MEMO #93–07
June 1993

Geometric Matching in Computer Vision
Algorithms and Open Problems

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

Abstract

In this paper, I review and discuss recent advances in the development of algorithms for solving the kinds of geometric matching problems that occur in computer vision, in particular under bounded error models. I examine both the one-model case (recognition) and the many-model case (indexing), and I discuss some open problems and directions for research.

1 Introduction

Visual object recognition is concerned with finding instances of objects in images. There are many aspects to this task, but one of the most important aspects is geometry. That is, given a geometric model of the shape of an object, a recognition system is to determine whether some projection of that shape is (partially) present in an image.

Visual object recognition has important applications in robotics, communications, and information retrieval. Because such applications usually have to cope with large amounts of image data, and since many of the problems in visual object recognition have a "combinatorial flavor", developing efficient algorithms for recognition problems is an important task.

Here, we wish to summarize recent important advances in the development of efficient algorithms for visual object recognition, and point out important areas of future research.

2 Bounded Error Recognition

Definition In order to study the problem of visual object recognition from an algorithmic point of view, we need to formalize it. More precisely, we have to decide on representations of images and object models, we have to describe the imaging process, and (to account for inevitable real-world variations) we need some error model.

The rationale for the approach described below is that we consider an object to consist of a collection of "features" (localizable visual properties like edges, corners, parts, surface markings), and that the more of those features can be identified in an image, the more certain we are that the corresponding object is actually present in the image (see Figure 1).

A very common formalization of this idea is bounded error recognition with point features.[3,16] In bounded error recognition with point features, a **model** is collections $\mathcal{M} = \{m_1, \ldots, m_{n_\mathcal{M}}$ of points in $\mathbb{R}^3$ and an **image** is a collection $\mathcal{B} = \{b_1, \ldots, b_{n_\mathcal{B}}\}$ of points in $\mathbb{R}^2$. The imaging process is described as a 3D rigid body **transformation** T (also commonly called a **pose**) followed by a **camera model** P, usually either central projection (pinhole camera) or "weak-perspective" (orthographic projection followed by a change of scale–a commonly used approximation to central projection). The **error model** is that of bounded error, that is, we assume that the position of features in the image may differ from their ideal positions by at most a small amount ϵ.

Given a particular transformation T, some model features will be mapped to within the given error bound of some image feature. We say that there is a **correspondence** between such model and image features under transformation T.

The set of all correspondences for a given transformation T form a bipartite graph;[9] we say that this bipartite graph is **c**onsistent with the transformation T. On this bipartite graph, we define an **evaluation function**, which assigns a quality of match value to the transformation T based on the graph and the geometric properties of the features participating in the match (see Figure 2). Usually, this evaluation function will compute the maximum number of features that can be brought into correspondence without using any model feature or image feature twice, i.e., it will compute the size of the maximal bipartite match.

A bounded error recognition algorithm has to find a maximal bipartite matching between model and image features that is consistent with some transformation T.

More formally, a bounded error recognition algorithm has to find a maximal set $\{(m_{\alpha_i}, b_{\beta_i}) : i = 1, \ldots, n_{\mathrm{match}}\}$ such that $\alpha_i \neq \alpha_j$ if $i \neq j$ and $\beta_i \neq \beta_j$ if $i \neq j$, and there exists a $T \in \mathcal{T}$ such that for all $i = 1, \ldots, n_{\mathrm{match}}$ the error bounds are satisfied:

$$\|P(Tm_{\alpha_i}) - b_{\beta_i}\|_2 \leq \epsilon \tag{1}$$

This problem is significantly harder than the usual maximal bipartite matching problem because of the dependence on T: potentially, a different matching problem needs to be solved for each $T \in \mathcal{T}$.

Variations Of particular practical importance is the consideration of different sets of transformations $\mathcal{T}$. Because $\mathcal{T}$ usually has a rich structure (group structure, topological structure) associated with it, it is usually called **transformation space**. Commonly used transformation spaces, their properties, and their applications can be found in Table 1 (note that several of these transformation spaces apply to 2D models and that our definition of model changes accordingly).

Another important variation is to substitute polygonal error bounds for the circular error bounds used in Equation 1. That is, we describe error bounds as a collection of unit vectors e_i and associated bounds ϵ_i and require that:

$$e_i \cdot (P(Tm_{\alpha_i}) - b_{\beta_i}) \leq \epsilon_i \tag{2}$$

For many transformation spaces, using polygonal error bounds means that all the geometric computations that need to be carried out reduce to operations with linear inequalities, resulting in a considerable simplification of recognition algorithms;[3] see also below.

There are many other possible variations on the basic paradigm of bounded error recognition; for example, we might also consider features that are more complex than point features (lines, curves, regions) or consider more complicated error models.

Approximations A particularly useful concept for describing approximations to the recognition problem is that of a **weak recognition algorithm** (defined in analogy to a weak membership problem[20]). Essentially, rather than insisting that the error bounds ϵ be satisfied exactly, we introduce a second parameter δ (which is part of the input) and allow the recognition algorithm to arbitrarily consider errors between ϵ and $\epsilon + \delta$ as either satisfying or violating the error bounds. Requiring only weak recognition (for arbitrary δ) makes a number of efficient approximations to the recognition possible (see RAST and view-based indexing below).

Another common and useful approximation is not to insist on a maximal bipartite matching, but to use the number of model features that correspond to some image feature as an approximation; this approximation can be justified for many kinds of point features.[21]

Constraint Sets A crucial observation,[3] underlying most of the analysis and development of efficient recognition algorithms, is that an individual correspondence between a model feature and an image feature implies a set of transformations that is compatible with that correspondence, the **constraint set**. More formally, given model point m and image point b, the constraint set is defined as:

$$\mathcal{C}(m, b) = \{T : \|P(Tm) - b\| \leq \epsilon\} \tag{3}$$

Furthermore, to test whether a particular bipartite matching $\{(m_{\alpha_i}, b_{\beta_i})\}$ is consistent with some transformation reduces to testing whether the intersection of the corresponding constraint sets is non-empty:

$$\bigcap_i C(m_{\alpha_i}, b_{\beta_i}) \neq \emptyset \tag{4}$$

If we use polygonal error bounds, then in the case of several important transformation spaces (T2, TRS2, A2, and A3 in Table 1), constraint sets turn out to be just polyhedra themselves. In that case, the test in Equation 4 can be carried out using linear programming.[3] Furthermore, other transformation spaces (TR2 and TRS3) can be viewed as the intersection of transformation spaces in which a linear solution is possible (TRS2 and A3) with a manifold defined by a fixed set of quadratic constraints. This lets us essentially solve these non-linear cases by solving the linear case in a larger space and rejecting solutions that do not satisfy the quadratic constraints.

3 Recognition Algorithms

Recognition algorithms can be divided into two major classes: correspondence-based algorithms and transformation-space based algorithms. Correspondence-based algorithms work by organizing the search for a maximal solution around considering pairs of model and image features. Transformation-space based algorithm, on the other hand, organize the search around considering different regions of transformation space.

3.1 Transformation-Space Based Algorithms

Sampling Probably the simplest and oldest recognition algorithm is based on sampling. That is, transformation space is discretized, and each transformation is tried and evaluated. Earliest examples of this method are correlation methods;[12] more recently, the method has been tried with evaluation functions like those used in bounded error recognition above.[8] Recognition algorithms based on sampling are generally weak (in the sense defined above), with δ being directly related to the degree of discretization of transformation space. Their complexity is therefore proportional to δ^D, where D is the dimensionality of transformation space.

Sweep The Critical Point Sampling (CPS) algorithm[9] is essentially a sweep of the arrangement[13] formed by the constraint sets $\mathcal{C}(m_i, b_j)$, for all i and j (for a related approach, see[1]). Each cell of this arrangement describes (at most) one bipartite graph of correspondences between model and image features. By solving the maximal bipartite matching problem for each of these cells, we can find the overall best match. Note that the CPS algorithm was the first known polynomial time recognition algorithm.

Note that even in the simplest case case of arrangements of constraint sets $\mathcal{C}$ that are halfspaces in a D-dimensional transformation space, the worst-case number of cells that needs to be tested is $\Theta((n_{\mathcal{M}} n_{\mathcal{B}})^D)$. For circular error bounds or non-linear transformation spaces (TR2, TRS3), the complexity for the worst case is even higher. Sweep methods also have high space complexities.

RAST The high complexity and overhead of sweep-based methods and search methods based on linear programming (see below) is due to two factors. (1) The arrangement of constraint sets is explored fully, even though in practice, there are often large regions of transformation space that can easily be eliminated from further consideration, even though they contain many cells. (2) Intersections of constraint sets are computed exactly, at considerable cost.

The RAST algorithm[6] (Recognition by Adaptive Subdivision of Transformation Space) attempts to remedy both these problems. Essentially, the method constructs a spatial subdivision (more concretely, kD-trie,[29] a data structure similar to a kD-tree) data structure in transformation space. For many evaluation functions, the maximum solution over each region of the trie can be evaluated, and only regions that potentially contain a solution better than the one found already need to be subdivided further.

The advantage of the RAST algorithm is that it replaces the costly general linear programming problem with a simpler problem of determining the intersections between individual constraint polyhedra and a hyperrectangle. Furthermore, large regions of transformation space are never subdivided very deeply at all. Finally, by exploring the trie carfully, the space requirements can be kept small: only a single path from the root of the trie to the current leaf needs to be kept.

If the depth of the spatial subdivision in the RAST algorithm is limited to d, it forms a weak recognition algorithm in the above sense with $\delta \propto e^{-d}$. In that case, it is guaranteed to complete in polynomial worst-case time. Otherwise, we can make a heuristic argument that the algorithm runs in polynomial time in the average case. At least for the cases of T2 and TRS2, the RAST algorithm probably currently represents the most efficient recognition algorithm for the problem of recognition from unlabeled point features.

3.2 Correspondence-Based Algorithms

Search The basic idea behind using depth-first search for object recognition is to start with an empty matching between model features and image features and non-deterministically add correspondences to it until it becomes inconsistent; the maximal matching obtained in this way is a solution to the recognition problem.

Depth-first search has been used by a number of researchers,[19,3,2] often combined with heuristic methods for speeding up the search. While such algorithms have exponential worst case and (in some cases) exponential average case running times,[17] they are very popular because they are easy to implement and can take advantage of diverse kinds of geometric and non-geometric constraints. If we choose our problems and features carefully and if we incorporate heuristics, they can offer good performance in some practical situations.

These exponential time search procedures can be modified to run in polynomial time.[5] Like the sweep-based methods, the resulting algorithm also visits every cell in the arrangement generated by the constraint sets. However, the amount of space required is less, and the order of exploration may allow existing heuristics and pruning techniques to be incorporated more easily.

Alignment Alignment[14,23] is a special kind of depth-first search algorithm. Alignment picks a minimum number of correspondences between model and image features necessary to determine a transformation uniquely in the error free case. This transformation is then used to transform the model and to evaluate the match between the model and the image.

Usually, this minimum number of correspondences is half the dimension D of transformation space (i.e., 2 in the case of TRS2 and 3 in the case of TRS3). Since alignment algorithms try most such correspondences, the complexity of an alignment algorithm is $\Theta((n_{\mathcal{M}} n_{\mathcal{B}})^{\lceil \frac{D}{2} \rceil} E)$, where E is the time required to evaluate a single match.

Alignment does not solve the bounded error recognition problem exactly. The reason is that there may be error on the location of the points used to determine the initial pose. However, an alignment within some error bound ϵ usually implies the existence of a bounded error match with some error bound $c\epsilon$, where c is a small constant factor[22,4,18] c. Note that alignment is not a weak algorithm in the sense described above, since we cannot choose c

to be arbitrarily small. Hence, alignment solves a genuinely different (though closely related) problem from bounded error recognition.

3.3 Open Problems

There are three key open problems in bounded error recognition. (1) We would like to know whether the complexity of alignment constitutes a lower bound on the worst case of recognition; even an answer for the case of $\epsilon = 0$ would be interesting. (2) Even if we cannot necessarily improve the worst-case performance of recognition algorithms, we would like to obtain average case algorithms with performance significantly better than that of alignment. (3) All known algorithms have complexity exponential in the dimension of transformation space; however, the interesting case of non-rigid matching represents an infinite-dimensional transformation space with a special structure, and the question arises whether there are efficient (possibly approximate) algorithms for this case as well.

What we have seen is that recognition algorithms are generally based on an exploration of the arrangment of the constraint sets for a particular recognition problem. There are several reasons to believe that better algorithms might exist than those described above. (1) The RAST algorithm already has slightly lower complexity than an alignment algorithm. (2) We are only interested in cells with specific properties (those maximal wrt. our evaluation function). (3) While there are $n_{\mathcal{B}} n_{\mathcal{M}}$ total constraint sets forming the arrangement, these sets are specified by $2n_{\mathcal{B}} + 3n_{\mathcal{M}}$ parameters, meaning that the arrangement is not completely general. (4) In practice, features are distributed highly non-uniformly and error bounds are small relative to the size of the image.

4 Indexing

In the previous section, we considered the *on-line* recognition problem, that is, we assumed that the recognition algorithm was given a model and an image and told to compute the optimal match between them. In practice, we can usually allow a recognition algorithm to perform off-line computation on one or more models before being confronted with the image.

Of particular interest is the case in which a recognition algorithm has to perform recognition from a large model base, that is, where it has to find matching models among a large number N of possible models. This problem is commonly known as the **indexing problem** (often, indexing is considered only an approximate pre-selection step, followed by the application of a recognition algorithm, but the two views of indexing are essentially equivalent).

The primary focus of indexing algorithms is to reduce the time complexity in N. Note that we can trivially achieve linear complexity in N simply by applying a recognition algorithm successively to each model.

As is common (though not necessarily justified) in the analysis of the indexing problem, we will assume for the time being that correspondences between model and image points are known. Then, the indexing problem is the following problem. Given a collection of N ordered sets of K model points $\mathcal{M}^n = \{m_1^n, \ldots, m_K^n\} \subseteq \mathbb{R}^3$ and an ordered set of K image points $\mathcal{B} = \{b_1, \ldots, b_K\} \subseteq \mathbb{R}^2$, determine for which n there exists a transformation such that for all k the error bounds are satisfied:

$$\forall k = 1, \ldots, K \ : \ \|P(Tm_k^n) - b_k\| \leq \epsilon \tag{5}$$

A change of perspective which greatly simplifies the analysis is to view a model $\mathcal{M}$ not as an ordered collection of K points in $\mathbb{R}^3$, but instead as a single point m in $\mathbb{R}^{3K}$ (called **model space**, and, by analogy, consider the image $\mathcal{B}$ as a single point b in $\mathbb{R}^{2K}$ (called **view space**). The bounded error condition from Equation 5 can then be shown to correspond to a metric d in image space, and we can define a predicate match as:

$$\mathsf{match}(m, b) := \exists T \in \mathcal{T} \ : \ d(P(Tm), b) \leq \epsilon \tag{6}$$

What makes the evaluation of this predicate difficult in practice is the existential quantification over T.

4.1 Trivial Transformation Space

To get a better idea of the difficulties of the indexing problem, let us first assume that transformation space is trivial and only consists of the identity transformation: $\mathcal{T} = \{\mathbf{1}\}$. Then, Equation 6 simply turns into:

$$\mathsf{match}(m, b) := d(Pm, b) \leq \epsilon \tag{7}$$

If we precompute Pm, then finding the set of n for which this predicate is satisfied can be seen simply to be a range query problem[28,29] in $\mathbb{R}^{2K}$. Such range query problems can be solved in worst-case time (we assume that the output is small) $O((\log N)^{2K})$ and using space $O(N(\log N)^{2K-1})$ using range trees. In practice, k-D trees are probably a better approach: they require space $O(KN)$ and their average case time complexity is $O(K \log N)$; unfortunately, the worst-case complexity is only guaranteed to be $O(KN^{1-\frac{1}{2K}})$.

4.2 Indexing by Point Location

The actual indexing problem is more complex, since transformation space is non-trivial. However, we can still eliminate the existential quantifier in the above equation as follows. We define the **view set** V_m of the model m under error ϵ as follows (see also the references[11,26]):

$$V_m = \{b \in \mathbb{R}^{2K} : \exists T \in \mathcal{T}\ d(P(Tm), b) \leq \epsilon\} \tag{8}$$

The match predicate defined above then becomes a simple test of set membership:

$$\mathsf{match}(m, b) := b \in V_m \tag{9}$$

It is not difficult to show that view sets V_m are semi-algebraic sets for the error bounds and transformation spaces that interest us; in fact, view sets inherit most of the structure (e.g., dimension, connectedness) of the transformation space used in the recognition problem).

Point location algorithms with asymptotic time complexity $O(\log N)$ in N are known for such sets.[10] With appropriate preprocessing of the arrangment generated by the N view sets V_m, using such algorithms, we can therefore solve the indexing problem in time logarithmic in N, which is optimal. However, a significant drawback of such approaches is that the only known bound on their space requirement in K is doubly exponential, and it is doubtful whether a straightforward implementation of such algorithms is practical.

4.3 View-Based Approximation and Canonicalization

Since we have better, more practical algorithms available for the problem of range queries than for the problem of point location in general collections of (semi-)algebraic sets, it would be nice if we could reduce the indexing problem to a range query problem even in the case of nontrivial transformation spaces.

There are two basic approaches. The first is the method of **invariants**,[27] which basically seeks to identify functions that are constant over the view-set for $\epsilon = 0$. However, there are known serious limitations to the application of invariants for the recognition of arbitrary objects.[7,11,26]

An alternative approach is based on the **view-based approximation**.[24,25,4,11,30,15] Here, we approximate the view-set V_m as a collection of balls. The method derives its name from the fact that at the center of each such ball is a different view of the object, for example, obtained by taking images of the object from different directions.

If we want to cover V_m with a finite set of balls of bounded size, it is necessary that V_m is compact. Unfortunately, the translational component of transformation space gives rise to a non-compact view set. We can avoid this problem by simply bounding translations. In that case, we need δ^{-6} balls of diameter δ to cover V_m.

It is significantly more efficient to use a technique called **canonicalization**, in which we perform a 2D alignment (see above) with a fixed set of points to "factor out" 2D equiform transformations (translation, rotation, and scale) before indexing. It can be shown that after canonicalization, only 2 parameters remain that describe the different possible viewing transformations (these parameters are often associated with the surface of the so called **viewing sphere**). Consequently, view sets are two-dimensional surfaces and number of views needed to cover them can be shown to grow as[4,11] δ^{-2}. Note that since canonicalization involves an alignment, the above comments about the nature of the alignment approximation to bounded error recognition apply here as well.

Therefore, using the view-based approximation and canonicalization, we can solve the 3D indexing problem for N objects approximately as a range query problem on $\delta^{-2}N$ views.

4.4 Open Problems

"Geometric Exponentiality" Even if we are given correspondences, all known indexing algorithms, including those given above, that achieve poly-logarithmic time complexity in N have have exponential requirements and/or time complexity in K. As we can see, this property of indexing algorithms is closely linked to the complexity of known range query and point location algorithms. The asymptotically best known indexing algorithm with polynomial complexity in K is using the view-based approximation and canonicalization to represent N objects by $\delta^{-2}N$ views, and solving the associated range query problem using k-D trees.

"Combinatorial Exponentiality" Above, we were assuming that correspondences between model and image features were known. In practice, such correspondences are difficult to extract reliably from images. If we take the "naive" approach towards handling the absence of correspondences and simply try all correspondences, we end up with an indexing algorithm that has space and/or time complexity exponential in K, even if we solve the problem of geometric exponentiality. The recognition algorithms mentioned in Section 3 can operate in polynomial time without such correspondence information. It would be interesting to see whether similar techniques can be incorporated into indexing algorithms.

5 Conclusions

We have seen that problems in visual object recognition are closely related to several fundamental problems in combinatorial and computational geometry. In particular, on-line recognition can be viewed as the problem of searching the arrangement generated by a collection of constraint sets, while recognition with pre-processing (indexing) and assuming correspondences can be viewed as a high-dimensional point-location or range query problem.

However, the standard methods for solving these problems are not entirely satisfactory for actual recognition problems, since they do not appear to take full advantage of the constraints inherent in the problems, and since they are also not optimized for the kind of average case that is common in actual recognition situations (e.g., small ϵ). It is to be hoped that this survey will encourage researchers in combinatorial and geometric algorithms to develop better algorithms. Even negative results (i.e., tight, inefficient lower bounds) are of great interest, since they would indicate that the problem itself needs to be approached differently.

References

[1] Helmut Alt, Kurt Mehlhorn, Huber Wagener, and Emo Welzl. Congruence, Similarity, and Symmetries of Geometric Objects. *Discrete and Computational Geometry*, 1988.

[2] N. Ayache and O. D. Faugeras. 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, 1986.

[3] Henry S. Baird. *Model-Based Image Matching Using Location.* MIT Press, Cambridge, MA, 1985.

[4] Thomas M. Breuel. Indexing for Visual Recognition from a Large Model Base. A.I. Memo No. 1108, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1990. (see also DARPA IU Workshop 1989).

[5] Thomas M. Breuel. An Efficient Correspondence Based Algorithm for 2D and 3D Model Based Recognition. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1991.

[6] Thomas M. Breuel. Fast Recognition using Adaptive Subdivisions of Transformation Space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1992.

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

[8] Todd A. Cass. Robust Parallel Computation of 2D Model-Based Recognition. In *Proceedings Image Understanding Workshop*, Boston, MA, April 1988. Morgan and Kaufman, San Mateo, CA.

[9] Todd A. Cass. Feature Matching for Object Localization in the Presence of Uncertainty. In *Proceedings of the International Conference on Computer Vision*, Osaka, Japan, December 1990. IEEE, Washington, DC.

[10] B. Chazelle. 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, 1985.

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

[12] Richard O. Duda and Peter E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[13] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer Verlag, 1987.

[14] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

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

[16] Eric Grimson. *Object Recognition by Computer.* MIT Press, Cambridge, MA, 1990.

[17] W. Eric L. Grimson. The Combinatorics of Heuristic Search Termination for Object Recognition in Cluttered Environments. A.I. Memo No. 1111, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1989.

[18] W. Eric L. Grimson, Daniel P. Huttenlocher, and David W. Jacobs. On the sensitivity of geometric hashing. Technical Report A.I. Memo 1250, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1990.

[19] W. Eric L. Grimson and Tomas Lozano-Perez. Model-Based Recognition and Localization From Sparse Range or Tactile Data. Technical Report A.I. Memo 738, MIT, August 1983.

[20] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer Verlag, Heidelberg, 1988.

[21] D. P. Huttenlocher and T. A. Cass. Measuring the quality of hypotheses in model-based recognition. In *Proceedings of the European Conference on Computer Vision*, 1992.

[22] Daniel P. Huttenlocher. *Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image.* PhD thesis, Massachusetts Institute of Technology, 1989.

[23] Daniel P. Huttenlocher and Shimon Ullman. Object Recognition Using Alignment. In *Proceedings of the International Conference on Computer Vision*, pages 102–111, London, England, June 1987. IEEE, Washington, DC.

[24] Matthew R. Korn and Charles R. Dyer. 3D Multiview Object Representations for Model-Based Object Recognition. *Pattern Recognition*, 20(1):91–103, 1987.

[25] Y. Lamdan and H.-J. Wolfson. 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, 1988.

[26] Yael Moses and Shimon Ullman. Limitations of non model-based recognition schemes. Technical Report 1301, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1991.

[27] Joseph L. Mundy and Andrew Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, MA, 1992.

[28] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry–An Introduction*. Springer Verlag, Berlin, 1985.

[29] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, 1990.

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

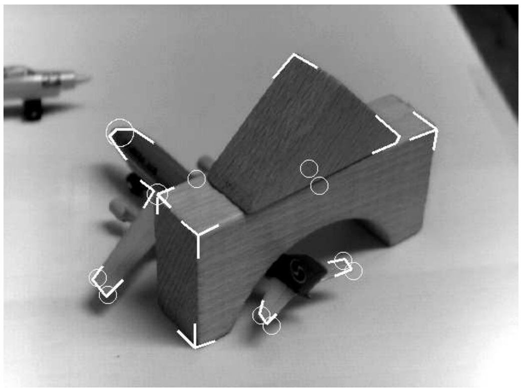

Figure 1: Example of features and error bounds. White lines illustrate features commonly returned by low-level feature extraction modules. Circles indicate error bounds for locations where features are predicted by a 3D model of the airliner. Note that there are image features that do not match model features, and that there are model features that do not match image features.

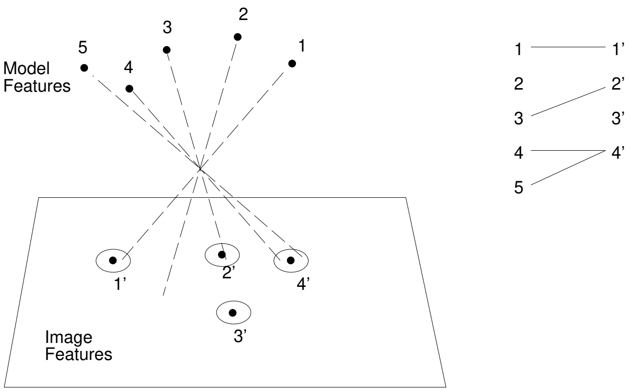

Figure 2: A formalization of the recognition problem under bounded error. Shown is a model of 5 points for some particular pose (transformation T). Its features are projected via central projection into the image and compared with the image features, each of which is associated with a circular error bound. The bipartite graph that represents this match is shown on the right.

	Problem	Dim	Structure	Applications	Comments
T2	2D translation	2	$\mathbb{R}^2$	OCR, line drawings, speech recognition, primitive for TR2/TRS2	simplest, most efficient case, transformation space is easy to visualize
TR2	2D translation and rotation (equiform transformations)	3	$\mathbb{R}^2 \otimes S_1$	industrial parts recognition, line drawings	can be reduced to T2 by sampling rotations
TRS2	2D translation, rotation, and scale	4	$\mathbb{R}^4$	industrial parts recognition, line drawings; solving TR2	linear relationship between error bounds and subsets of transformation space[3]
A2	2D linear or affine transformations	6	$\mathbb{R}^6$	laminar objects in 3-space can be modeled exactly as 2D objects under affine transformations[23]	linear relationship between error bounds and subsets of transformation space[3]
TRS3	3D translation, rotation, and scale	7	$\mathbb{R}^3 \otimes \mathbb{R}^+ \otimes SO(3)$	recognition of 3D rigid objects	
A3	3D linear or affine transformations	12	$\mathbb{R}^{12}$	solving TRS3	
	translation, rotation, scale, and smooth non-rigid deformation	∞		recognition of natural objects	no known efficient recognition algorithms

Table 1: Different kinds of transformation spaces useful for visual object recognition.