MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Lab Memo 1108
C.B.I.P. Memo 29

August 1990

Indexing for Visual Recognition from a Large Model Base

Thomas M. Breuel

Abstract

This paper describes a new approach to the model base indexing stage of visual object recognition. Fast model base indexing of 3D objects is achieved by accessing a database of encoded 2D views of the objects using a fast 2D matching algorithm. The algorithm is specifically intended as a plausible solution for the problem of indexing into very large model bases that general purpose vision systems and robots will have to deal with in the future. Other properties that make the indexing algorithm attractive are that it can take advantage of most geometric and non-geometric properties of features without modification, and that it addresses the incremental model acquisition problem for 3D objects.

In particular, the paper

- introduces the notion of quantized configurations as a means of compactly encoding the spatial relationships among features in an image in a way that is useful for efficiently representing important geometric and statistical properties of images for recognition and indexing,
- presents an indexing algorithm based on quantized configurations and gives a derivation of the expected indexing speedup for the case of 2D recognition of object models consisting of unlabelled points in the presence of noise and occlusions,
- analyses the complexity of using collections of encoded 2D views for the representation of 3D objects for indexing,
- proposes and analyzes a hashing scheme that allows the per-model overhead of matching to be reduced to a simple `BIT-AND` operation,
- presents data on the empirical performance of the indexing algorithm in the 2D and 3D case,
- and considers implications of the results for psychophysics and neurobiology.

This paper describes research done within the Center for Biological Information Processing, in the Department of Brain and Cognitive Sciences, and at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. This research is sponsored by a grant from the Office of Naval Research (ONR), Cognitive and Neural Sciences Division; by the Artificial Intelligence Center of Hughes Aircraft Corporation; by the Alfred P. Sloan Foundation; by the National Science Foundation; by the Artificial Intelligence Center of Hughes Aircraft Corporation (S1-801534-2); and by the NATO Scientific Affairs Division (0403/87). Support for the A. I. Laboratory's artificial intelligence research is provided by the Advanced Research Projects Agency of the Department of Defense under Army contract DACA76-85-C-0010, and in part by ONR contract N00014-85-K-0124.

1 Introduction

Feature based visual object recognition and indexing is based on the following observation. Assume we know the shape of an object and the positions of several points that are rigidly attached to the object. If we can identify at least three such points (features) in an image, we can recover the parameters of the viewing transformation. If we can identify more than three features, the equation for recovering the viewing transformation is overconstrained, and it is very likely that the only object model that will allow an (approximately) consistent solution of the equation for the viewing transformation is the object model corresponding to the object in the image.

Visual recognition can be based on properties of an image other than feature locations and types, as well as on semantics and context; the methods described in this paper are based only on features and their geometric relations. Feature based recognition is to be understood as one of a number of separate "modules" that operate on the image in parallel and whose outputs are subsequently integrated. The task that a feature based recognition module has is to take advantage of some of the constraints that geometrical optics imposes on the 2D image of a 3D object[1].

Previous feature based approaches to visual object recognition have met with some success (without claiming completeness, some recent relevant work on the subject can be found in Shirai, 1981, Bolles and Cain, 1982, Bolles *et al.*, 1983, Grimson, 1984, Grimson and Lozano-Perez, 1985, Baird, 1985, Ayache and Faugeras, 1986, Goldberg and Lowe, 1987, Huttenlocher and Ullman, 1987, Lowe, 1987, Grimson, 1988, Cass, 1988).

However, they have several undesirable properties:

1. They assume that features are rigidly attached to the surface of an object; they assume that the viewing transformation takes a fixed, known parametric form. There are many kinds of features for which this assumption does not hold, but which are nevertheless very useful for identifying objects (see Figure 1). They require labels to be assigned to features consistently across different views. They get rather complex if objects are parameterized.

2. They assume that features on the object and features in the image can be put into one-to-one correspondence (labeled uniquely), or they require search techniques to try out many such correspondences.

3. They require 3D object models, or, at least, require that the relative 3D positions of the features on the object are known. Many feature based approaches are therefore difficult to apply when object models must be acquired automatically from images with sparse or no depth information.

[1] However, by varying the "quantization parameter" (see below), the balance between geometrical contraints and mere presence of a set of features in the image can be set arbitrarily.

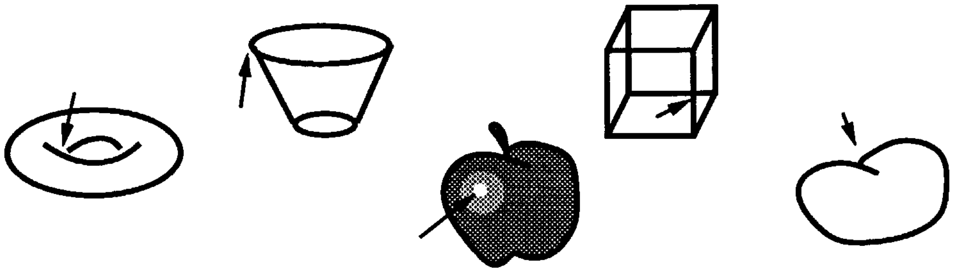


Figure 1: The spatial arrangement of a few simple features can tell us a lot about the identity of an object. But the position of features in in images is often not related in a simple way to the 3D shape of the object. Features marked by arrows in these cartoon drawings are easily recognized and give important clues about the identity of the object. Yet, they are not rigidly attached to the surface of the object and analytic expressions for predicting their location in the image are complex.

4. They often use rather arbitrary metrics to determine whether the locations of the features in the image are close enough to the locations predicted from the model and the viewing transformation. But the degree to which a particular arrangement of features suggests the presence of a particular object in the image depends on many factors, some of which are not even geometric in nature (see Figure 2 for an example; see also Besl and Jain, 1985, Turney *et al.*, 1985).

This paper describes an approach to feature based visual object recognition and indexing that addresses these problems. The paper gives a more detailed formal analysis of the method described in Breuel, 1987, and Breuel, 1989, and reports additional empirical results.

2 Motivation

The 3D recognition algorithm presented here consists of a feature-based 2D matching algorithm together with a simple way to organize and store different views of a single 3D object. Its key feature is that the cost of matching a particular model against an image is very low and can be implemented in very simple parallel hardware. Furthermore, because it uses only 2D matching operations, the algorithm does not depend on the exact analytic relationship between feature positions on the 3D object and feature locations in the 2D image; all that is required is sufficient smoothness of this map.

Any feature based recognition scheme (2D or 3D) tries to determine, for each model, whether there exists a consistent viewing transformation and a consistent assignment of

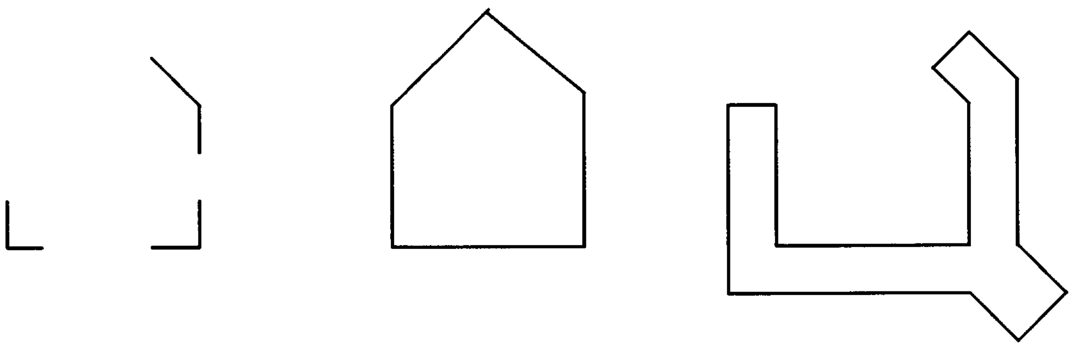


Figure 2: How well a particular arrangement of features in the image allows us to identify a particular object can depend on non-geometric factors. The features shown in (a) are very characteristic of object (b); however, if object (c) is also present in the model base, we need additional evidence to distinguish the two objects.

a subset of image features to a subset of model features, subject to some error measure. Therefore, a straightforward implementation of feature based recognition involves a search over all possible assignments of a subset of the image features to a subset of the model features, together with the explicit solution of the viewing equation. Such a naive computation can certainly be parallelized easily: each processor in a parallel machine can be assigned the task of testing one particular model, or even one particular assignment; in fact, even data level parallelism can be utilized to some degree. Nevertheless, the computation required for each object model is by no means simple.

A simple representation of the features contained in an image consists of a list of all the features (e.g., vertices, centroids of blobs, etc.) in the image, their type ("type" here means any attribute of a feature that can be used to decide that it cannot, under any spatial transformation, match another feature in the image; e.g., a 3-edge vertex can never match the centroid of a blob), and their 2D locations. To simplify the computation required to match a 2D object model against the image, we can choose a better representation, as we will see.

Ideally, we would like a representation that is invariant under renumbering features of the same type in the image, under 2D translation, rotation, and scaling (2D affine transformations), under deletion of features from the image, and under addition of spurious features to the image.

- To achieve invariance under 2D translations and scale, let us represent the spatial relations among features as angles rather than absolute or relative distances[2].

[2] An alternative, similar representation proposed independently by Lamdan and Wolfson, 1988, rep-

- To achieve invariance of the representation under small errors in localization of the 2D features, we "quantize" the angles that enter into the description of the relative locations of the features. I.e., if we have an object model that specifies certain relative angles among features in the image, we allow the angles to change by a small amount $\Delta\gamma$ before the representation changes.

- To achieve invariance under rotation, we assign a canonical orientation to each individual feature and represent angles of other features relative to the canonical orientation of this feature.

- To achieve invariance under renumbering of features with the same type, the encoding process is repeated relative to every feature in the image[3].

- To achieve invariance under deletion and addition of features from the image (for example, by occlusions or sensor noise), instead of matching all image and object features at once, selected subsets of image features are matched against subsets of object model features.

For example, an equilateral hexagon is represented by the co-occurrence of six vertices in a particular symmetric arrangement. However, an occluded equilateral hexagon might be represented by just five (or perhaps even fewer) vertices in the same spatial arrangement. A representation that would make these relations explicit would explicitly represent the geometrical relations of all subsets of features ("configurations") in the image.

A particular configuration is then evidence for the presence of some object in the image (of course, more inclusive configurations do not give statistically independent evidence). Larger configurations of features are more specific but also more sensitive to noise, and vice versa. At the extreme ends, the configuration consisting of all features in an unoccluded noiseless view of an object is the most specific one (*i.e.*, we cannot do any better for deciding presence or absence of the corresponding object in the image) but also the most sensitive to noise, whereas the configurations consisting only of one individual feature each are the least specific, often giving no information about the identity of an object whatsoever.

resents feature locations relative to a "base" formed by two points. A representation in terms of angles weighs errors in the position of features differently in a way that will be important for proving robustness of the representation under 3D rigid body motion.

[3] Another reason for repeating the encoding process relative to every feature in the image is that the 2D matching process we use is spatially non-uniform: assume that we encode two features that are located close to one another relative to a feature far from both. Small differences in the location of either far-away feature will not change the encoded angle very much, but if we repeat the encoding relative to either of the two features, these small differences will change the encoding. The motivation for using such non-uniform encodings lies in the 3D nature of the matching problem and we will analyze it further below.

Representing all configurations in an image, from those consisting of individual features to the configuration consisting of all features is clearly impractical, since the computational effort and size of the representation is exponential in the maximum number of features whose relative locations enter into the representation.

However, it is possible to set an upper bound d to the number of features that are considered together without affecting the performance of the algorithm greatly. It is true that larger configurations of features give better evidence for the presence of a particular object in the image; however, we can expect the improvement in specificity to diminish rapidly beyond a certain point. Furthermore, eventually, the algorithm will use the configurations (of various sizes) to determine whether a particular object is likely to be present in the image. In order to do so, it will have to consider the probabilities for the presence of a given object in an image conditional on the presence of a particular configuration of features in the image. Since there are many more larger configurations of features[4] than smaller configurations, and since the larger configurations specific to a particular object are more likely to be absent in training data because of occlusions and noise, the estimates of probabilities conditioned on larger configurations will be much poorer than the estimates of probabilities conditioned on smaller–more frequent–point, the improvement in specificity of larger configurations is canceled by the poor quality of the estimates of the corresponding conditional probabilities[5].

For example, in the case of the equilateral hexagon, if we are willing to consider, say, three vertices of the correct type and in the correct spatial arrangement as sufficient evidence for the presence of a partially occluded equilateral hexagon in the image, then it is sufficient to encode groups of no more three features in the representation, since any equilateral hexagon with more than three vertices present in the image will also have all the subsets present.

The explicit representation of different configurations of features also allows the representation of different degrees of "relevance" for specific configurations of features (in the case of Bayesian classification based on the conditional probabilities, this idea of relevance is taken into account automatically). In most recognition systems (as an example, consider the voting scheme of Lamdan and Wolfson, 1988), the quality of a match between model and image features is based on a simple *a-priori* quality measure that usually is a function of number of correspondences and the deviation from predicted to actual feature locations. However, different configurations of features that would be considered equally good under such quality measures can actually have vastly different relevance for determining object identity. For example, if a data base contains both the house-like

[4] This is true at least as long as the size of configurations that we are considering is smaller than half the total number of features in the image.

[5] Rather than setting a fixed upper limit d on the size of configurations, it is a straightforward extension to make the size of configurations adaptive and data dependent. In practice, a fixed, small d has worked sufficiently well, however.

and the robot-manipulator like object depicted in Figure 2, the configuration of three features shown at the left of that figure is not very useful for distinguishing between the two hypotheses. In this case, almost any configuration of size three that includes some other feature would be much better for distinguishing the two objects.

A concrete algorithm that implements these ideas is the following:

1. Input:

   a quantization parameter $\Delta\gamma$

   a "maximum configuration size" parameter d

   a list of features $1 \ldots n$ in the image. Their 2D locations are given by $\{(x_i, y_i)\}$, their 2D orientations in the image are given by $\{\gamma_i\}$, and their types are given as integers $\{t_i\}$.

2. Let S be a set of integers representing "configuration codes", initially empty.

3. Pick a feature i and a set (say, $\{j, k, l\}$) of $d-1$ other features.

4. Compute the absolute angle that features j, k, and l make with feature i (i.e., $\arctan \frac{y_j - y_i}{x_j - x_i}$) and subtract the absolute orientation of feature i. This gives the relative angles γ_{ij}, γ_{ik}, and γ_{il}.

5. Quantize the relative angles, i.e. compute

$$\hat{\gamma}_{ij} = \lfloor \frac{\gamma_{ij}}{\Delta\gamma} + 0.5 \rfloor$$

6. Combine the integers $\hat{\gamma}_{ij}$, $\hat{\gamma}_{ik}$, $\hat{\gamma}_{il}$, t_i, t_j, t_k, t_l uniquely into a single integer, the integer code f for the configuration $\{i, j, k, l\}$, and add f to the set S of configuration codes.

7. Goto step 3 until all the choices are exhausted.

A practical question is how to represent the set S in this algorithm. A representation using a hash table with collision detection would allow us to add elements to the set and test for membership almost in constant time. A simpler and more compact representation is desirable, however. A bit vector representation of the elements of the set S would be the simplest, but the range of possible configuration codes is very large. A hashed bit vector representation (i.e., a representation where each combined feature is hashed into a small range before being represented by a bit in the bit vector) is both computationally adequate and, if the vector is chosen sufficiently large, does not degrade performance significantly. While the issue of how to represent S may initially appear to be a minor one, it is very important for an efficient, practical implementation. The hashed bit vector representation can be implemented easily and efficiently on strictly data level parallel hardware (i.e., on data level parallel hardware without constant time array indexing instructions in every processor), or even in an "intelligent RAM chip" or simple "artificial neurons".

Now that we have a representation that allows us to match features between images efficiently, what is actually the matching process and how do we build a model base?

The first step for matching features using the set representation consists of determining which configuration codes are present in both the set of configuration codes for the image and for a view of the model (using the hashed bit vector representation, this reduces to a `BIT-AND` operation). Each configuration code that is present in both sets is evidence for a 2D object that is common to both images that the sets of configuration codes were derived from. How we use this evidence depends on how we have obtained the two images.

If we build a model base by computing the configuration codes for a set of noise-free views of isolated objects, all the combined features in the combined feature sets for models will belong to the object. Therefore, a simple (non-Bayesian) decision rule is the following: *An object is present in an image if the intersection of the set of configuration codes for the image with any one of the sets of configuration codes for views of that object is non-empty.*

For a Bayesian decision rule, we need to estimate the posterior probabilities that a certain object is present in the image given that a particular configuration code is present in the image. This requires keeping an $Q \times L$ table of probabilities[6], where Q is the number of known objects, and L is the length of the bit vector that represents the configuration codes. A simple Bayesian decision rule would be to give the probability that a certain object q is present in the image as the maximum of the posterior probabilities[7] $\max_{1 \leq l \leq L} \mathcal{P}(q|l)$.

To build a model base from training data, we proceed as follows. Training data consists of images and a list of objects that each image contains (in the simplest case, isolated objects in noise-free images). An algorithm for building a model base for the non-Bayesian decision rule simply records the configuration codes together with the labels. An algorithm for building a model base for the Bayesian decision rule updates the posterior probabilities for each set of labels and configuration codes. This method even works if the training examples consist of objects in context.

We still have to address the question of how to choose the quantization parameter $\Delta\gamma$. This parameter determines both sensitivity to noise in the localization of features and the degree of generalization of the matching process. A simple and effective procedure is to choose several different values for $\Delta\gamma$ and repeat the computation of the configuration codes. The non-Bayesian decision rule then prefers a match at a finer level of quantization to a match at a coarser level of quantization. The Bayesian decision rule, which must

[6] Note that this table can be much smaller than a data base of the configuration codes of every distinct view for each object.

[7] To make this decision rule useful, a classification algorithm must take into account that the posterior probabilities are only estimates.

already take into account that different posteriors are derived from different amounts of data, automatically extends to this case: in general, the posteriors of configuration codes at finer levels of quantizations will give more specific information about an object but will also have been estimated from less data.

3 Indexing Speedup for 2D Recognition

In order to gain some insights into the issues involved in indexing based on sets of configuration codes, let us consider the simpler problem of 2D indexing from a set of prototypes with error bounds. The problem and the scheme analyzed in this section is actually somewhat similar to the 3D indexing scheme proposed independently by Lamdan and Wolfson, 1988[8] However, the purpose of this section is mostly didactic; as we will see later, 3D indexing based on uniform, square 2D tilings is probably not a good idea.

Assume that we are given a set of N 2D models. Each 2D model is a list of r points. An image consists of n points from one of the models in the data base, plus m points that are randomly placed (or $n+m$ randomly placed points if no points from the model are present in the image). The 2D recognition algorithm has to determine a subset of r points in the image that can be translated and rotated[9] so that each of the r points is within a given distance ϵ from a point in the matching model. We assume that there is a 2D matching algorithm MATCH that takes as input a model, a subset of d image points, and (perhaps, optionally) a set of correspondences, and can check whether these correspondences can be extended to r correspondences that satisfy the error bounds. The problem of indexing is now to reduce the number of calls to the algorithm MATCH in order to achieve an overall speedup of recognition[10].

Imagine a tiling of the plane with squares of area δ^2, where $\delta > 2\epsilon$ (see Figure 3). Now number the squares by moving out in a spiral from the origin. In this way, each point in the plane is assigned an integer $q_\delta(x, y)$. Assume that we are given a set of points $S = \{(x_i, y_i) : i = 1 \ldots d\}$. If we only know that the position of each point in S is within a distance ϵ of its true position, the set S together with the bound ϵ actually represents an infinite number of configurations $C = \{\{(x_i + \Delta x_i, y_i + \Delta y_i) : i = 1 \ldots d\} : {\Delta x_i}^2 + {\Delta y_i}^2 < \epsilon^2\}$. To each configuration, we can assign a unique integer by computing the function q_δ for each point within the configuration and combining the resulting numbers in some suitable way (in practice, it is most convenient to use a hash function; for the following theoretical considerations, assume that the combinations are numbered uniquely). If

[8] Lamdan and Wolfson, 1988, do not analyze questions of noise or stability, which are central to the performance of the indexing algorithm.

[9] Allowing for scaling does not change the algorithm or result significantly but complicates the analysis.

[10] Alt *et al.*, 1988 have given an algorithm to solve this problem (arbitrary isometric mapping, unlabeled points) that runs in $O(n^8)$

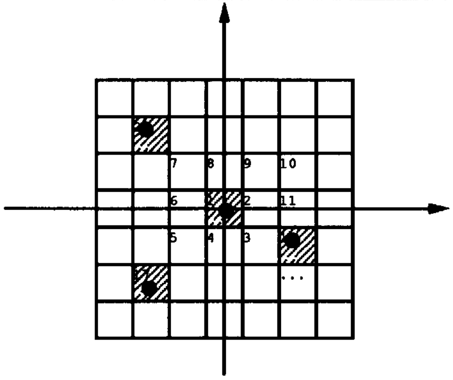


Figure 3: Encoding configurations using a tiling of the plane with square tiles. Tiles are numbered, and the approximate position of each point in a configuration is given by the number of the tile that it falls on. By combining tile numbers suitably, we can assign an integer code to every configuration of points.

$\delta > 2\epsilon$, it is easily seen that the set of integers C_q corresponding to the infinite number of configurations in C is finite, and, in fact, at most of size 4^d. If we have two sets of configurations of points C_1 and C_2, then showing that $C_{q1} \cap C_{q2}$ is empty shows that $C_1 \cap C_2$ must be empty as well.

If we choose δ sufficiently large and assume that points are picked randomly with respect to the grid of size δ, the expected size of C_q will actually be much smaller than 4^r. To analyze this, assume that we pick r points at random in the plane [11]. The total number of integers in C_q is the product of the number of tiles of area δ^2 that the ϵ balls around each point touch, $R = \#C_q = \prod_{i=1}^d T_i$. We would like to compute the expected value $\mathcal{E}(R)$. Since the points are picked independently (by assumption), $\mathcal{E}(R) = \mathcal{E}(\prod_{i=1}^d T_i) = \prod_{i=1}^d \mathcal{E}(T_i)$. Now, T_i is easy to compute: it is 4 with probability $\frac{\pi\epsilon^2}{\delta^2}$ (i.e., whenever the point is closer than ϵ to one of the corners of the tile), it is 2 with probability $\frac{4\delta\epsilon - 4\epsilon^2 - \pi\epsilon^2}{\delta^2}$, and 1 with probability $\frac{(\delta-2\epsilon)^2}{\delta^2}$ (see Figure 5 to see how these probabilities come about). After a little manipulation, we find that $\mathcal{E}(T_i) = (2\pi - 4)(\frac{\epsilon}{\delta})^2 + 4\frac{\epsilon}{\delta} + 1$. How large we make the ratio $\frac{\epsilon}{\delta}$ depends on several factors like data base size, cost of indexing vs. matching, etc. For example, if we pick $\delta = 5\epsilon$, the $\mathcal{E}(R) \approx 1.89^d$, which is a significant improvement over the strict bound $R \leq 4^d$ (d will turn out to be about 5 in practice).

We now have a quick means of testing whether a given set of points can be transformed into another set of points by moving each point by a distance of less than ϵ. But we would also like to have a quick test for the non-existence of translations and rotations that could

[11] Assume that the density of points is low on a a scale of δ so that the case that two points land in the same tile is rare.

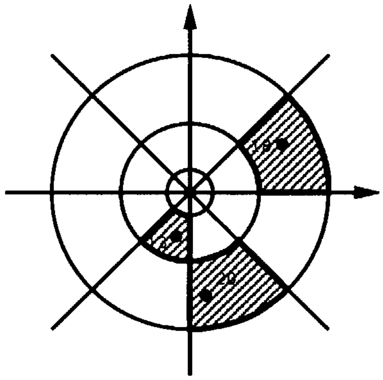


Figure 4: A tiling of the plane in terms of polar coordinates (or, equivalently, quantizing relative angle and distance) is more suitable for the recognition of 3D objects from 2D views.

transform one set of points into the other. If we pick one point of a set of points, where the position of each point is uncertain by an amount of ϵ, and move this point to the origin, we can establish the non-existence of a translation plus match within an ϵ error bound by establishing that the same translation of another set of points does not match the first set within a bound of 2ϵ. Likewise, to handle rotations, we can pick the two points with a largest distance[12], move one of them to the origin and rotate the other onto the x axis; non-existence of a match within 3ϵ establishes the non-existence of a translation, rotation, and match within an error of size ϵ at each point.

Naively, we can use these ideas for indexing as follows. Consider a set M of model points. For each two points $a, b \in M$, $a \neq b$, move a to the origin and rotate b onto the x axis; then, encode the resulting set of points with a $\delta > 6\epsilon$, giving rise to a corresponding set of integer codes of configurations M_q. If we want to prove quickly that a particular set S of points cannot, under arbitrary translation and rotation and within an error of ϵ at each point, match the model points, it suffices to pick the two points in S that have the largest distance, move one of them to the origin, rotate the other onto the x axis, compute the resulting configuration code c, and test whether $c \in M_q$. If $c \notin M_q$, then it is established that S cannot possibly match M; otherwise, another test is necessary.

The above scheme is not very practical for realistic recognition, however, since real images have both spurious and missing features. To get around this problem, instead of encoding all points in M, we encode, both in the image and in the models, only subsets of d points, where d is some small, fixed number. To encode n points, for example, we will,

[12] Within the error bounds, there may be several such pairs of points.

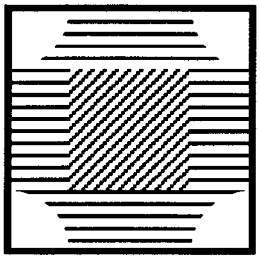


Figure 5: The white quarter circles at the corner of the tile have a radius of ϵ, and the sides of the tile are each δ long. The central area therefore has size $(\delta - \epsilon)^2$, the area covered by the four white areas in the corners is $\pi\epsilon^2$, and the region in between is what remains of the total area of δ^2.

on average, generate a total of $R^d n^{(2)} \binom{n}{d-2}$ configurations[13], which is $O(Rn^d)$. If we find no common configuration code of size d between an image and a model, then we know for certain that the model cannot occur in the image, under any translation or rotation.

In practice, we would organize 2D recognition based on the above ideas as follows. For each model, we compute the complete set of configuration codes (or generate it by stochastic sampling). We associate with each configuration code a list of the 2D models that give rise to that configuration code. As reasoned above, a data base of size N will contain $O(NR^d r^d)$ different configuration codes. An image consisting of n ($n < r$) model derived features and m spurious features will give rise to $O(R^d(n+m)^d)$ configuration codes. The indexing algorithm then only has to check the models that correspond to configuration codes that actually come from the image–that is, at most, the models corresponding to the $O(R^d(n+m)^d)$ configuration codes.

This indexing scheme only works well when the set of configuration codes corresponding to models makes up a small fraction the set of possible configuration codes, and, furthermore, the configuration codes for different models usually differ. Intuitively, we are looking at the set of model points at a coarser scale (a scale of δ) and use this coarse matching to pre-select models for a final matching step. In this, the algorithm is similar to multi-resolution matching approaches.

For 2D recognition of realistic shapes, the assumption that we can make many useful

[13] The notation $n^{(k)}$ means $\frac{n!}{(n-k)!}$. The factor $n^{(2)}$ arises from the first two points which we pick for the translation and rotation, the factor $\binom{n}{d-2}$ comes from the encoding of the remaining points, which can be invariant under permutations, and the factor R comes, as above, from the possible overlap of a single ϵ ball with several δ tiles.

distinctions at a coarser level is certainly not unreasonable. If, for the sake of argument, we assume that both models and noise consist of random, independently picked points, and that they can occur in an area of size $A = K\delta^2$ (i.e., that there roughly K different tiles), then we expect the number of models corresponding to each configuration code (for some constant C)[14]

$$t \leq C \frac{NR^d r^d}{K^{d-\frac{3}{2}}} \tag{1}$$

In this equation, the numerator is simply the total number of configurations in the data base, and the denominator is the total number of possible configuration codes.

We would certainly like t to be smaller (much smaller) than 1. We can achieve this by choosing ϵ, δ, and d such that the following inequality is satisfied (to simplify the derivation, we have replaced $d - \frac{3}{2}$ by d):

$$C\sqrt{(\frac{t}{N})^{\frac{1}{d}} \frac{A}{Rr}} \gg \delta \geq 2\epsilon \tag{2}$$

($\sqrt{A}$ here is the "scale" of the 2D models.) If these inequalities are satisfied, we expect the above indexing scheme to work well and require about $O(R^d(n+m)^d)$ calls to the algorithm MATCH for verification.

By choosing d large enough, we can always, in principle, counteract the effect of a large model base (i.e., large N). However, if the fraction $\frac{A}{Rr}$ is large, then for a given ϵ it may not be possible to satisfy the inequality. This is not surprising: if all models are very similar (in the extreme case, identical), meaning that either most of them or almost none of them actually match any given image within the given error bound ϵ, we cannot hope to achieve a large indexing speedup. In fact, in that case, the very concept of "indexing" is meaningless, since indexing is only a useful thing to do if we expect that for any given input image only a few images in the model base will match. If the models are distinct, we can, however, achieve this by choosing ϵ sufficiently small (for a closely related discussion on random patterns and the probability of matching, see Baird, 1985).

If we satisfy the inequality, and if ϵ and δ constant while varying the model base size N, this gives us an expected constant speedup proportional to $K^{d-\frac{3}{2}}$. But, for the same reasons discussed in the previous paragraph, as we increase the size of the model base, we must eventually also increase the resolution at which matching takes place, since at any given value of ϵ there is only a finite number of "dissimilar" objects[15]. Using this reasoning, we expect the indexing speedup not to remain constant, but improve, as we increase the model base size N and decrease ϵ at the same time.

[14] The exponent of $d - \frac{3}{2}$ takes into account the fact that configurations related by translation and rotation are the same.

[15] This notion can be made precise.

The point of the above exercise was to allow a comparison of the proposed 2D indexing scheme with similar existing approaches, to illustrate the source of the speedup and the limitations of the scheme. While convenient for theoretical analysis, the model of 2D recognition used in this section (matching within given error bounds) is not very practical for several reasons.

The requirement that a matching algorithm determines exactly whether a given set of points match within certain error bounds is overly restrictive because the choice of error bounds is rather arbitrary for many applications. By insisting on precise ϵ-bound matching, a lot of computational effort is expended without much gain in classification performance.

Likewise, a "correct" implementation of the above algorithm requires that all possible configurations of d points are stored in the data base and have equal significance for indicating the possibility of a match between object and model. However, in practice, as described in previous sections, we can estimate probabilities with which a given configuration of features in the image indicates the presence of a given object; if this probability is small enough (compared to the acceptable error rate), the presence of the corresponding object in the image never needs to be verified.

Finally, as analyzed above, the algorithm does not get any "labels" or feature types associated with the input points. However, when each input feature is labeled with one of k labels, the number of configurations of size d (and thereby the "specificity" of a particular configuration) increases by a factor of k^d. This means that much smaller d are sufficient to achieve good indexing performance.

4 3D Recognition based on 2D Views

We are primarily interested in the recognition of 3D objects from 2D views. In Section 2 it was suggested that 3D recognition might be possible by storing encoded collections of 2D views of a particular object. An important question is how many discrete views of each 2D object must be stored in order to have a sufficiently complete representation of its 3D shape.

We can derive a theoretical bound on this number under some standard assumptions of feature based object recognition: (approximately) orthogonal projection and features that are scale independent and attached rigidly to the surface of an object. To make the presentation more concrete, we will talk specifically about 2D images of vertices of 3D wire frame objects; the same analysis carries over to any angle-based representation of 2D images of 3D objects.

The encoding we will use is similar to the one used in Section 3 but uses a polar tiling instead of a square tiling (see Figure 4). A polar tiling is more suitable for 3D recognition from 2D images because the change in the relative location of two features in a 2D image

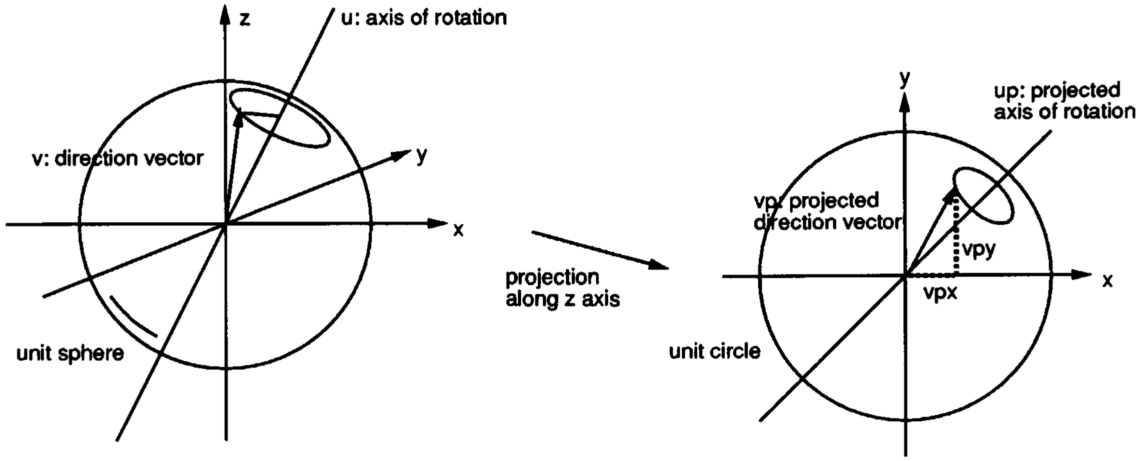


Figure 6: Deriving a bound on the rate of change of a projected angle: consider a 3D unit vector v. Under rotation around an axis u, its tip will describe a circle in 3-space. The projected vector v^p will lie on an ellipse parameterized by the angle of rotation in 3-space, and the angle it makes with the x axis is given by $\arctan \frac{v^p_y}{v^p_x}$.

caused by 3D rotations of a 3D object increases with their relative distance so that the tiles should become larger in the radial direction around a feature. We simplify the analysis by restricting it to polar tilings that make no distinctions in the radial direction.

To obtain a lower bound on the size of a patch on the viewing sphere that is "covered" by a particular view of a vertex, it suffices to determine an upper bound on how fast the projection of a feature changes as the object is rotated; by "covered" I mean that at a given level of quantization, changes of the viewing parameters that fall within the patch on the viewing sphere will give rise to the same encoded feature.

For concreteness, consider a vertex in 3-space[16] It is made up of a collection of half-lines. Let the directions and orientations of these half-lines be given by unit vectors. As already described in Section 2, we consider only the relative angles between these unit vectors.

Assume now that we are looking at one particular projection of the vertex. If any of the relative angles of the projected edges entering the vertex changes by more than the quantization angle $\Delta\gamma$, the encoding of the vertex will change. Of course, each of the $\frac{n(n-1)}{2}$ relative angles of the n projected edges[17] can only assume a value between 0 and

[16] In general, we are given a set of points in 3-space; if we pick one distinguished point, this is equivalent to a vertex.

[17] To avoid the complications that arise from the changes in the ordering of the edges around the vertex as the viewing parameters change, let us consider all pairs of angles here rather than just pairs of adjacent

2π, so there are at most $\frac{n(n-1)\pi}{\Delta\gamma}$ representations of this vertex possible. However, it is essential both for the discriminability of different vertices and the feasibility of storing representations of a single vertex for all general viewing positions that only a small fraction of these representations are actually used (we have calculated this fraction in Section 3 for the case of points distributed randomly in the plane).

We can prove this by deriving a bound on the rate of change of the relative angles of the projected edges under rotation around an arbitrary axis of rotation u. The rate of change of the relative angles of the vectors in projection is bounded by twice the rate of change of each individual vector with the x axis.

Let v be one of the unit vectors making up the vertex under consideration (see Figure 6). Under rotation around an axis u by an angle θ, its tip describes a circle in 3-space. In the image, the tip of its orthogonal projection v^p lies on an ellipse parameterized by θ. Without loss of generality, assume that the projection u^p of the axis of rotation u coincides with the y axis. The 2-vector v^p can then be written as:

$$\begin{pmatrix} v_x^p \\ v_y^p \end{pmatrix} = \begin{pmatrix} x_0 + a\,\cos(\theta+\theta_0) \\ y_0 + b\,\sin(\theta+\theta_0) \end{pmatrix} \tag{3}$$

Since the ellipse must lie within the unit circle, the parameters in this equation are bounded by:

$$|x_0|, |y_0|, |a|, |b| \leq 1 \tag{4}$$

The angle that v^p makes with the x axis is simply:

$$\gamma = \arctan \frac{v_y^p}{v_x^p} \tag{5}$$

Therefore, the derivative of γ with respect to θ is:

$$\gamma_\theta = \frac{1}{1+(\frac{v_y^p}{v_x^p})^2} \frac{v_{y\theta}^p v_x^p + v_y^p v_{x\theta}^p}{v_x^{p2}} = \frac{v_{y\theta}^p v_x^p + v_y^p v_{x\theta}^p}{v_x^{p2} + v_y^{p2}} \tag{6}$$

The numerator of this equation is easily seen to be bounded above by 2. The denominator is the square of the length of v^p. But v^p is not bounded away from zero: v^p can be zero when v is aligned with the z axis. This means that for small patches of the viewing sphere, namely those when the direction in which the observer is looking is nearly aligned with one of the edges in the object, the encoding of the feature could change rapidly. This corresponds to the notion of "non-general viewpoints", which are often excluded in recognition systems.

Let ϕ be the angle that v makes with the z axis (i.e., the direction along which we project). The length of v^p, the projection of v, is then $\sin\phi = \sqrt{v_x^{p2} + v_y^{p2}}$. To bound γ_θ

angles (as we did in the preceding section) .

from above, we must bound $\sin\phi$ from below. It is easy to derive that the fraction of the viewing sphere that we are thereby excluding is $f = 1 - \cos\phi$ for each edge. Outside these patches, the derivative of the projected angle is then bounded as

$$\gamma_\theta < \frac{2}{\sin^2\phi} \tag{7}$$

(note that for small ϕ, $\frac{\sin^2\phi}{2}$ is approximately equal to $f = 1 - \cos\phi$).

We are interested in seeing how much θ may change (for an arbitrary axis of rotation u) before the projected angle γ changes by more than $\Delta\gamma$. That is, $\Delta\theta < \frac{\Delta\gamma}{\gamma_\theta}$, or

$$\Delta\theta < \frac{\Delta\gamma \sin^2\phi}{2} \tag{8}$$

Therefore, if we record a view of an object in the data base for a particular set of viewing parameters, any view that is derived by a rotation about an arbitrary axis by an angle of less than $\Delta\theta$ in Equation 8 will have the same encoding, since none of the projected angles has changed by more than the quantization parameter. In different words, each encoding of an object with angular quantization covers at least a circular patch of area $\Delta A = 2\pi(1 - \cos\Delta\theta)$ on the viewing sphere. We can derive a bound on this as follows:

$$\Delta A = 2\pi(1 - \cos\Delta\theta) \leq 2\pi\Delta\theta \leq \pi\Delta\gamma \sin^2\phi \tag{9}$$

The total number N of circular patches required to cover the viewing sphere, if we could choose their placement, is then bounded (including a factor of 2 to account for that fact that we cannot cover the viewing sphere without overlap using circular patches):

$$N \leq \frac{4}{1 - \cos\Delta\theta} \leq \frac{4}{1 - \cos\frac{\Delta\gamma \sin^2\phi}{2}} \tag{10}$$

For small $\Delta\gamma$ and small ϕ, we can approximate the last expression and re-write it in terms of f, the fraction of the viewing sphere not covered:

$$N \leq \frac{\text{const}}{(\Delta\gamma)^2 f^2}, \tag{11}$$

If, instead of excluding a fraction f of the viewing sphere, we insist on covering the whole viewing sphere but assume that the patches of non-general viewpoint are non-overlapping, we obtain an additional linear dependence on the number of edges, one-fold overlap results in a quadratic dependence, *etc.* Of course, if the representative quantized views are pre-determined, the total number of views will depend on the number of vertices in analogy to the dependence R^d on configuration size d in the case of the quantization by tiling in Section 3.

In a realistic vision system, we do not get to choose the representative views that represent an object. Rather, we have to cover the viewing sphere stochastically. If we assume that views are uniformly chosen at random, we can apply a variety of results from stochastic geometry. It can be shown that the probability of complete coverage after n trials satisfies as $n \to \infty$ (see Hall, 1988):

$$p_n = 1 - (1 + o(1))n^2\rho(1-\rho)^{n-1}$$

Here, ρ is the fraction of the surface covered by one patch, i.e., $\rho = \frac{\Delta A}{4\pi}$. Note that p_n goes to zero exponentially.

Easier to calculate is the probability that a particular view is not covered after n trials, or, equivalently, the expected fraction of the viewing sphere that is not covered after n trials. This is simply:

$$p_n = (1-\rho)^n \approx e^{-\rho n}$$

5 Robustness to Noise of the Hashing Step

We cannot rely on the early vision modules to detect all features in the image reliably or not to add some spurious features to the image. Nor can we expect that grouping and saliency mechanisms will ensure that only features belonging to one object are used as input for our model base indexing algorithm. However, we might like to obtain some theoretical worst-case bounds.

As we have already discussed, there are two kinds of possible errors: collections of features originating not from a single object could accidentally be arranged like some configuration of features from an actual object; or, different feature configurations could be hashed to the same bit in the hashed bit vector.

The first problem is geometrical in nature and intrinsic to feature based recognition (and we have analyzed it in Section 3): if a number of features in the image accidentally are arranged like the features corresponding to an object in the data base that is not actually present in the image, any purely feature based scheme cannot tell the difference between this accidental arrangement of features and the actual presence of the object in the image. Of course, the situation can be improved by increasing the number of features on which the match between model and image is based, and by increasing the accuracy required for declaring a match. Resource limitations and the presence of noise in measurements limit the degree to which we can use these approaches, though. If we are content with using the algorithm for indexing, false matches will most likely be discovered in the verification step.

The second problem is a result of the use of hashing. Hashing and representation of the object features as bit vectors gives us a compact and efficient representation, and it allows us to choose different levels of quantization without increasing the size of the representation. However, the use of hashing introduces another possibility of error: two distinct configuration codes could be hashed to the same bit in the combined feature vector. The probability of this event can be reduced by increasing the size of the combined feature vector. How large we have to make the bit vector in order to keep the probability of misidentifying an object low is the subject of this section.

Let

- N be the number of models in the data base. We assume that the feature vectors corresponding to the models are statistically independent.
- L be the number of bits in a feature vector.
- r be the number of features in an unoccluded view of an object without any spurious features, i.e. a view corresponding to a model feature vector in the data base (we assume for the sake of simplicity that r is the same for all model feature vectors in the data base).
- n be the number of object-derived features present in the image to be classified (i.e. $r - n$ features are absent, perhaps because they are occluded or because the image is noisy).
- m be the number of spurious features in the image to be classified. We assume that the location and type of the spurious features is independent of the object present in the image.

Recall that each feature results from combining a number, say d, simple features based on their geometric relationship, and quantizing the result. If we have c features in an image (or in a "group" after some preprocessing using grouping and saliency mechanisms), they will give rise to c^d configuration codes, and each configuration code correspond to a bit in the feature vector after hashing[18].

Under the above assumptions, each model feature vector contains r^d configuration codes, whereas the feature vector for a noisy test image contains n^d features derived from the object and $(n+m)^d - n^d$ spurious features.

To ensure that the correct object is identified from the data base, the feature vector in the data base that corresponds to the correct model must have more matches against the feature vector derived from the test image than any of the feature vectors corresponding to incorrect models.

[18] To simplify the analysis, assume that each "bit" actually represents a count of the features that hashed into this bucket; the actual implementation described earlier used binary values for efficiency of representation. If the feature vectors are relatively sparse, there is little difference between the two models.

The correct feature vector will have n^d certain matches, representing the features in the image that were left after noise and occlusions. There are then $r^d - n^d$ bits left in the model feature vector to match the remaining $(n+m)^d - n^d$ bits in the image feature vector. The expected number of matches, call it A, between the model feature vector and image feature vector is then:

$$A = n^d + \frac{((n+m)^d - n^d)(r^d - n^d)}{L} \tag{12}$$

The expected number of matches A is certainly bounded from below by n^d, and we will use this bound below for the sake of simplicity.

For the incorrect feature vectors in the data base, we will simply try to match r^d bits randomly against the $(n+m)^d$ bits in the image feature vector, and the expected number of matches is:

$$B_i = \frac{(n+m)^d r^d}{L} \tag{13}$$

The probability that any of the B_i is greater than A is bounded from above by the Markov inequality (and using the fact that $A > n^d$):

$$\mathcal{P}(B_i > A) < \frac{\mathcal{E}B_i}{n^d} = \frac{(n+m)^d r^d}{Ln^d} \tag{14}$$

Using the Poisson approximation to the binomial distribution, we can determine the probability ϵ that any of the B_i is greater than A to be no greater than:

$$\epsilon = \mathcal{P}(\exists i : B_i > A) < (N-1)\mathcal{P}(B_i > A) = (N-1)\frac{(n+m)^d r^d}{Ln^d} \tag{15}$$

Here, N is the number of models in the data base. This means that by increasing L, we can limit the probability of error ϵ. In particular, we can derive the following asymptotic relation between a chosen probability of error ϵ_0 and the required corresponding size of the feature vector L_0:

$$L_0 = \mathcal{O}(\frac{(n+m)^d r^d}{n^d \epsilon_0}(N-1)) \tag{16}$$

This means that the size of the bit vector grows at most linearly in the number of models in the data base[19], as the inverse of the probability of error, and exponentially in the number of features that enter into the feature combination. Of course, if we allow the indexing scheme to return a small number m of candidate models, then the size of L is proportional to e^{-m}.

models	10 or 12 points picked at random within the unit circle
number of models in data base	50-200
number of possible labels per point	0 or 16
number of model points present in image	7
number of noise points present in image	7
rotations	arbitrary
translations	+/- 0.5
noise vector added to each point	+/- 0.01 (1%)
quantization (δ)	square tiles of size 0.03 (3%)
size of configurations	4

Table 1: The most important parameters used in the 2D recognition experiments.

6 Recognition of 2D Patterns

To test the performance of the the 2D indexing algorithm, some numerical experiments were conducted. These experiments used small numbers of unlabeled points in the plane for images and object models, and added noise, spurious, and missing features.

The experiments were meant to test the robustness of the algorithm to moderate amounts of noise, and additions and deletions of moderate numbers of features. For complex real-life scenes with large numbers of features belonging to many different objects, some kind of feature selection or grouping is necessary before the indexing algorithm as described here can be applied. However, since such algorithms are not perfect, robustness to spurious and missing features is still an important issue.

The most important parameters used in the experiment are given in Table 1. The encoding and matching was carried out as described in Section 3. However, the data base was built ("learned") by stochastically sampling the space of image transformations and noise vectors, rather than by generating all possible quantized configurations directly.

Figure 7 shows the performance of the 2D indexing algorithm under the strict interpretation used in the analysis in Section 3; this means that every model corresponding to any quantized configuration occurring in the image is verified.

Figure 8 shows the performance of the 2D indexing algorithm under a combinatorial interpretation. The "best match" here is the one with the largest number of corresponding configurations in the image. With the uniform tiling used in the experiment, this is equivalent to the voting scheme suggested by Lamdan and Wolfson, 1988.

The possible matches returned by the indexing algorithm were not actually verified

[19] This bound can be tightened.

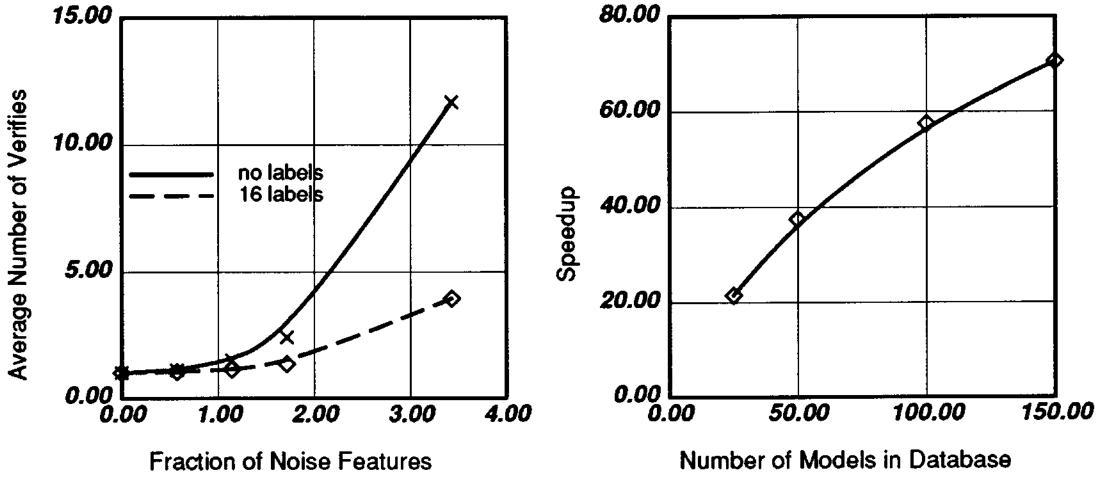


Figure 7: Empirical speedup for the *strict* version of the indexing algorithm. The graph on the left shows the average number of calls to the verification algorithm MATCH for different ratios of the number of noise feature to the number of model features in the image.
The graph on the right shows that for larger model bases, the achievable speedup (where the speedup is defined as the ratio of model base size to the average number of verifies for that model base size) is significantly greater. The parameters for these experiments were the same as for the other graph; the ratio of noise features to model features was fixed at 1. In both cases, the first choice returned by the indexing algorithm was the correct choice with probability 99.6% in the worst case.

in any of the experiments. Thus, a significant fraction of the candidate models returned by the indexing algorithm may represent true matches, and the "speedup" shown in the graphs is therefore only a conservative estimate of the true speedup.

For clarity, the number of noise features in the images used in the experiments is given as a fraction. This is not intended to suggest that the indexing algorithm actually depends only on this ratio. In fact, using a simple statistical argument, we can see that the performance of the best-match version of the algorithm improves at a constant ratio of model to noise features in the image as we include more and more total features into the match.

These results suggest that the 2D indexing algorithm can achieve significant speedups even in the presence of noise features and with relatively large model bases. The algorithm was implemented in "C" on a SparcStation 1, and the encoding and matching of each image takes less than a second.

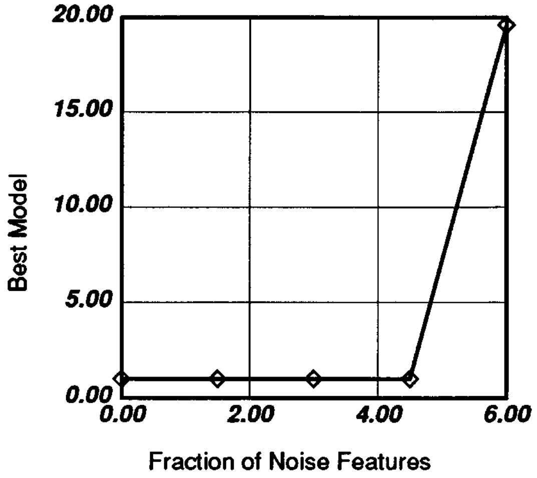


Figure 8: Empirical speedup for the best-model version of the indexing algorithm. The graph shows the average number of calls to the verification algorithm MATCH for different ratios of the number of noise feature to the number of model features in the image. The data base for this experiment consisted of 200 models, and 8 model points were present in each image.

7 Recognition of 3D Wire Frame Polyhedra

In the non-Bayesian decision rule, a separate combined feature vector is stored for each view of an object. In the Bayesian scheme, the size L of the combined feature vector that is required and the quality of the estimate of the posterior probabilities still depend on the number of models in the data base, i.e., the number of different views that a 3D object has, in the 3D recognition case. One might fear that the number of training examples that needs to be obtained (the sample complexity) and the number of views that needs to be stored per object is impractically large. The following numerical experiments address this question.

There are two different basic approaches to determine the limits of the approach of 3D recognition by 2D methods. Either we could fix computational and storage resources and look at error rate, or we could fix an error rate and determine what resources are needed to achieve this error rate. In the experiments, the second approach was used: the resolution of the matching step was chosen such that recognition was perfect (whenever possible; certain views of some pairs of 3D objects cannot be distinguished at all from their 2D projections–see comments below), and the number of different views that had to be stored for various different objects was determined (see Figure 9).

More specifically, orthogonal projections of wire frame polyhedra were generated at

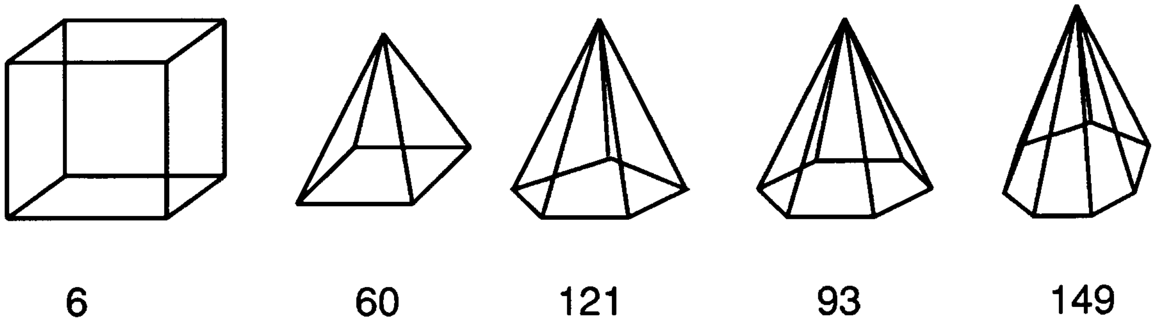


Figure 9: The number of distinct combined feature vectors (distinct "views") for the given objects. These numbers were obtained by random sampling of the viewing sphere. The degree of quantization used is sufficient to allow polyhedral objects of this kind to be distinguished with certainty. The size of the combined feature vector (the hash function) was chosen such that less than 10% of the bits in the combined feature vectors were set.

random orientations. For each projection, the position of the vertices were determined (excluding "X" junctions). For each vertex, the relative angles of the edges entering the vertex were computed (in counterclockwise order around the vertex). The list of relative angles was than permuted circularly so that it always began with the largest relative angle. The angles were quantized (truncated) using a quantization parameter $\Delta\gamma$. The list of quantized angles represents the "type" of this vertex; by interpreting the elements of the list as digits in a suitable positional number system, this type can easily be converted into a number. The canonical orientation of each vertex is defined as the orientation, relative to the y axis in the plane, of the edge entering the vertex before the largest angle. For each ordered pair of vertices, the feature type of the pair is given as the tuple consisting of the quantized orientation of the second vertex relative to the first one, the quantized orientation of the line connecting the two vertices relative to the orientation of the first vertex, and the types of the individual vertices. As before, we can convert this tuple into a number. We then apply a hash function to this value to obtain the hashed feature type ("combined feature") for this particular vertex pair. For any given image, the set of hashed feature types of all pairs of vertices in the image is represented by a bit vector, the "combined feature vector". The only adjustable parameters in this algorithm are the hash function, the size of the combined feature vector, and the various quantization parameters.

For each wire-frame polyhedron, random projections were generated until a large number of trials (> 100) did not yield any new combined feature vectors. The number of distinct combined feature vectors found in this way for each polyhedral object is an estimate of the number of different views the object has under the above encoding at the chosen level of quantization. The level of quantization was determined such that it was sufficient to distinguish objects of different types reliably. To test this, the similarity under Hamming distance was determined between all views of all pairs of objects. Many pairs of objects could be reliably distinguished for quite coarse quantizations. However, certain views of different objects cannot be distinguished at all on the basis of their orthogonal projections, and, therefore, the level of quantization for some objects had to be determined on a case-by-case basis. The algorithm was also tested (informally) under slight distortions of the object, addition of spurious features, or partial occlusions of objects.

Counts of the number of distinct views determined in this manner for a collection of simple wire-frame objects are shown in Figure 9. These are worst-case numbers, since two combined feature vectors were considered distinct even if they differed in only one bit. For accurate recognition, it is sufficient that the collection of views stored in the data base classify all new images correctly, i.e. that the combined feature vector for any new view of an object differs less from one of the corresponding training examples than from any other combined feature vector in the data base.

The problem of recognizing 3D wire frame polyhedra is probably not a very difficult one when compared to the recognition of more realistic 3D objects. However, the indexing algorithm presented earlier in this paper works at least as well in this domain as other algorithms. Its advantages are that it is very fast, significantly simpler to implement, and requires no domain-specific knowledge. The task of 3D recognition is harder than 2D recognition as described in Section 6 because each object corresponds to several views. However, in the case of polyhedral recognition (and probably recognition of real-life 3D objects as well), individual features in the image are labeled very specifically by vertex degree and the relative arrangement of edges around the vertex; this makes the task of recognizing an individual 2D view of such an object much easier than the task of recognizing unlabeled or weakly labelled groups of points as described in Section 6.

8 A "Neural Network"

The computational requirements for the indexing algorithm that we have analyzed in the previous sections are relatively simple, and it is interesting to speculate how it could be realized in neural hardware. A block diagram of a "neural network" implementation of the algorithm is shown if Figure 10. It consists of two main steps: an *encoding* step that computes the feature configurations occurring in the image, and an *indexing* step that determines which objects are likely to be present in the image based on the set of configurations.

The input to the encoding step consists of a feature map that has been computed during early visual processing. A way in which the encoding of feature configurations could take place is based on shifter circuits. In this model, shifter circuits would translate (and,

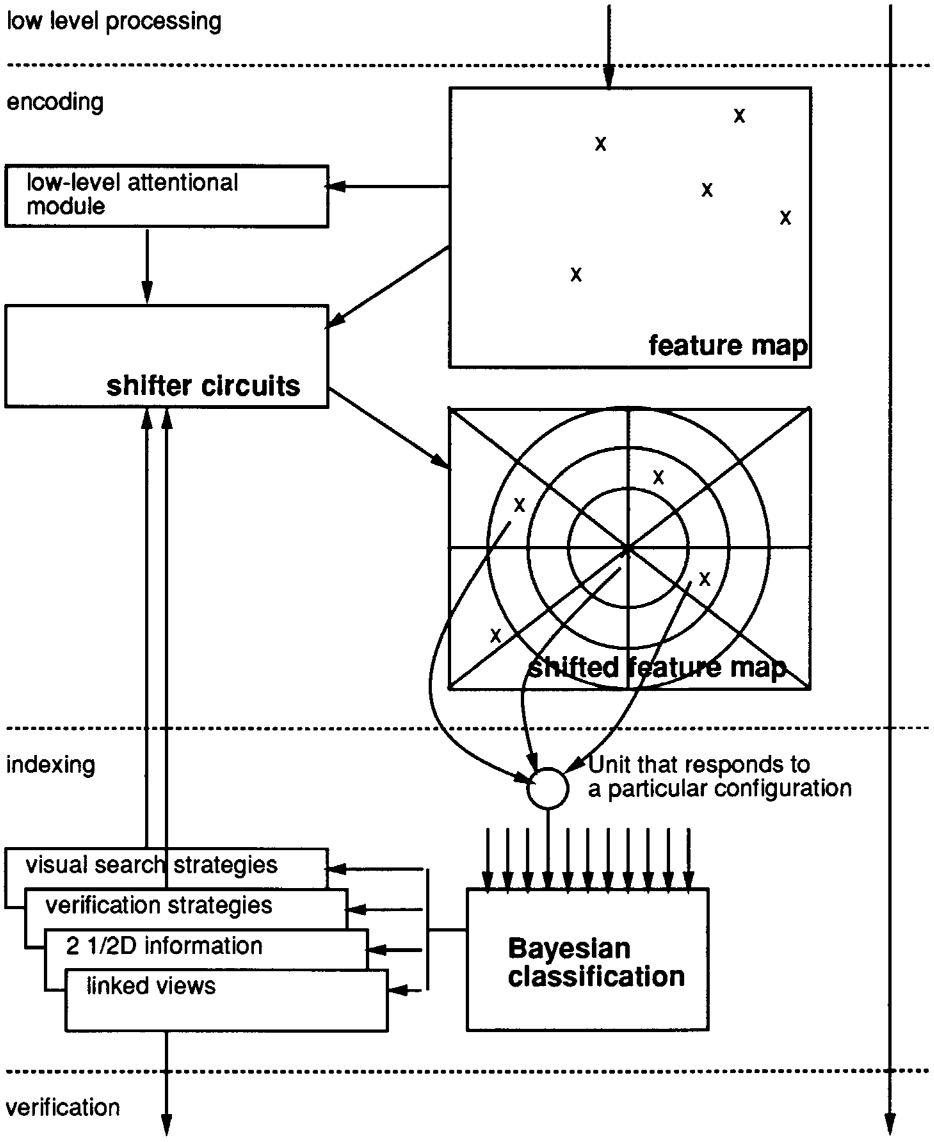


Figure 10: This block diagram illustrates how the proposed algorithm could be realized in "neural network" hardware. The computation of different configurations is carried out using parallel shifter circuits and units that respond to the simultaneous presence of different features in different locations in the shifted feature map. The shifter circuit are controlled either by a low-level attentional module (for example, texton based), or by higher level strategies that are invoked once partial information is available about the image.

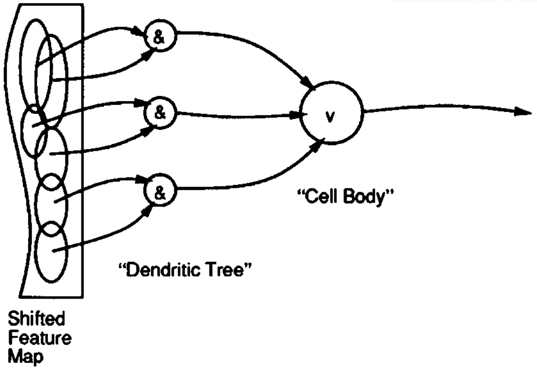
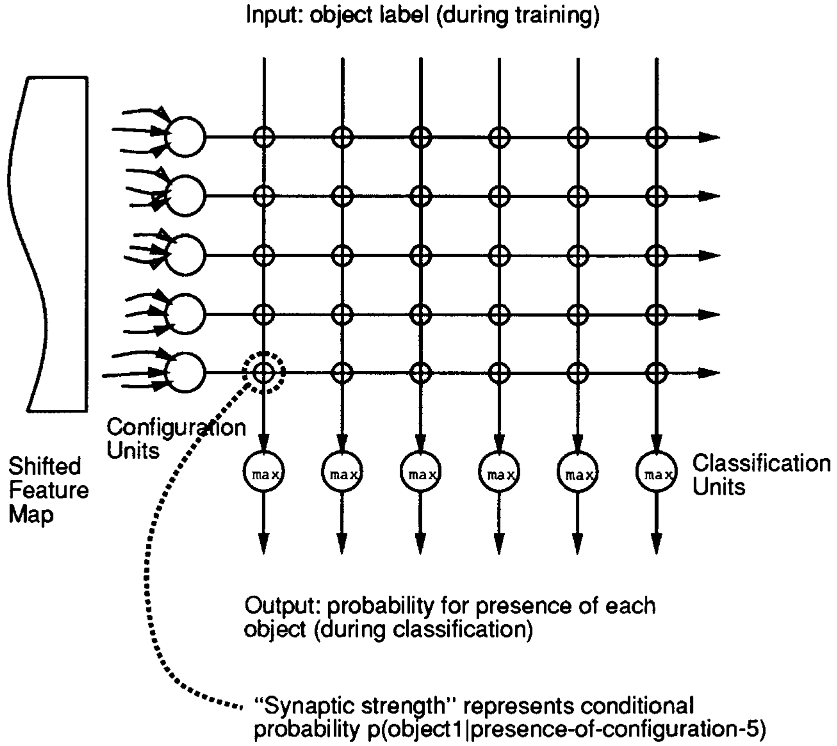


Figure 11: A network analog of the hashing scheme used in the serial implementation. Individual configurations could be recognized by small parts of a "dendritic tree", and an individual "neuron" could respond to several such configurations.

optionally, rotate) the feature map into a "canonical position", i.e., such that a particular feature is located in the center of the shifted feature map (SFM). If we divide the SFM into regions, combinations of regions correspond to configurations of features. To simplify the analysis in earlier sections of this paper, and to allow an efficient implementation on a serial computer, we have assumed a regular, non-overlapping tiling of the plane for the encoding step, but this is not a fundamental requirement for the indexing algorithm; in the context of biological systems, partially overlapping tiles without sharp boundaries (e.g., with Gaussian weighted inputs) are more natural for the encoding step.

This mechanism for encoding has both serial and parallel components: the shifter circuits serially move individual features into the center of the shifted feature map, but once the shifting has been carried out for a particular feature, the encoding proceeds in parallel. Initially, the selection of features to be moved into the center of the SFM could proceed completely in a bottom-up fashion, for example driven by textons. Once partial knowledge about the scene has been obtained from the configurations that have already been computed, search strategies associated with models that are compatible with the known configurations can be used to select further features for shifting.

The implementation of the encoding step in "neural network" hardware has some interesting properties: the units that respond to the presence of particular configurations in the SFM do not require specific point-to-point wiring, but can take advantage of locally randomized connections. Local randomization of connections has been suggested, for example, by Edelman, 1989, as an important principle of neural development and function, and from a biological point of view, "neural networks" that are not compatible with local randomization appear implausible for explaining cortical functions. (An implementation of shifter circuits in terms of neuronal hardware, and biological and psychophysical ev-



Input: object label (during training)
Shifted Feature Map
Configuration Units
max
max
max
max
max
max
Classification Units
Output: probability for presence of each object (during classification)
"Synaptic strength" represents conditional probability p(object1|presence-of-configuration-5)

Figure 12: A simple version of a Bayesian "neural network" that can be used for classifying based on configurations.

idence for their existence has been discussed, for example, in Andersen and van Essen, 1987).

The hashing scheme introduced in earlier sections to speed up the serial algorithm and reduce storage requirements also has a possible equivalent in the neural network implementation: rather than having individual "neurons" respond to individual configurations, parts of the dendritic tree could compute the presence of particular configurations, with the neuron responding to the disjunction of all the configurations represented by the various branches of the dendritic tree; this is illustrated in Figure 11.

The second major component of the indexing algorithm is the classification based on configurations. Many neural network implementations of classification algorithms are known, but we have argued before that a simple Bayesian classifier may be sufficient for the indexing problem; coding of the image in terms of configurations has already made the structure of the problem explicit enough that there is no need for more complex pattern recognition algorithms that try to recover "higher order" properties of the data.

A simple network that implements a version of Bayesian classification is shown in

Figure 12. The network is very similar to an "associative matrix" or "associative network". In the figure, horizontal lines represent the output of the configuration units, and vertical lines represent the object identity (coded in unary)[20]. Using a Hebbian-like learning rule, the connection strength between the output of a configuration unit and the input of a classification unit represents the conditional probability that a particular object is present if the given configuration in the image is present.

The problem of determining probabilities for the various possible objects given the conditional probabilities stored in the network is an ill-posed one since there are many possible probability distributions compatible with the measured conditional probabilities. The commonest solution to this problem is to use a maximum entropy approach. Maximum entropy distributions are costly to compute, in general, however. A simpler classification rule, the "maximum posterior rule" (Breuel, 1990), has been found to work well in practice, is easy to compute, and is more robust to dependencies among measurements than the maximum entropy method. In a neural network implementation of this decision rule, the individual classification units in the network in Figure 12 compute the maximum of their inputs, i.e., the maximum of the conditional probabilities. The classification unit with the maximum output is the considered the most plausible hypothesis of an object present in the image.

One problem that is common to any classification method based on conditional probabilities is that the probabilities themselves are only estimates. If some estimates are based on many examples, and others are based on few examples, then the performance of the maximum posterior rule can be poor. Breuel, 1990, describes a network that solves this problem using cross validation. A simple approach would be to suppress the contribution of a synapse unless the corresponding configuration unit has been sufficiently active in the past (so that the synaptic strength based on the Hebbian learning rule is an accurate estimate of the conditional probability).

9 Discussion

The approach to 2D and 3D model base indexing presented in this paper is appealing for several reasons. Foremost, it is easy to implement and widely applicable. It does not make any assumptions other than piecewise smoothness about the viewing transformation, and it can deal with any kind of feature labeling–even feature labels that depend on viewing position–and it easily takes into account information such as priors and context that are much more difficult to incorporate into other algorithms. But this indexing algorithm not only gives information about which objects might be present in the image, it can also be extended to give partial correspondences and grouping information that may be very

[20] For greater similarity with real neurons, the vertical "lines"–dendritic input to the classification units–should really be drawn as trees with connections at their leaves, but for simplicity they are shown as lines

useful for subsequent verification steps.

The algorithm trades off moderately expensive pre-processing for very low per-model matching overhead. In fact, the per-model processing is so simple that it can easily be implemented in hardware similar to an associative memory chip. As such, the algorithm also provides a plausible starting point for neural network models of low-level recognition in humans. The pre-processing of the image can be formulated in terms of textons and shifter circuits together with simple "higher order" feature detectors that respond to the simultaneous presence of several features in the scene. The Bayesian classification scheme used in identifying plausible models can be expressed in terms of a modified Hebbian algorithm, and an implementation of the algorithm does not require locally precise point-to-point wiring (and, in fact, can take advantage of locally random interconnections of neurons).

Previous authors have also recognized the advantages of representing 3D objects as collections of 2D images (and, possibly, $2\frac{1}{2}$D viewer centered depth maps). Such a representation makes the incremental model acquisition problem trivial and allows for fast matching. However, this paper appears to be the first one that analyzes the complexity and storage requirements for such a representation formally.

Perhaps most similar to the approach presented in this paper is the approach by T. Poggion and S. Edelman, 1990, based on Radial Basis Functions (RBF's); they also do not require explicit models of the viewing transformation and can deal with non-rigid and non-rigidly-attached features. However, as formulated, their approach requires known 1-1 correspondences between image and model features, it does not address the indexing problem, and the question of complexity is left open. The "neural network" presented in Section 8 could be re-formulated and interpreted in terms of units with response properties of RBS's.

The approach presented here is also related to Fourier- and Mellin-transform techniques; in fact, the encoded feature vector can be viewed as a kind of transform of the image. This transform shares with Fourier methods that it is invariant under 2D translations, rotations, and changes of scale. However, the behavior of the encoded bit vector representation under addition and removal of features is much more predictable since there is no "destructive interference". This predictability is what enabled us to derive formal bounds on the complexity and robustness of the indexing method.

The PARVO system (Bergevin and Levine, 1988) for recognition is similar in spirit to the indexing approach presented in this paper because it is based purely on line drawings and feature locations and attempts to assemble line segments and other features into components in a bottom-up fashion before recognition. Feature combinations can be considered such components. However, in other respects, the two approaches differ widely. PARVO is intended as complete recognition system, whereas the indexing algorithm presented here is intended only for indexing; PARVO incorporates many heuristics and much *a-priori* knowledge, as opposed to being based almost entirely on statistical information collected directly from large numbers of images. No formal analyses of complexity or robustness comparable to those presented in this paper have been given for PARVO.

This paper only considered representations as unstructured collections of 2D views. While we have seen arguments that such an unstructured representation may be feasible, additional information such as neighborhood relations on the viewing sphere and and infinitesimal generators for rotations can be incorporated to improve performance. Such additional information could be derived easily, for example, from image sequences, and augment the representation as 2D views, without going to more complex full 3D representations of object shapes.

The algorithms in this paper used configurations of fixed size. However, making the size of a configuration adaptive will probably improve performance greatly in practice. Using adaptive configuration sizes is similar to tree-based methods for approximating probability distributions, where deeper branches of the tree correspond to larger configurations.

If human recognition is based on principles similar to the ones discussed in this paper, there are several predictions. Firstly, because 3D objects are represented as collections of 2D views, inconsistent 3D models are possible, and recognition performance should depend not only on object familiarity, but also on how familiar a particular object is from a particular viewpoint; of course, effects like these have been observed already in certain specific settings (see Marr, 1982 for a straightforward example). The hypothesis that encoding is based on configurations and uses shifting and attentional mechanism also gives rise to several testable psychophysically and neurophysiological predictions.

In Section 4 we noted that for polar tilings, the resolution in the radial direction should decrease (equivalently, the tiles should increase in size in that direction). This theoretical fact could be taken as one explanation for the observation that in the visual system resolution decreases from the center to the periphery.

As we have seen, spatial configurations of small numbers of features capture much of the spatial structure of objects and images at the feature level. Configurations will probably not only be a useful tool for indexing, but also for grouping and other low-level, feature based operations.

Acknowledgements

The author would like to thank especially Eric Grimson for his careful reading of the manuscript and his many helpful comments. Tomaso Poggio, Shimon Ullman, Robert Thau, and Todd Cass also helped with comments on the drafts and discussions.

References

Alt H., Mehlhorn K., Wagener H., Welzl E., 1988, Congruence, Similarity, and Symmetries of Geometric Objects., *Discrete and Computational Geometry.*

Andersen C. H., van Essen D. C., 1987, Shifter circuits: a computational strategy for dynamic aspects of visual processing., *Proc. Natl. Acad. of Sci. USA*, 84:6297–6301.

Ayache N., Faugeras O. D., 1986, HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objecs, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):44–54.

Baird H. S., 1985, *Model-Based Image Matching Using Location*, MIT Press, Cambridge, MA.

Bergevin R., Levine M. D., 1988, Recognition of 3-D Objects in 2-D Line Drawings: An Approach Based on Geons., Technical Report TR-CIM-88-24, McGill University, Montreal, Canada.

Besl P. J., Jain R. C., 1985, Three-dimensional Object Recognition, *ACM Computing Surveys*, 17(1):75–145.

Bolles R. C., Cain R., 1982, Recognizing and Locating Partially Visible Objects: The Local–feature–focus Method, *International Journal of Robotics Research*, 1(3):57–82.

Bolles R. C., Horaud P., Hannah M., 1983, 3DPO: A Three-Dimensional Part Orientation System, In *Proceedings IJCAI*, pages 1116–1120.

Breuel T. M., 1987, 3D Recogntion from 2D Views, Unpublished manuscript.

Breuel T. M., 1989, Adaptive Model Base Indexing, In *Proceedings: Image Understanding Workshop*, pages 805–814, Los Angeles, CA, Morgan Kaufmann, San Mateo, CA.

Breuel T. M., 1990, A robust and efficient alternative to Bayesian maximum-entropy methods, *In preparation.*

Cass T. A., 1988, A Robust Implementation of 2D Model-Based Recognition, In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, Ann Arbor, Michigan.

Edelman G. M., 1989, *Neural Darwinism*, Wiley.

Goldberg R., Lowe D., 1987, Verification of 3D parametric models in 2D image data, In *Proceedings of the IEEE Computer Society Workshop on Computer Vision (Cat. No.87TH0210-5).*

Grimson W. E. L., Lozano-Perez T., 1985, Recognition and Localization of Overlapping Parts from Sparse Data, Technical Report A.I. Memo 841, MIT.

Grimson W. E. L., 1984, The Combinatorics of Local Constraints in Model-Based Recognition and Localization from Sparse Data, Technical Report A.I. Memo 763, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Grimson W. E. L., 1988, The Combinatorics of Object Recognition in Cluttered Environments using Constrained Search, Technical Report A.I. Memo 1019, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Hall P., 1988, *Introduction to the Theory of Coverage Processes*, John Wiley & Sons, New York, NY.

Huttenlocher D. P., Ullman S., 1987, Recognizing Rigid Objects by Aligning Them with an Image, Technical Report AI–Memo 937, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Lamdan Y., Wolfson H.-J., 1988, Geometric Hashing: A General and Efficient Model-Based Recognition Scheme, Technical Report Technical Report No. 368, Robotics Report No. 152, New York University, Robotics Research Laboratory, Department of Computer Science, New York, NY.

Lowe D., 1987, The viewpoint consistency constraint, Int. J. Comput. Vision (Netherlands), 1(1).

Marr D., 1982, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman and Company, San Francisco.

Shirai Y., 1981, Use of models in three dimensional object recognition, In *Sata T., Warman e., ed.s,* Man-machine communication in CAD/CAM. Proceedings of the IFIP wg5.2-5.3 working conference.

T. Poggion and S. Edelman , 1990, (3D Object Recognition using Radial Basis Functions, *Nature*, 343(6255):263–266.

Turney J. L., Mudge T. N., Volz R. A., 1985, Recognizing Partially Occluded Parts, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(4):421–410.

CS-TR Scanning Project
Document Control Form

Date : 4 / 19 / 95

Report # AIM-1108

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
□ Laboratory for Computer Science (LCS)

Document Type:

□ Technical Report (TR) ☒ Technical Memo (TM)
□ Other:

Document Information

Number of pages: 33(39-IMAGES
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
□ Double-sided

Intended to be printed as :
□ Single-sided or
☒ Double-sided

Print type:
□ Typewriter □ Offset Press ☒ Laser Print
□ InkJet Printer □ Unknown □ Other:

Check each if included with document:

☒ DOD Form (2) □ Funding Agent Form □ Cover Page
□ Spine □ Printers Notes □ Photo negatives
□ Other:

Page Data:

Blank Pages(by page number):

Photographs/Tonal Material (by page number):

Other (note description/page number):

Description : Page Number:

IMAGE MAP (1-33) PAGES #'ED 1-33 (INCLUDING TITLE PAGE)
(34-36) SCANCONTROL, DOD (2)
(37-39) TRGTS (3)

Scanning Agent Signoff:
Date Received: 4 / 19 / 95 Date Scanned: 4 / 20 / 95 Date Returned: 4 / 21 / 95

Scanning Agent Signature: Michael W. Cook

Rev 9/94 DS/LCS Document Control Form cstrform.vsd

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AI Memo 1108	2. GOVT ACCESSION NO. AD-A229320	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Indexing for Visual Recognition from a Large Model Base		5. TYPE OF REPORT & PERIOD COVERED memorandum
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Thomas M. Breuel		8. CONTRACT OR GRANT NUMBER(s) S1-801534-2, DACA76-85-C-0010, N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, Massachusetts 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd Arlington, Virginia 22209		12. REPORT DATE August 1990
		13. NUMBER OF PAGES 33
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, Virginia 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

indexing	object recognition
visual recognition	computational complexity
3D	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper describes a new approach to the model base indexing stage of visual object recognition. Fast model base indexing of 3D objects is achieved by accessing a database of encoded 2D views of the objects using a fast 2D matching algorithm. The algorithm is specifically intended as a plausible solution for the problem of indexing into very large model bases that general purpose vision systems and robots will have to deal with in the future. Other properties that make the indexing algorithm attractive are

(con't on back)

(box 20 con't.)

that it can take advantage of most geometric and non-geometric properties of features without modification, and that it addresses the incremental model acquisition problem for 3D objects.

In particular, the paper

- introduces the notion of quantized configurations as a means of compactly encoding the spatial relationships among features in an image in a way that is useful for efficiently representing important geometric and statistical properties of images for recognition and indexing,
- presents an indexing algorithm based on quantized configurations and gives a derivation of the expected indexing speedup for the case of 2D recognition of object models consisting of unlabelled points in the presence of noise and occlusions,
- analyses the complexity of using collections of encoded 2D views for the representation of 3D objects for indexing,
- proposes and analyzes a hashing scheme that allows the per-model overhead of matching to be reduced to a simple BIT-AND operation,
- presents data on the empirical performance of the indexing algorithm in the 2D and 3D case,
- and considers implications of the results for psychophysics and neurobiology.