Hierarchical Discriminant Analysis for Image Retrieval

Daniel L. Swets, *Member*, *IEEE*, and Juyang Weng, *Member*, *IEEE*

Abstract—A self-organizing framework for object recognition is described. We describe a hierarchical database structure for image retrieval. The Self-Organizing Hierarchical Optimal Subspace Learning and Inference Framework (SHOSLIF) system uses the theories of optimal linear projection for automatic optimal feature derivation and a hierarchical structure to achieve a logarithmic retrieval complexity. A Space-Tessellation Tree is automatically generated using the Most Expressive Features (MEFs) and the Most Discriminating Features (MDFs) at each level of the tree. The major characteristics of the proposed hierarchical discriminant analysis include: 1) avoiding the limitation of global linear features (hyperplanes as separators) by deriving a recursively better-fitted set of features for each of the recursively subdivided sets of training samples; 2) generating a smaller tree whose cell boundaries separate the samples along the class boundaries better than the principal component analysis, thereby giving a better generalization capability (i.e., better recognition rate in a disjoint test); 3) accelerating the retrieval using a tree structure for data pruning, utilizing a different set of discriminant features at each level of the tree. We allow for perturbations in the size and position of objects in the images through learning. We demonstrate the technique on a large image database of widely varying real-world objects taken in natural settings, and show the applicability of the approach for variability in position, size, and 3D orientation. This paper concentrates on the hierarchical partitioning of the feature spaces.

Index Terms—Principal component analysis, discriminant analysis, hierarchical image database, image retrieval, tessellation, partitioning, object recognition, face recognition, complexity with large image databases.

1 Introduction

A central task in computer vision module is the recognition of objects from various images of the environment where the machine is found [1].

Model-based object recognition is the domain where a model exists for every object in the recognition system's universe of discourse. The research emphasis in this paradigm has historically been on the design of efficient matching algorithms from a manually designed feature set with hand-crafted shape rules [2], [3], [4], [5].

Manually designing a feature set is appealing because such a feature set is very efficient. When designed properly, a very small number of parameters for each of the objects is sufficient to capture the distinguishing characteristics among the objects to be recognized. This premeditated efficiency is bitter sweet, however, in that generalization of the features to objects other than those for which they were designed is usually impossible. For example, parameters painstakingly tuned to efficiently discriminate between persons based on intereye distance will be useless in differentiating a car from a fire hydrant.

An alternative to hand-crafting features is the self-organizing approach, in which the machine will automatically derive what features to use and how to organize the knowledge structure such as the work of the Cresceptron [6] and eigenfaces [7] for view-based recognition. In this framework, the recognition phase of the system is preceded by a learning phase. The learning phase focuses on the *methods* by which the system can automatically organize itself for the task of object recognition, giving it a wide range of generality [8], [9]. Self-organizing object recognition systems are open-ended, allowing them to learn and improve continuously [10]. A large amount of work has been published in the domain of adaptation and learning using networks (e.g., [11], [12]).

Allowing the system to organize itself, however, raises some important efficiency issues. The first one is the feature selection issue. If the sample distribution is known, adding more features always produces better results (or at least not worse results) if the Bayesian estimation is used. However, typically these distributions are not known or too computationally expensive to estimate adequately. The result is the "curse of dimensionality"—more features do not necessarily imply a better classification success rate. For example, the principal component analysis (LPA), also known as the Karhunen-Loève projection and "eigenfeatures," has been used for face recognition [13], [7] and lip reading [14] among others. An eigenfeature, however, may represent aspects of the imaging process which are unrelated to recognition (for example, the illumination direction). An increase or decrease in the number of eigenfeatures that are used does not necessarily lead to an improved success rate.

The second issue is how the system should organize itself. Given a k-dimensional feature space with n objects, a linear search is impractical since each recognition probe requires $O(n)$ computations.

• *D. Swets is with the Computer Science Department, Augustana College, 2001 S. Summit Ave., Sioux Falls, SD 57197. E-mail: swets@inst.augie.edu.*
• *J. Weng is with the Computer Science Department, Michigan State University, East Lansing, MI 48824. E-mail: weng@cse.msu.edu.*

Manuscript received 26 Feb. 1996; revised 9 Nov. 1998. Recommended for acceptance by R. Picard.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107701.

So how does a self-organizing system automatically find the most useful features in the images? How can this be done without restricting the domain to which the system will be applicable? How can the system learn and recognize a huge number of objects (say a few million) without slowing the process down to a crawl? The work described here addresses these crucial issues. We discuss how to automatically find the most discriminating features for object recognition and how a hierarchy can be utilized to achieve a very low computational complexity in the retrieval phase. Our goals in this regard include the hierarchical decomposition of a large, complex problem into smaller, simpler problems. From a feature space tessellation standpoint, this involves the decomposition of a highly complex problem with nonlinear boundaries into simpler, smaller, lineally separable problems. At the same time, this hierarchical organization provides for an efficient retrieval mechanism, and produces approximately an $O(\log n)$ algorithm for image retrieval. To find a class to which a test probe belongs is typically a linear algorithm in the number of database items. The SHOSLIF-O uses the hierarchy that decomposes the problem into manageable pieces to provide approximately an $O(\log n)$ time complexity for an image retrieval from a database of n objects.

The SHOSLIF tree shares many common characteristics with the well known tree classifiers and the regression trees in the mathematics community [15], the hierarchical clustering techniques in the pattern recognition community [16], [17] and the decision trees or induction trees in the machine learning community [18]. The major differences between the SHOSLIF tree and those traditional trees are the automatic derivation of features and the direct computation of the most discriminating features at each internal node. SHOSLIF automatically derives features directly from training images,[1] while all the traditional trees work on a human pre-selected set of features. This point is very crucial for the completeness of our representation. In addition, the traditional trees either search for a partition of the corresponding samples to minimize a cost function at each internal node (e.g., ID3 [18] and clustering trees [17]), or simply select one of the remaining unused features as the splitter (e.g., the $k - d$ tree) [19]. The first option results in an exponential complexity that is far too computationally expensive for learning from high-dimensional input like images. The second option implies selecting each pixel as a feature, which simply does not work for image inputs (in the statistics literature, it generates what is called a dishonest tree [15]). The SHOSLIF directly computes the most discriminating features (MDF), using the Fisher's multiclass, multidimensional linear discriminant analysis [20], [21], [22], for recursive space partitioning at each internal node.

The SHOSLIF uses a tree structure to organize search hierarchically. The publications on decision trees is extremely rich. The reader is referred to some survey articles [23], [24], [25]. Decision trees have been traditionally used for making decisions in a vector space of a relatively low dimensionality, where each dimension corresponds to a human-defined feature [26]. Univariate trees (where each decision node uses only one feature component) have been used most frequently. However, oblique trees (where each decision node uses a linear combination of feature components) have been proposed quite early [27], [28]. In the work presented here, we apply a new hierarchical statistical partition scheme directly to samples in the image space, resulting in an oblique tree structure. In so doing, we face new problems that are not present in classical pattern recognition methods. These problems are caused by the high dimensionality of the image space and the number of samples, which is typically much smaller than the dimensionality. In this paper. we will present how SHOSLIF automatically finds desired hierarchical subspaces from such a high-dimensional image space.

In this work, we require "well-framed" images as input for training and query-by-example test probes, as in [13], [29]. By well-framed images we mean that only a small variation in the size, position, and orientation of the objects in the images is allowed. The automatic selection of well-framed images is an unsolved problem in general. Techniques have been proposed to produce these types of images, using, for example, pixel-to-pixel search [7], hierarchical coarse-to-fine search [6], or genetic algorithm search [30]. This reliance on well-framed images is a limitation of the work; however, there are application domains where this limitation is not overly intrusive. In image databases, for example, the human operator will preprocess the image data for objects to store in the database.

Furthermore, the method is view-based to deal with images directly rather than using other sensing modalities of limited applicability (such as range scanners). And although the system can handle multifarious variations without system modification, these variations must be covered in the training phase; many views are required as training input for nontrivial problems. Although image retrieval issues are fundamentally object recognition issues, most object recognition systems contain a reject option. The system described in this paper does not implement such an option. For image retrieval, it is desirable to present a few top matches for the human operator to select or reject. For object recognition, the system could learn a threshold that defines an acceptable level of response from the database; such an automatic rejection option has not been investigated in the work reported here.

2 The Self-Organizing Hierarchical Optimal Subspace Learning and Inference Framework (SHOSLIF)

The SHOSLIF uses the theories of optimal linear projection to generate a hierarchical tessellation of a space defined by the training images. This space is generated using two projections: a Karhunen-Loève projection to produce a set of *Most Expressive Features* (MEFs), and a subsequent discriminant analysis projection to produce a set of *Most Discriminating Features* (MDFs). The system builds a network that tessellates these MEF/MDF spaces to provide approximately an $O(\log n)$ complexity for recognizing objects from images.

1. We do not use the term *feature selection* here because it means to select from several predetermined feature types, such as edges or area. Also, the term *feature extraction* has been used for computation of selected feature type from a given image. *Feature derivation*, on the other hand, means automatic derivation of the actual features (e.g., eigenfeatures) to be used based on learning samples.

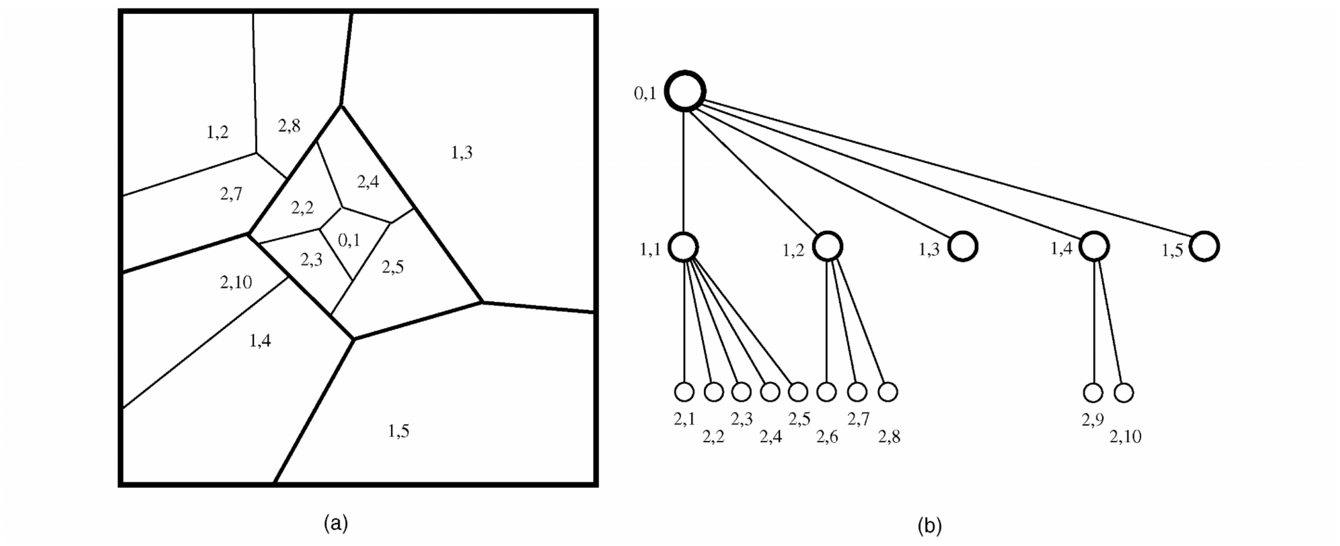


Fig. 1. (a) A sample partitioning of the feature space; (b) The tree structure associated with the tessellation shown. Each cell in the partition does not need to cover a meaningful class. Each cell operates in a different feature space, and the leaf nodes give a final tessellation. This setup can approximate virtually any complex decision region, and provides a logarithmic retrieval complexity.

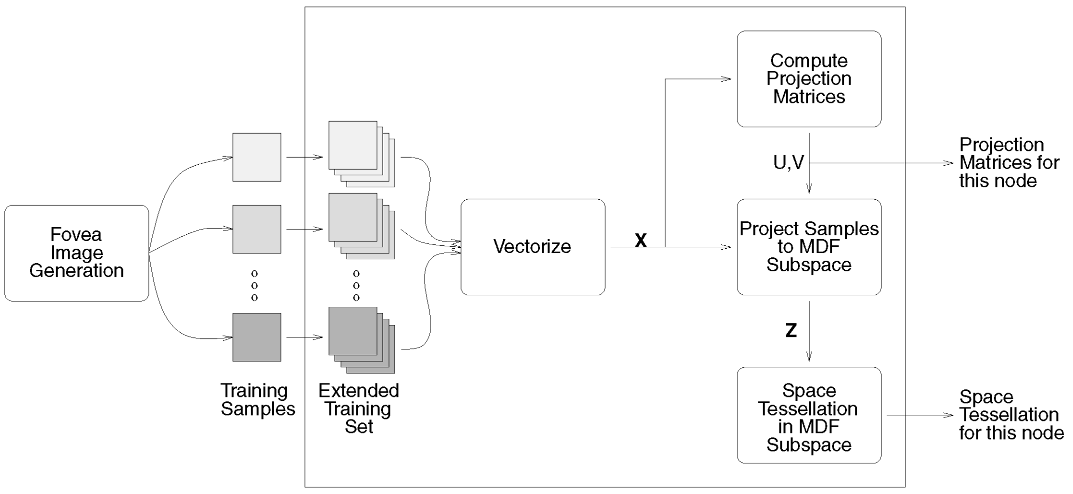

Fovea Image Generation
Training Samples
Extended Training Set
Vectorize
X
Compute Projection Matrices
U,V
Projection Matrices for this node
Project Samples to MDF Subspace
Z
Space Tessellation in MDF Subspace
Space Tessellation for this node

Fig. 2. A top-level flow of the processing performed at each node in the Space-Tessellation Tree during the training phase. A set of training samples which enter the processing element are extended to allow for learning-based generalization for position, scale, and orientation. These extended samples are vectorized and used to produce the projection matrices to the MEF and MDF subspaces. The extended samples are projected to the MDF subspace using these matrices, and a tessellation of the space covered by the node being worked on is produced. The projection matrices and the space tessellation for each node are produced in the learning phase.

2.1 System Overview

The network that is constructed takes the properties of an abstract tree structure. An example of such a tree is shown in Fig. 1. It is this *Space-Tessellation Tree* that provides the key to the efficient object recognition capability of the system described in this work. As the processing moves down from the root node of the tree, the Space-Tessellation Tree recursively subdivides the training samples into smaller problems until a manageable problem size is achieved. When a test object is presented to a node, a distance measure from each of the node's children is computed to determine the most likely child to which the test object belongs. At each level of the tree, the node that best captures the features of the test object is used as the root of the subtree for further refinement, thereby greatly reducing the search space for object model matches.

A top-level flow diagram for the processing done in each of the Space-Tessellation Tree's processing elements during the learning phase is given in Fig. 2. In order to minimize the limitation of our work to “well-framed” images, we want to allow for some variations in the position, scale, and orientation of the objects in the training samples. This can be accomplished either through more image acquisition, but that is expensive in terms of time, storage, and cost. The images this system receives provides an attention point and scale to be used to extract a fovea image of the object of interest. Rather than extracting just a single fovea image from this attention point and scale, a family of fovea images are generated by varying the attention point and scale from the supplied points. This will allow the system to learn some measure of positional and scale variation in the training set.

2.2 Background

The SHOSLIF utilizes two derived feature sets: the Most Expressive Features (MEFs) and the Most Discriminating Features (MDFs) [31].

2.2.1 The Most Expressive Features (MEF)

Each input subimage can be treated as a high dimensional feature vector by concatenating the rows of the subimage together, using each pixel as a single feature.

We can perform Principal Component Analysis on the set of training images [7], [32], [17]. This Principal Component Analysis utilizes the eigenvectors of the sample scatter matrix associated with the largest eigenvalues. These vectors are in the direction of the major variations in the samples, and as such can be used as a basis set with which to describe the image samples. Because they capture the major variations in the training data, they can express the samples well and can approximate the samples, where the reconstruction is very close to the original.

This LPA projection, also called the Karhunen-Loève projection, has been used to represent (e.g., Kirby and Sirovich [33]) and recognize face images (e.g., Pentland et al. [7], [13]), for planning the illumination of objects for future recognition tasks (e.g., Murase and Nayar [29]), and in a lip reading system (e.g., Bregler and Omohundro [14]), among others. Since the features produced in this projection give the minimum mean-square error for approximating an image [22], [34], [35] and show good performance in image reconstruction [33], we call them the *Most Expressive Features* in contrast to the Most Discriminating Features described below.

2.2.2 The Most Discriminating Features (MDF)

Although the MEF projection is well-suited to object representation, the features produced are not necessarily good for discriminating among classes defined by the set of samples. The MEFs describe some major variations in the set of sample images, such as those due to lighting direction; these variations may well be irrelevant to how the classes are divided.

If a labeling scheme is available for the training images, linear discriminant analysis (LDA) [17] can be performed, as in [36]. In LDA, the between-class scatter is maximized while minimizing the within-class scatter. In other words, the samples for each class are projected to a space where each class is clustered more tightly together, and the separation between the class means is increased. The features obtained using a LDA projection optimally discriminate among the classes represented in the training set, in the sense of linear transform [37], [21]. They are the eigenvectors of $W^{-1}B$ associated with the largest eigenvalues, where W and B are the within-class scatter and the between-class scatter matrices, respectively. Due to their optimality in discrimination among all possible linear features, we call them the *Most Discriminating Features* (MDF). For a performance difference comparison between the MEF and the MDF spaces, the reader is referred to [38].

The LDA procedure breaks down, however, when the number of samples is smaller than the dimensionality of the sample vectors. This problem can be resolved using the Discriminant Karhunen-Loève projection [38], where the LDA is performed in the MEF space (i.e., the Karhunen-Loève space), where the degeneracy does not occur.

2.3 Space Tessellation

We want to exploit the strengths of the MDF feature set while trying to overcome its limitations. At the same time, we want to provide an effective and efficient method for retrieval of images from the database. To effect this, the SHOSLIF produces a hierarchical space tessellation using the hyperplanes derived by the MDFs. The feature space of all possible images is partitioned into cells of differing sizes as shown in Fig. 1. A cell at level l is subdivided into smaller cells at level $l + 1$. The network structure that can effect this recursive tessellation is a *Space-Tessellation Tree* whose nodes represent cells at the corresponding levels as shown in Fig. 1. The tree is built automatically during the learning phase; this tree is used in the recognition phase to find the model in the learned database that best approximates an unknown image in approximately $O(\log n)$ time.

2.3.1 The Hierarchy Decomposes the Problem

The tree structure is able to decompose the problem of LDA into smaller, tractable problems. At the root node of the tree, where the entire database of samples are found, the classes may not be separable using the LDA technique. But because we do not attempt to completely separate the classes at a single level, we can successfully solve the problem in stages by breaking it down into simpler pieces.

The *DKL* projection does its best in separating classes, even for the case of many classes. When the database contains many classes, however, they may not be linearly separable. The Space-Tessellation Tree provides a mechanism for dealing with this problem. Children of a particular node decompose the difficult problem of separating many classes into several smaller problems. At each node of the tree, the set of features found in the LDA procedure are specifically tuned to the set of samples found in the node. So although the MDF space provides an optimal set of features for class selection in the sense of linear transform, this optimal set may be insufficient to separate classes. But even if a node cannot completely separate the classes, it can make a first attempt at separation, dividing the samples among its children nodes. Then at this child level, since fewer samples exist, the LDA procedure is more likely to succeed. Since this is applied recursively, eventually a successful separation of classes is achieved as shown in Fig. 5.

Fig. 3 shows an example of the difference in the complexity of the class separation problem for the root node and an internal node of the tree. A child node contains fewer samples than its parent does, and the MDF vectors can, therefore, be optimized to the smaller set of samples in the child node.

2.3.2 Hierarchical Quasi-Voronoi Tessellation

We tessellate the space covered by a node N using a *Hierarchical Quasi-Voronoi Tessellation*, with the resulting tesselated space shown in Fig. 4b for two dimensions. The Voronoi Tessellation as shown in Fig. 4a indicates retrieval of the nearest sample point at a single level. However, the nearest neighbor is not necessarily the best sample to retrieve

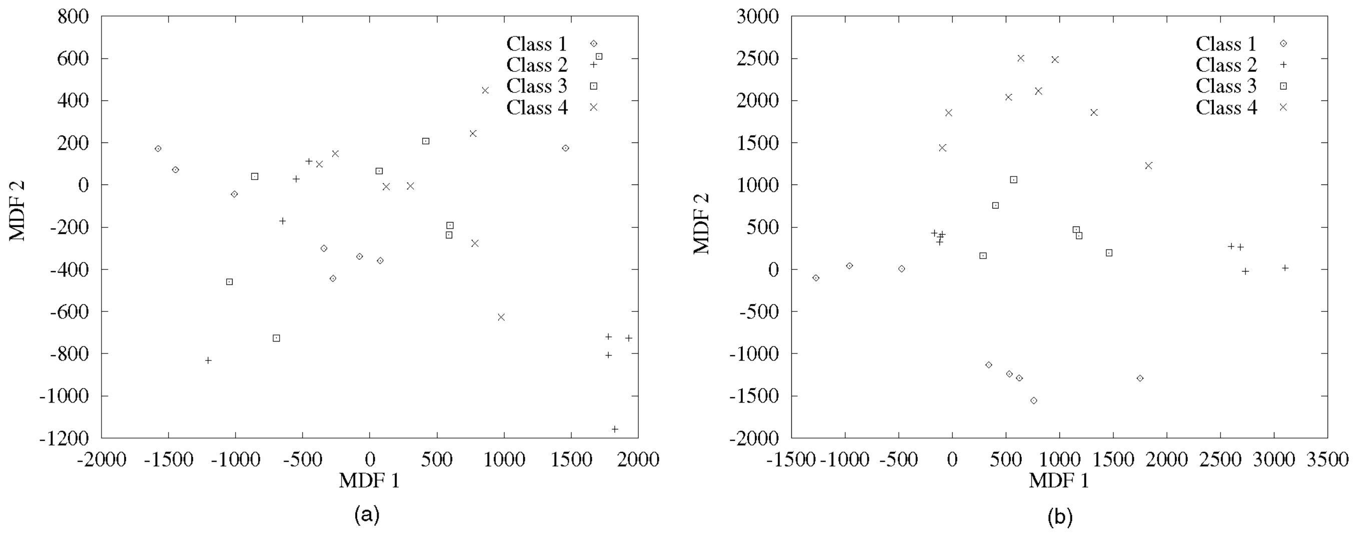


Fig. 3. An example showing the complexity of the class separation problem at two different levels of the tree. Sample data points that belong in the space defined by both the root node and an internal tree node are shown. The same samples were used in both cases to demonstrate the fact that the internal node has an easier time separating the different classes. Since the internal node contains many fewer total samples than the root node, the MDF vectors can cluster the classes contained in the node well. This figure shows a more effective clustering in (b) than in (a) because the number of samples and classes in (b) is smaller than in (a).

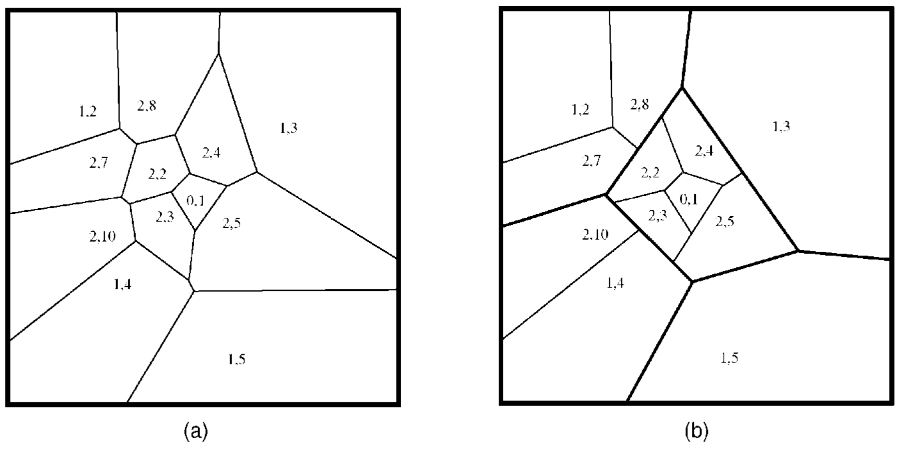


Fig. 4. (a) The Voronoi Diagram (Dirichlet tessellation). Each cell consists of points that are closer to the sample than to any other samples. (b) Hierarchical Quasi-Voronoi Tessellation. The space is recursively partitioned. A subset of all the samples in a cell is used as the points to define the Voronoi Tessellation boundaries within a particular cell; these are then recursively partitioned in like manner. The separation hyperplanes are not computed; they are realized implicitly in finding the nearest cell-centers.

because the class variation is not taken into account. We take this class shape into account when partitioning the space. Fig. 5 shows samples of the hierarchical quasi-Voronoi tessellation using binary trees.

2.3.3 Automatic Tree Construction

Each level of the tree has an expected radius $r(l)$ of the space it covers, where l is the level of the node and $r(l) < 1$ is a decreasing positive function based on the level. $d(\mathbf{X}, \mathbf{A})$ is the distance measure between node N with center vector $\mathbf{A}$ and a sample vector $\mathbf{X}$.

The expected radius (in the MDF space) is to limit the size of the cell so that nonlinear boundary of a large region can be broken into smaller segments. This will ultimately generate segments which are small enough to be approximated well by linear boundaries. The expected radius can—and is even likely—to break a class into different children, and a part of the class can have very few samples in a node. However, this just means that the child covers a smaller part of the class (if the samples are drawn according to the actual applications). The continued recursive partition should be able to isolate the small space out for this class lower in the hierarchy. In order words, locally at one level, the features are linear and the cell boundary is linear. However, globally, the composite effect of using linear features recursively is that the effective decision boundary is nonlinear.

A node N in the tree contains a set of labeled training images. Every node which contains more than a single training image computes a projection matrix V that is used to project the samples to the MEF space as described in Section 2.2.1. If the training samples contained in N are drawn from multiple classes, as indicated by the labels associated with each training sample, then the MEF sample vectors are used to compute a projection matrix W to project the samples to the MDF space as described in Section 2.2.2. Otherwise, the training samples in N are from a single

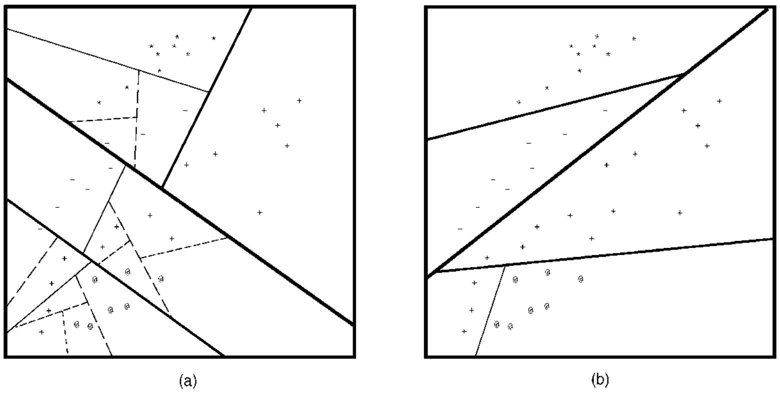


Fig. 5. (a) Binary tree built without class information taken into account, as would be built using the MEF space. (b) Binary tree built optimized to separate classes, as would be built using the MDF space. The MDF typically yields a smaller tree than the MEF space provides. The MDF is effective if the samples cover all the within-class variations. The samples of a class are denoted by a single type of character.

class, and we organize them into a subtree using MEFs for efficient image retrieval.

The MEF subtree is used to find the nearest neighbor for a test probe. Once that nearest neighbor is found, that sample is projected back to the nearest ancestor node that utilizes an MDF space. This is done because the center of the nearest MDF ancestor node may not be very near the test probe in this MDF space. But it represents a class of objects that contains a vector that is near the test probe. Therefore, the MEF subtree is used to find that nearest neighbor in order to compare the test probe with nearest neighbor sample using the most specific MDF subspace.

The tree is built one level at a time. The collection of children nodes represent a tessellation of their parent; this tessellation must be established before the samples are assigned to a child node.

Suppose that we want to add training sample $\mathbf{X}_i$ to node N which is at level l. If the feature vector for X_i is within the radius covered by one of the children of N, then $\mathbf{X}_i$ will be added as a descendent of that child as we increase the depth of the tree. If the feature vector for $\mathbf{X}_i$ is outside the expected radius for all the children of N, however, we would like to add $\mathbf{X}_i$ as a new child of N, to contribute to the tessellation of the space subtended by N. The algorithm is summarized in Algorithm 1.

Algorithm 1. *The Hierarchical Quasi-Voronoi Tessellation Algorithm.*

```
Input: Node $N$ at level $l - 1$, list of samples $\mathbf{X}$ to add.
Output: A tessellation of N based on the new samples.
  1) Compute the project matrices $V$ and $W$ to the MEF and MDF subspaces for this node.
  2) For each sample $X_i$
       • Project $X_i$ to the MEF space to get $Y_i$.
       • Project $Y_i$ to the MDF space to get $Z_i$.
       • If $d(Z_i, C_j) > r(l)$ for all $C_j$ children of $N$, add $Z_i$ as the center vector for a new child of $N$.
  3) For each feature vector $Z_i$, add $Z_i$ to the child $C_j$ with the nearest center vector.
  4) For each child $C_j$ of $N$, perform the space tessellation.
```

This algorithm is called the *Hierarchical Quasi-Voronoi Tessellation Algorithm* because the space of each node is partitioned into a Quasi-Voronoi Tessellation in a hierarchical manner. A characterization of a sample tree built using this algorithm is shown in Table 1. The generated tree is too large to display in detail, so a characterization of the tree showing the number of nodes produced on each level is given here.

2.3.4 Properties of the Hierarchical Quasi-Voronoi Tessellation Algorithm

The Hierarchical Quasi-Voronoi Tessellation Algorithm has several favorable properties. Theorem 1 uses the fact that both the dimensionality of the sample vectors and the expected radius that is used to create a node N's children are constants to assert that the maximum number of children that N can produce is bounded above by a constant κ, irrespective of the training samples used.

Before explaining the bound on the number of levels, we must introduce a concept.

DEFINITION 1. *Given n samples, a Bounded Unbalanced Tree with Unbalance Bound* $0 < \alpha < 1$ (α *constant*) *is a tree such that for any node N containing* $n_1 + n_2 + \cdots + n_k$ *samples, where N has k children with* n_i *samples assigned to node i and* $n_1 \geq n_2 \geq \cdots n_k$, $n_1 \leq \alpha(n_1 + n_2 + \cdots n_k)$.

The MEF and MDF used in the tree-building tend to produce a balanced tree as much as possible. This is the case

TABLE 1
CHARACTERIZATION OF A SAMPLE TREE BUILT USING THE HIERARCHICAL QUASI-VORONOIS TESSELLATION ALGORITHM

Level	Node count	Level	Node count
0	1	1	9
2	32	3	93
4	256	5	471
6	491	7	310
8	113	9	25
10	2	Total:	1803

The table lists the number of nodes found on each level of the tree. The tree is too large to show in detail.

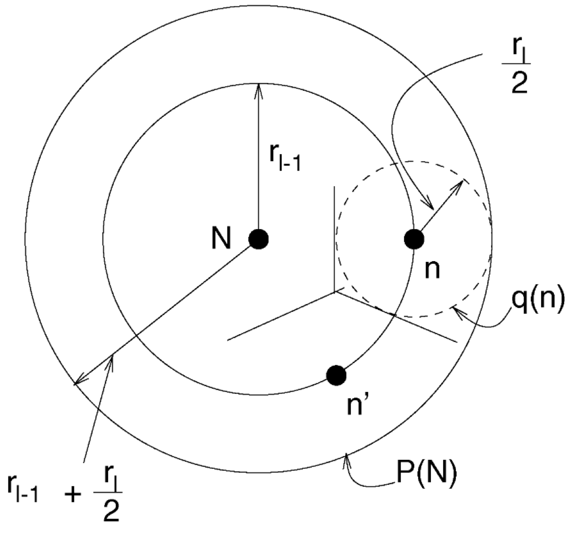


Fig. 6. Hyperspheres used in Theorem 1 proof.

because the MEF and MDF attempts to partition the samples in terms of the statistics of the distribution. The above definition limits the degree of unbalancedness. If Algorithm 1 produces a Bounded-Unbalanced tree, then Lemma 1 proves that there are $O(\log n)$ levels. Note that there is no proof that the trees produced by Algorithm 1 must indeed be Bounded-Unbalanced; however, in our studies we have obtained emperical evidence to suggest that they are.

2.4 Image Retrieval

When an unknown image X is presented to the recognition tree, the general flow shown in Fig. 7 is followed. When a node N of the tree is activated, X is projected to $X_{\mathbf{Y}}$, a vector in the MEF subspace of node N. $X_{\mathbf{Y}}$ is then projected to $X_{\mathbf{Z}}$, a vector in the MDF subspace for node N. $X_{\mathbf{Z}}$ is compared with each of N's children. The child with the best response is selected as the path of the tree to be explored.

THEOREM 1. *Given a fixed dimensionality d for the samples and a decreasing positive expected radius function r(l) based on the level of the tree, the number of children that a node N at level $l - 1$ can have is bounded above by a constant κ, regardless of the training set.*

PROOF. Let N be the center vector for a node and n, n' be children of node N. Let $P(N)$ be the hypersphere centered at N with radius $r_{l-1} + (r_l/2)$. Let $q(n)$ be the hypersphere centered at n with radius $(r_l/2)$. Then $q(n) \cap q(n') = \emptyset$ if $n \neq n'$. Otherwise if these hyperspheres overlap, then one of $\{n, n'\}$ will be the child of the other by Algorithm 1. See Fig. 6 for a 2D example of these hyperspheres. $P(N)$ is the extended space of N, i.e., the space that N might have to cover. $q(n)$ is the hypersphere that do not overlap with $q(n')$ for any children n, n' of N, since $d(n, n') \geq r_l$ by Algorithm 1. Note that $P(N)$ contains all $q(n)$, for all children n of N.

Since the volume of a d-dimensional hypersphere of radius R is $(2^{d-1}/d)\,\pi\, R^d$, the volume of $P(N)$ is $V_p = (2^{d-1}/d)\,\pi \leq (r_{l-1} + (r_l/2))^d$. Likewise, the volume of $q(n)$ is $V_q = (2^{d-1}/d)\,\pi \leq (r_l/2)^d$. Since $q(n)$ and $q(n')$ do not overlap, and $P(N)$ contains all $q(n)$, then the number of $q(n)$s, which is also the number of children of N, is bounded above by constant

$$\kappa = \frac{V_p}{V_q} = [\frac{r_{l-1} + (r_l / 2)}{(r_l / 2)}] = [1 + \frac{2r_{l-1}}{r_l}]^d . \quad (1)$$

□

COROLLARY 1. *If $r_{l-1} = \alpha r_l$, $\alpha > 1$, then the number of children of any node is not larger than $(1 + 2\alpha)^d$.*

In practice, we would like to present a number of top matches instead of only one. Moreover, a single-path search will guarantee that an exact match (i.e., when the input is the same as a sample in the database) can always be found, but it cannot guarantee that a nearest match in the subspace

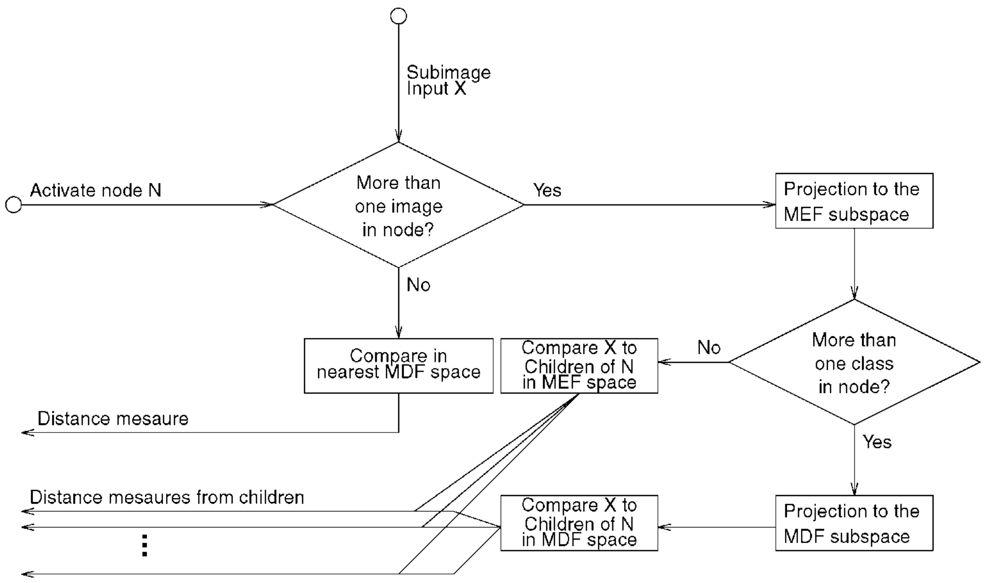

Subimage Input X
Activate node N
More than one image in node?
Yes
Projection to the MEF subspace
No
Compare in nearest MDF space
Compare X to Children of N in MEF space
No
More than one class in node?
Yes
Distance mesaure
Distance mesaures from children
Compare X to Children of N in MDF space
Projection to the MDF subspace

Fig. 7. The general flow of each SHOSLI Processing Element.

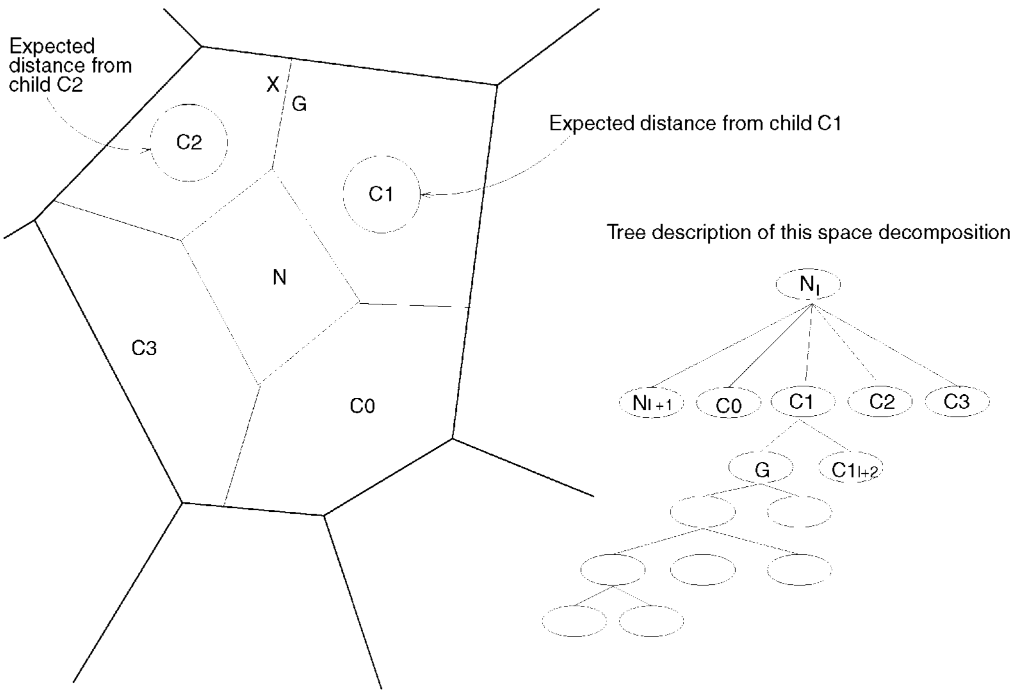

Expected distance from child C2
X
G
C2
C1
Expected distance from child C1
N
C3
C0
Tree description of this space decomposition
Nl
Nl+1
C0
C1
C2
C3
G
C1l+2

Fig. 8. A tessellation in which a single-path search may not find a nearest match. Node *N* at level l will shuttle test proble *X* to its nearest child *C*2. The nearest neighbor for *X* may well be found in the subteee of *G*, which is missed in a single-path search.

will always be found. As shown in Fig. 8, the best match for *X* may be found in the subtree rooted at *G*, but a single-path search will explore only *C*2 and its subtree. If the nearest neighbor lies in the subtree rooted at *G*, it will be missed in a single-path search.

Therefore, we define a constant *k* of nodes to be explored at every level of the tree. Using a constant *k* at every *level* ensures that only a band of the Space-Tessellation tree is explored, that band surrounding the nearest leaf node match of a test probe.

The retrieval algorithm is given in more detail by Algorithm 2. Theorem 2 uses the results from Theorem 1 (Fig. 6) and Lemma 1 to show that this Image Retrieval algorithm runs in $O(\log n)$ time when the tree is Bounded-Unbalanced.

2.5 SHOSLIF-O Options for Automatic Space Tessellation

The SHOSLIF-O (SHOSLIF for object recognition and image retrieval) has several options that need to be specified in order to function. These include the distance measure to be utilized for the similarity measure, the number of competing paths explored for each level of the tree, the expected radius, and the number features to utilize at every node.

2.5.1 Distance Measure

In the SHOSLIF tree, every node has its own different MEF and MDF spaces. Given an input, we must compare its match with all the competitive nodes. To handle this, a distance measure must be investigated.

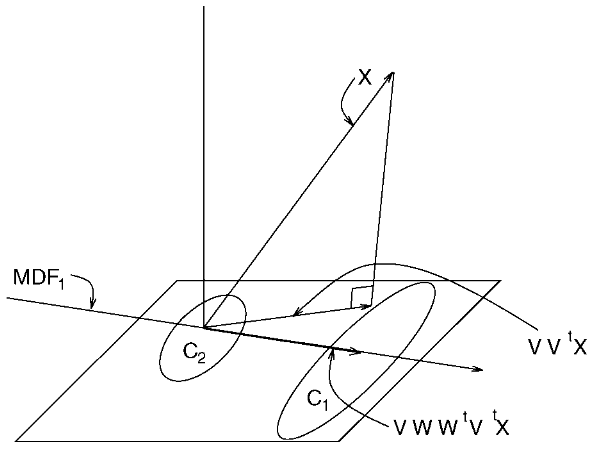
LEMMA 1. *The number of levels in a Bounded Unbalanced Tree with n samples is bounded above by* $\log_{(1/\alpha)} n$ *where* α *is the Unbalance Bound of the tree.*

PROOF. Each node *N* of the tree is assigned with $n_1 + n_2 + \cdots + n_k$ samples, where n_i is the number of samples assigned to the *i*th child of *N*. Rank these n_i's so that $n_1 \geq n_2 \geq \cdots \geq n_k$. Because the tree is a Bounded Unbalanced Tree, we know that $n_1 \leq \alpha(n_1 + n_2 + \cdots + n_k)$, and is true for all nodes *N* of the tree.

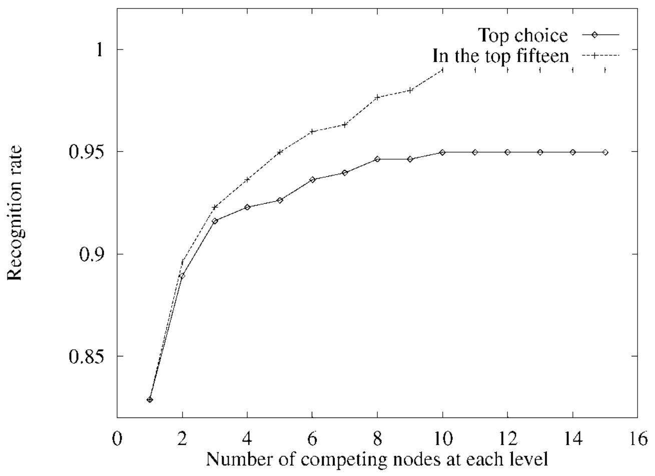
α is a constant. Note that α may be large, i.e., 99 percent, but it is still a constant. This means that each node will not assign a huge portion of samples to a single child. An extreme example would be where child 1 receives $n - 1$ samples and child 2 receives 1 sample. Since the tree is a Bounded Unbalanced Tree, this is not allowed, and in the worst case, child 1 would receive αn samples and child 2 would receive $(1 - \alpha)n$ samples. This is a significant constraint.

Each deeper level of the tree will reduce the number of samples by a factor of at least α. The *l*th level down the tree will receive $n\alpha^l$ samples. At tree height *h*, we have just a single sample by Algorithm 1. Then $n\alpha^h = 1$, and $\alpha^h = (1/n)$, or $(1/\alpha)^h = n$. Then the height of the tree $h = \log_{(1/\alpha)} n = (\log n / \log(1/\alpha))$. □

Distance from Subspace. A simple Euclidean distance in the feature space is insufficient to handle the hierarchical sets of features developed to solve the class separation problem. A test probes that comes into a node would be compared in the local MDF or MEF subspace for that node. It is entirely possible that a test probe vector that was miles away from a particular node's *subspace* would project *into* that subspace very near to the node's center vector. This would cause poor recognition results because test probes that were not at all similar to the node centers would erroneously be considered close.



(a)



(b)

Fig. 9. (a) Distance from subspace description for 3D. The subspace is shown as the plane, and is defined by the matrix V. Two classes are shown in this example, and the MDF vector that can separate them is shown. (b) A comparison of the performance for various k competing paths.

Instead, we need to take into account the distance from the subspace being compared. Each node constructs a different set of subspaces based on the samples contained in that node. The Distance from Subspace (DFS) distance measure takes into account the distance from the *projection space* in addition to the distance of the projection to the node centers.

The DFS distance measure is given by

$$d(X, A) = \sqrt{\left\|X - VV^tX\right\|^2 + \left\|VWW^tV^tX - VWW^tV^tA\right\|^2}$$

where X is the test probe, A is the center vector, V is the projection matrix to the MEF space, and W is the projection matrix to the MDF space. So the product $Y = V^tX$ is the projection of the test probe X onto the MEF subspace; $VY = VV^tX$ represents this MEF projection in the original image space. Likewise, $Z = W^tV^tX$ is the projection of X onto the MDF subspace, and $VWZ = VWW^tV^tX$ represents this MDF projection back in the original image space. Note that computationally, for $Z = W^tV^tX$, $A_Z = W^tV^tA$, and $VW = M$, $\|MZ - MA_Z\|^2 = \|M(Z - A_Z)\|^2 = (Z - A_Z)^tM^tM(Z - A_Z)$. Thus, only a small matrix multiplication need be performed to effect this distance measure. This M^tM matrix can be precomputed during the learning phase and stored in each node so that in the testing phase, the computational complexity of this operation is minimized.

Intuitively, what is being measured can be seen in Fig. 9. The first term under the radical indicates the distance of the original vector from the population (i.e., the subspace). The second term indicates the distance in the subspace from the center vector of a class. We neglected the component $VV^tX - VWW^tV^tX$ since it is the component neglected by the MDFs as unrelated to classification (e.g., lighting). In other words, we do not want to find the nearest neighbor in the Euclidean image space; we want to discount components that are neglected by the linear discriminant analysis. Fig. 5 visually explains why.

Algorithm 2. *The Image Retrieval Algorithm*

Input. Probe $\boldsymbol{X}$, level l, and a list of at most constant k nodes which were explored at level l.

Output: A list of nodes explored at level $l + 1$.

```
1) For each node $N_i$ in the list explored at level $l$:
   If $N_i$ is not a leaf node:
     • Project $X$ to the MDF subspace of node $N_i$, producing $Z$.
     • Compute $d(C_j, \boldsymbol{Z})$ for all children $j$ of $N_i$ with center vectors $C_j$.
     • Transfer at most constant $k$ of the children of $N_i$ to the output list such that those transferred are the $k$ nearest neighbors of $\boldsymbol{Z}$.
2) Truncate the output list to hold at most constant $k$ nodes to explore at the next level.
```

THEOREM 2. *For the k-competing path case, the number of nodes visited in a Bounded Unbalanced Tree is bounded above by $k\kappa\log_{(1/\alpha)} n$, where k is the number of competing paths explored, κ is the upper bound on the number of children that any node can have, α is the Unbalance Bound of the tree, and n is the number of samples in the database.*

PROOF. The proof follows directly from Theorem 1, Lemma 1, and the fact that a constant k nodes are explored for every level of the Bounded Unbalanced Tree. □

2.5.2 Use of Multiple Competing Paths

As described in Section 2.4, in order to find a nearest neighbor match in the recognition tree, it may be necessary to explore more than a single path of the tree. This constant k parallel paths to explore is a parameter that the system operator must determine. For the data sets used in this work, unless stated otherwise, we chose $k = 10$ based on the performance graph given in Fig. 9b. Based on the data shown in this graph, the performance of the system levels out at $k = 10$. We found that $k = 10$

TABLE 2
RESULTS OF SUBSPACE COMPARISON STUDY

		Flat database	Single subspace with tree	Multiple subspaces with tree
Image space	Correct class is top choice	90%	86%	86%
	Correct class is in top 15	100%	90%	90%
MEF space	Correct class is top choice	90%	76%	86%
	Correct class is in top 15	100%	83%	90%
MDF space	Correct class is top choice	90%	89%	97%
	Correct class is in top 15	100%	100%	100%

For the "flat" databases, only a single level of the tree was createaed. For the single projection databases, the MEF/MDF projection matrices were only computed at the root node, and the same projection matrices were used throughout the tree. For the multiple projection trees, a new projection was made at ech node of the tree.

gives good runners-up for best matches at a low computational cost.

2.5.3 Expected Radius

Node N uses the expected radius $r(l)$ to determine when it needs to create a new child to accommodate a training sample. $r(l)$ indicates the size of the cells that we would like to achieve at level l. This expected radius is a decreasing positive function based on the level of the tree. For this work, we have chosen $r(l) = 1.3^{36-l}$ for node N on level l of the Space-Tessellation Tree. This number was chosen because at the root node of the tree, i.e., at level $l = 0$, the expected radius is 12,646.219. This radius is large enough to cover all of the samples in the data sets utilized in this work.

The value 1.3 was used as a base because it gave favorable results in the shape of the tree. A tree that is too wide and fat produces inefficiencies because many children must be explored at a high level of the tree; a tree that is too thin and tall produces inefficiences because many projections must be done in order to find the nearest neighbor in the database.

3 EXPERIMENTAL RESULTS

In order to verify the proper functionality of the system, we experimented with a large multifarious set of real-world objects found in natural scenes. In this section, we demonstrate the ability of the MDF space to tolerate within-class variations and to discount such imaging artifacts as lighting direction, and show how the tree structure provides the ability for view-based recognition.

The images utilized for these experiments used a standard fovea size of 88 × 64 pixels. So when vectorized, the dimensionality of the original input vectors were $d = 5{,}632$. When projected to the MEF space, utilizing 95 percent of the variance, at the root node, typically between 30 and 60 principal components were utilized. Then when projected to the MDF space, the number of discriminating vectors utilized were limited to 5 or 95 percent of the variance for the samples contained in the node, whichever was less.

3.1 Space Comparison

Since the analysis showed that the MDF space should perform better than the MEF space (or the image space directly), a study was performed to demonstrate this fact. Furthermore, though the hierarchy described provides efficiency to both the learning and retrieval phases, the recognition performance may suffer somewhat from not examining all possibilities in the database. Therefore, a study was also performed to examine the performance difference between using a "flat" database, in which all images in the database are examined for possible matches, and the described hierarchical database. Finally, the recognition performance will be dependent on whether a single MDF projection is used, performing the space tessellation in this single space; or if a new MDF projection is performed at each node of the tree. A study examining the performance between these two modes was also done.

The training images come from a set of real-world objects in natural settings. At least two training images from each of 38 object classes were provided for a total of 108 training images; a disjoint set of test images were used in all of the tests. For each of the tests performed, the identical set of training images and test images were used.

For those tests where subspaces are used, the Distance From Subspace (DFS) distance metric was used, and 15 nodes were explored at each level when using the tree structure.

3.1.1 Effects of the Feature Spaces and the Tree Hierarchy

The results of the studies are summarized in Table 2.

As expected, the data in Table 2 shows the MEF subspace tracing the same sorts of responses that the original image space produces. This is expected because the MEF subspace is a good representation of the original image space, and can, therefore, be used as a compact means for storing images.

The data also shows, however, that the MDF subspace can outperform the MEF subspace. This is also expected, since the MDF subspace utilizes more information supplied by the user in the form of the image labels. The MDF performance for a flat database shows the same results as the MEF performance. This could be due to the fact that there was not enough within-class variation in the training samples that were needed to catch the variation present in each class. But it may be more likely that the classes in a flat database were not lineally separable. This is supported by the data, because when a single projection is done and a space tessellation tree produced based on that single projection, the rate drops off a little; when multiple projections are done throughout the tree, the rate improves significantly. The data lends credence to this claim that the multiple projections in the tree do indeed reduce the problem to a smaller, more manageable size, and can, therefore, successfully separate the classes more easily as the processing moves down the tree. The MDF subspaces generated at the various nodes of the tree are adaptive in that they optimally separate the classes for the samples contained in the node

TABLE 3
SUMMARY OF EXPERIMENT ON A FACE DATABASE OF 1,042 IMAGES (384 INDIVIDUALS)

Number of training images	1042 of 384 individuals
Number of nodes in generated tree	1761
Number of nodes expanded at each level	10
Average number of nodes explored per probe	102
Re-substitution:	
Correct retrieval is top choice	100.0%
Correct retrieval is second choice	98%
Disjoint test set:	
Number of test images	246 of 246 individuals
Correct retrieval is top choice	95.5%
Correct retrieval is in top 10	97.6%

For resubstitution each training image was given as a test probe. For the disjoint test set, a list of 246 images not found in the training set was used for testing.

TABLE 4
SUMMARY OF LARGE NATURAL SCENE EXPERIMENT

Number of training images	1316 from 526 classes
Number of test images	298 from 298 classes
Number of nodes in generated tree	2388
Number of nodes expanded at each level	10
Average number of nodes explored per probe	41.6
Correct retrieval is top choice	95.0%
Correct retrieval is in top 10	99.0%

The training images were drawn at random from the pool of available images, with the remaining images serving as a disjoint set of test images.

in the sense of linear transform. As the processing moves along the nodes of the tree, a different set of features tuned specifically for those samples contained in the node are utilized for subclass selection.

3.2 Face Database

In order to compare with the results of others in the community utilizing eigenfeature methods, a test was performed on a database comprised of only faces.

The face database was organized by individual; each individual had a pool of images from which to draw training and test data sets. Each individual had at least two images for training with a change of expression. The images of 38 individuals (182 images) came from the Michigan State University Pattern Recognition and Image Processing laboratory. Images of individuals in this set were taken under uncontrolled conditions, over several days, and under different lighting conditions. Classes of 303 (654 images) came from the FERET database. All of these classes had at least two images of an individual taken under controlled lighting, with a change of expression. Twenty-four of these classes had additional images taken of the subjects on a different day with very poor contrast. Sixteen classes (144 images) came from the MIT Media lab under identical lighting conditions (ambient laboratory light). Twenty-nine classes (174 images) came from the Weizmann Institute, and are images with three very controlled lighting conditions for each of two different expressions.

In this experiment, when an image that was used for training was also used as a test probe, such as is done with Photobook [39], [13], [40] (i.e., resubstitution method), the SHOSLIF always retrieved the correct image as its first choice 100 percent of the time. The second image retrieved on a database of 1,042 face images was a correct match for 98 percent of the test probes using the resubstitution method, which is comparable to the Photobook [40] response rate on a different data set. Table 3 shows a summary of the results obtained both by resubstituting the training samples as test probes and by using a disjoint set of images for testing.

3.3 Combination Database: Faces and Other Objects

We have trained the system on a wide range of scenes, in order to demonstrate the utility of the hierarchical methodology.

A small sample of images from the classes learned is given in Fig. 10. Most classes in the database were represented by two images, and 19 percent of the classes had three or more images, up to 12 for some objects (e.g., fire hydrant). Each image consisted of a well-framed object of interest. The different images from each class were taken either in a different setting or from a different angle; where possible a change in the lighting arrangement was used to provide variation in the training images.

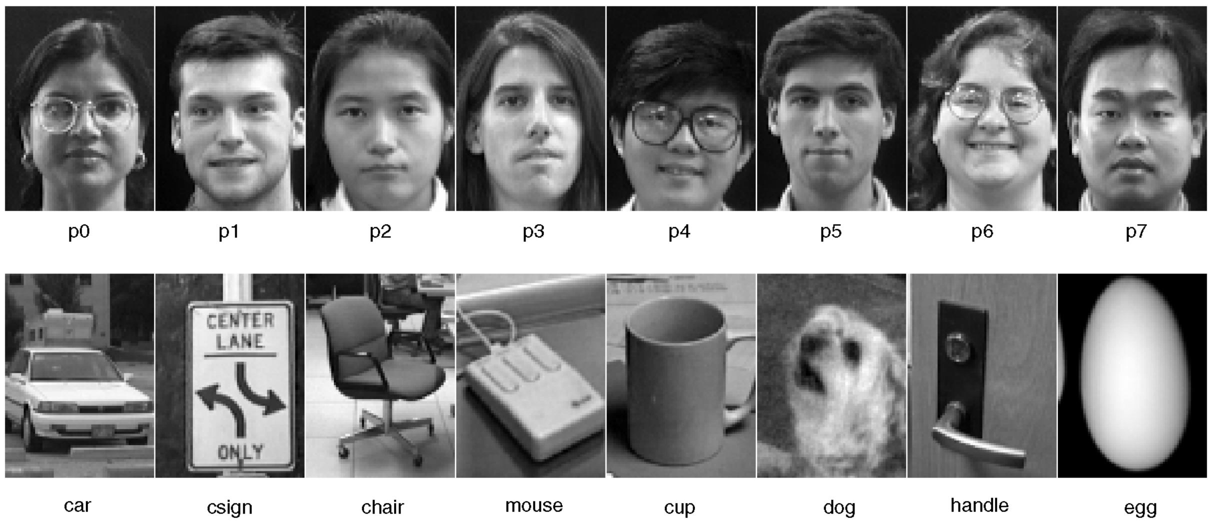


Fig. 10. Representative images from the difference classes.

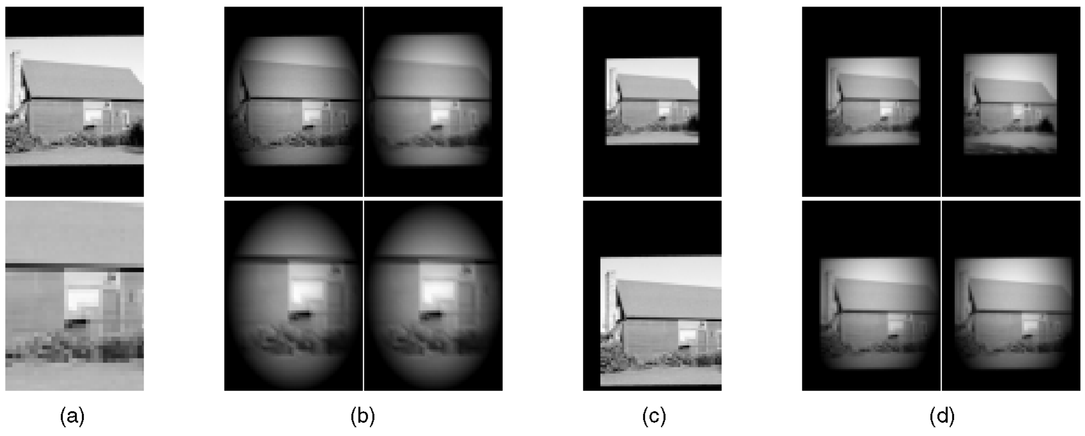


Fig. 11. Generalization for size and position of objects. The search probes were synthetically generated from images in a disjoint test set. Each search probe retrieved an image from the appropriate class. The scaling shown is 50 percent of the fovea size (both positively and negatively); the position change shows 12 percent negative change in the fixation point for both the row and the column coordinates. (a) Search probes. (b) Retrieved images. (c) Search probes. (d) Retrieved images.

Following training, the system was tested using a test set completely disjoint from the training set of images. A summary of the results are shown in Table 4.

The instances where the retrieval failed were due in large part to significant differences in object shape and three-dimensional (3D) rotation.

3.4 Handling 2D Variation

In order to alleviate some of our reliance on well-framed images, we want to train the system on a set of images which contains some variations in the position, scale, and orientation of the objects of interest. This can be accomplished by greatly increased image acquisition for points surrounding the fixation point and scale provided. The increase in image acquisition is expensive in terms of time and storage space, however, and could instead be accomplished by extending the training set for a grid of points surrounding the extracted fovea image in terms of the position, scale, and 2D orientation foviation parameters. Each fovea image comes from some attention mechanism that specifies a fixation point and a scale. This fixation point can be overlaid with a grid in both the position and the scale parameters, and a new fovea image extracted from the original image at each of these grid points. Thus more variation due to size and position can be handled by extracting a set of fovea images to add to the Space-Tessellation tree for each grid point surrounding the attention point and scale instead of just extracting a single image.

Fig. 11 demonstrates the variability that the system can handle by extending the training set in this manner. When the training set is thus extended, the Space-Tessellation tree grows in size, but not by the factor of the increased number of samples. If we assume that the search tree provides $O(\log s)$ levels for s training samples, then the tree has $\kappa \log s$ levels for some constant κ. Now if the number of training samples is expanded by $\mathcal{P}$ to handle the positional variation, and by $\mathcal{S}$ to handle the scale variation, then we have a total of $\mathcal{PS}s$ samples to put into a tree. Then there will be $\kappa \log \{\mathcal{PS}s\} = \kappa \log \mathcal{P} \log \mathcal{S} \log s = \kappa' \log s$ levels in the Space-Tessellation Tree, for constant κ', which is still $O(\log s)$ levels in the tree. Typical values for $\mathcal{P}$ will be $\mathcal{P} = 9$ or 25 for a 3×3 or a 5×5 grid surrounding the attention point, respectively; $\mathcal{S}$ will typically be $\mathcal{S} = 3$ or 5 to deal with the various scales. We have tested this approach to handling positional and scale variation on two different data sets.

3.4.1 Handling Scale

The first data set is the full combination data set that described in Section 3.3 with more than 1,300 original training images. We ran the experiment using P = 1 (i.e., on the attention mechanism's attention point alone) and $\mathcal{S} = 3$ (i.e., 3 grid points surrounding the attention mechanism's scale). For training, we set the scale span to be 30 percent of the fovea size. That is, each grid point represented a 15 percent change in the fovea size, one training image at a 15 percent smaller scale than the attention mechanism dictated, one at the attention mechanism's specified scale, and one at 15 percent larger than the attention mechnism's specified scale. For testing purposes, we took the disjoint test set described in 3.3 and genrated a set of test images from this set with a random scale change in ther ange of [–0, +20] percent of the fovea size. The characterization and results of this test is shown in Table 5. The table shows the difference between the perofrmance when the scaling was built into the training phase and when it was not. As can be seen from the data, for

TABLE 5
SUMMARY OF THE SCALE GENERALIZATION EXPERIMENT ON THE LARGE COMBINATION FACE AND OTHER OBJECT DATA SET

Num of training images	1316 (expanded to 3948)
Position	0% of fovea size
Scale	15% of fovea size
$\mathcal{P}$	1
$\mathcal{S}$	3 (3 grid points)
Num image probes	461
Num nodes generated	6324
Num competing paths	15

No scaling in training	
Top choice correct	47.9%
In top 15	64.6%

15% scaling in training	
Top choice correct	93.3%
In top 15	98.7%

A disjoint test set was used for testing the retrieval capability; each test probe was randomly scaled in the range of [–20, +20] *percent of the fovea size to test the ability of the system to generalize over various scales.*

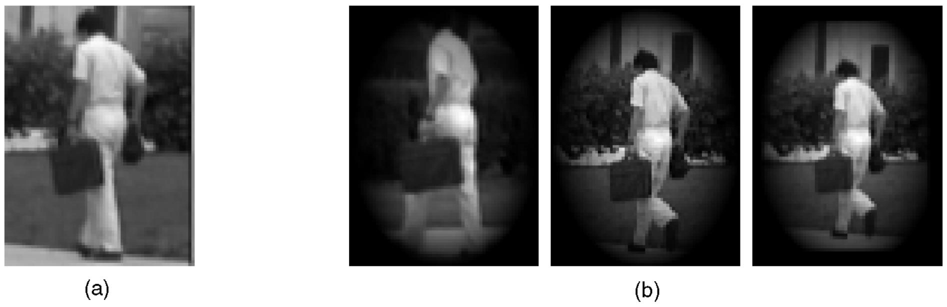


Fig. 12. Example probe and its retrieved images when scaling was enabled on the tree. (a) Test probe. (b) Retrieved images.

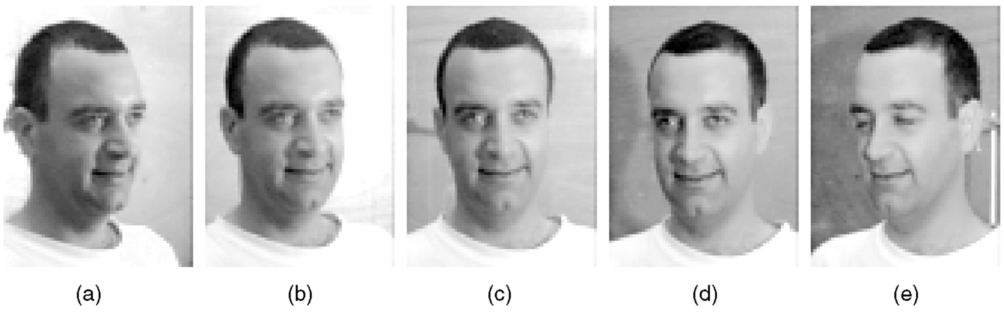


Fig. 13. A sample of the Weizmann Institute face data. Each individual for this experiment contained five viewpoints under identical lighting conditions.

the test probes, either the attention supplier must correspond to the scaling done in the training phase, or the training set must be expanded to include those images in the range of scales that need to be properly retrieved. Fig. 12 shows an example test probe and the images retrieved when scaling was enabled for the training phase.

3.4.2 Handling Scale and Position

Utilizing the system on the large combination data set shows the ability of the system both to operate on large image database sizes and to handle a specified variation in the scale of the extracted area of interest for an attention point and scale. To show the positional extension of the training data, a smaller original training set was used. For the second experiment in the demonstration of the system to handle 2D parameter changes, the data set described in Section 3.1 was used. Table 6 shows the data pursuant to this experiment.

3.4.3 Handling Different 3D Views

We want to determine whether the system can handle variation in 3D orientation. For this experiment, we used the Weizmann Institute face database. This database was well-suited to test the handling of 3D rotation because for each of 29 individuals, five camera viewpoints were available under identical lighting conditions. A sample of the available images is shown in Fig. 13.

When the training set contains a representative set of views of the objects under consideration, the system is able to successfully find objects from a novel view, as shown in Fig. 14. The image pool for a particular individual were images taken from five viewpoints under identical lighting and expression conditions. A total of four views from each individual were used for training, the remaining view left out to form a disjoint test set.

We used a disjoint test set for determining the accuracy of the learning-based view generalization. The results of this experiment are summarized in Table 7. Though Table 7 shows favorable results, 100 percent accuracy was not achieved. The failures occurred where the test probe viewing angle did not fall between two training sample viewing angles, as shown in Fig. 15.

3.4.4 Timing

The hierarchical organization of the database inherent in the SHOSLIF-O paradigm provides the means for efficient retrieval of images from the database. The tree structure provides O(log *n*) access time for *n* samples in the database. For example, on a database of 1,317 images, the system was built using both the tree mode and a nontree mode. In a

TABLE 6
SUMMARY OF THE LEARNING-BASED PARAMETER GENERALIZATION EXPERIMENT

Number of training images	86 (expanded to 2322)
Positional enhancement	20% of fovea size
Scale enhancement	20% of fovea size
$\mathcal{P}$	9 (3×3 grid)
$\mathcal{S}$	3 (3 grid points)
Number of image probes	29
Number of nodes in generated tree	4634
Number of nodes expanded at each level	≤ 10
Top choice correct	93.1%
In top 10	96.6%

A disjoint test set was used for testing the retriveal capability.

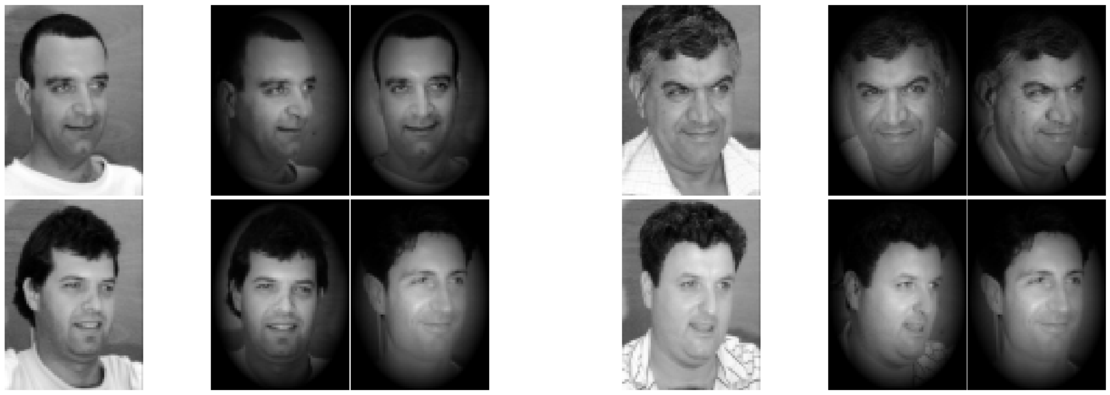


Fig. 14. View-based generalization. When sufficient training images are given for a particular class, the system is able to accurately retrieve images from the correct class. (a) Test probes. (b) Retrieved images. (c) Test probes. (d) Retrieved images.

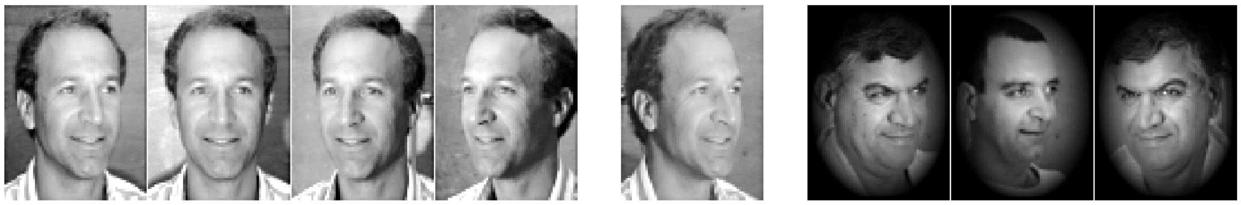


Fig. 15. Example failure in the learning-based view generalization experiment. The failures occurred only when the viewing angle of the test probe did not fall between the viewing angles of two training images. These images are courtesy of the Weizmann Institute. (a) Training images. (b) Test probe. (c) Retrieved images.

nontree mode, every test probe required 1,317 comparisons; in the tree mode with 15 competing paths, only 101 comparisons were required on average per test probe, showing an order of magnitude improvement.

Table 8 is the result quoted from Weng and Chen [41], which shows how the tree structure speeded up the retrieval process. The test was done on a Sun SPARC-20 for indoor autonomous navigation where each tree leaf node is associated with a navigation control signal (heading direction and speed) instead of class label. A total of 2,850 training images were used in the learning phase. Three schemes are compared in the table. The flat image space scheme uses a linear search for the nearest neighbor in the original image space (each pixel is a dimension). The flat MEF space scheme uses a linear search for the nearest neighbor in the MEF subspace and the projection time for the input image is included. The SHOSLIF tree scheme is a real-time version using a binary tree of the SHOSLIF [41]. The speed-up of the tree is more drastic when n is larger.

Table 2 has shown that both the tree structure and the different subspaces for different nodes have improved the recognition rate. When a flat database was used for the data set described in Table 4, the retrieval rate fell from 95 percent to 88.9 perecent for the top choice, and from 99 percent to 96.2 percent for the top 10 choices. Therefore, compared to a single space (i.e., the image space or a single MEF subspace using PCA which approximates the image space), our experiments have shown that the SHOSLIF method is not only faster but also that it produces better results.

TABLE 7
SUMMARY OF THE LEARNING-BASED VIEW GENERALIZATION EXPERIMENT

Number of training images	112
Number of image probes	28
Number of internal nodes in generated tree	177
Number of nodes expanded at each level	≤ 10
Average number of nodes explored per probe	28.5
Correct class is top choice	78.9%
Correct class is in top 10 choices	89.4%

A disjoint test set was used for testing the retrieval capability.

TABLE 8
AVERAGE COMPUTER TIME PER TEST PROBE, QUOTED FROM WENG AND CHEN [41]

Method	Flat image space	Flat MEF space	SHOSLIF tree
Time	2.854 s	0.738 s	0.027 s

4 CONCLUSIONS AND DISCUSSION

We have developed an object recognition system that performs automatic feature derivation and extraction, utilizes a hierarchical database organization to provide efficient retrieval of images. The system generalizes an image training set to handle size and position variations, and handles a wide variety of objects from natural scenes.

How do we categorize images? Of course, it depends very much on the application. Categories are objects that seem to belong together [42]. A class is different from a category in that it can specify any attributes (e.g., objects that weigh an even number of grams). Cognitive studies have indicated that humans often categorize objects into a taxonomy—a

Strengths	Limitations
The Generality of the representation obtainable by deriving features directly from intensity image pixel values. The only imposed representation is the digital image.	The exclusion of specific domain knowledge in the system design and implementation phases, since such knowledge is difficult to effectively apply to complex problems.
The capability of the system to handle multifarious variations without system modification.	Variations must be covered in the training samples.
The method is view-based to deal with images directly rather than using other sensing modalities of limited applicability (such as range scanners).	The system requires many views as training input for nontrivial problems.
The generality of real-world problems that can be handled by the system.	The system requires well-framed images, or requires a search scheme.
The logarithmic complexity and the class-separation problem decomposition made possible by the tree structure.	Occlusion has not been investigated, but will not be handled well without additional attention mechanisms.
Labeled and unlabeled samples (i.e., supervised and unsupervised learning) can be incorporated into the same tree.	The system does not cover high-level reasoning at the current stage.

Fig. 16. Major strengths and limitations of the described system.

hierarchy in which successive levels refer to increasingly more specific objects. There is an intermediate level which is more likely to be used to encode experience that superordinate or subordinate levels. For example, people use *apple* to refer to an experience rather than *fruit* or *McIntosh apple* [43]. The method presented here could be used to organize images based on an intermediate level of category that the users prefer to utilize. Retrieval with other superordinate or subordinate categories as well as other classifications schemes may be realized using pointers from symbolic attribute tables to the corresponding leaves of the SHOSLIF tree [44].

The automatic hierarchical discriminant analysis method used in this work recursively decomposes a huge, high-dimensional nonlinear problem into smaller, simpler, tractable problems. The hierarchical Quasi-Voronoi Tessellation provides a means for dividing the space into smaller pieces for further analysis. This allows the *DKL* projection to produce better local features for more efficient and tractable subclass separation. The Space-Tessellation Tree introduced in this work provides a time complexity of $O(\log n)$ for a search for the best match in a model database of n objects. This low complexity opens the door towards learning a huge database of objects.

The system described has several inherent strengths and limitations, which are summarized in Fig. 16. Although the system might be extended to other sensing modalities, the current system is view-based: in order to retrieve a correct image from an image database, the SHOSLIF requires that the system has been trained on an image taken at a very similar viewpoint (i.e., within a few degrees), as is demonstrated by Fig. 15. Object recognition systems typically have a reject option to indicate that an object was not retrieved from the database. The SHOSLIF-O could learn a reject distance threshold value to provide this functionality. In this work, however, all of the top k candidates were provided to the user.

Acknowledgments

This work was supported, in part, by the National Science Foundation under Grant No. CDA-9724462; NASA Grant Nos. NGT-40046 and NGT5-40042; the South Dakota Space Grant Consortuim to Swets; the National Science Foundation Grant Nos. IRI 9410741 and IIS 9815191; and the Office of Naval Research Grant No. N00014-95-06.

References

[1] K. Ikeuche and T. Kanade, "Automatic Generation of Object Recognition Programs," *Proc. IEEE*, vol. 76, no. 8, pp. 1,016–1,035, 1988.
[2] W.E.L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, 1990.
[3] D.P. Huttenlocher and S. Ullman, "Object Recognition Using Alignment," *Proc. Int'l Conf. Computer Vision*, pp. 102–111, London, England, 1987.
[4] D.J. Kriegman and J. Ponce, "On Recognizing and Positioning Curved 3-D Objects from Image Contours," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. 12, no. 12, pp. 1,127–1,137, 1990.
[5] F. Stein and G. Medioni, "Efficient Two Dimensional Object Recognition," *Proc. 10th Int'l Conf. Pattern Recognition*, Atlantic City, 1990.
[6] J. Weng, N. Ahuja, and T. S. Huang, "Learning Recognition and Segmentation Using the Cresceptron," *Proc. Int'l Conf. Computer Vision*, pp. 121–128, Berlin, May 1993.
[7] M. Turk and A. Pentland, "Eigenfaces for Rcognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
[8] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, pp. 59–69, 1982.
[9] T. Kohonen, "Self-Organized Network," *Proc. IEEE*, vol. 43, pp. 59–69, 1990.
[10] T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE*, vol. 78, pp. 1,481–1,497, 1990.
[11] E.B. Baum, "When are *k*-Nearest Neighbor and Back Propagation Accurate for Feasible Sized Sets of Examples?" *Proc. EURASIP Workshop, Sezimbra, Portugal*, L.B. Almedia and C.J. Wellekens, eds., pp. 2–25. New York: Springer-Verlag, 1990.
[12] J. Rubner and K. Schulten, "Development of Feature Detectors by Self-Organization," *Biological Cybernetics.*, vol. 62, pp. 193–199, 1990.
[13] A. Pentland, B. Moghaddam, and T. Starner, "View-Based and Modular Eigenspaces for Face Recognition," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition*, pp. 84–91, Seattle, Washington, June 994.

[14] C. Bregler and S. M. Omohundro, “Nonlinear Manifold Learning for Visual Speech Recognition,” *Proc. Int’l Conf. Computer Vision*, pp. 494–499, 1995.

[15] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*. Chapman & Hall, 1993.

[16] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons, 1973.

[17] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, N.J.: Prentice Hall, 1988.

[18] J. Quinlan, “Induction of Decision Trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[19] B.D. Ripley, *Pattern Recognition and Neural Networks*. New York: Cambridge Univ. Press, 1996.

[20] D.J. Hand, *Discrimination and Classification*. Chichester: John Wiley & Sons, 1981.

[21] S.S. Wilks, *Math.l Statistics*. New York: John Wiley & Sons, 1963.

[22] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, second edition, New York: Academic Press, 1990.

[23] G.R. Dattatreya and L.N. Kanal, “Decision Tress in Pattern Recognition,” *Progress in Pattern Recognition*, L. Kanal and A. Rosenfeld, eds., pp. 189–239, New York: Elsevier Science, 1985.

[24] S.R. Safavin and D. Landgrebe, “A Survey of Decision Tree Classifier Methodology,” *IEEE Trans. Systems, Man and Cybernetics*, vol. 21, pp. 660–674, May / June 1991.

[25] S.K. Murthy, “Automatic Construction of Decision Trees from Data: A Multidisciplinary Survey,” *Data Mining and Knowledge Discovery*, 1998.

[26] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. New York: Chapman & Hall, 1993.

[27] E.G. Henrichon, jr. and K.S. Fu, “A Nonparametric Multivariate Partitioning Procedure for Pattern Classification,” *IEEE Trans. Computers*, vol. 18, pp. 614–624, July 1969.

[28] J.H. Friedman, “A Recursive Partition Decision Rule for Nonparametric Classification,” *IEEE Trans. Computers*, vol. 26, pp. 404–408, Apr. 1977.

[29] H. Murase and S.K. Nayar, “Illumination Planning for Object Recognition in Structured Environments,” *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition*, pp. 31–38, Seattle, Washington, June 1994.

[30] D.L. Swets, B. Punch, and J.J. Weng, “Genetic Algorithms for Object Recognition in a Complex Scene,” *Proc., Int’l Conf. Image Processing*, pp. 595–598, Washington, D.C., Oct. 1995.

[31] D.L. Swets and J.J. Weng, “Using Discriminant Eigenfeatures for Image Retrieval,”*{IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 18, pp. 831–836, Aug. 1996.

[32] D.L. Swets and J.J. Weng, “Efficient Content-Based Image Retrieval Using Automatic Feature Selection,” *Proc., Int’l Symp. Computer Vision*, pp. 85–90, Coral Gables, Fla., Nov. 1995.

[33] M. Kirby and L. Sirovich, “Application of the Karhunen-Loève procedure for the characterization of human faces,” *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 12, pp. 103–108, Jan. 1990.

[34] I.T. Jolliffe, *Principal Component Analysis*. New York: Springer-Verlag, 1986.

[35] M.M. Loève, *Probability Theory*. Princeton, N.J.: Van Nostrand, 1955.

[36] T. Hastie and R. Tibshirani, “Discriminant Adaptive Nearest Neighbor Classification,” *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 18, pp. 607–616, June 1996.

[37] R.A. Fisher, “The Statistical Utilization of Multiple Measurements,” *Annals of Eugenics*, vol. 8, pp. 376–386, 1938.

[38] D.L. Swets and J.J. Weng, “SHOSLIF-O: SHOSLIF for Object Recognition and Image Retrieval (phase II),” Technical Report CPS 95-39, Dept. of Computer Science, Michigan State Univ., East Lansing, Mich., Oct. 1995.

[39] A. Pentland, R.W. Picard, and S. Scarloff, “Photobook: Tools for Content-Based Manipulation of Image Databases,” *SPIE* Storage and Retrieval Image and Video Databases II, no. 2,185, San Jose, Feb. 1994.

[40] B. Moghaddam and A. Pentland, “Maximum Liklihood Detection of Faces and Hands,” *Int’l Workshop Automatic Face- and Gesture-Recognition*, M. Bichsel, ed., pp. 122–128, 1995.

[41] J. Weng and S. Chen, “Incremental Learning for Vision-Based Navigation,” *Proc. Int’l Conf. Pattern Recognition*, vol. IV, pp. 45–49, Vienna, Austria, Aug. 1996.

[42] E.E. Smith, “Categorization,” *Thinking*, D.N. Osherson and E.E. Smith, eds., pp. 33–53, MIT Press, 1990.

[43] E. Rosch, C. Mervis, D. Gray, D. Johnson, and P. Boyes-Braehm, “Basic Objects in Natural Categories,” *Cognitive Psychology*, vol. 3, pp. 382–439, 1976.

[44] D. L. Swets, Y. Pathak, and J. J. Weng, “An Image Database System for with Support for Traditional Alphanumeric Queries and Content-Based Queries by Example,” *Multimedia Tools and Applications*, vol. 7, no. 3, 1998.

Daniel L. Swets (S’85–M’96) earned his BS degree in computer science from Calvin College, Grand Rapids, Michigan in 1986, and the MS and PhD degrees in computer science from Michigan State University in 1991 and 1996, respectively. From 1986-1987, he was a software engineer at Rockwell International, Downey, California, whre he worked on the Space Shuttle Orbiter Backup Flight System. From 1987-1992, he was a software engineer at Smiths Industries, Grand Rapids, Michigan, where he worked on software for aerospace applications. Concurrently, he was an instructor at both Grand Valley State University, Allendale, Michigan and the Grand Rapids Community College, Grand Rapids, Michigan. From 1992–1994, he was a teaching and research assistant at Michigan State University while pursuing his PhD degree, and an Ameritech fellow at Michigan State University from 1994-1995. In 1995, he joined the teaching staff at Augustana College, Sioux Falls, South Dakota, where he is now an assistant professor of computer science. He has been a NASA Space grant fellow from 1996–present. Concurrent with these activities, he has owned and operated a small business computer-consultant firm. His current research interests include parallel processing, algorithms for remote sensing, computer vision, and pattern recognition. He is a member of the IEEE and the IEEE Computer Society.

Juyang Weng (S’85–M’88) received his BS degree from Fudan University, Shanghai, People’s Republic of China, in 1982, and the MS and PhD degrees from the University of Illinois at Urbana-Champaign, in 1985 and 1989, respectively, all in computer science. From 1984–1988, he was a research assistant at the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. From 1989–1990, he was a researcher at Centre de Recherche Informatique de Montréal, Quebec, Canada, while adjunctively with Ecole Polytechnique de Montréal. From 1990–1992, he held a visiting research assistant professor position at the University of Illinois at Urbana-Champaign. In 1992, he joined the Department of Computer Science, Michigan State University, East Lansing, where he is now an associate professor. He is a coauthor of the book *Motion and Structure from Image Sequences* (Springer-Verlag, 1993). His current research interests include computer vision, human-machine multimodal interface using vision, speech, gesture and actions, autonomous learning robots, and automatic development of machine intelligence. He is a member of the IEEE and the IEEE Computer Society.