Matching Three-Dimensional Objects Using a Relational Paradigm

Linda G. Shapiro
John D. Moriarty*
Robert M. Haralick
Prasanna G. Mulgaonkar

Department of Computer Science
Virginia Polytechnic Institute
and State University
Blacksburg, VA 24061

Technical Report CS80014-R

This research was supported by the National Science Foundation under grants MCS-7923827 and MCS-7919741.

* Currently employed at Science Applications, Inc.

ABSTRACT

A relational model for describing three-dimensional objects has been designed and implemented as part of a database system. The models which provide rough descriptions to be used at the top level of a hierarchy for describing objects, were designed for initial matching attempts on an unknown object. The descriptions are in terms of the set of simple parts of the objects. Simple parts can be sticks (long, thin parts), plates (flat, wide parts), and blobs (parts that have three significant dimensions). The relations include an attribute-value table for global properties of the object, the properties of the simple parts, binary connection and support relationships, ternary connection relationships, parallel relationships, perpendicular relationships, and binary constraints.

An important use of the system is to characterize the similarity and differences between three-dimensional objects. Toward this end, we have defined a measure of relational similarity between three-dimensional object models and a measure of feature similarity, based only on Euclidean distance between attribute-value tables. In a series of experiments, we compare the results of using the two different similarity measures and conclude that the relational similarity is much more powerful than the feature similarity and should be used when grouping the objects in the database for fast access.

I. Introduction

In scene analysis, we are given one or more views of a three-dimensinal scene. As part of the analysis, we must identify the three dimensional objects using only the two-dimensional views. The data are arrays of numbers representing light intensities or distances or other measurable quantities, depending on the sensor. Noise, distortion, and sampling error are common. Segmentations of the data into objects or surfaces or cylinders are far from perfect.

In this kind of environment, it seems reasonable that the first atttempts at matching should involve only very rough three-dimensional object models. Exact dimensions and exact geometric specification will not be useful until the analysis procedure has narrowed down the choice of models. Instead, rough models that characterize the structure of the object can be used.

In Section III, we motivate and define our relational model for three-dimensional objects. The object models are intended for use in a scene analysis system. A database of such models has been set up, as described in Section V. However, our first experiments with these models are not concerned with scene analysis. Instead, we look at the more fundamental question, "What makes two three-dimensional objects similar?" Surely, if we cannot answer this question, we cannot hope to answer the harder question, "To which three-dimensional objects is this two-dimensional projection most similar?" In Section IV, we define a new kind of two-way relational matching to be used to compare two relational descriptions. In Section VI we describe several kinds of experiments comparing two three-dimensional models using 1) only their global features and 2) the entire relational structure.

II. Related Literature

We have divided the relevant literature into two categories: three-dimensional object representation and matching.

II.1 Three-Dimensional Object Representation

We have chosen to use relational models to represent three-dimensional objects. In this section, we survey these and other 3D object representations. There are several categories of applications that require modeling of three-dimensional objects. These include mechanical design and manufacturing, computer graphics, and computer vision. We will restrict our discussion to the representations used in computer vision. Other representations can be found in Badler and Bajcsy [3] and the proceedings of the Workshop on the Representation of Three-Dimensional Objects [44].

Surface-Edge-Vertex Models

All of the early work in vision and much of the present work used surface-edge-vertex models of three-dimensional objects. Roberts' [29] model included points, lines and planar surfaces in three-space implemented as a ring structure. The object of his work was to find junction points in a given line drawing that fit a transformation of some stored model. Huffman [19], Clowes [10], and Waltz [43] labeled the line segments of a line drawing as corresponding to concave, convex, boundary, and, in Waltz's work, crack or shadow edges in three-space. Regions delimited by line segments could be labeled as background or as a surface of one of the objects in the scene. There are no stored object models; what is stored is knowledge, in the form of all two-dimensional junction labelings that can correspond to trihedral blocks world objects.

Analysis by line labeling has been extended to curved surfaces as in Shapira and Freeman [33], Turner [40], and

Chakravarty [8]. The model that Shapira and Freeman employ allows either quadratic or planar surfaces, edges which are all or part of the intersection of two surfaces, and vertices which are the intersection of three or more edges. In their model a boundary is a closed chain of edges, a face is a bounded portion of a surface, and a body is a closed connected part of three-space, delimited by a finite number of faces. They used multiple views of objects in their analyses and were able to validate junctions that correspond to real vertices, connect some pairs of junctions by empty lines where no line appeared in the line drawing, and create synthetic junctions where the real junction was hidden. Their program finds the faces of each body and the corresponding region in each view.

Chakravarty worked with planar-faced or curved-surface solid bodies having vertices formed by at most three surfaces. Junctions are labeled with respect to the number of regions at the junction, the junction type (based on the arrangement of the lines), and the number of regions associated with the line leaving the junction. Lines are labeled as limb, non-occluding, occluding, partial limb, partial non-occluding, partial occluding, and concave. He developed a junction transition graph where a cycle having consistent line labels represents the traversal of a region's boundaries.

Another extension is to the Origami world where objects are created by folding paper. Origami world objects can have surfaces that are not part of a solid. Kanade [20] extends line labeling to the Origami objects where he will usually get several legal labelings per drawing. He then maps geometric properties of the line drawing such as parallel lines and skewed symmetry into gradient space where a gradient represents how a plane is slanted relative to the line of view. His problem is to uniquely determine the gradients of the surface of each object, and he has succeeded with several simple objects.

Surface-edge-vertex models have also been used by Nagao, et.al. [26] whose method was to estimate defects in the two-dimensional description, produce imperfect models from the perfect models, and match to the imperfect models; McKee and Aggarwal [23] who performed recognition on partial views of known objects; Richard and Hemami [28] who used Fourier descriptors of the silhouettes of objects stored as wire-frame models; and numerous others.

Relational Models

Relational or graph models have become very popular since it was discovered that the relational matching problem

(Barrow, Ambler, and Burstall [5]) can be greatly reduced by using relaxation processes. Two recent studies are of particular interest to our work. Chien and Selander [9] use object models that include networks of surface, edge, and vertex atoms, each having several properties and connected by surface-edge and edge-vertex arcs. The surfaces in their models may be planar, cylindrical, or spherical. A complete object model consists of several networks representing several views. Matching is from an image graph, extracted from the input image into a library of object models. The network matching utlizes a cost function and tries to find a low-cost association that pairs image parts with object parts.

Schneier [32] represents objects by primitives and relations, but with the special feature that common primitives and relations are shared across models and within models. His program produces a scene graph from several views of range data of an object. It tries to find an isomorphism between the scene graph and a structure derived from the graph of models where all three-dimensional models are represented. The matching process utilizes fast indexing; primitives and relation schemata index all models in which they occur, and models index all primitives and relation schemata within them. The main advantage is the elimination of the need to match against every one of a library of stored models.

Generalized Cylinders

The second major type of three-dimensional model used for computer vision is the generalized cylinder model suggested by Binford [7] and first used with laser range data to produce descriptions of curved objects (Agin and Binford [1]). A generalized cylinder is a volume defined by a space curve axis and the cross section function at each point of the axis. In Nevatia's work (Nevatia and Binford [27]), the three-dimensional models consist of generalized cylinders with normal cross sections for primitives, plus connectivity relations and global properties. Cylinders are described by length of axis, average cross-section width, ratio of the two, and cone angle. Global properties of an object include number of pieces, number of elongated pieces, and symmetry of the connections. In the matching phase, an indexing scheme is used to access objects that are likely to match an unknown. Each object has a three bit code describing each of its distinguished pieces. Encoded are 1) connectivity (one end or both) 2) type (long or wide) and 3) conical (true or false). Objects with the same code are grouped together and the correct group is found before full matching is started.

Marr and Nishihara [22] think of objects as stick figures where each stick is the axis in one or more generalized cylinders. They advocate hierarchical models; at the top level a hand may be represented by a single cylinder which is broken down further at subsequent levels. In order to describe the connections between cylinders they employ two vectors: $AXIS which can be placed along the axis of a cylinder whose connection is to be described and $SPASAR which can be used to describe the rotation of a second cylinder about the first. The relationship of two touching cylinders is described by a triple (p,i,g) where p is the position at which $SPASAR attaches to $AXIS, i is the inclination of $SPASAR to $AXIS, and g is the girdle angle describing the rotation of $SPASAR about $AXIS. If the cylinders do not touch directly, then the description uses the pair (d,e) where d is the perpendicular distance from $AXIS to the beginning of $SPASAR, and e is the girdle angle.

Marr and Nishihara also believe in the use of indexing in recognition. They distinguish between indexing clues that can be used before there is a guess at the three-dimensional configuration (for example, connectivity and some length comparisons) and those that cannot. Their matching scheme uses relaxation to rotate the model into the appropriate view to match the description obtained from the two-dimensional image.

Hollerbach [18] used generalized cylinders in his hierarchical models of pottery vases. His model of a vase is a main cylinder segmented into possible parts: foot, body, neck, and lip. Parts can be described by a general shape (i.e. ovoid) plus modifiers (i.e. protrusions, size, position). Soroka [39] used generalized cylinders with elliptical cross sections to model three-dimensional biological data obtained from tomographic data. In other recent work, Agin [2] has developed a new system where objects are modeled by generalized cylinders and arbitrary spatial relationships. Relationships include snakes (several cylinders grouped along a single axis), attachment points, and arbitrary transforms. Users can code S-expression descriptions such as (CUBE2 (ATTACH CUBE2 TOP) CUBE 1) to describe objects to the system.

General Knowledge Models

The models discussed so far have specific primitives (surface, edge, vertex, or generalized cylinder), some description of the properties of those primitives, and often some kind of connection relation. Several more general models have been proposed. Minsky [24] has defined a "frame" as a data-structure for representing a stereotyped situation. A frame is like a network of nodes and relations where the top levels are fixed and represent things that are always fixed about the situation and the lower levels have slots that must be filled with specific information. For a stereotyped scene of three-dimensional objects, Minsky's model is a set of frames describing the scene from different viewpoints plus the transformations between pairs of these frames representing the effect of moving the camera. A related model has been proposed by Ballard, Brown, and Feldman [4]. Their model is a semantic network where nodes represent primitive and complex objects and concepts such as assertions or procedures. Given this model, an image, and a query pertaining to a particular object, their system would construct a sketch map (an instantiation of part of the model that matches the scene) and use it to answer the query.

II.2 Matching and Constraint Satisfaction

Our three-dimensional models are relational structures. Relational matching, the process of finding relational homomorphisms between two structures is an NP-complete problem; in the worst case, its behavior is expected to be exponential. However, it has been shown that the use of look-ahead or relaxation operators can speed up the tree search used for finding a match. Since our relational matching will require some form of relaxation, we will survey some of the recent work in this important area.

Discrete Relaxation

In [17], we defined a general network constraint analysis problem, called the consistent labeling problem, which was a generalization of specific problems from several different specialty areas. In the general problem, we are given a compatibility model (U, L, T, R) where U = {1,...,M} is a set of M objects called units, L is a set of names for the units called labels, T $\subseteq$ U**N specifies N-tuples of units that constrain one another, and R $\subseteq$ (U x L)**N specifies N-tuples of unit-label pairs ((u1, l1), (u2, l2), ...,(uN, lN)) where unit u1 can have label l1, unit u2 can have label l2,..., and unit uN can have label lN, all at the same time. A labeling of U is a mapping f: U --> L that assigns a label to each unit. The consistent labeling problem is to find all labelings f that satisfy (u1,...,uN) $\in$ T implies (u1,f(u1),...,uN, f(uN)) $\in$ R. We have shown that the relational homomorphism problem is a consistent labeling problem.

Consistent labeling problems have traditionally been attacked by a depth search where the search procedure assigns labels to units as long as it can find a label for each new unit that is compatible according to R with the labels already fixed to previous units. Whenever the procedure cannot find a label for a new unit, it backtracks. Such a procedure suffers from thrashing; a poor choice of labels for one of the first units can cause failure of all paths stemming from that choice.

Ullman [41] first tried to avert this thrashing behavior in a matching application. Waltz [43] popularized discrete relaxation by using it in a program to label the edges of a line drawing as concave, convex, boundary, shadow, or crack. His 'filtering' program was applied prior to the tree search, and it removed so many possible labelings, that frequently the tree search became unnecessary. Rosenfeld, Hummel, and Zucker [30] formalized the relaxation operator used by Waltz. Using our consistent labeling notation, U is the set of edges, L is the set of edge labels, T = {(line 1, line 2) | line 1 connects to line 2}, and R = {((line 1, label 1), (line 2, label 2)| (line 1, line 2) ∈ T and there is some physically possible labeling of the junction where line 1 and line 2 meet in which line 1 can take label 1 while line 2 takes label 2}.

In the Rosenfeld, Hummel, Zucker formalism a labeling is an N-tuple of sets Lk = (Ł(1,k),...,Ł(M,k)) where Ł(i,k) is a set of labels that are still allowed for line i at iteration k. Ł(i,o) is the set {label | ((line i, label), (line j, label j)) $\in$ R for some j}. L(k+1) is obtained from L(k) by discarding from each Ł(i,k) any label l such that there exists a j with {((line i, l), (line j, lj)) $\in$ r | lj $\in$ Ł(j,k)} = $\emptyset$. The relaxation procedure obtains L(1) from L(0), L(2) from L(1),..., until some L(k+1) = L(k) whereupon it halts. If L(k) = ($\emptyset$, $\emptyset$,..., $\emptyset$), there is no legal labeling; if L(k) is single-valued, the single consistent labeling has been found; and otherwise a tree search in a reduced tree is necessary.

Discrete relaxation operators have also been proposed in Haralick and Shapiro [17] and by Ullman [42], Montanari [25], Haralick and Kartus [16], Mackworth [21], Freuder [12], Gaschnig [13], Davis [11], and others. In a recent paper, Haralick and Elliot [15] compared several discrete relaxation operators by constructing random binary compatibility models on which to test the operators. It was found that a very simple operator that they call <u>forward checking</u> performed best; that is it had fewer operations and smaller execution time. The more powerful operators searched less nodes of the tree than forward checking, but took many more operations to do so. Haralick and Elliot also found that a strategy of ordering the units by always taking the next unit having fewest possible labels left tended to cause more backtracking to occur higher in the tree and thus reduced search time.

Inexact Discrete Relaxation

In our past work we developed a structural model for describing two-dimensional shapes and a corresponding procedure for matching an unknown shape to a stored model [35]. Since the unknown shapes could be distorted or noisy, their decompositions into simple pieces and intrusions and the corresponding relational desciptions were generally not identical to the stored models. To deal with this problem, we define an inexact match or an ϵ-consistent labeling as a function f: U --> L that satisfies

$$\sum_{\substack{t \in T \\ f(t) \notin R}} w(t) \leq \epsilon$$

where w is a function assigning a weight to each N-tuple t in T. In continued work [36], we further generalized the concept of an inexact match and developed relaxation operators for inexact matching corresponding to the forward checking and lookahead-by-one operators used in exact matching. We found again that the forward checking operator used less operations and less time than the lookahead by one or the backtracking tree search.

Continuous Relaxation

Rosenfeld, Hummel, and Zucker [30] proposed a continuous version of the binary consistent labeling problem. In the continuous version, each possible label l for unit i has an attached weight (pi(l) indicating the certainty with which label l is attached to unit i. The weights for each label are between 0 and 1, and the sum of the weights of all labels of a given unit must be 1. The constraint relation R also has a weight attached to each tuple. In their notation this amounts to a set of coefficients {rij,i,j,∈ U} where rij(l,L') denotes the compatibility of label l on unit i with label l' on unit j. The rij's range from -1 to 1. The job of the relaxation operator is to increase pi(l) if other units' labels that have high weights are highly compatible with l at unit i and decrease pi(l) if other highly weighted labels are incompatible with l at unit i. The relaxation operator that updates the pi's at iteration k+1 is given by

$$pi\langle k+1\rangle(l) = \frac{pi\langle k\rangle(l)[1 + qi\langle k\rangle(l)]}{\sum_{l} pi\langle l\rangle(l)[1 + qi\langle k\rangle(l)]}$$

where $qi\langle k\rangle(l) = \sum_{j} dij \sum_{l'} rij(l,l')pj\langle k\rangle(l')$

and the dij's are coefficients that weight the total interaction between units i and j.

Continuous relaxation has been used by Hanson and Riseman [14] for edge enhancement, by Barrow and Tenebaum [6] for scene analysis, by Zucker and Hummel [45] for clustering, and by others. Current work in the area deals with theoretical analyses of the continuous relaxation process to determine exactly what problem it is actually solving. To this end Zucker, et al [46] have shown that under certain restrictions, continuous relaxation is equivalent to local maxima selection.

III. A Relational Model

In this section we first describe a relational model that provides a rough description of the structure of three-dimensional objects. This model is to be used at the top level of a hierarchy for describing objects. Lower levels will be more precise descriptions, including finer details. The rough descriptions will be used for initial matching attempts and as input to a clustering procedure that will group similar objects together.

The Parts of an Object: Sticks, Plates, and Blobs

The objects that we propose to work with are complex man-made objects, such as office furniture and industrial manufacturing parts. These objects are physically built from parts. The parts can have flat or curved surfaces, and they exist in a large variety. Instead of trying to describe each of these many parts, at the top level we classify each part as either a stick, a plate, or a blob. Sticks are long, thin parts that have only one significant dimension. Plates are flatish, wide parts with two nearly flat surfaces connected by a thin edge between them. Plates have two significant dimensions. Blobs are parts that have all three significant dimensions. All three kinds of parts are "near-convex"; that is, a stick cannot bend very much, the surfaces of a plate cannot fold very much, and a blob can be bumpy, but cannot have large concavities. Figure 1 shows several examples of sticks, plates, and blobs.

Because we wish to analyze the structure of objects, we need to define sticks, plates, and blobs more precisely. Formally, a stick is a 4-tuple ST=(En,I,Cm,L) where En is the set of two end points of the stick; I is the set of interior points of the stick; Cm is its center of mass; and L is its length. Since straight line segments have each of the components of a stick, we will be able to informally

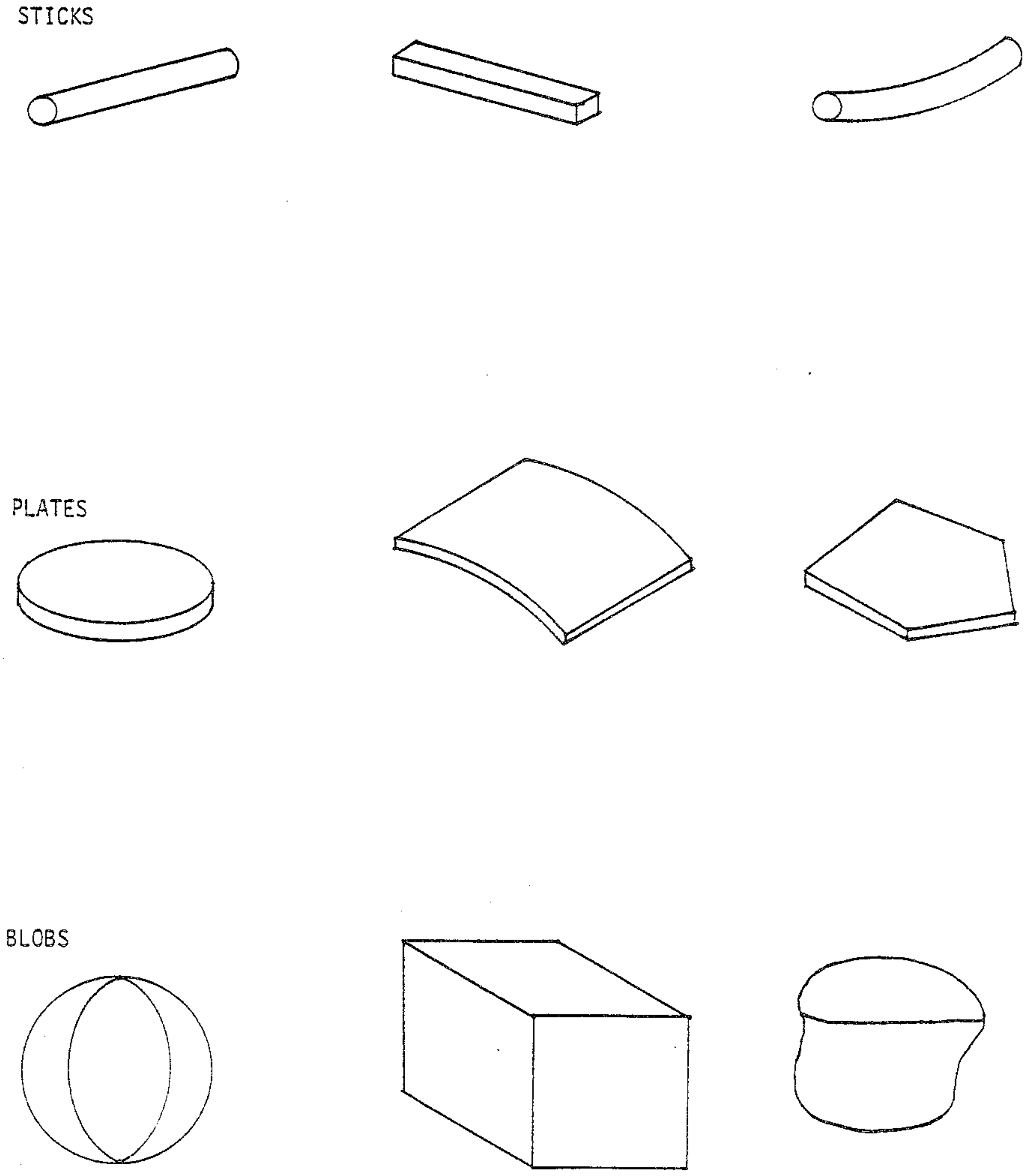


Figure 1 illustrates several examples each of sticks, plates, and blobs.

represent all sticks by straight line segments to simplify our thinking about them.

A plate is a 4-tuple PL = (Eg,S,Cm,A) where Eg is the set of edge points; S = {S1,S2} is the set of surface points of the plate, partitioned into the two surfaces; Cm is the center of mass; and A is the area. Again, to simplify analyses, we can informally represent all plates by circles.

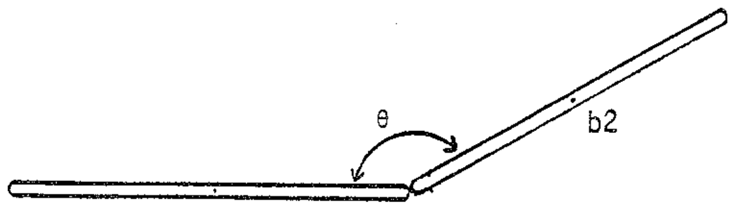
A blob is a triple BL= (S,Cm,V) where S is the set of surface points; Cm is the center of mass; and V is the volume of the blob. We can informally represent all blobs as spheres. We choose line segments, circles, and spheres because they have no corners that we might be tempted to use in our descriptions. At the top level, the descriptions are to be as general and as rough as possible.

Constraints on Assembling the Parts

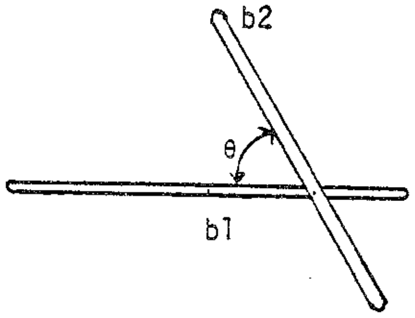
Our three-dimensional models must describe how the sticks, plates, and blobs are put together. These descriptions will also be rough; they cannot specify the physical points where two parts join. The stick has two logical end points, a logical set of interior points, and a logical center of mass that can be specified as connection points. The plate has a set of edge points, a set of surface points, and a center of mass. The blob has a set of surface points and a center of mass. We will now discuss how to use such minimal information in the models.

Binary Connections

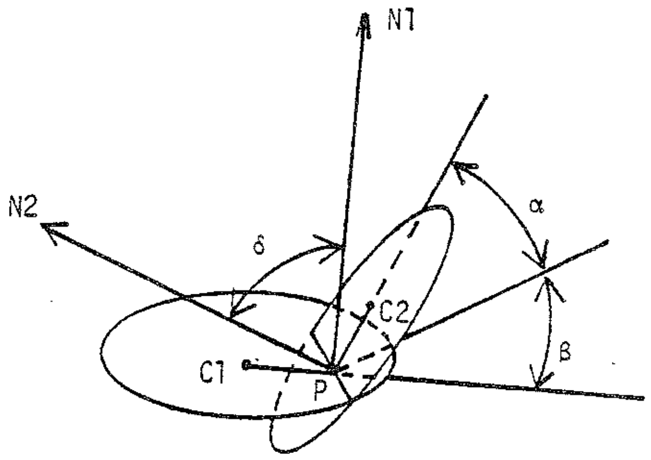
Clearly the connections between pairs of parts are an integral part of any three-dimensional relational model. We specify a connection between two parts by specifying 1) the type of connection and 2) the constrained angles of the connection. The type of connection describes which distinguished entity of the first part touches which distinguished entity of the second part. Thus possible connection types are end-end, end-interior, end-center, end-edge, and so on. For each type of connection, there is a corresponding set of angles which, when specified as single values or as ranges, constrain the binary connection further. For example, when two sticks join end-end or interior-center (as illustrated in Figure 2) a single angle constrains their connection. Figure 2 specifies the full range of this angle. If the angle is restricted to a single value, the connection is restricted to an exact form. If the angle is specified as an allowable range of values, then the form of connection is more flexible. At most three



TYPE	ANGLE CONSTRAINT
End-End	90° < θ < 180°



TYPE	ANGLE CONSTRAINT
Interior-Center of Mass	0° < θ < 90°



TYPE	ANGLE CONSTRAINT
Interior-Interior	0° < α, β < 180° 0° < δ < 90°

α: angle between projection of C2P on plane of C1 and C2P

β: angle between projection of C2P on plane of C1 and C1P

δ: angle between N1 and N2

Figure 2 illustrates three examples of the constrained connections of two simple parts.

angle ranges are required to uniquely describe a binary connection between two arbitrary parts.

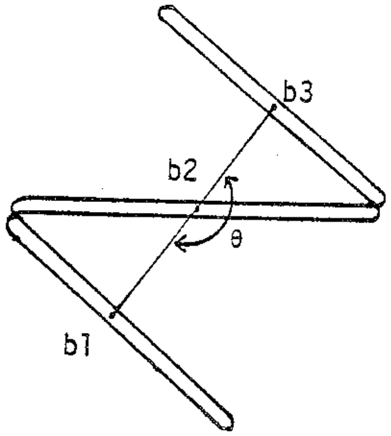
Ternary Connections

The binary connections are not sufficient to entirely describe a three-dimensional model since they do not place any global constraints on the resulting object. For example, two sticks each connected end-end to the same third stick might be at the same or opposite ends and, if at the same end, might coincide in space or not coincide, but might still be at the same angle with respect to the third stick. If we were trying to describe the object very precisely, we might need to specify N-ary connections for arbitrary N. However, we have determined that we can add powerful constraints to the model by considering triples of simple parts. Since at this level, our descriptions are rough anyway, we will only go as far as ternary relations to describe connections.

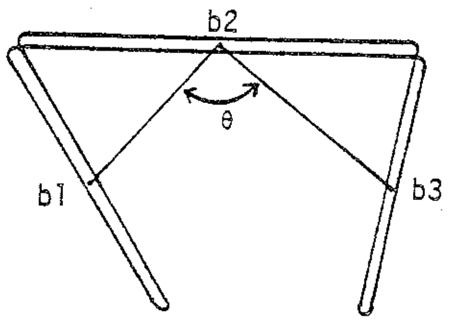
Let (s1,s2,s3) be a triple of simple parts satisfying that s1 and s3 both touch s2. The description of the spatial relationship between s1 and s3 with respect to s2 has two components. The first component specifies whether s1 and s3 meet s2 on the same end (or surface). The second component constrains the angle subtended by the centers of mass of s1 and s3 at the center of mass of s2. This angle constraint can also be a single value or an allowable range. Figure 3 illustrates several connections between the parts and the full range of the angles to be specified.

Support Structure

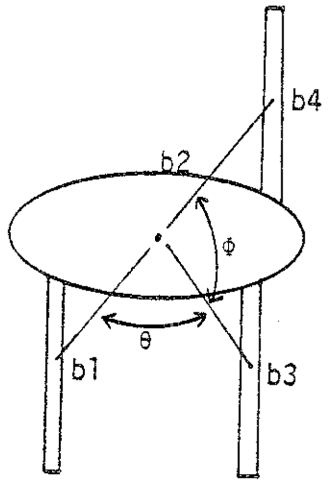
Another important aspect of a multi-part three-dimensional object is its support structure. The legs of a chair are not only connected to the seat, but they also support the seat. The support structure is related to the function of the object and its parts. For example, objects that have four upright sticks supported by the ground and supporting a horizontally-oriented plate tend to be stationary objects and tend to be used to set another object (book, vase, person) on. In addition to its importance in the three-dimensional description, the support structure can be useful in helping to identify two-dimensional perspective projections of an object, since much of the support structure of an object is often evident in a right-side-up two-dimensional view. Thus the support structure seems to be an essential component in a three-dimensional object model.



PARTS	SIDE	ANGLE CONSTRAINT
(b1,b2,b3)	OPPOSITE	$90° < \theta < 180°$



PARTS	SIDE	ANGLE CONSTRAINT
(b1,b2,b3)	OPPOSITE	$0° < \theta < 90°$



PARTS	SIDE	ANGLE CONSTRAINT
(b1,b2,b3)	SAME	$0° < \theta < 90°$
(b3,b2,b4)	OPPOSITE	$90° < \Phi < 180°$

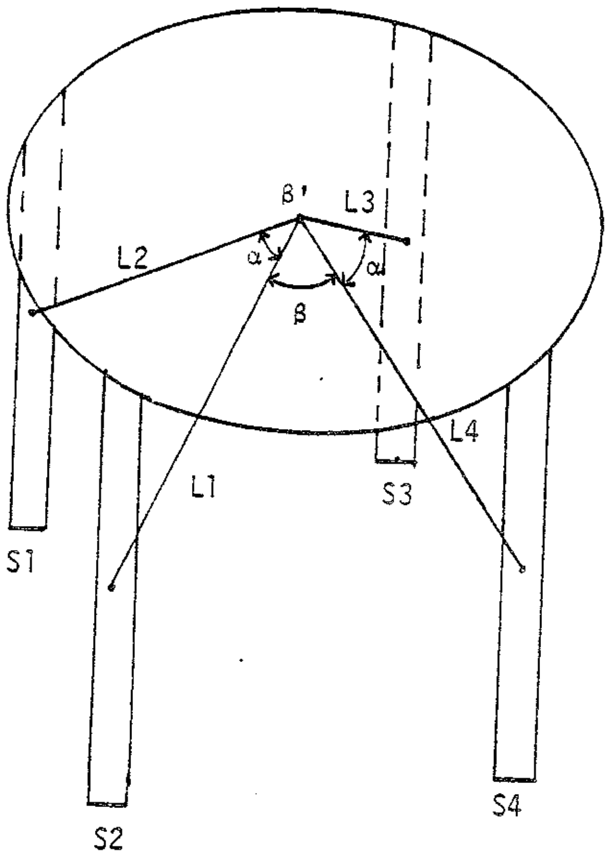
Figure 3 illustrates three examples of constrained connections among three simple parts.

Additional Constraints

The binary and ternary connection relations plus the support structure provide a basic description of the structure of an object. Additional constraints on groups of parts that do not touch each other may also be necessary to completely characterize some objects. Consider the standard table shown in Figure 4. The four sticks are all the same length, all parallel, and their centers of mass are equidistant from the center of mass of the plate. Furthermore, there are constraints on the angles between adjacent pairs of line segments from stick centers of mass to plate center of mass. These constraints rule out certain three-dimensional objects that are not standard tables, although they may have four sticks connected to a plate. The constraints are illustrated in Figure 4. For a given class of objects and a given application, there will be a set of relations that are important additional constraints.

The Relational Data Structure

In [34] we proposed a general relational data structure. The structure, which has also been called a spatial data structure or a structural description, can be formally defined as follows. A relational data structure D is a set



Constraints

parallel (S1,S2,S3,S4)
equi-length (S1,S2,S3,S4)
equi-length (L1,L2,L3,L4)
equi-angle (α,α')
equi-angle (β,β')

Figure 4 illustrates a standard table made of 4 sticks and a plate and some of the constraints on how the parts fit together.

$D = \{R1, \ldots, RK\}$ of relations. For each relation Rk, k = 1,...,K, there is a positive integer Nk and a sequence of domain sets S(1,k), ..., S(Nk,k) such that Rk $\subseteq$ S(1,k)X ... X S(Nk,k). The elements of the domain sets may themselves be atoms (nondecomposable) or relational data structures. In most relational data structures, one of the relations is an attribute-value table (A/V) and contains the values of global properties of the object being represented by the structure.

The relational data structure for a three-dimensional object will consist of an attribute-value table plus nine other relations. The unary SIMPLE PARTS relation is a list of the parts of the object. Each part is represented by a relational data structure consisting of an attribute-value table. The attributes of a simple part consist of TYPE (stick, plate, or blob), RELATIVE LENGTH, RELATIVE AREA, and RELATIVE VOLUME. The length, area, and volume values may be real numbers or may be marked "don't-care" when they are unimportant or inappropriate.

The CONNECTS/SUPPORTS relation contains some of the most important information on the structure of the object. It consists of 10-tuples of the form (s1,s2,SUPPORTS, HOW,vl1,vh1,vl2,vh2,vl3,vh3). The components s1 and s2 are simple parts, SUPPORTS is true if s1 supports s2 and false otherwise, and HOW describes the connection type of s1 and s2. The values in the HOW field are elements of the set {end-end, end-interior, end-center, end-edge, interior-center, center-center} where 'end' refers to an end of a stick, 'interior' refers to the interior of a stick or surface of a plate or blob, 'edge' refers to the edge of a plate, and 'center' refers to the center of mass of any part. The field pairs (vl1,vh1), (vl2,vh2), and (vl3,vh3) hold the low and high values for the allowed angle ranges for the (at most) three angles that can be specified for a binary connection.

The other eight relations express constraints. The TRIPLE CONSTRAINT relation has 6-tuples of the form (s1,s2,s3,same,vl,vh) where simple part s2 touches both s1 and s3, SAME is true if s1 and s3 touch s2 on the same end (or surface) of s2 and false otherwise, and vl and vh specify the permissible low and high values for the constrained angle as shown in Figure 3. The PARALLEL relation and the PERPENDICULAR relation have pairs of the form (s1,s2) where simple parts s1 and s2 are parallel (or perpendicular) in the model. The LENGTH CONSTRAINT relation has triples of the form (l1,l2,o) where l1 and l2 refer to specific line segments or part lengths and $o \in \{>,\geq,<,\leq,=,\neq\}$. The components l1 and l2 of this relation must be powerful enough to express such concepts as the line segment joining the centers of mass of two parts or the length of a stick. The BINARY ANGLE CONSTRAINT relation has triples of the form (d1,d2,o) where d1 and d2 specify angles and again $o \in \{>,\geq,<,\leq,=,\neq\}$. Finally, the AREA and VOLUME relations have triples of the form (a1,a2,o) and (v1,v2,o), respectively, where a1 and a2 refer to areas, v1 and v2 refer to volumes, and $o \in \{>,\geq,<,\leq,=,\neq\}$.

Note that specification of areas and volumes in the constraint relations may be redundant due to the fact that each part has a RELATIVE AREA and RELATIVE VOLUME attribute. The constraint relations will only be used when the constraint is meant to be emphasized as important to the object. It is expected that the inexact matching process to be used will be quite lenient about relative area and volume requirements in general. When a few constraints on area or volume need to be more severe, the thresholds that the constraint relations have to satisfy can be increased. Similarly, the PERPENDICULAR relation is redundant and will only be used for emphasis or severe constraints. See Shapiro and Haralick, 1979 [36] for the definition of inexact matching and for some fast algorithms to do it.

The attribute-value table of a tree-dimensional object contains its global properties. Our intention is to include as many properties as possible while keeping the description

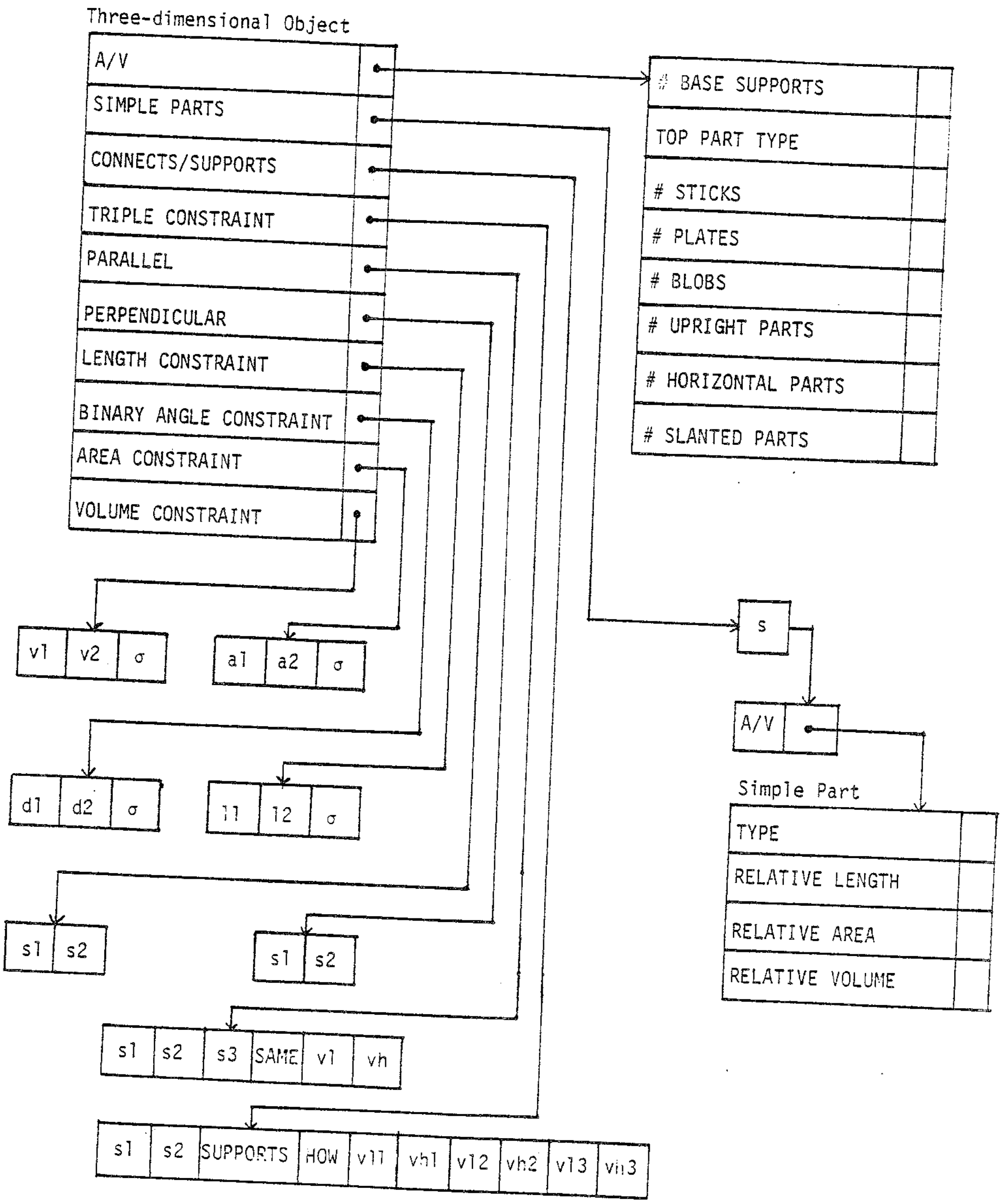


Figure 5 illustrates the logical structure of a relational data structure for a three-dimensional object model.

at a gross level. The set of attributes currently planned are number of base supports, type of topmost part, number of sticks, number of plates, number of blobs, number of upright parts, number of horizontal parts, and number of slanted parts, number of support levels, and position of topmost part. The logical structure of a three-dimensional object is illustrated in Figure 5. Notice that most of the information in the relational structure is invariant with respect to orientation of the object. Only the SUPPORTS field of the CONNECTS/SUPPORTS relation and the attributes that mention position (upright, horizontal, slanted) or support are related to orientation. These were included in the model because we expect to analyze scenes where objects are usually in their normal upright position. When this is not the case, these attributes can simply be ignored.

IV. Relational Matching

Going from relational descriptions to three-dimensional objects is essentially a matching problem: matching the relational description from the image to the relational description of the object. Doing this matching in a database of one hundred or one thousand separate objects would be computationally expensive. Given an unknown object (three-dimensional object or two-dimensional view), we do not want to compare its description with every object in the database. A solution to this problem which has been used by Salton [31] in an information storage and retrieval system is to cluster the objects in the database, represent each cluster by a profile description, and compare the description of an unknown object only to each profile description. If the unknown object is judged sufficiently close to one or more clusters, then it is compared only to objects in those groups.

Comparing Relational Descriptions

Suppose we are given two relational descriptions D = {R1, R2,..., RK} and D' = {R1', R2',..., RK'} where for each k = 1,...,K, Rk $\subseteq$ S(1,k) X...X S(Nk,k) and Rk' $\subseteq$ S'(1,k X...X S'(Nk,k). Intuitively, description D is similar to description D' if relation Rk is similar to relation Rk' for k = 1,...,K. Thus to measure the distance between two relational descriptions, we must first be able to measure the distance between two relations.

Let R $\subseteq$ S^N and R' $\subseteq$ T^N be two N-ary relations, and let f be a binary relation f $\subseteq$ S x T that associates an element of S with an element of T. We will define a measure of the error of the association f. We define the composition R∘f of N-ary relation R with binary relation f by

$R \circ f = \{(t1,\ldots,tN) \in T^N \mid$ there exists $(s1,\ldots,sn) \in R$ with $(sn,tn) \in f$ for $n = 1,\ldots,N\}$

There are four sets of N-tuple that can be used to describe the error of the association f.

1) $R \circ f - R'$

This set consists of N-tuples that arise when an N-tuple of relation R is transformed by f to an N-tuple of T^N, but this new N-tuple is not a part of R'.

2) $R' \circ f^{-1} - R$

This set consists of N-tuples that arise when an N-tuple of relation R' is transformed by f^{-1} to an N-tuple of SN, but this new N-tuple is not a part of the relation R. This set is the symmetric equivalent of set 1) and is used here because we are interested in two-way matching.

3) $R - R' \circ f^{-1}$

This set consists of N-tuples of R that are not included in the group of N-tuples obtained by applying f^{-1} to each N-tuple of R'.

4) $R' - R \circ f$

This set consists of N-tuples of R' that are not included in the group of N-tuples obtained by applying f to each N-tuple of R.

Example

Consider the two chairs C and C' shown in Figure 6 and the corresponding simplified binary connection relations

$R = \{(1,2),(2,3),(2,4),(2,5),(2,6)\}$ and

$R' = \{(A,B),(B,C),(B,D),(B,E),(B,F),$
$(C,G),(D,G),(E,H),(F,H)\}$.

Suppose we wish to measure the error of the association f given by

$f = \{(1,A,),(2,B),(3,C),(4,G),(6,F)\}$.

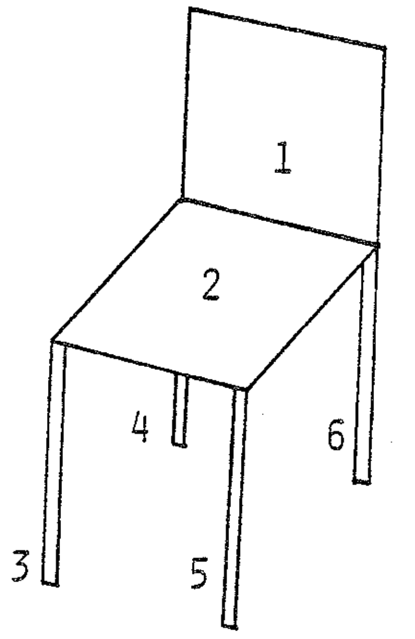
Then the two compositions are given by

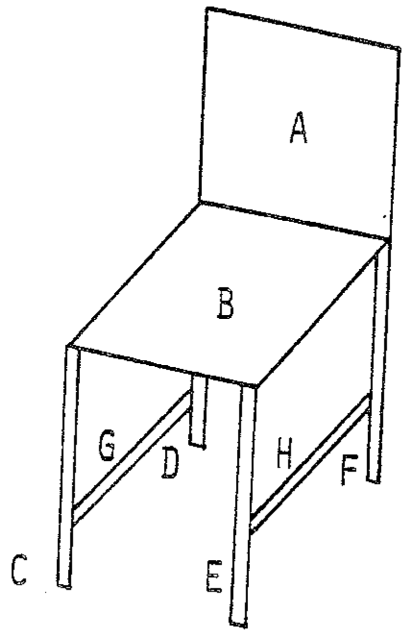
$R\circ f = \{(A,B),(B,C),(B,G),(B,F)\}$ and

$R'\circ f^{-1} = \{(1,2),(2,3),(2,6),(3,4)\}$,

and the four sets of interest are

	SET	NUMBER ELEMENTS
$R\circ f - R'$ =	$\{(B,G)\}$	1
$R'\circ f^{-1} - R$ =	$\{(3,4)\}$	1
$R - R'\circ f^{-1}$ =	$\{(2,4)(2,5)\}$	2
$R' - R\circ f$ =	$\{(B,D),(B,E),(C,G),(D,G),(E,H),(F,H)\}$.	6





R = (1,2), (2,3), (2,4)
(2,5), (2,6)

R' = (A,B), (B,C), (B,D),
(B,E), (B,F), (C,G),
(D,G), (E,H), (F,H)

Figure 6 illustrates two similar chairs and their binary connection relations. Two shapes match when their structural error is sufficiently low.

One method of defining the error of f is by a weighted normalized sum of the number of elements in each of the four sets.

$$E_{R,R'}(f) = \frac{a|R \circ f - R'| + b|R' \circ f^{-1} - R| + c|R - R' \circ f^{-1}| + d|R' - R \circ f|}{a|R \circ f| + b|R' \circ f^{-1}| + c|R| + d|R'|}$$

This measure will be 0 when R' is an isomorphic image of R and f is the isomorphism; and it will be 1 in the worst possible case when $R \circ f \cap R' = R' \circ f^{-1} \cap R = \emptyset$. It has the advantage of simplicity and the disadvantage of counting all N-tuples of a relation equally when some relationships may be more important than others.

Once such a measure of relational error has been defined, we can define the structural error SE(D,D') of two descriptions D and D' by

$$SE(D,D') = \min_{f} \sum_{k=1}^{K} wk \; E_{Rk,Rk'}(f)$$

where wk is the weight assigned to relation Rk.

Use of Global Attributes

The global attributes of a three-dimensional object are stored in the attribute-value table $A/V = \{(a,v) \mid a$ is an attribute and v is its value$\}$. The attribute-value table is essentially a feature vector and by itself cannot fully describe an object. Yet it is an important aspect of the total description. A human, when asked to describe a chair might answer, "It is an object having four legs, a back, and a seat. The legs are long, thin, and vertically oriented, the back is flat, wide, and horizontally oriented. The legs connect to and support the seat which connects to and supports the back." Note that in this description, it is very natural to mention the parts and their features before coming to the relational structure. Similarly, the human, when asked the difference between a chair and a table, might say, "The table has no back." Here the presence or absence of a part is important. This all suggests that when comparing two objects, we should first compare their attribute-value tables and only if these are judged similar enough should we continue with the full relational matching.

V. The Experimental Database System

In order to perform experiments in three-dimensional object matching, and for future use in scene analysis, we have implemented an experimental relational database system for the three-dimensional relational models. The system, which is written in PL/I and FORTRAN and runs on an IBM 370/158 under VM/CMS, gives the experimenter the ability to input, edit, compare and cluster three-dimensional object models.

Organization

Each object model is accessed by a unique integer. An object model consists of an attribute-value table plus five relations: SIMPLE PARTS, CONNECTS/SUPPORTS, TRIPLES, PARALLEL, and PERPENDICULAR. The relations contain the fields described in Section III, but the angle and size data has been omitted at this stage of experimentation and the supports component for a pair of parts (i,j) is true if i supports j or j supports i and false if no support is involved. Thus, in the objects on which the experiments were performed, we have five relations R1, R2, R3, R4, and R5 where

$R1 = \text{SIMPLE-PARTS} \subseteq \text{parts} \times \{\text{stick, plate, blob}\}$, R2 = CONNECTS-SUPPORTS $\subseteq$ parts x parts x {(supports, how) | supports $\in$ {true false} and how $\in$ {end to end, end to edge, end to interior, edge to interior, interior to interior},

R3 = TRIPLES $\subseteq$ parts x parts x parts x {same, opposite},

R4 = PARALLEL $\subseteq$ parts x parts, and

R5 = PERPENDICULAR $\subseteq$ parts x parts.

The attribute-value tables of 57 objects have been entered into the database so far. The full 5-relation structure has only been entered for 14 objects. The 57 objects are illustrated in Figure 7.

The database is organized as a set of possibly overlapping clusters of similar objects. In the current setup, only the attribute-value tables of each pair of objects were used in judging their similarity. With the attributes "number of upright pieces", "number of horizontal pieces" and "number of slanted pieces" weighted by 1 and the other seven attributes weighted by 10, the Euclidean distance between each pair of attribute-value tables was determined and thresholded, producing a binary relation. The relation was fed to the graph-theoretic clustering procedure which has been described in Shapiro and Haralick

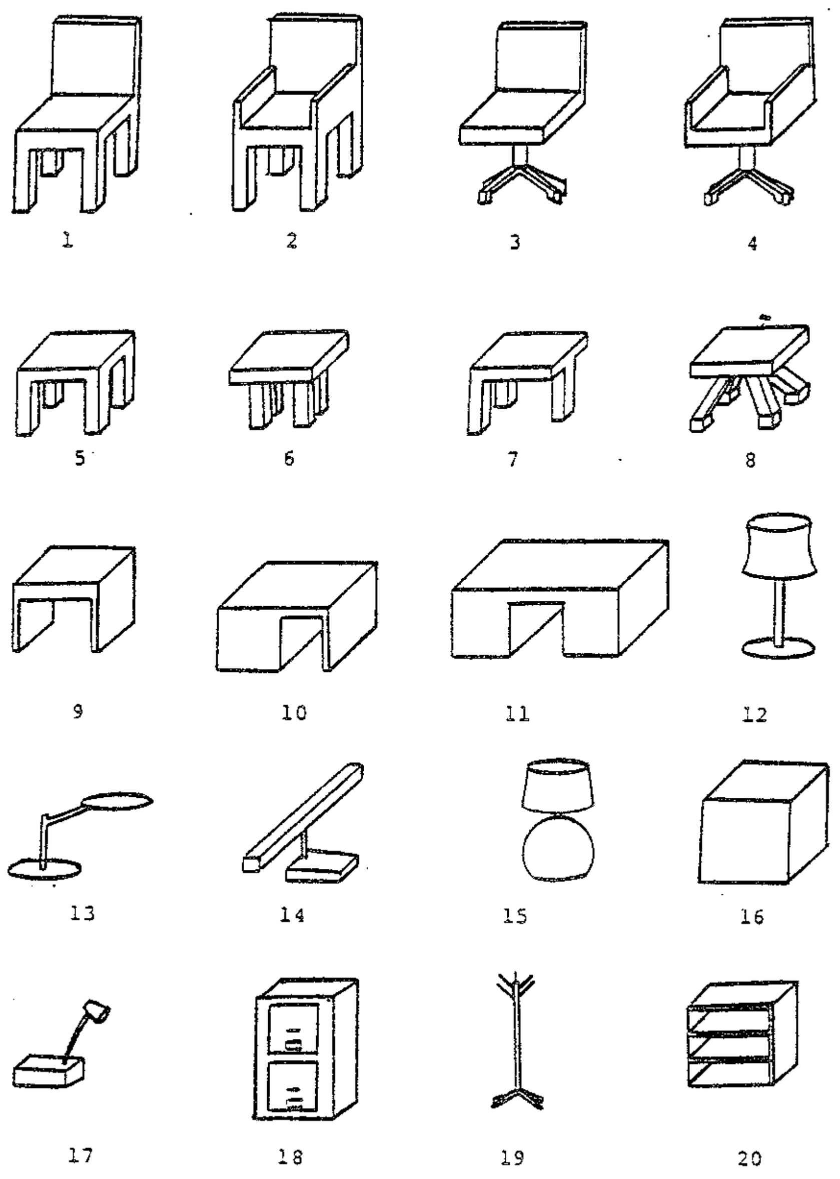


Figure 7 illustrates the 57 objects whose attribute-value tables are in the database.

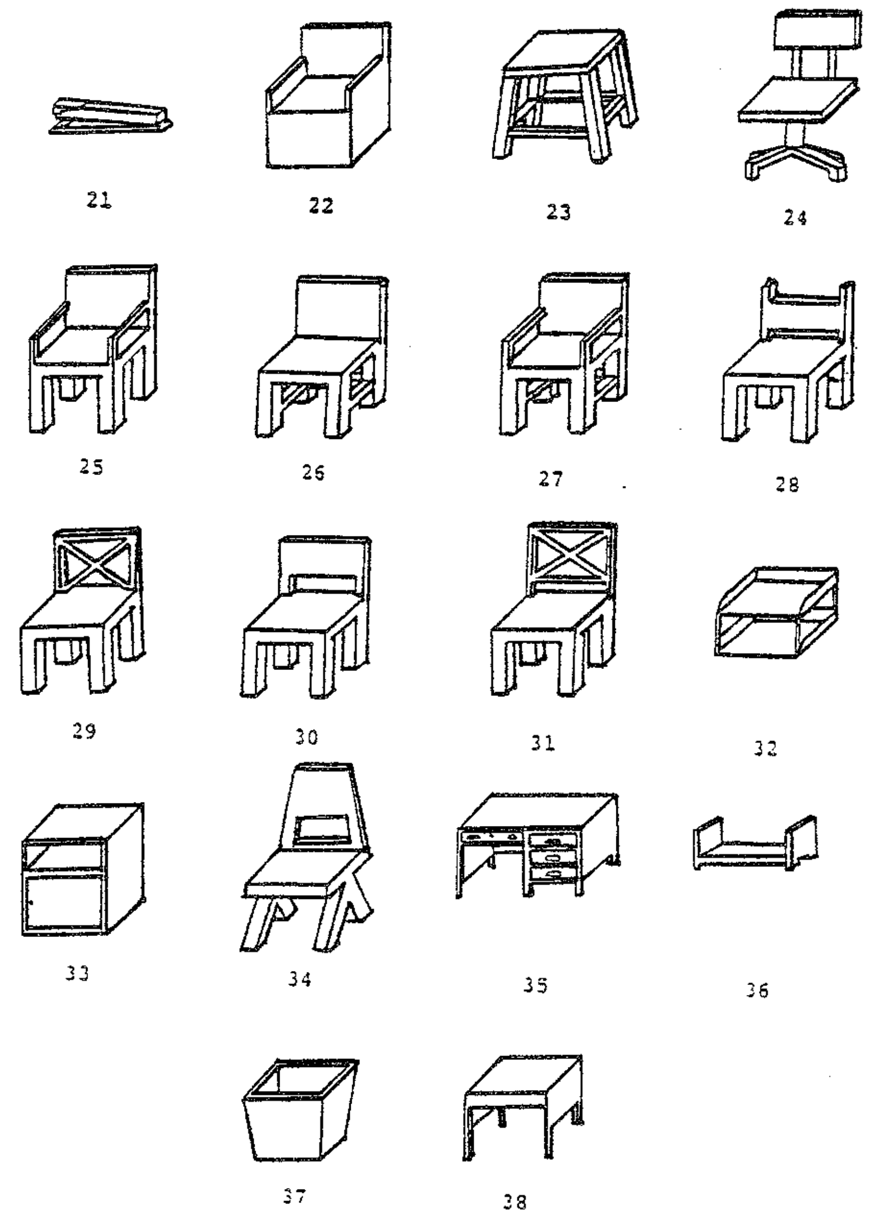


Figure 7. Continuation of 57 objects.

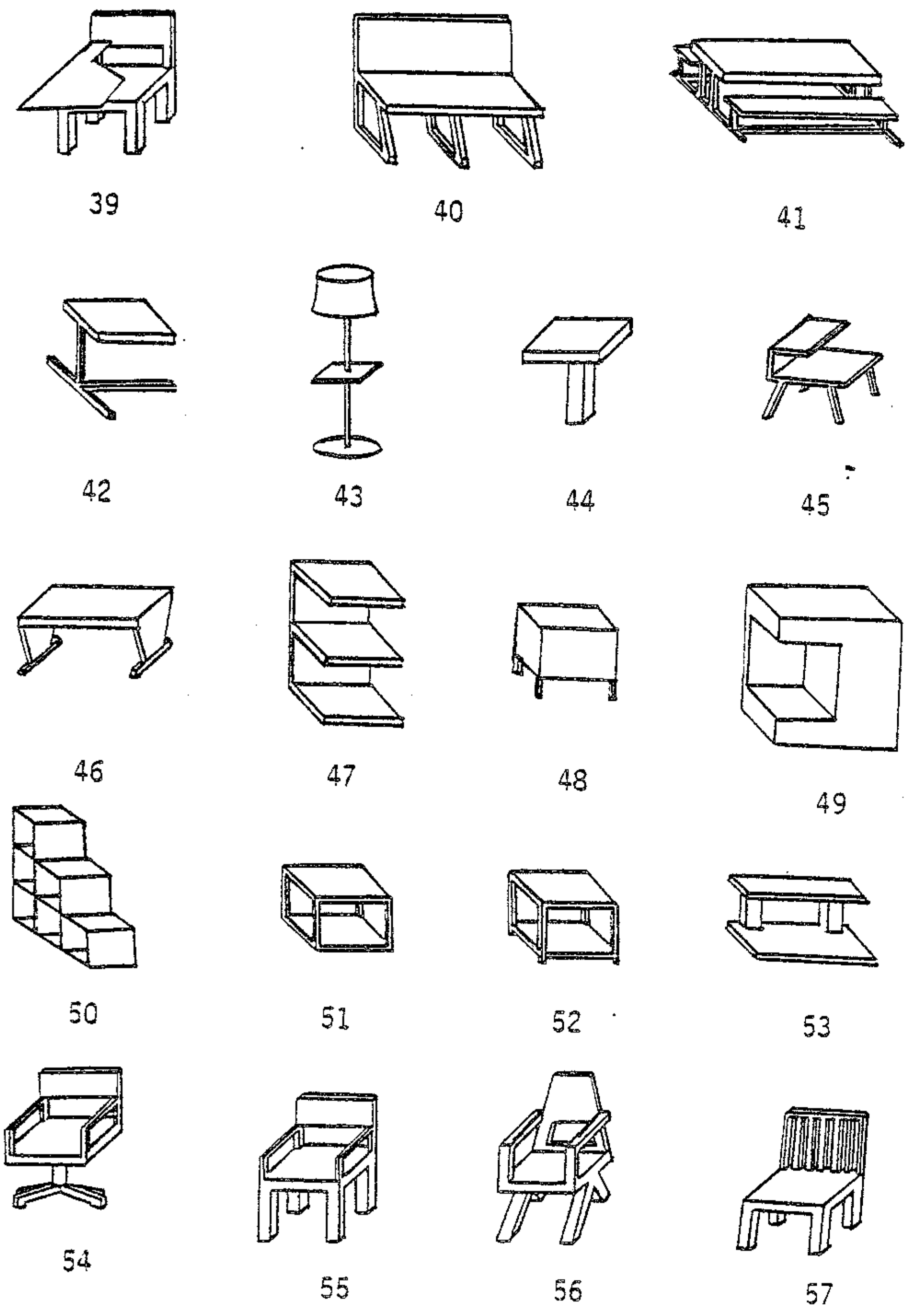


Figure 7. Continuation of 57 objects.

[37]. The set of resulting clusters currently being used in the database are shown in Figure 8.

In the standard mode of operation, an unknown object may be entered into the database or merely compared to some of the models without being entered. For object entry, the attribute-value table and five relations are input by the user, the attribute-value table is compared to the centroid attribute-value tables of each cluster, and the object is added to those clusters that it is most similar to. For matching, instead of being added, the unknown object is compared to each object in the best clusters and the results displayed to the experimenter.

Matching

The relational matching is performed using a treesearch with lookahead. Let $U = \{S_1, S_2, S_3, S_4, S_5\}$ be the unknown object, represented by its five relations and $M = \{T_1, T_2, T_3, T_4, T_5\}$ be the model. For a given association f, the total structural error is given by

$$E_s(f) = \sum_{i=1}^{5} (\#(Si \circ f - Ti) + \#(Ti \circ f^{-1} - Si))$$

and the total completeness error is given by

Cluster	Objects in Cluster
1	1, 2, 3, 4, 5, 6, 7, 8, 23, 24, 25, 26, 28, 30, 34, 35, 36, 38, 39, 40, 42, 45, 48, 52
2	9, 10, 11, 13, 14, 21, 22, 33, 42, 44, 46, 47, 49, 51, 53
3	12, 15, 16, 17, 18
4	19
5	20, 37, 47
6	23, 24, 25, 26, 27, 29, 31, 40, 41, 52, 54, 55, 56, 57
7	32
8	43
9	50

Figure 8 gives the clusters of the 57 objects of Figure 7.

$$E_c(f) = \sum_{i=1}^{5} (\#(Ti - Si\circ f) + \#(Si - Ti\circ f^{-1}))$$

where # denotes cardinality. In our matching experiments, we used a combined measure of total error:

$$E(f) = 4 * E_s(f) + E_c(f),$$

which stems from our intuitive feelings that structural error is more important than completeness error. The goal of a matching experiment between two objects U and M with parts P(U) and P(M), respectively, is to find that association $f \subseteq P(U) \times P(M)$ with minimum total error. In the experiments reported in this paper, we restricted the mapping f to being single-valued and one-one, to reduce search time.

Finding the best association is achieved with the help of two tree searches. Treesearch I, the "super-quick" search, follows only one path from the root of the tree to the bottom. At each level, it chooses that pair (p,p'), $p \in P(U)$, $p' \in P(M)$, with least accumulated error in the lookahead tables and performs forward checking [15,36] with respect to the new pair and the so-far-uninstantiated parts. The forward checking operation updates the lookahead tables and determines if this pair can be instantiated. If so, it is added to the association being constructed.

The association f obtained from Treesearch I falls into one of three categories:

1) Exact Match: $E_s(f) = 0$ and $E_c(f) = 0$
2) Subset Match: $E_s(f) = 0$ and $E_c(f) \neq 0$
3) Approximate Match: $E_s(f) \neq 0$.

In case 1) clearly the best association has already been found. In case 2), one object is contained in the other and the similarity of their attribute-value tables guarantees that not too many parts are missing. In case 3) there is no guarantee that f has minimal error and Treesearch II is called.

Treesearch II is similar to a branch and bound search using forward checking. Its job is to find an association g such that

1) $E(g) \leq E(f)$

[g's total error is not greater than f's]

2) $\#proj_1(g) \geq t^*(\#P(U))$

$\#proj_2(g) \geq t^*(\#P(M))$

[The projection of g onto its first (second, respectively) coordinate gives a set whose cardinality is at least the percentage specified by parameter t of the number of parts in U (respectively, M).]

3) $E(g') \leq k^* \sum_{i=1}^{5} (\#Si + \#Ti),\ g' \subseteq g$

[At each stage of Treesearch II, the partial association g' must satisfy the requirement that its total error is not greater than the percentage specified by parameter k of the sum of the number of N-tuple in all of the relations involved. The parameter k allows the user to control the size of the tree searched at the risk of not finding a best mapping whose error is very high near the top of the tree and very low near the bottom.

The extended forward checking algorithm used in TREESEARCH I and TREESEARCH II will be described in a forthcoming paper.

VI. Experiments

We have run several kinds of experiments using the database of relational models. The purpose of these experiment was to study the relationship between the Euclidean distance between a pair of objects obtained only from their attribute-value tables and the total relational error obtained from the tree search. Figure 9 illustrates the fourteen objects whose relational descriptions and attribute-value tables were used in these experiments.

Table 1 gives the Euclidean distances for each pair of the fourteen objects of Figure 9. Notice that, as far as the grouping in the database which was obtained using a distance threshold of 8, objects 1-7 and 9 fell into cluster 1, objects 8, 10, 11, and 13 fell into cluster 2, object 12 fell into cluster 3, and object 14 fell into cluster 4. Table 2 gives the structural and completeness errors for each pair of the same fourteen objects as obtained _only from Treesearch I_. Note that this matrix is not symmetric since TREESEARCH I does not necessarily find the minimal error mapping and may find a different upper bound when matching object i to object j than when matching object j to object i.

Table 3 gives the structural and completeness errors for each pair of the same fourteen objects as obtained from the full matching process -- Treesearch I, followed by Treesearch II, if necessary. Table 3 also gives the k-parameter used in Treesearch II for those matches where Treesearch II was required. An asterisk (*) next to the k-parameter indicates that Treesearch II ran out of time after

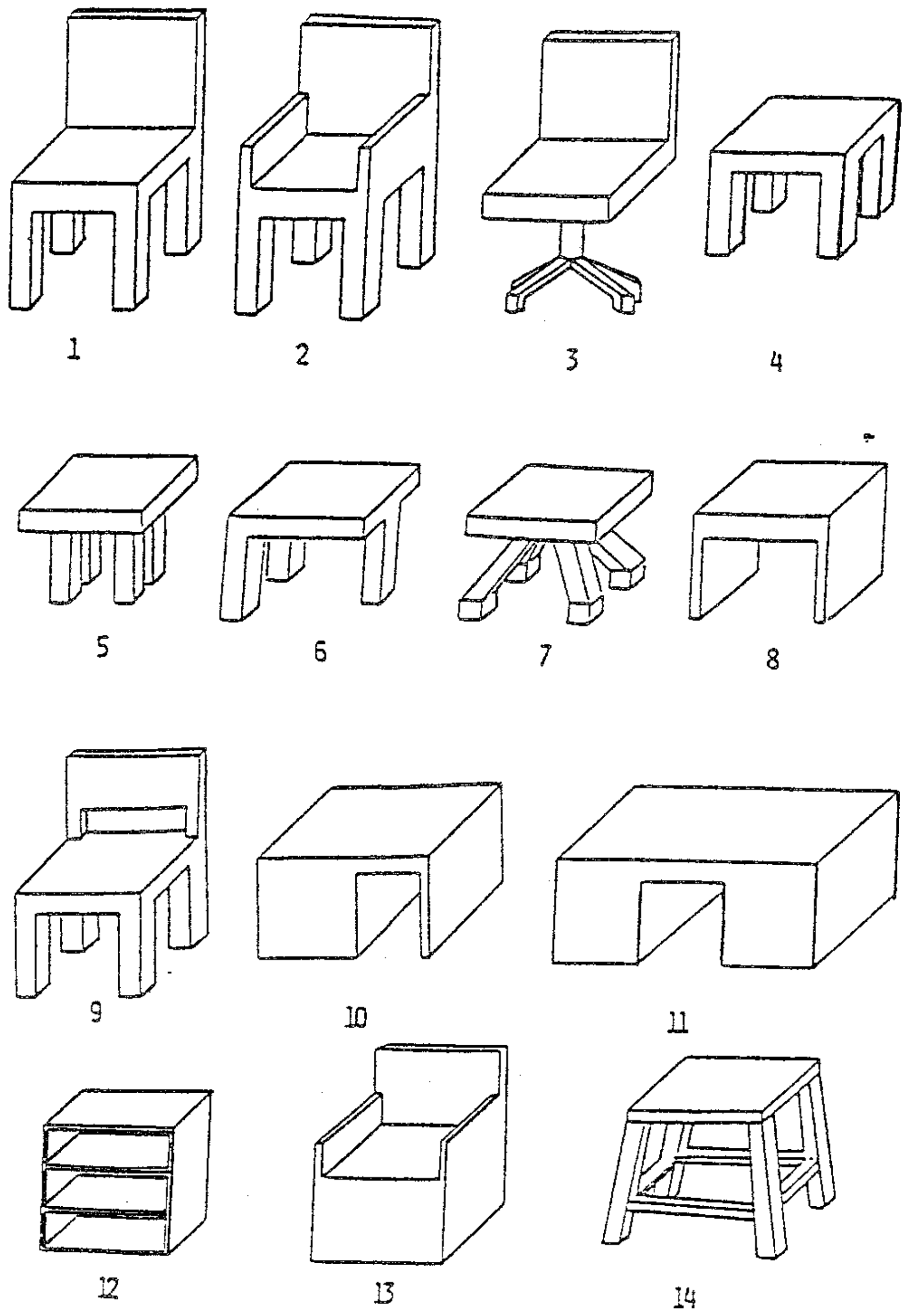


Figure 9 illustrates the 14 objects whose full relational models are in the database.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	3.46	5.20	5.57	5.57	7.21	6.24	9.64	5.66	9.70	10.49	11.66	10.19	8.66
2		0	6.24	6.56	6.56	8	7.81	8.72	6.32	10.30	11.26	11.31	9.75	9.33
3			0	7.48	7.48	8.66	7.21	10.68	4.36	10.72	11.66	11.70	11.22	8.94
4				0	0	4.58	2.83	9.01	7.94	9.11	9.17	12.04	10.59	6.63
5					0	4.58	2.83	9.01	7.94	9.11	9.17	12.04	10.58	6.63
6						0	5.20	7.81	9.17	7.87	7.94	11.14	9.54	8.06
7							0	9.27	9	9.22	9.17	12.21	10.86	7.21
8								0	11.18	4.58	6.48	7.94	5.66	11.22
9									0	11.22	12.12	12.17	11.62	8.19
10										0	4.58	9.17	5.74	11.27
11											0	10.25	7.35	11.31
12												0	8.66	13.53
13													0	12.49
14														0

Table 1. Distance Matrix for the Objects of Figure 9.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	36	0	0	0	10	10	6	10	4	65	76	24
	0	27	45	7	7	16	17	39	27	39	33	57	49	47
2	0	0	46	0	0	0	10	9	44	9	4	52	14	87
	27	0	78	34	34	43	44	66	28	56	60	86	74	74
3	30	67	0	26	26	15	16	16	46	16	26	91	34	54
	49	74	0	48	48	41	38	36	68	36	38	68	44	80
4	0	0	26	0	0	0	10	24	14	22	24	75	43	0
	7	34	48	0	0	9	10	24	50	22	24	58	28	40
5	0	0	26	0	0	0	10	24	14	22	24	75	43	0
	7	34	48	0	0	9	10	24	50	22	24	58	28	40
6	0	0	15	0	0	0	6	24	9	22	24	48	43	0
	16	43	41	9	9	0	11	15	49	13	15	51	19	49
7	10	10	16	10	10	6	0	27	14	25	27	70	43	10
	17	44	38	10	10	11	0	20	52	18	20	58	24	50
8	17	4	16	24	24	24	27	0	18	14	24	4	4	24
	33	54	36	24	24	15	26	0	52	6	8	40	12	64
9	34	88	52	14	14	9	13	18	0	18	28	89	39	82
	55	72	64	50	50	49	50	52	0	52	54	78	60	92
10	22	22	16	22	22	22	25	14	18	0	14	14	7	22
	29	56	36	22	22	13	18	6	52	0	6	42	16	62
11	30	29	26	24	24	24	27	24	28	14	0	30	15	24
	39	66	38	24	24	15	20	8	54	6	0	52	16	66
12	95	74	91	75	75	48	70	10	89	20	30	0	23	79
	63	80	68	58	58	51	58	46	78	48	50	0	42	74
13	37	30	34	43	43	43	43	4	39	9	15	23	0	45
	39	64	44	28	28	19	24	12	60	18	16	42	0	72
14	24	87	54	0	0	0	10	24	82	22	24	79	43	0
	47	74	80	40	40	49	50	64	92	62	64	76	68	0

STRUCTURAL ERROR

COMPLETENESS ERROR

Table 2. Completeness and Structural Errors Obtained after TREE_SEARCHI for Objects of Figure 9.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0 0 -	0 27 -	19 57 1°	0 7 ..	0 7 -	0 16 ..	3 35 1.2	0 39 1	0 29 1	0 39 1	10 39 1	13 71 2	4 42 2	0 47 1
2	0 27 ..	0 0 -	46 78 1°	0 34 -	0 34 -	0 43 ..	3 62 1.2	0 66 1	6 54 .99	0 66 1	10 66 1	52 86 1	14 74 1	87 74 1°
3	30 49 1°	67 74 1°	0 0 -	6 52 1.5	6 52 1.5	6 43 1.5	3 42 1.5	2 36 1	52 64 1°	2 36 1	26 38 1°	91 68 1°	34 44 1°	54 80 1°
4	0 7 -	0 34 -	6 52 1	0 0 -	0 0 -	0 9 -	3 28 1	12 30 1	0 22 1	10 28 1	12 30 1	75 58 1	43 28 1	0 40 -
5	0 7 -	0 34 -	6 52 1	0 0 -	0 0 -	0 9 -	3 28 1	12 30 1	0 22 1	10 28 1	12 30 1	75 58 1	43 28 1	0 40 -
6	0 16 -	0 43 -	6 43 1.5	0 9 -	0 9 -	0 0 -	3 19 1	12 21 3	0 31 1	22 13 1	24 15 1	48 51 1	43 19 1	0 49 -
7	3 35 1.2	3 62 1.2	3 42 1.0	3 28 1	3 28 1.2	3 19 1	0 0 -	12 22 1	1 56 1.2	25 18 1	27 20 1	70 58 1	43 24 1	0 72 1
8	2 35 1.2	0 62 1.0	2 36 1.2	12 30 3.0	12 30 3.0	12 21 3.0	12 22 3.0	0 0 -	1 52 1	2 12 1	24 8 1	0 48 1	0 20 1	1 70 1
9	0 29 1	6 54 1	52 64 1	0 22 1	0 22 1	0 31 1	3 50 1	1 52 1	0 0 -	2 52 2	12 52 2	69 78 1	15 60 1	10 60 1
10	2 35 1	2 62 1	2 36 1	10 28 3	10 28 3	10 19 3	11 20 3	2 12 1	2 52 2	0 0 -	14 6 1	2 48 1	2 20 1	1 68 2
11	12 39 3	12 66 3	12 40 3	12 32 3	12 32 3	12 23 3	12 24 3	24 10 1	12 54 1	2 12 2	0 0 -	12 52 2	1 20 2	12 72 1
12	95 63 2°	74 80 2°	91 68 1°	75 58 1°	75 58 1°	48 51 1	70 58 1	0 48 0.99	89 78 1	3 50 1	13 52 2	0 0 -	0 45 1	79 74 1
13	16 41 3	12 70 2	34 44 1	43 28 1	43 28 1	43 19 1	43 24 1	0 20 1	15 58 2	2 20 1	1 20 1	0 45 1	0 0 -	45 72 1
14	0 47 1	87 74 1°	54 80 1°	0 40 -	0 40 -	0 49 -	1 74 1	12 70 1	14 60 1°	1 68 1.5	12 70 1	79 74 1	43 68 1°	0 0 -

STRUCTURAL ERROR
COMPLETENESS ERROR
X-VAL

Table 3. Completeness and Structural Errors along with KVAL/100 after TREE_SEARCHII for Objects of Figure 9.

three-minutes and the best mapping found so far is reported rather then the true best mapping. The t-parameter used in these experiments was 75%. Again, notice that the matrix is not symmetric, although it ideally should be. This is due to our restriction on Treesearch II which forces it to find a single-valued function from one object to another instead of an unrestricted binary association. In the cases where they differ, the smaller of the two total errors may be used to estimate the relational distance between the two objects.

In Figure 10, we graphed the Euclidean distance between attribute-value tables versus total error (four times structural error plus completeness error) as determined by TREESEARCH II. As can be seen from the figure, there is some correlation, but the graph is far from a straight line. Part of the reason for this is that in the normal mode of operation of the database system, an object would only be matched against those objects that are in clusters whose centroid is deemed similar to the object. However, in these experiments, we allowed every object to be matched against every other. Comparing two objects that are extremely different can result in meaningless mappings. On the other hand, we did not expect the graph to be a straight line, since this would have indicated that structural descriptions are unnecessary and feature vectors are sufficient to distinguish between objects!

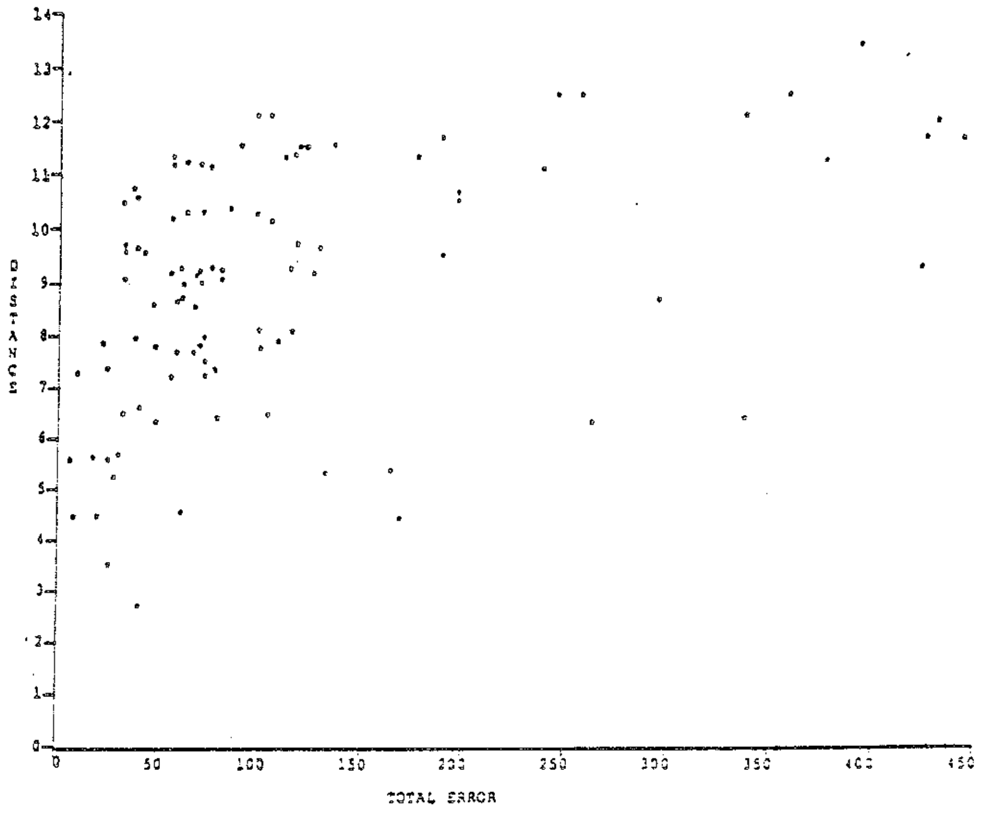


Figure 10 illustrates the graph of total error after TREE_SEARCHII versus distance from Table 1.

To analyze the situation further, we thresholded the total error between each pair of the fourteen objects to produce a binary relation. The threshold (59) was chosen so that this binary relation had the same number of pairs as the binary relation previously derived from the Euclidean distances with distance threshold 8. Clustering the relational distance binary relation, using the same graph-theoretic clustering procedure with the same parameters as used previously, gave the following results.

Cluster	Objects
1	1,2,4,5,6,7,9,14
2	3
3	1,8,10,11,12,13

Recall that the clusters obtained from the Euclidean distance binary relation were as follows.

Cluster	Objects
1	1,2,3,4,5,6,7,9
2	8,10,11,13
3	12
4	14

The main differences are:

1) Object 3 which grouped with 1, 2, 4, 5, 6, 7, and 9 using Euclidean distance, did not group with any objects using relational distance.

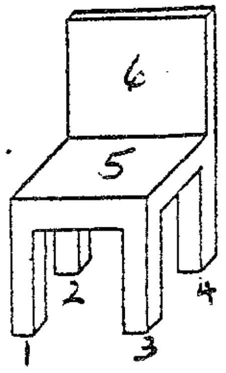
2) Objects 12 and 14 which did not group with any of the other twelve objects using Euclidean distance, each found a (different) group using relational distance.

3) Object 1 falls into two different clusters using relational distance, but only one using Euclidean distance.

What caused the differences? Consider object 1 and object 3 whose relational models are shown in Figure 11. As far as the attribute value tables, they differ in number of sticks, number of uprights, number of slanted pieces, and number of levels. In the highly-weighted attributes (number of sticks and number of levels) they differ only by one. In the low-weighted attributes (number of uprights and number of slanted pieces) they differ by 3 and 4, respectively. Thus the total difference was relatively small.

The mapping used in Table 3 from object 1 to object 3 had a structural error of 19 and completeness error of 57. It was a very unintuitive mapping that sent part 1 of object 1

OBJECT 1



ATTRIBUTE VALUE TABLE
****

BASE SUPPORTS	4
TOP TYPE	2
NO. STICKS	4
NO. PLATES	2
NO. BLOBS	0
NO. UPRIGHTS	5
HORIZONTALS	1
SLANTEDS	0
NO. LEVELS	3
TOP PCS. POS.	2

SIMPLE PARTS RELATIONS

SIMPT	TYPE	LENGTH	AREA	VOLUME
1	1	1.00	0.0	0.0
2	1	1.00	0.0	0.0
3	1	1.00	0.0	0.0
4	1	1.00	0.0	0.0
5	2	1.00	1.00	0.0
6	2	1.00	1.00	0.0

CONNECTS-SUPPORTS RELATIONS

SP1	SP2	SUPPORTS	HOW
1	5	TRUE	12
2	5	TRUE	12
3	5	TRUE	12
4	5	TRUE	12
5	6	TRUE	23

TRIPLES RELATIONS

SP1	SP2	SP3	SAME
1	5	2	TRUE
1	5	3	TRUE
1	5	4	TRUE
1	5	6	FALSE
2	5	3	TRUE
2	5	4	TRUE
2	5	6	FALSE
3	5	4	TRUE
3	5	6	FALSE
4	5	6	FALSE

PARALLEL RELATION

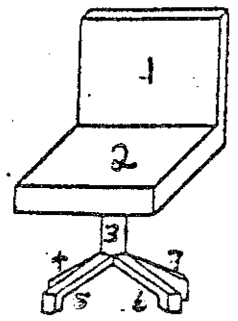
SP1	SP2
1	2
1	3
1	4
2	3
2	4
3	4

PERPENDICULAR RELATION

SP1	SP2
1	5
2	5
3	5
4	5
5	6

Figure 11 illustrates the full relational structures of objects 1 and 3 of Figure 9.

OBJECT 3



ATTRIBUTE VALUE TABLE
****

BASE SUPPORTS	4
TOP TYPE	2
NO. STICKS	5
NO. PLATES	2
NO. BLOBS	0
NO. UPRIGHTS	2
HORIZONTALS	1
SLANTEDS	4
NO. LEVELS	4
TOP PCS. POS.	2

SIMPLE PARTS RELATIONS

SIMPT	TYPE	LENGTH	AREA	VOLUME
1	2	6.00	28.26	0.0
2	2	6.00	28.26	0.0
3	1	2.00	0.0	0.0
4	1	4.00	0.0	0.0
5	1	4.00	0.0	0.0
6	1	4.00	0.0	0.0
7	1	4.00	0.0	0.0

CONNECTS-SUPPORTS RELATIONS

SP1	SP2	SUPPORTS	HOW
1	2	TRUE	23
2	3	TRUE	21
3	4	TRUE	11
3	5	TRUE	11
3	6	TRUE	11
3	7	TRUE	11
4	5	TRUE	11
4	6	TRUE	11
4	7	TRUE	11
5	6	TRUE	11
5	7	TRUE	11
6	7	TRUE	11

TRIPLES RELATIONS

SP1	SP2	SP3	SAME
1	2	3	FALSE
2	3	4	FALSE
2	3	5	FALSE
2	3	6	FALSE
2	3	7	FALSE
4	3	5	TRUE
4	3	6	TRUE
4	3	7	TRUE
5	3	6	TRUE
5	3	7	TRUE
6	3	7	TRUE

PARALLEL RELATION

SP1	SP2
1	3

PERPENDICULAR RELATION

SP1	SP2
1	2
2	3

(leg) to part 1 of object 3 (back), part 2 of object 1 (leg) to part 3 of object 3 (central leg), part 3 of object 1 (leg) to part 4 of object 3 (slanted leg) and part 4 of object 1 (leg) to part 5 of object 3 (slanted leg) and was the best match found within the time limit rather than the best possible. To study this further, we reran the experiment letting the program execute longer and obtained a better mapping that sent part 1 (leg) to part 3 (central leg), part 3 (leg) to part 4 (leg), part 5 (seat) to part 2 (seat), and part 6 (back) to part 1 (back). This mapping had a structural error of 8, completeness error of 49, and total error of 81. Since the threshold we calculated for deriving a binary relation from the relational distances was 59, even this much better mapping would not have allowed objects 1 and 3 to be called similar. The main problem is the lack of connectivity between the legs and seat of object 3 which causes differences in the connects/supports and triples relation and the slanted legs that cause differences in the parallel and perpendicular relations. The information in the attribute-value table is insufficient to detect all of these differences.

In the attribute-value table comparisons, object 12 had several more plates than the other objects, and object 14 had several more sticks than the others. Thus they were too dissimilar in heavily weighted attributes to the other objects to cluster with them. In relational matching, however, object 14 shares with objects 1, 2, 4, 5, 7, and 9 a seat and four legs in the same connection and triples relationships. In this case the relational matching is more powerful than the attribute-value table matching. Object 12 was considered similar to objects 8, 10, and 13 in the relational matching. This is really the case of a subset of object 12 having structure similar to objects 8, 10, and 13, since partial matches were allowed.

As far as object 1, it was deemed relationally similar to objects 2, 4, 5, 6, 7, 8, 9, 10, 13, and 14. Again we find that a subset of object 1 (the seat and back) has the same simple parts and same structure as a subset of objects 8, 10, and 13. Another subset of object 1 (the seat and four legs) has the same simple parts and same structure as a subset of all of objects 2, 4, 5, 6, 7, 9, and 14. This accounts for object 1 clustering with two different groups and makes intuitive sense also.

One criticism of these results might be that the total relational error, as used, was dependent on the number of N-tuples in the relations of an object. Objects with more N-tuples would necessarily generate more errors. To study the effects of this problem, we 'normalized' the total errors by dividing the error for object i vs object j by the total number of N-tuples in object i plus the total number of N-tuples in object j.

The results were, again, three clusters,

Cluster	Objects
1	1,2,4,5,6,7,8,9,10,14
2	3
3	8,11,12,13

as opposed to the former relationally obtained clusters:

Cluster	Objects
1	1,2,4,5,6,7,9,14
2	3
3	1,8,10,11,12,13.

These results are, of course, somewhat dependent on the threshold that produced the binary relation to be clustered. The threshold was again chosen so that the binary relation would include the same number of pairs as the previous two binary relations.

One other set of comparisons were made to help study the use of the attribute-value-table-based clusters. For each object, the five best relational matches with other objects

	With Clustering			Without Clustering		
Candidate	Structural Error	Completeness Error	Object	Structural Error	Completeness Error	Object
1	0	27	2	0	27	2
	0	7	4	0	7	4
	0	16	6	0	16	6
	0	29	9	0	29	9
	0	47	14	0	39	8
2	0	27	1	0	27	1
	0	34	4	0	34	4
	0	43	6	0	66	8,10
	6	54	9	0	43	6
	3	62	7			
3	3	42	7	2	36	8,10
	6	52	4	6	52	4
	30	49	1	3	42	7
	52	64	9	6	43	6
	6	43	6			
4	0	7	1	0	7	1
	0	9	6	0	9	6
	0	22	9	0	22	9
	0	34	2	0	34	2
	0	40	14	0	40	14
5	0	7	1	0	7	1
	0	9	6	0	9	6
	0	22	9	0	22	9
	0	40	14	0	40	14
	0	34	2	0	34	2

Table 4. Best 5 Matches with and without Clustering

	With Clustering			Without Clustering		
Candidate	Structural Error	Completeness Error	Object	Structural Error	Completeness Error	Object
6	0	9	4	0	9	4
	0	16	1	0	16	1
	0	43	2	0	43	2
	0	31	9	0	31	9
	0	49	14	0	49	14
7	0	72	14	0	72	14
	1	56	9	1	56	9
	3	19	6	3	19	6
	3	28	5	3	28	5
	3	35	1	3	35	1
8	2	12	10	2	12	10
	24	8	11	0	20	13
	0	48	12	1	52	9
	0	20	13	0	48	12
				0	62	2
9	0	22	4	0	22	4
	0	29	1	0	29	1
	0	31	6	0	31	6
	3	50	7	1	52	8
	6	54	2	2	52	10
10	2	12	8	1	68	14
	14	6	11	2	20	13
	2	48	12	2	36	3
	2	20	13	2	12	8
				2	35	1

	With Clustering			Without Clustering		
Candidate	Structural Error	Completeness Error	Object	Structural Error	Completeness Error	Object
	24	10	8	1	20	13
	2	12	10	2	12	10
11	12	52	12	12	23	6
	1	20	13	12	24	7
				12	32	4
	0	48	8	0	48	8
	3	50	10	0	45	13
12	13	52	11	3	50	10
	0	45	13	13	52	11
				48	51	6
	0	20	8	0	20	8
	2	20	10	0	45	12
13	1	20	11	1	20	11
	0	45	12	2	20	10
				12	70	2
	0	40	4	0	40	4
	0	47	1	0	47	1
14	0	49	6	0	49	6
	14	60	9	1	68	10,11
	1	74	7	1	74	7

in its own cluster were compared to the five best relational matches with any other objects. These comparison are shown in Table 4. For four of the objects, the results were the same. For four of the objects, one out of the five best matches was with a different object (one not found in the clusters deemed similar enough). For four of the objects, two out of the five best matches were with different objects. For the two remaining objects, three out of the five best matches were with different objects. This seriously questions the use of the attribute-value clusters.

VII. Conclusions

We have defined a relational model to be used as a rough description of a three-dimensional object. A database of such models has been constructed and used in a preliminary set of matching experiments. The database is currently organized into clusters of objects with similar features, based on the Euclidean distance between their attribute-value tables. When an unknown object is analyzed, its attribute-value table is compared with the centroid of each such cluster and relational matching takes place against the objects in those clusters deemed similar enough. The relational matching produces a mapping from the unknown object to the model plus a measure of their relational distance.

Our experiments comparing relational distance to Euclidean 'feature' distance showed that they are related, but not similar enough to trust the attribute value clusters. Clustering is a viable alternative, but it should be based on relational clusters. This relational grouping introduces some important new problems to study. In particular, how can we define the centroid of a relational cluster and how do we match an unknown object against the centroid description? These and other methods of reducing the number of models that participate in full relational matching are crucial to the use of a large object database. Thus, the results of the experiments reported here are to define important new work for the future.

Bibliography

1. Agin, G. J. and Binford, T. O., "Computer Description of Curved Objects", Third International Joint Conference on Artificial Intelligence, Stanford, August 1973.

2. Agin, G. J., "Hierarchical Representation of Three-Dimensional Objects Using Semantic Models", in [52].

3. Badler, N. and Bajcsy, R., "Three-Dimensional Representation for Computer Graphics and Computer Vision",

4. Ballard, D. H., Brown, C. M. and Feldman, J. A., Outline of a Query-Driven Vision System, Computer Science Dept., University of Rochester, October 1976.

5. Barrow, Ambler, and Burstall, "Some Techniques for Recognizing Structures in Pictures", in Frontiers of Pattern Recognition, ed. S. Watanage.

6. Barrow, H G. and Tenenbaum, J. M., "MSYS: A System for Reasoning About Scenes", SIR AI Tech. Report. 121, 1976.

7. Binford, T. O., "Visual Perception by Computer", IEEE Conference on Systems and Control, Miami, 1971.

8. Chakravarty, I., "A Generalized Line and Junction Labeling Scheme with Applications to Scene Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-1, No. 2, April 1979, pp. 202-205.

9. Chien, R. T. and Selander, J. M., "On the Use of Graph Models for Representation and Object Recognition", in [44].

10. Clowes, M. B., "On Seeing Things", Artificial Intelligence 2, No. 1, 1971, pp. 79-116.

11. Davis, L. S., "Shape Matching Using Relaxation Techniques", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAMI-1, No 1, January 1979, pp. 60-72.

12. Freuder, E. C., "Synthesizing Constraint Expression, CACM, Vol 21, No 11, 1978.

13. Gaschnig, J., "A General Backtrack Algorithm that Eliminates Most Redundant Tests", Proceedings of the

5th International Joint Conference on Artificial Intelligence, 1972.

14. Hanson, A. and Riseman, E., "Segmenttion of Natural Scenes", in Computer Vision Systems, A. Hanson and E. Riseman eds, Academic Press, New York, 1978, pp. 129-163.

15. Haralick, R. M. and Elliot G., "Increasing Tree Search Efficiency for Constraint Satisfaction Problems", Procceding of the 6th International Joint Conference on Artificial Inteligence, 1979.

16. Haralick, R. M. and Kartus, J., "Arrangements, Homomorphisms, and Discrete Relaxation", IEEE Transcations on Systems, Man, and Cybernetics, Vol SMC-8, August 1978, pp. 600-612.

17. Haralick, R. M. and Shapiro, L. G., "The Consistent Labeling Problem: Part II", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAMI-1, No. 2, April 1979, pp. 173-184.

18. Hollerbach, J. M., "Hierarchical Shape Descriptions of Objects by Selection and Modification of Prototypes", M.I.T. AI Lab, 1975.

19. Huffman, D. A., "Impossible Objects as Nonsense Sentences", in Machine Intelligence 6, Edinburgh University Press, 1978, pp. 295-323.

20. Kanade, T. Recovery of the Three-Dimensional Shape of an Object from a Single View, CMU U-CS-79-153, Dept. of Computer Science, Carnegie-Mellon University, Oct. 1979.

21. Mackworth, A., "Consistency in Network of Relations", Artificial Intelligence, Vol 8, 1977, pp. 99-118.

22. Marr, D. and Nishihara, H. K., Spatial Disposition of Ahes in a Generalized Cylinder Representation of Objects that do not Encompass the Viewer, MIT AI Lab, Memo No 341, December 1975.

23. McKee, J. W. and Aggarwall, J. K., Computer Recognition of Partial Views of Three-Dimensional Curved Objects, Tech Report No 171, Information Systems Research Lab, University of Texas at Austin, May, 1975.

24. Minsky, M. "A Framework for Representing Knowledge", in The Psychology of Computer Vision, P.H. Winston, ed., New York, McGraw-Hill, pp. 211-277.

25. Montanari, U., "Networks of Constraints: Fundamental Properties and Applications to Picture Processing", Information Sciences, Vol 7, 1974, pp. 95-132.

26. Nagao, M., Hachimoto, S. and Sakai, T., "Automatic Model Generations and Recognition of Simple Three-Dimensional Bodies", Computer Graphics and Image Processing 2, 1973, pp. 272-280.

27. Nevatia, R. and Binford, T. O., "Description and Recognition of Curved Objects", Artificial Intelligence 8, 1977, pp. 77-90.

28. Richard, C. W. and Hemami, H., "Identification of Three-Dimensional Objects Using Fourier Descriptors of the Boundary Curve", IEEE Transactions on Systems, Man, and Cybernetics, Vol SMC-4, No. 4, July 1974, pp. 371-378.

29. Roberts, L. G., Machine Perception of Three-Dimensional Solids, in Optical and Electro-Optical Information Processing, Tippitt et. al. ed, MIT Press, Cambridge, Mass., 1965, pp. 159-197.

30. Rosenfeld, A., Hummel, R. A., and Zucker, S. W., "Scene Labeling by Relaxation Operations", IEEE Transactions on Systems, Man, and Cybernetics, Bol CMS-6, No 6, Jue 1976, pp. 420-433.

31. Salton, Gerard, Dynamic Information and Library Processing, Englewood Cliffs: Prentice Hall, 1975.

32. Schneier, M., "A Compact Relational Structure Representation", in [44]

33. Shapira, R. and H. Freeman, "Computer Description of Bodies Bounded by Qundratic Surfaces from a Set of Imperfect Projections", IEEE Transactions on Computers, Vol C-27, No 9, Sept 1978, pp. 841-854.

34. Shapiro, L. G. and Haralick, R. M., "A General Spatial Data Structure", IEEE Conference on Pattern Recognition and Image Processing, June, 1978.

35. Shapiro, L. G., "A Structural Model of Shape", IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, No. 2, March 1980, pp. 111-126.

36. Shapiro, L. G. and Haralick, R. M., "Structural Descriptions and Inexact Matching", to appear in IEEE Transactions on Pattern Analysis and Machine Intelligence, 1981.

37. Shapiro, L. G. and Haralick, R. M., "Decomposition of Two-Dimensional Shapes by Graph-Theoretic Clustering", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 1, Jan 1979, pp. 10-20.

38. Shapiro, L. G., Moriarty, J. D., Mulgaonkar, P. G., and Haralick, R. M., "A Generalized Blob Model for Three-Dimensional Object Representation," IEEE Workshop on Picture Data Description and Management, Asilomar, CA, Aug 1980.

39. Soroka, B. I. "Generalized Cylinders from Parallel Slices", Proceedings of the IEEE Conference on Pattern Recognition and Image Processing, Chicago, 1979, pp. 421-426.

40. Turner, K.J., Computer Perception of Curved Objects Using a Television Camera, Ph.D., dissertation, School of Artificial Intelligence, Edinburgh University, 1974.

41. Ullman, J. R., "Associating Parts of Patterns", Information and Control, Vol 9, 1966, pp. 583-601.

42. Ullman, J. R., "An Algorithm for Subgraph Homomorphisms", JACM, Vol 23, January 1976, pp. 31-42.

43. Waltz, D. "Understanding Line Drawings of Scenes with Shadows", ed. P. Winston, McGraw-Hill, New York, 1975, pp. 19-91.

44. Workshop on the Representation of Three-Dimensional Objects, R. Bajcy, Director, University of Pennsylvania, Philadelphia, May 1-2, 1979.

45. Zucker, A. W. and Hummel, R. A., Computing the Shape of Dot Clusters I: Labeling Edge Interior, and Noise Points, Tech. Rep. TR-543, University of Maryland, May 1977.

46. Zucker, S. W. Leclerc, Y. G. and Mohammed, I. L., Continuous Relaxation and Local Maxima Selection: Conditions for Equivalence, Tech. Rep. No 78-15R, Computer Vision and Graphics Laboratory, McGill University, December 1978.