MEMO #93–08
June 1993

The 3D Indexing Problem

Thomas M. Breuel
IDIAP
C.P. 609, 1920 Martigny, Switzerland
tmb@idiap.ch

Abstract

This paper discusses the problem of 3D indexing. That is, given a collection of 3D object models (CAD models, formalized as collections of labeled 3D points), an indexing algorithm may perform off-line pre-processing to generate a data structure that allows it to decide quickly whether a given 2D image was contains any of the objects represented by the 3D models.

We first review and develop some mathematical machinery. Next, the indexing problem is related to well-known point-location and search problems in computational geometry. Based on such relationships, a number of asymptotically optimal indexing algorithms are discussed.

We then observe that all algorithms related to this problem that have been developed in computer vision as well as in computational geometry show either suboptimal (near-linear) complexity in the number of 3D models, or require a super-polynomial amount of space to represent the result of the pre-computation.

Such computational limitations are illustrated by a discussion of several commonly used approaches to 3D indexing (view-based indexing, indexing by invariants).

We conclude with the speculation that, ultimately, the situation for 3D indexing algorithms is likely to remain analogous to that of high-dimensional geometric query problems: while asymptotically efficient algorithms are known even in high dimensions, most practical applications use linear time algorithms and are content with methods that give significant constant-factor speedups.

Keywords: 3D model base indexing, view sets, point location algorithms, algebraic varieties, arrangements, visual object recognition, invariants, geometric hashing, 3D object recognition, computational complexity, computational geometry, computer vision.

1 Introduction

Motivation It is generally expected that future systems for visual object recognition will have to cope with very large model bases, possibly containing hundreds of thousands of geometric shapes. But research in algorithms for visual object recognition has concentrated on methods for matching a single image against a single model (although this has been changing over recent years). Using such recognition algorithms directly for the recognition of objects from large model bases would lead to a linear complexity of the recognition system in the number of models. Some researchers, driven by practical necessity, have come to conclude that significant gains in recognition speed can be made if we do two things. First, we can take advantage of the fact that the model base is usually (nearly) static. This means that we can perform a significant amount of off-line pre-processing in order to achieve faster on-line recognition. Second, we can attempt to find recognition algorithms that operate on multiple models simultaneously and whose complexity in the number N of models is sublinear. We will call recognition methods that allow pre-processing *indexing methods*.

A number of different indexing methods have been proposed in the literature. Individual authors have generally been enthusiastic about the efficacy of their method for large model bases. Unfortunately, experimental support for such optimism has been relatively limited (even databases containing a few hundred objects are unusually large for current experiments). Furthermore, only the asymptotic complexity (often "average case") of individual algorithms has been carried out, usually without even an explicit statement of the computational model used.

Goals This paper seeks to provide a common framework in which to analyze and compare indexing algorithms. Furthermore, we develop simple asymptotic bounds on the complexity of the indexing problem through the application of recently developed algorithms in computational geometry to the indexing problem.

It should be emphasized that the simple algorithms presented in this paper are only intended to illustrate asymptotic complexity bounds, and are *not* intended for practical application (with the exception of view-based indexing, which already has proven its applicability in several circumstances). The situation is analogous (and related, as we will see) to the situation with algorithms for nearest-neighbor lookup in high dimensional Euclidean spaces: a number of asymptotically efficient algorithms are known, but in practice, linear search seems to be the best exact (as opposed to heuristic) algorithm.

Essential for the formal complexity analysis of any recognition algorithm is the choice of an error and imaging model. The error model used is that of bounded error, which is in wide use and has proven its utility in many practical recognition systems and theoretical analyses of the recognition system. Our imaging model is that of 3D rigid body rotation with orthographic projection. Throughout most of the paper, we will assume that all correspondence are known.

This formalization of recognition is decidedly too optimistic: correspondences are not known in practice, and real cameras do not use orthographic projection. However, this strengthens many of the results: if an indexing algorithm (e.g., recognition by invariants) already fails to give a substantial speedup when correspondence are known, it is not going to do so when correspondence are unknown.

Approach We begin by analyzing the computational and geometric aspects of the indexing problem. We observe that (because of output complexity) $\Omega(\log N)$ constitutes a lower bound for the indexing problem. This raises the important question of whether this bound can be achieved (by an *optimal indexing algorithm*).

We then examine the properties of *view sets*, i.e., the set of views that an object can give rise to under different transformations, surveying and discussing both old and new results.

Using view sets, we can reformulate the indexing problem as a set membership or point location problem with pre-processing. By showing that view sets form an arrangement of algebraic varieties, we can apply recently developed algorithms from computational geometry to achieve $O(\log N)$ indexing, where N is the size of the model base. This is an optimal indexing algorithm by our definition above.

However, the algorithms used in the construction of an optimal indexing algorithm are general-purpose point location algorithms with poor space bounds and high complexity in the size of each model. Therefore, we will examine some ways in which this complexity can be reduced, by taking advantage of the special properties of view sets (as opposed to general algebraic sets), and by considering approximations.

One particularly important approximation that is already widely used as a heuristic in computer vision is the *view-based approximation* (also known as *multi-view representations*). We will relate indexing methods based on the view-based approximation to the geometric view of view sets developed in this paper.

Finally, we will compare the performance and complexity of several major approaches to the indexing problem, and we will point out directions for further research.

2 Geometry

Bounded Error Recognition Before discussing the indexing problem, we have to define precisely what we mean by "3D recognition". The most commonly used approach is the following.

A camera generates an image (or view) of an object in the real world. Objects are described by 3D object models. We assume that objects do not change their identity under 3D rigid body motions. In different words, object identity is *invariant* under 3D rigid body transformations. Furthermore, in order to allow for the possibility of modeling and sensing error, we allow slight deviations between the image of an object predicted from its model and its actual image.

For concreteness, we consider object models and images that consist of *point features*. That is, a model is a finite set of points in $\mathbb{R}^3$, and an image is a finite set of points in $\mathbb{R}^2$. It is important to realize that the framework we will develop below is equally applicable to more complicated features. For example, if we use curves described by 4th degree polynomials as features, each feature is a (single) point in $\mathbb{R}^{15}$.

Furthermore, for the time being, we will assume that the segmentation ("figure/ground") and correspondence problems have been solved. This assumption is commonly made in the analysis of indexing algorithms. Usually, so it is argued (e.g., Jacobs, 1988, Grimson, 1990), segmentation and correspondence information can be (and must be) derived from the image, and any remaining ambiguity can be resolved by trying different possibilities.

Another way of looking at this assumption is that the input to the indexing algorithm consists of a curve, represented as a spline of degree 1 (the methods discussed below also work in principle for splines of higher degree). This view may make the approach more palatable to those that consider the use of, say, invariants on algebraic curves as input to an indexing algorithm; using a curve described by a larger number of parameters as input is really equivalent to assuming knowledge of image segmentation and the correspondence between a number of image and model points.

If we make these assumptions, we can mathematically formalize the recognition problem as follows. Assume that the model and the image consist each of K points. Then, a model can be described as a point in $\mathbb{R}^{3K}$ (also called *model space*) and an image can be described as a point in $\mathbb{R}^{2K}$ (also called *view space*). Let us use the notation m_k for the location of feature k

in the model, and $m_{k,i}$, $i = 1 \dots 3$, for its 3 coordinates. Define b_k and $b_{k,i}$, $i = 1 \dots 2$, analogously for image features.

We declare a match between a model and an image if there exists a transformation $T \in \mathcal{T}$ that transforms model into the image to within given error bounds ϵ and under an error model d. Note that T is a function from $\mathbb{R}^3$ into $\mathbb{R}^2$. To simplify notation, we denote the transformed vector b defined by $b_k = Tm_k$ simply as Tm, keeping in mind that T is applied to each of the k point features of m.

With this notation, we define a predicate "match" between a model m and an image (view) b:

$$\mathsf{match}(m, b) := \exists T \in \mathcal{T} \;\; d(Tm, b) < \epsilon \tag{1}$$

Error Models We will consider two error models d. The first is the error model associated with least-square matching, d_{lsq}: the total error is equal to the square root of the sum of the squares of the errors associated with each model feature:

$$d_{\text{lsq}}(b, b') = \sqrt{\sum_{k=1}^{K} \|b_k - b'_k\|^2} \tag{2}$$

The second is the error model associated with bounded error matching, d_{be}: the total is the maximum of the errors associated with each model feature:

$$d_{\text{be}}(b, b') = \max_{k=1 \dots K} \|b_k - b'_k\| \tag{3}$$

It is not difficult to see that both of these error models define metrics in view space. For example, we can rewrite d_{lsq} as:

$$d_{\text{lsq}}(b, b') = \sqrt{\sum_{k=1 \dots K, i=1 \dots 2} (b_{k,i} - b'_{k,i})^2} \tag{4}$$

This is, of course, simply the Euclidean distance of b and b' considered as vectors in $\mathbb{R}^{2K}$.

This is a useful observation, since it implies an intuitive and useful relationship between the error-free recognition problem and the problem of recognition in the presence of error; we will also take advantage of this fact for the view-based indexing methods described below, since the point location algorithms in view-based indexing methods rely on metric properties of view space.

Note that the single inequality $d_{\text{be}}(b, b')^2 < \epsilon^2$ involving the non-polynomial function max can be considered a collection of K inequalities, one for each $k = 1, \ldots, K$, in each of which b and b' occur only in a quadratic. This means that in the subsequent considerations, when we derive results using d_{lsq}, analogous results involving K inequalities (instead of a just single inequality), hold for d_{be}.

Transformations Let us now turn to the question of classes of transformations $\mathcal{T}$. Transformations are made up of two parts: a fixed camera model that describes how the scene in front of the camera is projected onto the image plane, and a variable part that describes the relative position of camera and object.

For our camera model, we will be using *weak perspective*. That is, we model the camera as orthographic projection with a change of scale. Like our choice of point features, this is mainly for concreteness and simplicity; the arguments we make below also work for perspective projection or even more complicated (algebraic) camera models.

The transformations under which we consider 3D objects invariant are 3D rigid body transformations, defined by rotation matrices R and translations t. Therefore, we define the set of transformations $\mathcal{T}^r$ as follows:

$$\mathcal{T}^r = \{t(p) = P(rRp + t) : r \in \mathrm{IR}, R \in \mathrm{IR}^{3\times3}, t \in \mathrm{IR}^3, \textstyle\sum_{j=1}^3 R_{ij}R_{jk} = \delta_{ik}\} \quad (5)$$

In this definition, $P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, i.e., orthographic projection. Note that there is a set of 6 quadratic constraints imposed on the nine parameters of R.

We could also use Euler angles for parameterization; if we let s_i and c_i stand for the sine and cosine of Euler angle i, respectively, this gives rise to the following definition:

$$\mathcal{T}^r = \{t(p) = P(rR(s,c)p + t) : r \in \mathrm{IR},\ s, c, t \in \mathrm{IR}^3, s_i^2 + c_i^2 = 1\} \quad (6)$$

Here, we impose 3 constraints on the 6 parameters defining R.

The quadratic constraints imposed on the form of R complicate algorithms and analysis considerably. Several researchers have therefore studied the case of affine or linear 3D transformations, in which those constraints are dropped (from the large literature, the most relevant to our discussion here is Ullman and Basri, 1989). We define the corresponding set of transformations $\mathcal{T}^\ell$ under a weak perspective camera model as follows:

$$\mathcal{T}^\ell = \{t(p) = P(Rp + t) : r \in \mathrm{IR}, R \in \mathrm{IR}^{3\times3}, t \in \mathrm{IR}^3\} \quad (7)$$

To simplify notation, we will also drop the translational component from these definitions; analogous results to those stated below hold if translation is being considered. Then, the set $\mathcal{T}^r$ becomes:

$$\mathcal{T}^r = \{t(p) = PrRp : r \in \mathbb{R}, R \in \mathbb{R}^{3\times 3}, \sum_{j=1}^{3} R_{ij}R_{jk} = \delta_{ik}\} \tag{8}$$

In this case, the set $\mathcal{T}^\ell$ simply reduces to the space of 2×3 matrices:

$$\mathcal{T}^\ell = \mathbb{R}^{2\times 3} \tag{9}$$

Canonicalization A geometric property of the recognition problem that many indexing algorithms take advantage of is that certain (though not all) 3D rigid body transformations can be compensated for in a model independent way by transforming the image. Concretely, in the case of $\mathcal{T}^r$, i.e., rigid 3D body transformations, 2D equiform transformations of the object in the image plane (i.e., translation, rotation about the optical axis, and changes of scale) can be compensated for by 2D equiform transformations of the image. This follows easily from general considerations of symmetry of the system camera–object. In the case of 3D rigid body transformations, we can derive the result more specifically by considering an Euler-angle parameterization of the 3D rotation matrix.

This process lets us account for 4 of the 6 parameters that determine the relative pose of image and camera. It is well known that the remaining 2 parameters can be identified naturally with the surface of a 2-sphere known as the *viewing sphere* (cf. Horn, 1985).

Compensating for 2D equiform transformations of an object in a bottom-up (model-independent) way is referred to as *canonicalization*. Because different 3D objects may have identical 2D views (see, for example, Ullman, 1979), it is clear that canonicalization cannot be extended to compensate for arbitrary 3D transformations: a particular view that is shared by two different 3D objects would have to be canonicalized in two different ways (recently, this simple observation has been strengthened in the context of research on viewpoint invariants; see Burns *et al.*, 1990, Clemens and Jacobs, 1991, Moses and Ullman, 1991).

The observation that we can only compensate for 4 of the 6 parameters of 3D rigid body transformations naturally has lead researchers to the notion of *view-based indexing*, in which the remaining 2 parameters that have not been canonicalized for, are simply represented by sampling ("sampling of the viewing sphere"). We will provide references and analyze view-based indexing methods in more detail below.

A common way of implementing canonicalization under 2D equiform transformations is to use an alignment method, i.e., to pick two point features in the image and to transform the whole image so that those features lie on the origin and the point $(1, 0)$.

It should be noted that even in the case of canonicalizing for 2D equiform transformations, canonicalization can only be approximate in the presence of error. That is, there is no recognition algorithm based on canonicalization that solves a recognition under error exactly. The reasoning is the same as in the case of the impossibility of canonicalizing for all 6 parameters of 3D rigid transformations: two different models might share some views in the presence of error. Any such shared view would have to be canonicalized in two different ways. Intuitively, the reason for this is that the alignment points themselves are subject to a certain amount of unknown error.

3 View Sets

Definition A key idea for the approach to indexing described in this paper is that the definition of recognition given in Equation 1 can be rewritten as a set membership problem:

$$\mathsf{match}(m, b) := b \in B_m \tag{10}$$

In this equation, we call B_m the *view set* for the given model. Intuitively, what we are saying is that B_m is the set of all possible views that can be generated by the model m under the given set of transformations and error model, and that a view (image) matches a model if only if it is contained in the set of views generated by that model.

Of course, many researchers have implicitly precomputed, represented, and approximated view-sets in any number of ways, including hash tables, arrays, and linear spaces (see the section on view-based indexing below). In fact, it can be argued that *any* recognition algorithm that performs some kind of preprocessing represents properties of view sets that are helpful for later membership queries. Some properties of view sets have also been used in establishing a number of theoretical results (see, for example, Ullman and Basri, 1989, Lamdan *et al.*, 1990, Moses and Ullman, 1991, and Clemens and Jacobs, 1991).

What we will see in this paper is that view sets can be described exactly in terms of simple algebraic expressions, and that such representations can be used for asymptotically optimal indexing algorithms. Furthermore, a better understanding of the exact shape of view sets (even in the presence of errors) may help to design better actual recognition systems that may represent view sets only approximately.

Now, if we expand Equation 10, it appears that it is not particularly well suited to algorithmic membership queries. The reason is that the definition of the set B_m contains an existential quantifier:

$$B_m = \{b \in \mathbb{R}^{2K} : \exists T \in \mathcal{T}\ d(Tm, b) < \epsilon\} \tag{11}$$

Let us see whether we can eliminate this quantifier.

Linear Case Consider the case where $\mathcal{T} = \mathcal{T}^\ell$. In this case, it can be shown (Ullman and Basri, 1989) that, in the error free case and for sufficiently large K, the view set is a 6-dimensional linear space. Let $w_i(m)$, $i = 1 \ldots 6$, be an orthonormal basis for this space (for convenience, we will leave out the dependence on m for now). Then,

$$\begin{aligned} B_m^\ell &= \{b \in \mathbb{R}^{2K} : \exists T \in \mathbb{R}^{2\times 3}\ Tm = b\} & (12)\\ & \{b \in \mathbb{R}^{2K} : \exists \lambda_i \in \mathbb{R}, i = 1 \ldots 6\ \sum_{i=1}^{6} \lambda_i w_i = b\} & (13)\\ & \{b \in \mathbb{R}^{2K} : b - \sum_{i=1}^{6} (b \cdot w_i) w_i = 0\} & (14)\\ & & (15) \end{aligned}$$

Here, we take advantage of the fact that the expression $b - \sum_{i=1}^{6}(b \cdot w_i)w_i$, call it e, gives the component of the vector b that lies outside the linear space B_m^ℓ. Note that the last of these expressions does not contain any existential quantifier anymore.

Based on this expression for e, it is not difficult to extend the quantifier-free version of B_m^ℓ to the case in which we allow for error ϵ under the error measure d_{lsq}. Namely, we require that $\|e\| < \epsilon$, or, equivalently, that:

$$B_m^{\ell,\epsilon} = \{b \in \mathbb{R}^{2K} : (b - \sum_{i=1}^{6} (b \cdot w_i) w_i)^2 < \epsilon^2\} \tag{16}$$

Nonlinear Case In the case of $\mathcal{T}^r$, i.e., rigid body transformations, the resulting equations are, unfortunately, significantly more complicated. However, there are general procedures for eliminating the set of existential quantifiers (Tarski, 1948, Jacobson, 1974, Collins, 1975, Chazelle, 1985). In practice, we can carry out such computations using systems for symbolic mathematics (e.g., MACSYMA). In this way, we arrive at a set of polynomial inequalities Φ such that B_m^r can be written as follows:

$$B_m^r = \{b \in R^{2K} : \Phi(b, \epsilon)\} \tag{17}$$

Of course, in order to be able to perform this elimination, we need a sufficient number of starting equations, i.e., K must be large enough. In fact, K should be as large as the minimum number of points needed to determine a unique pose.

While Φ has a rather complicated form, there are some important general observations we can make about B^r_m. First, recall that in the error-free case (and neglecting translations), B^r_m is given by the set of vectors b that satisfy $b_{k,i} = \sum_j R_{ij} m_{k,j}$. If we re-order and rename the indices, what this says is that B^r_m is the image of the set of all transformation matrices $R \in \mathcal{T}$ under a linear map, say, M, defined by $m_{k,j}$.

In general, M may not have full rank (i.e., rank 6). However, we know that in many cases ("non-degenerate models") any image of a model uniquely determines the transformation matrix R. In different words, for such models, there is a 1-1 correspondence between views b and transformations R. By definition of B^r_m, R is also mapped onto B^r_m. Because B^r_m is a linear transformation of the manifold of transformations R, most of the structure of this manifold therefore carries over to B^r_m.

We can make an analogous argument in the presence of canonicalization, i.e. in the case where we compensate for rotations around the camera axis before performing computing the view set. We mentioned above that after canonicalization, the remaining 2 parameters describing the relative position of camera and object can be identified with the surface of a 2-sphere (S_2), called the viewing sphere, and by an argument analogous to the one given above, in the presence of canonicalization and the absence of error, view sets of non-degenerate objects under rigid body rotation are therefore isomorphic to S_2.

In the presence of error, the structure of the view set becomes more complicated. However, as we observed above, the commonly used error measures d_{lsq} and d_{be} define metrics in view space. Hence, the view set of an object under rigid body rotations in the presence of error ϵ is simply the *dilation* of the view-set for the error free case with an ϵ-ball.

We remarked above that $\mathcal{T}^r \subseteq \mathcal{T}^\ell$. It is easy to see that this property carries over to view sets for both cases, $B^r_m \subseteq B^\ell_m$ (even in the presence of error, because dilation with the same set preserves set inclusion). Because the structure of B^ℓ_m is significantly simpler than the structure of B^r_m, testing for membership in B^ℓ_m can be a fast way to exclude membership in B^r_m, suggesting a multi-stage approach to indexing. Note, however, that there are some common classes of objects (e.g., cubes, bricks, and lozenges) that have identical view sets under affine transformations, but are completely distinguishable under rigid body transformations.

Other Features We noted above that we might use more complex features like, parameterized polynomial curves or splines, as input for our recognition algorithms. Without wanting to discuss these more complicated cases in greater detail, it should be noted that the concept of a view set, and hence the algorithms below, apply to these cases as well. View space becomes the parameter space for the curve, and view sets become the allowable sets of parameters under which a distorted version of the curve still matches the model under our error model.

4 Indexing by Point Location

Chazelle, 1985, has recently described a general-purpose point location algorithm that lets us solve the following problem.

Assume we are given a collection $\mathcal{P} = \{P_1, \ldots, P_N\}$ of N rational, r-variate polynomial of degree $\leq d$. These generate an arrangement, i.e., a minimal collection $\mathcal{A}$ of connected regions in $\mathbb{R}^r$ over each of which the sign of each of the polynomials remains constant (see Edelsbrunner, 1987 for an introduction to arrangements). During a pre-processing stage, the algorithm assigns labels to the cells of the arrangement $\mathcal{A}$ and generates an intermediate data structure. Based on this data structure, given a point in $\mathbb{R}^r$, the point location algorithm can determine in time $O(\log N)$ the label of the cell in the arrangement generated by the polynomials.

Not coincidentally, the form in which we have transformed the recognition system above fits this algorithm exactly. Assume that we are given models $m_1, \ldots, m_N$, and consider first the linear case $\mathcal{T}^\ell$. In Equation 16, we saw that b was a member of the view set for m if it satisfied the polynomial inequality $(b - \sum_{i=1}^{6}(b \cdot w_i(m))w_i(m))^2 < \epsilon^2$. So, first, we simply set $P_n(b) = (b - \sum_{i=1}^{6}(b \cdot w_i(m_n))w_i(m_n))^2 - \epsilon^2$. Then, during the preprocessing step, we label each cell of the arrangement generated by the P_n by the set of P_n that are negative on that cell. When faced with a new view, the indexing algorithm retrieves that label (in time $O(\log N)$) and returns it.

We can make an analogous argument for the more complicated case of $\mathcal{T}^r$, essentially by using the formula Φ in the definition of $B_m^{r,\epsilon}$ in Equation 17 for each P_n. Note that, in general, it is not obvious that the elimination of the existential quantifier from Equation 1 results in a single polynomial (in-)equality (rather than a collection of polynomial (in-)equalities and logical connectives). The algorithm described by Chazelle, 1985, can cope with this more general case (for the specific case of indexing under 3D rigid body transformations, this does not appear necessary).

A technical issue of some interest is the following. In principle, the number models that match a particular cell might be as large as N and the output of the indexing algorithm itself might be as large as N. How can we then say that the indexing algorithm can work in time $O(\log N)$? The answer is the following. While a cell might represent a match consisting of as many as N models, it can be shown that the number of cells in the arrangment $\mathcal{A}$ grows only polynomially in N, and if we choose our labels efficiently, a unique label for any cell in $\mathcal{A}$ therefore can have size $O(\log N)$. The fact that the output complexity of the indexing algorithm might be large and dominate the cost of indexing is therefore an artifact of the "natural" representation that we have chosen for its output, namely a list of matching models. To avoid this, we might either allow the indexing algorithm to return a concise, suitably "encoded" representation of the set of matching models, we might take the size of the output into consideration as part of the complexity (stating that the complexity is $O(\log N + W)$, where W is the length of the output, or we might simply restrict ourselves to considering sets of objects such that only a bounded number of models share any particular view (we call such objects *distinct*).

The above considerations show that there exists an optimal indexing algorithm, i.e., a recognition algorithm with preprocessing whose complexity in the size of the model base is logarithmic. However, the general purpose point-location algorithm used in establishing this fact may not practical, due to its complicated structure and high constants.

One particular problem is the fact that the best known upper bound on the size of the pre-processed output (the *space requirement* of the indexing algorithm) is doubly exponential in the number of features K. Specifically, our case, this means an *upper bound* of $O(N^{2^{2K+6}})$ (though this bound is likely to be far from tight). This looks rather imposing; however, often, existing indexing algorithms assume that K is small and constant (e.g., many indexing algorithm based on invariants assume that K is 1 and rely on complex features instead).

Note that if we are willing to accept exponential complexity in K, indexing is still possible in logarithmic time even in the case of point features and if no correspondence information between image and model points is known. The idea is simply to represent all possible permutations of the K features explicitly as distinct models in the model base.

Obtaining better bounds and reducing the complexity for point algorithms like the one we used above is an active area of research (cf., for example, Clarkson, 1987, as well as related work in motion planning, e.g., Schwartz and Sharir, 1983). In addition to such efforts, the indexing problem is considerably more constrained than the problems for which the general purpose algorithms have been designed. Examples of such additional constraints that we might take advantage of are the following. First, all the polynomials P_n have the same form. Furthermore, as we saw above, the region on which each polynomial is negative is the dilation by a small amount ϵ of a very low-dimensional algebraic set (a 6-dimensional linear space in the case of $\mathcal{T}^\ell$ and a 2-dimensional surface contained in a 6-dimensional linear space in the case of $\mathcal{T}^r$) in view space. Finally, tradeoffs between higher (although still polylogarithmic) time complexity and the amount of space needed for the intermediate representation may be possible.

There are two approaches that are worth considering for taking advantage of such special properties. The first approach is to solve the case of indexing for transformations in $\mathcal{T}^\ell$ in logarithmic time with smaller space requirements; since many objects are dissimilar under affine transformations, this might be sufficient for achieving fast indexing for real model bases. Another approach is *view-based indexing*, a well-known, heuristic method for 3D object recognition that can be re-interpreted in a point location framework. This is what we will discuss in the next section.

5 View-Based Indexing

The idea behind view-based indexing is to use a 2D indexing algorithm to match the input image against a large collection of 2D views stored for each model.

View-based indexing is actually not a new technique; multi-view representations have been used extensively in computer vision (for a review, see Korn and Dyer, 1987). Some recent work on view-based methods for indexing specifically include Lamdan and Wolfson, 1988, Breuel, 1990a, Clemens and Jacobs, 1991, Stein and Medioni, 1991, and Flynn and Jain, 1992.

However, even though view-based indexing has been used previously in a number of recognition systems, it has not previously been related to the geometric view of indexing that we have taken in this paper. It turns out that interpreting the technique in the point-location framework discussed above shows that it is, in fact, very similar to "true" 3D-based indexing systems, while at the same time being empirically robust, efficient, and easy to implement.

To analyze this relationship between view-based methods and point-location methods for indexing, we need to make more precise what we mean by a view-based recognition or indexing method. The basic idea is that, in order to recognize a 3D object in a 2D image under bounded error ϵ, we match a collection of 2D views under bounded error δ, where δ is usually slightly larger than ϵ. A an analysis of the formal relationship between the two approaches can be found in Breuel, 1992.

View-based methods do not take any direct advantage of the structure of the view set or even of the fact that it is contained in a low-dimensional linear subspace of view-space. However, view based indexing can still be understood easily in such a framework.

For the purposes of this discussion, let us assume that the input to the preprocessing stage of the indexing method is error free (e.g., a CAD model). As is common in multiview recognition methods, a large number of views are generated by deterministic or stochastic sampling of the viewing sphere and stored. View-based indexing now proceeds by matching these individual views under error bound δ against an unknown input view. For every stored view that matches the image, a match to the corresponding model is declared.

As we noted above, view-based techniques for 3D object recognition are not new, but analyzing them in the theoretical framework we have developed above clarifies their relation to other 3D indexing methods.

First, recall that bounded error matching or least-square matching defines a metric in view space. Each view that is matched under an error bound δ therefore corresponds to a δ-ball in view space. The effect of sampling the (error-free) view set for a 3D object model and matching the samples under an error bound δ is therefore to cover the view set with δ-balls (in the metric in view space).

View-Based Approximation Now, it is important to note that, unlike the point location algorithm discussed in the previous section, view-based indexing is only an approximation to the indexing problem. The reason is that the view set of a 3D object cannot (in general) be represented as the union of a finite number of δ-balls.

However, by choosing the size and placement of the δ-balls, we can make the approximation arbitrarily close, and we can make tradeoffs between positive mistakes (incorrectly recognizing an object that is not present) and negative mistakes (failing to recognize an object that is actually present). Negative mistakes are usually much more serious in practice than positive mistakes, and hence we would like to make the probability of negative mistakes zero.

Geometrically, positive mistakes correspond to regions in view-space that are covered by the δ-balls of the view-based approximation but not by the view set of the corresponding model. Conversely, negative mistakes correspond to regions in view-space that are covered by the view set for some object but not by the union of the corresponding δ-balls.

Since we want to achieve zero negative mistakes, we must cover the view set for each object completely by δ-balls. In different words, we must approximate view sets *from the outside*.

Both the number of δ-balls needed to cover the view set and the probability of positive mistakes depend on the choices of δ and ϵ. Large δ mean that we need few δ-balls to cover each view set, but they also mean a high probability of mistakes. Small δ mean that we need many δ-balls to cover each view set, but that the probability of positive mistakes can be made small. Furthermore, by choosing δ closer to ϵ, we can make the probability of positive mistakes arbitrarily small (an analysis of the probability of complete coverage can be found in Breuel, 1989, and Breuel, 1990a).

Range Query Algorithms Algorithmically, "covering view sets by δ-balls" means the following. Given that we have reformulated the indexing problem as the problem of determining membership in the view set, under the view-based approximation, indexing then becomes the problem of determining whether a input view is contained in one of the δ-balls covering a view set. In different words we want to find those centers of δ-balls that are within a distance of δ of an input view.

The problem of indexing under the view-based approximation is therefore simply a $2K$-dimensional range query problem. The most common approaches to solving such problems are *binning*, *k-D trees*, and *range trees* (for detailed discussions and references, the reader is referred to the literature; e.g., Preparata and Shamos, 1985, Samet, 1990).

As we noted above, view based approaches to indexing have been taken previously. The algorithms for point location used in such approaches to indexing have usually been based on binning (Breuel, 1990b, Clemens and Jacobs, 1991). Among known range query algorithms, binning methods solve the range query problem fastest, both asymptotically and in practice. However, in the presence of error, their space requirements are exponential in K (see Breuel, 1990b for a more detailed analysis of the space requirements of such algorithms).

Using k-D trees for solving the point location problem results in a space requirement of $O(KN_v)$. This is clearly a tight, optimal bound. For k-D trees, the average case time complexity for a query is $O(\log N_v)$; unfortunately, the worst case time complexity, while sublinear, is only guaranteed to be $O(KN_v^{1-\frac{1}{2K}})$.

The third range query data structure, the range tree, guarantees asymptotic time complexity $O(\log^K N_v)$. However, its space requirements can be exponential in K, namely $O(N_v \log^K N_v)$.

In summary, from a complexity point of view, the view-based approximation, i.e., approximating the view-set of an object by a collection of δ-balls in view space, somewhat simplifies the indexing problem and lets us use range query algorithms rather than general point location algorithms in order to achieve fast indexing.

It is important to realize, that even in the simpler case of indexing under the view-based approximation, there are no known algorithms that have optimal logarithmic time (in N) model base access and polynomial space requirements in K in the presence of error; this is true even if correspondences between model and image features are known.

Number of Views In order to compare the complexity of indexing under the view-based approximation with the complexity of indexing based on point location, described in the previous section, we need to know how N_v depends on N. This question is relatively easy to answer if we fix δ, say, at 2ϵ, and if we use canonicalization (see Section 2)) to account for 2D equiform transformations of the image. Then, it can be shown that the number of δ-balls to cover the view set is proportional to δ^2; for fixed δ in the absence of occlusions, there is an upper bound independent of K on the number of δ-balls needed (see Breuel, 1990a; in the presence of occlusions, the number of views needed depends on K, see Ikeuchi and Kanade, 1988, for an asymptotic bound).

Effects of the View-Based Approximation One important question that remains what effect the approximate nature of view-based methods has on the ability of the indexing algorithm to distinguish different objects. The magnitude of this effect can be estimated in several ways (for a more detailed analysis, see Breuel, 1992). First, we can compare it to the effects of other, commonly made approximations in computer vision. It can be argued that for a choice like $\delta = 2\epsilon$, the effect of the view-based approximation is no more significant than a choice of metric or similarity measure (d_{lsq}, d_{be}, or alignment) or an independence assumption among the error vectors for each of the features. Second, we can show that by increasing K slightly, we can compensate for the small increase in the probability of mistakes under a view based approximation. Third, by choosing smaller δ, we can reduce the probability of mistakes arbitrarily (at the expense of more space).

6 Indexing by Invariants

An important approach to indexing is that based on invariants (Mundy and Zisserman, 1992, is a comprehensive collection of papers in the field).

The idea behind indexing by invariants is to identify functions for a whole class of objects with the property that these functions are constant over the view set of each object. Because for unrestricted collections of objects, view sets can intersect (i.e., different objects can have identical views), clearly there exist no invariants that are universally applicable (see also Burns *et al.*, 1990, Clemens and Jacobs, 1991, Moses and Ullman, 1991). Therefore, invariants are specific to classes of objects and, in practice, be developed by hand (or semi-automatically, using algebraic manipulation packages).

Another complication with using invariants for indexing is that invariants have primarily been developed for the error-free case. This has several problematic consequences. First, it is not difficult to see that there are no exact invariants (of algebraic curves, points, or even just smooth curves) that can be expressed as algebraic functions for any non-trivial error models, since such invariants would have to be constant over some small ball in view-space, which, by Taylor expansion, would imply that they are constant everywhere and make them rather uninteresting for indexing purposes. This observation, of course, does not preclude the existence of invariants expressed using non-algebraic (e.g., semi-algebraic) functions, but it does suggest that approaches towards finding invariants for interesting classes of objects in the presence of error may require rather different approaches from those currently taken.

Related to this problem is the following. Assume that we are interested in invariants for a parameterized collection $\mathcal{C}$ of objects (e.g., planar quadratic curves viewed from different viewpoints). Let us assume that the relationship between parameters and views of an object is "smooth" (uniformly continuous), and that we are using a "smooth" error measure for comparing views. Now, consider two objects A and B with very similar parameter values. Then, if we are given some error-free view of B, we can either interpret this view as coming from B, or as coming from A with a small amount of error added (small because of the continuity of the error measure). Therefore, under the above smoothness assumptions, there cannot exist any invariant function for distinguishing objects in $\mathcal{C}$ under error. In different words, any class of objects for which we can identify invariant functions in the presence of error must consist of a discrete collection of objects.

We might try to avoid this problem by considering approximate invariants (see Moses and Ullman, 1991), i.e., functions that vary only slowly in the presence of small amounts of error. However, it can be seen that indexing by (approximate) invariants then turns into a general point location problem, not obviously easier than the direct point location formulation of 3D indexing given above.

From the above considerations, we can infer that indexing by invariants in the presence of error requires that we consider only "discrete" classes of objects and use indexing functions that are not algebraic. But this is pretty much what indexing by point-location described in Section 4 does: given a collection of discrete objects, it computes a semi-algebraic function which, when applied to a view, returns a constant identifying which object(s) the view belongs to.

Altogether, indexing and recognition by invariants may be a useful practical tool to obtain significant constant factor speedups in the recognition of some classes of objects. But questions like those raised above about location error and model variation must be addressed.

7 Discussion

In this paper, we have studied one particular aspect of the problem of visual object recognition–that of determining quickly whether an collection of image features could have been derived from one of a large number (N) of given model features under 3D rigid body transformations and known correspondences. This formalization of the indexing problem (including the assumption of known correspondences) forms the basis for a lot of work on fast visual object recognition from large model bases. In this paper, we have described and analyzed two algorithms that solve this problem in optimal (logarithmic) time in N.

The first algorithm is based on a reformulation of the 3D indexing problem as a high-dimensional point location problem. This approach has allowed us to give an asymptotically optimal $O(\log N)$ (though not necessarily practical) algorithm for solving the 3D indexing problem.

Furthermore, we have considered one particularly important simplification of the 3D indexing problem, namely indexing under the view-based approximation. We have seen that indexing under the view-based approximation is equivalent to a high-dimensional point location problem. This has allowed us to relate the complexity of indexing under the view-based approximation to the complexity of a variety of existing range query algorithms.

In the analysis of these indexing algorithms, we have collected and developed a number of fundamental concepts that help us better understand the nature of the indexing problem. The understanding that indexing is a special case of point-location and range-query algorithms is of great practical and theoretical significance. On the one hand, such algorithms are constantly being improved, because they have a wide variety of applications, and, on the other hand, lower bounds that are being discovered for point-location and range-query algorithms may be translatable into lower bounds for the indexing problem. Furthermore, the geometric approach based on view sets helps us understand the tradeoffs and nature involved in various approximations (like view-based indexing) better.

As stated in the beginning, the goal of this paper was not to present a practical indexing algorithm. In fact, the paper has developed a number of asymptotically optimal (i.e., logarithmic in the number of models) indexing algorithms, most notably, one based on Chazelle's point location algorithm and another (approximate) based on the view-based approximation.

On the other hand, we have also seen that the indexing problem is closely related to[1] high-dimensional geometric query problems for which usually linear-time methods are being used, because the asymptotically logarithmic-time algorithms that are known either do not reach the asymptotic regime for realistic problem sizes or have unacceptably high space overhead.

Ultimately, I expect the situation to be similar for indexing in visual object recognition. There will likely be a number of algorithms, like indexing based on invariants or geometric hashing, that greatly reduce the constants of the complexity of linear search, but do not lead to significantly sublinear performance.

References

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., 1990a, Indexing for Visual Recognition from a Large Model Base, A.I. Memo No. 1108, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Breuel T. M., 1990b, Recognition is Polynomial Time Reducible to Verification, A.I. Memo 1268, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Breuel T. M., 1992, *Geometric Aspects of Visual Object Recognition*, PhD thesis, Massachusetts Institute of Technology, Also available as MIT AI Lab TR# 1374.

Burns J., Weiss R., Riseman E., 1990, View variation of point set and line segment features, In *Proceedings Image Understanding Workshop*, pages 650–659.

[1]This relationship can actually be formalized as an efficient transformation of the query problems into the indexing problem.

Chazelle B., 1985, Fast searching in real algebraic manifold with applications to geometric complexity, In *Proc. Coll. on Trees in Algebra and Progr. 1985, Lecture Notes in Comput. Sci. 185*, pages 145–156, Springer Verlag, Berlin.

Clarkson K. L., 1987, New applications of random sampling in computational geometry, *Discrete Comput. Geom.*, 2:195–222.

Clemens D. T., Jacobs D. W., 1991, Space and Time Bounds on Indexing 3-D Models from 2-D Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10).

Collins G. E., 1975, Quantifier elimination for real closed fields by cylindric algebraic decomposition, In *Proc. 2nd GI Conf. on Automata Theory and Formal Languages*, pages 134–183, Berlin, Springer Verlag.

Edelsbrunner H., 1987, *Algorithms in Combinatorial Geometry*, Springer Verlag.

Flynn P. J., Jain A. K., 1992, 3D Object Recognition Using Invariant Feature Indexing of Interpretation Tables, *Computer Vision, Graphics, and Image Processing*, 55(2):119–129.

Grimson E., 1990, *Object Recognition by Computer*, MIT Press, Cambridge, MA.

Horn B. K. P., 1985, *Robot Vision*, MIT Press, Cambridge, Mass.

Ikeuchi K., Kanade T., 1988, Applying sensor models to automatic generation of object recognition programs, In *Proceedings of the International Conference on Computer Vision*, pages 228–237, Tarpon Springs, FL.

Jacobs D. W., 1988, The Use of Grouping in Visual Object Recognition, A.I. Technical Report No. 1023, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Jacobson N., 1974, *Basic Algebra I*, W. H. Freeman and Company, San Francisco.

Korn M. R., Dyer C. R., 1987, 3D Multiview Object Representations for Model-Based Object Recognition, *Pattern Recognition*, 20(1):91–103.

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 Scie nce, New York, NY.

Lamdan Y., Schwartz J., Wolfson H., 1990, Affine invariant model-based object recognition, *IEEE Transactions on Robotics and Automation*, vol.6, no.5:578–89.

Moses Y., Ullman S., 1991, Limitations of non model-based recognition schemes, Technical Report 1301, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Mundy J. L., Zisserman A., 1992, *Geometric Invariance in Computer Vision*, MIT Press, Cambridge, MA.

Preparata F. P., Shamos M. I., 1985, *Computational Geometry–An Introduction*, Springer Verlag, Berlin.

Samet H., 1990, *The Design and Analysis of Spatial Data Structures*, Addison Wesley.

Schwartz J. T., Sharir M., 1983, On the "piano-movers" problem. II: General techniques for computing topological properties of real algebraic manifolds., *Adv. in Appl. Math*, 4:298–351.

Stein F., Medioni G., 1991, Structural Hashing: Efficient Three Dimensional Object Recognition, In *Proceedings Image Understanding Workshop.*

Tarski A., 1948, *A decision method for elementary algebra and geometry*, U. of California Press.

Ullman S., Basri R., 1989, Recognition by Linear Combinations of Models, A.I. Memo No. 1152, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Ullman S., 1979, *The Interpretation of Visual Motion*, MIT Press, Cambridge and London.