DETERMINING THE THREE-DIMENSIONAL MOTION AND MODEL OF OBJECTS FROM A SEQUENCE OF IMAGES[1,2]

J.W. Roach[3] and J.K. Aggarwal[4]

TR-80-2 October 1980

[1]This research was supported in part by the Air Force Office of Scientific Research under grant AFOSR 77-3190.

[2]This report is reprinted from the PhD. dissertation of J.W. Roach, completed in May 1980 at the University of Texas at Austin.

[3]Currently with the Department of Computer Science
Virginia Polytechnic Institute
Blacksburg, Virginia 24061

[4]Depatrments of Electrical Engineering and Computer Sciences
University of Texas at Austin
Austin, Texas 78712

ABSTRACT

The goal of this dissertation is to determine precisely how an object is moving in three-dimensional space and to determine the three-dimensional relationship of points on the surface of the object. The only information available is a sequence of photographic images taken as the object moves across the field of view.

The problem can be broken down into two sub-problems: the problem of determining the correspondence of feature points in one image with feature points in the next image; and once the correspondence is established, the mathematical analysis required to determine the model and the movement. The correspondence problem, i.e., matching, is investigated using images of moving blocks. The corners of the blocks constitute the feature points to be put in correspondence between images. Several matching methods are combined into a hierarchy so that if one method fails another method can take over to help complete the matching process. The top level of the hierarchy matches by searching for feature points in the image of expected positions as computed from the expected movement of the object. The next hierarchy level matches an object by its position relative to other objects in the image, a property that is assumed to change only gradually. The next hierarchy level matches a block's faces by relative position. Once faces have been matched,

feature points bordering the faces not already matched by expected position can be put in correspondence.

The mathematical analysis of the problem shows that there are an infinite number of geometrically similar solutions, each solution differing from the others by a scaling factor. A specific solution can be found by setting the scaling factor to an arbitrary number. Two views of six feature points or three views of four feature points are required to find the model and movement. For good accuracy, however, considerably more points, two views of twelve or fifteen points, for example, are needed.

CHAPTER 1

INTRODUCTION

Computer scene analysis beginning with Roberts [23] has concentrated on segmentation, object recognition and the mathematical analysis required to determine an object's three-dimensional position. The analysis of image sequences of moving objects has received some attention, almost entirely directed to the analysis of the two-dimensional movement of objects. The original motivation for studying two-dimensional motion came from a desire to analyze with a computer the vast quantity of satellite images of clouds (Endlich, et al. [8], Leese, et al. [13]). An abstract model of cloud movement was examined in Aggarwal and Duda [1]; this work was extended to the analysis of the two-dimensional movement of curvilinear shapes in Chow and Aggarwal [5] and Martin and Aggarwal [18]. Jain and Nagel [12] broke from cloud movement data and used difference pictures to analyze street scene images. Although the images show pedestrians and cars moving in three dimensions, there is no attempt to recover three-dimensional information from the images.

It is clear that past research work has mainly been concerned with two-dimensional motion. In part, this is because the interpretation of images of objects moving in three-dimensions is much more complicated than two-dimensional motion since rotation and movement in depth are difficult to analyze. For example, rotation in space is defined to be about a general three-dimensional line whereas rotation in a plane is defined to be about a single point in the plane. In addition, parts of an object can disappear from view as a result of rotation in space; rotation in a plane does not by itself cause an object to occlude itself. In this dissertation we shall examine and solve many of the problems involved in determining the three-dimensional motion of objects from a sequence of two-dimensional images.

Analyzing the three-dimensional motion of an object from two-dimensional images requires a means by which to grasp the problem mathematically, i. e., a mathematical formalization. Psychologists have classically studied movement in terms of texture gradients that aid human depth perception (see Gibson [10] and Braunstein [3]). These psychologists use images of points on the surfaces of objects to study the movement in depth of the objects; Roberts [23] also uses surface points to help determine object depth. By definition, the

three-dimensional relationship of points on a rigid object does not change over time. Consequently, changes in the two-dimensional spatial relationship of points between images must be caused by a relative movement between the camera and the object being imaged. In studies involving binocular vision (two cameras or eyes spaced a known distance apart), this change of position of a point between the two images is known as the "disparity" in the images of the point. A simple triangulation argument gives the depth of the point in this case. Thus, the change of position between images of points on an object's surface can be used to formalize the problem of determining the three-dimensional movement of objects in space.

Consider the sequence of images of a moving object given in figure 1. This sequence shows a truncated wedge rotating and translating. These simplified line drawings gloss over the difficult low-level processing problems, such as separating out the various objects in an image (segmentation) and identifying the same feature points (or tokens) on an object in each image despite possibly changing illumination conditions. This dissertation does not address these low-level processing problems; instead, images from the widely studied blocks world (Winston [29]) are used since edges and vertices (feature points) are

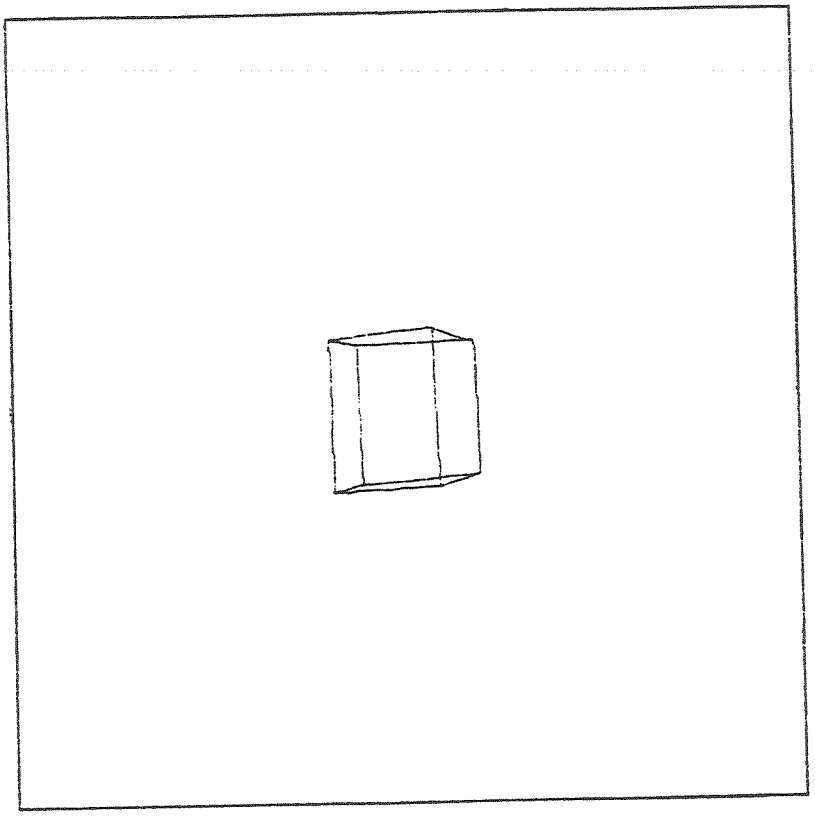


Figure 1

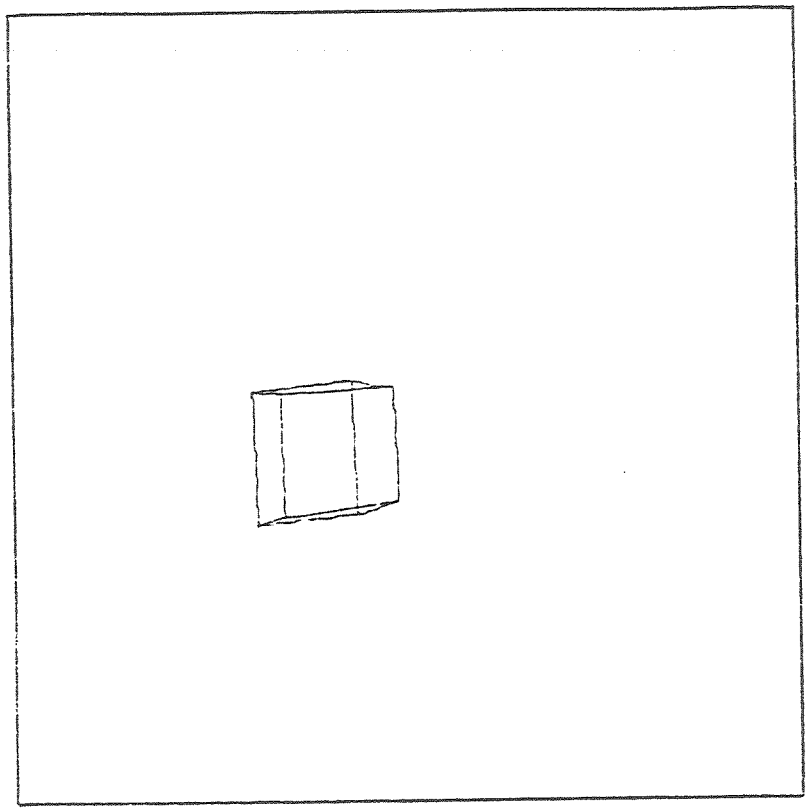


Figure 1 continued

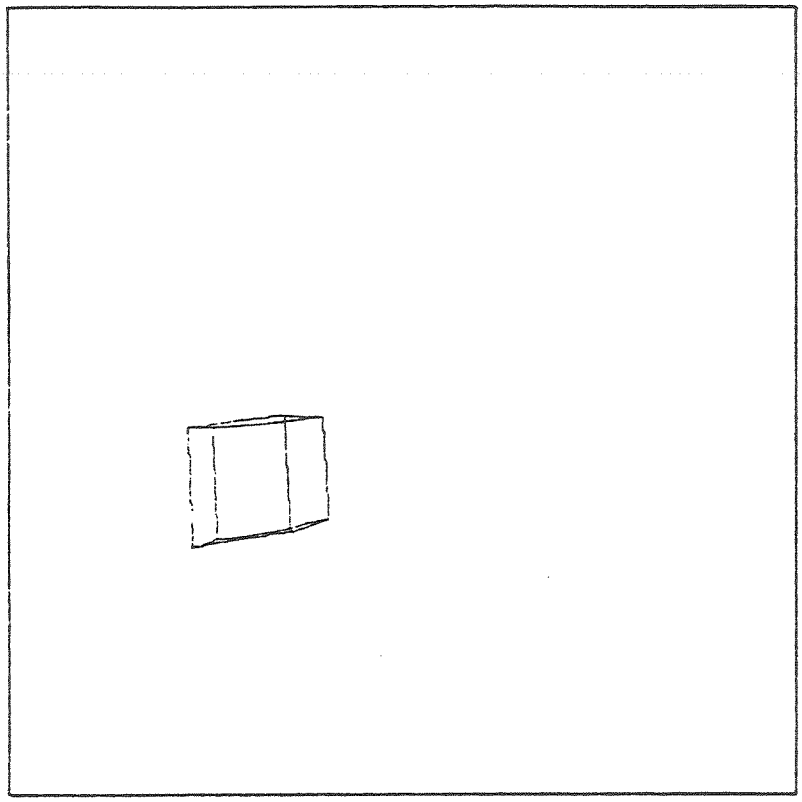


Figure 1 continued

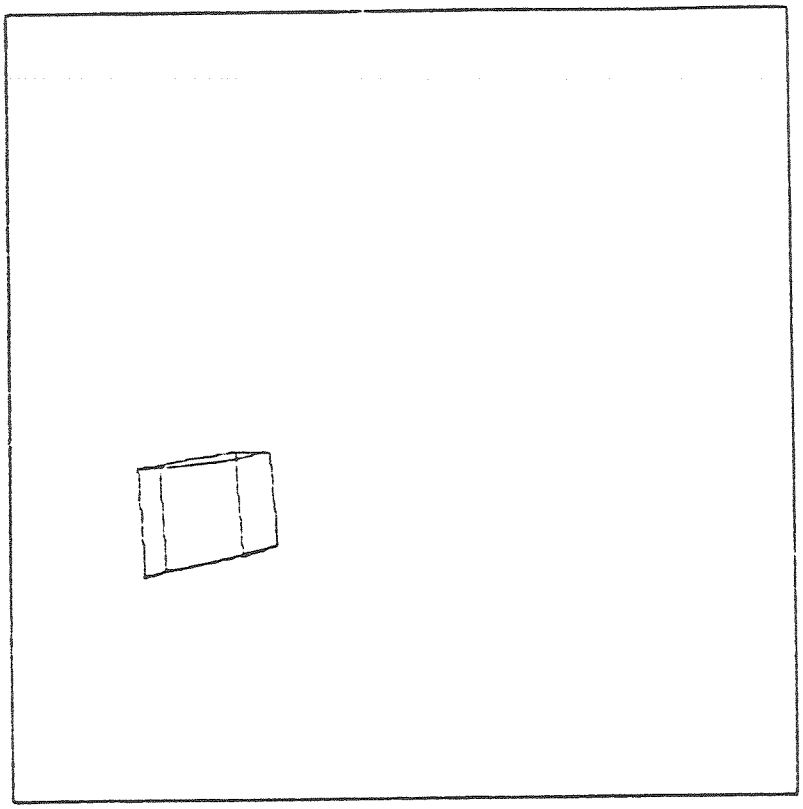


Figure 1 continued

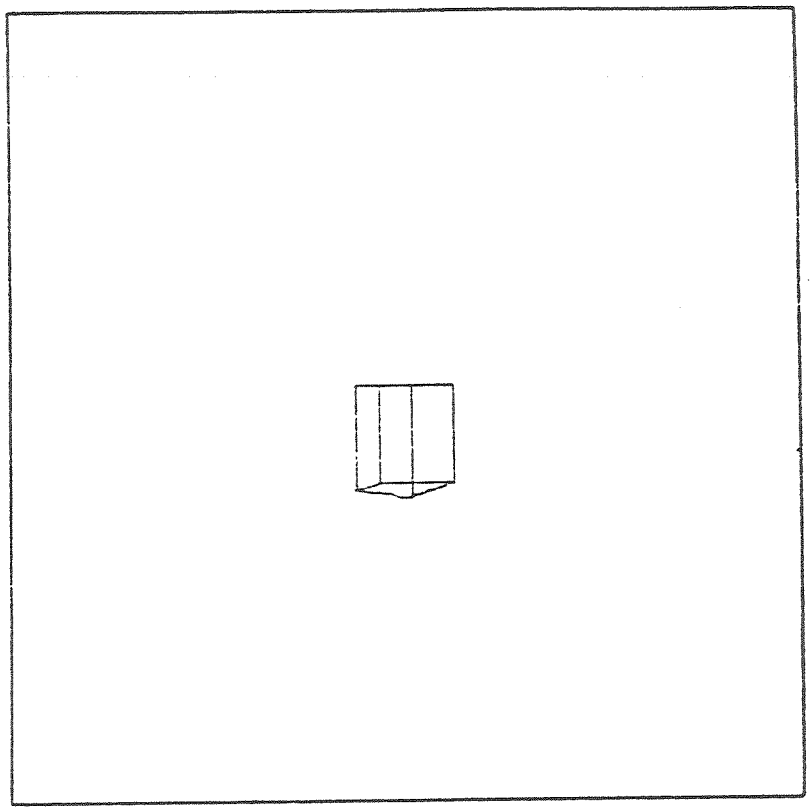


Figure 1 continued

easily determined. In images of the real world, however, we know of no general solution to finding the same points on an object's surface in each image.

Once points on an object's surface have been extracted in each image, we must determine the correspondence of points between consecutive images. By "correspondence" here we mean the mapping that takes an image of an object point to the image of the same object point in the next image of the object. This is a difficult problem since an image may have more than one moving object and thus many points to choose from. The correspondence problem is further complicated by the disappearance of points on an object due to occlusion from other objects, self occlusion as points rotate out of view, shadows, etc.

Once the correspondence of points has been established, we can attempt to analyze the motion. Here we are confronted with a basic question: how is the motion to be represented? One method might involve qualitative descriptions such as "moving left and away, rotating to the right," etc. as in Badler [2]. There still remains the question, however, of how the qualitative description is to be calculated. If a more exact mathematical calculation were used to derive the qualitative description, then the more precise

mathematical result would also be a valuable characterization. In this dissertation, we shall take the more quantitative approach and use matrices with "homogeneous coordinates," as Roberts [23] did, to describe the movements of objects. Homogeneous coordinates are an elegant means of representing movement since a four by four matrix can represent any rotation or translation. We have to be sure, however, that given these two elementary motion matrices, it is possible to represent any motion an object can have in space. For example, consider a planet travelling about the sun. It is rotating about an axis passing through the sun; at the same time it is rotating about its own polar axis. How is this two-axis rotating movement to be analysed? What if there are n axes of rotation? The translation and rotation four by four matrices are an adequate representation since a theorem from classical mechanics (see Coffin [6]) establishes that any motion, including the rotation within rotation problem just mentioned, can be decomposed into one rotation and one translation. The rotation and translation matrices can be multiplied together to form one matrix useful for predicting the next three-dimensional position of the object. In addition to representing motion, a four by four matrix in homogeneous coordinates can be used to model the projection of object

points onto the focal plane of the camera.

There are some fundamental problems involved in analyzing sequences of images of moving objects that we want to understand:

1. whether the extra images, and thus object motion, help the three-dimensional analysis, and
2. exactly how much of the original three-dimensional information can be recovered from a sequence of two-dimensional images.

The first problem is essentially concerned with determining what the value of motion is in analyzing scenes. The relationship of points on the surface of an object and the entries in a four by four matrix that is a calculation of the movement of those points.

Our analysis of three-dimensional motion will depend on certain key assumptions. For example, we assume throughout that all images are from one camera. We assume that there is no a priori knowledge of specific objects or their specific motions, that objects in general are rigid, that motion is smooth and continuous, and that central projection is the best geometrical description of the image formation process. By changing this last assumption to parallel projection, an exact model (to within a

reflection) for any four non-coplanar points can be derived given three different views of them, as Ullman [25] has shown. Badler [2] used a spherical projection model and was able to predict the point positions in succeeding images of translating objects. We shall see that the answers to our fundamental questions depend on the assumptions about image formation, the rigidity of objects, and the lack of specific prior knowledge of models or movement of objects.

The problems in determining the movement of an object from its images are identical in many ways to the problems encountered in optic flow analysis and stereopsis. Stereopsis, or binocular vision, is the problem of determining the depth of objects from two different images. The distance between the two imaging devices is assumed to be known. Optic flow analysis, originated by Gibson [10], depends on a vector field formed by points on object surfaces as a camera or eye moves through its environment. Some recent work by Williams [28] concentrates on deriving the "focus of expansion" which is the fixed point of the vector field in a sequence of images taken by a translating camera. The fixed point lies along the line of translation. In all of these problems separate views of an object are given in which the correspondence of points on the surfaces of

objects must be determined and the displacement of these points used to determine distance from the camera to the object. There are some differences; for example, binocular studies normally assume that the distance between cameras is known, and optic flow studies assume a moving camera. The problems encountered, nonetheless, are very similar, and the results of this study should be applicable to these other areas of research.

This dissertation is divided into two parts. The first part discusses the problem of putting the images of points on an object into correspondence for pairs of images. The second part discusses the mathematical problems in deriving the model and movement of an object given a sequence of images. Motion representation and a model of the image formation process are prerequisite knowledge that is discussed to aid in understanding the derivation of an object's model and movement.

CHAPTER 2

CORRESPONDENCE

2.1 The Correspondence Problem

In this chapter, we shall examine the correspondence problem: given the same identifiable tokens (or points) on the surface of an object in two different images of the object, find a one-to-one onto mapping from the first set of images of tokens to the second set of images of tokens on the object. This simple statement of the problem ignores difficulties caused by the disappearance of tokens due to occlusion or movement beyond the view of the camera. These difficulties will also have to be discussed.

Correspondence and motion determination are intimately related problems since if either the correspondence or the motion is known the other can be determined. Neither is known originally. Correspondence, or matching, of tokens must be determined first because motion determination depends on knowing the

correspondence.

One obvious strategy to match tokens between images would be to choose the "nearest neighbor" of a point in the next image. This strategy has not been chosen since it imposes assumptions that are untenable for the class of images available: the timing between images must be chosen in such a way that the closest neighbor to a point in the next image is itself; there is also an implicit assumption that camera registration is good between images. There are additional problems with breaking ties and mapping conflicts (a point in the next image being the nearest neighbor to two different points in the current image). Also, for rotating objects, in some cases points on the boundary of the object appear to exchange positions suddenly with interior points that are approaching the boundary of the object due to rotation. This case would possibly confuse a nearest neighbor algorithm. For these reasons, the nearest neighbor strategy is not adopted here.

The correspondence method described here follows the one developed in Roach and Aggarwal [22]. In that paper, the objects being tracked were blocks, and the main goal was to put blocks and their faces in correspondence. The images used for this purpose are given in appendix five. The method described here suggests an extension to

these ideas to put images of points into correspondence also.

2.2 Motion in the Blocks World

There is a trend in research on images of moving objects toward analyzing motion in everyday scenes. Real world scenes can be extremely complex; so to develop a fundamental understanding of motion it was decided to examine scenes containing a restricted class of objects.

A system is described here to find and track points on moving blocks through a series of two-dimensional images of scenes. The blocks world was chosen because it is a reasonably well understood domain, because it is complex enough to create interesting problems, and because the same identifiable tokens (the corners) can be found in each image. Blocks are assumed to be rigid, convex polyhedra. Scenes normally have several objects some or all of which may be moving. The blocks move independently of each other, although coincidental joint motion is possible. For example, a block might be sitting on a moving block and hence move with it. No attempt, however, is made to conclude that

this movement is caused by the support relationship. The blocks may move in different planes although there are practical support difficulties with taking sequential static pictures of blocks moving vertically. Blocks may enter and leave the scenes and may occlude one another. Where one object occludes another, contrast must be sufficient to reveal the occluding edge. Blocks may move toward or away from the camera and may change velocity. If objects accelerate more frequently or at a constant rate, the matching program might not be able to track the object. This may cause matching problems in case the position of the object relative to other objects is also changing very rapidly. It is assumed that the time interval between scenes is short enough that only small changes occur: objects do not magically appear or disappear, for example, nor do they move more than one-third of the way across the image between any two scenes.

The goals of the system are to find objects in each scene correctly, to disambiguate uncertainties, and to derive motion matrices for each object. The line labelling method proposed by Waltz [27] cannot be used to find objects in images because the noise and number of missing lines would be overwhelming. Since there is no prior knowledge of particular objects or their motions,

one of the goals of the system is to build models of the objects as they appear and make a record of their movements.

Solutions to fundamental correspondence problems in the blocks world may be extended to real scenes since many objects (cars, trains, airplanes, buildings, objects on conveyor belts, etc.) can be modelled by blocks. Badler's model [2] and most graphics programs use this idea.

2.3 Experimental Setup

An image dissector camera is used to take pictures of blocks. Edges are found using the gradient operator program described in McKee and Aggarwal [16]. The output of this program is a 112x112 binary array with ones representing points along the edges.

The camera is in the position of an observer standing several feet from the scene; its relation to any plane in which a block may be moving is unknown. Exact distances from blocks, tilt, pan, roll, and height of the camera are unknown.

When one object occludes another, occluding edges are not lost. Shadow lines do not appear against the flat black background but do appear when cast on blocks. The boundaries of the tabletop are invisible to the camera. Noise is introduced by digitization, edge finding, and by the edge-following-end-point-finding program. Since T-nodes are the main clue to occlusion, it is particularly difficult to handle the introduction of false T-nodes and the destruction of legitimate T-nodes by noise. Objects are moved by hand between scenes so variation in the velocity of constantly moving objects can be considerable.

2.4 The Analysis Problem

Originally, in Roach and Aggarwal [22], motion was represented using a comparison of image sizes of an object. A bigger image meant an object was moving closer, for example. This scheme was qualitative and imprecise. In addition, it was incorrect for rotating oblong objects moving in depth. The current movement computation is much more precise but has the drawback of needing many points to complete the analysis. In addition, more work is required to put points in correspondence than to put

objects in correspondence with greater possibility of error. In what follows, a description is given of the original object matching program that was implemented augmented by proposed changes needed to match points on objects.

Finding objects in an image and finding which points on objects in the previous image correspond to points in the current image are very hard problems. Two basic principles have guided the development of the object finding and points matching programs: multiple interpretations are allowed, and matching is a multi-level process. The basic philosophy of the system is that if a decision cannot be made with the current image, the uncertainty is carried forward in the hope that it can be resolved in a later image. The matching process has been designed to achieve additional flexibility by incorporating a hierarchy of methods. The highest level of matching depends on the velocity of the points on an object, a global property; the lowest level depends on local features, the faces adjoining to a vertex. Thus, matching is arranged in levels starting with global and going down to local features to achieve a match. Even with this degree of flexibility matching can fail by not finding objects or by consistently finding multiple matches; ambiguous matches carried from image to image

constitute matching failure. The remainder of this section discusses these ideas in greater detail.

Once the end points of lines in the image have been found, the next step, finding the faces, is not hard. It is the grouping of faces into objects that causes difficulties. Waltz's line labelling method [27] provides an elegant solution to the face grouping problem, but the images in this project have too many missing lines and too much noise for Waltz's method to be effective. So the method used here is heuristic, based on occlusion clues from T-nodes. Since there are multiple scenes to record movement, it is unnecessary to make a hard and fast decision of which faces belong together. It is possible instead to allow some connections to be uncertain and then use information from prior images to help resolve ambiguities. Prior images may also be ambiguous in which case no decisions need be made until later images are analyzed. The only requirement is that there be at least one match. This match must be a one-one onto mapping from points on a possible object in the current image to one of the possible interpretations of the prior image. The mapping must be one-one onto in the sense that every object in the current image must map to an object in the interpretation of the prior image or be marked as new, and every object in the interpretation of the prior image must

map to an object in the current image or be marked as gone. New objects appear on an image boundary, and objects that are gone have disappeared either off the boundary or behind another block. For each interpretation of the previous image, the matching routine looks for a corresponding interpretation in the new image. Thus multiple interpretations can be carried forward.

It is the goal of this chapter to develop a method for matching points between images. Methods used by researchers in the past will now be reviewed in an attempt to find methods that work for the problems of three-dimensional motion. Roberts [23] tried all combinations for making the correspondence between model points and points in an image. He selected the best fit (least squared error) from among all possibilities. This is not a reasonable method in theory or practice due to exponential blowup in computation time. Ullman [25] describes a method that matches an image point's nearest neighbor in the next image. This method requires that the time between images be very short. The nearest neighbor strategy was not pursued in this work partly because the kind of data needed to support it was not available. The possibility of incorrect matchings seems considerable (especially for rotating objects) but possibly not insuperable. Aggarwal and Duda [1] and Martin and

Aggarwal [18] both relied on local features (angular measure and curvilinear shape outline, respectively) to match pieces of objects between images. These methods are suitable for two-dimensional motion since they were assuming a parallel projection model in which the change of angle between camera and object has no effect on the object's image. In three-dimensional movement, however, angles and outlines can change drastically between images rendering this approach unsuitable. Chow and Aggarwal [5] use more global properties to put objects in correspondence. They use the predicted position of an object's centroid to locate the object in the next image. If an object changes velocity, however, another method must be used to find the object. The methods discussed so far are heuristic in nature. Attempts have been made, however, to introduce a mathematical approach using ideas from projective geometry. Underwood and Coates [26], for example, worked with faces of blocks and used shape numbers that do not change as the angle of view changes. This method depends on a clear view of each face and on a very noise-free image of the face. When faces are partly occluded or lines in an image are noisy, spurious shape numbers would be generated causing the matching process to fail. Another problem with this method is that regular faces would be hard to identify since their shape numbers

are not unique. Underwood and Coates assumed virtually noise free images of single convex objects, but for the images to be analysed in this project these assumptions are untenable. Duda and Hart [7] give an excellent treatment to the problem of projective invariants using feature points on the surfaces of objects. Their result, however, is not useful since it involves a rather large search. There are thus many different techniques available, but none of them is entirely satisfactory. Since any single method could fail it was decided to use multiple methods arranged in a hierarchy by importance.

The first matching level would use object points' expected position to search for the object in the scene. A matching point in the next image is sought within a small radius about the expected position. Failure to find object points at their expected positions means that the object's direction or speed has changed and another matching method is required: relative position matching. It is assumed that the time intervals between scenes are short enough that many "significant" changes do not occur at once. In particular this means that the position of an object relative to other objects in the scene changes only gradually. A large number of changes between scenes can defeat object position matching. If an object cannot be identified by its position relative to other objects,

matching proceeds at a face level by pairing faces that have similar positions relative to their neighboring faces. Once there is a consistent face mapping, the faces can be reassembled into objects. Face matching is especially needed to defeat errors in initial object segmentation. When a bad segmentation suggests too many objects or too few, the objects can be broken down into faces, matched, and then reassembled. If the top level, velocity matching, has failed, then the lower levels of matching must be followed by another matching routine that puts points into correspondence. This additional level of matching requires that the faces have been matched between images.

Once the faces of an object have been put in correspondence between images, it is relatively easy to put points bordering the faces into correspondence. A point that is common to faces A, B, C in one image will be common to X, Y, Z in the next image so long as A corresponds to X, B to Y, and C to Z. Of course, more than one point may adjoin the same corresponding faces in two images. In that case, the relative position of the points in the first image must be used to find the correspondence between points in the next image.

The discussion so far has ignored problems caused by occlusion. Points in one image may disappear in the

next image, and new points may appear. It is necessary, therefore, to determine when points have disappeared and to use new points in movement calculations as they appear.

2.5 The Analysis Program

This section describes the programs proposed to process and analyze the data starting from a sequence of images S_1, S_2,..., S_n and ending in sets of points put in correspondence. The end goal of this system is to create object models that tell the three-dimensional positions of the points on the surface of the object to within a scaling factor; the system also determines the motion for each object to within a scaling factor. Enough points must be available, however, to make the analysis accurate. Models and motion histories are the output of the system. During analysis, the current image, the immediately preceeding image, the models of objects determined from their images, and the expected movement of objects are the only information sources.

The initial stages of the program collect scenes S_1, S_2,..., S_n, find lines and their end points, and then group the lines into faces. These processes are well

understood and not of sufficient theoretical interest to be discussed in detail. Instead, the following sections discuss how objects are found and points matched.

Before entering into a detailed account of point matching in section B below, we shall give an overview of the matching process. We assume that faces have already been found and grouped into objects. The different layers of the hierarchy are labelled by numbers.

1. (motion matching)
   for each expectation for each object do
     if all points are close to their expected position
        then succeed else
     if some of the points are close to their expected
        position then
        if another object is occluding the object being
           searched for then succeed else
        if points have disappeared due to self-occlusion
           (rotated out of view) then succeed else
        if points can be matched by following edges from
           previously matched points then succeed else
2. (relative position matching)
   determine the relative positions of objects in the image
   if any object's position description matches the position
      description for the sought object then succeed, match
      faces (see 3 below), and then match points (see 4 below)

else

3. (relative face position matching)
   determine the relative positions of faces in the image
   for each face of the unmatched object
   if a face in the image has the same relative position description and the neighbors of that face also match the neighbors of the face from the unmatched object then succeed and match points (see 4 below)
   else fail.
4. (point matching)
   [note: this level is called from levels 2 or 3 after having matched faces]
   associate with each point the list of adjacent faces in the image. From the face matching map, match points that have equivalent lists of adjacent faces.

A. Object Finding

The program that groups faces into objects is based on the heuristic that a T-node indicates a point where one object occludes another. This heuristic is not entirely valid since noise and accidental alignments can cause spurious T-nodes. The strategy used to handle

spurious T-nodes is to allow ambiguous segmentation. Faces are related to adjoining faces in one of three ways: the faces that belong to the same object and are connected by the interfacing line, the faces that may possibly belong to the same object, and the faces that definitely do not belong to the same object. The line separating faces that may belong to the same object is marked as an uncertain connecting line. Two interpretations of the objects in the scene are produced, one in which all uncertain connections are allowed (giving a minimal number of objects) and one in which uncertain connections are not allowed (giving an interpretation with a maximal number of objects).

B. Matching Object Points Between Images

The system described up to this point has been bottom up or data directed. When matching points on objects between images, however, there are definite positional expectations based on the motion of objects. These expectations amount to goal direction, thus the entire system is a combination of data and goal directed processes.

The matching program works at four levels: motion matching, relative object position matching, face matching, and point matching. Ambiguous matches are allowed so there may be more than one possible interpretation of the motions of objects. Each interpretation has a list of expectations for the movement of objects. In the following discussion, it is assumed that the matching process is dealing with one interpretation at a time.

Motion matching normally proceeds by taking each expectation in turn and searching through the list of end points on objects in the image for all points within a limited radius of their expected positions in the image. If no points match, then before the expectation is discarded entirely, the program tries to determine whether the object has disappeared behind another block. This is achieved by determining whether a block is occupying the same part of the image where the points were expected. A block occupies the same area of the image if its centroid is close to the centroid of the expected position of the points that were not found. If an object is thus found in the image it could either be the image of the block being sought or the image of a different, occluding block. If there is another expectation that places a block in the correct position of the block that was found, or if no

face match can be found between the block being sought and the image of the block in the correct position, it is assumed that the image of a different block has been found. If a face match can be found and no other object is expected in that position, it is assumed that the object is found but its points are not in their correct positions. From the face match a points match is performed and a new movement calculated.

If some of the points are found in their expected position but others not, then several possibilities must be explored. One possibility is that the points not accounted for are simply occluded by another object. This is tested by checking whether the object to which the points already identified belong is occluded by another object in the image. This determination depends on a good grouping of faces in the image into objects. A second possibility is that points have disappeared and new points appeared due to object rotation. A simple means of deciding whether this is the case is to note that points that once were on the interior of the object in the preceding image now lie on the exterior boundary; new points may also have appeared. New points can be identified since they connect along new lines to previously known points that now have more connections than before. To assure as much accuracy as possible in

the movement and model calculations, as many points as possible must be used. Thus the identification of new points is important for they must be used in matching and motion analysis with the next frame. A third possibility is that due to noise the images of some points are just outside of the circle of expected position. Candidates for matching these points can be found by following connecting lines from identified points in the image to unidentified points. The pattern of connecting lines between all visible points is known from the previous image. This pattern can be used to guide a search for the points not yet matched by following connections from points that have already been matched. Thus, it is frequently possible to match points using topological (connection) information when motion matching partially fails.

For various reasons, motion matching alone is not sufficient. Velocity estimates may not be available, for example, or objects may change velocity. Further levels of matching are needed to prevent failure. For each object in an image, the position of the object's centroid is compared with the centroid of each adjacent object and a relative position description (above, left, right, below, above right, etc.) formed. For each object from a previous image attempting a match with an object in the

current image, the relative position descriptions are compared. If the two descriptions are very similar (one mismatch of the kind left versus left and below is allowed) then the object positions match. If, in addition, the adjacent objects also position match and the status (occluded, shadowed, or on the boundary) of the two objects matches, then the objects have been position matched.

In the face matching routine, the program attempts to put the faces into correspondence using the relative position method described above with faces instead of objects. After faces are successfully matched, they are reassembled into objects. Reassembly requires deciding which faces belong together. The objects of the previous scene are used as a guide for regrouping the faces of the current scene. Faces in the current scene can, however, only be grouped together provided there are no T-node clues that prohibit their being joined. One important reason to have this level of matching is to defeat errors in the initial segmentation of the current scene; another important reason is that face matching must precede point matching.

Point matching works by listing with each point the faces next to it. When the faces of an object in two different images are put in correspondence, points can

normally be placed in correspondence by simply matching points with equivalent associated lists of faces. The list of associated faces of a point, however, is not always a unique description; several points may have the same list of associated faces. In this case, the relative position of the points in question can be used to disambiguate the matching process for those points. If a face has disappeared from view (and that is noted in the face match list) then special care must be taken when assigning the lists of faces associated with points for points that are on the boundary of the object and that are on the face next to the face that has disappeared. Similarly, care must be taken when building the face association list for points that border a face that has just come into view.

Conclusions

Experience with this hierarchical method of matching, Roach and Aggarwal [22], shows that it is reasonably robust. If the time interval between images is not precisely equal, or if the objects are not moving with uniform velocity, then the predictive level of matching is

of little value. The size of the time interval between images is also reasonably important since the relative position of objects must not change too quickly. The matching scheme presented here, however, is less sensitive to the time interval than a nearest neighbor matching strategy which requires a very high sampling rate. A disadvantage of using a predictive matching method is that the amount of time required to compute the prediction may be large. The hierarchy of matching techniques increases the time needed to achieve correspondence. Processing time was not a significant problem in this project since tracking was not done in real time. For applications that require rapid real time processing, any predictive matching method may prove too time consuming. Theoretically, predictive methods of establishing correspondence are the best possible among methods. In practice, noise, camera registration, and movement and sampling variability reduce the advantages of being able to find a point's next position by prediction. It may be that a simpler algorithm, such as a modified closest neighbor method with rapid sampling, will be found in the future to be adequate.

CHAPTER 3

MOVEMENT

3.1 The Movement Problem

In this chapter, we want to determine how much of the original three-dimensional scene information can be recovered from a sequence of images. In the first parts of the chapter we develop the matrix mathematics necessary to move objects through space. In these sections, we assume that the translational velocity, rotational velocity, axis of rotation, etc. are given and show that a single four by four matrix in homogeneous coordinates is sufficient to represent any translational or rotational movement. We next develop the equations of central projection and their inverses also using matrices with homogeneous coordinates. We show that full recovery of the original three-dimensional information is impossible without more knowledge than the images alone. We finish the chapter by determining the conditions needed to recover the correct three-dimensional information (to

within a scaling factor) from the projections of points in a sequence of images; the conditions required to find the accurate four by four movement matrix (to within a scaling factor) are also determined. The methods developed in this chapter are quite general and not restricted to any special class of objects, such as blocks. The only requirement is that the objects be rigid.

3.2 Movement Calculations

The problem that is to be solved may be stated very simply: given all the parameters of motion of an object, its speed, direction, speed of rotation, direction of rotation, and axis of rotation, how can we calculate the position of the object in space at every time if we are given the position of the object at an initial time? This problem has been solved and the calculations are given in Newman and Sproull, The Principles of Computer Graphics, [20].

The discussion that follows is facilitated if we consider a specific object in space {(0,0,-10), (0,0,-8), (0,3,-9), (0,3,-10), (-4,0,-10), (-4,0,-8), (-4,3,-9), (-4,3,-10)}. These are the initial coordinates of the

vertices of a truncated wedge. It is convenient to represent a point in space as an "homogeneous coordinate." This means that a fourth coordinate is added so that (-4,3,-10) becomes (-4,3,-10,1), (-4,0,-8) becomes (-4,0,-8,1), etc. In addition, (-4,0,-8,1), (-12,0,-24,3), (8,0,16,-2), (-4α, 0α, -8α, α), α any real, all represent the same point. The reason why this is useful will be explained in the following.

Suppose now that the object is moving in the X-direction 1 unit, the Y-direction 3 units and the Z-direction -2 units, or (1,3,-2,0) altogether. The point (0,0,-10,1), therefore, moves to (1,3,-12,1). In general, any point on the object (X, Y, Z, 1) moves to (X+1, Y+3, Z-2, 1). Furthermore, if T_x is the movement of a point in space in the X-direction, T_y in the Y-direction, and T_z in the Z-direction, then translation of the point (X, Y, Z, 1) is given by,

$$[\,X \quad Y \quad Z \quad 1\,] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

when multiplied out, the result is

$$[X+T_x \quad Y+T_y \quad Z+T_z \quad 1]$$

as desired. Note the simplicity of the matrix for translation. This is due to the use of homogeneous coordinates. If the points had been left in normal three-dimensional form, then the equation would be:

$$[X \quad Y \quad Z] \begin{bmatrix} (X+T_x)/X & 0 & 0 \\ 0 & (Y+T_y)/Y & 0 \\ 0 & 0 & (Z+T_z)/Z \end{bmatrix}$$

$$= [X+T_x \quad Y+T_y \quad Z+T_z],$$

which shows that the matrix must be recalculated for each new point.

Now it is necessary to derive the matrices for rotation. Rotation is more complicated than translation since an axis of rotation must be specified as well as a direction and magnitude. First, we will derive elementary rotation information, then show how to perform a rotation about an arbitrary axis in three-dimensional space, and finally apply the result to the example object.

To help derive rotation results, consider figure two which shows X and Y axis with the Z axis coming out of the page.

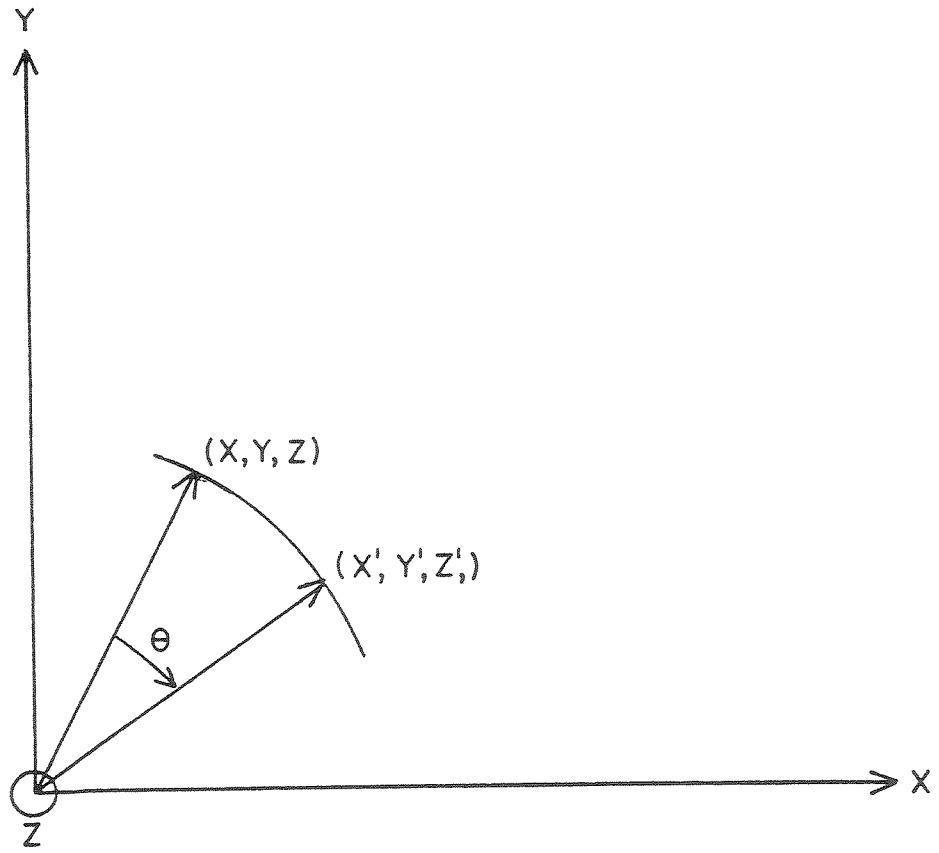


FIGURE 2

THEOREM.

If a point (x,y,z) is rotated clockwise through θ degrees about the Z-axis at a fixed radius, then the new coordinates of the point are related to the coordinates by the formulas

$$X' = X\cos\theta + Y\sin\theta$$

$$Y' = X(-\sin\theta) + Y\cos\theta$$

$$Z' = Z.$$

Normally, the rotational velocity $\omega = d\theta/dt$ is the quantity given for rotating objects where t denotes time. Therefore, the equations of interest are

$$X' = X\cos(\omega t) + Y\sin(\omega t)$$

$$Y' = X[-\sin(\omega t)] + Y\cos(\omega t)$$

$$Z' = Z.$$

In terms of matrices,

$$[X'Y'Z'1] = [X\cos(\omega t)+Y\sin(\omega t) \quad -X\sin(\omega t)+Y\cos(\omega t) \quad Z \quad 1]$$

$$= [X \quad Y \quad Z \quad 1] \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) & 0 & 0 \\ \sin(\omega t) & \cos(\omega t) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

If we know, for example, that rotational velocity is .1 radians/sec., and if we rotate the point (0,3,-10,1) about the Z axis, then the new position of the point after one second will be

(0 x .995 + 3 x .0998, 0 x (-.0998) + 3 x .995,-10,1)

=(.299, 2.985, -10, 1)

since cos 0.1 radian = .995 and sin 0.1 radian = .0998.

In a similar manner, the rotation matrices may be derived for clockwise rotation about the X and Y axes:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\omega t) & -\sin(\omega t) & 0 \\ 0 & \sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ocity

-10,1)

: after

1)

998.

erived

$$R_y = \begin{bmatrix} \cos(\omega t) & 0 & \sin(\omega t) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\omega t) & 0 & \cos(\omega t) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Now that we know the elementary rotations about X, Y, and Z axes we must derive the general matrices for rotation about an arbitrary axis in three-dimensional space.

Assume that the direction of the axis in three-dimensional space is given by a line from the origin to the point (a,b,c) such that the length of the line ($=\sqrt{a^2+b^2+c^2}$) is one; a, b, c are the so-called direction cosines of the axis. Furthermore, assume that the point (X_0, Y_0, Z_0) is a point on the axis. (See figure three.) In addition, given the rotational velocity of the object, we can calculate the positions of the rotated points.

The basic strategy is to move the object and the axis of rotation together until the axis of rotation coincides with one of the axes of the main coordinate system (the Z-axis, say), perform the rotation, and then reverse the sequence of the moves that aligned the axis of

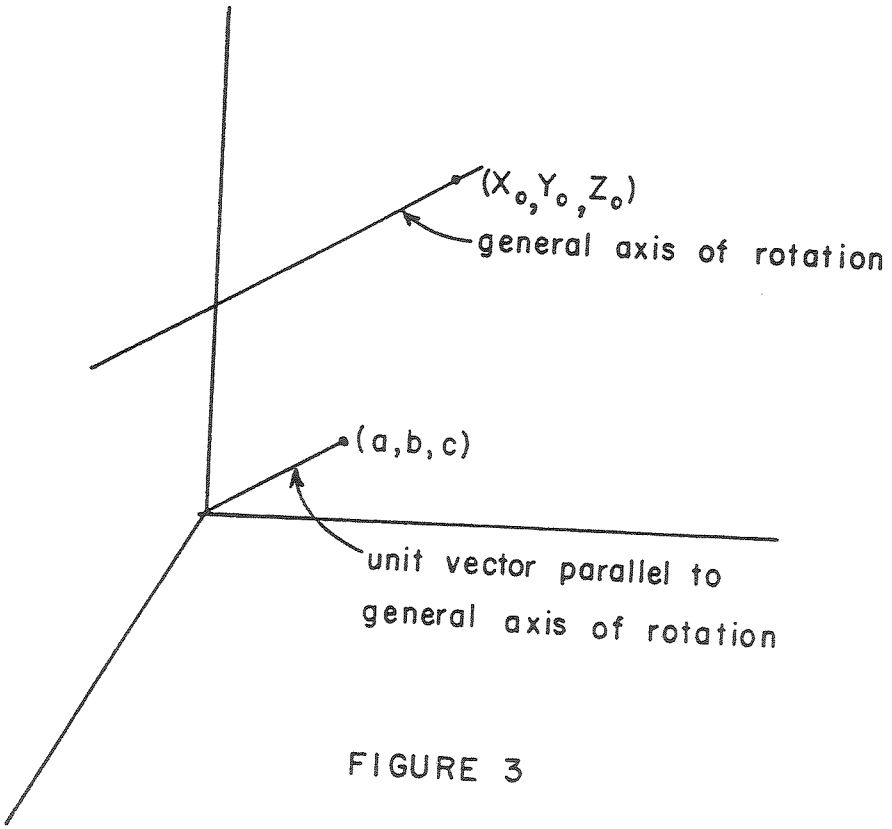

(X_o, Y_o, Z_o)
general axis of rotation
(a,b,c)
unit vector parallel to
general axis of rotation

FIGURE 3

rotation with the Z-axis. To align the axis of rotation with the Z-axis, it is necessary to translate (X_0, Y_0, Z_0) to (0,0,0) and then rotate the axis about the X- and the Y- axes so that the direction cosines are mapped into (0,0,1) - the Z-axis.

To translate (X_0, Y_0, Z_0) to (0, 0, 0) we simply multiply by this matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -X_0 & -Y_0 & -Z_0 & 1 \end{bmatrix} = T_{rotationhelp}$$

We now wish to rotate the rotation axis about the X-axis so that it lies in the X-Z plane (Y=0). Since the X coordinates do not change, the line (0, b, c) and (a,b,c) will rotate in exactly the same way. Figure four shows the angle that (0, b, c) -- and hence (a, b, c) -- must be rotated about the X-axis. The X-axis in this diagram points out of the paper at the reader. From trigonometry we know that $\cos\theta = c/\sqrt{b^2 + c^2}$ and $\sin\theta = -b/\sqrt{b^2 + c^2}$. So the rotation needed about the X-axis is

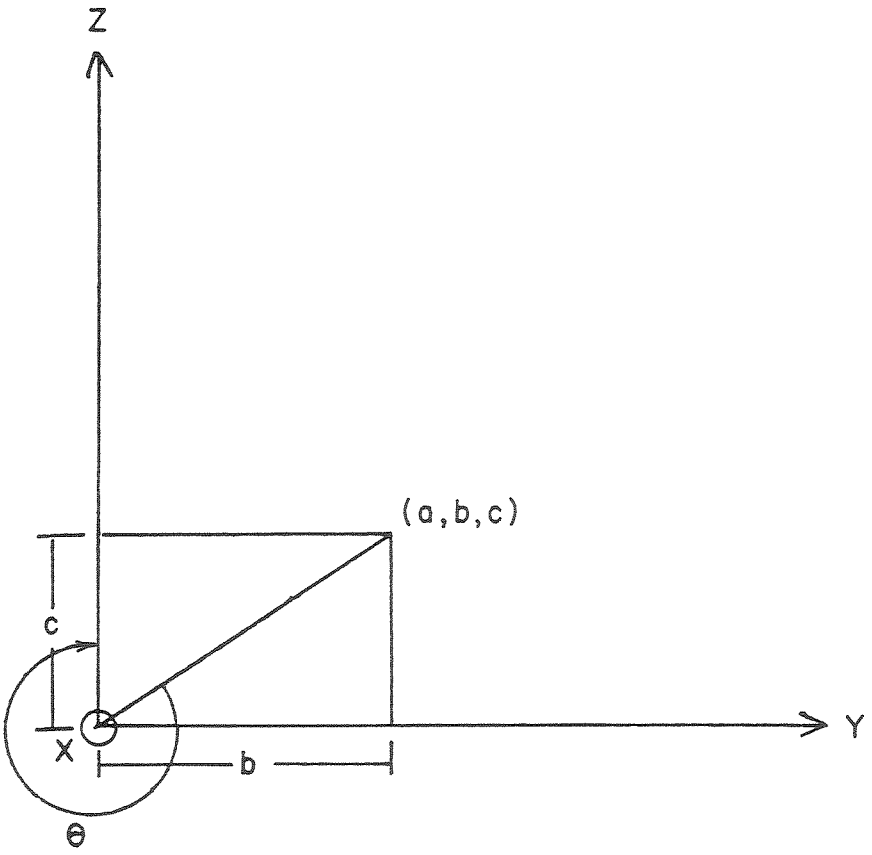


FIGURE 4

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure five shows the resulting position of the axis of rotation. The rotation needed now is through ϕ degrees about the Y-axis. $\cos\phi = \sqrt{b^2+c^2}/\sqrt{a^2+b^2+c^2}$ and $\sin\phi = a/\sqrt{a^2+b^2+c^2}$; since $\sqrt{a^2+b^2+c^2}=1$, $\cos\phi = \sqrt{b^2+c^2}$ and $\sin\phi = a$. Thus the rotation matrix is

$$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Now the axis of rotation is aligned with the Z-axis, and

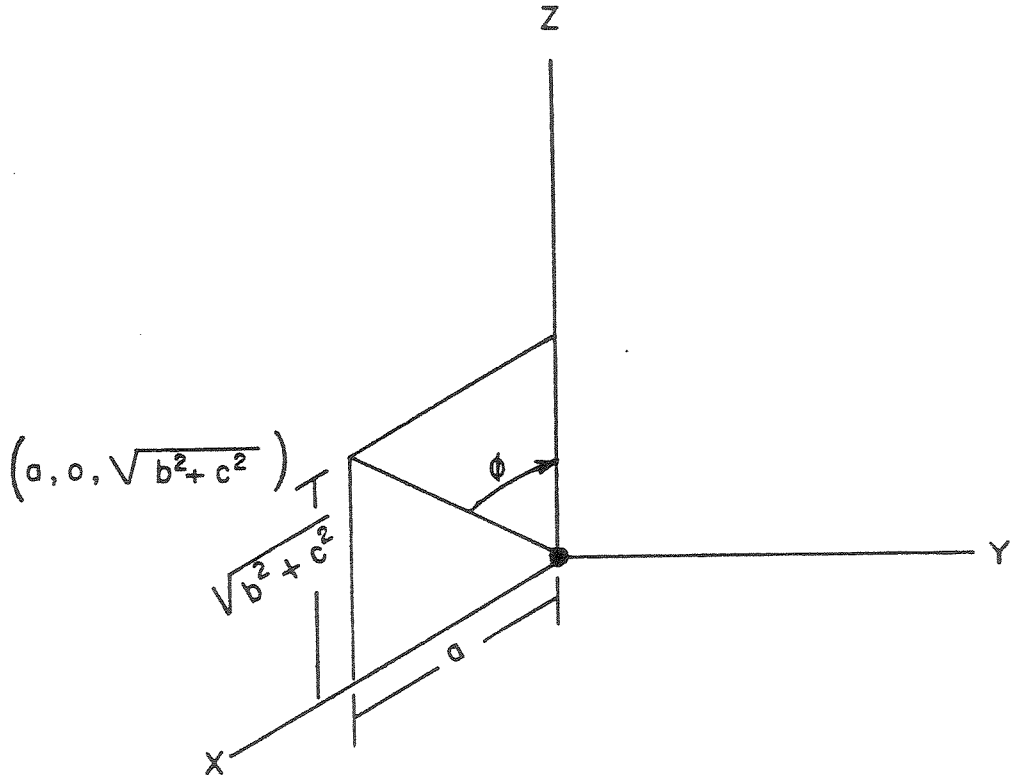


FIGURE 5

we can perform the rotation through ωt degrees. If we assume that the rotational velocity is ω then the rotation matrix is:

$$\begin{bmatrix} \cos(\omega t) & -\sin(\omega t) & 0 & 0 \\ \sin(\omega t) & \cos(\omega t) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = R_{z\omega}.$$

Y

Now that the actual rotation ($R_{z\omega}$) is finished, we must move the axis and the object back to a position so that the axis is in the same place as it was originally. This is achieved by undoing $R_{y\phi}$, $R_{x\theta}$, and $T_{rotationhelp}$, normally denoted by $R_{y\phi}^{-1}$, $R_{x\theta}^{-1}$, $T_{rotationhelp}^{-1}$. To undo a rotation would mean to move a point through an angle opposite to what it has been rotated through. Since $\cos(-\alpha)=\cos\alpha$ and $\sin(-\alpha)=-\sin\alpha$,

$$R^{-1}_{y\phi} = \begin{bmatrix} \cos-\phi & 0 & \sin-\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin-\phi & 0 & \cos-\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$R^{-1}_{x\theta}$ may be similarly derived:

$$R_{x\theta}^{-1}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{rotationhelp}^{-1}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X_0 & Y_0 & Z_0 & 1 \end{bmatrix}$$

We have just proven the following

THEOREM.

To rotate a point about an arbitrary three-dimensional axis in space whose direction cosines are (a, b, c) through point (X_0, Y_0, Z_0) with rotational velocity ω, multiply the coordinates of the point by

$$T_{rotationhelp} R_{x\theta} R_{y\phi} R_{z\omega} R_{y\phi}^{-1} R_{z\theta}^{-1} T_{rotationhelp}^{-1}.$$

A numerical example illustrating rotation and translation is given in appendix three.

3.3 Projection

In the last section we showed how to compute the positions in three-dimensional space of points on an object provided that the velocity, angular velocity and axis of rotation are known. When a picture of points is made, the image of each point is projected onto the focal point of the camera. We want to find the projective coordinates of any three-dimensional point in focal plane coordinates. Since the focal plane is two-dimensional there will only be two coordinates of interest. In effect three coordinates of $q=(x,y,z)$ will be mapped into two screen coordinates $q_s=(x_s, y_s)$, say, where the Z coordinate has become meaningless. We are transforming coordinates from a global coordinate system to a special camera-dependent coordinate system.

Cameras are not normally pivoted about their lens but rather about a gimbal offset from the lens which adds

a complicating factor. We shall assume that the pivot of the camera is at the lens for simplicity's sake. See Duda and Hart [7] for the more complicated computation.

The problem may be broken down into two parts: first we must relate coordinates of the focal plane to the global coordinate system; second, we must project $q=(x,y,z)$ onto the focal plane.

To relate the coordinate system of the camera to the global coordinate system, we must move the lens center to $(0,0,0)$, align it with one of the axes, the Z-axis say, and then rotate about the Z-axis until the X', Y' axes of the focal plane are aligned with the X, Y axes of the global coordinate system. This is precisely the same problem we solved in the previous section. Assume that the lens is at point (X_0, Y_0, Z_0) and that the orientation angles of the optical axis of the camera are θ and ϕ. We know from the previous section that the matrices required to translate the lens center to $(0,0,0)$ and align the optical axis with the Z-axis are

$$T_{(X_0,Y_0,Z_0)} R_{x\theta} R_{y\phi} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -X_0 & -Y_0 & -Z_0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

The lens center is now at (0,0,0) in the global coordinate system. We must now rotate the focal plane about κ degrees with the Z-axis as the rotation axis to align the X', Y' and X, Y axes:

$$R_{Z\kappa} = \begin{bmatrix} \cos\kappa & -\sin\kappa & 0 & 0 \\ \sin\kappa & \cos\kappa & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now the optical axis is aligned with the Z-axis and the focal plane axes are aligned with the global X, Y axes, and the lens is at (0,0,0). This is not sufficient,

however, since the projection of the point is onto the focal plane of the camera, not the lens itself. If we assume that the focal plane is F units away from the lens, then it is necessary to translate the lens -F units along the Z-axis, the necessary matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -F & 1 \end{bmatrix} = T_{FOCAL}$$

We have proven the

THEOREM

A camera has the center of its lens at (X_0, Y_0, Z_0); its optical axis has orientation angles θ and ϕ; X', Y' axes make an angle κ with the X, Y axes; the focal length of the camera is F. To align the focal plane of the camera so that (0',0') of the focal plane maps to (0,0,0) of the global coordinate system and (x',y') maps to (x,y,0), it is necessary to multiply by the matrices

$$T_{(X_0,Y_0,Z_0)} R_{x\theta} R_{y\phi} R_{z\kappa} T_{FOCAL}.$$

The first part of the problem, to relate global to focal plane coordinates, is solved. Now it is necessary to determine the central projection of point q=(x,y,z) onto the focal plane. In figure six we show the physical situation. We assume a front image plane to prevent the annoying problem of image inversion found with rear plane projection models. We want to determine the relationship between x, x', y, y', and z, z'. These relationships will determine the mathematical function of projection. By similar triangles,

$x'/F=x/(F+z)$

$y'/F=y/(F+z)$

$z'=0.$

So $x'=Fx/(F+z)$

$y'=Fy/(F+z)$, and

$z'=0.$

These equations show that given a point q=(x,y,z), its projective coordinates are (Fx/(F+z),Fy/(F+y),0). In homogeneous coordinates, the matrix equation is

$$
\begin{aligned}
&[X \quad Y \quad Z \quad 1]\ P \\
&= [X \quad Y \quad Z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/F \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= [X \quad Y \quad Z \quad (Z/F)+1] \\
&= [X \quad Y \quad Z \quad (Z+F)/F]
\end{aligned}
$$

dividing by the fourth coordinate to return to cartesian coordinates gives the result [Fx/z+F Fy/z+F Fz/z+F 1]. The third coordinate is not zero; this coordinate has become meaningless since all information about depth (Z-coordinate) is lost in the projection.

Given point q=(x,y,z), the complete transformation needed to find the projective coordinates relative to the focal plane is given by the equation

$$q' = qT_{(X_0, Y_0, Z_0)} R_{x\theta} R_{y\phi} R_{z\kappa} T_{FOCAL} P.$$

When the matrices are completely multiplied out and divided by the fourth coordinate, the equations for the

al to

ary

:)

;ical

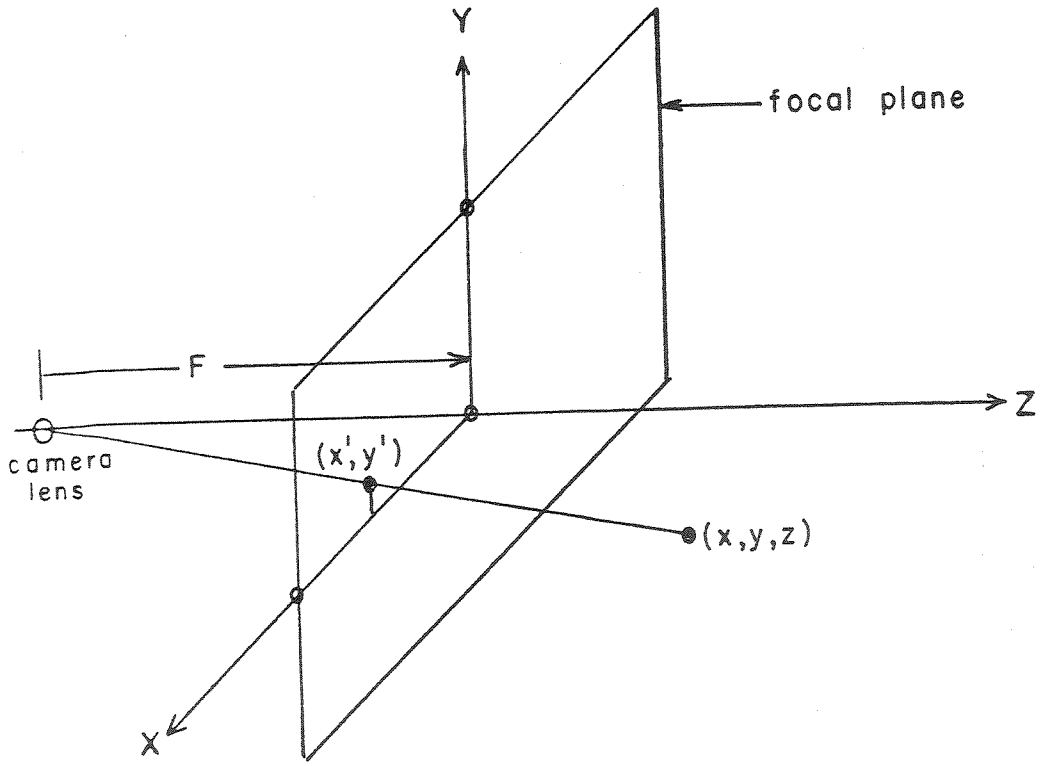
the

plane

nship

s will

By



its

In

FIGURE 6

projective coordinates are

$$x' = F \frac{a_{11}(x-X_0) + a_{12}(y-Y_0) + a_{13}(z-Z_0)}{a_{31}(x-X_0) + a_{32}(y-Y_0) + a_{33}(z-Z_0)} \qquad (+)$$

$$y' = F \frac{a_{21}(x-X_0) + a_{22}(y-Y_0) + a_{23}(z-Z_0)}{a_{31}(x-X_0) + a_{32}(y-Y_0) + a_{33}(z-Z_0)}$$

where $a_{11} = \cos\phi \cos\kappa$

$a_{12} = \sin\theta \sin\phi \cos\kappa + \cos\theta \sin\kappa$

$a_{13} = -\cos\theta \sin\phi \cos\kappa + \sin\theta \sin\kappa$

$a_{21} = -\cos\phi \sin\kappa$

$a_{22} = -\sin\theta \sin\phi \sin\kappa + \cos\theta \cos\kappa$

$a_{23} = \cos\theta \sin\phi \sin\kappa + \sin\theta \cos\kappa$

$a_{31} = \sin\phi$

$a_{32} = -\sin\theta \cos\phi$

$a_{33} = \cos\theta \cos\phi$

A numerical example of the projection process is given in appendix three.

The explanation we have given here for the mathematical model of central projection follows that given in Duda and Hart [7] with modifications to allow the camera to assume an unrestricted position with respect to the global coordinate system. As we shall see, our analysis of movement will depend on this more general treatment. By setting κ to zero in the above equations, the equations found in Duda and Hart [7] are obtained.

3.4 On the Ambiguity of Images of Moving Objects

In his classic paper, Roberts [23] defined scene analysis as the necessary mathematical analysis to locate objects in three-dimensional space from their two-dimensional images. In the previous section we derived the necessary equations to calculate the positions of points on the image plane of a camera provided we know the camera parameters and object movement parameters. In order to do proper three-dimensional scene analysis we want to invert this process, that is, determine the three-dimensional coordinates of object points and the object movement parameters (velocity, rotational velocity and axis of rotation) given the parameters and coordinates of object points in a sequence of images. We also assume that the correspondence of points between scenes is known. The purpose of this section is to show that without further information it is impossible to determine the exact three-dimensional coordinates of an object's points or its movement parameters. We shall show by construction that a sequence of images of a moving object is inherently ambiguous, that such a sequence can represent any number of similar yet distinct objects with different movement parameters.

In order to understand how different moving objects can have the same projective coordinates we first

need to show how different stationary objects can have the same projective coordinates. In the previous section the equation for projection was given by

$$v' = vT_{(X_0,Y_0,Z_0)}R_{x\theta}R_{y\phi}R_{z\kappa}T_{FOCAL}P \ ,$$

where v gives the (x,y,z) coordinates of a point in homogeneous coordinates and v' gives the projective coordinates (x',y') of the point in homogeneous coordinates (information about the Z-coordinates is lost in the projection). Now, however, we are given the problem in reverse: v' is given and we want to determine v. The appropriate equation is

$$v = v' \ P^{-1}T^{-1}_{FOCAL}R^{-1}_{z\kappa}R^{-1}_{y\phi}R^{-1}_{x\theta}T^{-1}_{(X_0,Y_0,Z_0)}$$

where

$$P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/F \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

F=focal length of camera

$$T_{FOCAL}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & F & 1 \end{bmatrix}$$

$$R_{Z\kappa}^{-1} = \begin{bmatrix} \cos\kappa & \sin\kappa & 0 & 0 \\ -\sin\kappa & \cos\kappa & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{y\phi}^{-1} = \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{x\theta}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and}$$

$$T^{-1}_{(X_0,Y_0,Z_0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X_0 & Y_0 & Z_0 & 1 \end{bmatrix}$$

(X_0,Y_0,Z_0) is a point on the optical axis of the camera.

If we let v'=(w'x', w'y', w'z', w') where z' is a free variable (since the Z-coordinate information is lost in projection) and multiply out the matrices, the resulting equation is

$$v=v'\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ -X_0/F & -Y_0/F & -Z_0/F & -1/F \\ X_0+a_{31}F & Y_0+a_{32}F & Z_0+a_{33}F & 1 \end{bmatrix}$$

$$= [w'x' \;\; w'y' \;\; w'z' \;\; w'] \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ -X_0/F & -Y_0/F & -Z_0/F & -1/F \\ X_0 - a_{31}F & Y_0 + a_{32}F & Z_0 + a_{33}F & 1 \end{bmatrix}$$

is a

lost

where the terms a_{11}--a_{33} are the same as those given for projection. When this is multiplied out and divided by the fourth coordinate, the result is

$$\begin{aligned} v_x &= X_0 + F/(F-z')(a_{11}x' + a_{21}y' + a_{31}F) \\ v_y &= Y_0 + F/(F-z')(a_{12}x' + a_{22}y' + a_{32}F) \qquad (*) \\ v_z &= Z_0 + F/(F-z')(a_{13}x' + a_{23}y' + a_{33}F) \end{aligned}$$

See figure 7. These equations (*) give a locus of points that form a straight line in space through (X_0, Y_0, Z_0) and the point (x', y') in the focal plane; each point on the line is determined by a specific value of the free parameter z'. As z' approaches $-\infty$, $F/(F-z')$ approaches 0 and $v=(X_0, Y_0, Z_0)$; if z' is zero then v is (x',y') given in global coordinates. The consequence of these calculations is that given only the image coordinates of a point on an object we can not recover the full three-dimensional information about the object point. The

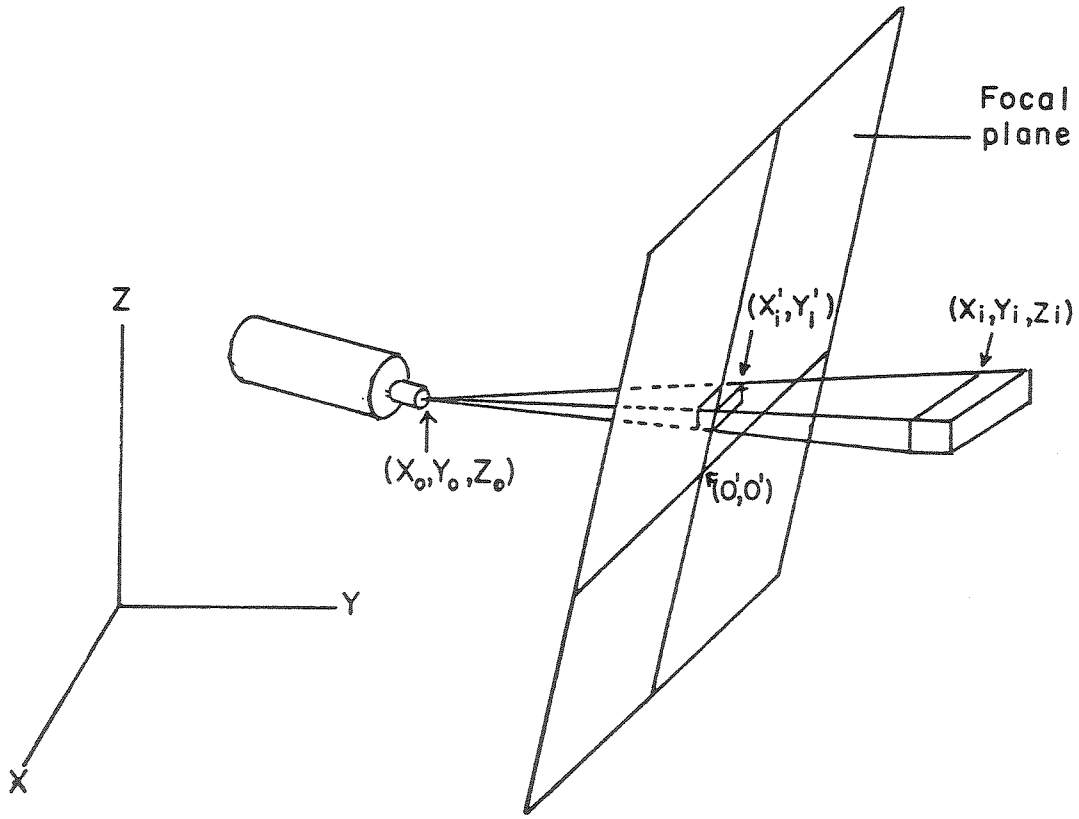
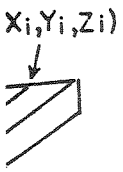


Figure 7

best we can do is find a parameterized equation for the ray on which the point falls.

Now it is easy to see why given only the projected image coordinates of points on an object we cannot uniquely determine the size or distance to the object. There are in fact an unlimited number of objects that give the same projected image.



The main question of interest is whether there are different moving objects that give the same image at all times. The motion of an object can in general be decomposed into a translation and a rotation. With the help of equation (*) above we shall first show that pictures of a translating object are inherently ambiguous and then that pictures of a rotating object are inherently ambiguous. The conclusion will be that a sequence of pictures of a moving object is insufficient to determine the exact movement, distance, or size of the object.

A. The Ambiguity of Images of Translating Objects

We shall use (*) in the following simplified form $v=(X_0,Y_0,Z_0)+\lambda(A,B,C)$, where $\lambda=F/(F-Z')$ and $A=\cos\phi\cos\kappa\, x' - \cos\phi\sin\kappa\, y' + \sin\phi =$ $a_{11}x' + a_{21}y' + a_{31}F$, etc. Consider an object with points $\{(x_1,y_1,z_1),\ldots,(x_n,y_n,z_n)\}$ that has projective coordinates $\{(x_1',y_1'),\ldots,(x_n',y_n')\}$. Assume that its translational velocity is (P,Q,R). For each projected point (x_i',y_i') there is a unique λ_i such that

$$\begin{aligned}(x_i,y_i,z_i) &= (X_0,Y_0,Z_0)+\lambda_i(A_i,B_i,C_i)\\ &= (X_0+\lambda_iA_i,Y_0+\lambda_iB_i,Z_0+\lambda_iC_i).\end{aligned}$$

After time t the position of point (x_i,y_i,z_i) derived from its projection will be

$$(X_0+\lambda_iA_i+tP,Y_0+\lambda_iB_i+tQ,Z_0+\lambda_iC_i+tR).$$

From equations (+) in the Projection section we know that the projective coordinates will be

$$x_i' = F\,\frac{a_{11}(\lambda_iA_i+tP)+a_{12}(\lambda_iB_i+tQ)+a_{13}(\lambda_iC_i+tR)}{a_{31}(\lambda_iA_i+tP)+a_{32}(\lambda_iB_i+tQ)+a_{33}(\lambda_iC_i+tR)}$$

$$y_i' = F\,\frac{a_{21}(\lambda_iA_i+tP)+a_{22}(\lambda_iB_i+tQ)+a_{23}(\lambda_iC_i+tR)}{a_{31}(\lambda_iA_i+tP)+a_{32}(\lambda_iB_i+tQ)+a_{33}(\lambda_iC_i+tR)}$$

We now wish to construct a different object that has a different velocity but the same projective coordinates at all times as the original object. The construction is as follows: for each point (x_i, y_i, z_i) of the original object determined by

$$(x_i, y_i, z_i) = (X_0, Y_0, Z_0) + \lambda_i (A_i, B_i, C_i)$$

derive a new point

$$(x_{si}, y_{si}, z_{si}) = (X_0, Y_0, Z_0) + s\lambda_i (A_i, B_i, C_i)$$

where s is a real number, $s>0$; s is a scaling factor. The new object so constructed is similar to the old object and is smaller if $s<1$, the same object if $s=1$, or larger if $s>1$. Thus, we call the equations that scale all the points of an object the equations of expansion and contraction. The velocity of the new object is (sP, sQ, sR). After time t the position of (x_{si}, y_{si}, z_{si}) will be

$$(X_0 + s\lambda_i A_i + stP, Y_0 + s\lambda_i B_i + stQ, Z_0 + s\lambda_i C_i + stR).$$

The projective coordinates of this point will be

$$x'_{si} = F\,\frac{a_{11}(s\lambda_i A_i+stP)+a_{12}(s\lambda_i B_i+stQ)+a_{13}(s\lambda_i C_i+stR)}{a_{31}(s\lambda_i A_i+stP)+a_{32}(s\lambda_i B_i+stQ)+a_{33}(s\lambda_i C_i+stR)}$$

$$= F\,\frac{s[a_{11}(\lambda_i A_i+tP)+a_{12}(\lambda_i B_i+tQ)+a_{13}(\lambda_i C_i+tR)]}{s[a_{31}(\lambda_i A_i+tP)+a_{32}(\lambda_i B_i+tQ)+a_{33}(\lambda_i C_i+tR)]}$$

$$= F\,\frac{a_{11}(\lambda_i A_i+tP)+a_{12}(\lambda_i B_i+tQ)+a_{13}(\lambda_i C_i+tR)}{a_{31}(\lambda_i A_i+tP)+a_{32}(\lambda_i B_i+tQ)+a_{33}(\lambda_i C_i+tR)}$$

and similarly,

$$y'_{si} = F\,\frac{a_{21}(\lambda_i A_i+tP)+a_{22}(\lambda_i B_i+tQ)+a_{23}(\lambda_i C_i+tR)}{a_{31}(\lambda_i A_i+tP)+a_{32}(\lambda_i B_i+tQ)+a_{33}(\lambda_i C_i+tR)}$$

But now we see that the projective coordinates of any point (x_i, y_i, z_i) of the original object after time t are the same as the projective coordinates of the point (x_{si}, y_{si}, z_{si}) of the constructed object after time t. This proves that there are uncountably many objects all similar to one another with similar translational velocities that have the same projective coordinates. See appendix three for a numerical example.

B. The Ambiguity of Images of Rotating Objects

We want to show now that a sequence of images of a rotating object is inherently ambiguous, that is, that more than one object can produce the same sequence of images. The proof technique we shall use here is similar to that used for translation. A three-dimensional point represented by fixing z' in equation (*) is rotated and its image coordinates determined; another point closer or farther away by a scaling factor s is rotated and its image determined. It is shown that the images of the two points are the same. The proof is more formidable looking for rotation since more information is needed. Rotational velocity and an equation for the axis of rotation are needed; the rotation axis is best specified by the direction cosines of the axis and the three-dimensional coordinates of a point on the axis. The best method to represent a rotation is a matrix using homogeneous coordinates (see Roberts [23] or Duda and Hart [7]). The general form of the rotation matrix is given in appendix one.

Consider an object with points $\{(x_1,y_1,z_1,1),\ldots,(x_n,y_n,z_n,1)\}$ that has projective coordinates $\{(x_1',y_1'),\ldots(x_n',y_n')\}$. Assume that the rotational velocity is ω, that the orientation angles of the axis of rotation are θ and ϕ, and that a point on the axis of

rotation is (p,q,r). Let the ith point be represented by (*) in the following form

$$v_i=(x_i,y_i,z_i,1)=(X_0+\lambda_i A_i, Y_0+\lambda_i B_i, Z_0+\lambda_i C_i, 1)$$

and let the point (p,q,r) on the axis of rotation be represented by

$$(p,q,r)=(X_0+\lambda' A', Y_0+\lambda' B', Z_0+\lambda' C').$$

We now rotate the points of the object by multiplying by the rotation matrix. The details are given in appendix two. When we find the image of the point v_i after rotation, its coordinates are

$$x'_{v_i} = F\,\frac{a_{11}A''+a_{12}B''+a_{13}C''}{a_{31}A''+a_{32}B''+a_{33}C''}$$

$$y'_{v_i} = F\,\frac{a_{21}A''+a_{22}B''+a_{23}C''}{a_{31}A''+a_{32}B''+a_{33}C''}$$

where A'', B'', C'' are complicated terms--see appendix two.

We now wish to construct a different object that will have the same image while rotating. For each point v_i of the original object construct

$$v_{si}=(X_0+s\lambda_i A_i, Y_0+s\lambda_i B_i, Z_0+s\lambda_i C_i, 1).$$

Let the rotational velocity be ω; let the orientation angles of the axis of rotation be θ and ϕ. Let the point on the axis of rotation be

$$(p_s, q_s, r_s)=(X_0+s\lambda' A', Y_0+s\lambda' B', Z_0+s\lambda' C').$$

We can now show that after time t the image of v_{si} after rotation will be the same as for v_i. The details of the rotation of v_{si} are given in appendix two. The projective coordinates of the rotated point applying equations (+) are

$$x'_{v_{si}} = F\,\frac{sa_{11}A'' + sa_{12}B'' + sa_{13}C''}{sa_{31}A'' + sa_{32}B'' + sa_{33}C''}$$

$$= F\,\frac{s[a_{11}A'' + a_{12}B'' + a_{13}C'']}{s[a_{31}A'' + a_{32}B'' + a_{33}C'']}$$

$$= F\,\frac{a_{11}A'' + a_{12}B'' + a_{13}C''}{a_{31}A'' + a_{32}B'' + a_{33}C''}$$

$$= x'_{v_i}$$

$$y'_{v_{si}} = F\,\frac{sa_{21}A'' + sa_{12}B'' + sa_{13}C''}{sa_{31}A'' + sa_{32}B'' + sa_{33}C''}$$

$$= F \frac{a_{11}A'' + a_{12}B'' + a_{13}C''}{a_{31}A'' + a_{32}B'' + a_{33}C''}$$

$$= y'_{v_i}$$

Thus we see that the projective coordinates of the two points are the same. This proves that the image of a rotating object is inherently ambiguous. See appendix three for a numerical example.

In conclusion, it is clear now that the three-dimensional position of points in space along a line of projection is inherently ambiguous, even when the point (or a group of points) is moving. The angular orientation of the axis of rotation and the rotational velocity are not, however, ambiguous.

In the previous sections we have shown that a sequence of images of a translating or rotating object is inherently ambiguous, that is, more than one object can produce the same sequence of images. Since any movement can be broken down as a translation followed by a rotation, this proves that a sequence of images under central projection of a moving object is inherently ambiguous. See appendix three for an example combining both rotation and translation.

3.5 Finding the Movement of an Object From a Sequence of Noise-Free Images

We want to know how much of the original three-dimensional information can be recovered given only the images of a moving object. We have shown that the best possible result would be to find the model and movement of an object determined up to an unknown scaling factor, but we have not yet determined whether we can achieve that good of a result. In this section we show how to find the movement and three-dimensional model of points on an object's surface from a sequence of noise-free images up to a scaling factor; that is, by setting the scaling factor to an arbitrary value we can find a particular movement and model for points on the object.

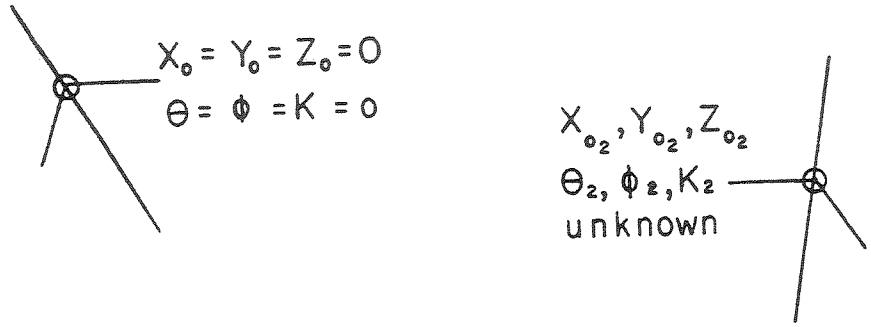
We have assumed that the camera is stationary and the object moves. It is convenient to reformulate the problem such that the object is stationary and the camera moves. This reformulation makes the solution easier.

To solve the problem, two views are needed of five points not all in the same plane. The x, y, z coordinates of each point are variable, so five points produce fifteen variables. The x, y, z coordinates and θ, ϕ, κ orientation angles for each camera position are also variable producing twelve more variables. Thus, there are

a total of twenty-seven variables in the problem. Each point produces two projection equations (given by (+)) per camera position for a total of twenty non-linear equations. We assume that the correspondence of image points between images is known. To make the number of equations and unknowns come out even, seven variables must be known including one variable that will determine the scaling factor. There are a number of ways to set the seven variables correctly. For example, the three-dimensional coordinates of two points and one of the three coordinates of another point are known: (0,0,0), (1,0,0), and (?,?,0). The points (0,0,0) and (1,0,0) fix the X-axis and also the scaling factor; the coordinate system can be rotated about this X-axis until the third point lies in the X-Y plane, thus setting its third coordinate to zero. Although this problem setup is correct, it is difficult to solve numerically since good original estimates for unknown variables (especially camera orientation angles) cannot be determined. Another correct setup for this problem sets all coordinates and orientation angles of the first camera to a value of zero. This sets six variables. The seventh variable and the scaling factor are set by letting the x-coordinate of the second camera be an arbitrary constant (equal to 1.0, say). Reasonable estimates can be made for the unknown

variables, but the difficulty with this problem setup is that in some special cases the x-coordinate of the second camera should be zero. In these cases, setting the x-coordinate to a non-zero constant is incorrect. In fact, it is possible that the x, y, z coordinates of the second camera are all zero (no camera translation) in which case a different means of setting the scaling factor must be sought. Thus we need a different formulation for the problem which will now be explained.

We will set the X_0, Y_0, Z_0 position and θ, ϕ, κ orientation angles of the first camera by making all six variables equal to zero. In addition, the z-component of any one of the five points is set to an arbitrary positive constant. We showed earlier that the best result possible in locating the three-dimensional position of a point on an object is to find (sx, sy, sz) where s is an arbitrary scaling factor. By setting the z-component of the position of a point to an arbitrary constant, we are fixing the scaling factor. Once the z-component of a point is known, the x and y components can also be found using the inverse of the projection equations (by determining lambda from z and the focal length). The situation is shown in figure eight. There are now eighteen projection equations in eighteen unknowns actually, there are twenty equations, but two of them


$X_o = Y_o = Z_o = 0$
$\Theta = \Phi = K = 0$
$X_{o_2}, Y_{o_2}, Z_{o_2}$
Θ_2, Φ_2, K_2
unknown

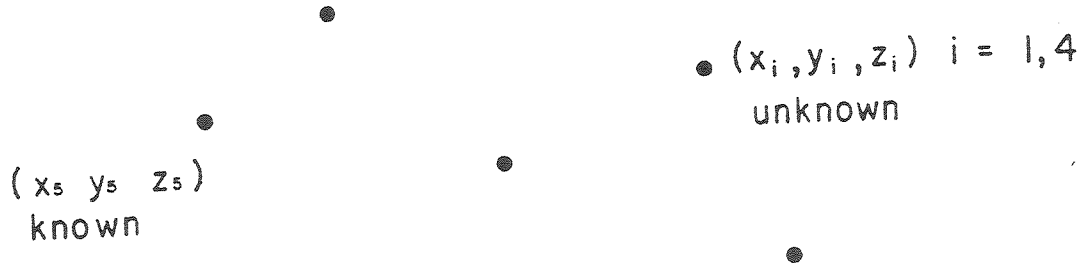

(x_i, y_i, z_i) $i = 1,4$
unknown
$(x_5\ y_5\ z_5)$
known

Figure 8

8

have no unknowns), the equations of projection, however, are non-linear.

Unlike linear equations, there is no developed systematic theory for solving systems of simultaneous non-linear equations. Finding a closed form solution for any system of non-linear equations is rare. Consequently, most non-linear systems are solved using methods developed by numerical analysts that achieve approximate answers. Numerical methods fail sometimes either because they do not converge or converge to the wrong answer. All numerical methods require an initial guess for the unknown parameters. How good the initial guess is normally determines whether the numerical method converges to the correct answer. We have found that the system of non-linear projection equations explained above can be solved by using a modified finite difference Levenberg-Marquardt algorithm due to Brown [4],[14], and [17] without strict descent that minimizes the least squared error of the eighteen equations. This routine is available under IMSL as ZXSSQ [30]. It is a modification of the classical least squared error technique originated by Gauss at the end of the eighteenth century. Brown's method performs a smoothing operation not available with other techniques which normally permits convergence to the correct answer. Least squared error techniques are

1,4

frequently used to solve systems of simultaneous equations that involve observational data (such as photocoordinates from images) known to have errors or noise, although the method works as well for error-free data. Because of error in the data, there is no solution that satisfies all the equations. Adjustments (numerical changes) are made to the values of variables by assuming a model for the distribution or errors in the data (usually, a normal distribution is assumed). These adjustments are designed to minimize the inconsistencies between observations and the answers computed by the equations when the variable values are substituted into the equations. Minimization is effectively achieved by following a gradient operator to a local minimum in the equation space. It can be shown that a series of simple matrix calculations are equivalent to the gradient operation. For full details concerning the least squared error method see Mikhail [19].

The method employed here is iterative and requires an initial guess for each unknown parameter. If we assume that the camera is taking snapshots rapidly, then its position will change only slightly between photographs. In particular, we can make the simplifying assumption that the only camera movement is translation in the x-direction, that is, $\theta_2=\phi_2=\kappa_2= 0$, and the y and z coordinates are no different for the second camera

ions
tes
he

; all
ade
e

gned
and
ole
tion
ator
shown
valent
ing

equires
assume
ts
phs.
on that

position than the first. These assumptions allow us to get reasonable initial estimates using simple parallax. Figure nine shows the physical situation. Let x_1', y_1' be the photocoordinates in the first image of a point with unknown three-dimensional coordinates (x,y,z) and x_2', y_2' be its photocoordinates in the second image. Let X_{0_2} be the x-coordinate of the second camera and let F be the focal length. For the first image, by similar triangles

$$\frac{x}{x_1'} = \frac{z}{F}$$

or

$$x = \frac{x_1' z}{F} .$$

For the second camera,

$$\frac{X_{0_2} - x}{-x_2'} = \frac{z}{F}$$

Substituting for x in this second equation and reducing, we have

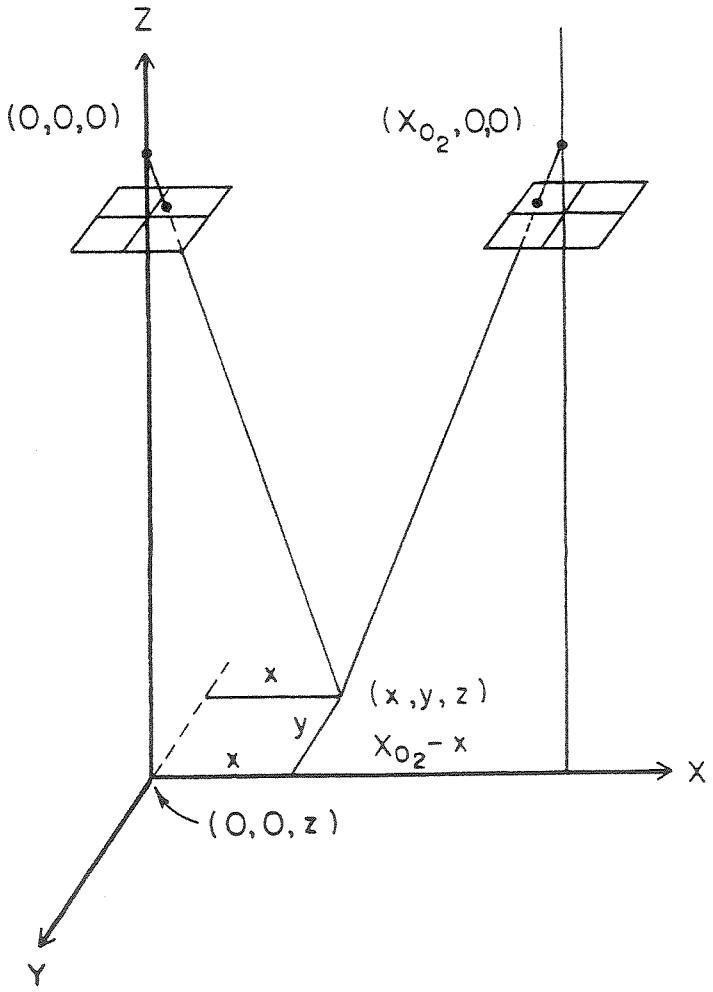


Figure 9

$$z = \frac{F \cdot X_{0_2}}{x'_1 - x'_2}$$

$$= \frac{F \cdot X_{0_2}}{\text{XPARALLAX}} \quad .$$

Substituting for z in the first equation and reducing,

$$x = X_{0_2} \cdot (x'_1/\text{XPARALLAX}) \quad .$$

Using similar triangles again, we can derive the following value for y,

$$y = X_{0_2} \cdot (y'_1/\text{XPARALLAX}) \quad .$$

X_{0_2} is unknown and must itself be estimated before the parallax equations can be applied. This is easily achieved since the three-dimensional coordinates of one point are known: the point for which the z-coordinate is set to an arbitrary constant. Using the known coordinates of this reference point and the parallax equations gives several somewhat different estimates of X_{0_2}.

The possibility of several different values for X_{0_2}, and hence different original estimates for all the points, raises the question of determining which initial estimate is best. Indeed, what is the best means of selecting which of the five object points is to be the reference point whose three-dimensional coordinates are known? We allow each point in turn to become the reference point and then for each guess of X_{0_2} estimate the three-dimensional position of each of the other four points as well as Y_{0_2} and Z_{0_2} (using the inverse of the central projection equations). The least squared error of this set of estimates is calculated by substituting the estimates for unknown variables in the equations for central projection (assuming for camera two that $\theta_2 = \phi_2 = \kappa_2 = 0.0$ degrees) and the answers differenced with the observed projective coordinates of the points. The errors are all squared and added together; the set of estimates with the least squared error is taken as the best initial guess. There are only fifteen sets of initial guesses so this pre-processing time is quite small, and without it, the initial estimate is not always good enough to cause convergence to the right answer. Experiments show that original estimates that are extremely close to the correct answer frequently do not converge to the correct answer whereas estimates that are

not especially close to the correct answer but that have good projective coordinates that are close to correct (low squared error) will converge to the correct answer.

Brown's method without strict descent was very successful in converging to the correct answer in a large number of trials with analytically correct data. (The data were generated by a computer program that takes as input the three-dimensional position of points on an object, the movement of the object, and the position of the camera; the program moves the object and mathematically projects an image at prescribed time intervals. The photocoordinates are used to ten places of accuracy.) The method failed to find the exact answer only in cases where a moving object was rotating (no translation) and the axis of rotation passed through the lens of the camera. The answer that this numerical method computes gives both a model for the object and its movement since the movement of the camera is also calculated. By negating all values of the solution, a second solution is attained. This solution amounts to setting the z coordinate of the reference point to the negation of the original arbitrary scalar.

It should be noted that a large number of other methods were tried on this system of equations: Brown's algorithm with strict descent, Newton's method, normal

least squared error, Fletcher-Powell [9], and several other numerical methods designed to deal with simultaneous non-linear equations. Newton's method and least squares were also used with many different step lengths. Some of these methods converge to answers that are close to correct, but others do not converge at all. Only Brown's method without strict descent regularly converges to the correct answer.

3.6 Finding Answers from Noisy Images

We have been making two very important assumptions: that the objects being observed are rigid and that the images of the object are noise free and thus completely accurate. The first assumption is an important restriction since the problem solution presented does not work with images of moving, highly non-rigid objects. The second assumption is not reasonable. In this section we shall examine and solve the problems introduced when the images are noisy.

In effect, when noise is added to the images of the object it is equivalent to taking perfect photos of an object that is not quite rigid -- jello-like is an apt

metaphor. Thus we see that the rigidity and noise-free images assumptions are interrelated. To test the effect of sensor noise and digitization error on the numerical method described in the previous section, from one to four pixels were randomly added or subtracted from the exact photocoordinate data for a moving object.

One of the main reasons for using a least squared error technique to solve a problem is to make adjustments to observations (i. e., image data points) that contain error (noise). Adjustment is only possible, however, when there are more equations than unknowns. Two views of five points is therefore inadequate for noisy data since there are the same number of equations as unknowns. Two views of six points or three views of four points produce twenty two equations in twenty one unknowns using the same problem model discussed in the previous section. Examination of experimental runs using over-determined systems of equations shows that minimal over-determination is not very accurate. It is only with considerable over-determination (two views of twelve or even fifteen points; three views of seven or eight points) that the results become accurate. Appendix four has a graphical comparison of the experiments run with over-determined sets of equations. For the two views case, the model of the object improves considerably, and the camera position

improves somewhat as the number of points increases. For three views, the opposite effect seems to hold: the camera positions' accuracies improve considerably and the model points' positions (originally fairly good) improve somewhat. Clearly, attaining good accuracy depends on considerable over-determination. It should be noted that in some examples we tried, Brown's method without strict descent would not converge using a reference point that produced the minimum squared error from its original estimate. By trying a different reference point, and thus a different initial estimate, which produced a small squared error, convergence to a satisfactory result was achieved.

In addition to synthetic data, an experiment was run using laboratory images, 108x108 pixels, from a rather noisy image dissector camera. The images used appear in Roach and Aggarwal [22] and are reproduced in appendix five. In general, these images contain too few points to assure accurate results. The images of three different objects were used with three views of four points and three views of five points. The results verify our findings that too few points result in an inaccurate answer.

The problem setup we have described requires that we set the z-coordinate of the reference point to some

numeric value, usually arbitrary. In some cases, however, there may be a priori knowledge: the distance between two points, or the distance from the camera to a point on the object in one of the images (using a ranging device) may be known. In these cases, the model and movement of the object is found as shown above using an arbitrary constant for the z-coordinate of the reference point. The scale of the attained solution can then be changed using equations of expansion and contraction as explained in the section "On the Ambiguity of Images of Moving Objects" until the distance between two points in the model or the distance from the camera to a model point is as desired.

Conclusions

This work is directly related to the problems encountered by researchers in optic flow studies, as mentioned in the introduction, although the findings here have no bearing on finding the "focus of expansion" for a translating camera. J. J. Gibson's texture gradients [10], however, seem to be very much related to the work presented here.

Roberts [23] originally used least squares analysis together with simple models and a support assumption to locate objects in space given one image of six points on the object. This study shows that it is possible with more than one view to dispense with the model. It is possible, in fact, to determine the three-dimensional realtionship of the points on the object as well as its movement (up to a scale factor) from the multiple views. This is also the conclusion of Ullman's recently available dissertation [25b] which uses a novel closed form solution for restricted motions of objects from noise-free central projection images. For unrestricted motion, his method requires an impractical computation in a very large (infinite) search space. Another problem with central projection according to Ullman is that perspective effects are often small (especially for small objects), and thus noise makes central projection an unsuitable model for determining three-dimensional structure and movement. Obviously, no method can succeed in the case where noise effects overwhelm image changes due to motion. We have shown in this thesis that given noisy central projection images, the movement and three-dimensional relationship of points can be attained only when there are considerably more equations than unknowns. We have assumed that noise does

not overwhelm image changes caused by the movement of objects. We chose to use synthetic data originally to ensure that the numerical method adopted converges to the correct answer and later as a control over the accuracy of the answer when noise was added to the data. The need for considerable over-determination and thus the images of many points re-emphasizes a problem that no one has solved since Roberts' paper: how to find the correct tokens on the surface of the object in images (in our case, the same tokens in every image). Note that we are not referring to the well known correspondence problem. Without the ability to determine reliably the same feature points in each image, the whole analysis scheme fails. Finding points on blocks as in figure one is easy; finding identical points in each image of a sequence from the real world is considerably more difficult. Future research will have to devise a reliable low-level processing solution for this problem.

CHAPTER 4

SUMMARY

This dissertation has presented and solved a number of problems related to determining the three-dimensional model and motion of an object given only a sequence of different two-dimensional images of the object. The most important and difficult problem, the development of a formalized method to permit mathematical analysis, involves locating the same feature points on the surface of an object in each image. Roberts [23] also used this idea together with a "support assumption" (which sets the scaling factor by positing that an object in an image sits on a known plane or tabletop) to find the three-dimensional position of a block from a single image. Once the basic decision to use feature points is made, the steps needed to process a sequence of images are: segmentation of each image (possibly a differencing operation between images to determine what objects have moved), extraction of feature points from each imaged object's surface, correspondence of feature points between images, and mathematical analysis to determine the three-dimensional model and movement of the points.

Low-level processing problems leading up to the extraction of feature points from images of natural scenes have been ignored in this dissertation. Ideally, the feature point extraction process should extract points only on the surfaces of moving objects; it should ignore immobile objects. Giving the correspondence process extra feature points of the background could only serve to slow down the process and possibly cause it to fail. The segmentation process, then, should be designed to find only moving objects in a sequence of images. This is the subject of the research reported in Potter [21] and in Jain, Martin, and Aggarwal [11].

An original and powerful solution to the correspondence problem for images of objects moving in space was given for the blocks world in Roach and Aggarwal [22]. In this dissertation, a simple extension to the methods described in that paper was given to determine the correspondence of points instead of objects. The methods described are heuristic. More mathematical approaches employing projective geometry (see Duda and Hart [7], chapter eleven for a fine discussion) have failed, so far at least, to produce a satisfactory result.

In mathematically analyzing the motion problem, this dissertation has established for the first time a practical means for solving the motion problem for

noise-free and noisy sequences of central projection images of rigid objects. The solution only assumes that objects are rigid; there are no special assumptions as to object type (blocks), etc. The solution given determines the relative movement between the camera and an object. Thus, it would be possible to study the problem of a camera moving through an environment containing mobile and immobile objects. This might be a rather simple problem, but it should be investigated. The problems of moving jointed objects (such as animals) and of non-rigid objects that vary slowly (like clouds) have not been approached in the research for this dissertation. Also worthy of further investigation are problems dealing with non-rigid objects such as springs, plants that bend in the wind, etc. Although the mathematical techniques and results developed here are for rigid objects, they are quite general and powerful. They will be basic building blocks for future research in jointed movement if not also for less rigid objects.

Psychologists, of course, have their own theories concerning human perception of movement. The current vogue in psychological studies concerning low-level visual processes is a paradigm that assumes the eye is computing fourier transforms on the input (see, for example, Lindsay and Norman [15], second edition). This paradigm appears

to be antithetical to the basic premise of this paper that low-level processing must extract feature points from images. It should be interesting to see how psychologists account for movement computations with their paradigm, or whether the fourier transform approach is compatible with the feature point extraction method of determining an object's motion.

APPENDIX 1

Matrix for Rotation about a General Axis in Space

This appendix presents the matrix necessary to rotate a point about an arbitrary axis in three-dimensional space. An explanation of the matrices necessary to lead to this result are given in Newman and Sproull [20]. Here we give the movement matrix in terms of both direction cosines and orientation angles of the axis of rotation. Let ω be the rotational velocity and (x,y,z) a point on the axis of rotation. The rotation matrix is

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

If (a,b,c) are the direction cosines of the axis of rotation and $v = \sqrt{b^2+c^2}$, then

$$a_{11} = v^2 \cos\omega t + a^2$$

$$a_{12} = ab \cos\omega t - c \sin\omega t + ab$$

$$a_{13} = -ac \cos\omega t + b \sin\omega t + ac$$

$$a_{14} = 0$$

$$a_{21} = -ab \cos\omega t + c \sin\omega t + ab$$

$$a_{22} = (ab/v)^2 \cos\omega t + (c/v)^2 \cos\omega t + b^2$$

$$a_{23} = a^2bc/v^2 \cos\omega t - ac^2/v^2 \sin\omega t - ab^2/v^2 \sin\omega t - bc/v^2 \cos\omega t + bc$$

$$a_{24} = 0$$

$$a_{31} = -ac \cos\omega t - b \sin\omega t + ac$$

$$a_{32} = a^2bc/v^2 \cos\omega t + ab^2/v^2 \sin\omega t + ac^2/v^2 \sin\omega t - bc/v^2 \cos\omega t + bc$$

$$a_{33} = (ac/v)^2 \cos\omega t + (b/v)^2 \cos\omega t + c^2$$

$$a_{34} = 0$$

$$a_{41} = -v^2x \cos\omega t + aby \cos\omega t + acz \cos\omega t - cy \sin\omega t + bz \sin\omega t - a^2x - aby - acz + x$$

$$a_{42} = (abx - a^2b^2y/v^2 - a^2bcz/v^2) \cos\omega t - ab^2z/v^2 \sin\omega t + (cx - ac^2z/v^2) \sin\omega t + (-c^2y/v^2 + bcz/v^2) \cos\omega t - abx - b^2y - bcz + y$$

$$a_{43} = -(-acx + a^2bcy/v^2 + a^2c^2z/v^2) \cos\omega t + (ac^2y/v^2) \sin\omega t - (bx - ab^2y/v^2) \sin\omega t + (bcy/v^2 - b^2z/v^2) \cos\omega t - acx - bcy - c^2z + z$$

$$a_{44} = 1.$$

If θ, ϕ, κ are the orientation angles of the axis of rotation, then

$$a_{11} = \cos^2\phi \cos\omega t + \sin^2\phi$$

$$a_{12} = \sin\theta \cos\phi \cos\omega t - \cos\theta \cos\phi \sin\omega t - \sin\theta \sin\phi \cos\phi$$

$$a_{13} = -\cos\theta \sin\phi \cos\phi \cos\omega t - \sin\theta \cos\phi \sin\omega t + \cos\theta \cos\phi \sin\phi$$

$$a_{14} = 0$$

$$a_{21} = \sin\theta \sin\phi \cos\phi \cos\omega t + \cos\theta \cos\phi \sin\omega t - \sin\theta \sin\phi \cos\omega t$$

$$a_{22} = \sin^2\theta \sin^2\phi \cos\omega t + \sin\theta \cos\theta \sin\phi \sin\omega t - \cos\theta \sin\theta \sin\phi \sin\omega t + \cos^2\theta \cos\omega t + \sin^2\theta \cos\phi \cos\omega t$$

$$a_{23} = -\sin\theta \cos\theta \sin^2\phi \cos\omega t - \cos^2\theta \sin\phi \sin\omega t - \sin^2\theta \sin\phi \sin\omega t + \sin\theta \cos\theta \cos\omega t - \sin\theta \cos\theta \cos\phi \cos\omega t$$

$$a_{24} = 0$$

$$a_{31} = -\cos\theta \sin\phi \cos\phi \cos\omega t + \sin\theta \cos\phi \sin\omega t + \cos\theta \cos\phi \sin\phi$$

$$a_{32} = -\sin\theta \cos\theta \sin^2\phi \cos\omega t + \sin^2\theta \sin\phi \sin\omega t + \cos^2\theta \sin\phi \sin\omega t + \sin\theta \cos\theta \cos\omega t - \sin\theta \cos\theta \cos^2\phi$$

$$a_{33} = \cos^2\theta \sin^2\phi \cos\omega t - \sin\theta \cos\theta \sin\phi \sin\omega t + \sin\theta \cos\theta \sin\omega t + \sin^2\theta \cos\omega t + \cos^2\theta \cos^2\phi$$

$$a_{34} = 0$$

$$a_{41} = X_0(1-\cos^2\phi \cos\omega t - \sin^2\phi) + Y_0(-\sin\theta \sin\phi \cos\phi \cos\omega t - \cos\theta \cos\phi \sin\omega t + \cos\theta \sin\phi \cos\phi) + Z_0(\cos\theta \sin\phi \cos\phi \cos\omega t - \sin\theta \cos\phi \sin\omega t - \cos\theta \sin\phi \cos\phi)$$

$$a_{42} = Y_0(1-\sin^2\theta \sin^2\phi \cos\omega t - \sin\theta \cos\theta \sin\phi \sin\omega t$$
$$+ \sin\theta \cos\theta \sin\phi \sin\omega t - \cos^2\theta \cos\omega t - \sin^2\theta \cos^2\phi)$$
$$+ X_0(\sin\theta \sin\phi \cos\phi \cos\omega t + \cos\theta \cos\phi \sin\omega t$$
$$+ \sin\theta \sin\phi \cos\phi)$$
$$+ Z_0(\sin\theta \cos\theta \sin^2\phi \cos\omega t - \sin^2\theta \sin\phi \sin\omega t$$
$$- \cos^2\theta \sin\phi \sin\omega t - \sin\theta \cos\theta \cos\omega t$$
$$+ \sin\theta \cos\theta \cos^2\phi)$$

$$a_{43} = Z_0(1-\cos^2\theta \sin^2\phi \cos\omega t + \sin\theta \cos\theta \sin\phi \sin\omega t$$
$$- \sin\theta \cos\theta \sin\phi \sin\omega t - \sin^2\theta \cos\omega t - \cos^2\theta \cos^2\phi)$$
$$+ X_0(\cos\theta \sin\phi \cos\phi \cos\omega t + \sin\theta \cos\phi \sin\omega t$$
$$- \cos\theta \sin\phi \cos\phi)$$
$$+ Y_0(\sin\theta \cos\theta \sin^2\phi \cos\omega t + \cos^2\theta \sin\phi \sin\omega t$$
$$+ \sin^2\theta \sin\phi \sin\omega t - \sin\theta \cos\theta \cos\omega t$$
$$+ \sin\theta \cos\theta \cos^2\phi)$$

$$a_{44} = 1.$$

APPENDIX 2

Details of the Ambiguity of Images of Rotating Objects

Consider an object with points $\{(x_1,y_1,z_1,1),\ldots,(x_n,y_n,z_n,1)\}$ that has projective coordinates $\{(x_1',y_1'),\ldots(x_n',y_n')\}$. Assume that the rotational velocity is ω, that the orientation angles of the axis of rotation are θ and ϕ, and that a point on the axis of rotation is (p,q,r). Let the ith point be represented by (*) in the following form

$$v_i=(x_i,y_i,z_i,1)=(X_0+\lambda_i A_i, Y_0+\lambda_i B_i, Z_0+\lambda_i C_i, 1)$$

and let the point (p,q,r) on the axis of rotation be represented by

$$(p,q,r)=(X_0+\lambda' A', Y_0+\lambda' B', Z_0+\lambda' C').$$

We now rotate the points of the object by multiplying by the rotation matrix. Note: the proof given here uses direction cosines (d, e, f) for the angular orientation of the axis of rotation; $\cos\theta = f/\sqrt{e^2+f^2}$, $\sin\theta = -e/\sqrt{e^2+f^2}$, $\cos\phi = \sqrt{e^2+f^2}$, $\sin\phi = d$; (p, q, r) is the point on the axis

of rotation. After time t the position of the rotated point is

$$
\begin{aligned}
&((X_0 + \lambda_i A_i)(v^2 \cos \omega t + d^2) \\
&+ (Y_0 + \lambda_i B_i)(-de \cos \omega t + f \sin \omega t + de) \\
&+ (Z_0 + \lambda_i C_i)(-df \cos \omega t - e \sin \omega t + df) \\
&- v^2 p \cos \omega t + deq \cos \omega t + dfr \cos \omega t - fq \sin \omega t \\
&+ er \sin \omega t - d^2 p - deq - dfr + p, \\
&\quad (X_0 + \lambda_i A_i)(-de \cos \omega t - f \sin \omega t + de) \\
&+ (Y_0 + \lambda_i B_i)(d^2e^2/v^2 \cos \omega t + f^2/v^2 \cos \omega t + e^2) \\
&+ (Z_0 + \lambda_i C_i)(d^2ef/v^2 \cos \omega t - de^2/v^2 \sin \omega t \\
&+ df^2/v^2 \sin \omega t - ef/v^2 \cos \omega t + ef) \\
&+ (dep - d^2e^2q/v^2 - d^2efr/v^2) \cos \omega t \\
&+ (-de^2r/v^2 + fp - df^2r/v^2) \sin \omega t \\
&- (f^2q/v^2 - efr/v^2) \cos \omega t - dep - e^2q - efr + q, \\
&\quad (X_0 + \lambda_i A_i)(-df \cos \omega t + e \sin \omega t + df) \\
&+ (Y_0 + \lambda_i B_i)(d^2ef/v^2 \cos \omega t - df^2 \sin \omega t \\
&- de^2/v^2 \sin \omega t - ef/v^2 \cos \omega t + ef) \\
&+ (Z_0 + \lambda_i C_i)(d^2f^2/v^2 \cos \omega t + e^2/v^2 \cos \omega t + f^2) \\
&- (-dfp + d^2efq/v^2 + d^2f^2r/v^2) \cos \omega t \\
&- (-df^2q/v^2 + ep - de^2q/v^2) \sin \omega t \\
&+ (efq/v^2 - e^2r/v^2) \cos \omega t \\
&-dfp - efq - f^2r + r, \\
&1).
\end{aligned}
$$

By substituting $X_0 + \lambda' A'$ for p, $Y_0 + \lambda' B'$ for q, and $Z_0 + \lambda' C'$ for r and cancelling terms, the above vector becomes:

$$
\begin{aligned}
&(X_0 + \lambda_i A_i (v^2 \cos \omega t + d^2) \\
&\quad + \lambda_i B_i (-de \cos \omega t + f \sin \omega t + de) \\
&\quad + \lambda_i C_i (-df \cos \omega t - e \sin \omega t + df) \\
&\quad - v^2 \lambda' A' \cos \omega t + de\lambda' B' \cos \omega t + df\lambda' C' \cos \omega t \\
&\quad - f\lambda' B' \sin \omega t + e\lambda' C' \sin \omega t - d^2 \lambda' A' - de\lambda' B' \\
&\quad - df\lambda' C' + \lambda' A', \\
&Y_0 + \lambda_i A_i (-de \cos \omega t - f \sin \omega t + de) \\
&\quad + \lambda_i B_i (d^2 e^2 / v^2 \cos \omega t + f^2 / v^2 \cos \omega t + e^2) \\
&\quad + \lambda_i C_i (d^2 ef / v^2 \cos \omega t + de^2 / v^2 \sin \omega t \\
&\quad + df^2 / v^2 \sin \omega t - ef / v^2 \cos \omega t + ef) \\
&\quad + (de\lambda' A' - d^2 e^2 \lambda' B' / v^2 - d^2 ef\lambda' C' / v^2) \cos \omega t \\
&\quad + (-de^2 \lambda' C' / v^2 + f\lambda' A' - df^2 \lambda' C' / v^2) \sin \omega t \\
&\quad - (f^2 \lambda' B' / v^2 - ef\lambda' C' / v^2) \cos \omega t - de\lambda' A' - e^2 \lambda' B' \\
&\quad - ef\lambda' C' + \lambda' B', \\
&Z_0 + \lambda_i A_i (-df \cos \omega t + e \sin \omega t + df) \\
&\quad + \lambda_i B_i (d^2 ef / v^2 \cos \omega t - df^2 / v^2 \sin \omega t \\
&\quad - de^2 / v^2 \sin \omega t - ef / v^2 \cos \omega t + ef) \\
&\quad + \lambda_i C_i (d^2 f^2 / v^2 \cos \omega t + e^2 / v^2 \cos \omega t + f^2) \\
&\quad - (-df\lambda' A' + d^2 ef\lambda' B' / v^2 + d^2 f^2 \lambda' C' / v^2) \cos \omega t \\
&\quad - (-df^2 \lambda' B' / v^2 + e\lambda' A' - de^2 \lambda' C' / v^2) \sin \omega t \\
&\quad + (ef\lambda' B' / v^2 - e^2 \lambda' C' / v^2) \cos \omega t \\
&\quad - df\lambda' A' - ef\lambda' B' - f^2 \lambda' C' + \lambda' C',
\end{aligned}
$$

1).

To simplify matters, rewrite this vector as

$$(X_0+A'',\ Y_0+B'',\ Z_0+C'',1).$$

When we find the image of the point v_i after rotation, its coordinates are

$$x'_{v_i} = F\frac{a_{11}A''+a_{12}B''+a_{13}C''}{a_{31}A''+a_{32}B''+a_{33}C''}$$

$$y'_{v_i} = F\frac{a_{21}A''+a_{22}B''+a_{23}C''}{a_{31}A''+a_{32}B''+a_{33}C''}$$

We now wish to construct a different object that will have the same image while rotating. For each point v_i of the original object construct

$$v_{si}=(X_0+s\lambda_i A_i, Y_0+s\lambda_i B_i, Z_0+s\lambda_i C_i, 1).$$

Let the rotational velocity be ω; let the orientation angles of the axis of rotation be θ and ϕ. Let the point on the axis of rotation be

$$(p_s, q_s, r_s)=(X_0+s\lambda'A', Y_0+s\lambda'B', Z_0+s\lambda'C').$$

We can now show that after time t the image of v_{si} after rotation will be the same as for v_i.

Now we want to rotate the scaled point v_{si} with

$$v_{si} = (X_0+s\lambda_i A_i, Y_0+s\lambda_i B_i, Z_0+s\lambda_i C_i, 1).$$

The rotation of v_{si} is given by

$$
\begin{aligned}
&((X_0 + s\lambda_i A_i)(v^2 \cos \omega t + d^2) \\
&\quad + (Y_0 + s\,\lambda_i B_i)(-de \cos \omega t + f \sin \omega t + de) \\
&\quad + (Z_0 + s\,\lambda_i C_i)(-df \cos \omega t - e \sin \omega t + df) \\
&\quad - v^2(X_0 + s\lambda' A') \cos \omega t + de(Y_0 + s\lambda' B') \cos \omega t \\
&\quad + df(Z_0 + s\lambda' C') \cos \omega t - f(Y_0 + s\lambda' B') \sin \omega t \\
&\quad + e(Z_0 + s\lambda' C') \sin \omega t - d^2(X_0 + s\lambda' A') \\
&\quad + de(Y_0 + s\,\lambda' B') - df(Z_0 + s\,\lambda' C') + X_0 + s\lambda' A', \\
&(Y_0 + s\lambda_i A_i)(-de \cos \omega t - f \sin \omega t + de) \\
&\quad + (Y_0 + s\lambda_i B_i)(d^2e^2/v^2 \cos \omega t + f^2/v^2 \cos \omega t + e^2) \\
&\quad + (Z_0\, s\lambda_i C_i)(d^2ef/v^2 \cos \omega t + de^2/v^2 \sin \omega t \\
&\quad + df^2/v^2 \sin \omega t - ef/v^2 \cos \omega t + ef) \\
&\quad + (de(X_0 + s\lambda' A') - d^2e^2(Y_0 + s\lambda' B')/v^2 \\
&\quad - d^2ef(Z_0 + s\lambda' C')/v^2) \cos \omega t \\
&\quad + (-de^2(Z_0 + s\lambda' C')/v^2 + f(X_0 + s\lambda' A') \\
&\quad - df^2(Z_0 + s\lambda' C')/v^2) \sin \omega t \\
&\quad - (f^2(Y_0 + s\lambda' B')/v^2 - ef(Z_0 + s\lambda' C')/v^2) \cos \omega t \\
&\quad - de(X_0 + s\lambda' A') - e^2(Y_0 + s\lambda' B') \\
&\quad - ef(Z_0 + s\lambda' C') + Y_0 + s\lambda' B', \\
&(Z_0 + s\lambda_i A_i)(-df \cos \omega t + e \sin \omega t + df) \\
&\quad + (Y_0 + s\lambda_i B_i)(d^2ef/v^2 \cos \omega t - df^2/v^2 \sin \omega t \\
&\quad - de^2/v^2 \sin \omega t - ef/v^2 \cos \omega t + ef) \\
&\quad + (Z_0 + s\lambda_i C_i)(d^2f^2/v^2 \cos \omega t + e^2/v^2 \cos \omega t + f^2) \\
&\quad - (-df(X_0 + s\lambda' A') + d^2ef(Y_0 + s\lambda' B') \\
&\quad + d^2f^2(Z_0 + s\lambda' C')/v^2) \cos \omega t
\end{aligned}
$$

$$
\begin{aligned}
&- (-df^2(Y_0 + s\lambda'B')/v^2 + e(X_0 + s\lambda'A') \\
&- def(Z_0 + s\lambda'C'/v^2)) \sin \omega t \\
&+ (ef(X_0 + s\lambda'B')/v^2 - e^2(Z_0 + s\lambda'C')/v^2) \cos \omega t \\
&- df(X_0 + s\lambda'A') - ef(Y_0 + s\lambda'B') \\
&- f^2(Z_0 + s\lambda'C')\ Z_0 + s\lambda'C',
\end{aligned}
$$

1)

$$
\begin{aligned}
= (X_0 &+ s\lambda_i A_i(v^2 \cos \omega t + d^2) \\
&+ s\lambda_i B_i(-de \cos \omega t + f \sin \omega t + de) \\
&+ s\lambda_i C_i(-df \cos \omega t - e \sin \omega t + df) \\
&+ (-v^2 s\lambda'A' + des\lambda'B' + dfs\lambda'C') \cos \omega t \\
&+ (-fs\lambda'B' + es\lambda'C') \sin \omega t - d^2 s\lambda'A' - des\lambda'B' \\
&- dfs\lambda'C' + s\lambda'A', \\
Y_0 &+ s\lambda_i A_i(-de \cos \omega t - f \sin \omega t + de) \\
&+ s\lambda_i B_i(d^2e^2/v^2 \cos \omega t + f^2/v^2 \cos \omega t + e^2) \\
&+ s\lambda_i C_i(d^2ef/v^2 \cos \omega t + de^2/v^2 \sin \omega t \\
&+ df^2/v^2 \sin \omega t - ef/v^2 \cos \omega t + ef) \\
&+ (des\lambda'A' - d^2e^2s\lambda'B'/v^2 - d^2efs\lambda'C'/v^2) \cos \omega t \\
&+ (-de^2s\lambda'C'/v^2 + fs\lambda'A' - df^2s\lambda'C'/v^2) \sin \omega t \\
&- (f^2s\lambda'B'/v^2 - efs\lambda'C'/v^2) \cos \omega t - des\lambda'A' \\
&- e^2s\lambda'B' - efs\lambda'C' + s\lambda'B', \\
Z_0 &+ s\lambda_i A_i(-df \cos \omega t + e \sin \omega t + df) \\
&+ s\lambda_i B_i(d^2ef/v^2 \cos \omega t - df^2/v^2 \sin \omega t \\
&- de^2/v^2 \sin \omega t - ef/v^2 \cos \omega t + ef) \\
&+ s\lambda_i C_i(d^2f^2/v^2 \cos \omega t + e^2/v^2 \cos \omega t + f^2) \\
&- (-dfs\lambda'A' + d^2efs\lambda'B' + d^2f^2s\lambda'C'/v^2) \cos \omega t
\end{aligned}
$$

$$- (-df^2 s\lambda'B'/v^2 + es\lambda'A' - defs\lambda'C'/v^2) \sin \omega t$$

$$+ (efs\lambda'B'/v^2 - e^2 s\lambda'C'/v^2) \cos \omega t - dfs\lambda'A'$$

$$- efs\lambda'B' - f^2 s\lambda'C' + s\lambda'C',$$

1)

$$= (X_0 + sA'', Y_0\ sB'', Z_0 + sC'', 1).$$

The image of the rotated point by equations (+) is

$$x'_{v_{si}} = F \frac{sa_{11}A'' + sa_{12}B'' + sa_{13}C''}{sa_{31}A'' + sa_{32}B'' + sa_{33}C''}$$

$$= F \frac{s[a_{11}A'' + a_{12}B'' + a_{13}C'']}{s[a_{31}A'' + a_{32}B'' + a_{33}C'']}$$

$$= F \frac{a_{11}A'' + a_{12}B'' + a_{13}C''}{a_{31}A'' + a_{32}B'' + a_{33}C''}$$

$$= x'_{v_i}$$

$$y'_{v_{si}} = F \frac{sa_{21}A'' + sa_{12}B'' + sa_{13}C''}{sa_{31}A'' + sa_{32}B'' + sa_{33}C''}$$

$$= F \frac{a_{11}A'' + a_{12}B'' + a_{13}C''}{a_{31}A'' + a_{32}B'' + a_{33}C''}$$

$$= y'_{v_i}$$

Thus we see that the image of the two points is the same. This proves that the image of a rotating object is inherently ambiguous.

APPENDIX 3

Numerical Examples

Example of rotation and translation from "Movement Calculations"

The final rotation matrix multiplications look quite formidable and rather abstract so we shall make it more tangible by giving an example. We have defined a truncated wedge and a velocity of (1,3,-2,0). Now add an axis of rotation with direction (-12/13,3/13,4/13) and let point $P=(X_0, Y_0, Z_0)=(-6,1,3)$ be a point on the axis of rotation. Let the rotational velocity $\omega=.1$ radians/sec. The matrices for rotation are:

$$T_{rotationhelp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 6 & -1 & -3 & 1 \end{bmatrix}$$

$$R_{x(\theta)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 4/5 & 3/5 & 0 \\ 0 & -3/5 & 4/5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{y\phi} = \begin{bmatrix} 5/13 & 0 & 12/13 & 0 \\ 0 & 1 & 0 & 0 \\ -12/13 & 0 & 5/13 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

for t=1 sec,

$$R_{z(\alpha)} = \begin{bmatrix} .995 & -.0998 & 0 & 0 \\ .0998 & .995 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R^{-1}_{y\phi} = \begin{bmatrix} 5/13 & 0 & -12/13 & 0 \\ 0 & 1 & 0 & 0 \\ 12/13 & 0 & 5/13 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{x\theta}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 4/5 & -3/5 & 0 \\ 0 & 3/5 & 4/5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{\text{rotationhelp}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -6 & 1 & 3 & 1 \end{bmatrix}$$

Now we give the computer printout from a program that tells the original position of the object, the rotation and translation information. Thus, a point p is moved to position p' by the calculation

$$p T_{\text{rotationhelp}} R_x R_y R_z R_y^{-1} R_x^{-1} T_{\text{rotationhelp}}^{-1} T_{(1,3,-2)} = p'.$$

```
 ROTATIONAL VELOCITY    .1000000
 POINT (X, Y, Z) ON ROTATIONAL AXIS
  -6.0000000   1.0000000   3.0000000
 DIRECTION COSINES OF ROTATIONAL AXIS
   -.9230769    .2307692    .3076923
 VELOCITY VECTOR COMPONENTS
   1.0000000   3.0000000  -2.0000000
```

INPUT OBJECT COORDINATES:

0.00000	0.00000	-10.00000
0.00000	0.00000	-8.00000
0.00000	3.00000	-9.00000
0.00000	3.00000	-10.00000
-4.00000	0.00000	-10.00000
-4.00000	0.00000	-8.00000
-4.00000	3.00000	-9.00000
-4.00000	3.00000	-10.00000

TIME= 1.00000

3-D COORDS ARE:

1.2838585E+00	4.0074262E+00	-1.1903994E+01
1.2349436E+00	3.8238278E+00	-9.9130400E+00
1.3483624E+00	6.9014376E+00	-1.0630991E+01
1.3728198E+00	6.9932368E+00	-1.1626468E+01
-2.7131854E+00	4.1345549E+00	-1.1990472E+01
-2.7621003E+00	3.9509565E+00	-9.9995181E+00
-2.6486815E+00	7.0285664E+00	-1.0717469E+01
-2.6242241E+00	7.1203655E+00	-1.1712946E+01

Example of projecting points onto a focal plane from "Projection"

Let's project the points that were rotated and moved in the "Movements Calculations" section onto a focal plane. Assume that the center of the lens is at (0,0,-2), that the focal length is 2, and that the direction cosines of the optical axis of the camera are (0,0,-1). The matrices of interest are,

$$T_{(0,0,-1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$R_{x\theta} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$R_{y\phi} = R_{z\kappa} =$ identity matrix

$$T_{FOCAL} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We showed earlier the movement computation needed to derive the three-dimensional position of points. We can determine the projective coordinates by multiplying the object point positions by the projection matrices and dividing by the fourth coordinate to return to cartesian coordinates. The projection q' of point q is determined by the equation

$$qT_{(0,0,-1)}R_x R_y R_z T_{FOCAL}P = q'.$$

The result for our example is:

```
PROJECTIVE COORDS
   2.5926074E-01   -8.0925455E-01
   3.1212875E-01   -9.6646240E-01
   3.1244670E-01   -1.5992225E+00
   2.8521774E-01   -1.4529185E+00
```

```
-5.4315458E-01    -8.2769957E-01
-6.9056666E-01    -9.8779863E-01
-6.0767212E-01    -1.6125245E+00
-5.4035593E-01    -1.4661597E+00
```

The first two coordinates of each line are the screen coordinates of the rotated, translated truncated wedge.

Numerical Example of Ambiguous Images from Translating Objects

Here we give the computer printout for two different translating objects that result in the same image:

```
TIME INTERVAL IS 1.0000000
POSITION OF CAMERA LENS CENTER
  0.0000000  0.0000000 -2.0000000
DIRECTION COSINES OF CAMERA OPTICAL AXIS
  0.0000000  0.0000000 -1.0000000
VELOCITY VECTOR COMPONENTS
  1.0000000  3.0000000 -2.0000000
```

```
INPUT OBJECT COORDINATES:
  0.00000    0.00000 -10.00000
  0.00000    0.00000  -8.00000
  0.00000    3.00000  -9.00000
  0.00000    3.00000 -10.00000
 -4.00000    0.00000 -10.00000
 -4.00000    0.00000  -8.00000
 -4.00000    3.00000  -9.00000
 -4.00000    3.00000 -10.00000

 PROJECTIVE COORDS BEFORE MOVEMENT
    0.0000000E+00     0.0000000E+00
    0.0000000E+00     0.0000000E+00
    0.0000000E+00    -8.5714286E-01
    0.0000000E+00    -7.5000000E-01
   -1.0000000E+00     0.0000000E+00
   -1.3333333E+00     0.0000000E+00
   -1.1428571E+00    -8.5714286E-01
   -1.0000000E+00    -7.5000000E-01

  TIME=  1.00000
  3-D COORDS ARE:
     1.0000000E+00     3.0000000E+00    -1.2000000E+01
     1.0000000E+00     3.0000000E+00    -1.0000000E+01
     1.0000000E+00     6.0000000E+00    -1.1000000E+01
```

```
 1.0000000E+00    6.0000000E+00   -1.2000000E+01
-3.0000000E+00    3.0000000E+00   -1.2000000E+01
-3.0000000E+00    3.0000000E+00   -1.0000000E+01
-3.0000000E+00    6.0000000E+00   -1.1000000E+01
-3.0000000E+00    6.0000000E+00   -1.2000000E+01

 PROJECTIVE COORDS
 2.0000000E-01   -6.0000000E-01
 2.5000000E-01   -7.5000000E-01
 2.2222222E-01   -1.3333333E+00
 2.0000000E-01   -1.2000000E+00
-6.0000000E-01   -6.0000000E-01
-7.5000000E-01   -7.5000000E-01
-6.6666667E-01   -1.3333333E+00
-6.0000000E-01   -1.2000000E+00

TIME=  2.00000
3-D COORDS ARE:
 2.0000000E+00    6.0000000E+00   -1.4000000E+01
 2.0000000E+00    6.0000000E+00   -1.2000000E+01
 2.0000000E+00    9.0000000E+00   -1.3000000E+01
 2.0000000E+00    9.0000000E+00   -1.4000000E+01
-2.0000000E+00    6.0000000E+00   -1.4000000E+01
-2.0000000E+00    6.0000000E+00   -1.2000000E+01
-2.0000000E+00    9.0000000E+00   -1.3000000E+01
```

-2.0000000E+00 9.0000000E+00 -1.4000000E+01

PROJECTIVE COORDS

3.3333333E-01	-1.0000000E+00
4.0000000E-01	-1.2000000E+00
3.6363636E-01	-1.6363636E+00
3.3333333E-01	-1.5000000E+00
-3.3333333E-01	-1.0000000E+00
-4.0000000E-01	-1.2000000E+00
-3.6363636E-01	-1.6363636E+00
-3.3333333E-01	-1.5000000E+00

TIME= 3.00000

3-D COORDS ARE:

3.0000000E+00	9.0000000E+00	-1.6000000E+01
3.0000000E+00	9.0000000E+00	-1.4000000E+01
3.0000000E+00	1.2000000E+01	-1.5000000E+01
3.0000000E+00	1.2000000E+01	-1.6000000E+01
-1.0000000E+00	9.0000000E+00	-1.6000000E+01
-1.0000000E+00	9.0000000E+00	-1.4000000E+01
-1.0000000E+00	1.2000000E+01	-1.5000000E+01
-1.0000000E+00	1.2000000E+01	-1.6000000E+01

PROJECTIVE COORDS

4.2857143E-01	-1.2857143E+00

```
 5.0000000E-01   -1.5000000E+00
 4.6153846E-01   -1.8461538E+00
 4.2857143E-01   -1.7142857E+00
-1.4285714E-01   -1.2857143E+00
-1.6666667E-01   -1.5000000E+00
-1.5384615E-01   -1.8461538E+00
-1.4285714E-01   -1.7142857E+00
```

For the second object:

```
TIME INTERVAL IS 1.0000000
POSITION OF CAMERA LENS CENTER
  0.0000000  0.0000000 -2.0000000
DIRECTION COSINES OF CAMERA OPTICAL AXIS
  0.0000000  0.0000000 -1.0000000
VELOCITY VECTOR COMPONENTS
  2.0000000  6.0000000 -4.0000000

INPUT OBJECT COORDINATES:
  0.00000   0.00000 -18.00000
  0.00000   0.00000 -14.00000
  0.00000   6.00000 -16.00000
  0.00000   6.00000 -18.00000
 -8.00000   0.00000 -18.00000
 -8.00000   0.00000 -14.00000
```

```
 -8.00000    6.00000 -16.00000
 -8.00000    6.00000 -18.00000

PROJECTIVE COORDS BEFORE MOVEMENT
   0.0000000E+00    0.0000000E+00
   0.0000000E+00    0.0000000E+00
   0.0000000E+00   -8.5714286E-01
   0.0000000E+00   -7.5000000E-01
  -1.0000000E+00    0.0000000E+00
  -1.3333333E+00    0.0000000E+00
  -1.1428571E+00   -8.5714286E-01
  -1.0000000E+00   -7.5000000E-01

TIME=  1.00000
3-D COORDS ARE:
   2.0000000E+00    6.0000000E+00   -2.2000000E+01
   2.0000000E+00    6.0000000E+00   -1.8000000E+01
   2.0000000E+00    1.2000000E+01   -2.0000000E+01
   2.0000000E+00    1.2000000E+01   -2.2000000E+01
  -6.0000000E+00    6.0000000E+00   -2.2000000E+01
  -6.0000000E+00    6.0000000E+00   -1.8000000E+01
  -6.0000000E+00    1.2000000E+01   -2.0000000E+01
  -6.0000000E+00    1.2000000E+01   -2.2000000E+01

PROJECTIVE COORDS
```

2.0000000E-01	-6.0000000E-01
2.5000000E-01	-7.5000000E-01
2.2222222E-01	-1.3333333E+00
2.0000000E-01	-1.2000000E+00
-6.0000000E-01	-6.0000000E-01
-7.5000000E-01	-7.5000000E-01
-6.6666667E-01	-1.3333333E+00
-6.0000000E-01	-1.2000000E+00

TIME= 2.00000

3-D COORDS ARE:

4.0000000E+00	1.2000000E+01	-2.6000000E+01
4.0000000E+00	1.2000000E+01	-2.2000000E+01
4.0000000E+00	1.8000000E+01	-2.4000000E+01
4.0000000E+00	1.8000000E+01	-2.6000000E+01
-4.0000000E+00	1.2000000E+01	-2.6000000E+01
-4.0000000E+00	1.2000000E+01	-2.2000000E+01
-4.0000000E+00	1.8000000E+01	-2.4000000E+01
-4.0000000E+00	1.8000000E+01	-2.6000000E+01

PROJECTIVE COORDS

3.3333333E-01	-1.0000000E+00
4.0000000E-01	-1.2000000E+00
3.6363636E-01	-1.6363636E+00
3.3333333E-01	-1.5000000E+00

```
 -3.3333333E-01   -1.0000000E+00
 -4.0000000E-01   -1.2000000E+00
 -3.6363636E-01   -1.6363636E+00
 -3.3333333E-01   -1.5000000E+00

TIME=  3.00000
3-D COORDS ARE:
   6.0000000E+00    1.8000000E+01   -3.0000000E+01
   6.0000000E+00    1.8000000E+01   -2.6000000E+01
   6.0000000E+00    2.4000000E+01   -2.8000000E+01
   6.0000000E+00    2.4000000E+01   -3.0000000E+01
  -2.0000000E+00    1.8000000E+01   -3.0000000E+01
  -2.0000000E+00    1.8000000E+01   -2.6000000E+01
  -2.0000000E+00    2.4000000E+01   -2.8000000E+01
  -2.0000000E+00    2.4000000E+01   -3.0000000E+01

 PROJECTIVE COORDS
    4.2857143E-01   -1.2857143E+00
    5.0000000E-01   -1.5000000E+00
    4.6153846E-01   -1.8461538E+00
    4.2857143E-01   -1.7142857E+00
   -1.4285714E-01   -1.2857143E+00
   -1.6666667E-01   -1.5000000E+00
   -1.5384615E-01   -1.8461538E+00
   -1.4285714E-01   -1.7142857E+00
```

Numerical Example of Ambiguous Images from Rotating Objects

Here are the computer printouts for different rotating objects that have the same images:

```
TIME INTERVAL IS 1.0000000
POSITION OF CAMERA LENS CENTER
  0.0000000  0.0000000 -2.0000000
DIRECTION COSINES OF CAMERA OPTICAL AXIS
  0.0000000  0.0000000 -1.0000000
ROTATIONAL VELOCITY    .1000000
POINT (X, Y, Z) ON ROTATIONAL AXIS
 -6.0000000  1.0000000  3.0000000
DIRECTION COSINES OF ROTATIONAL AXIS
  -.9230769    .2307692    .3076923
VELOCITY VECTOR COMPONENTS
  0.0000000  0.0000000  0.0000000

INPUT OBJECT COORDINATES:
  0.00000    0.00000 -10.00000
  0.00000    0.00000  -8.00000
  0.00000    3.00000  -9.00000
  0.00000    3.00000 -10.00000
 -4.00000    0.00000 -10.00000
 -4.00000    0.00000  -8.00000
 -4.00000    3.00000  -9.00000
```

```
-4.00000    3.00000 -10.00000

PROJECTIVE COORDS BEFORE MOVEMENT
   0.0000000E+00    0.0000000E+00
   0.0000000E+00    0.0000000E+00
   0.0000000E+00   -8.5714286E-01
   0.0000000E+00   -7.5000000E-01
  -1.0000000E+00    0.0000000E+00
  -1.3333333E+00    0.0000000E+00
  -1.1428571E+00   -8.5714286E-01
  -1.0000000E+00   -7.5000000E-01

 TIME=  1.00000
 3-D COORDS ARE:
    2.8385846E-01    1.0074262E+00   -9.9039943E+00
    2.3494363E-01    8.2382781E-01   -7.9130400E+00
    3.4836237E-01    3.9014376E+00   -8.6309911E+00
    3.7281978E-01    3.9932368E+00   -9.6264683E+00
   -3.7131854E+00    1.1345549E+00   -9.9904724E+00
   -3.7621003E+00    9.5095651E-01   -7.9995181E+00
   -3.6486815E+00    4.0285664E+00   -8.7174693E+00
   -3.6242241E+00    4.1203655E+00   -9.7129465E+00

  PROJECTIVE COORDS
    7.1826586E-02   -2.5491572E-01
```

```
   7.9466275E-02   -2.7864781E-01
   1.0507098E-01   -1.1767284E+00
   9.7769968E-02   -1.0472047E+00
  -9.2940322E-01   -2.8397692E-01
  -1.2541341E+00   -3.1701096E-01
  -1.0863262E+00   -1.1994298E+00
  -9.3977681E-01   -1.0684284E+00

 TIME=  2.00000
 3-D COORDS ARE:
   5.9503308E-01    1.9922526E+00   -9.7090902E+00
   4.9201642E-01    1.6283092E+00   -7.7451827E+00
   7.1417331E-01    4.7536653E+00   -8.1727290E+00
   7.6568164E-01    4.9356369E+00   -9.1546828E+00
  -3.3931720E+00    2.2537534E+00   -9.8698310E+00
  -3.4961887E+00    1.8898100E+00   -7.9059235E+00
  -3.2740318E+00    5.0151661E+00   -8.3334698E+00
  -3.2225234E+00    5.1971377E+00   -9.3154236E+00

PROJECTIVE COORDS
   1.5437180E-01   -5.1685803E-01
   1.7127964E-01   -5.6684332E-01
   2.3139629E-01   -1.5402151E+00
   2.1403650E-01   -1.3796941E+00
  -8.6232398E-01   -5.7275776E-01
```

-1.1839600E+00	-6.3997104E-01
-1.0338825E+00	-1.5837025E+00
-8.8102169E-01	-1.4208713E+00

TIME= 3.00000

3-D COORDS ARE:

9.3041470E-01	2.9446391E+00	-9.4172352E+00
7.6864977E-01	2.4054061E+00	-7.4981053E+00
1.0937778E+00	5.5481677E+00	-7.6297925E+00
1.1746602E+00	5.8177842E+00	-8.5893575E+00
-3.0431572E+00	3.3464128E+00	-9.6392811E+00
-3.2049221E+00	2.8071798E+00	-7.7201512E+00
-2.8797941E+00	5.9499413E+00	-7.8518384E+00
-2.7989117E+00	6.2195578E+00	-8.8114034E+00

PROJECTIVE COORDS

2.5087911E-01	-7.9399912E-01
2.7960533E-01	-8.7499456E-01
3.8856770E-01	-1.9710026E+00
3.5653255E-01	-1.7658123E+00
-7.9671297E-01	-8.7610672E-01
-1.1205725E+00	-9.8150544E-01
-9.8423570E-01	-2.0335289E+00
-8.2183113E-01	-1.8262192E+00

For the second object:

```
 TIME INTERVAL IS 1.0000000
 POSITION OF CAMERA LENS CENTER
   0.0000000  0.0000000 -2.0000000
 DIRECTION COSINES OF CAMERA OPTICAL AXIS
   0.0000000  0.0000000 -1.0000000
 ROTATIONAL VELOCITY    .1000000
 POINT (X, Y, Z) ON ROTATIONAL AXIS
-12.0000000  2.0000000  8.0000000
 DIRECTION COSINES OF ROTATIONAL AXIS
  -.9230769    .2307692    .3076923
 VELOCITY VECTOR COMPONENTS
   0.0000000  0.0000000  0.0000000

INPUT OBJECT COORDINATES:
   0.00000   0.00000 -18.00000
   0.00000   0.00000 -14.00000
   0.00000   6.00000 -16.00000
   0.00000   6.00000 -18.00000
  -8.00000   0.00000 -18.00000
  -8.00000   0.00000 -14.00000
  -8.00000   6.00000 -16.00000
  -8.00000   6.00000 -18.00000
```

```
PROJECTIVE COORDS BEFORE MOVEMENT
   0.0000000E+00    0.0000000E+00
   0.0000000E+00    0.0000000E+00
   0.0000000E+00   -8.5714286E-01
   0.0000000E+00   -7.5000000E-01
  -1.0000000E+00    0.0000000E+00
  -1.3333333E+00    0.0000000E+00
  -1.1428571E+00   -8.5714286E-01
  -1.0000000E+00   -7.5000000E-01

 TIME=  1.00000
 3-D COORDS ARE:
    5.6771693E-01    2.0148524E+00   -1.7807989E+01
    4.6988726E-01    1.6476556E+00   -1.3826080E+01
    6.9672473E-01    7.8028753E+00   -1.5261982E+01
    7.4563956E-01    7.9864737E+00   -1.7252937E+01
   -7.4263708E+00    2.2691098E+00   -1.7980945E+01
   -7.5242005E+00    1.9019130E+00   -1.3999036E+01
   -7.2973630E+00    8.0571327E+00   -1.5434939E+01
   -7.2484482E+00    8.2407311E+00   -1.7425893E+01

  PROJECTIVE COORDS
    7.1826586E-02   -2.5491572E-01
    7.9466275E-02   -2.7864781E-01
    1.0507098E-01   -1.1767284E+00
```

```
   9.7769968E-02   -1.0472047E+00
  -9.2940322E-01   -2.8397692E-01
  -1.2541341E+00   -3.1701096E-01
  -1.0863262E+00   -1.1994298E+00
  -9.3977681E-01   -1.0684284E+00

TIME=  2.00000
3-D COORDS ARE:
   1.1900662E+00    3.9845052E+00   -1.7418180E+01
   9.8403283E-01    3.2566184E+00   -1.3490365E+01
   1.4283466E+00    9.5073305E+00   -1.4345458E+01
   1.5313633E+00    9.8712739E+00   -1.6309366E+01
  -6.7863440E+00    4.5075067E+00   -1.7739662E+01
  -6.9923773E+00    3.7796200E+00   -1.3811847E+01
  -6.5480635E+00    1.0030332E+01   -1.4666940E+01
  -6.4450469E+00    1.0394275E+01   -1.6630847E+01

PROJECTIVE COORDS
   1.5437180E-01   -5.1685803E-01
   1.7127964E-01   -5.6684332E-01
   2.3139629E-01   -1.5402151E+00
   2.1403650E-01   -1.3796941E+00
  -8.6232398E-01   -5.7275776E-01
  -1.1839600E+00   -6.3997104E-01
  -1.0338825E+00   -1.5837025E+00
```

-8.8102169E-01 -1.4208713E+00

TIME= 3.00000

3-D COORDS ARE:

1.8608294E+00	5.8892782E+00	-1.6834470E+01
1.5372995E+00	4.8108122E+00	-1.2996211E+01
2.1875555E+00	1.1096335E+01	-1.3259585E+01
2.3493204E+00	1.1635568E+01	-1.5178715E+01
-6.0863144E+00	6.6928255E+00	-1.7278562E+01
-6.4098442E+00	5.6143595E+00	-1.3440302E+01
-5.7595883E+00	1.1899883E+01	-1.3703677E+01
-5.5978233E+00	1.2439116E+01	-1.5622807E+01

PROJECTIVE COORDS

2.5087911E-01	-7.9399912E-01
2.7960533E-01	-8.7499456E-01
3.8856770E-01	-1.9710026E+00
3.5653255E-01	-1.7658123E+00
-7.9671297E-01	-8.7610672E-01
-1.1205725E+00	-9.8150544E-01
-9.8423570E-01	-2.0335289E+00
-8.2183113E-01	-1.8262192E+00

Numerical Example of Ambiguous Images from Rotating and Translating Objects

Here we give the computer printouts for two different objects that are both rotating and translating yet producing the same sequence of images. This sequence of images for times one through five is illustrated in the computer plots in figure one of the introduction.

```
 TIME INTERVAL IS 1.0000000
 POSITION OF CAMERA LENS CENTER
   0.0000000  0.0000000 -2.0000000
 DIRECTION COSINES OF CAMERA OPTICAL AXIS
   0.0000000  0.0000000 -1.0000000
 ROTATIONAL VELOCITY    .1000000
 POINT (X, Y, Z) ON ROTATIONAL AXIS
  -6.0000000  1.0000000  3.0000000
 DIRECTION COSINES OF ROTATIONAL AXIS
   -.9230769    .2307692    .3076923
 VELOCITY VECTOR COMPONENTS
   1.0000000  3.0000000 -2.0000000


 INPUT OBJECT COORDINATES:
   0.00000    0.00000 -10.00000
   0.00000    0.00000  -8.00000
   0.00000    3.00000  -9.00000
   0.00000    3.00000 -10.00000
```

```
-4.00000    0.00000 -10.00000
-4.00000    0.00000  -8.00000
-4.00000    3.00000  -9.00000
-4.00000    3.00000 -10.00000

PROJECTIVE COORDS BEFORE MOVEMENT
   0.0000000E+00    0.0000000E+00
   0.0000000E+00    0.0000000E+00
   0.0000000E+00   -8.5714286E-01
   0.0000000E+00   -7.5000000E-01
  -1.0000000E+00    0.0000000E+00
  -1.3333333E+00    0.0000000E+00
  -1.1428571E+00   -8.5714286E-01
  -1.0000000E+00   -7.5000000E-01

TIME=  1.00000
3-D COORDS ARE:
    1.2838585E+00    4.0074262E+00   -1.1903994E+01
    1.2349436E+00    3.8238278E+00   -9.9130400E+00
    1.3483624E+00    6.9014376E+00   -1.0630991E+01
    1.3728198E+00    6.9932368E+00   -1.1626468E+01
   -2.7131854E+00    4.1345549E+00   -1.1990472E+01
   -2.7621003E+00    3.9509565E+00   -9.9995181E+00
   -2.6486815E+00    7.0285664E+00   -1.0717469E+01
   -2.6242241E+00    7.1203655E+00   -1.1712946E+01
```

PROJECTIVE COORDS

2.5926074E-01	-8.0925455E-01
3.1212875E-01	-9.6646240E-01
3.1244670E-01	-1.5992225E+00
2.8521774E-01	-1.4529185E+00
-5.4315458E-01	-8.2769957E-01
-6.9056666E-01	-9.8779863E-01
-6.0767212E-01	-1.6125245E+00
-5.4035593E-01	-1.4661597E+00

TIME= 2.00000

3-D COORDS ARE:

2.5950331E+00	7.9922526E+00	-1.3709090E+01
2.4920164E+00	7.6283092E+00	-1.1745183E+01
2.7141733E+00	1.0753665E+01	-1.2172729E+01
2.7656816E+00	1.0935637E+01	-1.3154683E+01
-1.3931720E+00	8.2537534E+00	-1.3869831E+01
-1.4961887E+00	7.8898100E+00	-1.1905923E+01
-1.2740318E+00	1.1015166E+01	-1.2333470E+01
-1.2225234E+00	1.1197138E+01	-1.3315424E+01

PROJECTIVE COORDS

4.4325102E-01	-1.3651364E+00
5.1143555E-01	-1.5655549E+00

5.3361754E-01	-2.1142144E+00
4.9587813E-01	-1.9607258E+00
-2.3474167E-01	-1.3907112E+00
-3.0207959E-01	-1.5929479E+00
-2.4658354E-01	-2.1319395E+00
-2.1608090E-01	-1.9790930E+00

TIME= 3.00000

3-D COORDS ARE:

3.9304147E+00	1.1944639E+01	-1.5417235E+01
3.7686498E+00	1.1405406E+01	-1.3498105E+01
4.0937778E+00	1.4548168E+01	-1.3629792E+01
4.1746602E+00	1.4817784E+01	-1.4589357E+01
-4.3157185E-02	1.2346413E+01	-1.5639281E+01
-2.0492212E-01	1.1807180E+01	-1.3720151E+01
1.2020587E-01	1.4949941E+01	-1.3851838E+01
2.0108834E-01	1.5219558E+01	-1.4811403E+01

PROJECTIVE COORDS

5.8587550E-01	-1.7804919E+00
6.5552536E-01	-1.9838758E+00
7.0401562E-01	-2.5018792E+00
6.6320465E-01	-2.3540175E+00
-6.3283666E-03	-1.8104199E+00
-3.4969193E-02	-2.0148511E+00

```
    2.0284764E-02    -2.5228055E+00
    3.1392086E-02    -2.3759392E+00
```

For the second object:

```
 TIME INTERVAL IS 1.0000000
 POSITION OF CAMERA LENS CENTER
   0.0000000  0.0000000 -2.0000000
 DIRECTION COSINES OF CAMERA OPTICAL AXIS
   0.0000000  0.0000000 -1.0000000
 ROTATIONAL VELOCITY    .1000000
 POINT (X, Y, Z) ON ROTATIONAL AXIS
-12.0000000  2.0000000  8.0000000
 DIRECTION COSINES OF ROTATIONAL AXIS
   -.9230769    .2307692    .3076923
 VELOCITY VECTOR COMPONENTS
   2.0000000  6.0000000 -4.0000000

 INPUT OBJECT COORDINATES:
   0.00000    0.00000 -18.00000
   0.00000    0.00000 -14.00000
   0.00000    6.00000 -16.00000
   0.00000    6.00000 -18.00000
  -8.00000    0.00000 -18.00000
  -8.00000    0.00000 -14.00000
```

```
-8.00000    6.00000 -16.00000
-8.00000    6.00000 -18.00000

PROJECTIVE COORDS BEFORE MOVEMENT
   0.0000000E+00    0.0000000E+00
   0.0000000E+00    0.0000000E+00
   0.0000000E+00   -8.5714286E-01
   0.0000000E+00   -7.5000000E-01
  -1.0000000E+00    0.0000000E+00
  -1.3333333E+00    0.0000000E+00
  -1.1428571E+00   -8.5714286E-01
  -1.0000000E+00   -7.5000000E-01

TIME=  1.00000
3-D COORDS ARE:
   2.5677169E+00    8.0148524E+00   -2.1807989E+01
   2.4698873E+00    7.6476556E+00   -1.7826080E+01
   2.6967247E+00    1.3802875E+01   -1.9261982E+01
   2.7456396E+00    1.3986474E+01   -2.1252937E+01
  -5.4263708E+00    8.2691098E+00   -2.1980945E+01
  -5.5242005E+00    7.9019130E+00   -1.7999036E+01
  -5.2973630E+00    1.4057133E+01   -1.9434939E+01
  -5.2484482E+00    1.4240731E+01   -2.1425893E+01

PROJECTIVE COORDS
```

2.5926074E-01	-8.0925455E-01
3.1212875E-01	-9.6646240E-01
3.1244670E-01	-1.5992225E+00
2.8521774E-01	-1.4529185E+00
-5.4315458E-01	-8.2769957E-01
-6.9056666E-01	-9.8779863E-01
-6.0767212E-01	-1.6125245E+00
-5.4035593E-01	-1.4661597E+00

TIME= 2.00000

3-D COORDS ARE:

5.1900662E+00	1.5984505E+01	-2.5418180E+01
4.9840328E+00	1.5256618E+01	-2.1490365E+01
5.4283466E+00	2.1507331E+01	-2.2345458E+01
5.5313633E+00	2.1871274E+01	-2.4309366E+01
-2.7863440E+00	1.6507507E+01	-2.5739662E+01
-2.9923773E+00	1.5779620E+01	-2.1811847E+01
-2.5480635E+00	2.2030332E+01	-2.2666940E+01
-2.4450469E+00	2.2394275E+01	-2.4630847E+01

PROJECTIVE COORDS

4.4325102E-01	-1.3651364E+00
5.1143555E-01	-1.5655549E+00
5.3361754E-01	-2.1142144E+00
4.9587813E-01	-1.9607258E+00

```
  -2.3474167E-01   -1.3907112E+00
  -3.0207959E-01   -1.5929479E+00
  -2.4658354E-01   -2.1319395E+00
  -2.1608090E-01   -1.9790930E+00

TIME=  3.00000
3-D COORDS ARE:
   7.8608294E+00    2.3889278E+01   -2.8834470E+01
   7.5372995E+00    2.2810812E+01   -2.4996211E+01
   8.1875555E+00    2.9096335E+01   -2.5259585E+01
   8.3493204E+00    2.9635568E+01   -2.7178715E+01
  -8.6314371E-02    2.4692826E+01   -2.9278562E+01
  -4.0984423E-01    2.3614360E+01   -2.5440302E+01
   2.4041174E-01    2.9899883E+01   -2.5703677E+01
   4.0217667E-01    3.0439116E+01   -2.7622807E+01

PROJECTIVE COORDS
   5.8587550E-01   -1.7804919E+00
   6.5552536E-01   -1.9838758E+00
   7.0401562E-01   -2.5018792E+00
   6.6320465E-01   -2.3540175E+00
  -6.3283666E-03   -1.8104199E+00
  -3.4969193E-02   -2.0148511E+00
   2.0284764E-02   -2.5228055E+00
   3.1392086E-02   -2.3759392E+00
```

APPENDIX 4

Statistical Results

Here we present graphical evidence that considerable over-determination improves the computation of both the model of the object and the camera position.

The three graphs show three experiments, labelled A, B, C, run with two views of a varying number of points. Each experiment was run using a different object that had its own movement. The Z-distance to the reference point in each example was set to 10. The typical distance to the reference point, therefore, was on the order of 15 or 20 units. Three sets of statistics were kept for each experiment: average distance from computed model points to correct model points, distance from the computed camera position to the correct camera position, and average angular error of θ, ϕ, and κ.

Each graph shows an average value for each of the three experiments. The "A" token denotes the value for the A experiment, etc. In addition, to indicate the trend of the data, the average of the three experiments is connected by a solid line in the graphs.

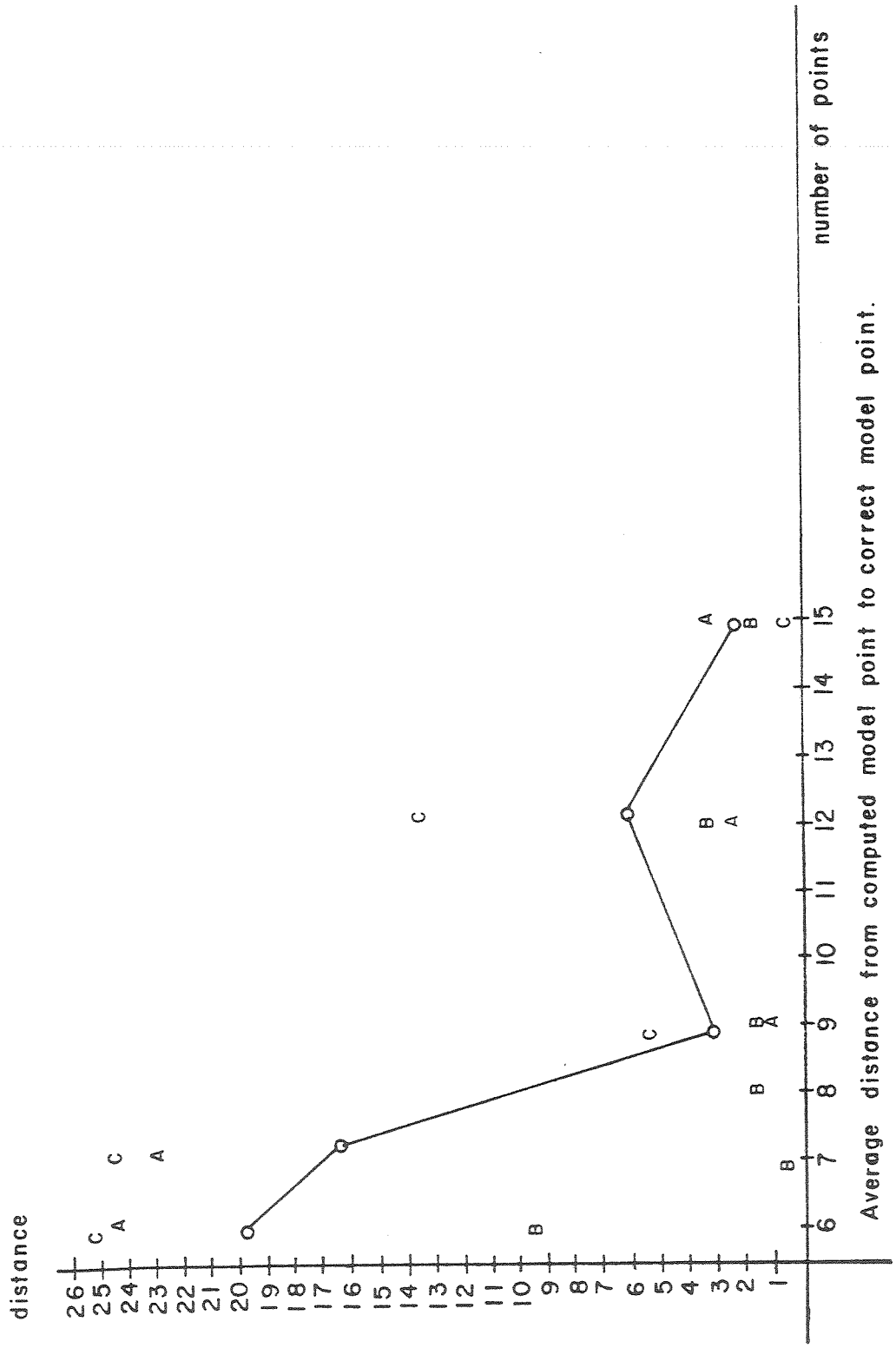
The first graph shows the average distance between the model points for the computed answer given noisy data and the correct answer assuming no noise. For the 6, 7, or 8 point cases, there is one point whose computed coordinates are extremely poor. Note that there is no average distance for the A or C experiments with eight points since their averages were too large to fit on the graph. In general, two views of fewer than nine points result in a numerically unstable model of an object.

The second graph shows the improvement in the second camera position as the number of points increases. The improvement does not appear to be very great mainly due to the unusually good camera position for the A experiment when six or seven points were used.

The third graph shows the average error for the angles θ, ϕ, and κ. There does not seem to be any improvement here as the number of points increases. The average error is about .1 radians.

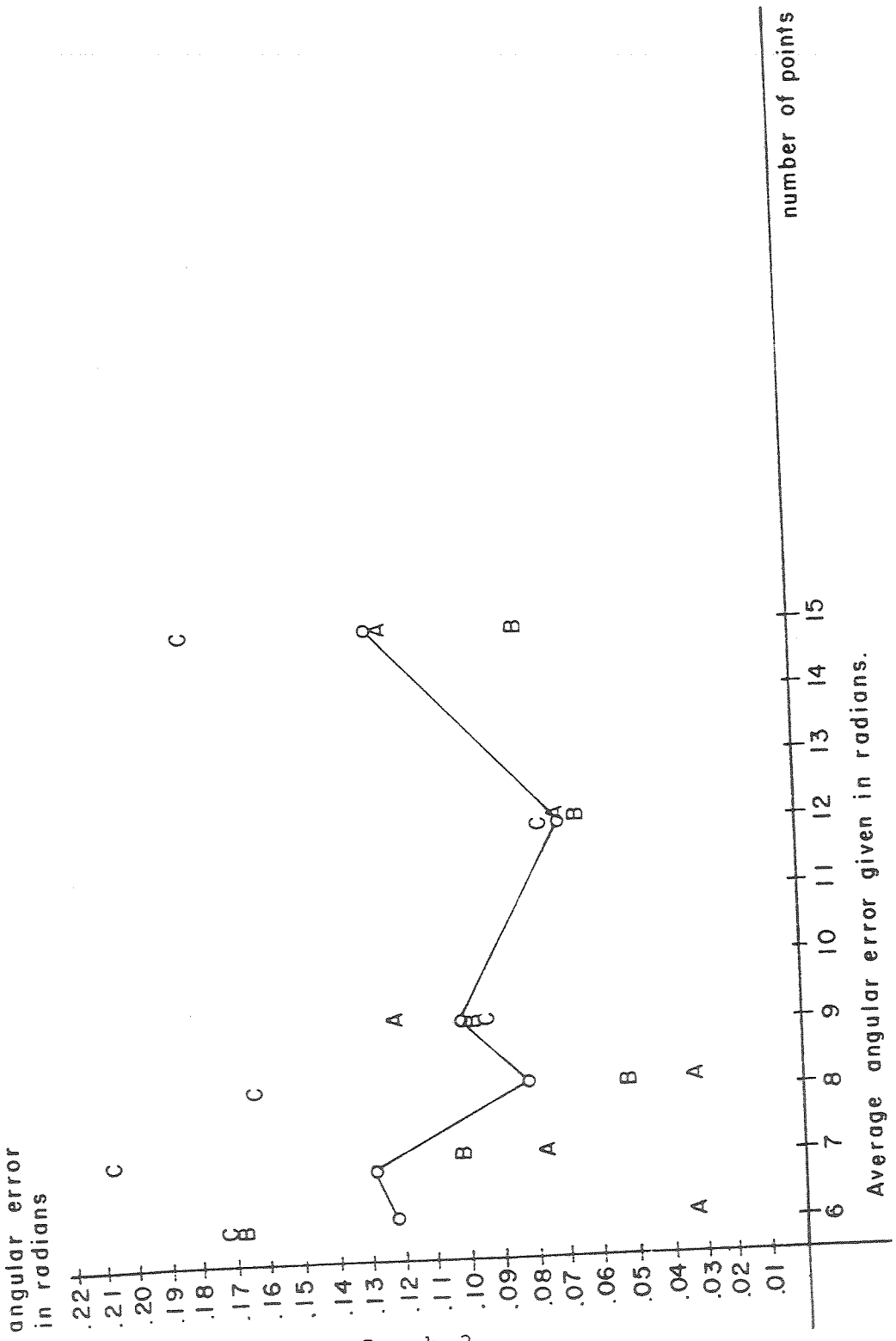
In conclusion, the model of the object showed a marked improvement as the number of points increased, the camera position showed a modest improvement, and the angular orientation of the camera showed little or no improvement.

Experiments with three views showed the model of the object to be fairly good with minimal

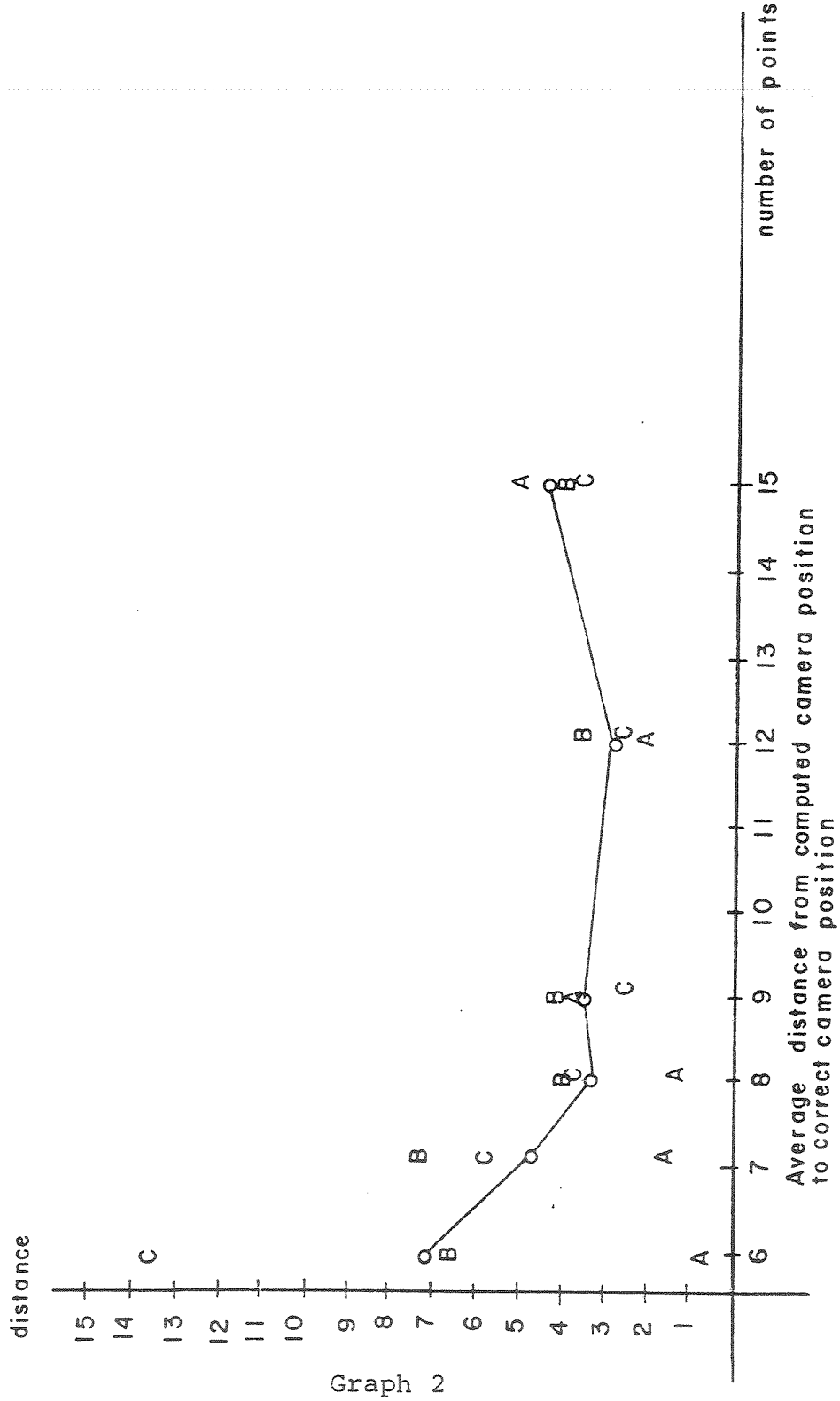

distance
number of points
Average distance from computed model point to correct model point.

Graph 1

over-determination, but the camera positions were poor until there were three views of at least seven points.


angular error in radians
.22
.21
.20
.19
.18
.17
.16
.15
.14
.13
.12
.11
.10
.09
.08
.07
.06
.05
.04
.03
.02
.01
6
7
8
9
10
11
12
13
14
15
number of points
Average angular error given in radians.

Graph 3


distance
number of points
Average distance from computed camera position to correct camera position

Graph 2

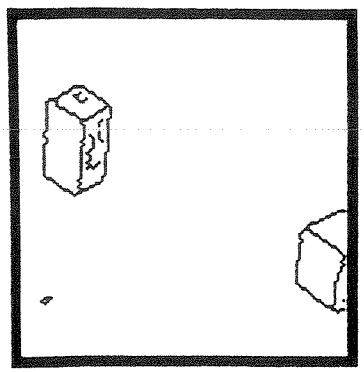


Image 3

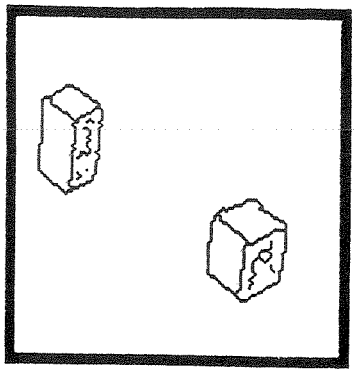


Image 5

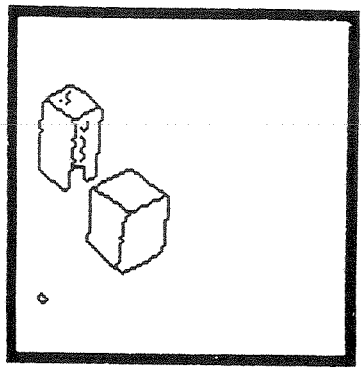


Image 8

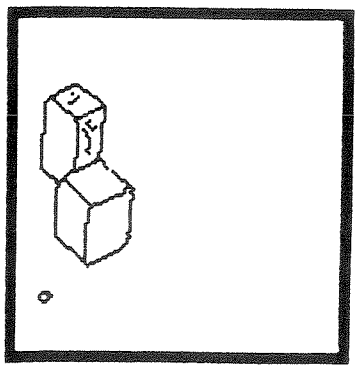


Image 9

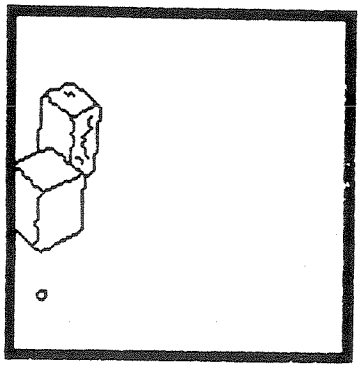


Image 10

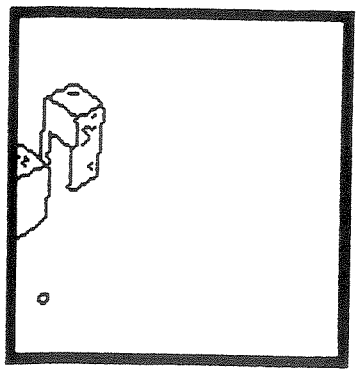


Image 11

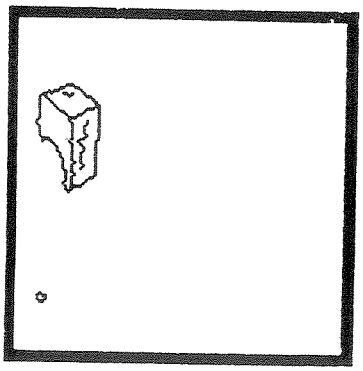


Image 12

Figure 10
First Sequence

APPENDIX 5

Images from Roach and Aggarwal [22]

In this appendix, two sequences of images from Roach and Aggarwal [22] are presented that were used to test an hierarchical object tracking system. Later, images of points on the objects (the corners in fact) were used to find the model and movement of the objects. There were, however, too few points available to attain any accuracy.

The first sequence is given in figure ten and the second sequence is given in figure eleven. In figure ten, not all the images at the first of the sequence are given since they look rather similar.

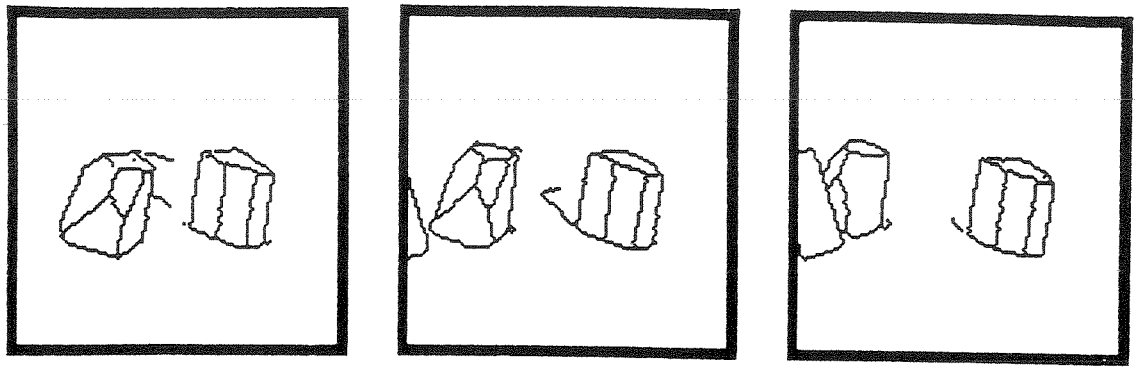


Image 1 Image 2 Image 3

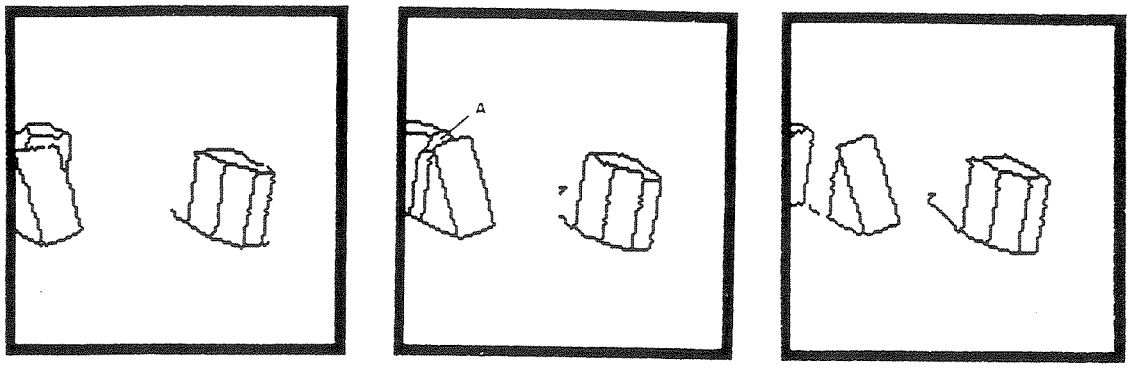


Image 4 Image 5 Image 6

Figure 11
Second Sequence

Bibliography

[1] J. K. Aggarwal and R. O. Duda, "Computer analysis of moving polygonal images," IEEE Transactions on Computers, vol. C-24, no. 10, October, 1975, pp. 966-976.

[2] N. Badler, "Temporal scene analysis: Conceptual descriptions of object movements," Ph. D. dissertation, University of Toronto, TR80, 1975.

[3] M. L. Braunstein, Depth Perception Through Motion, New York: Academic Press, c. 1976.

[4] K. M. Brown and J. E. Dennis, "Derivative free analogues of the Levenberg-Marquardt and Gauss algorithms for nonlinear least squares," Numerische Mathematik, 18, 1972, 289-297.

[5] W. K. Chow and J. K. Aggarwal, "Computer analysis of planar curvilinear moving images," IEEE Transactions on Computers, vol. C-26, No. 2, Feb. 1977, pp. 179-185.

[6] J. Coffin, Vector Analysis, New York: Wiley and Sons, second edition, c. 1911.

[7] R. O. Duda and P. Hart, Pattern Classification and Scene Analysis, New York: Wiley Interscience, c. 1973.

[8] R. M. Endlich, D. E. Wolf, D. J. Hall, and A. E. Brain, "Use of a pattern recognition technique for determining cloud motions from sequences of satellite photographs," J. of Applied Meteorology, vol. 10, Feb. 1971, pp. 105-117.

[9] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," Computer Journal, vol. 6, 1963, 163-168.

[10] J. J. Gibson, The Perception of the Visual World, Boston: Houghton Mifflin, c. 1950.

[11] R. Jain, W. Martin, and J. K. Aggarwal, "Segmentation through the detection of changes due to motion," Computer Graphics and Image Processing, vol. 11, 1979, pp. 13-34.

[12] R. Jain and H. -H. Nagel, "On the analysis of accumulative difference pictures from image sequences of real world scenes," IEEE PAMI, vol. PAMI-1, no. 2, April, 1979, pp. 206-213.

[13] J. A. Leese, C. S. Novak, V. R. Taylor, "An automated technique for obtaining cloud motion from geosynchronous satellite data using cross-correlation," J. of Applied Meteorology, vol. 10, Feb., 1971, pp. 118-132.

[14] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," Quart. Appl.

Math., 2, 1944, pp. 164-168.

[15] P. H. Lindsay and D. A. Norman, Human Information Processing, an Introduction to Psychology, New York: Academic Press, second edition, c. 1977.

[16] J. W. McKee and J. K. Aggarwal, "Finding the edges of surfaces of three-dimensional curved objects by computer," Pattern Recognition, vol. 7, 1975, pp. 25-52.

[17] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," J. Siam, 11(2),1963.

[18] W. Martin and J. K. Aggarwal, "Computer analysis of dynamic scenes containing curvilinear figures," Pattern Recognition, vol. 11, 1979, pp. 169-178.

[19] E. M. Mikhail, Observations and Least Squares, New York: IEP, c. 1976.

[20] W. M. Newman and R. F. Sproull, Principles of Interactive Computer Graphics, New York: McGraw-Hill, Inc., c. 1973.

[21] J. L. Potter, "Motion as a cue to segmentation," Milwaukee Symposium on Automatic Control, 1974, pp. 100-104.

[22] J. Roach and J. K. Aggarwal, "Computer tracking of objects moving in space," IEEE PAMI, vol. PAMI-1, no. 2, 1979, 127-135.

[23] L. G. Roberts, "Machine perception of three-dimensional solids," in Aggarwal, Duda, and Rosenfeld, editors, Computer Methods in Image Analysis, N. Y.: IEEE Press, c. 1977, 285-323.

[24] M. M. Thompson, Manual of Photogrammetry, Third edition, Falls Church, Virginia: American Society of Photogrammetry, c. 1966.

[25] S. Ullman, "The Interpretation of Structure from Motion," A. I. Memo 476, M. I. T. Artificial Intelligence Laboratory, Oct. 1976.

[25b]---------, The Interpretation of Visual Motion, Cambridge, MA.: M. I. T. Press, c. 1979.

[26] S. Underwood and C. L. Coates, Jr., "Visual learning from multiple views," IEEE Transactions on Computers, vol. C-24, no.6, June 1975, pp. 651-661.

[27] D. Waltz, "Understanding line drawings of scenes with shadows," in The Psychology of Computer Vision, P. H. Winston, ed., N. Y.: McGraw-Hill, Inc., c. 1975, pp. 19-95.

[28] T. D. Williams, "Region analysis for a moving scene," Workshop on Computer Analysis of Time-Varying Imagery, April, 1979, Philadelphia, Pa.

[29] P. H. Winston, The Psychology of Computer Vision, New York: McGraw-Hill, c. 1975.

[30] ZXSSQ, International Mathematical and Statistics

Libraries, Houston, Texas, Version 7, c. 1979.