Computer Analysis of Moving Polygonal Images

J. K. AGGARWAL, MEMBER, IEEE, AND RICHARD O. DUDA, MEMBER, IEEE

Abstract—**A general mathematical model is developed as an idealization of the problem of determining cloud motions from satellite pictures. The model consists of superimposed planes of rigid moving polygons. The problem is to determine from a sequence of scenes the linear and angular velocities of the figures, and to decompose the scene into its component figures. Study of the model reveals a number of fundamental relations that form the basis for an analysis program. In particular, a systematic anaylsis is given of the topological changes that can occur when overlapping figures move together or apart. A computer program based on these results is described, and experimental results are presented.**

Index Terms—**Change detection, cloud motion, motion, moving images, occlusion, overlapping figures, pattern recognition, polygonal figures, scene analysis, topological changes.**

I. INTRODUCTION

THIS PAPER describes an investigation of a particular problem in the analysis of pictures of moving two-dimensional objects. The general problem of detecting significant changes in a sequence of pictures is a difficult one. Although Ulstad [1] traces its history back to 1920, most of the work in this area is relatively recent, depending as it does on digital image processing. In particular, considerable attention has been devoted to the determination of cloud motions from digitized satellite photographs [2]–[4]. Since that problem served to motivate the current investigation, some of its characteristics will be described briefly.

In a typical satellite photograph the regions of cloud cover are considerably brighter than the background, whether the clouds are low stratus layers or high cirrus layers. Although cloud velocities in any local region are relatively constant, nonuniform patterns of motion show up over larger areas. The cloud velocity fields are closely related to wind velocity fields, and major changes in wind speed and direction occur over large areas, such as the mid-Pacific region. The purpose of deriving cloud motions from satellite photographs is to obtain these wind velocity fields.

The basic technique used in programs that track cloud motions is cross correlation. In some programs the cross correlation is done directly, small local zones being matched from one picture to the next [2], [3]. With other programs the pictures are first clustered to find brightness centers, and these centers are matched from one picture to the next [4]. In either case, good results can be obtained provided that the pictures are properly registered, that clouds are not in a state of rapid formation or dissipation, and that there is only one layer of clouds in any given area.

Unfortunately, it is not at all uncommon for several cloud layers to be present at once, and each layer can be moving with its own speed and in its own direction. This invalidates the use of cross correlation, unless the various layers can somehow be separated before successive images are matched. If the photographs are viewed individually, it is surprisingly difficult to distinguish between the various layers. In particular, the brightness differences between different layers are quite small, and one must discriminate on the basis of rather subtle shape and texture information. Viewed in a time-lapse sequence, however, the various layers are much more clearly distinguishable, the joint motions of clouds in a given layer often being quite striking. This paper is primarily concerned with ways of extracting, from a sequence of pictures, the information implicit in these joint motions.

Both to avoid the many complications introduced by the use of real imagery and to obtain somewhat greater generality, we have investigated an idealized model of this problem. The model, defined precisely in the next section, can be understood intuitively by imagining a black background, overlaid by a number of transparent sheets, upon which opaque white polygons have been drawn. As the sheets are translated (and perhaps rotated), the exposed and hidden parts of the different polygons shift and change, but the figures on any given sheet maintain their shapes and relative positions. The basic problem is to derive a description of the figures on each sheet and the motion of the sheets from a sequence of views of the superimposed sheets.

Clearly, this model is closely related to the way in which animated cartoons are made. Viewed in a time-lapse sequence, the pictures would form animated silhouettes. Thus, the model is relevant to many other problems involving motion in scene analysis besides the problem of tracking cloud motions. Similar models have been used before in some of the scene analysis literature concerning line drawings. Specifically, Potter [5] used a two-layer version of this model in investigating the use of joint motions to segment a scene of moving, nonoccluding objects, and Badler [6] used a spherical projection model in developing a system that produces natural language descriptions of pictures of known, moving three-dimensional objects. Of course, all such models are highly idealized, and only a first approximation to actual imagery. However, even our simplified model poses problems too difficult to solve by

Manuscript received October 20, 1974; revised April 10, 1975. This work was supported in part under NSF Grant GK42790.

J. K. Aggarwal is with the Department of Electrical Engineering, University of Texas, Austin, Tex. 78712.

R. O. Duda is with the Artificial Intelligence Center, Stanford Research Institute, Menlo Park, Calif. 94025.

mathematical analysis. Our purpose is to understand better how these problems can be solved before additional complications are introduced.

II. MATHEMATICAL MODEL

The following mathematical model is a slight generalization of the transparent sheet model just described. It is based upon a collection of planar figures moving in several parallel planes. For simplicity, the shapes of the figures are restricted to polygons, and the polygons are assumed to be rigid. Thus, each figure retains its shape and size during the course of its motion. The polygons are otherwise unrestricted. In particular, they may have one or more holes and need not be convex. The motions of the polygons in a given plane may be correlated or may be independent. That is, the figures may share a joint motion, or each figure may have its own linear and angular velocities. It is envisaged that there will be several planes and several figures moving in each plane. Although the present paper concentrates on independent motions, the programs generated are equally applicable to correlated motions. Fig. 1 shows a simple example in which Plane A contains a single triangle with a rectangular hole and Plane B contains a nonconvex pentagon.

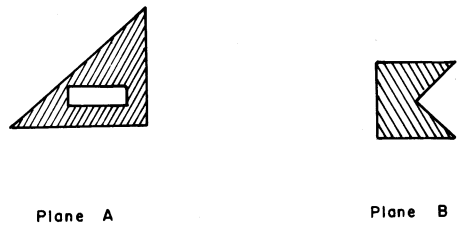


Fig. 1. The individual planes.

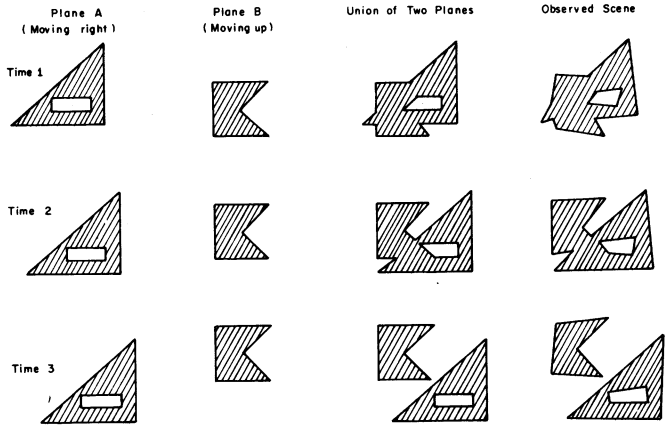


Fig. 2. Motion as seen by observer.

It is assumed that the observer sees a perturbed, parallel projection of the planes, and is unable to distinguish the different planes from a single view. To be more specific, the input picture at any instant in time is taken as the union of the polygons in the various planes, with the coordinates of the vertices perturbed by the addition of random noise. It is further assumed that the time interval between successive pictures is sufficiently small that only "small" changes occur from picture to picture. Fig. 2 illustrates the observer's view of the polygons shown in Fig. 1.

The objective of the present investigation is to determine the linear and angular velocities of the figures, and to decompose the scene into its component figures from a time sequence of scenes as described above. In the general case, both the number of planes and the number of polygons in each plane are unknown to the observer.

The difficulty of this problem in the general case will be apparent from the following sections. Here, some additional comments on the model are presented to show some of the considerations involved in its selection. The restriction of the figures to polygonal shape is motivated by computational convenience. The nature of the study would not be changed significantly by assuming curvilinear boundaries for the figures, but the computational burden would be severely increased. It is believed that the introduction of curved boundaries would bring in extraneous issues without adding significantly to the generality of the problem.

The assumption that the polygons are rigid is also made for simplicity. In a more realistic situation, one will encounter cases where figures appear and disappear, or change in size or shape, or both. By adding random noise to the coordinates, we are in fact allowing small changes in size and shape to occur. In addition, this noise precludes the trivial solutions that can be obtained if the rigid polygons can be observed with unlimited precision. If one assumes that the polygons are obtained by fitting straight lines to gray scale data, it would be more realistic to model uncertainty by randomly inserting or deleting certain vertices. We hope to be able to investigate a more complicated model of this type in the future.

In the present formulation there is no way to tell whether one plane is occluding or is occluded by another. This idealization, which is a good first approximation to the satellite cloud photograph problem, means that no depth clues are available to aid in resolving the figures. Inherent in the model is the assumption that there are no distinguishing features for the various planes. Such features do exist in real applications, and their introduction will certainly make the solution to the problem easier and not more difficult.

III. THE ANALYSIS PROBLEM

The analysis problem is as follows. Given a sequence of scenes generated according to the model, find the linear and angular velocities of the figures and decompose the scene into its component figures. Although the problem as posed is essentially mathematical, our approach to its solution is heuristic and computational. The general problem can be arbitrarily complicated, and it is not sensible to ask for a general solution. However, a reasonably general and interesting computational solution must take into account the problems posed by time sampling, overlapping figures, topological changes, and uncertainty, whether due to quantization or noise, and must solve them

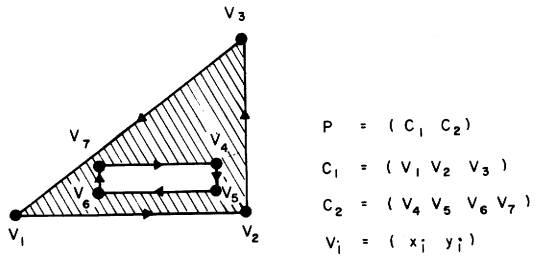


Fig. 3. List representation of a polygon.

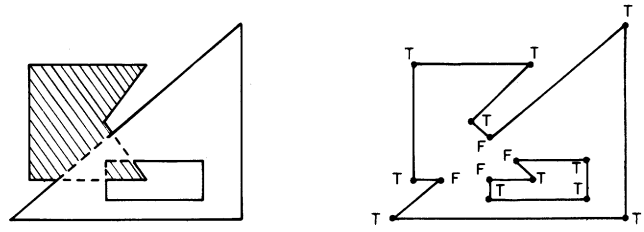


Fig. 4. True and false vertices.

under the usual requirements for economy in the use of time and memory. Some of the various kinds of information that can be exploited by the analysis program are described in this section, starting, for simplicity, with the noise-free case.

A. The Noise-Free Case

In our computer programs a polygon P is represented by a list of boundary components—the external boundary and, if present, the internal or hole boundaries. Each boundary component is a list of vertices appearing in counterclockwise order for the external component and in clockwise order for the internal components (see Fig. 3). With this convention, if one follows the edges joining successive vertices of a boundary component, then the interior of the polygon always lies to the left. Finally, a vertex is described by a pair of coordinates (x,y). From this description, the lengths of the edges, the angles at the vertices, and other properties of the polygon are readily computed.

Let us temporarily assume that the input scene is free of noise and that none of the polygons overlap. Then, since the polygons are assumed to be rigid, the lengths of the edges and the angles at the vertices will remain constant as the polygons move. Given two successive scenes of this type, the only real problem is the identification of corresponding vertices. Once any vertex in the first scene is paired with a vertex in the second scene, all other vertices for that polygon can pair in only one way, and the two polygons either will or will not match. While the matching problem is essentially rather trivial, it raises some problems that also appear in more complex cases. For example, multiple solutions can be encountered when there is symmetry or several identical figures. More important, when there are many polygons to match, the straightforward exhaustive procedures become very inefficient, and the need for good search strategies becomes apparent.

If two polygons overlap, the resulting polygon will contain two kinds of vertices, vertices from the original polygons and vertices where the polygons intersect (see Fig. 4). For brevity, we call the former *true vertices* and the latter *false vertices*. In the absence of noise, true and false vertices can be easily distinguished, since the angles at true vertices stay constant as the polygons move, while the angles and the lengths of the edges incident on false vertices generally change. Of course, if an angle does not change, the vertex is not necessarily true, but if an angle changes then the vertex must be false, and if the length of an edge changes then at least one of the vertices at the ends of the edge must be false.

This suggests a straightforward way of analyzing noise-free pictures of moving rigid polygons, provided that the changes from picture to picture are sufficiently small that no old vertices disappear and that no new vertices are formed. One merely pairs vertices as before, demanding a perfect match for one angle but requiring, say, only a topological match for the rest of the polygon (same number of boundary components and the same number of vertices for each component). If a polygon in the first scene matches more than one polygon in the second scene, a criterion function that measures the differences between corresponding lengths and angles can be used to determine a best-match solution. A variation of this procedure plays an important role in the analysis program described in the next section.

The efficiency and even the accuracy of the search procedure can be increased significantly by exploiting additional facts about the problem. For example, if a polygon has been successfully tracked from one scene to the next, one can easily estimate its linear and its angular velocities, and thus predict its location and orientation in the next scene. Efficiency can be increased by searching for matches only in the vicinity of the expected vertex locations; accuracy can be increased by including positional errors in computation of the matching criterion function. For another example, if it is expected that several figures will share a joint motion, then the accuracy of the velocity estimates can be increased by clustering. In particular, when a number of vertices have been paired, angular velocities can be computed and clustered, using any of a number of simple one-dimensional clustering procedures. All vertices belonging to the same polygon must yield the same angular velocity, of course, as must vertices for polygons sharing the same motion. If the angular velocity is zero, then all of these vertices must have the same linear velocity. More generally, if the angular velocity vector is ω and if the velocity of a vertex located at r_i is v_i, then the velocity of a vertex at r_j must be $v_i + \omega \times (r_j - r_i)$. These relations express the information provided by joint motions, and should be particularly useful when noise introduces uncertainty.

B. Topological Changes and Noise

When enough time elapses so that old vertices get hidden and new vertices appear, a general matching strategy be-

comes much more complicated. Here having good velocity estimates becomes especially important. If the positions of all of the polygons in the next scene can be predicted, then future overlap can be anticipated and properly taken into account. It is more difficult to anticipate how or even when two overlapping polygons will separate, but at least the possibility of matching problems can be anticipated if some of the vertices are known to be false.

The problems arising because of topological changes are extemely interesting, but difficult to solve in a way that is both efficient and general. The nature of the problem changes with the sampling rate. If the interval between successive pictures is sufficiently short, topological changes will occur infrequently, and, although they may have global consequences, they can be analyzed essentially locally. However, there is a tradeoff between the computational burden of local processing of many pictures taken at short time intervals and that of more global processing of a few pictures taken at long time intervals. At present that tradeoff is not well understood. The program described in the next section takes an essentially local approach, based on the assumption that topological changes can be treated as rarely occurring special cases to be considered when the standard matching procedure fails.

When the vertex locations are perturbed by noise, all of the angles and edge lengths change at least slightly from scene to scene. Thus, as long as no old vertices disappear and no new vertices appear, one can only state the probability that a vertex is true or false. Of course, with enough independent observations these probabilities will tend to certainty, but this is a rather uninteresting result. There is little need to try to distinguish between a noisy true vertex and a false vertex that is changing extremely slowly. For most practical purposes the two figures involved might just as well be treated as one. The main reason for wanting to detect false vertices is to avoid the serious errors in velocity estimates that can arise when two or more overlapping figures moving in different directions are treated as one rigid figure.

Actually, a single scene provides considerable information about which vertices are true and which are false. For example, true vertices are often acute, having an interior angle of less than 180°, while false vertices are often obtuse. As illustrated in Fig. 4, this is true whether the angle is on the external boundary or on an internal boundary. In fact, while true vertices need not be acute, false vertices, with one exception, must be obtuse. The exception[1] arises when a false vertex is formed where two true vertices coincide, as illustrated in Fig. 5. Excluding this as a probability zero event, we can automatically label all acute vertices as true vertices. It is also possible in principle to make more subtle inferences from other geometrical relations. For example, by extrapolating lines such as the dotted lines in Fig. 4, one can make plausible conjectures about the composition of the scene. These conjectures might be rejected after subsequent scenes are analyzed, but if they are confirmed they might speed the analysis greatly. However, we have had no experience with such procedures, and their true usefulness remains to be seen.

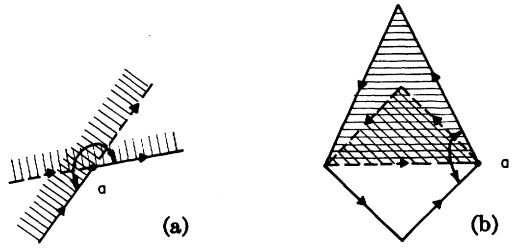


Fig. 5. (a) Normal obtuse false vertex. (b) Acute false vertex formed by coincidence of two true vertices.

IV. THE ANALYSIS PROGRAM

A. Overview

In this section we describe the major parts of a Lisp program—implemented by Petermann [7]—that analyzes pictures of moving polygons. The input to the program is a sequence of scenes $S_0,S_1,\cdots,S_k,\cdots$ where each scene is a collection of polygons. As the program runs it constructs and continually refines a model of the component polygons, including their estimated linear and angular velocities. This information is the principal output of the program. In analyzing a new scene and updating the model, the program has access to the new scene, the immediately preceding scene, and the current model. Mathematically, if M_k is the model at time k, then the program computes M_{k+1} as a function of M_k,S_k, and S_{k+1}.

Although the details of the program occasionally get rather involved, the overall strategy is straightforward. The basic operations performed are summarized in Fig. 6. Because the program begins with no past history of the motions, the initial procedure for matching the first two scenes is somewhat different from the strategy for matching subsequent scenes. This latter strategy is the more significant of the two. It is based on the assumption that the time interval between successive scenes is so short that most of the time there will be little trouble in matching a polygon in scenes S_k with a polygon in S_{k+1}, there being only one match that is at all reasonable. On occasion, however, the matching procedure will fail, primarily because the way that the figures overlap one another has changed, resulting in a change in topology. When this happens a special routine is invoked to identify the type of topological change that has occurred, and to complete the matching process.

This is clearly a heuristic strategy. No attempt is made to guarantee that matches made between successive scenes are in any sense globally optimal. No provisions for backtracking are included to allow the program to recover from an erroneous match. The primary purpose of the program is to demonstrate that a fairly simple, computationally feasible strategy for tracking overlapping figures

[1] First pointed out to us by R. Petermann.

Initialization

1. Read first two scenes, S_0 and S_1.
2. Mark acute angles as "True," obtuse angles as "Potentially False."
3. Under the assumption of no topological changes, match each polygon in S_0 with a polygon in S_1. (Failure to find a match causes program termination.)
4. Compute the linear velocities of the vertices and the angular velocities of the edges.
5. Compare the angular velocities of edges incident on each "Potentially False" vertex. If the difference exceeds a preassigned threshold, mark the vertex as "False".
6. Generate the initial model, splitting off components bounded by "False" vertices.

Main Loop

7. Read the next scene.
8. For each polygon in the previous scene, attempt a match.
9. If the match succeeds and if the numbers of vertices in the two matched polygons are the same, go to Step 13.

Topological Analysis (See Section IV-D)

10. From vertex counts, determine the type of topological change. (If there was no topological change, the program fails at this point.)
11. Locate the vertex which changed its visible/occluded status.
12. Update the model to account for topological changes.

Model Updating

13. Compute and update the linear velocities of vertices and the angular velocities of edges.
14. Compare the angular velocities of edges incident on each new "Potentially False" vertex. If the difference exceeds a preassigned threshold, mark the vertex as "False."
15. Update the model, splitting off components bounded by "False" vertices.
16. Go to Step 7.

Fig. 6. Basic steps in the tracking program.

can handle a nontrivial class of problems that frequently arise.

B. Polygon Matching

The following procedure for matching a polygon P in S_k with a corresponding polygon P' in S_{k+1} is a basic component of the tracking program. It begins by trying to pair a vertex V on the external boundary of P with external vertices of polygons in S_{k+1}. The vertex V selected for matching is not crucially important, but two criteria are used to screen out poor choices. To avoid troubles with false vertices that might appear, disappear, or change greatly in angle, a vertex is automatically excluded if its angle is obtuse. To avoid troubles with multiple matches, a vertex is excluded if its angle is within δ_1 of some other angle on the external boundary of P. If no vertex meets this latter criterion, as would happen with a square, then a repeated angle will be selected for matching, but this is assumed not to be a typical situation. If more than one vertex passes these tests, the first one encountered is used.

Associated with the vertex V is an estimated velocity vector $\boldsymbol{v}$, which initially is zero. The routine uses this estimated velocity to predict the location of V in S_{k+1}. In addition, it uses a radius of uncertainty Δr chosen with due consideration of velocities, time interval between successive scenes and the dimensions of the scene. The list of vertices of the external components of polygons in S_{k+1} is then searched to find a list of candidate vertices satisfying two conditions.

1) The location of a candidate must be within the radius of uncertainty of the predicted position of V.
2) The angle of a candidate must be within $\delta_2 (= \delta_1/2)$ of the angle at V. The parameters δ_1 and δ_2 are chosen with due consideration of noise.

If no vertex in S_{k+1} passes these tests, another vertex on P is chosen and the process is repeated until either a match is made or P is declared to be unmatchable. In this latter case a topological change is presumed to have occurred, and a special procedure for analyzing topological changes, described in Section IV-D, is invoked.

If one or more candidates are found, the best matching one is tentatively selected and a criterion function is computed that measures the angular and distance errors encountered in matching all of P with P'. If the resulting mismatch is below some arbitrary threshold, P is declared to be matched to P', P is removed from S_k, P' is removed from S_{k+1}, the model M_k is updated as described below, and the next polygon in S_k can be matched against the remaining polygons in S_{k+1}. However, if the mismatch is too great, another candidate is selected and the process is repeated until either a match is made or P is declared to be unmatchable. This latter case usually results in an uncorrectable error, although it sometimes can be handled by the section of the program that handles topological changes.

The success of this suboptimal, sequential matching approach depends to a large extent on the fact that if fairly good velocity estimates can be obtained then it is quite unlikely that an incorrect match with a nearby polygon will yield small errors. In other words, it is implicitly assumed that the scenes contain a fairly heterogeneous collection of polygons rather than a field of virtually identical figures. If the polygons are quite similar and if they move appreciably between successive scenes, then this approach places quite a burden on obtaining good initial velocity estimates. Such estimates might be obtained by using a more exhaustive or perhaps an "optimal" matching procedure whenever new figures appear and good initial velocity estimates must be obtained.[2] Another approach is to eliminate the various thresholds and to use a tree searching procedure rather than a sequential procedure during matching, allowing backtracking and rematching whenever large mismatches are encountered. We have chosen this faster and simpler approach because we were more concerned with the problem of topological changes than the problem of optimal matching.

C. Updating the Model

A polygon P in scene S_k may, of course, be the union

[2] This approach was used in the radar echo tracking program of Blackmer, Duda, and Reboh [8].

of two or more component polygons. After P has been matched with P' in S_{k+1}, changes in angles and edge lengths provide evidence about the existence of these components. The program stores these components, to the extent that they are known, in a model. In our present program, each component of P can itself be split into further components whenever angular changes at pairs of obtuse vertices are significantly greater than the changes expected due to noise (see Fig. 7). No provisions are currently made for rejoining components that might previously have been erroneously separated. The rationale here is that our chief concern is to avoid the meaningless results that are obtained when a velocity estimate is computed by averaging the different velocities of different components of a polygon. If a single polygon is occasionally represented by several components in our model, the resulting loss in accuracy in the velocity estimates is not considered to be as serious a problem.

After a polygon has been separated into components, the estimates of the velocity for each vertex are updated. In addition, the angular velocities for the edges of each component are averaged and used to update the angular velocity estimate for the whole component. This information is then available for clustering to determine the number of layers, and possibly to reunite incorrectly separated components, although neither of these operations has yet been programmed.

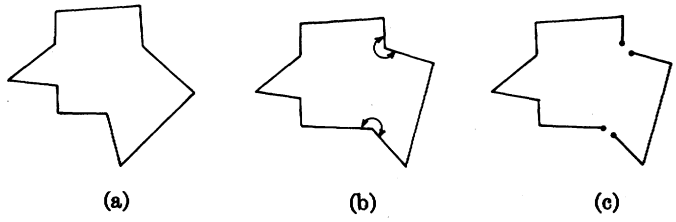


Fig. 7. Detecting components for the model. (a) Polygon at time k. (b) Obtuse angles that change significantly. (c) The two model components.

D. Topological Changes

When old vertices disappear or new vertices appear, the matching procedure described above fails and leaves one or more polygons in S_k unmatched. As was mentioned earlier, our program tacitly assumes that this is a fairly rare situation. To be more specific, we assume that the time interval between successive scenes is sufficiently short that no more than two components are involved in any topological change.

Even with this restriction there are a surprisingly large number of situations that can arise. Consider, for example, the situation in which a vertex, each of whose incident edges are visible, passes under the edge of another component and becomes hidden. The case that first comes to mind is when the two components are initially separated and then come together, as shown in Fig. 8(a). This case is obviously characterized by the fact that the number of figures in the scene is different at the two times. However, there are several other possibilities. Fig. 8(b) and (c) show that if the components overlap initially then the penetration of the vertex will create a hole. In Fig. 8(b) the vertex is initially on an external boundary component, while in Fig. 8(c) it is initially on an internal boundary component. Note, however, that we have tacitly assumed that the vertex is acute. If it is obtuse, then the components *must* overlap initially, and although no new holes are ever formed, a small old hole can disappear. In Fig. 8(d) the vertex is a true vertex on an external boundary component. In Fig. 8(e) it is a false

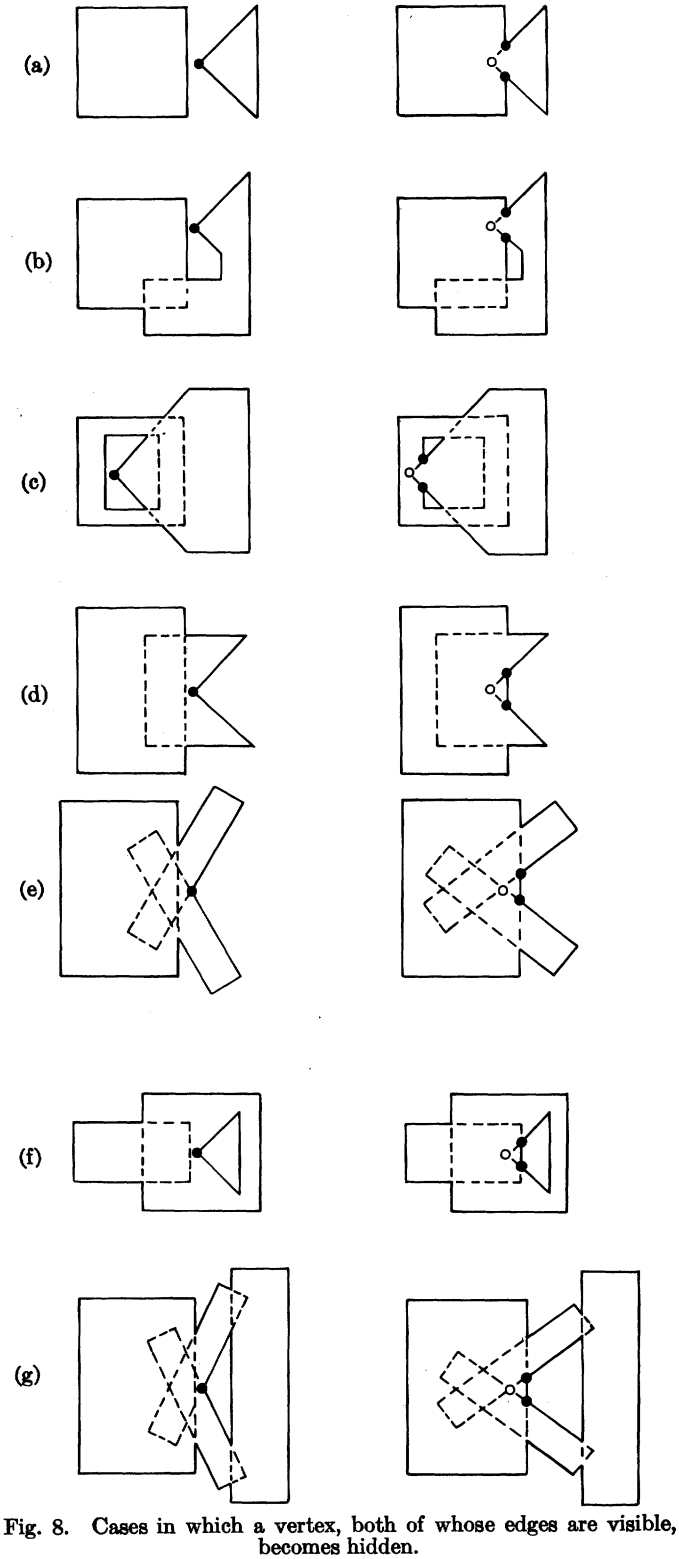


Fig. 8. Cases in which a vertex, both of whose edges are visible, becomes hidden.

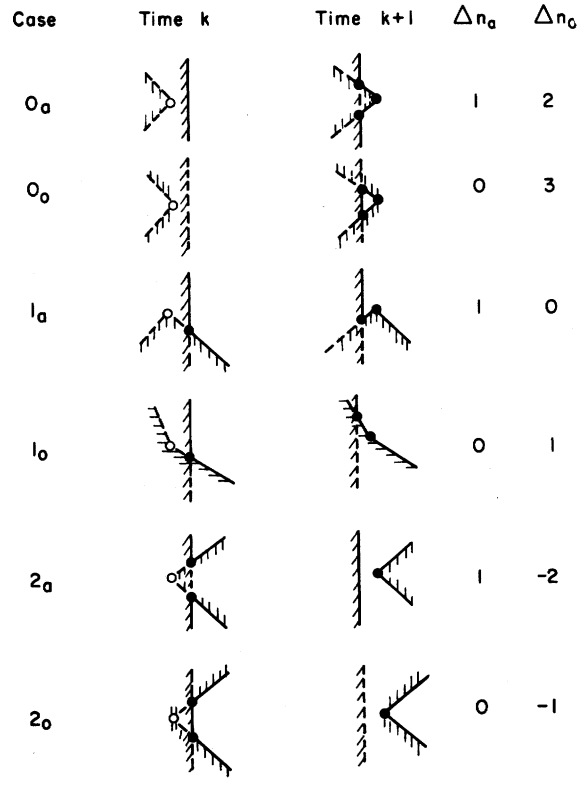


Fig. 9. Case analysis for topological changes.

vertex on an external boundary component. The two remaining situations arise when the vertex is on an internal boundary component, as illustrated in Fig. 8(f) and (g).

The various cases that can arise can be systematically classified using the following criteria.

1) Whether the number of visible edges incident on the hidden vertex is 0, 1, or 2.
2) Whether the vertex angle is acute or obtuse.
3) Whether the vertex was initially hidden and becomes visible, or initially visible and becomes hidden.
4) Whether the vertex is true or false.
5) Whether the incident edges are on an external or an internal boundary component.

The first criterion—the number of visible incident edges—provides a simple basic classification. Let case i ($i = 0,1,2$) denote the situation in which i incident edges are visible. These three cases can be further subdivided according to the second criterion—whether the vertex angle is acute or obtuse. We denote these subcases by case ia and case io, respectively. These six cases are illustrated in Fig. 9. Note that case $0o$ is rather special, since a convex vertex with no visible incident edges occurs only just before a hole is exposed.

Additional subcases can be distinguished on the basis of the remaining three criteria. Some of these distinctions are more important than others. For example, the time sequence is certainly important. In Fig. 9 the vertex is shown as initially hidden, but Fig. 9 obviously illustrates the opposite sequence equally well, and in effect shows twelve cases. Knowledge of whether the vertex is true or false can be quite helpful at times, but this information is not always available, and whenever possible we have tried to avoid depending upon it. Finally, whether the incident edges are on external or internal boundary component is primarily helpful in keeping the bookkeeping straight in case 2.

Fig. 9 also shows the change in the number of acute and obtuse angles, indicated by Δn_a and Δn_o, respectively. The numbers shown assume that the vertex in question was initially hidden; for the opposite sequence the algebraic signs should be reversed. Examination of Fig. 9 discloses that all but two of the twelve resulting cases can be distinguished by these gross vertex counts. The two that cannot be distinguished are case $1o$, in the sequence shown, and case $2o$, in the opposite sequence, since $\Delta n_a = 0$ and $\Delta n_o = 1$ in each instance. These two cases in fact cannot be distinguished without knowing which vertices are true and which are false, and to establish this may require waiting several additional time periods, if it can be done at all.

There are two reasons for wanting to distinguish between these various cases. One is to prevent the matching program from becoming confused by the topological changes in matching the remaining vertices. The other is to allow the model updating program to make the proper adjustments to the model. Fortunately, the inability to distinguish the two cases mentioned above does not effect the matching procedure. However, it does effect the updating procedure, forcing a postponement of the decision as to which model component receives the new vertex.

E. Matching and Updating When Topological Changes Occur

In this section we describe briefly the procedure used to match vertices when a topological change is known to have occurred. The first step is to count the changes Δn_a and Δn_o in the number of acute and obtuse vertices to determine which of the ten distinguishable cases has occurred.

Consider first case $0a$ (see Fig. 9). This case is characterized by the appearance of a new acute vertex having two short incident edges. The two other new vertices terminating those edges must be obtuse and close together, and the two other edges incident on those vertices must be collinear. These properties are used by a routine that searches the boundary components to locate these three new vertices. Once they are located, they are tagged so that the matching program will skip over them as it matches corresponding vertices at the two times.

This general strategy is followed in all of the other cases. The vertices that change are characterized, and the figures are searched to locate the new vertices. Once located, the vertices are tagged so that they will not interfere with the matching program, which can then proceed essentially as if no topological changes had occurred. Case 2 does present some additional difficulties, since corresponding

vertices at times k and $k+1$ can sometimes lie on different kinds of boundary components, external at one time and internal at another. Fig. 8 illustrates the various ways that this can occur. However, the difficulties essentially involve care in bookkeeping, and present no fundamentally new problems.

V. EXPERIMENTAL RESULTS

The tracking program has been run on a large number of different problems to verify that the many different cases involving topological changes are properly treated. The following two examples are relatively simple cases that are intended to illustrate the kinds of problems that the tracking program can handle, rather than to evaluate the effectiveness of the heuristic parts of the procedure.

The first example is shown in Fig. 10. The objects in the scene are a large triangle that initially completely hides a smaller triangle containing a triangular hole. Thus, this second triangle is not visible in scenes S_0 or S_1, and the initial model, M_1, is just the larger triangle. In S_3 a vertex of the obscured triangle appears (case $0a$) and the model M_3 shows that this triangle is treated as a separate component of the scene. Note that when a new vertex appears in this way, the associated obtuse vertices are known to be false without the need for computing angular velocities.

At time 5 a vertex of the hole appears (case $0o$). Although in this instance one can logically conclude that the two edges incident on that vertex must be associated with the partly exposed triangle, this association is not made in the model. This is one of many special instances in which logical inferences could be made to achieve better performance, but at a cost of greater complexity.

At time 8 a second external vertex of the hidden triangle appears (case $1a$), at time 9 a second internal vertex appears (case $1o$), at time 11 the third internal vertex appears (case $2o$), and at time 17 the third external vertex appears (case $2a$). Thus, this simple example involves all six of the cases listed in Fig. 9. Every one of these events resulted in a matching failure that had to be corrected by topological analysis. The final model, M_{17}, looks just like the corresponding scene, but of course the existence of two distinct components in the scene was known from time 3. It is this ability of the tracking program to decompose the scene that is its most important attribute.

The second example is shown in Fig. 11. It consists of three figures, two as shown in scene S_0, and a triangle with a triangular hole hidden completely by the polygon on the right hand side. In the following sequence of scenes, the triangle moves from its initial position to a position where it is completely hidden by the polygon on the left hand side. The triangle is never visible in its entirety. However, the program correctly constructs the description of the triangle together with that of the hole. In scene S_3 a vertex of the hidden triangle becomes visible, in scene S_6 the vertex of the hole becomes visible, in scene S_8 one of the vertices of the triangle gets hidden by the left hand polygon. These facts are correctly displayed in M_8. Eventually, in scene S_{19}, the triangle is completely hidden by the left hand polygon, and the program has generated the correct description of the triangle, as shown in model M_{19}. The ability of the program to generate the complete description of objects, from its partial descriptions, is another important feature of this program.

The examples above are relatively simple in terms of the number of moving figures. However, the program is applicable to scenes containing an arbitrary number of figures and has been successfully run on fairly complex scenes.

VI. DISCUSSION

We have presented an idealized model of the problem of analyzing pictures of overlapping layers of essentially two-dimensional objects. This model is a natural one for investigating the problem of tracking the motions of several layers of clouds, and, as was indicated in the Introduction, undoubtedly has other applications as well. However, like any model of real phenomena, this one focuses on certain aspects of the situation and ignores others. Here the emphasis was on the behavior of joint motions, shape changes, and false vertex motions that accompany images of moving occluding objects.

With our model, this was virtually the only kind of information available for decomposing the scene and determining the velocities. In most real applications, of course, much additional information would be available. For example, in the cloud tracking problem there are measurable differences in brightness, boundary shape, and texture between clouds in different layers. In principle, at least, such additional information should simplify the problem of decomposing the scene. In the future we hope to address the question of how such different types of information can in fact be jointly exploited.

Even within the limitations of the current model a great variety of problems arise. The model allows both very simple and arbitrarily complex scenes to be generated. This makes it difficult to provide an objective evaluation of any analysis program, for one can always generate scenes that are within the ability of the program, or that can overwhelm the program in a variety of ways.

In designing the program described in Section IV we chose to address some questions and to ignore others. Specifically, we were primarily concerned with seeing what was required to match a relatively small number of overlapping figures when only "small" changes occurred between successive scenes. The case analysis given in Section IV shows that even this restricted problem can involve a surprising variety of situations, but that it does yield to a systematic approach.

Some of the restrictions on the present program could be relaxed with relatively little effort. In particular, the restriction that no more than two components be involved in any topological change is not essential to our approach. This restriction was imposed merely to allow the case analysis to be decided by gross vertex counts. If several topological changes were allowed between successive

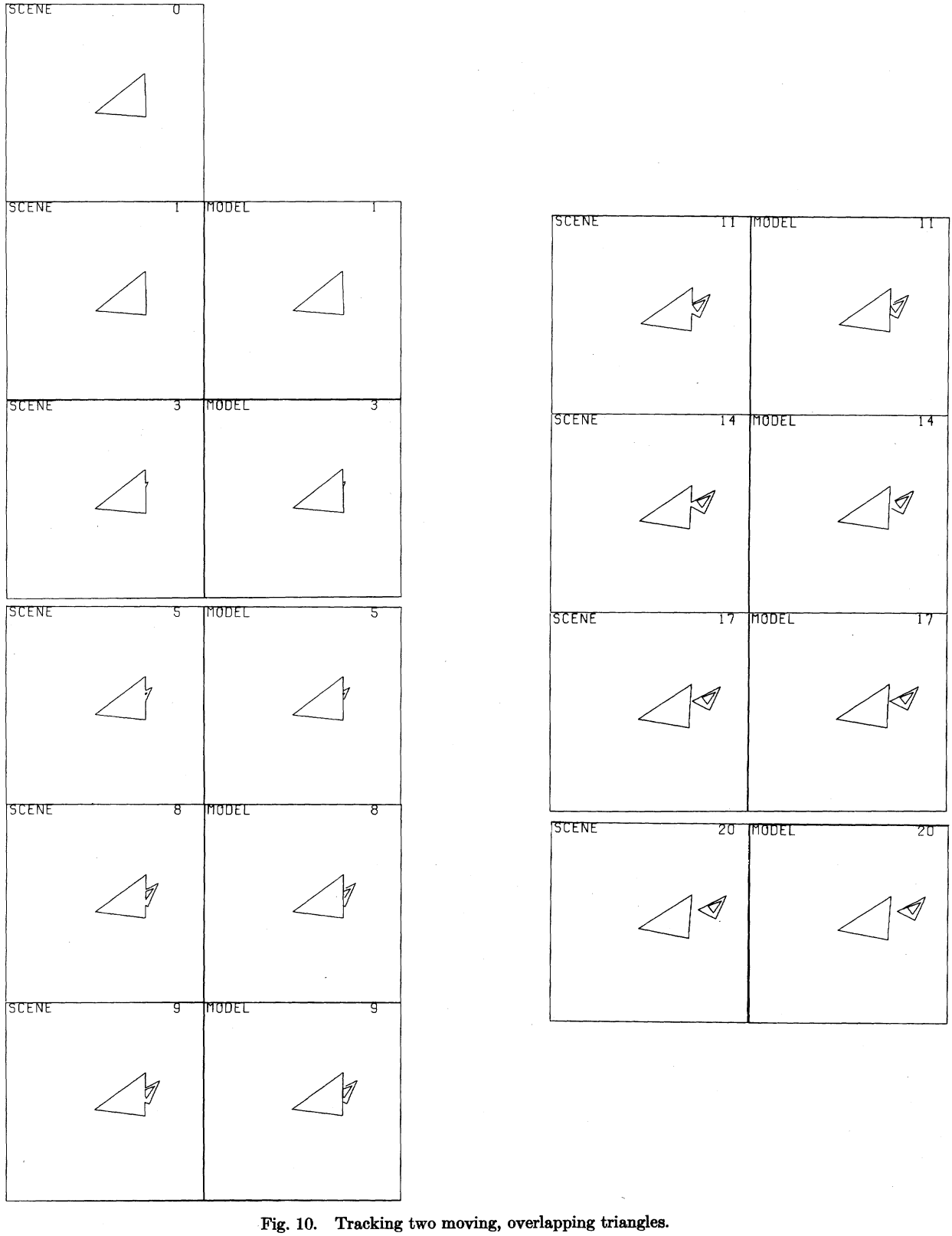


Fig. 10. Tracking two moving, overlapping triangles.

scenes, more elaborate methods would have to be used to localize the troublesome areas, but the procedure would be otherwise essentially unchanged. It would be more difficult to change the program to cope with large numbers of polygons, nonrigid polygons, or polygons whose number of sides and/or components could change from time to time. All of these problems present interesting and challenging topics for further study.

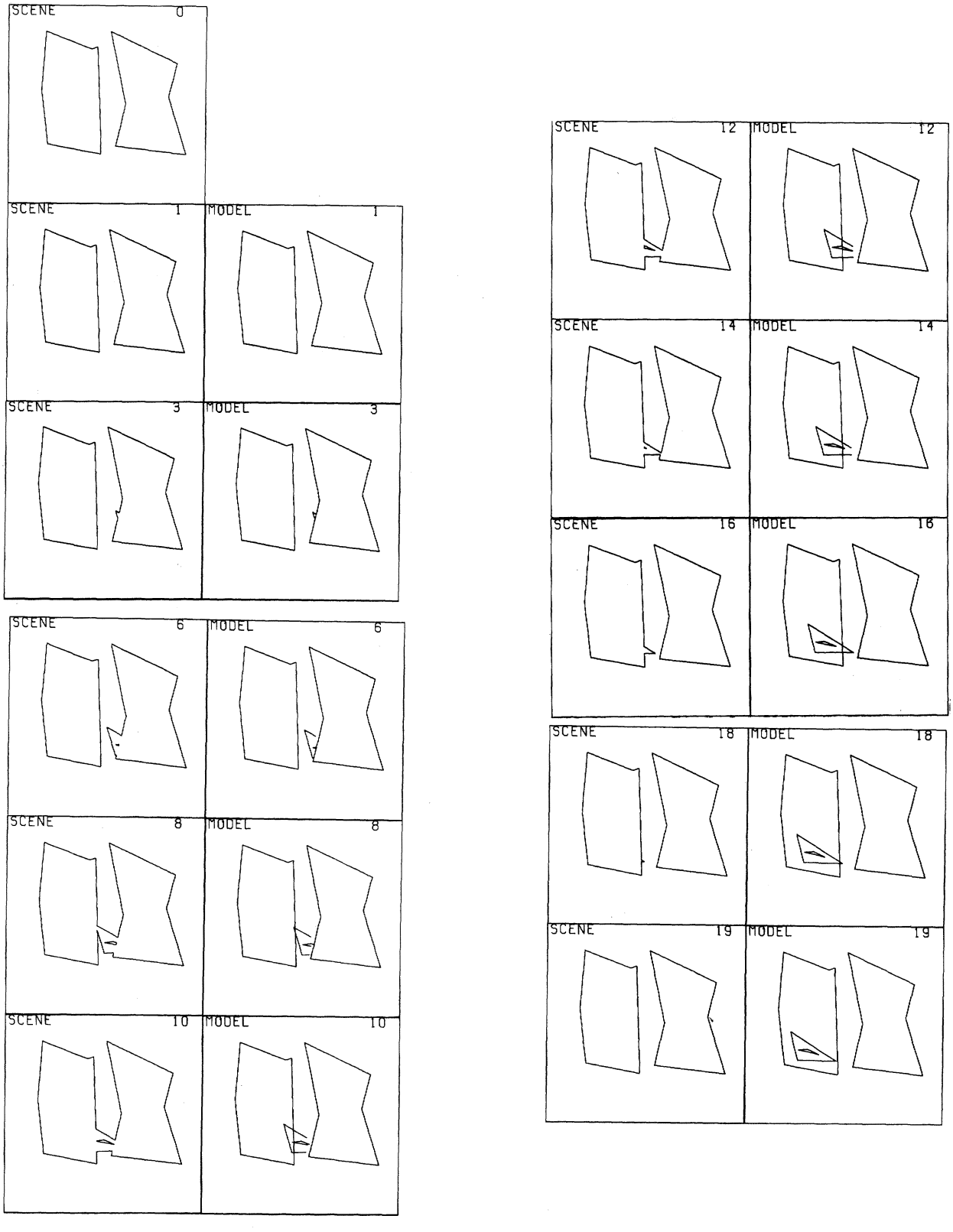


Fig. 11. Tracking three moving figures.

ACKNOWLEDGMENT

The authors would like to acknowledge the valuable assistance of R. Petermann, who wrote the analysis program described in Section IV and who contributed significantly to its final design. We would also like to thank C.-C. Lin, who devised and implemented the algorithm for generating the data used to test the analysis program, and Dr. S. A. Underwood for his help during this research.

REFERENCES

[1] M. S. Ulstad, "An algorithm for estimating small scale differences between two digital images," *Pattern Recognition*, vol. 5, pp. 323–333, Dec. 1973.

[2] J. A. Leese, C. S. Novak, and V. R. Taylor, "The determination of cloud motion patterns from geosynchronous satellite image data," *Pattern Recognition*, vol. 2, pp. 272–292, Dec. 1970.

[3] E. A. Smith and D. R. Phillips, "Automated cloud tracking using precisely aligned digital ATS pictures," *IEEE Trans. Comput.*, vol. C-21, pp. 715–729, July 1972.

[4] D. J. Hall, R. M. Endlich, D. E. Wolf, and A. E. Brain, "Objective methods for registering landmarks and determining cloud motions from satellite data," *IEEE Trans. Comput.* (Short Notes), vol. C-21, pp. 768–776, July 1972.

[5] J. L. Potter, "Motion as a cue to segmentation," in *Proc. Milwaukee Symp. Automatic Control*, Milwaukee, Wis., Mar. 1974, pp. 100–104.

[6] N. Badler, "Three-dimensional motion from two-dimensional picture sequences," in *Proc. 2nd Int. Joint Conf. Pattern Recognition*, Aug. 1974, pp. 157–161.

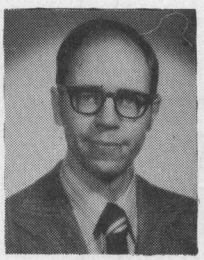
[7] R. A. Petermann, "Computer analysis of planar motion of polygons," M.S. thesis, Dep. Elec. Eng., Univ. Texas, Austin, Tex., Jan. 1975.

[8] R. H. Blackmer, Jr., R. O. Duda, and R. Reboh, "Application of pattern recognition techniques to digitized weather radar data," Stanford Res. Inst., Menlo Park, Calif., Final Rep., Contract 1-36072, SRI Project 1287, May 1973.

J. K. Aggarwal (S'62–M'65) was born in Amritsar, India, on November 19, 1936. He received the B.S. degree in mathematics and physics from the University of Bombay, India, the B. Eng. degree from the University of Liverpool, Liverpool, England, and the M.S. and Ph.D. degrees from the University of Illinois, Urbana, in 1956, 1960, 1961, and 1964, respectively.

He joined the University of Texas, Austin, in 1964. He was a Visiting Assistant Professor at Brown University, Providence, R.I., in 1968 and a Visiting Associate Professor at the University of California, Berkeley, during 1969–1970. Currently he is a Professor of Electrical Engineering and Computer Science. He has published numerous technical papers and a textbook, *Notes on Nonlinear Systems* (Princeton, N.J.: Van Nostrand Reinhold, 1972), in the series Notes on Systems Sciences. His current research interests are digital filters, computational methods, and image processing.

Dr. Aggarwal is an AdCom member of the IEEE Circuits and Systems Society, Chairman of the Technical Committee on Signal Processing, Associate Editor of the Society's TRANSACTIONS and Co-Guest Editor of the Special Issue on Digital Filtering and Image Processing, March 1975. He is the past Editor of the Circuits and Systems Society Newletter. He is a member of Sigma Xi, Eta Kappa Nu, and Rotary International.



Richard O. Duda (S'57–M'58) was born in Evanston, Ill., on April 27, 1936. He received the B.S. and M.S. degrees in engineering from the University of California, Los Angeles, in 1958 and 1959, respectively, and the Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1962.

He joined the staff of the Stanford Research Institute, Menlo Park, Calif., in 1962, where he participated in research on character recognition, statistical pattern classification, adaptive and learning systems, visual scene analysis, and artificial intelligence, together with their applications to meteorology and industrial automation. He has been the author or coauthor of a number of papers in these fields, and the coauthor with Dr. P. E. Hart of the book *Pattern Classification and Scene Analysis* (New York: Wiley-Interscience, 1973). During the 1973–1974 academic year he was a Visiting Associate Professor in the Department of Electrical Engineering and the Department of Computer Sciences at the University of Texas, Austin.

Dr. Duda is a member of Tau Beta Pi, Sigma Xi, Phi Beta Kappa, the Association for Computing Machinery, and the American Association for the Advancement of Science, and is a past Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS.