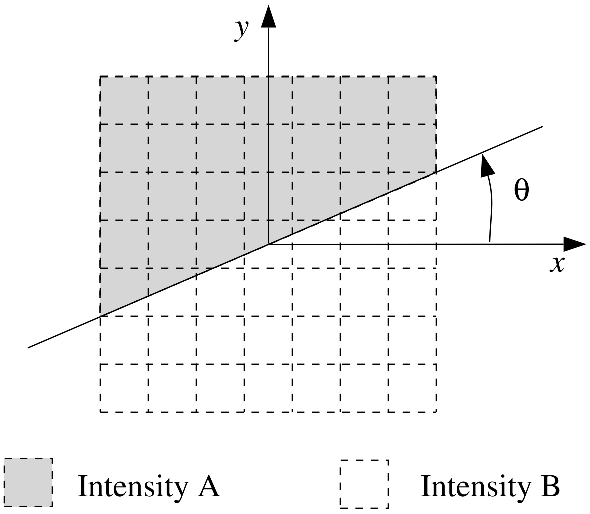

Figure 9: The ideal model of a step edge. We consider a window of size 7×7 around the center pixel. A straight line at angle θ to the x-axis separates the window into two constant intensity regions. When discretized, the pixels which the line passes through are assigned their intensity levels using an anti-aliasing algorithm that calculates the average pixel intensity.

ture. The step edge is the simplest and most widely explored feature. Efficient edge detectors have been proposed, however, the more sophisticated detectors (for instance, complete implementations of the Canny edge detector [Canny 86] and the Nalwa-Binford detector[Nalwa and Binford 86]) are less efficient. Furthermore, elaborate detectors are unavoidable in the case of more complex features, such as lines and corners, as shown in [Nayar et al. 95]. For such features, there is no obvious equivalent to the gradient or Laplacian operators that are often used for edges. In short, as a rule of thumb, high feature complexity and/or high detection accuracy require the use of computationally expensive detectors. This makes feature detection a prime candidate for the application of rejection theory.

The major methods of edge detection are categorized in [Nalwa 93] by how they define the set, W_f. Difference operators, such as the Canny edge operator [Canny 86] and the Marr-Hildreth operator [Marr and Hildreth 80], implicitly define W_f in terms of the magnitude of the gradient (or the Laplacian), of the underlying image intensity function. Model matching methods such as [Nalwa and Binford 86] define W_f using an ideal parameterized model of the edge, which is mapped into the classification space, S, by modeling the imaging process. We follow the model based approach since it gives us an explicit, rather than implicit, definition of W_f.

We used a three parameter model of a step edge, which is illustrated in Figure 9. The edge model occupies a window that includes 7×7 pixels in the image, which leads to a classification space of dimension, $d = 49$. The parameters consist of the two intensity levels, A and B, on the opposite sides of the edge, and the angle, θ, of the edge. The following

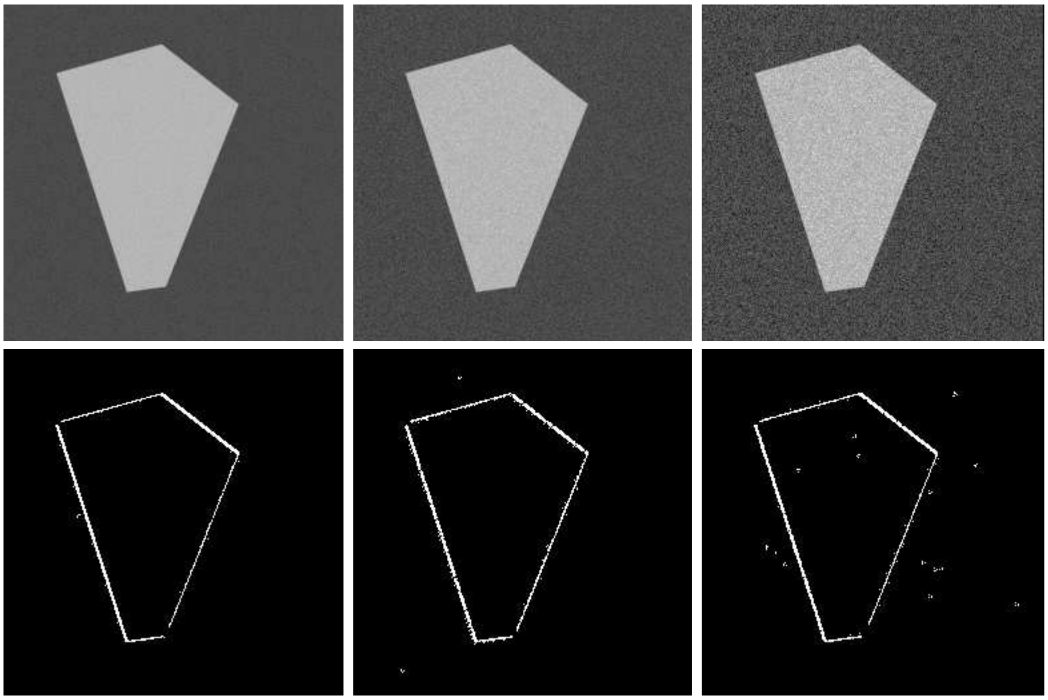

Figure 10: The edge rejector applied to 3 noisy synthetic images. The top row shows the noisy images whose pixels the rejector is applied to. The image on the left has added Gaussian noise of standard deviation 1 grey level, the middle image has noise of 2 grey levels, and the right image has noise of 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) that could be fed into an elaborate edge detector such as the one described in [Nayar et. al 95].

three step normalization allows us to eliminate both of the intensity parameters, A and B, without effecting the underlying edge structure:

1. Given a vector $x = (x_1, \ldots, x_{49})^T$, calculate $\overline{x} = \frac{1}{49} \cdot \sum_{i=1}^{49} x_i$.
2. Subtract $\overline{x}$ from each coordinate of x to get, $x' = (x_1 - \overline{x}, \ldots, x_{49} - \overline{x})^T$.
3. Calculate the norm $\|x'\|$ of x' and return the unit vector $x'/\|x'\|$.

If the input vector is found to conform to the edge model, the parameters A and B may be recovered using, $A,\ B \approx \overline{x} \pm \|x'\|$, the approximation arising from the fact that the images are discretely sampled.

We constructed a composite rejector and applied it to a set of synthetic images and to a real image. The results are displayed in Figures 10 and 11, respectively. The synthetic images in Figure 10 are of size 256×256 pixels, and consists of a high intensity polygon on a low intensity background. The polygon is bounded by 5 line segments at angles $15.4°, 67.4°, 107.9°, 187.9°$, and $322.2°$ to the horizontal axis. The difference in the

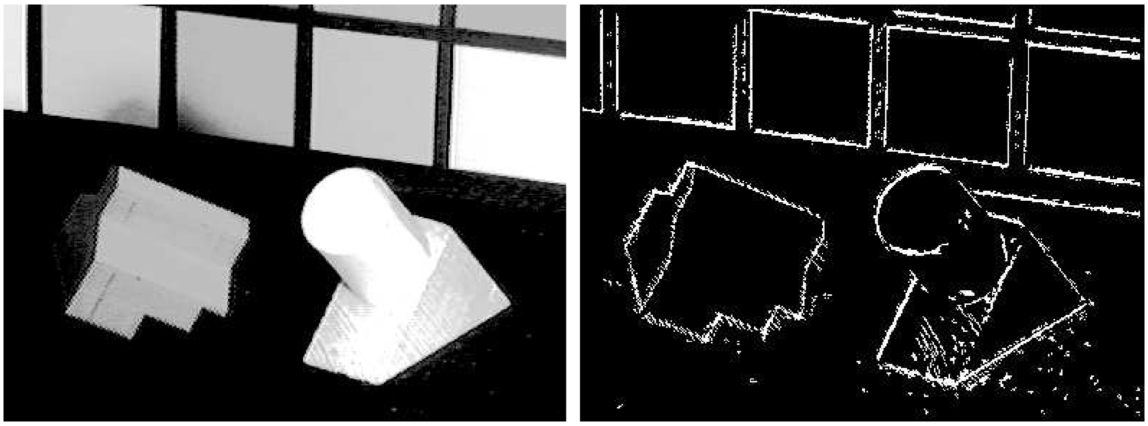

Figure 11: The edge rejector applied to a real scene. The output on the right consists of a large number of pixels (marked black) which the rejection algorithm has quickly eliminated from further consideration and a small number of pixels (marked white) which it has decided are candidates to be verified by a sophisticated edge detector.

intensity across each segment is 50 grey levels. We used three synthetic images to which we added Gaussian noise. In the first image the added noise had standard deviation 1 grey level, the second 2 grey levels, and in the third 4 grey levels.

A composite rejector consisting of 6 rejectors was applied to each of the synthetic images in Figure 10. The output images were passed through a simple relaxation algorithm to remove a few scattered false positives. Since the rejector terminates as soon as it first rejects the pixel as not containing an edge, not all 6 rejectors are used at all pixels. In the least noisy image an average (computed over the whole image) of 1.61 rejectors were used. For the more noisy images, 1.82 rejectors and 2.34 rejectors were used, respectively.

In Figure 11, we show similar results for a real image of size 393×289 pixels taken in the laboratory. We used a composite rejector with 11 rejectors, of which an average of 1.81 were required at each pixel. Again, the output of the rejector is shown after it has been passed through the relaxation algorithm to remove isolated false positives.

6 Discussion

Our major contribution has been to focus on computational (as opposed to representational) approaches to general recognition problems, and to introduce a framework, centered upon the pattern classes, in which the complexity of such problems can be studied. More specifically, the key results of our work include:

1. We have provided conditions for logarithmic growth in time complexity as a function of the number of classes, and verified this behavior empirically for an important ap-

plication in computational vision. However, further investigation of these conditions is needed to enhance our understanding of when they can be expected to hold.

2. We analyzed the growth in the number of rejectors required to construct a composite rejector. The key is the number of possible outputs of the rejectors and the amount of intersection between them. This growth, rather than the time complexity, may well turn out to the limiting factor in the scalability of rejection. A comparison with the much less conservative k-d trees [Friedman et al. 77] would probably throw light on what is essentially a time-space tradeoff.

3. The class assumption is at the heart of our technique for constructing rejectors. As expected, it holds for some classes far more than for others. Further study of when and why it holds would be useful. Given the derived relationship between rejection and the K-L expansion, this is equivalent to asking how well the K-L expansion can be expected to perform. This question was raised in the context of object recognition in [Murase and Nayar 95] and still remains unanswered in that application.

4. We have compared pattern rejection with Fisher's discriminant analysis and demonstrated rejection to be superior. Although discriminant analysis is formally "optimal," its optimality is more with respect to representation and not efficiency. Further, it is only the first Fisher vector that can really be regarded as optimal. We have shown that far better accuracy, efficiency, and discriminating power results from the hierarchical structure of a composite rejector.

Acknowledgements

This research was conducted at the Center for Research on Intelligent Systems at the Department of Computer Science, Columbia University. It was supported in parts by ARPA Contract DACA-76-92-C-007, DOD/ONR MURI Grant N00014-95-1-0601, a NSF National Young Investigator Award, and by a David and Lucile Packard Fellowship.

A Estimation of the Thresholds

The only property the thresholds, δ_i, must comply with, for the derived rejector, ψ_r, to behave correctly as a rejector, is that each class W_i and its rejection domain $R_i^{\psi_r}$, should be disjoint. To ensure this, we only require:

$$x \in W_i \;\Rightarrow\; |\langle r, x\rangle - \langle r, c_i\rangle| \leq \delta_i \quad (\text{since then Def'n 8} \Rightarrow\; x \notin R_i^{\psi_r}) \tag{11}$$

Further, the smaller we can make δ_i without compromising the correct behavior of the rejector, the more effective we can expect ψ_r to be.

The exact details of how to estimate the best value of δ_i are largely application dependent, since it depends heavily on the nature of the distribution of the random variable, $x \to \langle r, x\rangle$, and also the tolerance to error. One possibility is to select δ_i based on measurements of the number of errors made by the implemented rejector. This was the approach taken in our feature detection experiments, where careful adjustments can be made during the design and testing of the feature detector. Another method is to assume a general parameterized form for the distribution and select δ_i based on the estimated parameters of the distribution. In the object recognition experiments, it was found empirically (see Figure 4) that the distributions for all objects reasonably approximated normal distributions. To be conservative, we chose a confidence level of over 99.9%, and so δ_i was selected as 3.5 times the estimated standard deviation of the distribution.

B Implementation of the Derived Rejector

We address a potential problem with the original definition of the derived rejector. We rewrite the definition here for convenience:

$$i \in \psi_r(x) \;\Leftrightarrow\; |\langle r, x\rangle - \langle r, c_i\rangle| \leq \delta_i \tag{12}$$

The intervals, $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$, may overlap in complicated ways leading to a rejector with a large number of different output sets, possibility as large as $2n$. Hence, there is a danger that this rejector design will lead to a very large composite rejector.

We redesign the rejector by introducing *buckets* which form a partition of $[-1, 1]$. The concept of a bucket is illustrated in Figure 12. We divide $[-1, 1]$ into m neighboring buckets, $b_1, \ldots, b_m$, where $\forall j,\ b_j = [\mathrm{cut}_{j-1}, \mathrm{cut}_j]$. We also require that $\mathrm{cut}_0 = -1$, and $\mathrm{cut}_m = 1$. Each point cut_j is referred to as a *cut-point.* Once we have decided on the cut-points, and hence the buckets, we associate with each bucket b_j the set of classes W_i, with which the bucket intersects the interval, $[\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i]$:

$$\mathrm{classes}(b_j) = \{i :\ b_j \cap [\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i] \neq \emptyset\}. \tag{13}$$

It follows from equation (13) that:

$$x \in W_i \text{ and } \langle r, x\rangle \in b_j \;\Rightarrow\; i \in \mathrm{classes}(b_j) \tag{14}$$

Hence, $\psi_r(x) = \mathrm{classes}(b_j)$, where b_j is the unique bucket for which $\langle r, x\rangle \in b_j$, is a valid redefinition of a derived rejector. Using a binary search and a lookup table, the modified rejector can be implemented with one inner product and a logarithmic (in the number of buckets) number of comparisons. The new derived rejector will not be quite as effective as the original one, but will lead to a much smaller composite rejector. This is a classic example of the time-space trade-off.

The reason for introducing the notion of a bucket is so that we may carefully select the cut-points, so as to follow the design guidelines introduced in Section 3.6. We use the

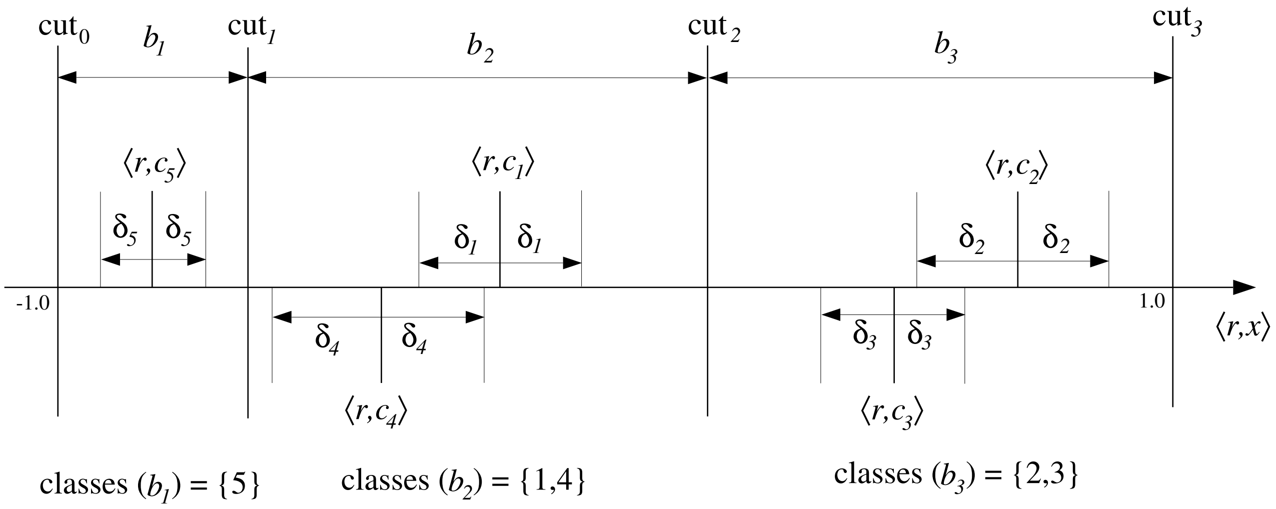

Figure 12: The interval $[-1, 1]$, which corresponds to all possible inner products with a rejection vector, is partitioned into buckets. Neighboring buckets are separated by cut-points. Each bucket is associated with a set of classes, namely, those which have a non-empty intersection with the bucket. We amend the design of the derived rejector to return the set of classes associated with the bucket into which the measurement vector is projected.

following algorithm that aims to: (a) keep the number of buckets, and hence the number of outputs, small, (b) balance the sizes of the output subsets, classes(b_j), and (c) minimize the intersection between the output subsets:

Algorithm: Choice of the cut-points, $\{\mathrm{cut}_j : \ j = 0, 1, \ldots, m\}$.

1. Initialize the set, $J = \{-1, 1\}$.
2. Put $M = \{\langle r, c_i\rangle - \delta_i : \ i = 1, 2, \ldots, n\} \cup \{\langle r, c_i\rangle + \delta_i : \ i = 1, 2, \ldots, n\}$, and $M' = \emptyset$.
3. Sort the set M. For each consecutive pair of numbers in M, put their mean in M'.
4. For each point, $x \in M'$, in turn, insert x into J if and only if:
$$\forall i = 1, 2, \ldots, n, \quad x \notin [\langle r, c_i\rangle - \delta_i, \langle r, c_i\rangle + \delta_i] \tag{15}$$
5. Add to J, the points which maximize over all $y \in M'$, the expression:
$$\min(|\{i : \ y < \langle r, c_i\rangle - \delta_i\}|, |\{i : \ y > \langle r, c_i\rangle + \delta_i\}|) \tag{16}$$
6. Return the set of cut-points, J.

References

[Bentley 80] J.L. Bentley, "Multidimensional divide-and-conquer," *Communications of the ACM,* 23:214–229, 1980.

[Besl and Jain 85] P.J. Besl and R.C. Jain, "Three-dimensional object recognition," *ACM Computing Surveys*, 17:75–145, 1985.

[Brunelli and Poggio 93] R. Brunelli and T. Poggio, "Face Recognition: Features versus templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1042-1052, 1993.

[Canny 86] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[Chin and Dyer 86] R.T. Chin and C.R. Dyer, "Model-Based Recognition in Robot Vision," *ACM Computing Surveys*, 18:67–108, 1986.

[Chittineni 81] C.B. Chittineni, "On the extraction of features from slowly wandering patterns," *IEEE Transactions on Aerospace and Electronic Systems*, 17:461–466, 1981.

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

[Edelman and Weinshall 91] S. Edelman and D. Weinshall, "A self-organizing multiple-view representation of 3-D objects," *Biological Cybernetics*, 64:209–219, 1991.

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

[Friedman et al. 77] J.H. Friedman, J.L. Bentley, and R.A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software,* 3:209-226, 1977.

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

[Henrichon and Fu 69] E.G. Henrichon and K-S. Fu, "A Nonparametric Partitioning Procedure for Pattern Classification," *IEEE Transactions on Computers,* 18:614–624, 1969.

[Knuth 81] D.E. Knuth, *The Art of Computer Programming, Volume II: Seminumerical Algorithms*, Addison-Wesley, 1981.

[Lamdan et al. 88] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson, "Object Recognition by Affine Invariant Matching," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 335–344, 1988.

[Marr and Hildreth 80] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London, Series B,* 207:187–217, 1980.

[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.

[Nalwa 93] V.S. Nalwa, *A Guided Tour of Computer Vision*, Addison-Wesley, 1993.

[Nalwa and Binford 86] V.S. Nalwa and T.O. Binford, "On detecting edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:699–714, 1986.

[Nayar et al. 95] S.K. Nayar, S. Baker, and H. Murase, "Parametric Feature Detection," *Columbia University Technical Report,* CUCS-028-95, 1995.

[Oja 83] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press, 1983.

[Payne and Meisel 77] H.J. Payne and W.S. Meisel, "An Algorithm for Constructing Optimal Binary Decision Trees," *IEEE Transactions on Computers,* 26:905–916, 1977.

[Pentland et al. 94] A.P. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 84–91, 1994.

[Poggio and Edelman 90] T. Poggio and S. Edelman, "A network that learns to recognize 3D objects," *Nature*, Vol. 343, pp. 263–266, 1990.

[Sirovich and Kirby 87] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Journal of the Optical Society of America*, 4:519–524, 1987.

[Tarr and Pinker 89] M. Tarr and S. Pinker, "Mental rotation and orientation-dependence in shape recognition," *Cognitive Psychology*, 21:233–82, 1989.

[Turk and Pentland 91] M.A. Turk and A.P. Pentland, "Face recognition using eigenfaces," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–591, 1991.

[Ullman and Basri 91] S. Ullman and R. Basri, "Recognition by linear combinations of models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:992–1005, 1991.

[Weng 94] J. Weng, "SHOSLIF: The Hierarchical Optimal Subspace Learning and Inference Framework," *Michigan State University Technical Report*, CPS 94-15, 1994.

[Yianilos 93] P.N. Yianilos, "Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces," *Proc. of ACM-SIAM Symposium on Discrete Algorithms,* pp. 311–321, 1993.