Multidimensional Indexing for Recognizing Visual Shapes

Andrea Califano, *Senior Member, IEEE,* and Rakesh Mohan, *Member, IEEE*

Abstract—**This paper introduces an analytical framework for studying some properties of model acquisition and recognition techniques based on indexing. The goal is to demonstrate that several problems previously associated with the approach can be attributed to the low dimensionality of invariants used. These include limited index selectivity, excessive accumulation of votes in the look-up table buckets, and excessive sensitivity to quantization parameters. Theoretical results demonstrate that using high-dimensional, highly descriptive global invariants produces better results in terms of accuracy, false positive suppression, and computation time.**

A practical example of high-dimensional global invariants is introduced and used to implement a 2-D shape acquisition/recognition system. The acquisition/recognition system is based on a two-step table look-up mechanism. First, local curve descriptors are obtained by correlating image contour information at short range. Then, seven-dimensional global invariants are computed by correlating triplets of local curve descriptors at longer range.

This experimental system is meant to illustrate the behavior of a high-dimensional indexing scheme. Indeed, its performance shows good agreement with the analytical model with respect to database size, fault tolerance, and recognition speed. Model acquisition time is linear to cubic in the number of object features. Object recognition time is constant to linear in the number of models in the database and linear to cubic in the number of features in the image. The system has been tested extensively, with more than 250 arbitrary shapes in the database. Unsupervised shape and subpart acquisition is demonstrated.

I. Introduction

OBJECT recognition is a central problem in computer vision. It involves a set of object models that must be recognized in images. General surveys on model-based object recognition systems can be found in [8] and [20]. The basic paradigm contains three components: *matching, pose computation*, and *verification* [36]. During matching, features extracted from an image are compared and associated with features of the object models. As all feature combinations may have to be explored, brute-force matching is computationally equivalent to an exponential search. One distinguishing contribution of various recognition systems has been solutions for cutting down the complexity in this area. However, even when techniques aimed at optimizing this process are used (e.g., alignment [35]), each model in the database must be separately considered. In applications with large databases, this computation burden has to be borne every time an image is processed.

Recently, an alternate paradigm called *indexing* or *hashing* has been proposed [40], [21]. In indexing, the feature correspondence and search of model database are replaced by a table look-up mechanism. The latter approach is described in detail in Section II and in the rest of this paper. Indexing schemes share a uniform underlying structure. They compute invariants from an image that are then used as indexes to look-up a table containing references to the object models. The table look-up returns a list of candidate models with associated weighs indicating their likelihood. Proposed indexing schemes mainly differ in the choice of invariants employed as indexes.

The benefits of indexing over traditional search-based matching schemes should be specifically evident in applications involving large model databases. Such applications could include image databases for multimedia applications, information services, libraries, and archives where the number of objects of interest ranges in the thousands. Indexing does not require considering each model separately and is thus less dependent on the database size. Additionally, indexing offers a straightforward approach to automatic or semiautomatic model acquisition, a useful feature in most applications.

Although there is an abundance of proposed indexing schemes [38], [40], [21], there have been few computational principles guiding their design. Some straightforward error analysis is provided in [43], while an estimate of the upper bounds on the database size for an optimally designed similarity geometric hashing techiques is given in [50].

Given the lack of a computational framework and the practical difficulty of rigorously testing such experimental systems on large libraries under varying conditions, it is difficult to evaluate the performance of indexing schemes. This is especially true in the context of their natural applications, namely large model databases. This problem has recently been highlighted by Grimson and Huttenlocher [30] in their critique of geometric hashing. More recently, some papers have been aimed at answering these remarks, and to extend the accuracy and capacity of the original geometric hashing with the introduction of a Bayesian scheme [50].

This paper presents a computational framework in which acquisition/recognition techniques based on indexing are analyzed. The criteria include computational efficiency, noise tolerance, and database as a function of the amount of information contained in the invariants used as indexes. A result of our analysis is that for indexing schemes to be practical, they must take advantage of look-up tables containing a large

Manuscript received July 29, 1991; revised April 27, 1993. Recommended for acceptance by Associate Editor D. Huttenlocher.

The authors are with the IBM Thomas J. Watson Research Center, Exploratory Computer Vision Group, Yorktown Heights, NY 10598.

IEEE Log Number 9214453.

number of coarsely quantized entries. We propose that this can be best accomplished using very descriptive invariants and indexes, typically of a high-dimensional nature. In the context of this paper, we will therefore use the term high dimensional to indicate invariants that have a large number of coarsely quantized distinct values.

The following analysis compares high-dimensional indexing techniques to the more conventional low-dimensional schemes, such as geometric hashing. A first result is that the use of high-dimensional indexes drastically increases the signal-to-noise ratio of the approach. That is, the number of votes for randomly generated incorrect hypothesis becomes smaller as the dimensionality of the indexes increases, while the average number of votes for correct hypotheses is kept constant. A second, equally important, result is that the amount of computation required by the technique is significantly reduced. Section III presents this analysis in detail.

We will show that the use of high-dimensional indexes is crucial to overcoming some drawbacks of table look-up based techniques such as geometric hashing [38], [42] when large object model databases are considered. These include poor index selectivity, excessive accumulation of votes in each bucket, sensitivity to noise and quantization parameters, and probability of false positives. The latter would require further processing to be successfully eliminated.

Grimson [30], for instance, suggests that geometric hashing performs well on simple scenes with little sensor noise, but that its performance degrades significantly even with limited amounts of clutter or perturbation. This paper shows that these problems, far from being inherent limitations of the underlying approach, are mainly a result of the low dimensionality of the indexes of choice. In Grimson's error analysis, buckets are characterized by 2-D invariant vectors of limited descriptive power. With these assumptions and a reasonable choice of quantization parameters, a practical table would have on the order of 10^4 useful buckets. Even under ideal conditions (e.g., uniform index distribution in the table), as the number of object models stored in the table increases, the number of entries per bucket becomes inevitably large. This will be discussed in detail in Section III. When quantization effects, nonuniform distribution of the indexes, and projective distortion of the objects are taken into account, the complexity of the analyzed scenes, the maximum size of the database, and the computation time become issues, as pointed out in [30].

On the other hand, our analysis shows that indexing techniques based on higher dimensional look-up tables are faster, can reliably handle very large model databases, and can reduce the occurrence of false positives. As a practical example, in Section IV we will introduce an experimental 2-D acquisition/recognition system based on a seven-dimensional table look-up mechanism. This test system uses global shape descriptors that are invariant to 2-D similarity transforms. The corresponding look-up table can hold about 10^6 buckets. Assuming a perfectly uniform index distribution, and on the order of 10^3 indexes per model, the resulting shape table could accommodate on the order of 10^3 objects before saturation (i.e., before its average number of entries per buckets becomes equal to 1). Under similar assumptions, standard geometric hashing techniques would be limited to a table size of about 10^4 buckets. This could hold a scarce 10 objects before saturation, a two orders of magnitude difference. Nonuniform index distributions would degrade performance similarly both in the low- and high-dimensional approaches. Since rehashing methods [50] can alleviate the problem of nonuniformity, we do not consider it an issue. Additionally, in our example, coarser quantization is selected along each parameter axis (i.e., fewer buckets per axis) with respect to that assumed for standard geometric hashing. This offers the benefit of an increased noise resilience. The relationships between quantization and index dimensionality is studied in detail in Subsection III-D.

The novelty of the proposed approach is the use of correlated complex local shape parameters (i.e., four-dimensional local shape descriptors) to produce viewpoint invariant, high-dimensional global descriptors. These descriptors are used as indexes in a look-up table to identify the 2-D shapes from a database and recover their position, scale, and orientation. Local shape descriptors are also extracted within a look-up table paradigm by correlating over short distances information such as local edge orientation and position.

The seven-dimensional invariant used in the test 2-D acquisition/recognition system is intended to be only one practical example of a high-dimensional global descriptor. Further analysis will undoubtedly lead to more robust and/or higher dimensional global invariants. In fact, such invariants based on the long-distance correlation of lower dimensional local shape descriptors can be easily obtained in a number of different ways. These include using nongeometric parameters, e.g., intensity ratios at different image locations, color, range information, and other visual clues. This makes high-dimensional indexing a viable candidate for analyzing a variety of inputs modalities, not limited to 2-D contour shapes.

The design of the test system [18] has allowed us to focus on various design issues (beyond that of index dimensionality) involved in the making of a practical working recognition system. These include local versus global descriptors, feature-subpart hierarchy, and heuristics for reducing recognition time. Finally, the proposed approach tries to address what we consider important limitations of conventional table look-up techniques. For instance, they often fail to offer a straightforward mechanism to support important notions in shape representation, such as subparts identification and hierarchical database organizations. The system is described in Sections IV through X, and some experimental results arepresented in Section XI.

Again, we wish to emphasize that we are presenting a specific implementation of the high-dimensional indexing paradigm. The test system is an example intended mainly to illustrate some advantages of the approach. Many other alternative embodiments are possible, and we make no claims regarding the specific robustness of the global invariant of choice. The computational analysis of Section III, however, is of a general nature and is valid for any other indexing schemes regardless of the specific choice of invariants. Therefore it can be used to guide the design and predict the performance of any such system.

II. INDEXING

In general, index-based recognition systems compute *invariants* from an image and are then used to *index* in a *look-up table* During model acquisition, the locations (*buckets*) of the table so indexed are filled with *entries* containing references to object models and some additional parameters. The latter, for instance, can be used for pose recovery. During recognition, the models listed in the indexed entries are collected into a list of candidate models. Next, the most likely of the candidate models are selected on the basis of the number of votes they have received, i.e., the number of times they were indexed. Recognition may include pose computation and verification. The look-up tables used in the process are usually sparse in that, on average, only a few of their entries are used. The look-up tables are, therefore, often implemented as a hash tables to save storage.

One well-known index-based approach is geometric hashing [42], [40], [21]. In its basic form, geometric hashing handles 2-D shapes under similarity (affine) transformations. In the image, two (three) feature points are chosen as a reference frame, also called a basis, and the orthogonal (affine) coordinates (α, β) of each other model point in that basis are used as indexes. Hence, the *dimensionality* of the index is 2. Each indexed entry is updated with the basis vectors and with a reference to the corresponding object model. This process is repeated using each couple (triplet) of model points as a basis. In its original conception, geometric hashing is therefore a *single*-step, bottom-up scheme. That is, object models are recognized directly from sampled image points without grouping or correlating any intermediate geometric features or subparts.

A. Indexing Versus Matching

The benefits of indexing over traditional matching-based schemes should be specifically evident in applications involving large collections of object models. Typical examples would be image databases for multimedia applications, information services, libraries, archives, and department stores, where the number of objects that need to be recognized number in the thousands. Indexing does not involve a search over the image database and is thus less sensitive to the size of the model database. For this reason it is very important to study how well these techniques scale with respect to the number of object models in the database.

In these applications, model acquisition is performed only once, usually off-line, while recognition is performed repeatedly, often under timing constraints. The acquisition step can thus afford to be computationally expensive, while recognition must be as computationally efficient as possible. This is precisely what happens in the case of indexing techniques, where most of the computational load is shifted from recognition to acquisition time. This is accomplished by precomputing a large number of model representation invariants and storing them in fast access look-up tables.

Another benefit of indexing over matching for large model databases is automated model acquisition. Automated model acquisition is important if a system has to deal with a large number of objects. Yet, many recognition paradigms lack the means for automatic model building [2], [6], [7], [10], [12], [24], [28], [33], [37], [38], [47], [53], [55]. The object models, rather than being acquired from raw data, are either hand crafted or generated assuming that precise geometric information is available. Model database organization and indexing schemes are critical whenever a scalable recognition behavior is to be accomplished. On the other hand, in indexing the scheme for acquiring models is similar to that for recognition. Models are acquired simply from characteristic images of the objects, making precise 3-D models or CAD/CAM-type models superfluous. Model acquisition is incremental. A system does not have to be recomputed from the start if a model must be added (or deleted).

B. Local Versus Global Features

Footprints [38] and structural indexing [55] use high-dimensional indexes. This is due to the specific choice of invariants and has not been motivated by any underlying computational scheme. Both techniques use *local* shape descriptors directly as indexes.

Local shape, however, tends to be a less discriminating visual clue in large databases where each different model may share a substantial portion of the local shape structure with many others. A more discriminating factor is the *geometrical configuration* of local shapes. In general, indexes that convey global configurational characteristics are more selective than their local counterparts. This can also be understood by considering that, by definition, the amount of visual information associated with a local shape descriptor is lower than that associated with a combination of several such descriptors and their relative geometric configuration on the object model. As a consequence, locally derived indexes must be quantized at a finer grain to obtain an equivalent range of quantized values. This has the undesired effect of making them more susceptible to noise.

A similar situation, where local versus global descriptors are compared, can be found in the discussion of single- versus multi-window Hough transforms [15].

C. Segmentation

Segmentation is not required in indexing, but even partial segmentation will aid any indexing scheme. Various bottom-up techniques, such as region segmentation and perceptual organization [45], will help. Indexing schemes in general have ignored the issue of segmentation and have been implemented without it. [21] relies on simple perceptual organization to aid indexing.

We have experimented with simple heuristics similar to the local focus feature approach of [10]. This has allowed us to reduce the recognition complexity from cubic to linear in the number of image features. The scheme is described in detail in Section IX-B. There are, however, shortcomings to using such heuristics, such as scale dependency and the need for feature saliency assignment. Another viable scheme is random sampling with a threshold on the number of votes. This latter heuristic is similar to various search cutoff schemes [29].

III. Multidimensional Indexing

The following analysis compares high-dimensional indexing techniques [18] to the more conventional low-dimensional schemes, such as geometric hashing. Not unexpectedly, a result is that the use of high-dimensional indexes drastically increases the signal-to-noise ratio of the approach. That is, the number of votes for randomly generated incorrect hypothesis becomes smaller as the dimensionality of the indexes increases, while the average number of votes for correct hypotheses is kept constant. Another important result is that the amount of computation required by the technique is significantly reduced.

A. A Framework for Indexing Techniques

To justify this claim let us introduce a formal framework for indexing techniques. Define $\{S_i\}_{i=1}^{N_M}$ as a set of model shapes. For each model shape S_i, a number N_{S_i} of independent d-dimensional vector indexes $\mathbf{a}_j = (\alpha_{1j}, \alpha_{2j}, \cdots, \alpha_{dj})$; $j = 1, \cdots, N_{S_i}$ is generated. These are usually invariant with respect to a given coordinate transformation (similarity, affine, etc.). Later we will use $\overline{N_S}$ as the average number of index generated from a model shape.

In the case of geometric hashing with similarity transforms, for instance, the invariant indexes are 2-D bases (α, β) that represent the Cartesian projections of a model shape point on the Cartesian frame defined by other two shape points (see Fig. 1(a)). In the affine case, (α, β) are the affine projections of a shape model point on the affine coordinate frame defined by other three model shape points (see Fig. 1(b)).

Storage: For each index $\mathbf{a}_j$, a tuple $(S_i, \mathbf{G})$ is then appended to the entry of a hash table H_0 indexed by the indexes $\mathbf{a}_j$, where S_i is the reference to the ith model shape and $\mathbf{G}$ is a vector of geometric parameters that can be used to recover the complete (partial) pose of the model shape in an image. For example, $\mathbf{G}$ could contain the coordinates of the center of mass of the shape in the Cartesian or affine base determined by the index $\mathbf{a}_j$.

After the process is repeated for each model shape, entries in the table H_0 will contain lists of $(S_i\mathbf{G})$ tuples. We will use and $(\overline{N_E})$ for the number of tuples $(S_i\mathbf{G})$ that each bucket of the table H_0 will hold on average.

Recognition: N_I indexes $\mathbf{a}_k$ are generated in a similar way from an image. The value of N_I usually depends on the number of objects in the image. The indexes are used to retrieve entries in H_0, that is, lists of $(S_i, \mathbf{G})$ tuples. Each reference S_i corresponds to a model shape that, with the proper pose in the image, could have generated the index. The complete (or a partial) pose $\mathbf{P} = ((x_0, y_0), \vartheta, s)$ of the shape is recovered, using the coordinate base associated with the index $\mathbf{a}_k$ and the vector $\mathbf{G}$. Each tuple $(S_i, \mathbf{P})$ thus generated from the list in H_0 is treated as evidence for the corresponding model shape hypotheses at the corresponding pose in the image. This evidence is accumulated in a second hash table H_1, where entries are indexed by the tuples $(S_i, \mathbf{P})$ and have a value proportional to the number of times that the corresponding tuple has been generated by the image indexes. H_1 is therefore a histogram of the votes received by each possible shape hypothesis. If $N_{\mathbf{P}}$ is the number of different quantized poses that a model shape can have in the image, the table H_1 will have a virtual number of entries given by $N_{H_1} = N_M N_{\mathbf{P}}$, where N_M is the number of model shapes.

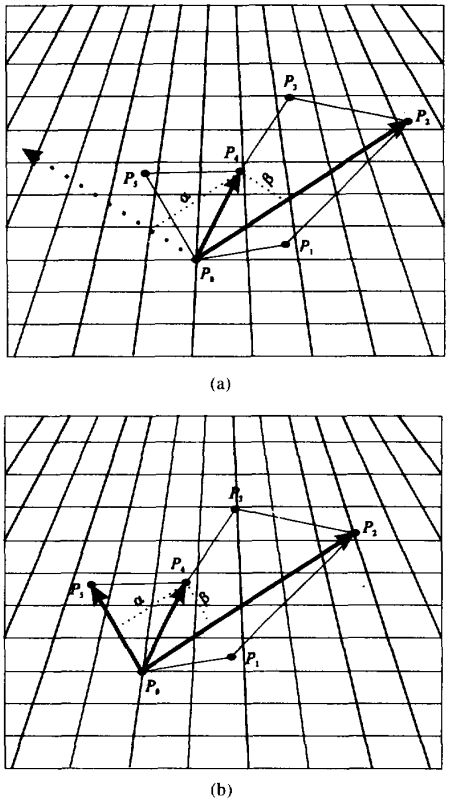

Fig. 1. Similarity (a) and affine (b) index generation in geometric hashing.

B. Outline of the Analysis

The goal of the analysis is first to identify the role of the different parameters $N_M, \overline{N_S}$ (the average number of index generated from a model shape), etc., on the behavior of indexing techniques. The second objective is to understand the mechanism responsible for false positives with large databases of model shapes. Finally, we will show how using more descriptive higher dimensional indexes decreases the probability of false positives and reduces the amount of computation required. This will be accomplished by studying the behavior of $\overline{N_G}(d)$ (the average number of votes for a correct hypothesis) and $P_B(k, d)$ (the probability of getting k votes for an incorrect hypothesis because of random cooccurrences) as a function of d (the index dimensionality). Specifically, we will keep $\overline{N_G}(d)$ constant as d grows, by simultaneously increasing the number of indexes per model or using coarser quantization of the indexes. We will then show that $P_B(k, d)$ effectively decreases as d increases for values of k around $\overline{N_G}(d)$.

In the analysis, we will assume uniform distribution of the indexes in H_0 for simplicity. A nonuniform distribution produces identical effects on the low- and high- dimensional indexing. Therefore, the comparison between the two still

holds. Also, a number of techniques can be used to generate a more uniform distribution (see for instance [49]).

C. Index Formation Techniques

Typically, indexes are formed by first generating tuples of interest features (e.g., interest points) and then by using their global configuration and associated information to produce the indexes. If l is the size of the tuple, and the model contains n points, we can then generate up to $\binom{n}{l}$ l-tuples. The total number can be bounded by means of a variety of techniques that will be mentioned in the following.

We will restrict our analysis of high- versus low-dimensional techniques to the cases where the number of local interest features in a tuple, l, is constant. The increase in dimensionality will therefore come from the use of extra local information, such as 1st- and 2nd-order local shape properties, color, texture, or other visual clues that can be used with the l-tuple geometric structure to generate invariants. Under this assumption, the number of considered l-tuples, both at storage and at recogntion time, is constant and independent of the dimensionality of the approach.

Then, the effects of occlusion can be simply modeled as follows. Let us assume that a minumum ratio of unoccluded local interest features p_u is available. That is, p_u is the ratio of unoccluded local interest features with respect to the total number of local interest features in an image. Then, if l is the tuple size and N_T is the number of l-tuples, then at least $N_T p_u^l$ are unoccluded. We can simplify this by defining a minimum bound $p_U = p_u^l$ on the probability that an l-tuple is unoccluded. This probability is constant and independent of the dimensionality d of the invariants that have been selected.

D. Problems with Conventional Approaches

The limit of indexing techniques lies primarily in the finite size of the first hash table H_0. As the number of models N_M stored in H_0 grows larger, the lists of tuples $(R_i, \mathbf{G})$ stored in the table will also increase. This leads to two important consequences:

- The time required to process an image $T_R(N_M)$ increases since all the indexed tuples must be processed.
- The probability $P_B(k, d)$ of having many votes (large values of k) for incorrect hypotheses generated by random cooccurrences increases. This may ultimately lead to false positives, that is, incorrect hypotheses that have more support than correct ones.

Correct Votes: To demonstrate these claims, let us first determine the average support that a model shape in the image will generate for the corresponding hypothesis in H_1.

Given a certain level of noise, a noise model, and a quantization step $\Delta\alpha_i$ for each axis of the index vector, there will be a probability

$$p_G(d) = \prod_{i=1}^{d} p_{g_i} \tag{1}$$

that the quantized value of index computed from a noisy scene will match the correct index generated at storage time by a given model shape present in the image. Here the p_{g_i} are the probabilities that there is equivalence on the ith axis of a d-dimensional index vector. If the values of the p_{g_i} are assumed, for simplicity, to be all equal to p_g we will have

$$p_G(d) = p_g^d \tag{2}$$

Then, with $\overline{N_S}$ the average number of indexes generated per model shape, the probability of getting k matching indexes from a model shape in the image can be obtained from the binomial distribution

$$P_G(k, d) = \binom{\overline{N}_S}{k} p_g^{kd} (1 - p_g^d)^{\overline{N}_s - k}. \tag{3}$$

The average number of matching indexes is then

$$\overline{I_G}(d) = p_g^d \overline{N}_s. \tag{4}$$

From the previous section, p_U is the worst case probability that any such index is not destroyed by occlusion. The average effect of occlusion could then be roughly modeled by introducing p_U in the estimate of $\overline{I_G}(d)$ by replacing p_g^d by $p_U p_g^d$. As stated before, p_U is independent of d and is rather a funtion of the number of independent local interest points used to generate the indexes.

A similar approach can be used to model image clutter. The main effect of clutter, i.e., multiple objects in the scene, on indexing techniques is to reduce the probability of generating correct indexes while increasing that of incorrect ones. Both average effects can be modeled by smaller values of p_g. For m unoccluded object models in a scene, for instance, the probability of generating a correct index for a specific object model is p_g^d/m^l, where l is again the number of independent local interest points used to compute the d-dimensional index. Heuristics such as the radius of coherence (Section IX-B) or the local focus feature approach [10] can be used to bring this value close to p_g^d/m. For the sake of simplicity, in the following discussion we will use only the term p_g, by factoring in it both the occlusion and clutter contributions. In any case, the negative contribution of clutter will affect both techniques in the same way since the number of local features l used to generate the indexes is independent on the dimensionality of the technique. This is consistent with the comparative nature of this paper.

Since for each correct index the corresponding list in H_0 will contain at least one correct reference to the corresponding shape, the average number of votes that correct hypotheses will receive in H_1 is at least equal to $\overline{I_G}(d)$. $\overline{N}_G(d) \geq \overline{I_G}(d)$. The inequality sign reflects random votes that may accumulate in the correct hypothesis bucket. The probability of getting k votes for a correct shape-pose pair is then $P_G(k, d)$ or better.

It is obvious that unless $p_g \approx 1$, these values will strongly depend on the dimensionality of the index. Lower dimensionality will in general correspond to a higher percentage of correct votes. On the other hand, incorrect hypotheses with many votes will also be more probable. To study the use of higher dimensional indexes objectively, we will normalize $\overline{N_G}(d)$ so that it is kept constant with respect to d, that is,

$$\overline{N}_G(d) = \overline{N}_G(d-1) = \cdots = \overline{N}_G(1). \tag{5}$$

This can be accomplished in three ways:

- By increasing the average number of indexes generated by each model shape $\overline{N_S}$ as d increases. In this case, $\overline{N}_S(d) = \overline{N}_S(1)/p_g^{d-1}$.
- By using coarser quantization step $\Delta\alpha = \Delta\alpha(d), \Delta\alpha_d$ for short, along each index axis as d increases. An appropriate choice should yield larger values of $p_g(\Delta\alpha_d)$ (in this case a function of $\Delta\alpha$) as d increases such that the global probability of getting a good d-dimensional index

$$p_G(d, \Delta\alpha_d) = p_g^d(\Delta\alpha_d) \tag{6}$$

  is constant with respect to d. That is $p_G(d, \Delta\alpha_d) = p_G(d-1, \Delta\alpha_{d-1}) = \cdots = p_G(1, \Delta\alpha_1)$.
- By a combination of the above techniques.

To avoid a larger number of indexes to process as d grows, we will rely on coarser quantization.

Using Coarser Quantization: Let us assume a Gaussian model with variance σ. For simplicity's sake, we will assume the same model over all the index axes. If the quantization step $\Delta\alpha$ is of the same order as σ, then doubling or redoubling $\Delta\alpha$ should make the probability $p_g \approx 1$. Therefore, from (6), reducing the number of quantization intervals n_b by a small factor (i.e., $2, 3\cdots$) satisfies the requirements.

$$p_G(d, \Delta\alpha_d) = p_G(d-1, \Delta\alpha_{d-1}) = \cdots = p_G(1, \Delta\alpha_1). \tag{7}$$

A more interesting case is when the original $\Delta\alpha$ is small compared to σ. In this case, a much larger reduction in the number of good votes results when high-dimensional indexes are used. We will analyze this as a worst-case scenario. Since $\Delta\alpha$ is smaller than the width of the error distribution the probability of an index generated during recognition to match the corresponding one stored at acquisition will be the sum over all ith quantization steps of the combined probability

$$\left[\frac{1}{\sqrt{2\pi}\sigma}\int_{A_0+i\Delta\alpha_d}^{A_0+(i+1)\Delta\alpha_d} \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]dx\right]^2 \tag{8}$$

that both measurements fall within the same ith quantization step of width $\Delta\alpha$. Here, x_0 is the expected value for the measurement and A_0 the start of the useful range of values on the index axis (assumed to be the same on all axes for simplicity). The probability of a match is then

$$p_g(d) = \frac{1}{2\pi\sigma^2}\sum_{i=0}^{n_b(d)}\left[\int_{A_0+i\Delta\alpha_d}^{A_0+(i+1)\Delta\alpha_d} \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]dx\right]^2, \tag{9}$$

where $n_b(d)$ the number of quantization steps on each index axis as a function of d (also assumed to be the same on all axes for simplicity). If A is the range of admissible values for the index along an axis, $n_b(d)$ is computed as

$$n_b(d) = \frac{A}{\Delta\alpha_d}. \tag{10}$$

In any case, the probability of a correct index match will be larger than

$$\tilde{p}_g(d) = \left(\frac{\Delta\alpha_d}{\sqrt{2\pi}\sigma}\right)^2, \tag{11}$$

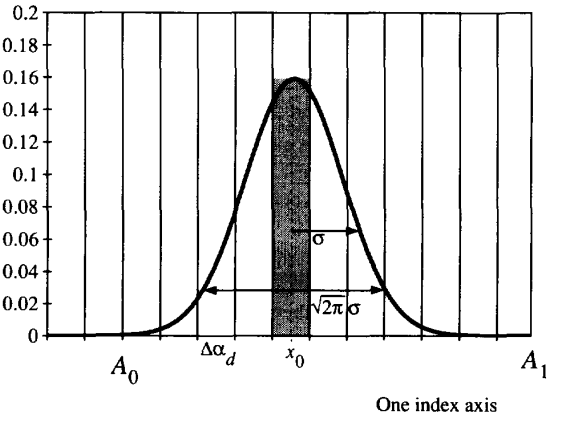

Fig. 2. Probability distribution of one index value with Gaussian noise assumption.

which is the approximate value of the sum in (9) over the quantization step closest to the midpoint in the Gaussian (see gray area in Fig. 2). We will use this value as a lower bound. It is then possible to compute a pessimistic estimate of how much more coarsely we should quantize the index axes in order to insure a constant average number of correct votes as d increases. The requirements of (7) are satisfied by choosing fewer quantization steps as d increases, as expressed by

$$n_b(d) = \left[\sqrt{p_{g1}}\right]^{1-\frac{1}{d}} n_{b1}, \tag{12}$$

where $p_{g_1} = p_g(1)$ and $n_{b1} = n_b(1)$, for simplicity. This corresponds to a larger quantization step,

$$\Delta\alpha_d = \left[\sqrt{p_{g1}}\right]^{\frac{1}{d}-1}\Delta\alpha_1. \tag{13}$$

Incorrect Votes: The mechanism that allows incorrect votes to accumulate in a given entry of H_1 (model-pose hypothesis) is slightly more complex. First, we must distinguish between correct indexes and incorrect ones. Let us consider a correct index generated from the image, that is, one that matches exactly the one produced by the corresponding model shape at storage time. Then, at least one of the $(S_i, \mathbf{G})$ tuples in the corresponding entry in the table H_0 will be correct. Thus, if $\overline{N_E}$ is the average length of the list stored in H_0, on average the other $\overline{N_E} - 1$ tuples will vote for random model shapes that happen to share the common index.

Since the model shapes that share an index are typically completely uncorrelated, it is reasonable to assume that these incorrect votes will spread uniformly over H_1.

If the index is incorrect, all of the entry tuples in the corresponding H_0 entry will support incorrect hypotheses. Therefore, on average, $\overline{N}_E$ incorrect votes will be generated. Since in general $\overline{N}_E \gg 1$, we can ignore the correct contribution in the first case and assume that any image index will in general generate $\overline{N}_E$ incorrect votes that will spread uniformly in the second table H_1.

Under this assumptions, the probability that an entry in H_1 randomly accumulates k votes is given by the binomial

distribution:

$$P_B(k,d) = \binom{N_I\overline{N_E}}{k}\left(\frac{1}{N_{H_1}}\right)^k\left(1-\frac{1}{N_{H_1}}\right)^{N_I\overline{N_E}-k}. \quad (14)$$

Since $N_I\overline{N_E} \gg k$, this can be rewritten as

$$P_B(k,d) \approx \frac{(N_I\overline{N}_E)^k}{k!}\left(\frac{1}{N_{H_1}}\right)^k\left(1-\frac{1}{N_{H_1}}\right)^{N_I\overline{N}_E}. \quad (15)$$

The only dependency on d is in the term $\overline{N}_E$. This will strongly depend on the size of the table H_0 and therefore on the dimensionality of the indexes. $\overline{N}_E$ can be computed as the number of references stored in H_0 divided by the size of H_0

$$\overline{N}_E(d) = \frac{N_M\overline{N}_S}{[n_b(d)]^d}. \quad (16)$$

Here, $n_b(d)$ is the number of separate quantized values on each index axis, assumed to be the same along each axis for simplicity. Substituting (16) in (15), we obtain

$$P_B(k,d) \approx \frac{1}{k!}\left(\frac{N_IN_M\overline{N}_S}{[n_b(d)]^d}\right)^k\left(\frac{1}{N_{H_1}}\right)^k \times\left(1-\frac{1}{N_{H_1}}\right)^{\frac{N_IN_M\overline{N}_S}{[n_b(d)]^d}}. \quad (17)$$

The final result is obtained by substituting the normalized value of $n_{b(d)}$ as a function of d from (13), where we use the notation $n_b = n_b(0)$ for simplicity. Note that N_{H_1} has also been replaced by N_MN_P, as discussed in Section III-A.

$$P_B(k,d) \approx \frac{1}{k!}\left(\frac{N_IN_M\overline{N}_S}{n_b\left(n_b\sqrt{p_g}\right)^{d-1}}\right)^k\left(\frac{1}{N_PN_M}\right)^k \times\left(1-\frac{1}{N_PN_M}\right)^{\frac{N_IN_M\overline{N}_S}{n_b\left(n_b\sqrt{p_g}\right)^{d-1}}}. \quad (18)$$

Since $\overline{N}_G(d)$ is kept constant, we can determine the behavior of the technique by looking at the ratio $P_B(k,d)/P_B(k,1)$. If this value is reduced as d increases, then the use of high-dimensional indexes is effective in reducing the chances of false positives.

$$\frac{P_B(k,d)}{P_B(k,1)} \approx \left(\frac{1}{n_b\sqrt{p_g}}\right)^{k(d-1)} \times\left(1-\frac{1}{N_PN_M}\right)^{\frac{N_IN_M\overline{N}_S}{n_b}\left[\frac{1}{\left(n_b\sqrt{p_g}\right)^{d-1}}-1\right]}. \quad (19)$$

The most important factor in this equation is the value of $n_b\sqrt{p_g}$. If this is greater than one, the probability that an hypothesis will receive a large number (k large) of votes is reduced exponentially with both k and d. If the value is smaller than one, the opposite effect will be observed. Also, the second term in (19) will be almost independent of d. $n_b\sqrt{p_g}$ is larger than 1 unless the noise produces complete indetermination on the value of the index.

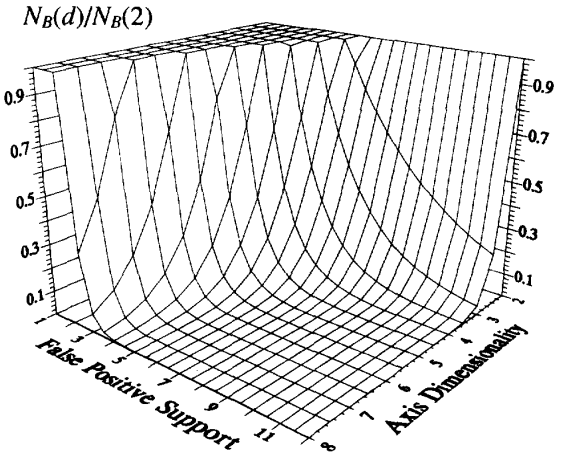

Fig. 3. Ratio of bad votes in multidimensional and 2-D indexing.

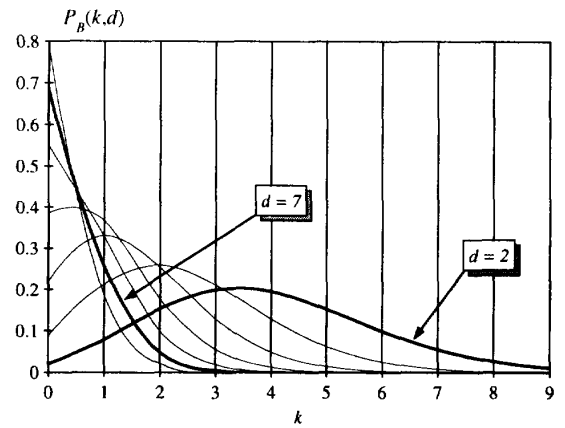

Fig. 4. Probability of false positive with k votes.

If $\Delta\alpha$ is the size of the quantization on the parameter α, and $A = A_1 - A_0$ is the allowable range of values, then $n_b = A/\Delta\alpha$. In this case, we have $n_b\sqrt{p_g} = \frac{A}{\sqrt{2\pi}\sigma}$, which is greater than 1 unless the Gaussian width is larger than the allowable range for the parameter.

Fig. 3 plots the value of (19) as a function of both k and d for $\overline{N}_S(0) = 100, N_M = 1000, N_I = 1000, n_b = 64$, and $p_g = 0.1$. Note that even for such a low probability of getting correct 1-D index values, the probability ratio drops exponentially as d is increased from 2 to 7. Analogous results can be obtained for all practical values of the fixed parameters. Fig. 4 plots the expected value for the probability of obtaining incorrect hypotheses with k votes as d increases (different curves) for the same set of parameters. Notice how the probability of bad hypotheses tends to accumulate toward the lower votes range as d increases. Therefore, the performance of low-dimensional indexing techniques can become exponentially better if more information can be used reliably to build higher dimensional, more descriptive indexes.

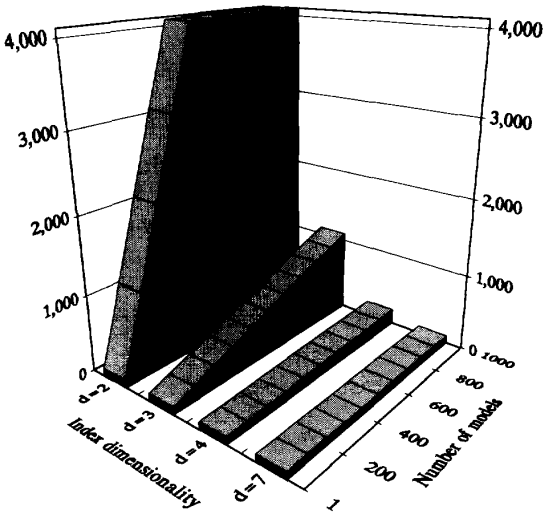

Fig. 5. Recognition time versus database size as dimensionality of index changes.

E. Time Requirements

As d increases and $\overline{N_G}(d)$ is kept constant, $\overline{N}_E(d)$ becomes

$$\overline{N_E}(d) = \frac{N_M \overline{N_S}(0)}{[p_g n_b]^d}. \tag{20}$$

Since $n_b p_g$ is always larger than 1, the value of $\overline{N_E}(d)$ will then decrease exponentially with d. In the best case, when both look-up tables are implemented as arrays, the average time required to process one image index is given by

$$\overline{T}_I(d) = t_0 + \overline{N}_E t_1 = t_0 + \frac{N_M \overline{N}_S(0)}{[p_g n_b]^d} t_1, \tag{21}$$

where t_0 is the time required to compute the index and to access the corresponding entry in H_0 and t_1 is the time required for generating the index and updating the corresponding entry in H_1. Therefore, the total time required to process the N_I image indexes will be

$$\overline{T}_{\text{Tot}}(d) = N_I \left[t_0 + \overline{N}_E t_1 = t_0 + \frac{N_M \overline{N}_S(0)}{[p_g n_b]^d} t_1 \right]. \tag{22}$$

Fig. 5 shows the time to process 1000 image indexes as the model in the database grows from 1 to 1000 and for values of $d = 2$ (geometric hashing) to $d = 7$ (multidimensional indexing). Assuming that the hash table is stored on disk (for large databases), we choose $t_0 = 0.1$ s and $t_1 = 0.01$ s. The other fixed parameters have the same value chosen in the previous subsection. As we can see, although the behavior is linear in both cases, the growth rate is significantly different, leading to much faster recognition times for the higher dimensional case.

When the size of the table is approximately 10^6, the linear growth constant is so small that recognition time on a shape table with 100 objects is only 10% larger than the one with a single object. By comparison, with a table 10 times smaller, the increase is of 123%. The memory required in the two cases, however, is the same since the number of entries stored in the table is not a function of its total size and sparse data can be represented in a compact form by means of a hashing paradigm.

Index-based techniques can perform recognition when the average number of entry per bucket is much above saturation, i.e., larger than 1. Nonetheless, as this value keeps increasing, so do the recognition time and the chances of detecting false positives. Besides, having fewer entries per bucket corresponds to a higher discriminating power of the table. Fewer shape instance hypotheses are produced, so the clustering process in parameter space or the checking of the candidate object model become much easier tasks. Since the performance of the table is based on its size and sparseness (see (22)), shape tables are ideally implemented as hash tables.

IV. Approach Outline

To substantiate in practice the results of Section III, we have implemented a 2-D recognition system based on high-dimensional indexing associated with global descriptors. This section and the following one should therefore be viewed as the description and study of one out of many possible implementations of a high-dimensional indexing scheme. We make no claims for the robustness and stability of the invariants used as indexes other than they are high dimensional (e.g., very descriptive) and, to the extent of our purposes, invariant to similarity transformations. Better descriptors can be devised, more refined techniques for detecting local shape can be used, and better ways for correlating local shape over long distances can be identified. The point here is only to provide a simplified test system to study the practical behavior of a recognition system based on the above-discussed approaches.

A two-step process is proposed (see Fig. 6). In the first stage local features are detected from a 2-D contour image. Edge-curves are segmented and labeled on the basis of their local shape. An indexing-based recognition scheme is used to recognize curve shapes. This is only to show that a homogeneous approach to recognition where only indexing is used is viable. Other fundamentally different schemes could be equally well used to detect the local features.

In the second stage, the local curve features are used to compute high-dimensional invariants. These invariants are used to index into a shape table containing references to object models. Thus the scheme performs indexing twice: First, short-range autocorrelation operators on edges are used to index into a small set of simple localized shape descriptors, and next global autocorrelation operators on local curve shapes are used to index into a global look-up table that contains the shape model representations.

During model acquisition, one or more characteristic views of the model are presented. Local curve-shape features are detected, and high-dimensional indexes are computed from combinations of the curve features. At each corresponding indexed location in the shape table (bucket), an entry is appended with a reference to the model and specific parameters to recover pose.

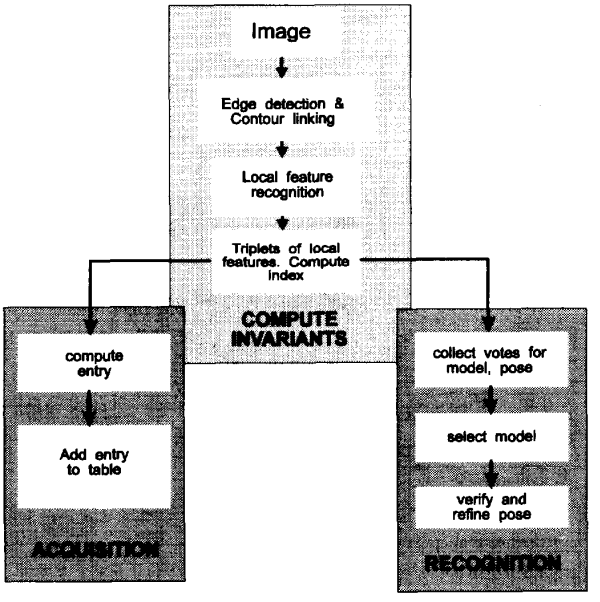

Fig. 6. The system architecture flowchart.

During recognition, high-dimensional indexes are computed with an identical approach. These indexes are subsequently used to index into the shape table and to recover all the models stored in the respective table entries. Also, from the entry and the configuration of the local shapes used to compute the index, the pose—i.e., the location, orientation, and scale of the identified shape—is computed. The hypothesized shape instances (i.e., models and their pose) are then ordered on the basis on the number of times they were indexed. Only those shape instances with the best match to the input data are selected, thus effectively segmenting the image into distinct recognized objects.

A performance analysis with respect to fault tolerance and recognition behavior with a large number of shapes (≈ 290) is given in Section X. Both acquisition and recognition are efficient; specifically, recognition is not exponential in the problem size as it is the case for many systems [29] and grows very slowly with the number of models in the database. It grows approximately as $K + cQ$, where K is a constant time, Q is the number of objects in the database, and c is of the order of 0.001. In Sections V through IX, the approach is discussed in detail and compared with existing techniques. Section X deals with subpart detection. Finally, Section XI provides some examples to support the theoretical analysis.

V. Shape Autocorrelation

The computation of indexes, both for the local curve features and for the global high-dimensional invariants, is based on the concept of *shape autocorrelation* [17]. We introduce shape autocorrelation operators and describe their application in a generalized parameter transform framework. Some notions are recalled from previous work [15], [16]. The main result here is that for computing shape descriptors, in our case invariant indexes, it is better to use information spatially distributed over the object shape rather than at localized portions of the shape (as is done in footprints [38] or structural hashing [55]).

A. Extending the Generalized Neighborhood Framework

The generalized Hough transform, proposed by Ballard in [4], has shown how stochastic evidence integration techniques can be applied to the recognition of arbitrary shapes in 2-space. However, due to the locality of the parameter estimation technique typical of the Hough paradigm, the complexity of the model search space becomes large once arbitrary rotation and scale transformation are allowed in the input.

Recent work [15], [16] has shown how it is possible to further generalize the notion of parameter transform to include a mechanism for fusing evidence embedded in distant portions of the image. This is achieved by extending the neighborhood concept to include compact nonconnected data set (generalized neighborhoods) and by devising transform operators for this new sparse data structure.

In the usual parameter transform formulation, a local mapping operator $f(\mathbf{P}, x, y)$ is devised to estimate the likelihood of the presence of a specific feature $P = (p_1, p_2 \cdots, p_s)$, in a small local neighborhood $A(x, y)$ in the image space, centered around the point (x, y). Here, the parameters $p_1, p_2, \cdots, p_S$ uniquely identify the feature of interest. Evidence integration is performed by integrating the estimator over the entire image:

$$\mathbf{P} = \int_{\text{Image}} f(\mathbf{P}, x, y) dx\, dy. \tag{23}$$

The resulting value is a confidence measure on the presence of the corresponding feature in the input. By repeating the operation for each possible feature, a density function over the parameter space of the features is generated. Peaks in the density function indicate high values of confidence with respect to the corresponding feature. Selection techniques are used to isolate the peaks from the background noise.

The mapping operator, in the case of generalized neighborhoods, is structured as a shape autocorrelation function of the form

$$E(\mathbf{P}, x, y,) = \int_{A(\{(x,y)\})} \cdots \int_{A(\{(x_{k-1}, y_{k-1})\})} [f(\mathbf{P}, x, y, x_1, y_1, \cdots, x_k, y_k)]^2 dx_1 dy_1 \cdots dx_k dy_k, \tag{24}$$

where k determines the order of the correlation function. $A(\{(x, y)\})$ is a neighborhood for partial evidence integration which can depend on the values of the different (x, y). Such neighborhoods, usually, can be the entire image or a portion of the image centered around (x, y).

Parameter transforms based on generalized neighborhoods achieve two or more orders of magnitude better accuracy in the parameter estimation of simple parametric features. They have been used to reliably extract up to eight-dimensional parametric features, such as conic sections in 3-space from range data [16]. Specific advantages of autocorrelation mapping operators are discussed in [16].

In conventional nonparametric feature extraction paradigms, such as the generalized Hough transform, analytical operators

are replaced by a look-up table. Identically, in our framework, shape autocorrelation operators are replaced by a multidimensional table look-up mechanism where the indexes are function of various parameters extracted at different location over the image. For instance, given three points $(x_1, y_1), (x_2, y_2)$, and (x_3, y_3), one could have an index $\langle \frac{s_1}{S}, \frac{s_2}{S} \rangle$, where

$$\begin{aligned} s_1 &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\ s_2 &= \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} \\ S &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\ &\quad + \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2} \\ &\quad + \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}. \end{aligned} \tag{25}$$

Here, the index $\langle \frac{s_1}{S}, \frac{s_2}{S} \rangle$ is invariant with respect to similarity transformations.

VI. Index Computation

We present the general scheme for computing invariant indexes using spatial autocorrelation operators. Since the basic principles for computing indexes is the same for both the local curve-feature detection stage and the following object detection (or acquisition) stage, they are treated here together. Later, we will describe in detail individually the procedure for detecting local features and the procedure for object recognition/acquisition.

A. Shape Autocorrelation Operators

In its most general formulation, we define a *shape autocorrelation operator* of order k to take a combination of k different feature points as an input and to return an n-tuple, called an *index*. The index is computed on the basis of the geometrical relationship between the k points and local properties at those points. These local measures can include the 0th- and 1st-order properties (location and tangent) of the edge points or higher order local information such as local curve shape. The combinations of k points considered can be constrained in various ways. For example, to extract local curve shape, the combinations can be limited to points lying in small local neighborhoods on the same connected curve piece, while for complete object recognition, the points may lie in distant portions of the image and on different curves. The operator of (25) that generates the 2-D vector $\langle \frac{s_1}{S}, \frac{s_2}{S} \rangle$ from three points, for instance, is one of the simplest possible 3^{rd}-order shape autocorrelation operators.

In a k^{th} order shape autocorrelation transform, first a set of k-tuples of points is generated from an image. Successively, shape autocorrelation operators are applied to the k-tuples, and the result is used to index in a look-up table. Finally, the output from the table is used to produce a density function on a parameter space where vectors correspond to specific feature instances.

B. Index

In the domain of similarity transforms on 2-D shapes, 3rd-order spatial autocorrelation operators are more than adequate, since rotation, scale, and translation between model and model instance can be uniquely determined from two corresponding points. Each feature point in the image can be associated with all possible combinations of two other points, thus forming a set of triangles (see Fig. 7). For each triangle, for instance, we can generate a four-dimensional index $\langle \frac{s_1}{S}, \frac{s_2}{S}, \alpha_1, \alpha_2 \rangle$ consisting of two ratios $\frac{s_1}{S}$ and $\frac{s_2}{S}$ (where $S = s_1 + s_2 + s_3$), which determine the geometry of the triangle, and the two angles α_1 and α_2, which describe 1^{st}-order properties of the curve with respect to the given 3-point set. Here, α_1 is the angle between the tangent at $p2$ and line $p2p3$; α_2 is a similar angle for point $p3$ (see Fig. 7). While the tangent at the first point can also be used as a part of the index, we have chosen not to do so. Then, if the tangent measurement at $p1$ is in error, at least all the indexes generated here will not be in error.

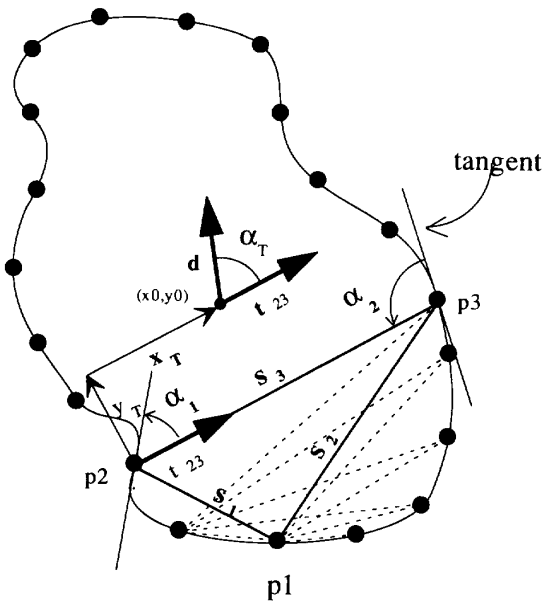

Fig. 7 Computing index from feature triplets.

Different quantization criteria can be applied to each of the index parameters to limit them to a finite number of discrete values (in our experiments, this number ranges from 8 to 32). The generated index is invariant to similarity transformations. In fact, if other global properties of the tuples of points are also invariant, they can be used to form indexes of higher dimensions, as we will show later. In general, all ratios between visual parameters (e.g., intensity, curvature, etc.) are good candidates for invariance.

VII. Shape Acquisition and Recognition

We present the general scheme for acquiring and recognizing shapes. The basic procedure is the same for both the local curve-feature detection stage and the following object detection (or acquisition) stage. For local curve features, the models are a small set of curve shapes (4 in our present implementation), and the feature detection process consist of recognizing these curve shapes. For object shapes, the local curve shapes serve as features. The object models are acquired

from characteristic views of the object, and recognition is performed on images following feature detection.

A. Acquisition

During the acquisition phase, we consider a single, unoccluded 2-D shape in the image so that the position of its center of mass (x_0, y_0), its scale μ (default value is 1.0), its orientation vector $\mathbf{d}$ (Fig. 7), and a symbolic label $\mathbf{L}$ are known. In the case of 3-D shapes, different views corresponding to different aspects of the object [39], [19] can be acquired. Alternatively, given the high computational cost of finding different aspects of an object [27] and the ability of our system to recognize shapes from partial instances and with some amount of projective invariance (Section XI), the Gaussian viewing sphere can be sampled at regular intervals. Finally, as shown in [40] and [21], 3-D pose can be recovered directly from 2-D contours using tuples of points.

For each triplet of feature points, we compute the index and another n-tuple, called an *entry*, containing four elements:

1) The label $\mathbf{L}$ of the object
2) The position (x_T, y_T) of the center of mass in the new right-hand coordinate system defined by the normalized vector $\mathbf{t}_{23}$ and its corresponding orthonormal one
3) The ratio $\rho = \frac{\mu}{S}$.
4) The angle α_T between the two vectors $\mathbf{t}_{23}$ and $\mathbf{d}$.

These parameters depend on scale, orientation, and location and are used to recover the position, of an object shape from the corresponding 3-point sets. The generated discrete index addresses a bucket in a look-up table to which we add the entry $\langle L, (x_T, y_T), \rho, \alpha_T \rangle$.

B. Recognition

During recognition, in a similar fashion, all possible combinations of three edge points from an image are used to generate a set of indexes. From each entry contained in the table-bucket addressed by the index generated by a triplet of points, we obtain a shape label $\mathbf{L}$ and compute the location, orientation, and scale of this shape $\mathbf{L}$.This triplet votes for an instance of shape $\mathbf{L}$ in the recovered pose. A *shape instance* is one particular instance of a shape and consists of a shape label and specific values for location, rotation, and scale. The process is repeated for all triplets of points, and votes are accumulated for the different shape instances computed from the entries. Triplets located on the same shape in general provide correct estimates for the parameters, thus accumulating votes for the correct shape instance. Other triplets produce pseudorandom results that are scattered over numerous shape instances and result in negligible accumulation.

Let us define *discrimination* = $\frac{V_c}{V_w}$, where V_c are the votes for the correct shape instance and V_w the maximum votes received for an incorrect shape instance, as a measure of the capability of this technique to distinguish between different shape instances. If a shape instance A shares a fraction p of points and their local properties with another shape instance B, then the discrimination between the two shapes is approximately p^{-3}.

For each shape hypothesis, we record the features used to generate its value, and record the number of times it was indexed, as votes for that hypothesis. We assume that distinct objects in the image do not share features. Thus, different shape hypotheses generated by overlapping feature subsets compete, and only the one with highest number of votes is selected. Since these selected shapes do not share image data, they inherently constitute a segmentation of the input image.

By modifying the number of points, properties, and relationships considered, the same framework of evidence integration can be extended to other domains. In 3-space, for instance, one could use the relative position of combinations of three points from a range data image. The correspondence between three model and three range points fully determines 3-D object position. The 1st-order properties (surface normals) can be represented, for instance, by the angle between the tangent planes and the plane containing the three 3-D points.

VIII. Local Shape Features

The scheme outlined above for computing indexes could be used directly for recognizing object shapes with edge-points serving as features. However, the dimensionality of the index computed from just edge points is limited to at most five (of which we used only four above for noise rejection). Also, the $O(n^3)$ computational complexity with respect to the number of points n considered makes the approach unappealing when large data sets are explored.

To overcome this difficulty, we can consider only a small specific subset of the edge points. In this case, the stability and robustness of the transform would depend on reliably selecting the same subset of points when acquiring and recalling the shapes, that is, selecting the same points for the same objects in different images. Our solution is to encode local 2-D shape of edge-curves into a small set of localized and symmetric shape descriptors. Next, we use this intermediate representation as the features for the spatial transform that computes indexes for object shapes.

Curve shapes have been labeled based on curvatures measures [35] and by fitting parametric functions [56]. However, derivatives and retrieving the curve parameters from an image is a hard task requiring optimal segmentation of the input. Also, the existence of a parametric representation does not guarantee its stability. In other words, small variations in some of the parameters can produce arbitrary large variations in the shape of the associated 2-D curve, and vice versa. Thus, matching the extracted representations to the stored ones becomes almost impossible. In these cases, parametric representations are not viable candidates for either the acquisition or the recognition process. Curvature requires computation of high-order derivatives, and is scale dependent.

We have chosen to use the same framework we have for object recognition to also serve as a feature detection. Thus, we have replaced feature detection by model-based recognition. A number of curve shapes (four in the current implementation) are chosen as models. These models are acquired by presenting digitized images of the curve-shape under various similarity transformations. The best curve shape at each edge point is

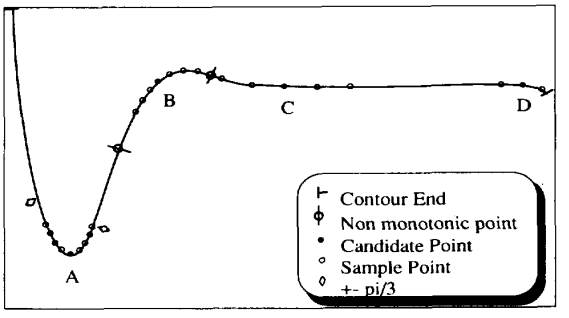

Fig. 8. Sampling.

recognized using indexing. Then, among the overlapping curve shapes, the best ones are chosen, giving a segmentation of the edge-curves into nonoverlapping labeled sections. The location of the curve-shape is given by the pose computation.

A. Detecting Local Shapes

The framework for the acquisition and extraction of local shape descriptors is analogous to the one described in Section VII-A. While global (object) shapes are not associated with any specific point on the shape and are positioned through their centers of mass, local shapes are associated with a specific point on the shape. For instance, elliptical arcs are associated with points where curvature reaches local extrema. Four local shape descriptors used by us are lines, circular arcs, and elliptic arcs (minima and maxima of curvature). Larger sets of local shape descriptors can also be considered [1], [9].

Given the image of an object, edges are extracted using standard techniques such as a Canny edge detector [14]. Edges are then linked into curves using simple eight neighbors connectivity. We assume that the shape of the object is completely captured by these 2-D edge-curves. However, no assumption is made on the connectivity of the contours other than piecewise continuity. Hence, noisy or broken contours are acceptable.

Given a point (x_0, y_0) on a edge-curve, we symmetrically sample around that position along the curve (see Fig. 8), generating a set $\{(x_i, y_i), -N \leq i \leq N\}$. The sampling step is proportional to the length of the longest symmetrical interval around the point for which the tangent behavior is monotonic on a coarse scale, and the total tangent variation is less than $\frac{2\pi}{3}$. It is also inversely proportional to the total variation in tangent angle on the symmetric interval for tangent angle less than $\frac{2\pi}{3}$. We assume that faster variations in tangent correspond to smaller features, which accordingly require finer sampling for detection. As the sampling is based on angles and not on distances, it is scale independent. Next, we form all possible combinations of the point (x_0, y_0) with two others from the set $\{(x_i, y_i)\}$. The index $\langle \frac{s_1}{S}, \frac{s_2}{S}, \alpha_1, \alpha_2 \rangle$ described in Section II-B, is computed from each triplet and it is used to address a bucket in the local-shape look-up table. The point to note here is that all possible triplets of edges on the curve are not considered. One of the points is *always* at the center of the symmetric sampling; the second point always comes from one side of this point and the third from the other side.

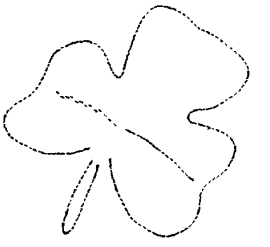

Fig. 9. Shape of leaf # 1.

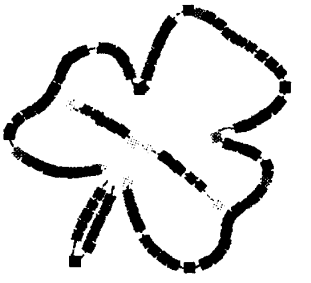

Fig. 10. Local shape descriptors.

During acquisition, images of the curve shapes are presented in various poses. The computation for indexes is performed with (x_0, y_0) fixed at the center of symmetry for the model curve shape. Since during acquisition one of the points in the triplets is always at the center of symmetry of the curve, the strongest response during recognition will also be when the edge point (that we are sampling symmetrically around) is actually at the center of symmetry for its local curve shape. For each triplet, the entry $\langle \lambda, \beta, S \rangle$ is inserted in the table. Here, λ is the symbolic label for the local curve shape. β is the angle between the vector $\mathbf{t}$ and the normal to the contour $\mathbf{n}$, required to recover the shape orientation from the triplet. $S = s_1 + s_2 + s_3$ is used for scale normalization.

During recognition, given (x_0, y_0) and the sample set $\{(x_i, y_i)\}$, triplets are formed and indexes computed in an analogous fashion. For each entry in the table addressed by these indexes, a feature instance is computed that uniquely identifies the local shape and its orientation and scale. After all the triplets have been considered, the feature instance with the highest data support (i.e., votes) describes the shape around (x_0, y_0). By considering the subset of $\{(x_i, y_i)\}$ that successfully voted for a given feature, it is also possible to recover the section of the contour associated with it. The process is repeated for each edge point. Since sampling rates are recomputed for each point, sample density varies dynamically along the curve.

After the feature detection phase, local descriptors supported by overlapping portions of a curve are compared. From each overlapping set, the feature with the highest number of votes is chosen. In this way, curves are naturally segmented into a small set of nonoverlapping localized shapes (see Figs. 9, 10, and 11). These local descriptors are positioned at coarsely sampled contour locations. To increase stability, a finer localization is obtained by extracting new descriptors on

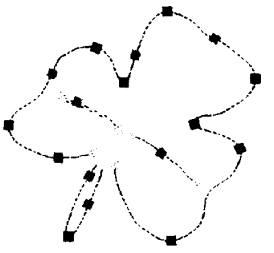

Fig. 11. Selected descriptors.

a pixel-by-pixel basis, in a small neighborhood of the original local descriptors, and selecting again from among the results the one with highest support. In our experimental testing, this approach has shown stability with respect to scale, rotation, translation, and limited projective transformations of the input data.

IX. Object Shapes

Object shape acquisition and recognition is performed on images after the local curve shape detection stage. The overall scheme follows that outlined in Section VII. One main difference is that the index is now seven-dimensional, with the additional information provided by the feature labels. Also, a number of heuristics to improve speed are introduced.

A. Model Acquisition

The shape database is similar to associative memories [32], which usually employ distributed representations, i.e., each global shape (object) description is not localized to a particular location but is distributed over the memory. Such a holographic representation engenders fault tolerance to loss of parts of the memory. The distributed representation of visual shapes is contained in the shape table and selection mechanisms described in Section II. Shapes are represented as a collection of entries distributed over the table. Tables, because of their extreme sparseness (see Section III), are implemented as hash tables.

New shapes are acquired by performing a shape autocorrelation transform on presented instances from images. To enhance recognition in the presence of noise and small projective transforms of the input, two heuristics are employed. Different instances of the same shape in different poses are acquired. For each object, the acquisition is repeated on different views until the indexes no longer hit empty buckets in the shape table with a significant rate. Secondly, a stochastic index perturbation mechanism during acquisition is performed. That is, a small randomness (order of the index parameter quantization) is added to a index along each of its dimensions. This, while not modeling the noise and its effect on the index, increases the fault tolerance of recognition.

One potential problem is that an incorrect measurement may be caused by incorrect noise and segmentation and, yet it will still be recorded in the library. Insuring that a minimum number of votes be measured from an index before it is accepted as a valid invariant [48] will help alleviate this problem.

The local shape of a curve around a point provides additional dimensions for the indexes used for object shapes. The index becomes $\langle \frac{s_1}{S}, \frac{s_1}{S}, \alpha_1, \alpha_2, \lambda_1, \lambda_2, \lambda_3 \rangle$, where λ_i is the symbolic label for the local curve shape at point (x_i, y_i). These indexes are of a higher dimensionality, endowing the transform with even more selectivity. The scale (or size) of the local curve descriptors could be used to generate additional parameters for the index, but these are not used in this paper. The entries used and the acquisition process is the same as described in Section VII-A.

B. Object Shape Recognition

To recognize objects in an image, the first stage is feature detection. Edges are detected in the image and local curve features then detected using the scheme detailed in Section VIII. Triplets of local-curve shapes are used to generate the seven-dimensional indexed as described in the previous section. The object recognition scheme proceeds as described in Section VII-B.

A direct implication of the use of 3rd-order autocorrelation function is $O(n_s^3)$ time complexity, where n_S is the number of local shapes. It is further possible to reduce the time requirement to $O(n_s)$, as shown below.

Local shape descriptors are assigned a weight w_r, proportional to their visual relevance. This measure uses the length of their support normalized with respect to the size of the model (this favors large features) and/or the tangent angle variation per unit length that they describe (this favors corners). Larger features have a higher chance of being correctly identified from the input. Rapid variations in tangent correspond to curvature maxima that are well localized on contours. For each model, we isolate a small constant number c of such highly relevant descriptors $\mathbf{D}_r = \{D_i : i \leq c\}$ from the others $\mathbf{D} = \{D_j : j \leq n_s\}, \mathbf{D}_r \subseteq \mathbf{D}$. We then generate all possible triplets with the first element from the set $\mathbf{D}$ and the other two from the set $\mathbf{D_r}$. The total number of triplets formed, $n_s\binom{c}{2}$, is a linear function of n_s. Thus, the time required to acquire a shape model, or one view for a 3-D object, is $O(n_s)$. This representation is still highly redundant since each feature is present in at least $\binom{c}{2}$ triplets.

Furthermore, since real-world objects have in general a compact structure, it is possible to exploit this property to reduce the number of triplets considered. For each identified descriptor we generate a circle of coherence centered at its location. The radius of this circle is proportional to the size of the local shape. Other descriptors from the same shape are more likely to be found within such a circle. This technique has been demonstrated to reduce the number of triplets from cubic to a linear with respect to the image size [16].

To make recognition robust with respect to noise in cluttered environments, we extend the radius of coherence until a required constant number of descriptors with desired visual significance w_r, normalized with respect to that of the considered descriptor, are found, and we create triplets only with these. This search can be made efficient using multiresolution

analysis techniques [13], or by maintaining the features in a heap structure. In either case, maximum search time is within $O(n \log n)$.

The heuristics of saliency and locality used here to reduce recognition time are similar to the local focus feature approach of Bolles and Cain [10] and the salient feature methods of Turney *et al.* [57]. These heuristics are only approximate, and they can fail. Under our scheme, it is not possible to assign saliency in a totally scale-independent manner. Similarly, locality is only a very crude approximation to object level segmentation. Sophisticated bottom-up segmentation techniques such as perceptual organization [45] will obviously aid this system but have not yet been incorporated into it. Therefore, while saliency and locality can often reduce the recognition time to linear in the number of features, for images where these approximations fail features other than the salient ones are considered and the search is expanded over the full image.

X. Subpart Identification

We introduce a method for using indexing to automatically detect subparts in the acquired objects. This, coupled with the flexible structure of the global shape indexes, allows for the formulation of a hierarchical organization of the shape database. Such a layered structure of the shape table, however, has not yet been implemented in the current system.

Complex object shapes can be broken into smaller and simpler components that are termed subparts. These subparts, if considered as separate shapes, may recursively be broken into their component subparts. Subparts allow description of complex shapes in terms of simpler ones rather than directly in terms of primitive features, in this case the local shape descriptors. Thus subparts allow a hierarchical representation of shape, with the shape hierarchy starting from local features, through subparts and finally object shapes. This way of structuring the information allows for more compact representation, more efficient extraction and higher data abstraction capabilities. The hierarchical shape representation is useful for recognition, shape representation [23], and possibly in reasoning about the objects. In indexing systems, the number of levels can be increased by using subparts. This will add to the dimensionality of indexes used for object recognition.

The different varieties of subparts can be categorized by whether they can overlap or are strictly nonoverlapping, and by the technique used to categorize a part of a shape as a subpart. For example, in [23] subparts are nonoverlapping and are formed by segmenting the shapes at corners; in [45] subparts can be overlapping and are segmented on the basis of perceptual organization criteria.

We define subparts as locally connected parts of one shape that are shared (rotated and scaled) among a significant number of other shapes. The subparts can be overlapping. There is a lower bound on the number of shape descriptors required to form a subpart, as very small sections of a shape (such as a portion of a curve or straight line) can be shared by numerous objects and lack descriptive power. Our definition of subparts does not depend on segmentation, i.e., on the reliable decomposition of different instances of a shape in an identical way. However, we can take advantage of any subpart segmentation method to speed up our computation by constraining the generalized neighborhoods to the segmented sections.

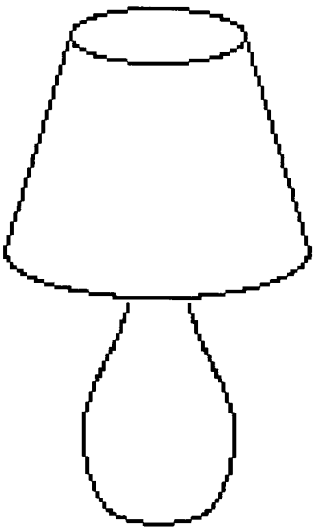

Fig. 12. Lamp # 1.

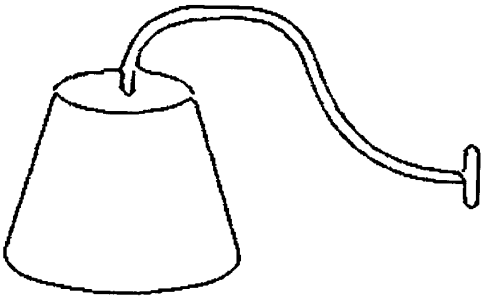

Fig. 13. Lamp # 2.

In our framework, the concept of subpart finds natural expression in terms of evidence integration across several models. Given a table containing a large number of shapes, we use the indexes generated during the acquisition process to extract information from the image, based on previously acquired shape information. If a significant match is found between the new shape and other shapes from the database, the set of matching elements (i.e., single local shape descriptors that produced matching triplets) is analyzed as a possible candidate for a subpart. If it satisfies some criteria, such as being compact and connected, it is classified as such and inserted in the subpart database.

The experimental result of unsupervised subpart detection from the set of objects shown in Figs. 12, 13, 14, and 15 is shown in Fig. 16.

Subparts assume semantic roles identical to local shape descriptors. That is, they are located at their center of mass and assigned a label, an orientation, and scale. As such, they participate in the recognition of the shapes they belong to, by

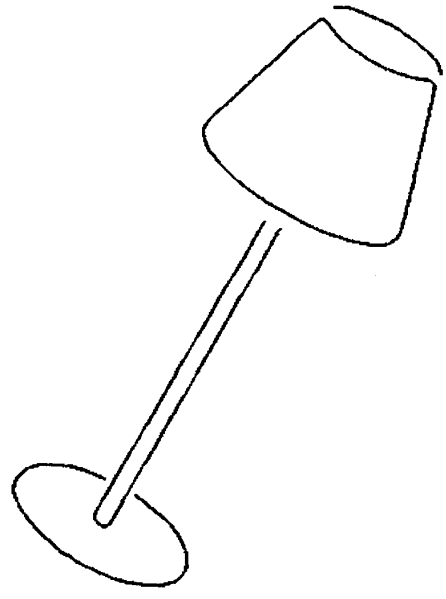

Fig. 14. Lamp # 3.

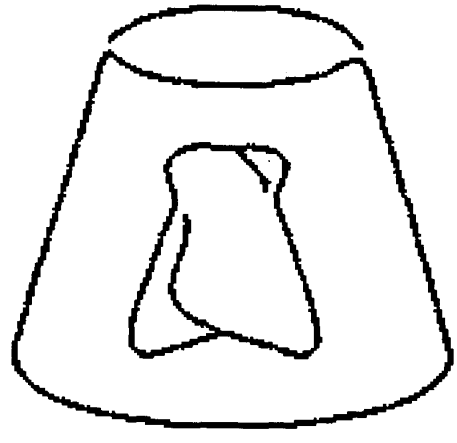

Fig. 15. Lamp # 4.

forming triplets with other local descriptors, including other subparts and generating indexes in the shape table.

More formally, suppose there are two distinct objects A and B and we wish to know the subparts shared by them. The process of obtaining the subparts is:

1) Detect the local shape descriptors for object B and perform recognition on the global shape table that already contains a representation of A.
2) If there are no significant votes accumulated for the object A, then B and A do not share any subpart.
3) If significant votes are obtained for object A, then B and A are likely to share a subpart. For each shape instance of A with significant votes the local shape descriptors in B that voted for that instance are known. Note that more than one shape instance of A can be created if A and B share more than one subpart at different scales or relative position and orientation, or due to spurious votes that do not lead to a valid support. From this set, we select features that have participated in more than $\frac{N_t}{2n}$ votes, where n is the number of used shape descriptors, i.e., we select those whose votes were not produced by stochastic or correlated noise [16]. N_t is the total number of triplets formed by the descriptors.
4) Next, we check if the selected features form any subpart that A and B share. To belong to a subpart, the features must exhibit spatial coherence, i.e., they must be related by spatial proximity and continuity along edge-contours. Continuity is checked by looking for overlap or adjacency in the edge support for each feature. If an edge-contour has more than 10% of its features missing, then it is not a viable candidate for forming a subpart, and the subpart itself is not considered. Therefore, for a set of selected local shape descriptors in B to be a subpart of A, the features must belong to a spatially coherent segment of B and only to that segment of B. A minimum number of spatially coherent features of B must be selected, insuring that the entity defined as a subpart is more than simply a local shape or a small part of curve.

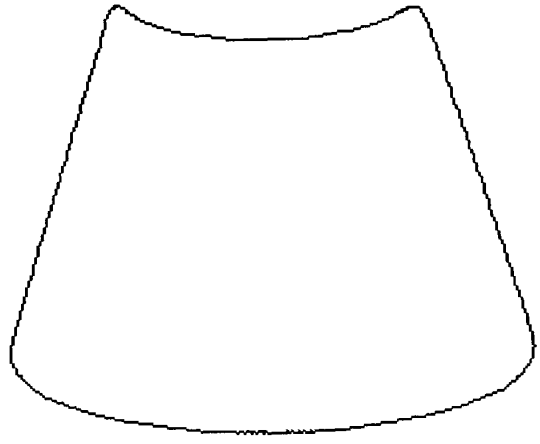

Fig. 16. Detected subpart.

Subpart B is not limited to sharing subparts with a single object. If more than one set of features has significant vote, each of those feature sets has to be checked for being a subpart (steps 3 and 4).

The process described above is for finding common subparts between object models during their acquisition. We have not yet used subparts for recognition purposes.

XI. Experiments

We report some experiments testing the performance of the multidimensional indexing scheme presented before. These experiments provide some empirical test of one multidimensional indexing system. It is difficult to test and quantify how well a system will perform under real-world conditions. There is no way of trying out all real-world situations with varying numerous parameters like object shapes, viewing angles, distances,

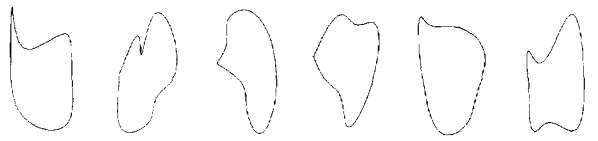

Fig. 17. Some of the objects used for testing database size.

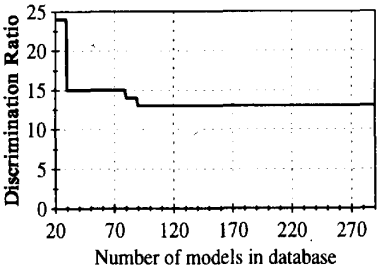

Fig. 18. Average discriminability vs. size of database.

scales, noise, occlusion, and scene complexity. That is exactly why we provided a computational framework for analyzing indexing. The system implementation and experimentation are to be treated more as a proof of concept.

The system was implemented in Lisp on a Symbolics 3650. The size of the shape table was 2^{20} with the seven index dimensions (see Section IX-A) quantized to $2^4, 2^4, 2^3, 2^3, 2^3, 2^2, 2^2$, and 2^2 levels, respectively. For the object hypotheses table H_1, the number quantization levels for pose parameters was 24 for (x_T, y_t), and 8 for ρ and α_T. Note that the quantization levels for the axes is much lower than the 10^2 value typically used in geometric hashing, and the shape table size is larger by a factor $\approx 10^2$.

A. Model Database Size

To test the recognition performance as a function of the number of object models in the shape library, we sequentially added 290 models to the shape memory and tested the recognition of the first five objects introduced into the shape table. Some of the shape models used are shown in Fig. 17. Discrimination ratio (Section VII-B) was used as the parameter to evaluate recognition performance. The shapes were generated by a random process and contain a similar number of local features (5 to 10) and are of similar size (perimeter). Thus, the recognition task was hard in the sense that there was no large variation in shape.

Fig. 18 shows the plot of discrimination versus number of objects in the database. The plotted discrimination value is the average discrimination for the five test shapes. It is interesting to note how performance degraded abruptly once the first few shapes were added to the database but the asymptotic behavior shows a discrimination power larger than 12, i.e., correct instances received at least 12 times more votes than the number of votes for the next best shape.

The experiment indicates that the indexing scheme adopted is good in the sense that the table does not saturate quickly. This experiment is limited in that the recognition test was carried out on noise-free, unoccluded, and uncluttered scenes.

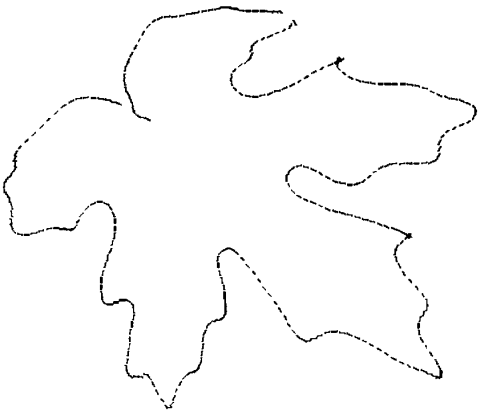

Fig. 19. Shape of leaf # 2.

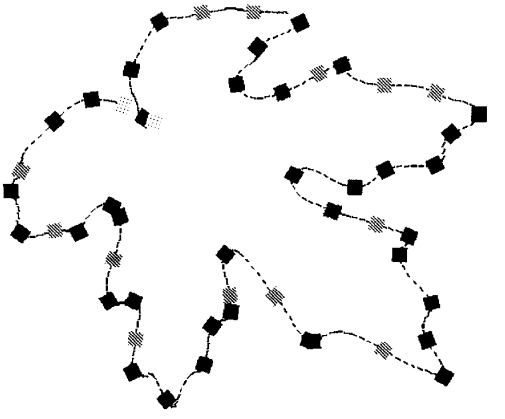

Fig. 20. Local shape descriptors.

The average recognition time was 6 s per object. On the average, there were 5 local shape descriptors per object, resulting in 125 indexes per object for acquisition and recognition.

B. Recognition

We report some experiments as examples of invariance of recognition capabilities to geometric transformations of the model in the image.

We selected the domain of leaf shapes to demonstrate the acquisition and recognition of complex nonparametric shapes. The images used for the experiments are obtained from a database of drawings. The drawings were treated as images and the sampled drawings treated as edge maps. Thinning, contour linking, etc., were run on these edge maps just like on conventional edge maps obtained from intensity images. The images are cleaner than real images, but they provided more control over the experiments. Two of the leaves from this set are shown in Figs. 9 and 19. In this and following figures we show the contours obtained by linking the edges. The contours have been smoothed with a Gaussian filter. The local shape descriptors detected along these two leaves are shown in Figs. 11 and 20.

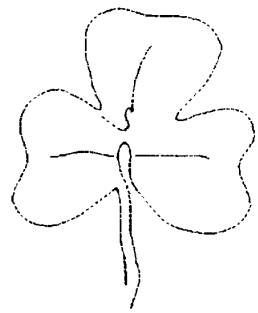

Fig. 21. Rotated leaf.

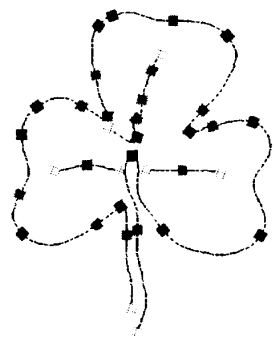

Fig. 22. Local descriptors.

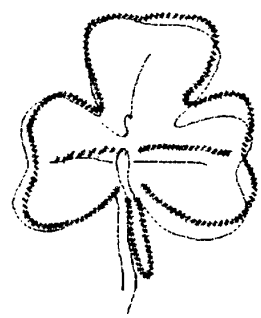

Fig. 23. Recognition/location.

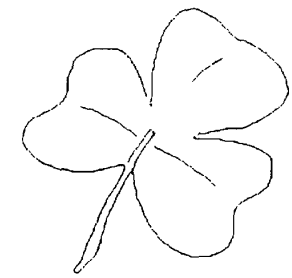

Fig. 24. Magnified leaf.

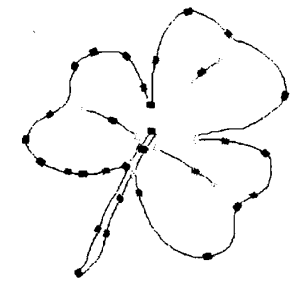

Fig. 25. Local descriptors.

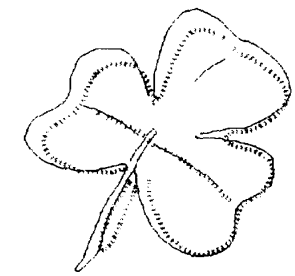

Fig. 26. Recognition/location.

1) *Rotation and translation in the image plane:* An image of leaf #1 that has undergone rotation and translation in the image plane is considered (Figs. 23 and 24). The system correctly identifies the shape and the parameters for rotation, translation, and scale. The original leaf #1 model, translated and rotated with the recovered amounts, is overlaid (Fig. 25). The difference in registration is due to the coarsely quantized computation of the geometrical transform parameters, e.g., the position of the center of mass. If needed, a more precise registration of the model to the image can be obtained by reducing the size of the quantization, rematching only the selected local shapes, and accumulating evidence only for the matched model.
2) *Scaling:* An image of the leaf is taken at a shorter distance, leading to projected shape of the leaf being 1.8 times larger than the model (Figs. 26 and 27). The recognition phase returns the correct label for the leaf and the correct scale factor. Fig. 28 shows the recognized model, scaled by the computed scale factor, and projected onto the image.
3) *Skew:* Slant and/or tilt of the object corresponds to viewing the object from different viewpoints. This introduces a skew in the projected shape of the object—for example, a rectangle appears as a parallelogram. A leaf is viewed with the image plane slanted *and* titled about $\frac{\pi}{6}$ from a plane parallel to the leaf (i.e., projective transform, Figs. 29 and 30). There is also some magnification due to differences in the viewing distances. Even given the relatively large change in viewing direction, the correct model is recognized. This model, with the scale and translation accounted for, is overlaid on the image in Fig. 31. Currently, recovery of viewing directions is not incorporated since objects are 2-D shapes.
4) *Views, generalization, and separability:* The current system handles 2-D shapes. One way of handling 3-D

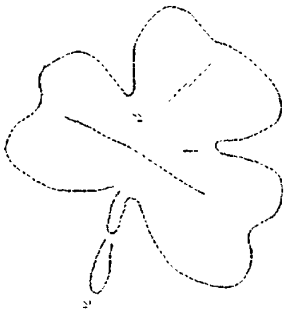

Fig. 27. Skewed leaf.

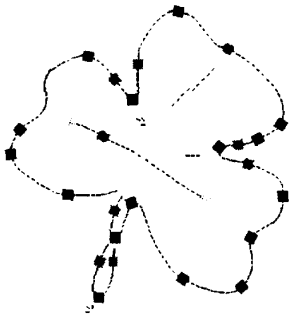

Fig. 28. Local descriptors.

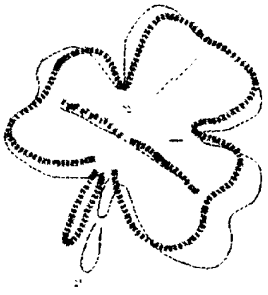

Fig. 29. Recognition/location.

objects is to have different 2-D *views* of the 3-D object. These views can sample the Gaussian viewing sphere or be obtained from an aspect graph. *Generalization* is the ability of a recognition that on being presented with a shape that is not in its database, it categorizes it as the shape it is most similar to. *Separability* means if the unlearned shape, which is categorized as a shape similar to it, is subsequently learned, new presentations of it get correctly categorized to the newly learned shape.

To show that viewpoint direction parameters could be recovered by learning the projected representations of the model at different viewing angles, we acquired the skewed leaf shape as a separate leaf model. In subsequent recognition of the skewed shape, the correct instance is recognized. This not only indicates that the system can be incremented to handle 3-D shapes but also the factors of generalization and separability, i.e.,

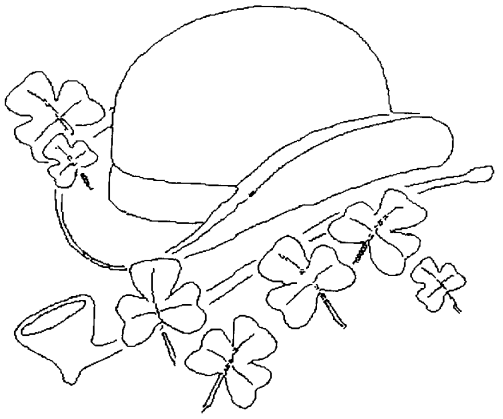

Fig. 30. Complex scene.

shapes are categorized as the most similar objects in the shape memory (generalization), but after learning of the shape, the system is able to distinguish between that similar shape and the newly learned one (separability).

5) *Clutter and Occlusion:* Finally, to show that the system works correctly for complex scenes, we analyze an image, shown in Fig. 32, containing multiple instances of the same object at different locations, and in the presence of other objects not in the database. All the instances of leaf #1 are correctly recognized. Due to the presence of multiple objects, other models in the memory also receive some votes. These models are suppressed by the constraint satisfaction mechanism. Again, the difference in the registration of the object model with the image (Fig. 33) reflects the quantized values used for computing the transforms.

The average recognition time was 25 s per object for the similarity transform and 5 min for the cluttered scene. On the average, there were 30 local shape descriptors per object, resulting in 1500 indexes per object.

XII. Conclusion and Future Directions

We have provided a computational framework for the analysis of indexing-based model acquisition and recognition systems. This analysis shows that high-dimensional indexing strategies can solve a number of problems that have hampered conventional look-up table techniques. The benefits include improvements in recognition time, discriminating power, and maximum size of the database. As a proof of concept, a high-dimensional indexing system for the automatic acquisition and recognition of complex visual shapes has been presented. The associative memory structure used for storing and recalling shapes instances exhibits the properties of robustness, generalization, and recall from partial descriptions. The mechanism is also capable of unsupervised detection of subparts shared by newly acquired and stored models.

Such a memory scheme is efficiently implemented both in terms of time and space requirements. Model acquisition is accomplished in time complexity of $O(n^3)$, where the model is described by n local shape descriptors. The requirement

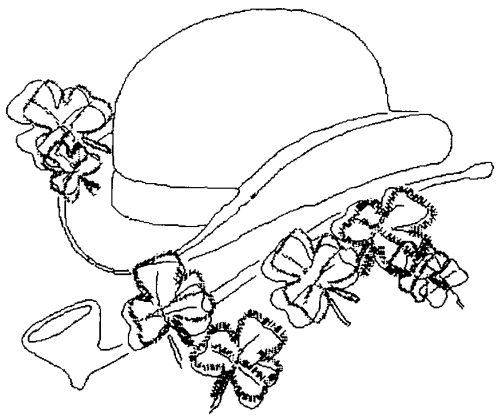

Fig. 31. Recognition/location.

for memory space is $O(mn)$ to $O(mn^3)$, with m the number of distinct object shapes (or distinct object shapes times their different aspects for 3-D shapes) and n the average number of local shapes per model. Recognition time grows very slowly with the number of models in the database and is given by $N_{Tr}(k + cQ)$, where N_{Tr} is the number of feature triplets in the image, k and c are constants, and Q is the number of models in the database. c in our case is as low as 0.001.

Current work is focusing on the extension of the multidimensional indexing paradigm to 3-D shapes and applications beyond object recognition.

REFERENCES

[1] H. Asada and M. Brady, "The curvature primal sketch," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 1, 1986.

[2] N. Ayache, O. Faugeras, and B. Faverjon, "A geometric matcher for recognizing and positioning 3-D rigid objects," *Int. Conf. Intelligent Robots and Computer Vision, Proc. SPIE*, Oct. 1984.

[3] D. H. Ballard, "Parameter nets: A theory of low level vision," in *Proc. 7th Int. Joint Conf. Artificial Intelligence*, Aug. 1981, pp. 1068–1078.

[4] ——, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recogn.*, vol. 13, no. 2, 1981, pp. 111–122.

[5] R. M. Bolle, A. Califano, R. Kjeldsen, and R. W. Taylor, "Computer vision research," in *Proc. DARPA Image Understanding Workshop*, Palo Alto, CA, May 1989.

[6] R. M. Bolle, A. Califano, R. Kjeldsen, and R. W. Taylor, "Visual recognition using concurrent and layered parameter networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Diego, June 1989.

[7] B. Bhanu and O. D. Faugeras, "Shape matching of two-dimensional objects," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 2, pp. 137–155, Mar. 1984.

[8] P. J. Besl and R. C. Jain, "Three-dimensional object recognition," *Comput. Surveys*, vol. 17, no. 1, pp. 75–145, Mar. 1985.

[9] A. P. Blicher, "A shape representation based on geometric topology: Bumps, Gaussian curvature, and the topological zodiac," in *Proc. 10th Int. Joint Conf. Artificial Intelligence*, Aug. 1987, pp. 767–770.

[10] R. C. Bolles and R. A. Cain, "Recognizing and locating partially visible objects: The local feature focus approach," *Int. J. Robotics Research*, vol. 1, no. 3, pp. 57–82, Fall 1982.

[11] R. C. Bolles, P. Horaud, and M. J. Hannah, "3-DPO: A three-dimensional part orientation system," in *Proc. 8th Int. Joint Conf. on Artificial Intelligence*, Aug. 1983, pp. 1116–1120.

[12] R. A. Brooks, "Model-based three-dimensional interpretations of two-dimensional images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 5, no. 2, pp. 140–150, Mar. 1983.

[13] P. J. Burt, "Smart sensing within a pyramid vision machine," *Proc. IEEE*, vol. 76, no. 8, pp. 1006–1015, Aug. 1988.

[14] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 6, pp. 679–698, Nov. 1986.

[15] A. Califano, "Feature recognition using correlated information contained in multiple neighborhoods," in *Proc. 7th Nat. Conf. Artificial Intelligence*, July 1988, pp. 831–836.

[16] R. M. Bolle and R. W. Taylor, "Generalized neighborhoods: A new approach to feature extraction," in *Proc IEEE Conf. Computer Vision and Pattern Recognition*, June 1989.

[17] A. Califano and R. Mohan, "Generalized shape autocorrelation," in *Proc. AAAI-90*, July 1990, pp. 1067–1073.

[18] A. Califano and R. Mohan, "Multi-dimensional indexing for recognizing visual shapes," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 1991, pp. 28–34.

[19] I. Chakravarty and H. Freeman, "Characteristic view as a basis for three-dimensional object recognition," in *Proc. SPIE Conf. Robot Vision*, 1982, pp. 37–45.

[20] R. T. Chen and C. R. Dyer, "Model-based recognition in robot vision," *ACM Comput. Surveys*, vol. 18, no. 1, pp. 66–108, Mar. 1986.

[21] D. T. Clemens and D. W. Jacobs, "Model group indexing for recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 1991, pp. 4–9.

[22] R. O. Duda and P. E. Hart, "Use of the Hough transform to detect lines and curves in images," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[23] G. J. Ettinger, "Large hierarchical object recognition using libraries of parameterized model sub-parts," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 1988, pp. 32–41.

[24] O. D. Faugeras and M. Hebert, "A 3-D recognition and positioning algorithm using geometric matching between primitive surfaces," in *Proc. 8th Int. Joint Conf. Artificial Intelligence*, Aug. 1983, pp. 996–1002.

[25] J. A. Feldman and D. H. Ballard, "Connectionist models and their properties," *Cognitive Sci.*, vol. 6, 1981, pp. 205–254.

[26] W. E. L. Grimson and T. Lozano-Perez, "Model-based recognition and localization from sparse range data or tactile data," *Int. J. Robotics Research*, vol. 3, no. 3, pp. 3–34, Fall 1984.

[27] Z. Gigus and J. Malik, "Computing the aspect graph for line drawings of polyhedral objects," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 12, no. 2, Feb. 1990.

[28] C. Goad, "Special purpose automatic programming for 3-D model-based vision," in *Proc. DARPA Image Understanding Workshop*, 1983, pp. 94–104.

[29] W. E. L. Grimson, "The combinatorics of heuristic search termination for object recognition in cluttered environments," Massachusetts Institute of Technology, Cambridge, MIT AI Memo 1111, May 1989.

[30] W. E. L. Grimson and D. P. Huttenlocher, "On the sensitivity of geometric hashing," in *Proc. 3rd Int. Conf. Computer Vision*, Osaka, Japan, 1990

[31] C. Hansen and T. Henderson, "CAGD-based computer vision," in *Proc. Workshop on Computer Vision*, Nov.–Dec. 1987, pp. 100–105.

[32] G. Hinton and J. Andreson, *Parallel Models of Associative Memory*: Lawrence Erlbaum, 1981

[33] P. Horaud and R. C. Bolles, "3D PO's strategy for matching three-dimensional objects in range data," in *Proc. IEEE 1984 Int. Conf. Robotics*, Mar. 1984, pp. 78–85.

[34] P. V. C. Hough, "Methods and means for recognizing complex patterns," U. S. Patent 3069654, 1962.

[35] D. P. Huttenlocher and S. Ullman, "Object recognition using alignment," in *Proc. 1st. Int. Conf. Computer Vision*, pp. 102–111, 1987.

[36] D. P. Huttenlocher, "Three-dimensional recognition of solid objects from a two-dimensional image," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MIT AI-Lab. Tech. Rep. # 1045, 1988.

[37] K. Ikeuchi and T. Kanade, "Automatic generation of object recognition programs," *IEEE Proc.*, vol. 76, no. 8, Aug. 1988, pp. 1016–1035.

[38] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir, "Two-dimensional, model-based, boundary matching using footprints," *Int. J. Robotics Research*, vol. 6, no. 4, Winter 1986.

[39] J. Koenderink and A. van Doorn, "The internal representation of solid shape with respect to vision," *Biological Cybern.*, vol. 32, pp. 211–216, 1979.

[40] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "On recognition of 3D object from 2D images," in *Proc. IEEE Conf. Robotics and Automation*, pp. 1407–1413.

[41] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "Object recognition by affine invariant matching," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 335–344.

[42] Y. Lamdan and H. J. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *Proc. 2nd Int. Conf. Computer Vision*, Dec. 1988.

[43] Y. Lamdan and H. J. Wolfson, "On the error analysis of geometric hashing," Robotics Lab, New York University, Tech. Rep. 213, Oct. 1989.

[44] D. G. Lowe, "The viewpoint consistency constraint," *Int. J. Comput. Vision*, vol. 1, 1987, pp. 57–72.
[45] R. Mohan and R. Nevatia, "Perceptual organization for scene segmentation and description," in *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 14, no. 6, June 1992, pp. 616–635.
[46] J. L. Mundy and A. J. Heller, "The evolution and testing of a model-based object recognition system," in *Proc. 3rd. Int. Conf. Computer Vision*, pp. 268–282, 1990.
[47] R. Nevatia and T. O. Binford, "Description and recognition of curved objects," *Artificial Intell.*, vol. 8, no. 1, pp. 77–98, 1977.
[48] Suggested by one of the referees, during the review process.
[49] I. Rigoutsos and R. Hummel, "On a scalable parallel implementation of geometric hashing on the connection machine," Courant Inst. of Math. Science, New York Univ., Tech. Rep. 554, Apr. 1991.
[50] I. Rigoutsos and R. Hummel, "Robust similarity invariant matching in the presence of noise," in *Proc. 8th. Israeli Conf. Artificial Intelligence and Computer Vision*, Dec., 1991.
[51] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic, 1982.
[52] D. E. Rumelhart and McClelland, Eds., *Parallel Distributed Processing: Explorations in the Microsructures of Computing*. Cambridge, MA: MIT Press, 1986.
[53] D. Sabbah, "Computing with connections in visual recognition of origami objects," *Cognitive Sci.*, vol. 9, no. 1, Jan.–Mar. 1985, pp. 25–50.
[54] R. Shapira and H. Freeman, "Reconstruction of curved-surface bodies from a set of imperfect projections," in *Proc. 5th Int. Joint Conf. Artificial Intelligence*, Aug. 1977, pp. 22–26.
[55] F. Stein and G. Medioni, "Efficient two dimensional object recognition," in *Proc. Int. Conf. Pattern Recognition*, Alantic City, NJ, June 1990.
[56] G. Taubin, "Nonplanar curve and surface estimation in 3-space," in *Proc. IEEE Conf. on Robotics and Automation*, Apr. 1988.
[57] J. L. Turney, T. N. Mudge, and R. A. Volz, "Recognizing partially occluded parts," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 7, no. 4, pp. 410–421, July 1985.
[58] I. Weiss, "Projective invariants of shapes," in *Proc. DARPA Image Understanding Workshop*, pp. 1125–1134, Apr. 1988.
[59] A. Witkin, "Scale space filtering," in *Proc. 8th Int. Joint Conf. Artificial Intelligence*, Aug. 1983, pp. 1019–1021.

Andrea Califano (SM'90) was born in Napoli, Italy. He received the Laurea in physics from the University of Florence, Florence, Italy, in 1985. He continued his thesis research on the chaotic behavior of high-dimensional dynamical systems as a research associate at the Instituto Nazionale di Ottica in Florence, Italy.

In 1986 he spent six months as a Visiting Scientist at the Information Mechanic Group at the Massachusetts Institute of Technology, Cambridge, MA, where he was involved in research on cellular automata. From 1986 to 1991, he was a Research Staff Member in the Exploratory Computer Vision Group at the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he is currently the Manager of the Computational Biology and Pattern Matching Group. His current research interests are in the areas of computer vision, pattern matching, and their applications to molecular biology and genetics.

Rakesh Mohan (S'84–M'90) received the B.Tech degree from the Indian Institute of Technology, Kanpur, India, in 1983 and the M.S. and Ph.D. degrees from the University of Southern California, Los Angeles, in 1989, all in computer science.

He is currently a Research Staff Member at the IBM T. J. Watson Laboratory, Yorktown Heights, NY. His research interests include computer vision, robotics, and neural networks.

Dr. Mohan is the coeditor of the book *Progress in Neural Networks: neural Networks in Vision*.