COGNITIVE SCIENCE 7, 67-93 (1983)

A Parallel-Process Model of Mental Rotation*

BRIAN V. FUNT

Department of Computing Science
Simon Fraser University
Vancouver, British Columbia V5A 1S6

It is argued that some of the phenomena identified with analog processes by Shepard can be understood as resulting from a parallel-process algorithm running on a processor having many individual processing elements and a restricted communication structure. In particular, an algorithm has been developed and implemented which models human behavior on Shepard's object rotation and comparison task. The algorithm exhibits computation times which increase linearly with the angle of rotation. Shepard found a similar linear function in his experiments with human subjects. In addition, the intermediate states of the computation are such that if the rotation process were to be interrupted at any point, the object representation would correspond to that of the actual object at a position along the rotation trajectory. The computational model presented here is governed by three constraining assumptions: (a) that it be parallel; (b) that the communication between processors be restricted to immediate neighbors; (c) that the object representation be distributed across a large fraction of the available processors. A method of choosing the correct axis of rotation is also presented.

INTRODUCTION

The Shepard and Metzler (1971) mental rotation result is puzzling in that although there are many obvious algorithms for comparing two objects which might provide a computational model of the mental rotation process, none is naturally constrained to behave so that its computation time will increase

*I would like to thank Leigh Hunt Palmer of the Physics Department for his help with the moments of inertia, Karen Funt for her comments on earlier drafts, Edwin Bryant for developing the line-drawing interpretation algorithm, and Thomas Strothotte for much interesting discussion. The referees' comments and criticisms were particularly thoughtful and valuable.

The financial support of the Natural Sciences and Engineering Research Council of Canada under grant A4322 is gratefully acknowledged.

linearly with the angular disparity between the objects. In the Shepard-Metzler experiment, subjects are asked to determine whether two objects presented in a line drawing similar to that of Figure 1 are the same or different. What is found is that in cases where the objects are identical, the subject's reaction time increases in proportion to the angle which would be required to actually rotate one object into the other in three dimensions. This is the case whether or not the rotation is in the picture plane or in depth.

The claim of this paper is that the linear-reaction-time result follows naturally from the inherent constraints of parallel processing. These constraints, with their effect on reaction time, are part of a computational model of the mental rotation process which has been implemented as a computer program. It differs in two fundamental respects from the imagery model of Kosslyn and Shwartz (1977). Theirs is based on sequential rather than parallel processing, and two-dimensional rather than three-dimensional representations. It also contrasts with the standard matrix-multiplication technique commonly used for rotation in computer graphics systems which involves sequential processing and does not provide an adequate model for the Shepard and Metzler data.

In considering the design of a computational model for mental rotation, we begin with the assumption that it should be based on parallel processing. Parallelism provides speed, and there is also a great deal of neurophysiological evidence of parallelism, particularly in the visual system (Hubel & Wiesel, 1979).

Parallelism, in turn, implies a need for a representation in which information is distributed across many processors. In other words, each processor must have some piece of information to process—if only one processor has access to information then the processing is once again sequential. This argument applies regardless of the problem that the processing system is solving. No matter what the task, if a group of processors is to cooperate in handling it, each must have access to a piece of relevant information. As a simple example of parallel processing with distributed information, consider a theater full of people where the person on stage asks the audience, "Does anyone know how long Frédéric Chopin lived?" Each person acts as an independent processor and has access to information in his own memory which might be relevant to the question. Note that there is no single memory store in which all the information known collectively by those in the theater is stored; rather it is distributed across a number of individual memories to which only each individual "processor" has access.

A feasible implementation of a parallel processor implies restricted communication between processors. To have every processor communicate directly with every other processor would require a communication link (wire, neural connection) between every processor pair, or for P processors,

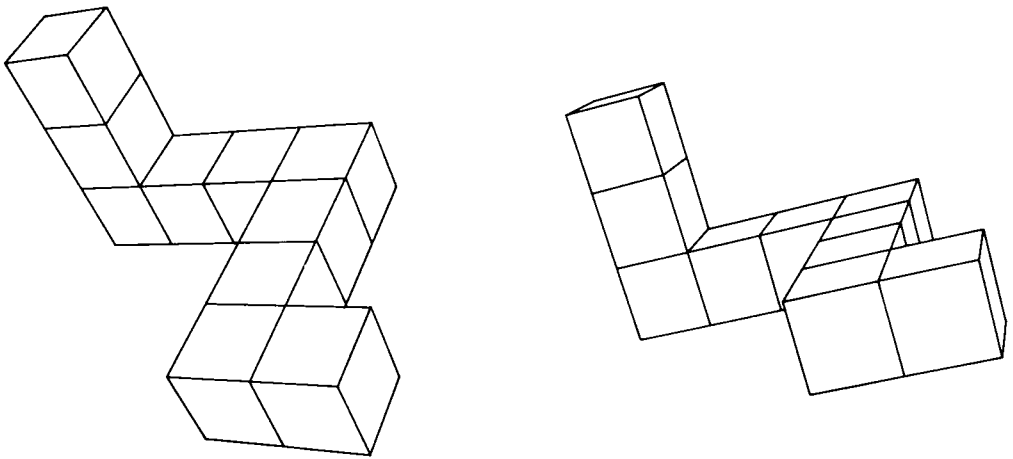

Figure 1. An object similar to the ones used by Shepard. This pair shows a rotation in depth.

P (P-1)/2 two-way links. A reasonable restriction is that each processor is linked directly to only its immediate neighbors.

Finally, the combined effects of distributed representation and restricted neighborhood communication imply a constraint on the speed at which a given type of problem can be solved. This constraint is the number of messages which must be passed between processors and the number of times they must be relayed by intermediate processors. Again using the example of Chopin's lifespan, we see that if one person knows when Chopin was born while another knows when he died, then by one communicating his information to the other, or by them both communicating to a third party, the problem can be solved. If each person, rather than shouting out what he knows, says to his neighbors, "Chopin's birth/death was... Pass it on", then when anyone has heard both messages he can compute the correct answer. The time taken before someone has heard both messages is going to be a linear function of the "distance" (in terms of communication links) between the message originators.

SYSTEM DESCRIPTION OVERVIEW

The computer program takes as input two two-dimensional line drawings of the kind used in the experiment by Shepard and Metzler (1971), and determines whether or not they depict the same or different objects. The number of operations performed by the program, and hence its "reaction-time", increases linearly with the angular separation of the two objects. The program provides a computational model of mental rotation; namely, a parallel-processing algorithm describing mental rotation as a process. Beyond the specification that the algorithm must be carried out by a collection of processors acting in parallel with neighborhood restricted inter-processor communication, the model does not rely on the exact characteristics (e.g., speed, shape, material) of the hardware executing the algorithm.

The way in which three-dimensional objects are represented can be envisioned by imagining a hollow sphere with an object inside it. The sphere's surface is dotted with processors. A spoke (radial line) from a processor to the center of the sphere may pierce the object's surface several times. Each processor stores the distances along its spokes to the places where it and the surface intersect. Since each processor knows its own location relative to the center of the sphere, the radial distance to a point fully determines the point's location. In this representation the information describing the object's shape is distributed over the processors with each holding only a piece of the total.

When the line-drawing images of two objects are input, the program first extracts their three-dimensional structure. This three-dimensional in-

formation is then represented in what amounts to spherical coordinates by a set of processors spread evenly over the surface of a sphere. Each processor can directly communicate only with its immediate neighbors on the sphere, and holds information about only a small piece of each object. From the three-dimensional shape information, the program then computes a canonical set of axes for each object. Except for some highly symmetrical objects, this procedure, which is based on the concept from physics of an object's principal moments of inertia, provides a unique specification of the object's orientation. From the canonical axes of the two objects, the program calculates the parameters of the rotation which will take one object into the other. These are the rotation's axis, direction, and magnitude.

It is important that the program be able to compute the proper axis of rotation, for without it some sort of search process would have to be invoked. And the search space, hence the search time, would most certainly not grow linearly with the angular difference in the objects orientations. It is difficult to accept Shepard's linear reaction-time results as evidence for his hypothesis that mental rotation is an analog process without a corresponding hypothesis about the method a subject uses to choose the correct axis of rotation in a constant (or linearly increasing) amount of time. The moments-of-inertia technique is one possible constant-time method.

Once the axis of rotation has been determined, the program begins the rotation. It proceeds in a number of small steps. The term " rotation" here means that information flows from processor to processor in such a way that the shape of the object is preserved. In other words, the shape information is passing between processors so that at all times each piece of shape information maintains a fixed distance (measured in terms of intermediate processors) from every other piece of shape information. To accomplish this, each processor computes the effect of a small rotation increment on the point it holds. To do so it must determine the point's new location and if it is outside the local area of the processor, send the information about that point as a message to the appropriate neighbor. The correct neighbor is the one whose domain encompasses the new point's location. During a rotation, the processors along the "equator" will be constantly sending messages, while those at the "poles" will send none.

The comparison of the two objects is started after the rotation is completed. Initially the two objects, A and B, were represented as points stored in many separate processors. The processors holding information about A were different from those holding information about B because A and B were at different orientations. The effect of rotating A into alignment with B is to shift the information about A, so that its description is held by the same set of processors as hold information about B. To compare the shape of A and B each processor independently checks whether its information about A corresponds to that about B. If there are no major discrepancies

then the program concludes that A and B are similar objects. Since each processor makes its comparison independently, establishing the similarity of the two objects is a parallel computation.

THE COMPUTATIONAL MODEL IN DETAIL

The Representation of Three-Dimensional Objects

Earlier work involving the parallel processing of two-dimensional shapes (Funt, 1980) led to the question of how parallel processing could be applied in three dimensions. The computer problem-solving system, WHISPER, used its parallel processing "retina" shown in Figure 2 to look at a diagram relevant to whatever problem it was currently solving. The retina was very effective in rotating, comparing, and analyzing the two-dimensional shapes which appeared in the diagrams. Each "circle" in Figure 2 represents a processor, and each processor is marked if it lies over part of an object when the retina is superimposed on a diagram. The size and location of each circle represents the portion of the diagram from which it receives an input. The actual physical size and location of the processor is independent of the logical size and location of the circles.

An organized message-passing regime is sufficient to carry out a rotation. To rotate clockwise each processor checks whether it is marked—if not, it does nothing—and if so it simply sends a "mark" message to its neighbor in the clockwise direction while erasing its own mark. Because the processors are aligned along radial lines, the angle between any two circles in the same concentric ring is always constant. As a consequence, a simple shift of information between processors results in a uniform rotation. Trehub (1977) has developed a neural network which rotates two-dimensional shapes.

It might appear that the two-dimensional rotation technique could be straightforwardly generalized to three dimensions by making every circle a sphere and filling out a sphere of spheres in the same pattern as on the two-dimensional retina. But this does not work. Such a collection of spheres cannot be packed into three-space so that there is an equal angular separation between all neighboring spheres. Other likely generalizations are also not physically realizable. For instance, it is not possible to spread dots over the surface of a sphere so that there is an equal angular separation between the dots for rotations about two independent axes. The one axis case is possible, and corresponds to putting a processor at the intersection points of the longitude and latitude lines on a globe. To accommodate the general class of three-space rotations, however, one axis is insufficient.

With somewhat more complex computations than the simple shifts of the two-dimensional case, three-dimensional rotations can still be accom-

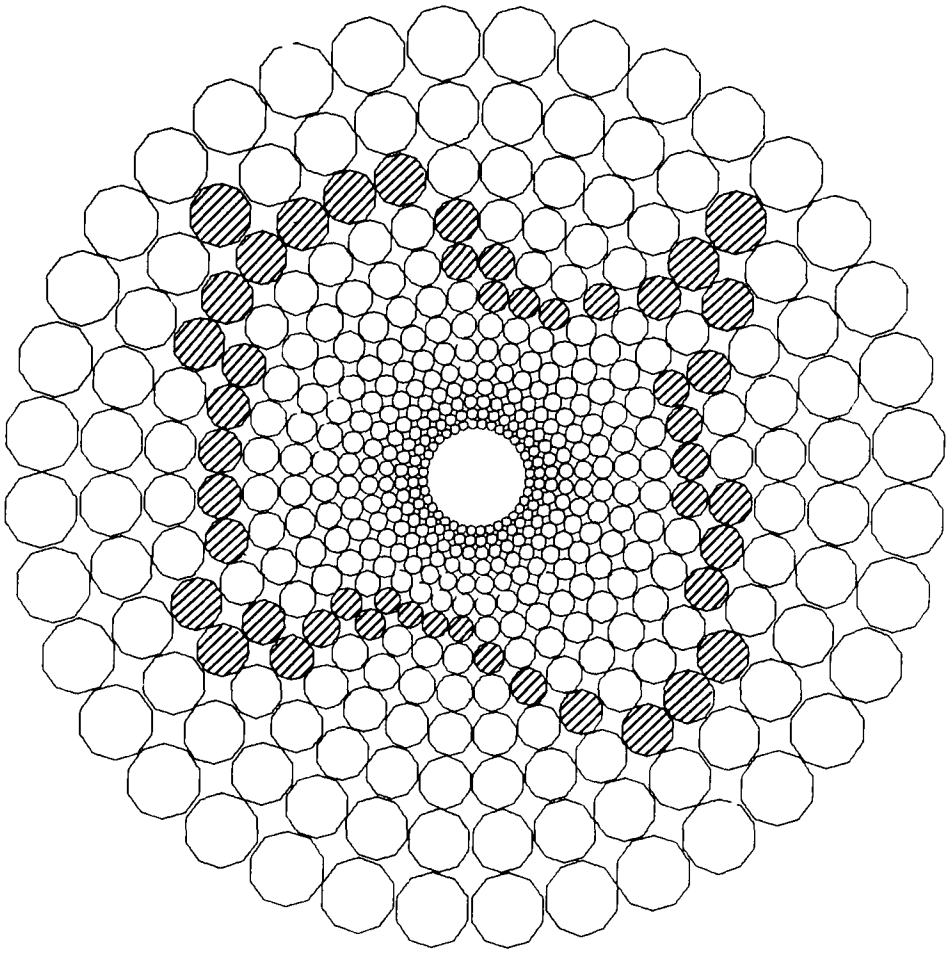

Figure 2. WHISPER'S parallel processing retina. Each "circle" represents a processor. The shaded ones are processors which are marked to represent the contour of a two-dimensional object.

plished by neighborhood message passing. The optimal arrangement for the processors is a uniform distribution over the surface of a sphere. The uniform processor distribution results in a uniform grain of resolution in the object representation. The processors are also better utilized covering the sphere, rather than filling it, because we are only interested in representing the surface, not the interior of an object. A full three-dimensional array of processors with each processor representing a point on the object's surface would be very sparsely filled with data, and most processors would have nothing to do.

The optimal arrangement for the processors would be to have them spread uniformly over the surface of a sphere, but this requires a uniform

tesselation of a sphere. Unfortunately, there does not exist such a uniform tesselation with more than 20 tiles. Geodesic domes provide a good approximation. At most nodes of the geodesic dome, six triangles share a common vertex. By merging the triangles at every third node of the dome, we obtain a set of hexagonal tiles which almost completely covers it. When the basis for the dome is an icosohedron, there will be 12 locations requiring pentagonal tiles as shown in Figure 3 (Kenner, 1976).

The fact that the processors are to be logically spread over the surface of a sphere does not mean that they must physically form a sphere. The logical and physical topologies are entirely separate. The physical topology simply requires that each processor be linked by communication lines to six (five for the 12 processors corresponding to the pentagonal tiles) other pro-

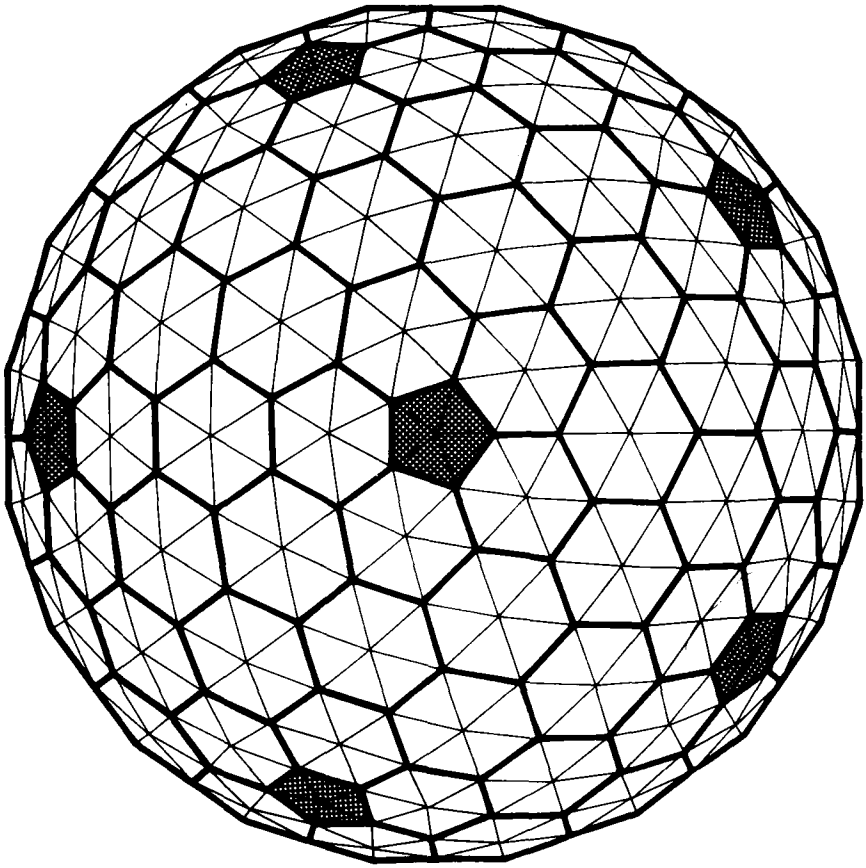

Figure 3. Geodesic dome with a processor located at every third vertex. The result is a set of mainly hexagonal tiles as shown by the thick lines. The visible pentagons are shaded. In the simulation, the density of processors is actually approximately three times that shown.

cessors. These others are neighbors strictly by virtue of their communication links. One natural, but by no means only, physical layout would be a hexagonally packed array where each processor communicates with its geographically closest neighbors. In addition to the six neighbor links, every processor is on a single communication bus (party line) to a special supervisory processor. All processors are capable of completely general computations (i.e., are Turing Machine equivalent) and have their own local memory storage.

The way in which three-dimensional objects are represented can be envisioned if you image a hollow sphere dotted with processors and an object inside it. Figure 4 shows a cross-sectional view of this situation. Each processor stores the radial distance to the object's surface as it passes through

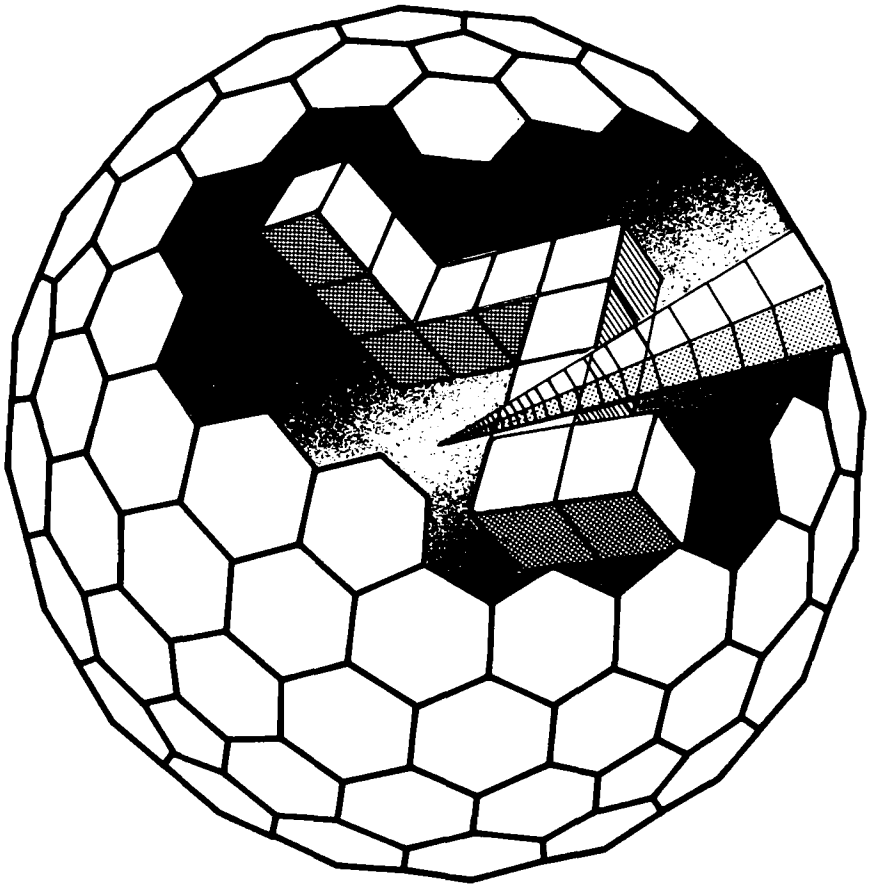

Figure 4. A cut-out view of the sphere of processors with an object inside of it. Each processor is responsible for representing the part of the object within its "cone." One of the 482 cones used in the simulation is shown.

its "cone." As we can see from the figure, the surface may pass through the cone several times, so there may be several radial values for a processor to store. In other words, all the surfaces between the processor and the sphere's center are represented including those that would be visually hidden from the processor. Each cone is divided into a finite number of discrete segments. For each cone segment the surface passes through, the radial distance from the sphere's center to the cone segment's center is stored. Where the surface enters and leaves the cone segment does not affect the value. Associated with each processor is its longitude ($\varnothing$) and latitude (Θ) coordinates. In conjunction with the radial values, r, for the cone segments, this gives a full specification of a sampling of points on the objects's surface in spherical coordinates (r, Θ, $\varnothing$).

Initially the axis of a processor's cone will be from the center of the sphere through the center of the processor, but during rotations the axis will change. A set of local coordinates ($d\Theta$, $d\varnothing$) are kept which represent the offset of the axis from the center of the processor. The actual point where the axis intersects the sphere is then ($\Theta + d\Theta$, $\varnothing + d\varnothing$). This local coordinate scheme is derived from a similar one used by Baker (1973) for handling rotations of two-dimensional shapes in Cartesian coordinates.

Rotations about a Known Axis

An object is said to rotate as the information about it moves from processor to processor. Each processor represents a different location in three-space, so the information travelling between processors represents the motion of a three-dimensional object. The problem is to organize the information flow so that what is represented is a rigid rotation.

Due to the neighborhood communication restriction, the full rotation must be broken down into a series of incremental rotations. At each increment, a point must move no further than the distance represented by two neighboring processors. The axis of rotation remains the same for all increments.

The algorithm for carrying out a rotation about a known axis is as follows:

1. The supervisory processor computes the coordinates of the rotation's "north pole", the point where the axis of rotation intersects the sphere of processors.
2. The supervisor broadcasts the coordinates of this pole point to all the other processors over the party line communication bus.
3. Each processor computes what its coordinates would be if the pole point were to be moved to (1, 0, 0) (i.e., the north pole on a unit

sphere). We will call these new hypothetical coordinates a processor's "virtual" coordinates.

4. The supervisory processor computes the number, N, of rotation increments of magnitude R required to carry out the full rotation.
5. The number N is broadcast to all processors.
6. Each processor does steps 6a through 6e N times.
   a. Each processor increments the $\varnothing$-component of its local coordinates of its cone axis by the same amount, R.
   b. Each processor checks whether the new coordinate of its cone axis is outside the bounds of its region of the sphere.
   c. If it is not outside, then back to step 6a.
   d. If it is outside, then the neighbor whose region the cone axis now must lie in is found, and a message is sent to it giving (a) the coordinates of the cone axis, and (b) all the radial values in the cone.
   e. When a processor receives a message, it computes the local coordinates of the new cone axis relative to its own center, and stores the incoming radial values.

The effect of step 3 is to introduce a new coordinate system in which the rotation is particularly easy. In spherical coordinates, the coordinates of any point $(r, \Theta, \varnothing)$ after a rotation of $\varnothing'$ about an axis through (1, 0, 0) are simply $(r, \Theta, \varnothing + \varnothing')$. The new coordinate system is introduced so that the axis of rotation passes through (1, 0, 0). This simplifies the computation of the new coordinates for a point after a rotation increment. Only one addition operation is required for step 6a.

Notice in steps 6a through 6e, that although a processor may be storing several radial values for points lying within its cone, it nonetheless does not need to consider their coordinate transformations individually. Rotation about the center of the sphere does not affect their radial values, and each has the same Θ and $\varnothing$ value.

The rotation increment, R, is fixed for the processor sphere. It depends on the total number of processors present on the sphere. The more processors on the sphere, the more processors there will be around the "equator", and hence the smaller the angular separation between processors on the equator. During a rotation, it is the points at the equator which will be moving the fastest and this will be reflected in the highest rate of message passing. The maximum rate is one message per rotation increment, so R is restricted to being no larger than the angular separation between processors on the equator. The time to compute the new coordinates for a point and to send a message is the same for each rotation increment, and these rotation increments occur sequentially. Therefore the total time to carry out a full rotation is a linear function of the number of rotation increments, which, in turn, is a linear function of the angle of the full rotation.

Calculating the Axis of Rotation

Prior to rotation, the descriptions of the two objects are differently dispersed over the set of processors because the objects are in different initial orientations. Before they can be compared, they must first be aligned, and in order to align them the axis of rotation must be known. As we have already seen, the alignment process corresponds to and represents the rotation of an object in three-space. To determine the axis of rotation, the principal moments of inertia of the two objects are calculated. Corresponding to them are principal axes whose directions are also found. The principal axes of a non-symmetrical object are unique, so if two objects are the same they will only match when their principal axes are aligned. If aligning the principal axes does not result in a match, then the objects must not have the same shape. Incidentially, the non-uniqueness of the principal axes for symmetrical objects does not pose a serious problem, since it simply reflects the fact that two identical symmetrical objects match in many orientations. The problem of aligning two objects is therefore reduced to the problem of aligning their principal axes.

Moments of inertia have been previously used in recognizing two-dimensional patterns (Hu, 1977) and in matching two-dimensional contours (Baumgart, 1974). More recently Smith (1979) has also thought of using them for three-dimensional matching. The physical concept of moments of inertia is explained in Goldstein (1950), and the method of solving for principal axes is outlined in Wells (1967). Basically, the moments-of-inertia concept is a generalization of the concept of center of mass.

The principal axes are computed from the moments and products of inertia. Relative to the x, y, z-axes of our frame of reference these are defined by six equations of the form:

$$Ix = \int (y^2 + z^2)\, dm \quad \text{(moments of inertia about the x-axis)}$$
$$Ixy = \int (xy)\, dm \quad \text{(product of inertia)}$$

The discrete approximation to these equations can be computed in parallel by the sphere of processors. Each processor assigns each of its points a mass proportional to the square of its radial value (the surface area which a point represents increases with its distance from the center of the sphere), and then computes the total contribution of all its points to each of the six inertial summations. Since this calculation involves only local information, all processors can compute their contributions simultaneously.

It should be mentioned that the principal axes of an object must be computed relative to its center of mass. To facilitate this, the sphere of processors represents an object with its center of mass at the center of the

sphere. As the three-dimensional information is extracted from the two-dimensional input figure, it is stored so that this will be the case. All rotations are, therefore, about the center of mass of an object, and any two objects always have their centers of mass aligned in the representation. Clearly, if two objects are to be aligned so that they exactly match they must have corresponding centers of mass, so this is reasonable. The center of the sphere of processors is also the origin of the frame of reference coordinate system, and it will be at the origin of the coordinate systems defined by the principal axes. To dispel any possible confusion, perhaps it should be reiterated that an object's representation is dependent on its orientation, but independent of its position. Two objects physically separated in space are stored in the sphere of processors as if they occupied the same space inside the sphere.

The axis of rotation is easily computed from the two sets of principal axes. Let P and Q be points on the x- and y-axes of one object and P ′ and Q ′ be corresponding points on the other. The proper rotation is one which takes P into P ′ and Q into Q ′. Both vectors (P ′-P) and (Q ′-Q) lie in planes which are perpendicular to the axis of rotation as shown in Figure 5. The axis of rotation must therefore be perpendicular to both these vectors. Thus the axis of rotation is a line through the origin—the center of mass of the two objects—with the same direction as the vector $(P' - P) \times (Q' - Q)$.

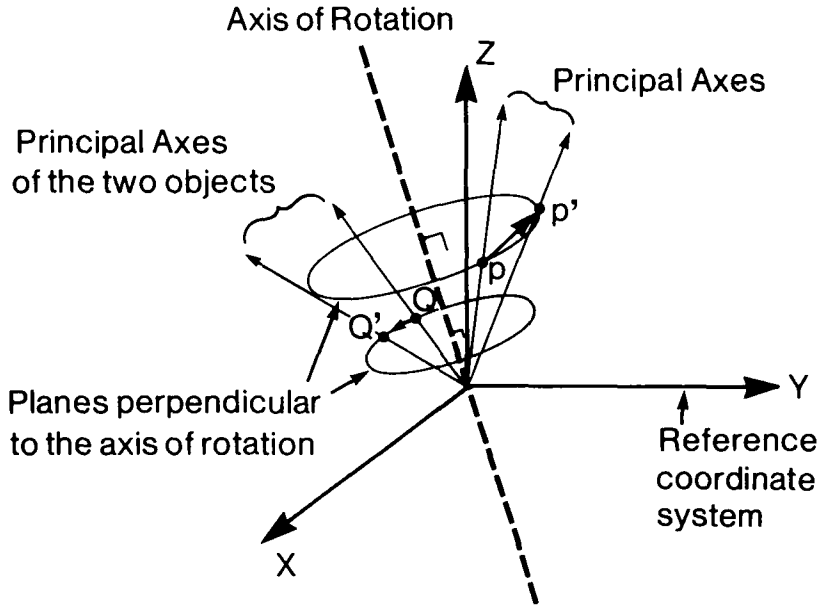

Figure 5. The axis of rotation calculation. Its direction is $(P' - P) \times (Q' - Q)$.

The angle of rotation can also be determined. From Figure 6 we see:

$$\sin (a/2) = |v| / 2d$$

or

$$a = 2 \arcsin (|v| / 2d)$$

where

$$d = |P| \sin (b)$$

and

$$b = \arccos ((P \bullet A) / (|P| |A|)$$

One detail which did not become apparent until the program was being tested concerns an ambiguity in the sign of the principal axes. It is possible to align the positive x-axis of one object's set of principal axes with either the positive or negative x-axis of the other. A similar possibility holds for the y- and z-axes. Of the eight combinations, half are eliminated because they result in coordinate systems of opposite handedness, which of course could never be aligned. Each of the four remaining ways of associating the two sets of axes would call for a different rotation in order to align the axes. The ambiguity is eliminated by recourse to the third order moments of inertia which are of the form:

$$\int x^3 dm; \quad \int y^3 dm; \quad \int z^3 dm.$$

These moments are useful because they are orientation dependent. The program choses the association of the axes which will result in the best match of the third moments of the two objects after rotation. It might appear that this would require four separate object rotations, but we are not forced into that because four differently oriented coordinate systems can be used instead. In other words, we simply use the same third-moment computation, but with the parameters changed to reflect the orientation of the coordinate system, to predict what the third moments of the object would become when it is actually rotated. To pick the correct axis association and determine the correct rotation for the objects therefore involves four separate third-moment calculations (one relative to each coordinate system), and a comparison of the results.

The Comparison Test

Once object A has been rotated so that its principal axes are aligned with those of B, the shape of the two objects is compared. First A is matched to B and then vice versa. Each processor has a set of zero or more radial values for A and another set for B. The two sets match if they have the same cardi-

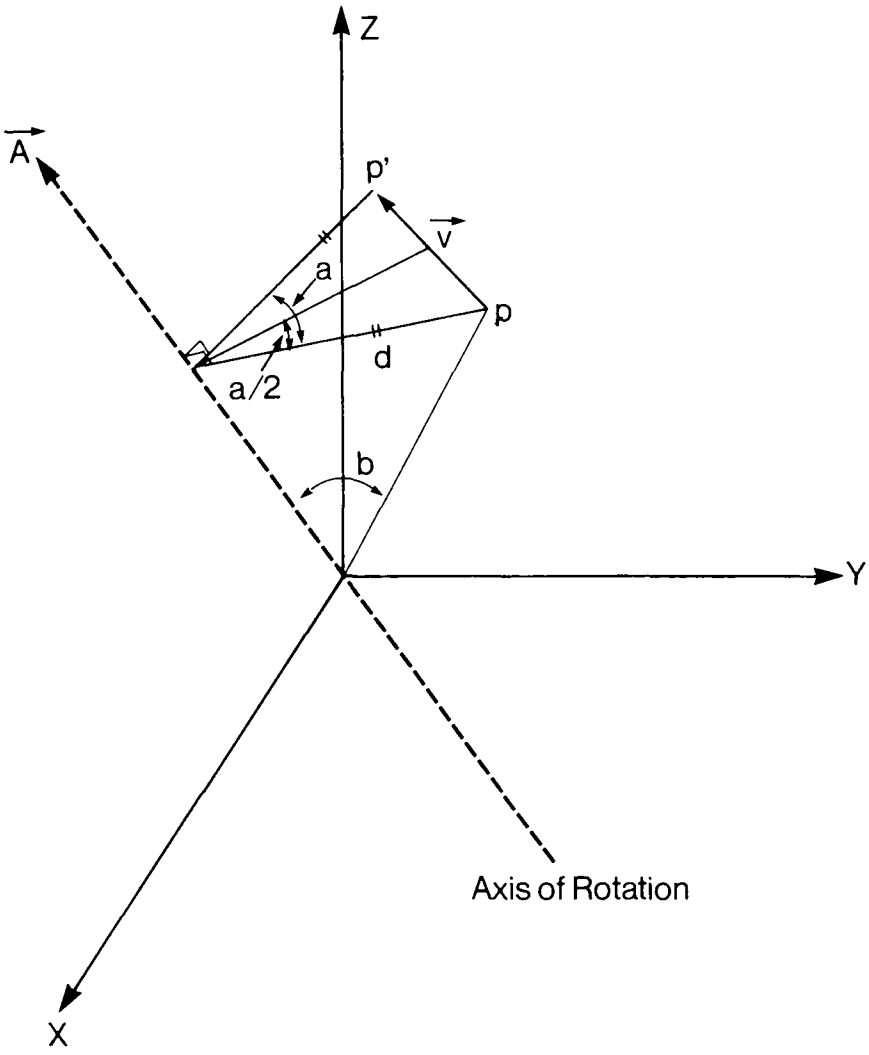

Figure 6. The angle of rotation calculation. Note that d is the perpendicular distance from P to the axis.

nality and their elements correspond within a fixed tolerance. If processor P's set for A does not match its set for B, then P asks its immediate neighbors about B to see if a match can be found. Each neighbor, N, sends its set of B's radial values to P and then P matches it to its own A set. This neighborhood test is necessary because a very slight shift in the coordinates of a point might cause it to be handled by a neighboring processor, so points which are very closely but not exactly aligned may be found at neighboring

processors. Of all the processors which have a non-empty set of radial values for A, some fraction of them will find a corresponding set of radial values for B. If the fraction surpasses a threshold then A matches B. As can be seen from this brief description, simple local comparisons performed in parallel suffice for testing the similarity of two objects.

THE LISP SIMULATION

Test Results

To test the computational model of the preceding sections a LISP program was written and then run on the pairs shown in Figures 7 through 9. It determines that a rotation of -27.2776 degrees will align the principal axes of object A with those of B. The rotation is about an axis through (-.44, .87, .16) and the origin (the center of gravity of A).

Once the rotation is complete, of the 180 processors holding information about A, 132 have matching data about B and 36 find matching data at a neighboring processor for a 93% (168/180) match. Similarly, there are 127 direct matches of B to A and 40 neighbor matches for a 92% match. For the pair shown in Figure 8, C is rotated -83.8398 degrees about an axis through (.002, .02, .99) and the origin (i.e., the rotation is in the picture plane). After rotation, out of a total of 476, C matches D directly at 412 processors as well as at 45 neighboring processors (96% match). D matches C directly at 387 processors and at 76 neighboring processors out of a total of 482 (96% match). By contrast, for the pair shown in Figure 9 there was only a 27% match after rotation. The left-hand object was rotated by -144.3 degrees about an axis through (.84, .50, .21) and the origin. The program reports that the objects are not the same shape.

Input of Figures

The objects are input as line drawings. That is, the picture coordinates of the endpoints of each line segment are given to the program. From the two-dimensional picture coordinates it computes the three-dimensional scene coordinates which are then used to postulate the hidden structure of the objects. This part of the program is included for completeness—it is not part of the computational model of the rotation process. It simply demonstrates the existence of a method for inputting the test pairs, it does not claim to model human processes.

Shepard and Metzler's test objects are of a severly limited class. They are all made from identical cubes joined together in different configurations, and thus every unobscured face in one of the figures represents a square in

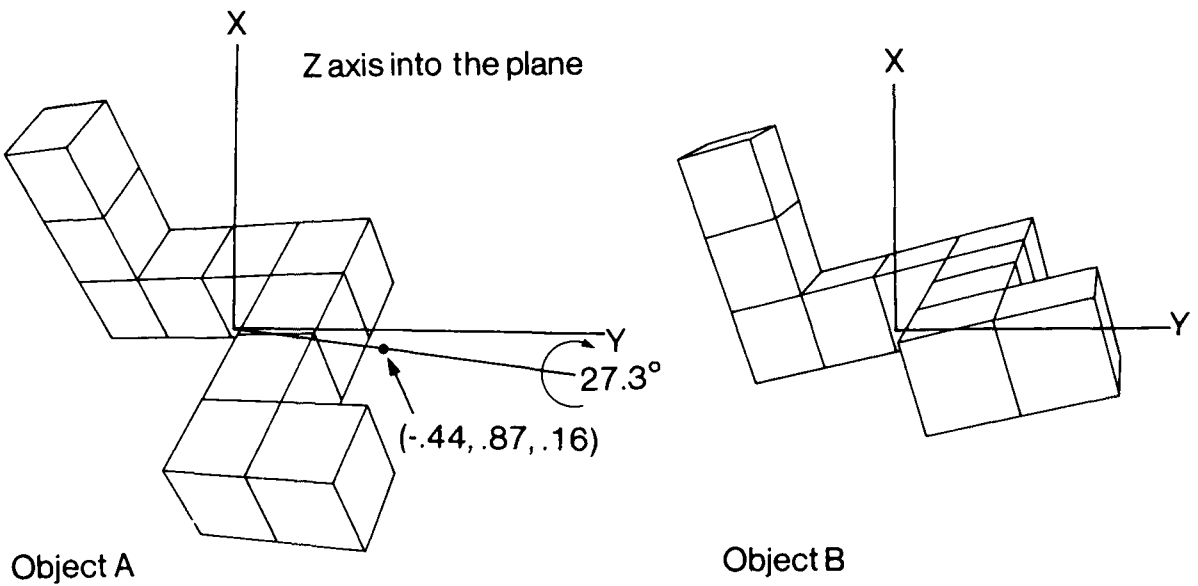

Figure 7. The objects from Figure 1 with the axis of rotation calculated by the program superimposed.

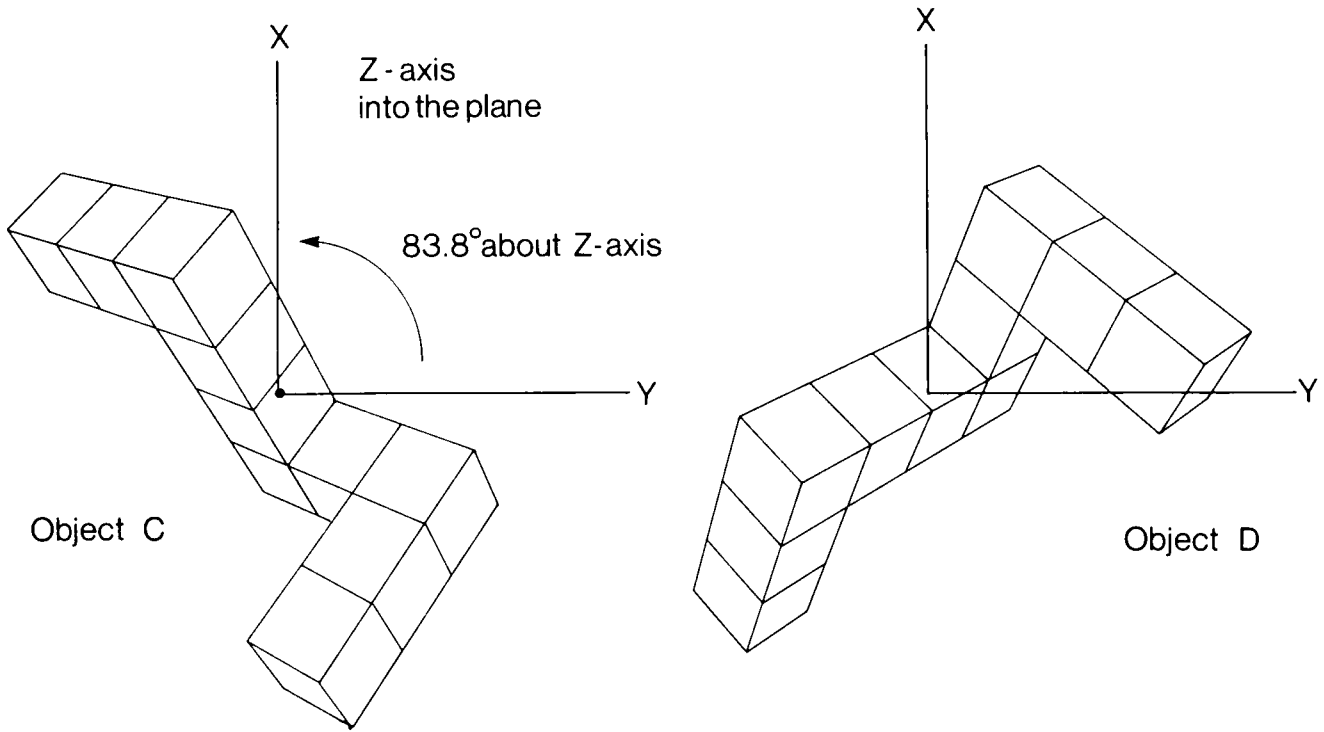

Figure 8. A rotation in the picture plane.

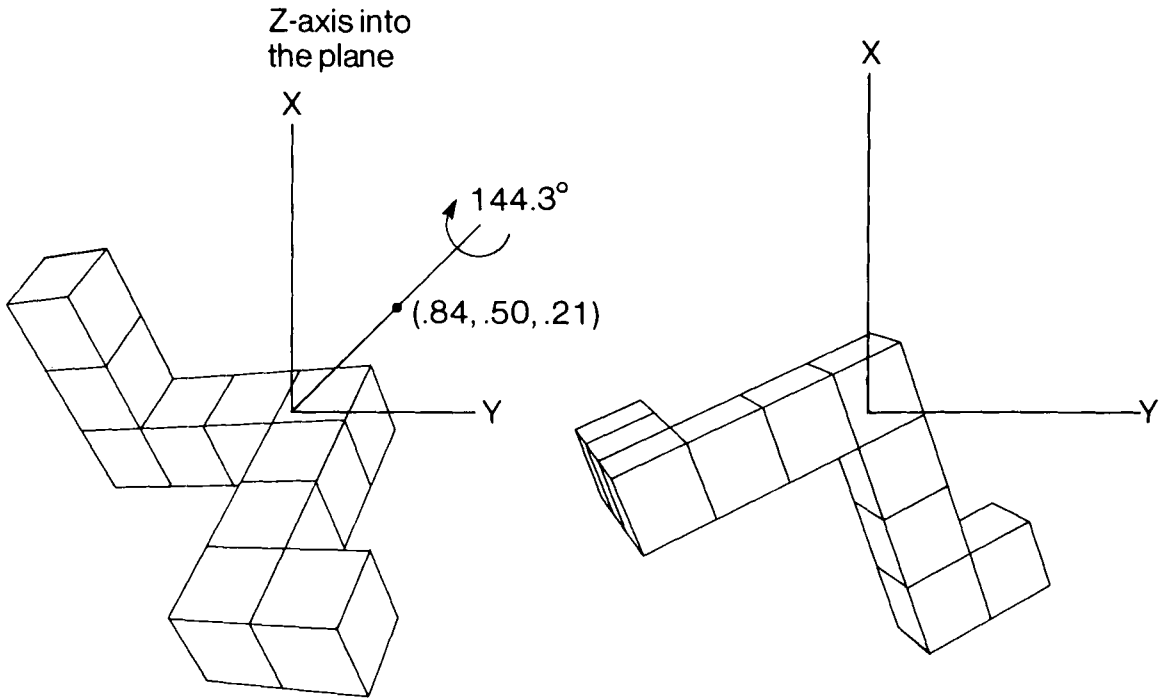

Figure 9. Objects which do not match. To discover that they do not match, the program rotates the left one as shown before comparing it to the right one.

three-space. Since all the cubes are the same size, so are all the squares. If the parameters of the perspective transformation (i.e., the camera focal length) are known, then these restrictions provide sufficient constraints to solve for the exact three-dimensional coordinates of all the visible faces in the input figure.

The hidden surfaces of the objects are subsequently derived from the positions established for the visible faces. A single square face defines two possible cubes. In this case the cube we want is the one which has its center further from the camera than the face generating it (otherwise the face would not be visible). For each face in the input figure, the faces of the cubes they define are added to a list of faces for the object. Duplicate faces and ones between touching cubes are eliminated.

The final stage in the input process is the distribution of the object description to the processors. First, the coordinates of the faces are adjusted to a new coordinate system which has its origin at the center of gravity of the object. Then the faces are broadcast one at a time to the complete set of processors. Each processor checks whether its cone intersects the face. If it does, it stores the distance of the intersection points from the origin as the appropriate radial value. Each processor carries this operation out simultaneously, so the time required to store the object is proportional to the number of faces. The same technique could be used for recalling objects from memory. Objects need not, in general, be restricted to ones which can be described by a set of planar faces. Any type of surface description can be used so long as it remains easy to compute the intersection of a line with it.

The Sphere of Processors

The simulation is based on a set of 482 processors. There are 50 radial divisions within each cone, so an object is represented by a sampling of 24,100 discrete locations. The resolution with which an object is represented could be increased by adding processors. Doing so would however increase the time taken for rotations although it would remain proportional to the angle of rotation. The more processors, the less the angle between them, and therefore, the greater the number of message relays required in rotating an object through a fixed angle. Rotation time will increase as the square root of the total number of processors.

Each processor is represented by a LISP atom. In LISP every atom has a property list associated with it and this is used as the processor's local memory. It holds the processor's coordinates, the radial values, and a list of neighboring processors. The pointers which make up the neighbors list model the direct communication links between processors. The processor coordinates and neighbor lists are computed by a program which is run just once to set up the data structure which simulates the sphere of processors.

Message passing is simulated by shifts of radial values stored on one processor's property list to that of another's. Parallelism is enforced by strict use of the LISP mapping functions. They apply the same function to every element of a list. Here the function is the algorithm that a single processor is to execute, and the list is a list of all processors.

LIMITATIONS AND RELATIONSHIP TO OTHER PSYCHOLOGICAL EXPERIMENTS

One of the most severe limitations of the model is that it provides no explanation or mechanism for mental translations. Pinker (1980) reports linear-time results for translations similar to those for rotations. The main problem appears not to be so much with translations alone, but rather with how to incorporate rotations and translations into a single model. While the current model is based on spherical coordinates, one based on Cartesian coordinates would handle translations but not rotations.

In contrast to the problem with translations, scalings are almost too easy. Bundesen and Larsen (1975) found that the reaction times of subjects matching two-dimensional shapes of different sizes increased linearly with the size ratio of the figures. Our parallel-processing model would predict instead that size differences would have little or no effect on the reaction time. An object can be scaled by a factor S simply by multiplying all radial values by S. This small amount of extra processing could be included in the object-matching process: the appropriate scale factor is easily determined as the ratio of the maximum radial value of A to the maximum radial value of B.

In a similar vein, Shwartz (1979) reports a dependency of rotation rate on size. In his experiment, subjects required longer to mentally rotate a large version of a two-dimensional polygonal figure through a fixed angle than a smaller version of the same figure. The parallel-processing model presented here would, on the other hand, predict no dependence of rotation rate on size. This prediction, it should be remembered, is for the case of three-dimensional objects, not the two-dimensional polygons of Shwartz's experiment. This difference could be quite significant, so some caution must be exercised in evaluating this discrepancy.

Intuition would say that it seems unlikely that we use moments of inertia to define unique axes for objects, but clearly we need some solution to the problem and whatever it is appears to be far removed from introspective insight. The important point here is that with the moments-of-inertia technique we have demonstrated the existence of at least one algorithmic method which can determine unique object axes and derive from them the correct axis of rotation without recourse to non-linear search. In some cases, particularly with symmetrical objects, the moments-of-inertia axes do in fact correspond to what might be called the object's natural axes.

Since the moments-of-inertia method produces an angle of rotation as well as an axis, there is no need to continuously perform object comparisons during the rotation process. The rotation is performed first and then the objects are compared. If some other method of axis determination were to be used which did not also generate an angle of rotation, it would not endanger the linear-reaction-time result. It would simply mean that object comparison would have to be performed after every rotation increment. Since object comparison is a parallel process, the extra comparisons would not unduly increase the total computation time.

Largely as a result of the model's dependence on parallel processing, the rate of rotation is more or less independent of the complexity of the rotating object. By varying the model's message-passing scheme slightly, we can make it either completely independent or very slightly dependent. From a series of experiments designed to test the effect of the complexity of the internal representation on the rate of mental rotation, Cooper and Podgorny (1976) concluded that the rate was not systematically related to the complexity. In the case of our model, object complexity could affect the number of radial values stored at a processor because the more convexities an object has, the more times a radial line is likely to intersect the surface, and handling these extra radial values might slightly decrease the rotation rate. In the case of the two-dimensional test shapes used by Cooper and Podgorny, increased complexity generally does result in more convexities, which means more radial values on average at each processor in our model. The more radial values a processor holds, the more values it must send to its neighbor at rotation time; and the more values, the longer the message, and therefore the longer the time to send it. The increased time to send longer messages may not be very significant, however, because much of the time required to send a message is associated with initializing the message rather than actually shipping the data.

Another factor which confounds the question of complexity dependence is that during many rotation increments a processor may not need to send any messages at all. Processors at the "poles" of the rotation never need to send messages, processors at the "equator" constantly send them, and those in between only intermittently. The effect of object complexity on rotation rate depends therefore on whether the complex part of the object (the part with the most convexities resulting in the most radial values) is nearer the poles or the equator of the rotation. We would expect, on this basis, that the rotation rate would show a slight complexity dependence but with quite a bit of random fluctuation, that is unless considerable care is taken to ensure that the complexity of the object is uniformly distributed over its surface.

The effect of complexity also depends on its definition. Cooper and Podgorny define it as the number of vertices in the shape. While the com-

plexity of the representation in our model is likely to increase with the number of vertices, it is not guaranteed to increase because it is the nesting of convexities which is important, and the nesting factor is only indirectly related to the total number of vertices. A cube (8 vertices) and an icosohedran (12 vertices), for example, both have the same complexity of internal representation in the sphere of processors. This is a further reason why any complexity effect measured as a function of the number of vertices is likely to be very small.

From the standpoint of the synchronization of the parallel processing, it would actually be simplest to allocate a fixed amount of time to message transmission. Since lengthy messages take only slightly longer than short ones, allocating sufficient time for the longest possible message would be reasonable. The rotation rate of a model based on this principle would exhibit no dependence on object complexity.

It is possible to generate a two-dimensional perspective image of an object from its representation in the sphere of processors. The image is represented by the same set of processors, but each processor is used to store a part of the image instead of a part of the object. The image is computed in parallel by each processor carrying out the appropriate perspective transformation on the information it has regarding the object. The results from one processor form a small segment of the image and usually need to be sent by a relay of message-passing to some other processor. The appropriate destination processor is the one which is storing the information about that part of the image where the image segment should appear. The details of this process are described in Strothotte and Funt (1981). They were originally worked out because of their application to high-speed computer graphics. In the present context, the process demonstrates that the underlying representation used in the parallel-processing model and the neighborhood-restricted communication structure are sufficient for the generation of two-dimensional perspective information which, as Pinker's (1980) results suggest, is a facility subjects have.

Shepard and Judd (1976) found that the rate of mental rotation for subjects perceiving an apparent rotation was markedly greater than that for subjects deciding the similarity of two objects. Our model would predict that the rate of rotation would be the same in these two cases because there is no way of speeding up the rotation. However, one possible explanation is that in the case of apparent motion the object is not actually being mentally rotated. The apparent rotations were periodic so during the first cycle the object representation could be spread over the processors and then it need only be compared during an external rotation, not actually rotated.

Just and Carpenter (1976) collected eye fixation data for subjects performing the Shepard-Metzler rotation and comparison task. They found what appeared to be three processing stages corresponding to (a) search for

two corresponding object parts; (b) transformation and comparison of the parts; and (c) confirmation that the same transformation which brought these parts into congruence will bring the rest of them into congruence. Our parallel-processes model is instead based on two stages: (a) object input consisting of the interpretation of the two-dimensional figure as a three-dimensional object; and (b) rotation and comparison of complete objects. The only prediction we can make about eye fixations in this case is that they all should be relevant to the interpretation of the figure as a three-dimensional object. Unfortunately, Just and Carpenter, in their experiment, only presented subjects with objects rotated in the picture plane. In this situation, subjects are not forced to interpret the figures as three-dimensional objects. The result is that the task may have degenerated into a comparison of simple two-dimensional figures in which case some of the difficult questions such as that of how the proper axis of rotation is found evaporate, thereby making their three-stage model feasible.

CONCLUSION

The linear-time behavior of the model presented here is not merely an accidental phenomenon, but rather a direct and necessary result of its dependence on parallel processing with neighborhood communication. The need for speed implied a need for parallelism; parallelism a need for communication between processors and a distributed representation of objects. A feasible implementation of communication links required limiting the number of inter-processor connections, while the combination of a distributed object representation and limited communication links resulted in a linearly increasing amount of time being required to shift an object's representation into position before it could be compared to another's.

The use of a distributed object description based on low-level primitives made the object-matching process particularly simple. This was partly due to the fact that it is a canonical representation in the sense that two identical or almost identical objects have identical or almost identical representations. Of course, their representations are orientation dependent, but there is a simple and well-defined rotation transformation which eliminates any difference in representation due to orientation. This is in contrast to a representation in terms of a high-level primitive such as generalized cones (Agin & Binford, 1973; Marr, 1978) where the same object can be described in two entirely different ways. This aggravates the matching process because identical objects may on occassion be described very differently, and there is no simple, well-defined way of transforming one description into the other. Generally the motivation for using primitives such as generalized cones in Artificial Intelligence systems is that they produce much more suc-

cinct descriptions than those based on low-level primitives. Succinctness is very important to those systems because they depend on sequential processing. It is more economical for them to handle a few complex primitives than it is to handle a large number of simpler primitives.

With the introduction of parallelism, the notion of efficiency and complexity changes. An efficient parallel algorithm is one which accomplishes the task in the minimum amount of elapsed time, not the minimum number of instructions. In a parallel processing system it may well be more efficient, in terms of minimizing elapsed time, to execute more instructions. Those extra instructions are being executed by processors which would otherwise be idle. Just this sort of trade-off is occurring in the choice of low-level shape primitives for the parallel model presented here. Many more instructions are executed in total than might be the case in a sequential processing system using generalized cones, but these instructions are handled by a vastly increased number of processors, so the total elapsed time for a solution is actually less.

Considered in terms of cost-effectiveness, it may now often be the case that to gain an order of magnitude in computation speed, it is cheaper to construct a system with an order of magnitude more processors than it is to develop a processor which runs an order of magnitude faster. Because of the problems of distributing the load among cooperating processors, it may, in fact, be necessary to provide somewhat more than an order of magnitude increase in the number of processors. Nonetheless, it may still be cheaper. Although it is a rather speculative point, it seems clear that the same factors might well have applied in the evolution of natural systems. Basically, it might be more likely to evolve a system with more of the same components, than to evolve one with faster components. Of course we must be careful—a similar argument would imply that to run faster we should have evolved more legs.

I believe that much of the controversy over analog representations might evaporate if they were viewed from the perspective of parallel processing. While not all problems are amenable to parallel processing (Minsky & Papert, 1969), many spatial problems are. Shepard had used his own and similar experiments to argue that the mental representation of an object bears the relation of a "second-order isomorphism" to the object in the real world. His is not a naive picture-in-the-head theory with all the well-known problems of the homunculus, but rather a characterization of some of the abstract qualities of mental representations. He argues that the rotation is carried out by an analog process. He states: "By an analogical or analog process I mean just this: a process in which the intermediate internal states have a natural one-to-one correspondence to appropriate intermediate states in the external world" (Shepard, 1978, p. 135). We can see that the parallel-processing model described in this paper has the characteristics of

an analog process, but there is no confusion as to the nature and status of the processes involved (see Pylyshyn, 1973, for a full discussion of these issues). There is no question here as to whether or not the process is continuous, whether or not there is some physical correlate to the external objects rotating within the parallel processor, no issue of pictures-in-the-head, and no searching for the homunculus. For other spatial-reasoning tasks it may also be possible to invent models based on parallel processing with neighborhood-restricted communication having analog characteristics without the usual, accompanying ambiguity.

Sequential processes manipulating structural descriptions of the type argued for by Hinton (1979) can interact with a parallel-process model of the sort described here in a well-defined way if need be. There is no reason to argue for a purely "analog" parallel-process solution to most problems, nor does it make sense to argue for a purely sequential one. We have seen that it is possible to found a computational model for a spatial-reasoning task on principles which provide a middle ground between the rather vague notion of an analog process and purely sequential processing of structural descriptions.

REFERENCES

Agin, G. J., & Binford, T. O. *Computer description of curved objects. Proceedings of the Third International Joint Conference on Artificial Intelligence.* Stanford, 1973, 629–640.

Baker, R. A spatially-oriented information processor which simulates the motions of rigid objects. *Artificial Intelligence,* 1973, *4,* 29–40.

Baumgart, B. G. *Geometric modeling for computer vision.* Unpublished doctoral dissertation, Stanford University, 1974.

Bundesen, C., & Larsen, A. Visual transformation of size. *Journal of Experimental Psychology: Human Perception and Performance,* 1975, *1,* 214–220.

Cooper, L. A., & Podgorny, P. Mental transformations and visual comparison processes: Effects of complexity and similarity. *Journal of Experimental Psychology: Human Perception and Performance,* 1976, *2,* 503–514.

Funt, B. V. Problem-solving with diagrammatic representations. *Artificial Intelligence,* 1980, *13,* 201–230.

Goldstein, H. *Classical mechanics.* Reading, MA: Addison-Wesley Publishing Company, 1950.

Hinton, G. Some demonstrations of the effects of structural descriptions in mental imagery. *Cognitive Science,* 1979, *3,* 231–250.

Hu, M. K. Visual pattern recognition by moment invariants. *Computer methods in image analysis.* In J. K. Aggarwal, R. O. Duda, & A. Rosenfeld, (Eds.), New York, NY: John Wiley & Sons, 1977.

Hubel, D. H., & Wiesel, T. N. Brain mechanisms of vision. *Scientific American,* 1979, *241,* 150–162.

Just, M. A., & Carpenter, P. A. Eye fixations and cognitive processes. *Cognitive Psychology,* 1976, *8,* 441–480.

Kenner, H. *Geodesic math and how to use it.* Berkeley, CA: University of California Press, 1976.

Kosslyn, S., & Shwartz, S. Simulating visual imagery. *Cognitive Science,* 1977, *1,* 265-295.

Larsen, A., & Bundesen, C. Size scaling in visual pattern recognition. *Journal of Experimental Psychology: Human Perception and Performance,* 1978, *4,* 1-20.

Marr, D. Representing visual information—A computational approach. In A. R. Hanson & E. M. Riseman, (Eds.), *Computer vision systems.* New York: Academic Press, 1978.

Minsky, M., & Papert, S. *Perceptions.* Cambridge, MA: MIT Press, 1969.

Pinker, S. Mental imagery and the third dimension. *Journal of Experimental Psychology: General,* 1980, *109,* 354-371.

Pylyshyn, Z. What the mind's eye tells the mind's brain: A critique of mental imagery. *Psychological Bulletin,* 1973, *80,* 1-24.

Shepard, R. N. The mental image. *American Psychologist,* 1978, *33,* 125-137.

Shepard, R. N., & Judd, S. A. Perceptual illusion of rotation of three-dimensional objects. *Science,* 1976, *191,* 952-954.

Shepard, R. N., & Metzler, J. Mental rotation of three-dimensional objects. *Science,* 1971, *171,* 701-703.

Shwartz, S. P. Studies of mental image rotation: Implications for a computer simulation of visual imagery. Unpublished doctoral dissertation, The Johns Hopkins University, 1979.

Smith, D. *Using enhanced spherical images for object representation.* (MIT AI Memo No. 530). Cambridge, MA: MIT, 1979.

Strothotte, T., & Funt, B. V. Raster graphics using parallel processing. *Proceedings of Computer Graphics 81,* London, England, October 1981.

Trehub, A. Neuronal models for cognitive processes: Networks for learning, perception and imagination. *Journal of Theoretical Biology,* 1977, *65,* 141-169.

Wells, D. A. *Schaum's outline of theory and problems of lagrangian dynamics.* New York: McGraw-Hill, 1967.