Efficiently Using Invariant Theory and Grouping Information for Model-based Matching

Peter C. Wayner

TR 90-1170
November 1990

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

Efficiently Using Invariant Theory
and
Grouping Information for Model-based Matching

Peter C. Wayner
Computer Science Department
Cornell University
Ithaca NY 14853

Abstract

This paper presents a method for efficiently maintaining and searching a database of three-dimensional models so they can be reliably recognized from arbitrary two-dimensional projections in the presence of noise and occlusion. The core of the process is the *topologically-defined network of invariants* which breaks three-dimensional models down into small. local groups of features and indexes these groups using translation. rotation, scaling and orthographic projection invariant functions. The network encodes the geometrical relationships between these groups so that grouping information can be used to increase the speed of matching.

1. Introduction

The problem of matching a large collection of models to an image can be broken down into two components: 1) finding a group of features in the image that are likely to be one object and 2) finding the best match for this group in the model-base. The first step has taken the name of *grouping* and recent work by [HW], [L] and [J] among others have produced algorithms that produce experimentally valid groups in low-order polynomial time. This paper approaches the second step by creating an *topologically-defined network of invariants* which acts as an index to the collections of models that allows the parts to be lookuped in the collection from any pose and in spite of a certain measure of occlusion.

The core of this network is a set of functions that maps a set of points to a tuple of values. The important detail is that these functions are invariant under any combination of translation, rotation and scaling followed by an orthographic projection. This network encodes the collection by breaking each model up into small clusters of features, applying the invariant function to each cluster and storing the tuples returned in a database. A cluster of points in the image can be identified by applying the invariant function to it and finding all clusters in the network which return the same or similar values.

This system has been designed to integrate easily with a grouping methods which identifies features in an image which are likely to correspond to the same object. (E.G. [HW], [J], or [L].) These groups are then broken into the small parts which are looked up with the indexing function. The grouping method provides the geometrical information about the relationship between the small parts and this information corresponds to the structure of the invariant data base.

The major features of this system are:

1. *Works with Three-Dimensional Objects.* This system uses either invariants of three-dimensional sets of points. The structure of the network also allows invariants to three-dimensional sets of points to be constructed from two sets of two-dimensional sets of points. This allows three-dimensional objects to be recognized easily.

2. *Works with One Two-Dimensional Image.* The system requires only the use of one, two-dimensional image. Stereopsis and range data are not necessary.

3. *No Need to Store Descriptions of Multiple Views.* Many systems for calculating the appearance of an object form descriptions of all potential views such as aspect graphs, producing complex descriptions of the objects with computationally expensive algorithms. This method of reducing an object into topologically defined networks of invariants is simple to implement and compute.

4. *Resistant to Missing Features* The clusters of features are deliberately kept small (4 points) to ensure that some cluster of points is still visible in the image from any pose. Only one cluster is necessary to begin identification.

5. *Uses Grouping to Reduce Complexity.* This method is well-integrated with a robust method for identifying groups in an image. This substantially reduces the complexity of the matching process. Work by [CJ] shows that grouping is necessary for fast recognition.

6. *Expandability-* The set of indexing functions can be expanded using other invariant functions. This system can combine information from different indexing functions on the same object in the same image.

This paper begins with description of how transformation-invariant functions have been used in other work. The basic mathematics of three different invariants will be summarized in the next section. After this. the paper will describe a technique for creating a topologically defined network of invariants by decomposing a model into small clusters of features so they can be encoded as a network of invariant indices. Finally, experimental results of using the system will be presented.

The current system discussed in the paper will be limited to polygonal models, although there are many cases where the work can easily be extended to handle curved objects. In these cases, suggestions for ways for extentions will be included.

2. Prior Work

The problems of indexing objects has been examined by a number of different researchers in several different contexts. Forsythe, Mundy. Zisserman and Brown [FMZB] give a detailed explication of the mathematics of two-dimensional planar, projective invariants and show how several different invariants could be constructed and successfully used to identify two-dimensional forms. Their method relies upon finding either two second-order, co-planar curves (conic sections) or four co-planar, parallel lines in an image. They describe functions of these two geometric arrangements which are invariant under projective transformation. Experimental verification is provided by demonstrating the method on cards with two conics printed upon them and several objects, like scissors. which have a silouette with two elliptical figures.

Hopcroft and Huttenlocher [HH] described an algorithm for recognizing sets of points which may have been changed by an affine transformation. The algorithm relies upon defining sets of affine invariants which uniquely describe the locations of the points. These algorithms, and the several others described here using affine coordinates, are useful for machine vision because if co-planar points are rotated in three-space and then projected orthographically onto a plane, this corresponds to an affine transformation of the orthographic image of the points.

The problem of recognition using affine invariants was also addressed in earlier work by Huttenlocher and Ullman in [HU]. The paper showed how to align three-points in an image with three-points in a model and compare other features for recognition. In [TM], Thompson and Mundy also discuss using affine coordinates to align objects with the image.

An affine invariant hashing scheme first described by Lamdan and Wolfson in [LW] and later optimized by Mauro Costa. Robert Haralick and Linda Shapiro in [CHS] also relies upon the mathematical structure of the affine plane. Their method computes the affine coordinates of all features in a two-dimensional model relative to three features in the model and then hashes the values in a two-dimensional table. It repeats this process for every possible triple of features. Recognition is achieved by finding a set of features in an image. choosing a subset of three features, computing the affine coordinates relative to this subset and then checking the appropriate entries in the hash table for matches.

In [SM1], Fridtjof Stein and Gérard Medioni. propose a hashing system based upon an approximation to curvature called super-structures. which are converted into a grey-code-like number and stored in a table for lookup. This work was later extend to three-dimensional objects in [SM2] which could recognize objects using range data.

Clemens and Jacobs [CJ] discuss some of the theoretical constraints to indexing a collection of three-dimensional models for recognition using two-dimensional images. They show that grouping is a necessity if the system is to work sufficiently quickly. Grimson [Grimson] also considers some of the complexity issues of indexing.

Bruel [Br] also attacks the problem of indexing in a large data base and describes a method for minimizing the number of stored views and creating an indexing function which can use a simple bitwise AND operation for search.

3. Grouping Methods

The method described in this paper looks-up small, local collections of features from an image in a large model-base. Although it is possible to simply iterate through all possible combinations of features in an image and use the network of invariants successfully, the method is intended to work in concert with a grouping algorithm like the one in [HW], [L] or [J]. In each of these algorithms, properties like parallelism or convexity are used to identify sets of features which might belong to the same object.

Jacobs [J] considers pairs of features and determines whether or not they are likely to be from the same objects and then given this result, proceeds with recognition. Lowe's [L] method's uses properties like parallelism or proximity of edge-segments to produce larger groups which are then used by the recognition step. The structure of the topologically-defined network of invariants can be modified to include clusters that might be returned from this grouping method.

The technique in [HW] groups features together if they could be part of a convex object and this grouping principle has the advantage of being both invariant under projection and locally computable. Also, Jacobs has shown in [J] that this convexity is unlikely to occur at random in images. Although objects are often non-convex, they are usually composed of convex sub-pieces, and the algorithm reliably will identify these. [HW] uses a Delaunay triangulation to define the neighborhood around each feature and tries to string the features together into a path with a constant turning direction. The algorithm produces a $O(n)$ groups in $O(n \log n)$ time where n is the number of features in the image. This efficient bound is mainly due to the fact that convexity is locally computable. The Delaunay triangulation computes this neighborhood quickly. A more elaborate summary follows:

The process begins after an edge detector identifies all of the high-contrast regions and primatives are fitted to the edges. The method uses straight-lines as the sole primative, but there is no reason why it cannot be extended to second-order curves as well. These primatives become nodes in a graph and arcs are added between local neighbors which could be part of convex objects.

More specifically, each primative (line) has two endpoints. A local neighborhood of each of these primatives is defined by computing the Delaunay triangulation of the endpoints. Any pair of endpoints from different image primatives which are joined by the Delaunay triangulation are said to be *neighbors*. This relationship is converted into a *local neighborhood graph* which has one node for each image primative and one arc for each pairing in the triangulation. (The word *arc* will be used in this paper to refer to graphs and the word *edges* to refer to regions of high-gradiant in an image. In this particular case, each arc is a symbol for an edge in the image.) Since each node corresponds to an image primative with two endpoints, each arc in the graph could correspond to a triangulation line pairing one of the four possible pairs. The arcs are marked according to which of the four possible pairings they represent. This information is later used to define consistent paths that enter one endpoint of an edge and leave the other.

The local neighborhood graph is converted into a *local convexity graph* by removing all arcs representing pairings that could not belong to the same convex object. Figure 1 shows six images edges in two configurations. In the first, all triangulation lines are drawn between the endpoints. This represents the local neighborhood graph. The second only has the triangulation lines drawn which join two images edges that could be part of the same convex object. This, in effect, shows what the local convexity graph is.

A path in the graph can correspond to a convex form in the image if every arc entering a node through one endpoint leaves through an arc which leaves through the other endpoint of the primative. The algorithm enumerates a useable set of these paths and cycles by using a choice function to make local decisions at each node. This reduces the complexity of the final ordering to be linearly proportional to the number of image edges. Some choices which have been used successfully are ones which either choose the nearest neighbor of the most-inline neighbor.

Figure 2 shows some of the groups detected in this very poor image of the pentagon. The grouping method successfully finds many different parts of the image which correspond to individual convex objects. These groups make ideal starting points for the matching algorithm described here because they are local sets of features which are quite likely to match sub-parts of the pentagon.

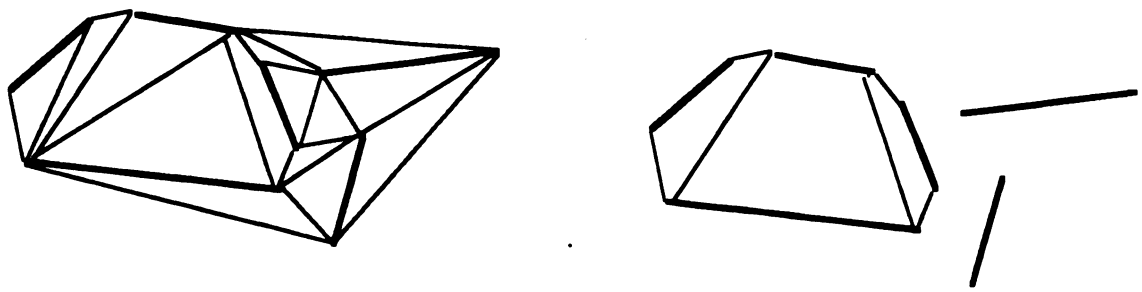


Figure 1: Two intermediate views of the (a) local neighborhood graph and (b) of the local convexity graph. Note that arcs from the triangulation that join two edges in a way that couldn't conform to a convex object are missing from (b).

4. Invariants and Indices

The grouping techniques will return clusters of image features that are likely to be part of the object. The task is to identify which models might have subparts which might match the clusters. The approach is to construct a database containing all possible small clusters which are subsets of the models. The invariant functions which follow are used to create a pose-independent index to this data-base. They are all applied to clusters of four features. This keeps the complexity of the data-base low and also introduces a measure of resistance to occluded features.

These next several sections will describe three different indexing functions based upon invariant theory. These functions complement each other well and at least one is applicable to any set of four features. In some cases, the limitations of some functions are the strengths of others. They are:

1. f_A: Uses affine coordinates and will be useful for sets of features which lie in a plane.

2. f_O: This indexes three orthogonal vectors which represent four points in three space. It relies upon the basic invariants of three-dimensional objects under rotation.

3. f_S: This version uses matrix theory of singular values to classify three non-orthogonal vectors.

In each of the cases, the paper will analyze the sensitivities to error and the distribution of index values over the domain.

The structure of the network of invariants can also be extended to use other invariant functions. They will not be specificially discussed or implemented here, but their potential will be noted. The work by Forsythe et. al. [FMZB] notes that a function of two co-planar conic segments (essentially ellipses) can be constructed which is invariant under projective transformation. The action of the projective group models the effects of a camera quite well and provides an efficient way to index into the space of objects. This method. though, relies upon finding objects with two easily detectable conics which are also co-planar.

4.1 Co-Planar Points– The Affine Case.

As the section on previous work has made clear, one section of invariant theory which has proven to be useful to problems in recognition is the structure of the affine plane. If a camera is modeled by orthographic projection, then a three-dimensional similarity transformation applied to co-planar points is equivalent to a two-dimensional affine transformation of the plane. In this case, all of the invariants of the two dimensional affine plane can be used to recognize co-planar points translating, rotating and scaling in three-space. This approximation of the camera action works best if the distance from the camera to the object is either much greater than the depth of the object and/or if the camera has a tele-photo lens. In the affine trasformation, parallel lines are preserved so there is no bending together associated with lines heading off to the horizon. Train tracks heading to the horizon are one good example of an image which breaks the orthographic projection. The mathematics of the affine plane are the basis for most of the work described and left unproven here. Curious readers can examine [E] or [HH] which gives a good summary of many of the properties.

One of the classical theorems of invariant theory states that all functions of a two-dimensional sets of points which remain invariant under affine transformation can be expressed in terms of ratios of the area of triangles. ([E], pg 402) A simple example of this can be created from four co-planar points a, b, c and d, Let $\triangle a, b, c$ represents the area of the triangle with the corners a, b and c. The function:

$$f_A(a, b, c, d) = (\frac{\triangle a, b, c}{\triangle b, c, d}, \frac{\triangle a, b, d}{\triangle b, c, d}) \tag{1}$$

is unchanged as the points undergo any two-dimensional affine transformation and consequentally is also unchanged when the points undergo a three-dimensional similarity transformation. Although, f_A is defined as a pair of numbers here, it is better to think of it as a pair of ranges because noise and error introduce fluctuation. The next section introduces the methods for computing the bounds of the ranges.

f_A is an indexing function for a small part of an object and can be constructed from sets of four co-planar points. If four features are found in an image, corresponding models in the model base can be found simply computing f_A for these points and looking for a set of four features in the model base with similar values. After the error properties of this particular function and the details of the other functions are described in the next section, the rest of the paper will concentrate on how to efficiently segment the models into subsets of points so that the information from the grouping method can be effectively used.

4.2 Error Bounds and Distribution Functions of the 2-D Affine Invariants

While invariant functions might make an excellent indexing function in theory, there are two properties which govern their practical value for model-based vision. First, if the ratios are highly unstable or chaotic, position error from sensing data will make them un-useable. Second, if all of the index values are clustered in a region smaller than the error bounds, then every set of four points could potentially match any other set of points in the model-base. Therefore, it is desirable that the error-bounds and the distribution of points be examined to anticipate problems.

To estimate the effects of error on the affine-index function, consider the diagram in Figure 3 which shows a typical pair of triangles constructed from four points. Note that one side of the triangle is shared. Let this be the base of both triangles. Since the area of a triangle is just one half of the base times the height, the ratio of the area becomes:

$$\frac{\frac{1}{2} base \cdot height_a}{\frac{1}{2} base \cdot height_d}$$

where $height_a$ is the perpendicular distance of point a from the base. The length of the base cancels out. Any error in the position of the points a and d govern their perpendicular distance from the base line. If maximum sensing error for each point is ϵ then the error bounds become:

$$\frac{height_a - 2\epsilon}{height_d + 2\epsilon} < \frac{height_a}{height_d} < \frac{height_a + 2\epsilon}{height_d - 2\epsilon} \tag{2}$$

In practice, the bounds on the error can be calculated simultaneously with the index function itself. When the table is checked, all index values which fall within the range can selected. In practice, most cases will not vary much beyond this value.

Estimating the distribution of the values of the index function can use the same construction. First assume that a, b, c and d occur randomly. If b and c form the shared base of the triangle, the distribution depends upon the relative distances of the other two points. a and d. If the four points are randomly located throughout the entire infinite plane, then all ratios are equally likely.

This assumption of infinity, though, is deceptive and doesn't match with practice. Most digital cameras computers only have resolutions of 512×512 and so the distances from the base-line in these images will only fall within a certain range. To visualize this, construct a two-dimensional grid of n by n points representing the possible pairs of perpendicular distances of a and d to the base line. Counting the points using good combinutorical arguments is possible, but a continuous approximation will be substituted here for simplicity. The question is how many pairs lead the same ratio. This can be approximated by the length of a line drawn through the origin of a square at the correct angle. The slope of the line is equal to the ratio of the area of the triangles. This length varies from n to $n\sqrt{2}$ back to n. Figure 4 shows the normalized distribution which is constructed of $secant(arctan^{-1}(x))$ when x lies between 0 and 1 and $secant(90 - arctan^{-1}(x))$ when x is greater than 1. This graph gives impression of the relative distribution of the values of f_A for random points.

The usefulness of this approximation, though, is limited because man-made objects don't always consist of randomly distributed proportions. Note that there is a peak at the index value of 1. This peak is even more exaggerated in real applications because man-made objects are often rectangular and a rectangular set of points always has an index function of 1. Unfortunately, this cannot be avoided. Note that any two-dimensional sets of points forming a parallelogram can be transformed into any other parallelogram-shaped set of points in an image three-dimensional rotation, scaling is followed by an orthographic transformation. The next several sections will show several hashing functions which complement this one and work well in these situations.

4.3 Three-Dimensional Points: The Orthogonal Case

The last section described an indexing function that had two limitations: 1) it only works for co-planar sets of points and 2) it hashes all rectangular objects to the same value. The second problem is more significant because man-made objects will often be rectangular. This next function works only on collections of three vectors which are orthogonal in three-space. These vectors can be constructed from four points in the image. Since this function discriminates between rectangular objects with different ratios of length, height and width, it complements the previous function. Together they compensate for each others weaknesses.

The function is based on another fundamental theorem of invariance ([W], pg 53) which states that all functions of vectors invariant to rotation in three-dimensions are functions of the mutual dot-products of the vectors with each other. These invariants only hold for three-dimensions and do not, in general, carry through an orthographic projection. The next sections will show one way to reconstruct the value of this three-dimensional invariant from a two-dimensional image.

4 points in $\Re_3$ can be represented as an 3×3 matrix by filling column i of the matrix with the vector between point $i-1$ and point i. Let A signify the matrix, A_i the i-th column, A_x, A_y the first and second row and $A_{i,x}$ the first entry (x-coordinate) of the i-th column.

The mutual dot-products are formed by taking the inner square, A^tA, where A^t is the transpose of A. Let $\bar{A}$ be shorthand for A^tA and $\bar{A}_{xy}$ represent $A^t_{xy}A_{xy}$. Note that the lengths of the vectors, A_i, lie along the diagonal of A^tA and the dot-product of column A_i and column A_j is in entry (i, j). The matrix is symmetric. If the three vectors in A are orthogonal, than A^tA or $\bar{A}$ is diagonal. This matrix is invariant under multiplication by rotation matrices, R, because

$$(RA)^t(RA) = A^tR^tRA = A^tA = \bar{A}. \tag{3}$$

which follows from the fact that $R^t = R^{-1}$ because R is orthogonal.

Let A_{xy} represent the first two rows of the matrix A. This 2×3 matrix can be constructed from 4 points in a two-dimensional image as before, by setting column A_i to be t the vector between point $i-1$ and point i. Orthographic projection corresponds to leaving out the last row of a matrix. This is the result of projecting the points from $\Re_3$ down to the two-dimensional plane defined by the x and y axes. Similarly, A_z represents the lost information of the depth of these points. The effects of projection can be quantified as:

$$\bar{A} = \bar{A}_{xy} + \bar{A}_z \tag{4}$$

In this case, $\bar{A}_{xy}$ differs from A^tA by a rank-one matrix, $\bar{A}_z$. If the columns in A are orthogonal, as they are in many $3x3$ cases from the man-made, rectangular world, then the vector A_z can be reconstructed. In this case, equation (4) becomes:

$$\begin{pmatrix} A_1 \cdot A_1 & 0 & 0 \\ 0 & A_2 \cdot A_2 & 0 \\ 0 & 0 & A_3 \cdot A_3 \end{pmatrix} = \begin{pmatrix} A_{1,x}^2 + A_{1,y}^2 & -(A_{1,z}A_{2,z}) & -(A_{1,z}A_{3,z}) \\ -(A_{1,z}A_{2,z}) & A_{2,x}^2 + A_{2,y}^2 & -(A_{2,z}A_{3,z}) \\ -(A_{1,z}A_{3,z}) & -(A_{2,z}A_{3,z}) & A_{3,x}^2 + A_{3,y}^2 \end{pmatrix} + \begin{pmatrix} A_{1,z}^2 & A_{1,z}A_{2,z} & A_{1,z}A_{3,z} \\ A_{1,z}A_{2,z} & A_{2,z}^2 & A_{2,z}A_{3,z} \\ A_{1,z}A_{3,z} & A_{2,z}A_{3,z} & A_{3,z}^2 \end{pmatrix} \tag{5}$$

The off-diagonal elements of $\bar{A}_{xy}$ are enough to reconstruct the values of A_z because the off-diagonal elements of A^tA are zero. That is:

$$A_{1,z}^2 = \frac{(A_{1,z}A_{2,z}) \cdot (A_{1,z}A_{3,z})}{(A_{2,z}A_{3,z})} = \frac{(\bar{A}_{xy})_{1,2} \cdot (\bar{A}_{xy})_{1,3}}{(\bar{A}_{xy})_{2,3}} \tag{6}$$

$$A_{2,z}^2 = \frac{(A_{2,z}A_{3,z}) \cdot (A_{1,z}A_{2,z})}{(A_{1,z}A_{3,z})} = \frac{(\bar{A}_{xy})_{2,3} \cdot (\bar{A}_{xy})_{1,2}}{(\bar{A}_{xy})_{1,3}} \tag{7}$$

$$A_{3,z}^2 = \frac{(A_{1,z}A_{3,z}) \cdot (A_{2,z}A_{3,z})}{(A_{1,z}A_{2,z})} = \frac{(\bar{A}_{xy})_{1,3} \cdot (\bar{A}_{xy})_{2,3}}{(\bar{A}_{xy})_{1,2}} \tag{8}$$

The numerical subscripts in $(\bar{A}_{xy})_{i,j}$ mean the entry (i, j) in matrix $\bar{A}_{xy}$ because the alphabetical subscripts have no direct meaning.

The lengths of the three vectors in three-dimensional space can be calculated by adding the amounts from these equations together. That is:

$$A_1 \cdot A_1 = A_{1,x}^2 + A_{1,y}^2 + A_{1,z}^2 = (\bar{A}_{xy})_{1,1} + \frac{(\bar{A}_{xy})_{1,2} \cdot (\bar{A}_{xy})_{1,3}}{(\bar{A}_{xy})_{2,3}} \quad (9)$$

$$A_2 \cdot A_2 = A_{2,x}^2 + A_{2,y}^2 + A_{2,z}^2 = (\bar{A}_{xy})_{2,2} + \frac{(\bar{A}_{xy})_{2,3} \cdot (\bar{A}_{xy})_{1,2}}{(\bar{A}_{xy})_{1,3}} \quad (10)$$

$$A_3 \cdot A_3 = A_{3,x}^2 + A_{3,y}^2 + A_{3,z}^2 = (\bar{A}_{xy})_{3,3} + \frac{(\bar{A}_{xy})_{1,3} \cdot (\bar{A}_{xy})_{2,3}}{(\bar{A}_{xy})_{1,2}} \quad (11)$$

Once these values are computed. they can be combined to form an scale-invariant indexing function of these three-orthogonal points:

$$f_O(a, b, c, d) = (\frac{A_1 \cdot A_1}{A_2 \cdot A_2}, \frac{A_2 \cdot A_2}{A_3 \cdot A_3}, \frac{A_3 \cdot A_3}{A_1 \cdot A_1}) \quad (12)$$

where the A is the matrix assembled from the four points and the values of $A_i \cdot A_i$ are either calculated directly from a 3×3 matrix or indirectly from a 2×3 matrix and the equations [9 - 11]. As before, the three-tuple, f_O, should be thought of as three ranges not three numbers. The details for computing these ranges is described in the next section. f_O solves two problems with f_A because it works for three-dimensional points and it differentiates between rectangular objects with different proportions. f_O doesn't, however, work for general 3×3 matrices. One for these will be discussed later.

4.4 Error Bounds and Distribution Functions of the 3-d Orthogonal Function

The distribution properties of the orthogonal indexing function in Equation (12) discriminate between all rectangular objects which have different proportions. This makes it easy to rely upon the function to index a table filled with many different entries. The sensitivities of the function to error are not as simple to analyze because there is one singularity caused by the division in the equations used to calculate the lost depth information.

The error in equations [9- 11] can be calculated in two ways. In absolute terms it becomes:

$$\lfloor(\bar{A}_{xy})_{1,1}\rfloor + \frac{\lfloor(\bar{A}_{xy})_{1,2}\rfloor \cdot \lfloor(\bar{A}_{xy})_{1,3}\rfloor}{\lceil(\bar{A}_{xy})_{2,3}\rceil} < A_1 \cdot A_1 < \lceil(\bar{A}_{xy})_{1,1}\rceil + \frac{\lceil(\bar{A}_{xy})_{1,2}\rceil \cdot \lceil(\bar{A}_{xy})_{1,3}\rceil}{\lfloor(\bar{A}_{xy})_{2,3}\rfloor} \quad (13)$$

$$\lfloor(\bar{A}_{xy})_{2,2}\rfloor + \frac{\lfloor(\bar{A}_{xy})_{1,2}\rfloor \cdot \lfloor(\bar{A}_{xy})_{2,3}\rfloor}{\lceil(\bar{A}_{xy})_{1,3}\rceil} < A_2 \cdot A_2 < \lceil(\bar{A}_{xy})_{2,2}\rceil + \frac{\lceil(\bar{A}_{xy})_{1,3}\rceil \cdot \lceil(\bar{A}_{xy})_{2,3}\rceil}{\lfloor(\bar{A}_{xy})_{1,3}\rfloor} \quad (14)$$

$$\lfloor(\bar{A}_{xy})_{3,3}\rfloor + \frac{\lfloor(\bar{A}_{xy})_{1,3}\rfloor \cdot \lfloor(\bar{A}_{xy})_{2,3}\rfloor}{\lceil(\bar{A}_{xy})_{1,2}\rceil} < A_3 \cdot A_3 < \lceil(\bar{A}_{xy})_{3,3}\rceil + \frac{\lceil(\bar{A}_{xy})_{1,3}\rceil \cdot \lceil(\bar{A}_{xy})_{2,3}\rceil}{\lfloor(\bar{A}_{xy})_{1,2}\rfloor} \quad (15)$$

where $(\lfloor(\bar{A}_{xy})_{i,j}\rfloor, \lceil(\bar{A}_{xy})_{i,j}\rceil)$ is the interval of the real numbers where the value of $(\bar{A}_{xy})_{i,j}$ must lie. The upper and lower ends of this interval can be calculated by estimating the possible error in calculating the dot-products which make up $\bar{A}_{xy}$. In the two dimensional case which holds for images. the error in calculating the dot-product of two vectors, [p,q] and [r.s] when p.q.r and s can be shifted by some error bound, ϵ, is:

$$\begin{aligned}
[p+\epsilon_p, q+\epsilon_q]\cdot[r+\epsilon_r, s+\epsilon_s] &= pr+qs+r\epsilon_p+p\epsilon_r+q\epsilon_s+s\epsilon_q+\epsilon_p\epsilon_r+\epsilon_q\epsilon_s \quad (16)\\
&< pr+qs+\epsilon(|p|+|q|+|r|+|s|)+\epsilon^2\\
&> pr+qs-\epsilon(|p|+|q|+|r|+|s|)-\epsilon^2
\end{aligned}$$

where $\epsilon_p, \epsilon_q, \epsilon_r, \epsilon_s$ are errors in the individual coordinates which satisfy $-\epsilon < \epsilon_p, \epsilon_q, \epsilon_r, \epsilon_s < \epsilon$.

As, before the bounds on the error can be calculated simultaneousl with the index function itself to restrict the search of the table. Experiments have shown, however, that the system is rarely as sensitive to error as the absolute bounds in equations (13 - 15). It is been prudent, in most cases. to simply compute the bounds using traditional gaussian models of noise. If this fails to produce an acceptable answer, than the absolute bounds can be used.

4.5 Knowing When the Orthogonal Indexing Function Doesn't Apply

The equation (12) can be applied to any collection of points in an image, even if the collection of four points could not be an image of three orthogonal vectors. The mathematical structure of this particular case allow a stronger result than in the other cases. Collections of four points which could not possibly be generated by three orthogonal vectors can be eliminated using the following fact about 3×3 matrices composed of orthogonal vectors. A_1,A_2 and A_3:

$$\sum_{i=1}^{3} \frac{A_{i,z}^2}{A_i\cdot A_i} = 1 \quad (17)$$

This can be proven by noting that every 3×3 matrix A composed of orthogonal vectors, can be decomposed into RD where R is a rotation matrix from $SU(3)$ and D is a diagonal matrix with the lengths of the vectors along the diagonal. If the columns of A are normalized to unit length, then the matrix becomes a member of $SU(3)$. The basic format of these rotation matrices are:

$$\begin{pmatrix}
cos\psi cos\phi - cos\theta sin\phi sin\psi & cos\psi sin\phi + cos\theta cos\phi sin\psi & sin\psi sin\theta\\
-sin\psi cos\phi - cos\theta sin\phi cos\psi & -sin\psi sin\phi + cos\theta cos\phi cos\psi & cos\psi sin\theta\\
sin\theta sin\phi & -sin\theta cos\phi & cos\theta
\end{pmatrix} \quad (18)$$

where ψ,ϕ and θ are the Euler angles of rotation which the unit basis is rotated. The last row of R represents the z-coordinate of the rotation matrix whose length is:

$$sin^2(\theta)sin^2(\phi) + sin^2(\theta)cos^2(\phi) + cos^2(\theta) = 1 \tag{19}$$

This can be extended to prove equation (17) by generalizing the result to non-unit vectors. Equations (13 - 15) can be extended to calculate the bounds of error on equation (17) or models of gaussian noise can be used for better estimations.

4.6 Three-Dimensional, Non-orthogonal Sets of Points.

The previous three sections described indexing function which worked for four-points which could be represented by three orthogonal vectors. This function is only useful for rectangular solids. This next section will describe another function which uses the same vector representation, but is not limited to orthogonal vectors.

The function is based upon the fact that singular values are invariant under rotation ($A = UDV, RA = (UR)DV = \bar{U}DV$) and an important theorem from matrix algebra ([GVL], page 286) that states that if σ_1 and σ_2 are the singular values of A_{xy} and $\bar{\sigma_0}$, $\bar{\sigma_1}$ and $\bar{\sigma_2}$ are the singular values of A, then:

$$\bar{\sigma_0} \leq \sigma_1 \leq \bar{\sigma_1} \leq \sigma_2 \leq \bar{\sigma_2} \tag{20}$$

These can be converted into an indexing function by computing the ratio:

$$f_S(a, b, c, d) = \frac{\bar{\sigma_2}}{\bar{\sigma_0}} \tag{21}$$

where $\bar{\sigma_2}$ and $\bar{\sigma_0}$ are the largest and smallest singular values of the 3×3 matrix derived from the points, a, b, c and d. This ratio is also known as κ_2, which is the condition number of the matrix A under the norm, $\|\cdot\|_2$. It roughly measures the numerical stability of the matrix.

Four features in a 2-D image can be looked up in the table by converting them into a 2×3 matrix and computing the two singular values σ_0 and σ_1. The result of equation (20) shows that these points could only potentially match values when:

$$1 < \frac{\sigma_1}{\sigma_0} < \frac{\bar{\sigma_2}}{\bar{\sigma_0}}.$$

Like the other two functions, f_O and f_A, this function specifies a range of values whose size is determined by the error. Unlike the other two functions, this range has a fixed lower-bound of one for all cases. Unfortunately, if f_S of four features in an image is 1, then it could potentially match all other features. There is some intuition, though, that suggests that it is not possible to do better than this. Consider two examples, 1) three equal-length, orthogonal vectors and 2) three almost co-linear vectors of widely differing lengths. It is possible to rotate 2) to appear like (1) , but it is not possible to do the reverse. Figure 5 illustrates this case.

4.7 Distribution and Error Properties of the Non-Orthogonal Index Function

It is possible to estimate the effects of error on this indexing function using some classic theorems from matrix algebra. A corollary to the Wielandt-Hoffman theorem ([GVL], pg 287) states that if A and a perturbed matrix, $A+E$, are in $\Re_{m\times n}$. then:

$$\sum_{k=1}^{n} \sigma_k[A+E] - \sigma_k[A] \leq ||E||_F^2 \tag{22}$$

where $||E||_F$ is the Frobenius norm, or the square-root of the sum of the squares of the entries in E. The 2×3 case is the important one, and if the simple assumption is made that all the error affects either the larger or the smaller singular value then:

$$\frac{\sigma_2 - \sqrt{6}\epsilon}{\sigma_1 + \sqrt{6}\epsilon} < \frac{\sigma_2}{\sigma_1} < \frac{\sigma_2 + \sqrt{6}\epsilon}{\sigma_1 - \sqrt{6}\epsilon} \tag{23}$$

As before, this value is an absolute bound on the error and not practical for regular usage. An empirical estimate of the bounds was computed by constructing 10,000 random 3×3 matrices and permuting the entries by 5%. Figure 6a histograms the distributions of the percentage change in f_S for these matrices.

The distribution of the values of f_S has been extensively studied in physics and multivariant analysis. The work of [Ed] provides a good introduction to the distribution of singular values in the limiting case as the size of the matrices increases. Since this paper is only concerned with 3×3 cases, the distribution in Figure 6b was calculated empirically using MATLAB. The condition number of 50,000 random 3×3 matrices was calculated and the distribution is shown here. All values larger than 100 were set to 100 to increase the resolution of the graph.

5. Breaking up Three-Dimensional Models

Once a collection of index functions has been chosen and their error and distribution properties analyzed, the problem becomes how to convert a collection of models into a model-base so potential matches can be indexed and evaluated. The main considerations are:

1. Size – How large does the data-base grow with the number of models? How does it depend on the complexity of the model?

2. Access Complexity– How many matches are made? What percentage of them are viable?

3. Grouping Information– How can the data base be structured so that grouping information can be used successfully?

This part of the paper will consider the problem of breaking apart a three-dimensional model and adding it to a topologically-defined network of invariants. It will also describe how the structure of this network of invariants will make it possible to look up groups of features in the model efficiently. Although it won't be discussed until the next section, the basić structure of the network is designed to interface with the grouping method that finds convex groups such as [HW].

5.1 Construction the Topologically Defined Network of Invariants

Consider an object. There are certain, characteristic areas that are likely to appear as sharp shifts in intensity in digital images. These are often corners and edges which delinate different planes that receive different illumination, but they can include patterns, textures and different colorings of the surface. The boundaries between the distinctive areas can be identified in advance and their pattern converted into a graph where the *arcs* represent the boundaries and the *nodes* represent the corners where two or more arcs meet. The graph of a simple, uniformly colored cube would have eight nodes and twelve arcs linking the nodes.

If an object is present in an image, then the network of image edges and their intersections should be a subset of this graph. Some parts of the graph will be missing because the orientation of the object obscures half of the object and occlusion may obscure othe parts. Also, some arcs that should be visible might be missing because of the lighting of the scene. Noise and lighting can also act counter to our interests by introducing new, spurious edges that are not part of the object. The observation to be made here (and it has been made many times before in Computer Vision) is that the nodes in the graphs have a geometrical relationship with each other, and the goal is to encode these relationships so that small subsets can be recognized. The first several sections on the transformation-invariant give ample information about how to encode subsets of these nodes so they can be recognized from any pose. The principle problem is encoding this relationship efficiently so grouping information can be used.

Wolfson's method [LW] simply chooses three feature points on the model and stores the affine-invariant coordinates of all other features relative to these three points. This step is then repeated for all triples on the model. In Costa, Haralick and Shapiro's paper [CHS], only numerically stable triples are used, but this number is still $O(n^3)$ triples. While this may be reasonable in two-dimensional cases when most features could be visible, it becomes less efficient in three-dimensions where one half of the object is obscured from the camera.

The topologically-defined network of invariants will only use the transformation-invariant functions discussed earlier on *neighboring* features. The geometrical relationship between these small, locally-connected groups of features are encoded with a network of pointers between overlapping groups. There is no need to hash collections of features relative to all possible frames of reference. This substantially reduces the size of the data base and simultaneously encodes geometric and topological structure which can usefully be exploited by a grouping system like the one described in [HW].

The topologically-defined network of invariants is constructed by taking each arc in the model and encoding the subsets formed by considering all combinations of the arc's neighbors. Before beginning, let each arc be denoted, a_i with endpoints, n_a and n_b. Let $Attach(n_b)$ be the set of arcs which are attached to n_b. More formally:

1. Construct a list of models, $\mathcal{M}$. Each model should be a list of nodes representing corners and arcs representing edges between corners.

2. Construct one database for each index function: $\mathcal{D}_\mathcal{A}$,$\mathcal{D}_\mathcal{O}$, $\mathcal{D}_\mathcal{S}$ and/or any additional functions. All possible triples composed of the edges in $\mathcal{M}$ will be constructed in the subsequent steps and placed in the appropriate database. They can either be simple sorted lists or more complex structures for better access. The sections on the distribution of values (4.2,4.4 and 4.7) can be used to design the implementation of the database. The use of special memory hardware like that described in [B],[WS] and [Z] can also be used effectively.

3. Each arc, a_i, connects two nodes n_j and n_k. For each arc, find $Attach(n_j) - a_i$ and $Attach(n_k) - a_i$ which are the sets of arcs attached to the two endnodes less the arc, a_i.

4. For each arc, find all possible triples of arcs, (a_x, a_i, a_y) such that $a_x \in Attach(n_j) - a_i$ and $a_y \in Attach(n_k) - a_i$.

5. For each triple, apply the appropriate indexing function and add it to the correct database. (Either (12) if the three edges are orthogonal, (1) if the three edges are co-planar or (21) if the three edges are neither.)

6. Add a pointer from the arc itself, a_i, in the list of models, $\mathcal{M}$, to this triple, (a_x, a_i, a_y), in either $\mathcal{D}_\mathcal{A}$,$\mathcal{D}_\mathcal{O}$ or $\mathcal{D}_\mathcal{S}$. This will speed the intersection process described later.

The size of this network can be estimated from the complexity of the model. If $\{a_i\}$ is the set of arcs, then the size of the list of models, $\mathcal{M}$, is equal to the size of $\{a_i\}$. The size of each additional database depends upon the complexity of the models. If the set is entirely made up of flat objects, then the affine index function, f_A will apply and the size of $\mathcal{D}_\mathcal{A}$ will contain all of the triples. In any case, the sum of all of these will be:

$$\sum_{\forall a_i}(Attach(n_j) - 1) \times (Attach(n_k) - 1) \tag{24}$$

where n_j and n_k are the nodes attached to a_i.

The figure 7 illustrates this process and shows one edge from a cube being broken up into triples. There are four different sub-parts containing this node since each endpoint has two adjacent edges. Two of them contain three co-planar edges and the other two contain three orthogonal edges. The affine index function, f_A, would be applied to the first two and the orthogonal indexing function, f_O, would be applied to the second two because it is a cube with right-angles.

5.2 Matching with the Network of Invariants

Matching a set of features is straight-forward. In the next section, the actual interface with a grouping algorithm such as [HW] will be discussed, but this section just assumes that there is a subset of image features which are presumed to belong to the same object. The algorithm needs to identify all possible items in the model base which might correspond to it.

Begin with the simple case of four ordered features which can converted into three edges. If there is no ordering to the features then all possible ordering can be tried. Each possible indexing function (i.e. f_A,f_O,f_S and any others) is applied and the values which are returned are used to look up possible matches in either $\mathcal{D}_{\mathcal{A}}$,$\mathcal{D}_{\mathcal{O}}$ or $\mathcal{D}_{\mathcal{S}}$. Equation (17) is used to screen the search of $\mathcal{D}_{\mathcal{O}}$. The error functions described in the earlier sections guides the search through $\mathcal{D}_{\mathcal{A}}$,$\mathcal{D}_{\mathcal{O}}$ or $\mathcal{D}_{\mathcal{S}}$ so that all potential matches are noted given preliminary assumptions about the noise and error bounds. At this point, a potential pose is calculated from the positions of the four points and after a hidden line algorithm determines what the model should like from this pose, the image and this two-dimensional projection of the model are aligned. The amount of correspondence between the features is noted and the model system can rank the results using a method like [HK]. If no satisfactory match emerges, then the assumptions about the potential for error can be re-evaluated and the processes repeated.

If more than four features are identified in the image, then there are two or more triples of data available and the network of pointers can be exploited. Again, assume that there is an ordering drawn from the grouping algorithm so that the features can be converted into a list of arcs. The figure (8) shows two different sets of points: a polygonal chain of straight-lines with 5 corners and a three-dimensional 'H' of points from two faces sharing the same central line. The network of invariants can be used to lower the number of potential matches that must be tested and ranked by alignment by dividing the set of features into two triples of edges, matching the first and then using the second triple to eliminate inconsistent candidates.

The process of intersection can be demonstrated easily with a 5 sided polygon example. Let the polygon have the corners $\{c_1, c_2, c_3, c_4, c_5\}$. Here are the steps:

1. Compute $f_X(c_1, c_2, c_3, c_4)$ and $f_X(c_2, c_3, c_4, c_5)$ for $X = A, O, S$ and any other indexing functions used.

2. $f_X(c_1, c_2, c_3, c_4)$ matches a list of triples of arcs of the form (a_i, a_j, a_k) in the $\mathcal{D}_X$. Let $F_X(c_1, c_2, c_3, c_4)$ stand for this set of triples of arcs.

3. Find all models which are consistent with both triples. The model base, $\mathcal{M}$, contains pointers from each arc to all the triples in either $\mathcal{D}_A$,$\mathcal{D}_O$ or $\mathcal{D}_S$ derived from it and these pointers can be combined with the value of $f_X(c_2, c_3, c_4, c_5)$ to reduce the number of matched models. For each $(a_i, a_j, a_k) \in F_X(c_1, c_2, c_3, c_4)$, the model-base arc of a_k contains a pointer to each triple formed with a_k at the center. If $f_X(c_2, c_3, c_4, c_5)$ does not match any of of these triples, then the (a_i, a_j, a_k) is inconsistent. This triple can then be eliminated from consideration.

4. Pose determination and alignment can be used to complete the process with the new, smaller intersection of the two lists of models.

If an 'H' arrangement of features is identified. then the same intersection process can be used by considering the other class of pointers. If the form has corners $\{c_1, c_2, c_3, c_4, c_5, c_6\}$ with c_3 and c_4 as the shared base, then compute $f_X(c_1, c_2, c_3, c_4)$ and $f_X(c_3, c_4, c_5, c_6)$, look up one and use the second to eliminate mismatches by using the pointer to intersect the sets. Alignment finishes the process.

It should be noted, though, that this structure does not make any assumptions about the amount of data which is present. There only needs to be four features grouped into three edges. If some features are missing because of occlusion or noise, the system uses the information it has and will try to find the best match through alignment.

6. Using Grouping.

The grouping algorithm described in the paper by [HW] returns groups of lines which could be part of the same convex object. It will serve as an example on how to interface the two methods. Other techniques, such as [L] or [J], can also be used successfully. The convex forms can be reduced to a set of points in the following manner. If $\{l_1 \dots l_n\}$ is the set of line segments, let $\{c_1 \dots c_n\}$ be the set of points where c_i is the intersection of l_i and l_{i+1}. If the grouping algorithm returns the set of lines as a closed curve, then c_n is the intersection of l_n and l_1. These sets of points can be identified using the pointers between planar overlapping quadruples.

The grouping process makes it likely that these points correspond to the same object, but it does not guarantee it. For that reason, it is necessary to first compute the index function of the successive quadruples, $\{c_1, c_2, c_3, c_4\}, \{c_2, c_3, c_4, c_5\} \dots \{c_n, c_1, c_2, c_3\}$. Then look up each value and use the intersection procedure to produce viable matches. The best chances will be closed, convex curves.

One advantage of the convex-grouping method in [HW] is that it will place a particular line in two different types of groups– one for each side. This information can be used to reduce the set of possible matches by using the technique of intersections between "H"-overlapping quadruples.

7. Experimental Results

This section contains two different types of experimental results. The first type comes from tests with real data from a camera and objects resting on the table. The second type was gathered from simulated experiments using a large collection of models which were recognized in artificially constructed images. These results, which are not as not desirable as real data, still are useful for confirming many claims about the average behavior of the system.

The real data was gathered with Panasonic camera and digitized using a MaxVideo card. A Canny edge detector found the important edges In the image in Figure 9, the system was able to recognize the four models despite occlusion. The dark lines superimposed over the models shows the triples of image edges used for looking up the data. In each case, several false matches were found and eliminated after alignment.

The simulated experiments were conducted by building a collection of 300 random models in memory. They were either rectangular blocks, bent blocks, s-shaped blocks. Figure possible-blocks shows some of the shapes. The dimensions of the blocks were determined randomly.

The first experiment consisted of choosing a random model, displaying it on the screen from a random pose, choosing a random triple of edges along the side, permuting the end-points by a random amount and then using f_O to look-up the value in the $\mathcal{D}_O$ This process was repeated 100 times. The next reports the average number of triples in $\mathcal{D}_O$ which fell within the error bounds. This number is essentially the number of models which needed to be aligned and checked for correspondence. Note that in many cases two triples from the same models have the same value of f_O because of symmetry. This means that many triples found refer to the same model.

Percentage error	*Triples in table*	*Average Triples Found*
2	11402	43.33
3	11402	69.68
4	11402	99.51
5	11402	172.03

This table shows the data for a model-base consisting of 300 random bent blocks and accesses using the affine function, f_A.

Percentage error	*Triples in table*	*Average Triples Found*
1	4155	102.77
2	4155	192.97
3	4155	294.76
4	4155	607.29

The main advantage of the topologically-defined network of invariants are the links between the overlapping triples. In the second simulated set of experiments, the value of these links was tested using the first database of rectangular blocks. As before a random block was displayed at a random pose. This time, five features were extracted, permuted and the algorithm from section 5.2 was used to try and recognize the object. Table shows the number of triples that needed to found before and after the intersection.

Percentage Error, ϵ	*Triples in table* ($\mathcal{D}_O$ and $\mathcal{D}_A$)	*Average Triples Before Intersection*	*Average Triples After* Intersection
2	22804	24.27	16.10
3	22804	80.33	36.9
4	22804	83.24	43.17
5	22804	129.82	77.99

8.Conclusion

The system described in this paper can be used to efficiently characterize a large collection of three-dimensional objects so that one of them can be recognized quickly when it appears in an image. The topologically-defined network of invariants allows the smallest possible collections of features to act as indices for the table of models. while efficiently storing relative positions of the small parts.

Although the paper has only discussed one well-known invariant and two new ones, it should be quite easy to extend the concept of the network to include other invariant functions like the ones discussed by [FMZB]. It is just a matter of defining a new function and its error bounds. The grouping method in [HW] can be extended to find second order curves. When this is done, it would remove any limitation to polygonal objects. The basic strength of the structure, though. remains.

The possible structure of the storage of the network of invariants was not discussed at all. Standard methods for constructing hash-tables or indexed data-bases can certainly be applied. The specialized memory hardware like the chips designed by the Database Accelerator project worked on by Charles Sodini, Jon Wade, Sharon Marie Britton and Richard Zippel [WS], [B],[Z] at MIT could also be used effectively to search the table in constant time.

To a large extent, this paper has ignored the question of what happens when grouping information is not available. This is because several papers [Grimson], [Grimson & Lozano-Pérez] have suggested that model-based recognition must rely upon it. One section suggested trying all possible orderings of four features. This approach is clearly $O(n^4)$ when n is the number of features in the image. But grouping methods such as [HW] or [J] operate quickly (e.g. $O(n \log n)$) and so this approach is clearly less desirable if the grouping methods work successfully.

The approach to indexing here also underscores the desirability of grouping. The system only makes one entry in the table for each triple of adjoining edges. For a cube, there are 48 entries. This grows linearly if the connectivity of the features remains constant. [LW] are forced to hash their models using all possible combinations of three features. For a cube this is 220 entries and it grows cubically. Grouping is responsible for this gain.

Acknowledgements

The author would like to thank Dan Huttenlocher for his continuous and particularily thoughtful comments and advice. Allen Back contributed substantially to my understanding of the mathematical theory of group invariance.

References

[AHU] Aho, A.V.. Hopcroft, J.E. and Ullman. J.D. *Data Structures and Algorithms.* Addison-Wesley, 1983.

[B] Britton, Sharon Marie, *8k-trit Database Accelerator with Error Detection.* PhD Thesis submitted to MIT in February 1990.

[Br] Breuel, Thomas M. "Indexing for Visual Recognition from a Large Model Base", M.I.T. A.I. Lab Memo 1108, August 1990.

[CHS] Costa. Mauro, Robert Haralick and Linda Shapiro. "Optimal Affine-Invariant Point Matching." Procedings of the International Conference on Pattern Recognition, 1990.

[CJ] Clemens, David T. and David W. Jacobs. "Indexing: The Lamborghini of Object Recognition", M.I.T. Artificial Intelligence Laboratory Memo, 1246, October 1990.

[Ed] Edelman, Alan. *Eigenvalues and Condition Numbers of Random Matrices.* PhD Dissertation, MIT Department of Mathematics, 1989.

[E] Effimov, N.V. *Higher Geometry,* translated by P.C. Sinha. Mir Publishers, Moscow, 1980.

[FMZB] Forsyth, David, Joseph Mundy, Andrew Zisserman and Christopher Brown. "Applications of Invariant Theory in Computer Vision." Procedings of the First European Conference on Computer Vision.

[GVL] Golub, Gene and Charles Van Loan, *Matrix Computations,* The Johns Hopkins University Press, 1983.

[Grimson] Grimson, W.Eric L. "The Effect of Incexing on 212zthe Complexity of Object Recognition". M.I.T. A.I. Lab Memo 1126 April, 1990.

[Grimson & Lozano-Pérez] Grimson, W.E.L. and Lozano-Pérez, T. "Localizing Overlapping Parts by Searching the Interpretation Tree", *IEEE Trans. on Pattern Analysis and Machine Intelligence*. Vol. 9. pp. 469-482, 1987.

[HH] Hopcroft, John and Dan Huttenlocher. "Presented in Canadian Conference on Computational Geometry.1989. Cornell University Technical Report, 89.

[HK] Huttenlocher, Dan and Klara Kedem. *Procedings of the 6th ACM Symposium on Computational Geometry*, Berkeley, June 4-8. 1990.

[HU] Huttenlocher. Dan and Shimon Ullman. "Recognizing Solid Objects by Alignment." Int. Journal of Computer Vision. 5(2), pp. 255-274, 1990.

[HW] Huttenlocher, Dan and Peter Wayner "Finding Convex Edge Groupings in an Image". Cornell Technical Report 90-1116.

[J] Jacobs, David "Grouping for Recognition"." MIT AI Memo 1177.

[LW] Lamdon, Y. and H.J. Wolfson "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme", in the Proceedings of the Second International Conference on Computer Vision, Tampa, FL, December 5-8, 1988.

[L] Lowe, D.G. *Perceptual Organization and Visual Recognition*, Kluwer, Boston, 1985.

[L1] Lowe, D.G. "Organization of Smooth Image Curves at Multiple Scales", in the Proceedings of the Second International Conference on Computer Vision, Tampa, FL, December 5-8, 1988.

[SM1] Stein, Fridtjof and Gérard Medioni, "Greycoding and Indexing: Efficient Two Dimensional Object Recognition." In *Proceedings of the International Conference on Pattern Recognition*, Atlantic City, New Jersey, June 1990.

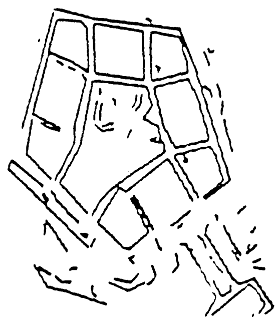
[SM2] Stein, Fridtjof and Gérard Medioni, "TOSS- A System for Efficient Three-Dimensional Object Recognition" in *Proceedings of the Image Understanding Workshop*. 1990.

[TM] Thompson, DW and Joseph L. Mundy. "Three-Dimensional Model Matching from an Unconstrained Viewpoint." in *Proceedings of the IEEE International Conference on Robotics and Automation*, August 1987.

[WS] Wade, Jon P. and Charles G. Sodini, "A Ternary Content Addressable search Engine" *IEEE Journal of Solid-State Circuits*, 24(4):1003-1013, August 1989.

[W] Weyl, Hermann. *Classical Groups*, Princeton University Press, 1946.

[Z] Zippel, Richard. "Programming the Data Structure Accelerator," *Proceedings of Jerusalem Conference on Information Technology*, Jerusalem, Israel, October 1990.



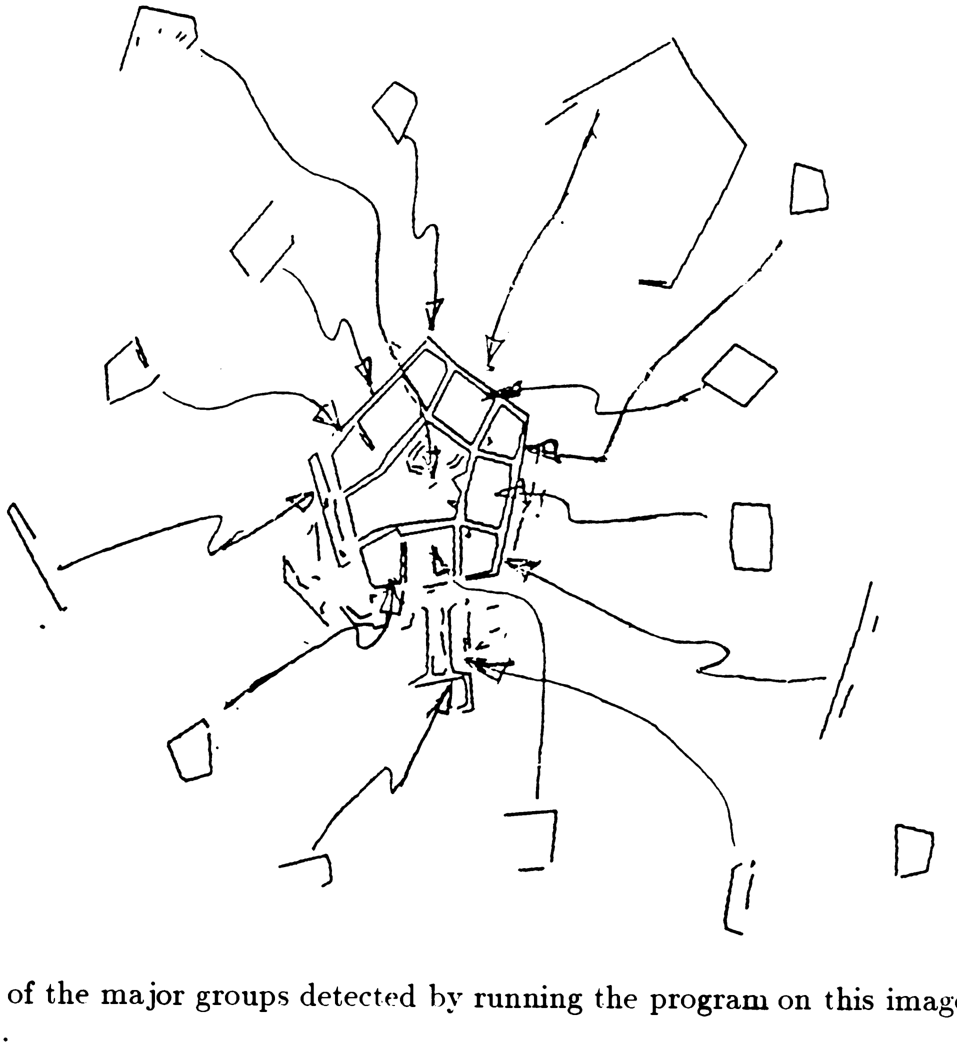


Figure 2: Some of the major groups detected by running the program on this image of the pentagon.

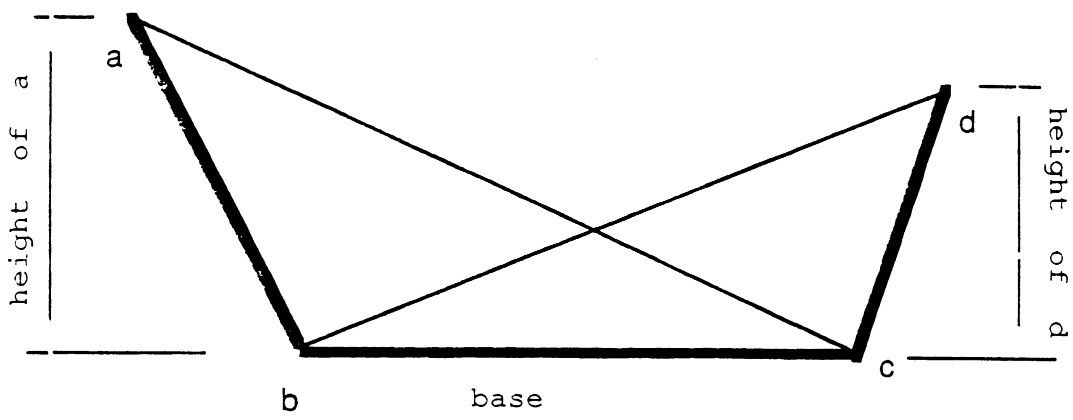


Figure 3: Two triangles formed by four points sharing a common base-line.

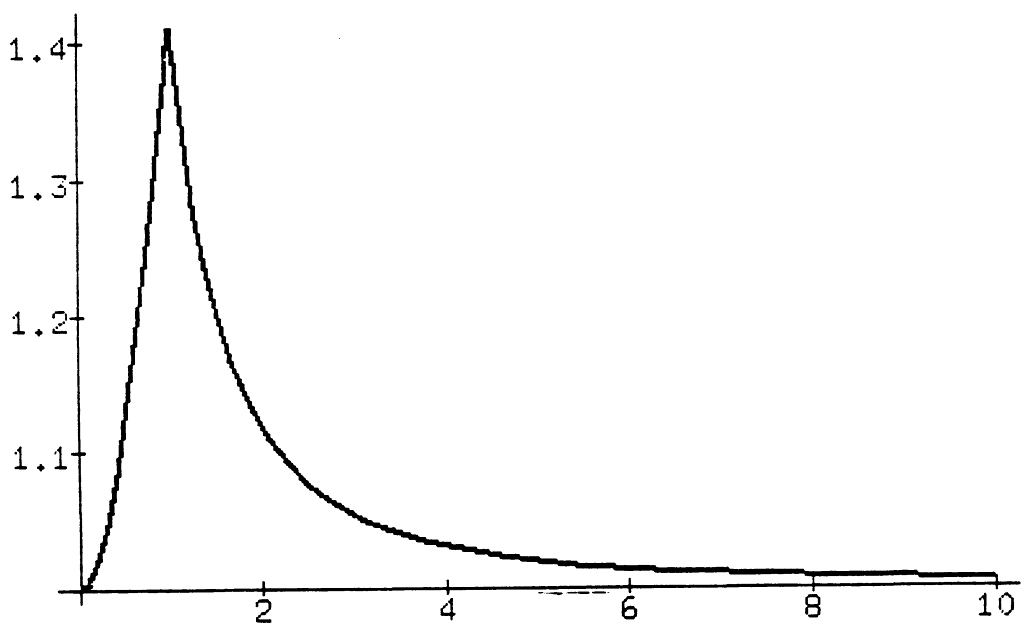


Figure 4: The graph showing an approximation of the relative distribution of the values of the affine indexing function when applied to four random points.

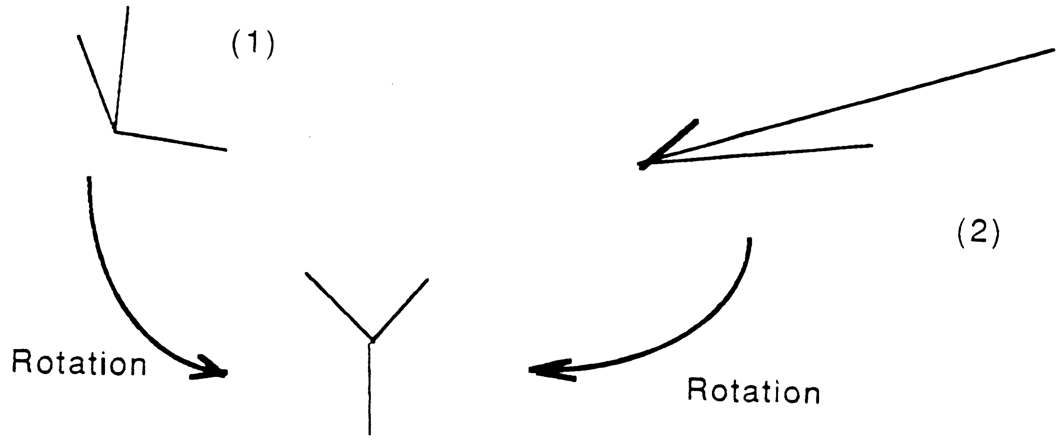


Figure 5: Intuitional evidence for why Equation credible. (2) can be rotated to look like (1). but (1) cannot look like (2).

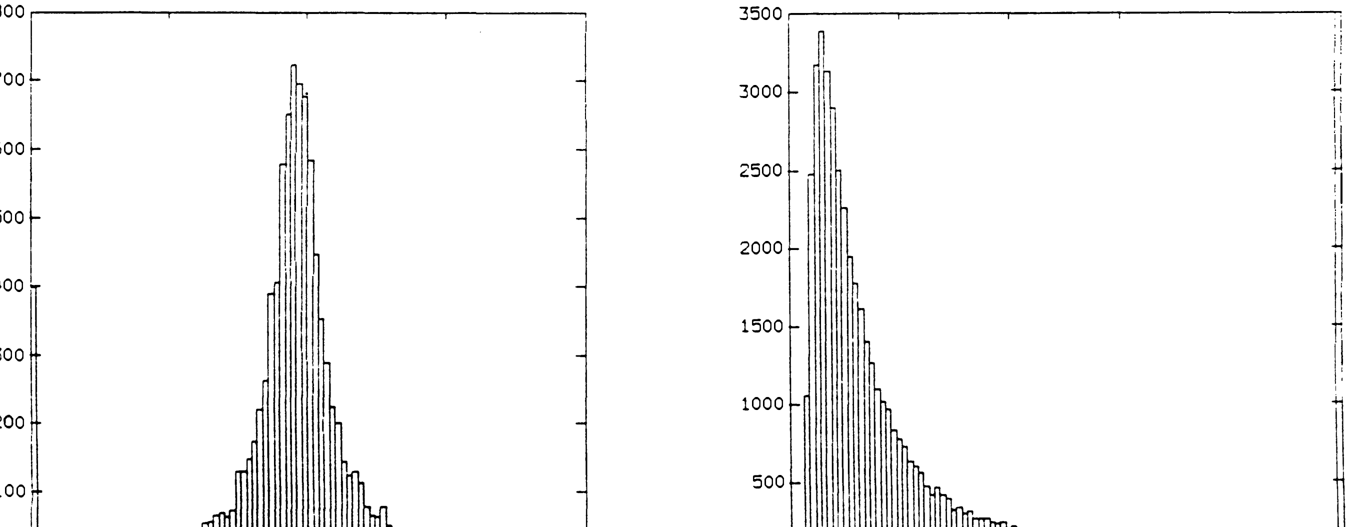


Figure 6: (a) The graph shows a histogram of the percentage change of f_S when 10.000 3×3 matrices are permuted by a random 5%. (b)The graph showing the distribution of f_S for 50.000 random, 3×3 matrices.

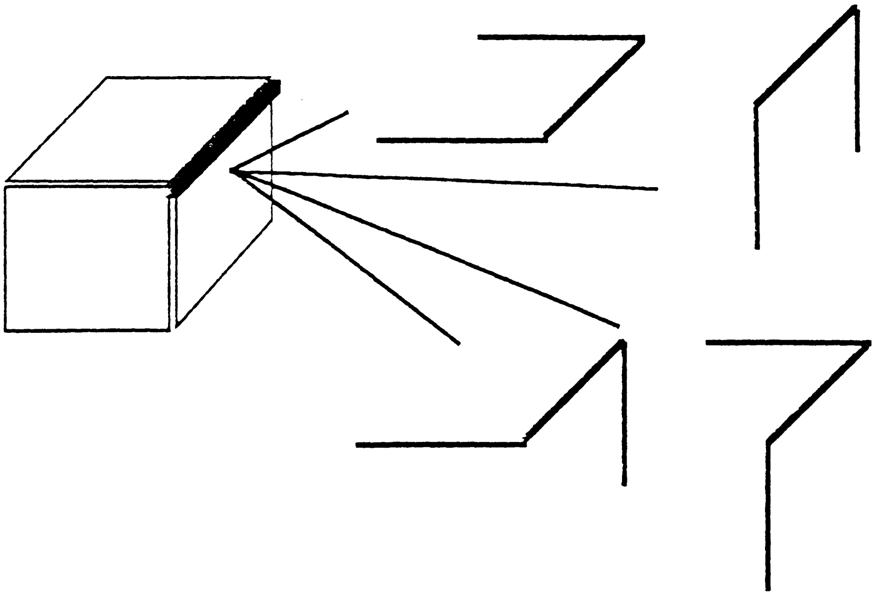


Figure 7: Some of the groups formed by constructing triples from one arc on the side of a cube.

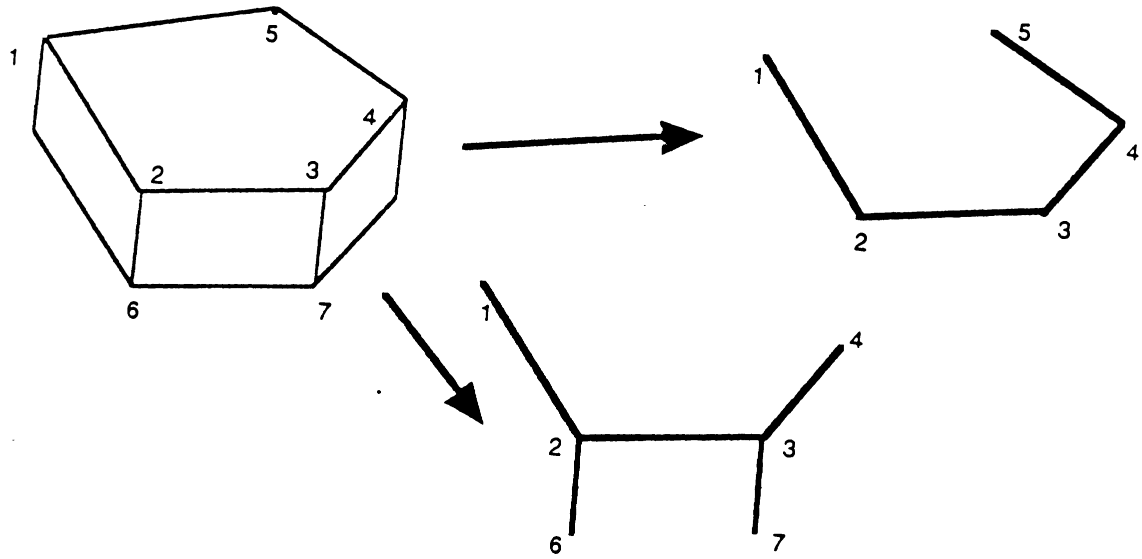


Figure 8: Two different groups with more than four points. (a) shows five planar points and (b) shows six non-planar points in an '**H**' pattern.

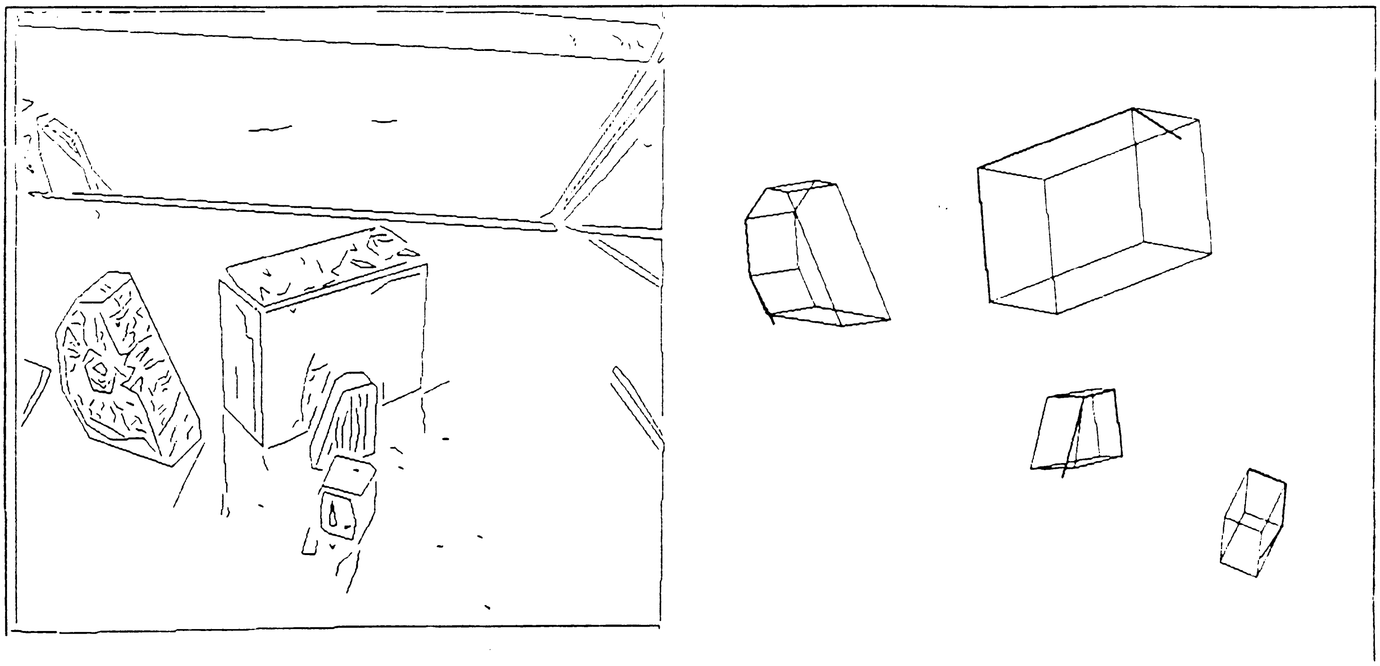


Figure 9: The edges from a picture and the four models recognized in it. The dark lines superimposed over the models show the triple of image edges used in the recognition step.