DYNAMIC SCENE ANALYSIS:

THE STUDY OF MOVING IMAGES*

by

W. N. Martin and J. K. Aggarwal

Departments of Computer Science
and Electrical Engineering

Technical Report No. 184
January 15, 1977

INFORMATION SYSTEMS RESEARCH LABORATORY

ELECTRONICS RESEARCH CENTER
THE UNIVERSITY OF TEXAS AT AUSTIN
Austin, Texas 78712

*This research was supported in part by the Air Force Office of Scientific Research Grant 77-3190, and by the Joint Services Electronics Program under Contract F44620-76-C-0089.

ABSTRACT

Scene analysis has long been a fusion point between the fields of pattern recognition and artificial intelligence: it integrates techniques from both disciplines. Usually, these techniques are applied to single frames containing static images, but recently there has been growing interest in developing techniques which could be applied to scenes containing moving images. This report contains two major parts: the first part is a survey of the computer systems, as reported in the literature, which attempt to analyze scenes containing moving images; the second part is a detailed description of a system developed by the authors to analyze scenes containing moving planar curvilinear objects. Included are examples which indicate the success of the authors' system,as well as the need for further study.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This report has two main purposes. The first is to present a detailed description of previous attempts to develop computer systems which analyze the temporal features of a visual scene. Chapter 2 begins with a general discussion of this problem domain, but is primarily concerned with a survey of the recent literature on what we will refer to as dynamic scene analysis. The second purpose is to present the results of our attempt to develop a dynamic scene analysis system, and Chapter 3 contains a rather detailed discussion of the program as well as some of the examples which have been analyzed. A brief description of the human visual system will comprise the remainder of this chapter and set the context for the further presentations.

The human eye, indeed any biological vision system, has an enormous capability for efficiently and effectively transferring complex information. The study of biological vision systems has shown that the eye is not a mere transducer which accepts visual stimuli and transmits neural stimuli; the transference process also involves the correlation and integration of both spatial and temporal features of the visual stimulus. The psychological literature is replete with studies of visual perception, but a few representative texts are those of Refs. 1-3. The endeavor to develop computer systems with these capabilities is generally called scene analysis.

The papers presented in Refs. 4-6 give a good indication of the scope and depth of the field of scene analysis: they show that the prime concern has been to develop methods to detect, represent, store, and manipulate the

spatial features of a scene. Thus scene analysis systems analyze a single static image or picture in order to determine the constituents of the image as well as obtain information about their structure. In most cases the problem of analyzing the temporal features of the visual stimuli has been put aside for further study; but it is clear that in biological systems the dynamic nature of the stimulus is used extensively. In fact, Johannson [7] states that "a frog or chameleon, for example, can perceive and catch its prey only if the prey is moving." Various studies [8,9,10] on the eyes of cats, frogs,and rabbits have found physiological structures specifically for detecting motion. MacKay [11] presents a theory that humans have similar structures,and Schouten [12] concludes from his experiments on humans that "these findings strongly support MacKay's hypothesis that 'detectors of motion as such' exist in the human visual system."

The retina of the human eye has an uneven distribution of receptors, with the highest density around the fovea. Price [13] states that "visual acuity is best at the fovea," and "since acuity is so poor at the periphery, the eye must be moved around the scene so that areas of interest are projected on the fovea." The attentive processes of the human visual system operate on the stimuli which impinge upon the fovea, while the processes on the periphery must recognize "areas of interest" for future attention. Thus there must exist parallel processes, some of which perform the detailed attentive functions at the fovea, while others "watch" the peripheral areas of the visual field for interesting features. Clearly, these

features could be shape, texture, and color; but more importantly, movement is such a feature. Price [13] states "the periphery is sensitive to motion; a movement in the periphery attracts attention." Just as the peripheral processes must be able to detect motion and direct the attentive processes to it, the attentive processes must be able to track the movement and attend to the details of the objects in motion. These two phases of visual perception are exemplified by a person who is crossing a street. The person must be able to notice "out of the corner of his eye" the movement of a car, as well as be able to look at the car and judge its speed and direction of movement. Chen and Jones [14] give a sketchy description of a similar multileveled system.

It should be noted that the two phases mentioned above are, essentially, not cognitive processes. A higher-level cognitive process is required to decide questions such as which detected area of interest should be attended to, and what detailed features should the attentive processes consider? This multilevel structure implies that the various processes deal with the stimuli at different levels of abstraction: the peripheral or motion detection processes react in terms of movement itself in various areas of the visual field; the attentive processes refer to the motion of particular (if yet unnamed) objects; while the cognitive processes relate both of these to the current "psychological set" of the person, his knowledge, and expectations.

2. A HISTORICAL SURVEY

2.1 Dynamic Scene Analysis in General

This chapter will describe several computer systems which perform the functions of the peripheral and attentive processes discussed in the previous chapter. Since the analysis required by these two types of processes is quite different and since no system currently performs both functions, the description will be split into two sections. But first, we will discuss the general problems involved in computer analysis of the temporal features of a visual scene, referred to here as dynamic scene analysis. The general, problems discussed in this section are problems encountered by both the peripheral and attentive phases of dynamic scene analysis.

In contrast to the single static image usually taken as input by standard scene analysis systems, a dynamic image is the input for the systems described here. A "dynamic image" is a series of static images (referred to here as frames) with a given or assumed time function relating the order and the elapsed interval between elements of the series. This, of course, is taken from the paradigm of the dynamic images generated by the static frames of a movie; however, the concept is founded, as is any discretizing function, on the principle that if the sampling interval is smaller than the interval required to resolve the smallest feature of interest, then no important information is lost by the function.

The analysis of dynamic images differs from standard scene analysis in that not only must information be extracted from each frame, but informa-

tion must also be extracted from the series as such: this means that the details derived from each image must be integrated into a coherent whole. This integration is not a simple compiling of facts, because change is occurring in the appearance of the scene between frames; thus a given feature or structure of a scene must be recognized even though it is continually changing. But the changes as well as the structures are important in dynamic image analysis, so the techniques used to recognize the structure, regardless of a certain set of changes, must also be able to identify the changes that occur. This process is further complicated by the existence of noise in the scene: noise introduces changes which must be ignored. Thus, a dynamic image analysis system must be able to separate the constancies from the changes, and be able to separate the interesting changes from the noisy ones. In a slightly different form, Futrelle [15] presents this aspect of the problem by saying that analysis of dynamic images

> ...is not, in fact, a case of concatenating the analysis of a number of static frames. Explicit algorithms would have to be devised to correlate sequential frames, to associate apparently different but semantically identical items, to handle objects which progressively occlude one another or otherwise appear or disappear, etc., ad nauseum.

Figure 1 contains the gray-scale overprints of two frames which show how "semantically identical" images may appear different in dynamic scenes.

Of particular interest in Futrelle's quote is that he included occlusion as part of the ad nauseum. Indeed, occlusion remains a major problem in both static and dynamic scene analysis. Although some amazing results

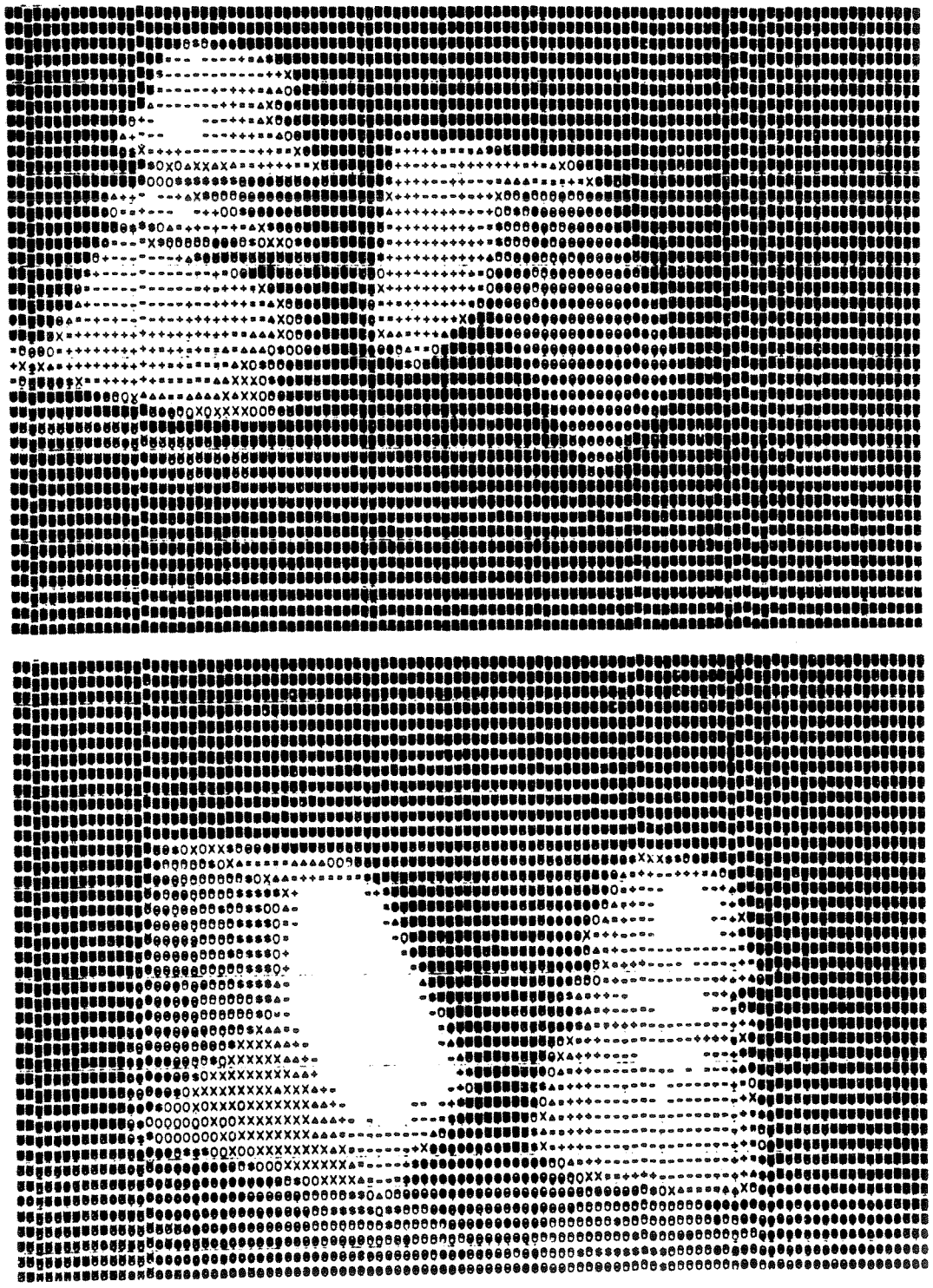


Figure 1. Two Frames Showing the First Major Problem in Dynamic Scene Analysis: Associating Semantically Identical Images Although They Appear Different.

for analyzing occluding objects have been obtained for polygonal shapes in static [16] and dynamic [17] scenes, these methods do not seem to be generalizable to scenes containing curvilinear shapes. Kasvand [18] describes the problem of occlusion as a paradox. In his words,

> ... we immediately encounter a paradox, i.e., it is impossible to extract one object from the picture before the object is recognized and it is impossible to recognize the object before it is extracted, standardized for size, etc. In other words, attempts to process one complete object for recognition will not succeed.... Most often this difficulty is avoided by defining that the objects are not to touch or overlap. However, the paradox is one of the fundamental problems in picture analysis....

Figure 2 shows several frames from Ref. 19 which show an apparently single object as analyzed into its appropriate occluding parts.

As the task of dynamic scene analysis has been described here, it contains two major problems: (a) to associate "semantically identical" images although they appear different; and (b) to solve the occlusion "paradox". We shall now turn to a description of various attempts to analyze dynamic scenes, and later return to these problems to present some promising directions for the further research needed to solve them. Table 1 lists the papers surveyed in the next two sections in a chronological ordering within the major subdivisions, with common techniques noted where applicable.

2.2 Motion Detection

The majority of what little work has been done on dynamic scene

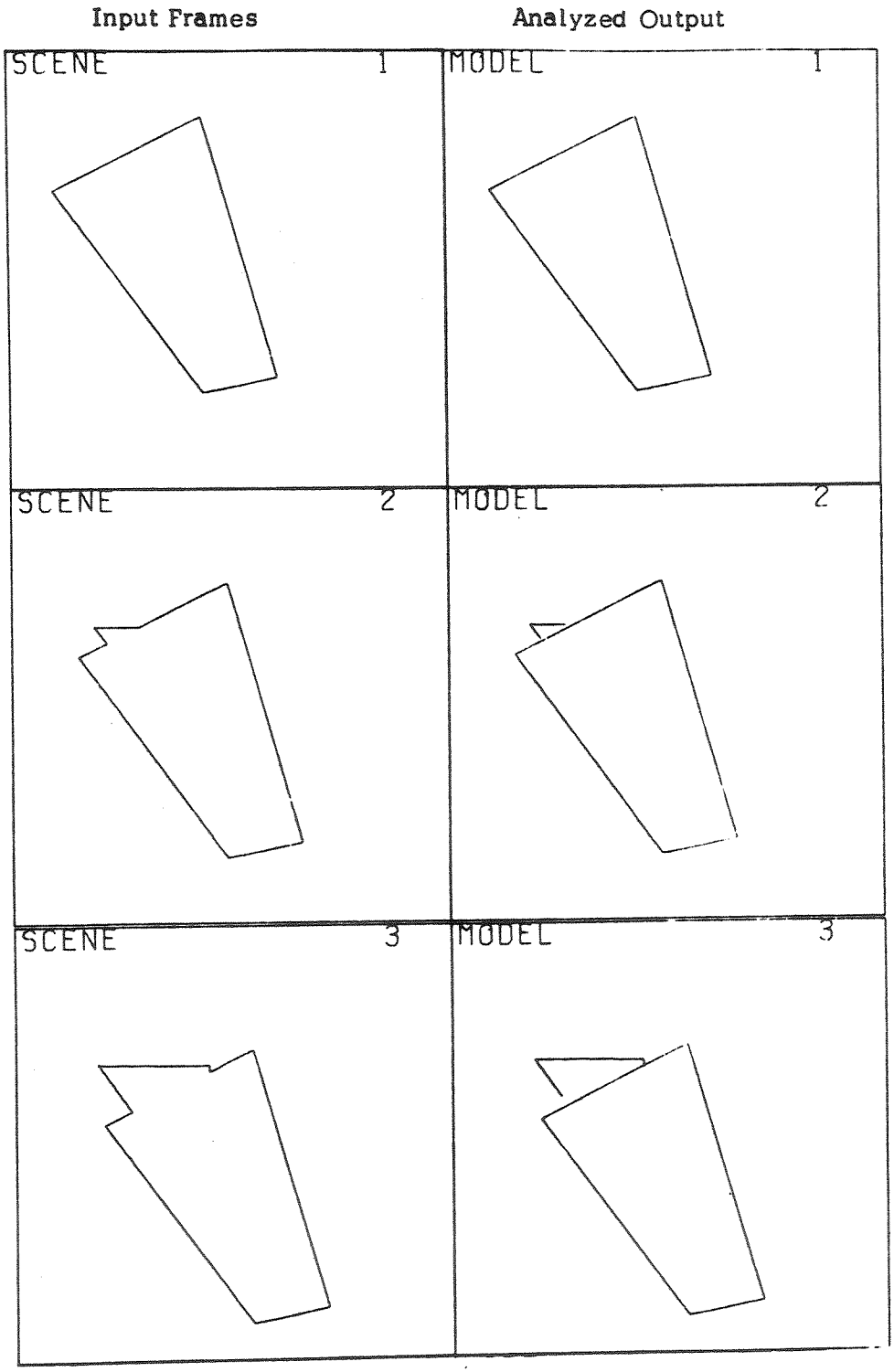


Figure 2. Consecutive Frames Exemplifying the Second Major Problem in Dynamic Scene Analysis: Occlusion.

Table 1. Chronological Listing of Motion Studies Within the Major Subdivisions, and With Common Techniques Noted Where Applicable.

Peripheral processes: motion detection			Attentive processes: motion analysis	
cross-correlation	image differencing	miscellaneous	centroid matching	shape analysis
Leese, Novak, Taylor, 1970 (20)	Lillestrand, 1972 (30)	Potter, 1974 (28)	Endlich, Wolf, Hall, Brain, 1971 (43)	Aggarwal, Duda, 1975 (17)
Leese, Novak, Clark, 1971 (23)	Ulstad, 1973 (31)	Potter, 1975 (29)	Greaves, 1975 (46)	Peterman, 1975 (45)
Smith, Phillips, 1972 (42)	Limb, Murphy, 1975 (32)	Fenton, Vas, 1974 (38)		
Lo, Parikh, 1973 (25)	Nagel, 1976 (33)	Lappalainen, Tervonen, 1975 (40)		
Lo, Mohr, Parikh, 1974 (26)		Chien, Jones, 1975 (14)		
Lo, Mohr, 1974 (27)		Nevatia, 1976 (41)		
Arking, Lo, Rosenfeld, 1975 (24)		Chow, Aggarwal, 1976 (36)		

analysis concerns motion detection systems. The methods used in these systems are best suited for the peripheral "attention attracting" processes because they involve features which are in some sense global. In most cases the analysis yields a motion vector for an entire frame or some arbitrary subset of it. They do not pick out entities within the scene, which means that although they tend to solve the first of the two major problems listed above, they cannot solve the second problem. The details of these problems will be presented as each system is discussed.

The first dynamic scene analysis problem to gain much attention was the automated detection and measurement of cloud motions from satellite photographs, and we will discuss two such systems in this section and one in the next section. Leese, et al [20] use two methods to implement their automated system: in each case, they compare two successive pictures with the first picture divided into systematic sections (64 x 64 pixels); then for each section, some reasonable area of the second picture is searched for a good match to the original section. The first technique is to form a "cross-correlation coefficient" using the fast-Fourier transform on the full gray-scale values within the section. The cross-correlation coefficient is computed for each pairing of the original section to a candidate section in the second picture; the candidate section which yields the maximum coefficient is chosen as the match. The second technique is to first form binary images of the original pictures by recording a one for each spatial point whose gray-scale value is within a given interval, and a

zero otherwise. This interval was chosen to yield the edges of the cloud formations, i.e., a value greater than the dark background yet less than the bright interior of the formations. The sections of the first binary image are then mapped to an appropriate match in the second binary image by a "relatively simple matching technique," which is a modification of a method developed by Moore and refined for cloud motion study by Bristor et al [21,22].

In both cases, the motion vector is computed as the distance and direction between the center of the original section to the center of the matched section; this essentially assigns a motion vector to the section, not to any feature within the section. Herein lies the major drawback of this system: the pictures are sectioned arbitrarily so that if there are two or more features in a section, each having a different motion vector, the section cannot be matched in the second image; alternatively, the section is matched but the resulting vector is a weighted average of the actual vectors. This particular problem is discussed in more detail in Ref. 23: the authors state that "neither technique was successful in discriminating the motion vectors when there were two or more layers of clouds present"; they conclude that "the inability of the automated techniques to discriminate among multiple cloud layers present in the same image sector precludes their implementation as a completely automated operation."

Arking, Lo,and Rosenfeld [24] present a summary of a series of papers [25-27] on the use of two Fourier-transform techniques to estimate cloud motions from a pair of successive pictures: the first technique is

to perform cross-correlation on the gray-scale values of the two pictures to form a measure of the indicated motion; the second technique is a phase-shift analysis in the frequency domain of the transformed images. The phase-shift method has the advantage that,computationally,it is simpler than cross-correlation; it has the additional feature that it yields "many independent estimates" of the motion, whereas the cross-correlation method yields one. This would appear to be an advantage; however, the various estimates have proved difficult to analyze coherently in all but the most straightforward examples.

Each technique was applied to both simulated and real images. Figure 3a shows consecutive simulated frames (taken from [24]) in which two clouds are moving together. Both techniques gave good results even though most of one cloud has moved out of the field of view. Two more frames in which the clouds are moving in various directions are presented in Figure 3b; in this case, the cross-correlation yields a weighted average of the velocities, while the phase-shift method gave inconclusive results.

The authors conclude that other techniques are necessary to pre-process the images in order to separate the clouds into groups that are likely to be moving together,and then the Fourier-transform methods can be used to analyze the motion of each group. They suggest that if the altitude of the clouds could be determined,it would be an appropriate feature on which to base the separation processor.

Potter [28,29] describes a system which uses motion as a method to

GRAY LEVELS OF PICTURE NO. 1

```
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0
 0  0 20 20 20 20 20  0  0  0  0  0  0  0  0  0
 0  0 20 20 20 20 20  0  0  0  0  0  0  0  0  0
 0 20 20 20 20 20 20 20  0  0  0  0  0  0  0  0
 0  0 20 20 20 20 20  0  0  0  0  0  0  0  0  0
 0  0 20 20 20 20 20  0  0  0 20  0  0  0  0  0
 0  0  0  0 20  0  0  0 20 20 20 20 20  0  0  0
 0  0  0  0  0  0  0 20 20 20 20 20 20 20  0  0
 0  0  0  0  0  0  0 20 20 20 20 20 20 20  0  0
 0  0  0  0  0  0 20 20 20 20 20 20 20 20 20  0
 0  0  0  0  0  0  0 20 20 20 20 20 20 20  0  0
 0  0  0  0  0  0  0 20 20 20 20 20 20 20  0  0
 0  0  0  0  0  0  0  0 20 20 20 20 20  0  0  0
 0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

GRAY LEVELS OF PICTURE NO. 2

```
20 20 20  0  0  0  0  0  0  0  0  0  0  0  0
20 20 20 20  0  0  0  0  0  0  0  0  0  0  0
20 20 20  0  0  0  0  0  0  0  0  0  0  0  0
20 20 20  0  0  0 20  0  0  0  0  0  0  0  0
20  0  0  0 20 20 20 20 20  0  0  0  0  0  0
 0  0  0 20 20 20 20 20 20 20  0  0  0  0  0
 0  0  0 20 20 20 20 20 20 20  0  0  0  0  0
 0  0 20 20 20 20 20 20 20 20 20  0  0  0  0
 0  0  0 20 20 20 20 20 20 20  0  0  0  0  0
 0  0  0 20 20 20 20 20 20 20  0  0  0  0  0
 0  0  0  0 20 20 20 20 20  0  0  0  0  0  0
 0  0  0  0  0  0 20  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Figure 3a. Example Frames for which Cross-Correlation Gave Good Velocity Estimates.

GRAY LEVELS OF PICTURE NO. 1

```
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0
 0  0  0  0 20  0  0  0  0 20 20 20 20 20  0  0
 0  0 20 20 20 20 20  0 20 20 20 20 20 20 20  0
 0  0 20 20 20 20 20  0 20 20 20 20 20 20 20  0
 0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
 0  0 20 20 20 20 20  0 20 20 20 20 20 20 20  0
 0  0 20 20 20 20 20  0 20 20 20 20 20 20 20  0
 0  0  0  0 20  0  0  0  0 20 20 20 20 20  0  0
 0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0
```

GRAY LEVELS OF PICTURE NO. 2

```
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0
 0  0 20 20 20 20 20  0  0  0  0  0  0  0  0  0
 0  0 20 20 20 20 20  0  0  0  0  0  0  0  0  0
 0 20 20 20 20 20 20 20  0  0  0  0  0  0  0  0
 0  0 20 20 20 20 20  0  0  0 20  0  0  0  0  0
 0  0 20 20 20 20 20 20 20 20 20 20 20  0  0  0
 0  0  0  0 20  0 20 20 20 20 20 20 20  0  0  0
 0  0  0  0  0  0 20 20 20 20 20 20 20 20  0  0
 0  0  0  0  0 20 20 20 20 20 20 20 20 20  0  0
 0  0  0  0  0  0 20 20 20 20 20 20 20  0  0  0
 0  0  0  0  0  0 20 20 20 20 20 20 20  0  0  0
 0  0  0  0  0  0  0 20 20 20 20 20  0  0  0  0
 0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0
```

Figure 3b. Example Frames for which Cross-Correlation Gave a Weighted Average Estimate Instead of the Correct Velocity.

Figure 3. Cross-Correlation Example.

segment a scene into objects. His original work [28] focused on the points of a grid superimposed over each of two gray-scale images. For each point of the grid, he calculates the distance to a discontinuity in the gray-scale image or to the boundary of the image in each of four directions; this yields eight distance measurements (four for each picture) for each grid point. From these distance measurements, eight motion measurements are calculated, which in turn allow each point to be classified as being in either the "body" of an object, the motion "shadow" of an object, or the background. Clearly, the problem with this type of analysis is that it classifies the points of the grid and not features, in either of the pictures. The motion "shadow" classification is one good example of this because it represents a feature which is in neither of the pictures. An area on the grid is called a "shadow" if an object occupied that area in one of the pictures, but did not occupy it in the other; actually, "shadows" are features of the difference image between the pictures, not the pictures themselves.

Potter recognized the problem of focusing directly on grid points, and in his latest work [29] he presents a variable "cross-shaped template" which is generated from the first picture, then searched for in the second picture. The term "cross-shaped" refers to the fact that the template has an "arm" in each of the four directions. The template is variable because the length of each arm is determined by the distance in the direction of the arm from the center of the template to the nearest discontinuity in the gray-scale image of the first picture. After such a template has been generated,

a heuristic search for a match is begun at the same grid point of the second picture. If a match is found, then a velocity value is computed by comparing the position of the matched templates and this value is associated with the grid point of the center of the template in the first picture; if no match is found before a preset search limit is reached, then a null value (different than zero) is associated with the point. Now, the scene of the first picture can be segmented by grouping together points which are associated with the same velocity values.

Potter states, "the avowed purpose of this procedure is to obtain a crude first approximation of the basic object segments in a scene." Indeed, this must be taken as a simple tool to be used to complement other segmenting processes. The clearest example of this is that if a series of scenes have an object moving in them the process will correctly extract the object, but if the object slows to stop it will be swallowed up by the background. There are several fairly trivial problems such as just mentioned, but there are two debilitating problems: the first is that the procedure cannot analyze occluding objects; the second and most important is that the objects are not allowed to exhibit rotational velocity. The second problem is of prime importance because rotational velocity is an integral part of any motion, and yet it does not appear that the process can be readily extended to analyze rotational velocity.

The papers [30,31] described in the following do not deal directly with motion analysis; they do not attempt to recognize any particular

feature in either of the two successive pictures, therefore they cannot ascribe motion to a feature or even to the scene as a whole. They do, however, address the problem of determining areas of change between two images of the same scene. The areas of difference are found by a simple subtractive process, but the simplicity of this operation requires that the images must be carefully aligned by both spatial coordinates and intensity value. The spatial registration is done by considering one image as the reference image and then distorting the other image until they are aligned: the distortion is a localized procedure which operates on subregions (Ulstad calls them submatrices) of the images. Cross-correlation techniques are used to compute the amount of distortion necessary to align a subregion with its corresponding subregion in the reference image. After the spatial registration has been completed, the gray-scale values must be matched: Lillestrand calls this "transparency rectification", while Ulstad refers to it as "moment matching" because the process matches the first two central moments of the gray-scale values of a given subregion with the moments in the corresponding subregion in the reference image.

Once the images have been "rectified" a point-to-point subtraction process generates a third image which displays the small-scale differences between the given images. "Small-scale" is an important qualification because it seems that the areas of change must be of a size which is inconsequential to the rectification process, otherwise these processes would become lost when trying to match the subregions.

A hardware implementation of another subtractive method is discussed by Limb and Murphy [32]: here the first of two consecutive frames from a television camera is delayed so that the corresponding pixels of each frame can be compared. In addition to those between-frame comparisons, within-frame comparisons were made among pixels and their suitable neighbors. The within-frame comparisons are used to normalize the between-frame comparisons: these comparisons are essentially the absolute differences of the pixel intensity values and are summed over the entire frame. This yields a velocity estimate for the frame as a whole and not for any feature in particular; in fact, the system has not been tested on scenes with more than one moving object,much less on scenes containing occluding objects. Additionally, it appears that there are some rather subtle relationships between the texture of the background and the effectiveness of the system. In the face of these difficulties the system did give fairly accurate estimates for the velocity of an object which was moving in the range of 0 to 3 pixels per frame: the authors propose extensions to the normalizing process,which should give better velocity estimates. Yet, the system remains fundamentally a motion-detecting process because it "provides an estimate of the average picture element displacement" between the frames.

A rather different form of image-differencing is used by Nagel [33] to initially extract a single moving object from a dynamic scene; the velocity measures calculated from the initial extraction are then used to further refine the form of the extracted object. This system uses a complex

dynamic image. From the original sequence of frames two subsequences must be chosen so that the initial phase can analyze the differences between corresponding frames of the two subsequences, while the refinement phase can analyze the similarities of the consecutive frames of the second subsequence.

Each frame is segmented by a modified version of Yakimovsky's region-growing algorithm [34,35]. The remaining analysis is performed on these regions. Corresponding frames of the two subsequences are compared to find overlapping regions whose gray-value distributions are similar; these matched regions are then removed from the frames of the second subsequence leaving the unmatched regions of each frame. If the two subsequences have been chosen correctly, then the largest 4-connected group of unmatched regions will be a good initial estimate of the moving object. Velocity vectors are now computed for the "object-candidates" by performing cross-correlations of the region boundaries in consecutive frames of the second subsequence. Using the velocity estimates, the "object-candidates" of all the frames can be spatially normalized, superimposed, and passed through a thresholding procedure to yield a final representation of the moving object.

This last process is similar to Potter's approach in that it requires the object to exhibit only translational velocity, while the initial process is similar to the differencing schemes previously mentioned in that it cannot handle more than one moving object. The entire approach, however, has

the more serious problem of specifying the proper subsequences. This choice is critical to the performance of the system, yet the author can only suggest how it might be done automatically, since in the examples given it was chosen manually.

The next paper, Chow and Aggarwal [36], is dissimilar from the other papers in this section because it relies on a preprocessor to extract the objects from the frames; but, the system is more importantly similar to the other systems in that it analyzes features global to the objects. Some of the features used are the area, the position of the centroid, and the angle (with respect to horizontal) of the second central moment of the figure. The dynamic scenes analyzed by this system are taken from an image-dissector camera viewing rigid curvilinear opaque two-dimensional figures. A preprocessor [37] is used to generate a binary image with the boundaries of the figures marked as ones and the remainder of the image unmarked. The figures are allowed to move at various velocities about the field-of-view of the camera; however, when one figure occludes another the camera and preprocessor are unable to distinguish the intersection so that the boundaries seem to merge and separate as the figures move.

As long as no occlusion occurs,the figures are tracked through their various motions by simply matching the global features of the current frame to those which are stored in the model. The features of the matched figures in the model are then updated and new velocity estimates for each figure are calculated. The occlusion of two or more figures requires the generation

of a "predictive" model. Using the calculated velocities and figure descriptions stored in the model, a frame is generated which "predicts" the appearance of the figure resulting from the occlusion. If the actual figure in the input frame matches the generated figure of the "predictive" model, then the system assumes that its current velocity estimates are correct and simply updates the model for each figure. In the example of Fig. 4, the "predictive" model is not required until the third frame; at that point, the images occlude one another and the system must form a joint image from the descriptions retained in the model in order to correctly analyze the input frame. If the "predictive" model does not match the actual figure, then the system halts. This brings out the most serious restriction of the system, that once two or more figures start occluding they must move with constant velocities until they separate again. This restriction is due to both the simple nature of the predictive model and the global aspect of the identifying features.

There have been several rather novel applications of motion analysis, and we will finish this section by discussing three of them. Fenton [38] used a grating to project contour lines and reference points onto a moving object, then analyzed the deformations of the contours to measure the movement of the object (in this case the object was a living dog's heart). This is essentially a range-finding method similar to those used in some static scene analysis systems [39].

A binocular range-finder was used by Lappalainer and Tervouen [40]

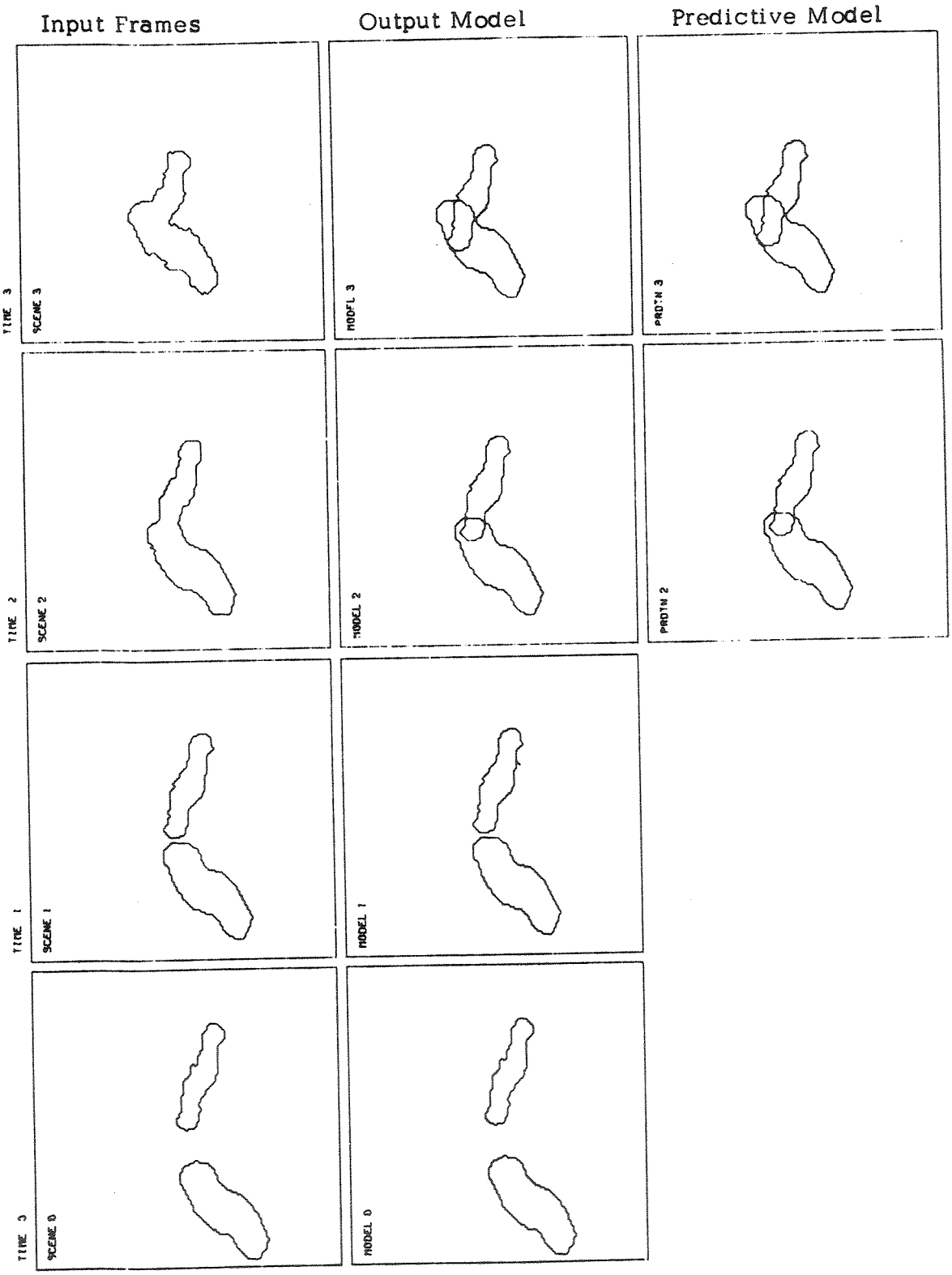


Figure 4. Image-Tracking Using a Predictive Model.

to track as many as four "reflective markers" as they moved through the scene. Although the motion analysis system is minimal and the marker identifying procedure is rather simple, this system is a first step at the hardware level to correlate information from two cameras as well as from a sequence of frames.

In contrast to the two systems above, Nevatia [41] proposes that the series of frames in the dynamic scene taken from a moving camera be used to simulate a stereo camera system in order to derive range information. The idea is that the small differences between consecutive frames in the series would allow common areas to be matched by cross-correlation or normalized mean-square-difference techniques; the range information would then be computed from the large differences between common areas (matched through intermediate frames) of widely separated frames of the series. The major problem with this system is that it gives no basis for determining the relative camera positions in the two frames used by the range-finding process. In the author's words, "for an actual moving observer, the camera transforms may be difficult to obtain and could be a major source of errors." For a discussion of the problem in an orbital photography system see Smith and Phillips [42].

2.3 Motion Analysis

In the following section we will discuss several systems whose complexity and degree of detailed analysis are at the level of the "attentive"

process, which we described in the introduction. Again, the first system we shall discuss was developed to measure cloud movements.

The motion analysis described by Endlich et al [43] is carried out on overlapping systematic sections (120 x 120 pixels using every fourth point) taken from a satellite photograph after each section has passed through two preprocessing phases. Here the sections in both pictures are taken to represent the same geographical areas, while the overlap is used to connect the sections within a photograph and detect motions across a section's boundary. The first of the preprocesses extracts the clouds from the background: this is done by a simple thresholding on the gray-scale values. The photographs are digitized into sixteen (0-15) levels of brightness,and the threshold is set at six; thus every point having a brightness less than six is declared to be background, while all the rest are considered to be clouds. This process yields points in a three-dimensional space over the x direction, the y direction,and the brightness level; the second preprocess called ISODATA for Iterative Self-Organizing Data Analysis, works on these points. ISODATA is a multivariate clustering technique [44] which detects clusters and yields their center points: these points are called the "brightness centers", and the motion analysis is performed on them.

The actual motion analysis is an iterative procedure to pair brightness centers in a section of the first picture to centers in a section of the second picture such that the pairings indicate the most consistent motion vectors. For this procedure $\Delta x_{ij} = x_j - x_i$, $\Delta y_{ij} = y_j - y_i$, and $\Delta B_{ij} = B_j - B_i$

are calculated for each ij, where i ranges over the indices of the centers of the first photograph and j ranges over the indices of the centers of the second photograph. From these values a "fitting function" F is formed such that

$$F(ij) = ((\Delta x_{ij} - X)^2 + (\Delta y_{ij} - Y)^2 + (\Delta B_{ij})^2)^{\frac{1}{2}} .$$

For the first iteration of the motion procedure, X is set to be the median of the Δx_{ij} values and Y is set to be the median of the Δy_{ij} values. F is evaluated for each pair ij,and brightness center i is matched to brightness center j if F(ij) is close to zero. These pairings are then used to calculate the average horizontal velocity Δx and the average vertical velocity Δy for the given section. The remaining iterations are computed in a similar manner, except X is set to be Δx and Y is set to be Δy, with Δx and Δy computed by the immediately preceding iteration. The authors state that "three iterations gave stable results in pair-matching (and therefore also in the motion vectors) in all cases investigated." Although not every center is matched, this process yields a separate motion vector for each pair of brightness centers.

The presence of two or more banks of clouds having various velocities still presents a problem for this system. The authors state that "computed motions are permitted to vary within different portions of the region treated," and while the statement is true,it deserves further discussion. It is, of course, the degree to which the velocity vectors are permitted to vary that

is at question. The problem occurs at two levels, the first of which is recognized by the authors who state that "there are many cases where cloud motions change with altitude and this can cause ambiguity in computer processing when all clouds are lumped together, as they are by using brightness alone to describe them." The problem here is that separate centers cannot be generated for the interspersed points of the banks of clouds which occur at various altitudes. The use of infrared data is set forth as a solution to the problem at this level: "It appears that the addition of IR data to position and brightness measurements would permit ISODATA to give separate centers for clouds at different IR temperatures (i.e., altitudes)."

The second level at which this problem occurs is in the method used to make the pairings between centers in the two sections. The authors again seem to expect the solution to be found in the infrared data: "The motion program can be generalized to include an IR measurement, and should generate separate cloud motions for different altitudes." However, as we stated earlier, the motion program pairs brightness centers which yield the most consistent motion vectors; possible pairings which have a motion vector widely different from the other pairings are discarded. The authors even point out an example (see Fig. 5) in which brightness would have served to separate the clouds at different altitudes. Indeed, the brightness difference helped pair the centers of the dominant cloud bank, yet the two centers of the subordinate cloud bank are not paired. The reason for this is that the motion vector, the dotted line labeled i in Fig. 5, which would

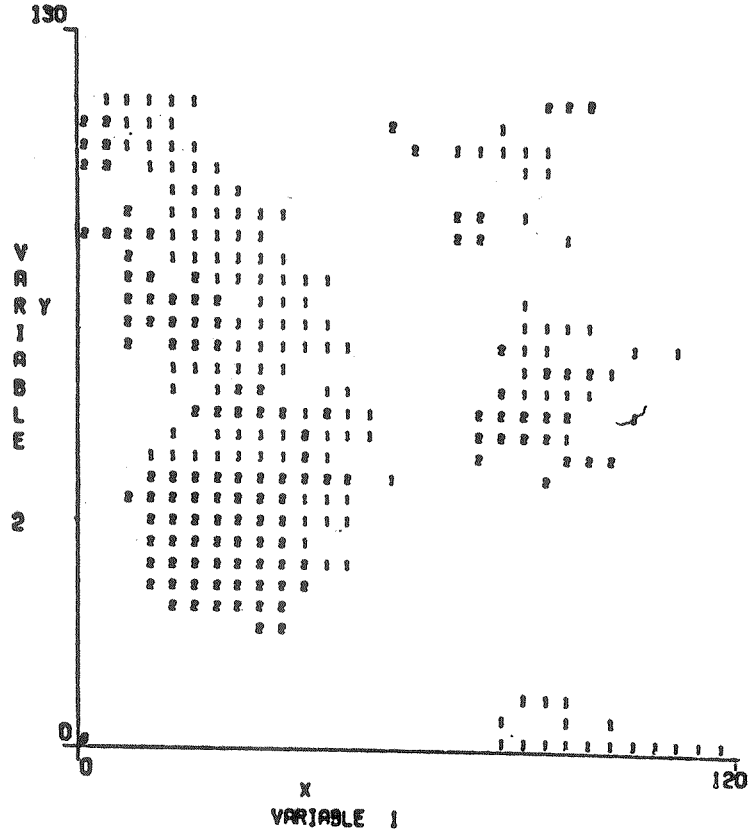


Figure 5a. The 1's mark the cloud points for the first frame, 2's mark the second frame.

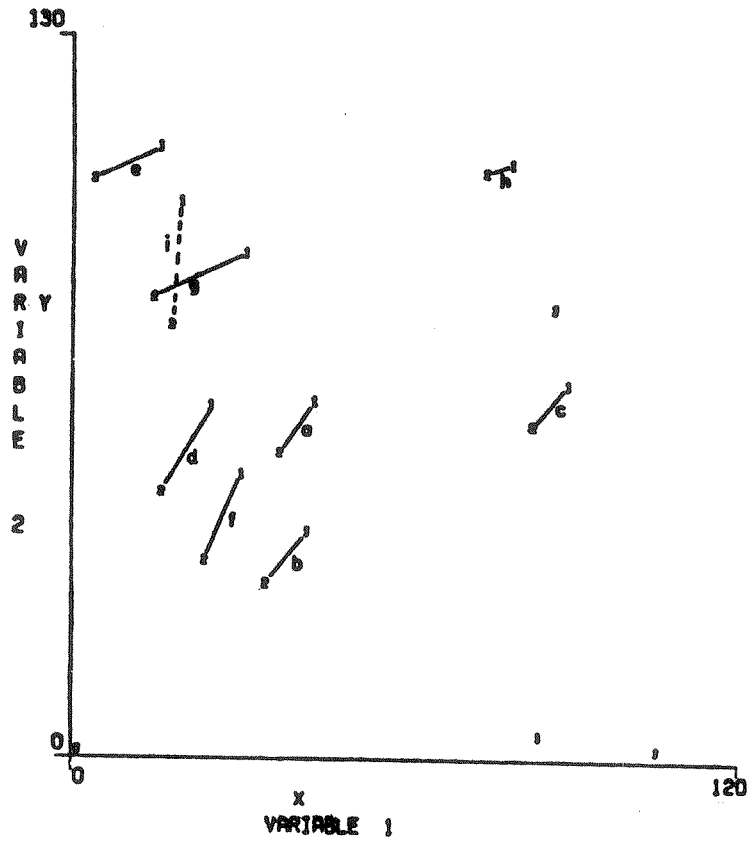


Figure 5b. The 1's mark the brightness centers for the cloud points of the first frame in Fig. 5a, 2's mark the second frame. The lines connect matched centers.

Figure 5. Example of Cloud Motion Analysis by Centroid Matching.

have resulted from pairing the two centers would have been almost perpendicular to all the other accepted motion vectors (i.e., the value of F at the two centers would have been large for all iterations of the motion program).

While detecting the separate motions of occluding cloud banks remains a problem for this system, the use of brightness centers is an appropriate method for describing cloud images, for two reasons: the large reduction in the amount of data required to represent the photographs; and the ability of a center to easily represent the amorphous nature of a cloud image.

The system discussed by Aggarwal and Duda [17], and Peterman [45] does not actually use machine-sensed data, rather the input is simulated by using software-generated two-dimensional scenes. The scenes are allowed to contain arbitrarily complex opaque rigid polygonal figures (possibly containing holes) moving with various translational and rotational velocities. The input is actually the spatial coordinates of the visible vertices of the perturbed parallel projection of each frame. The projections are perturbed in that a preset amount of additive noise is introduced at the coordinates of the vertices. A vertex is visible unless it is occluded by another polygon, in which case new vertices are generated at the points where the boundaries of the polygons intersect. Thus, there are two types of vertices in this problem domain: the first type includes the actual vertices of the polygons, while the second type includes the vertices generated by the intersection of occluding polygons. The authors refer to these as "real" and "false" vertices, respectively. It is one of the main tasks of this system to correctly classify

each of the input vertices into one of these two groups. This task is made manageable by requiring that the polygons be rigid, in which case the angular measure of the "false" vertices will change between frames, while the angular measure of the "real" vertices will vary only with the additive noise. The change at the "false" vertices is due to the difference in the rotational velocities of the occluding polygons. This problem is not solved by the requirement of rigid polygons, for two reasons: (a) in order to determine that angular change is occurring between frames at a vertex one must be able to identify the vertex in both frames; (b) if the occluding polygons do not exhibit any relative rotational velocity, then the angular measure of the "false" vertices will not be changing.

The task is actually solved by making one further observation about the problem domain and one further restriction on the input scenes: the observation is that an acute-angled vertex cannot be generated by the intersection of two polygons (i.e., cannot be "false"). This means that any acute-angled vertex must be a "real" vertex of some actual polygon. Thus, if the polygons in the scene are reasonably heterogenous, then a polygon in one frame can be identified in the next frame by searching for a suitable number of matches to its "real" vertices; but, the entire polygon may not be matched by this process because it is the union of two or more actual polygons and contains "false" vertices. At this point, the authors use a final restriction, which is that at most one "real" vertex may become occluded or become visible between any two consecutive frames. The implications of

this restriction are that all possible "false" vertices can be classified into six groups based on the difference in the number of acute angles between the first frame and the second frame, as well as the difference in the number of obtuse angles. This classification scheme allows special procedures to be written to identify the "false" vertices in each case.

Throughout the processing of a dynamic scene,this system forms and continually updates a model for each actual polygon encountered. Thus, if an apparent polygon in a particular frame is the union of several actual polygons,then a model will be associated with the apparent polygon for each of the constituent actual polygons. The association is through the visible "real" vertices of the actual polygon. The authors point out that these models not only allow the system to track occluding polygons, but also allow the system to generate a complete description of actual polygons having seen only a series of partial views of it. Figure 6 contains every other frame of an example from Ref. 19, exhibiting the analysis of complex occluding images.

The final paper to be included in this survey (Greaves [46]) reports on a rather novel system which is used to analyze the movements of micro-organisms. The dynamic scenes analyzed by this highly interactive system are composed of frames taken from a video image of a deep well slide as viewed through a microscope. A major assumption of the system is that the organisms viewed at a particular time in this manner can be represented by a single point. In order to form this representation and to remove noise and other unwanted features, the interactive user sets a threshold to be used on

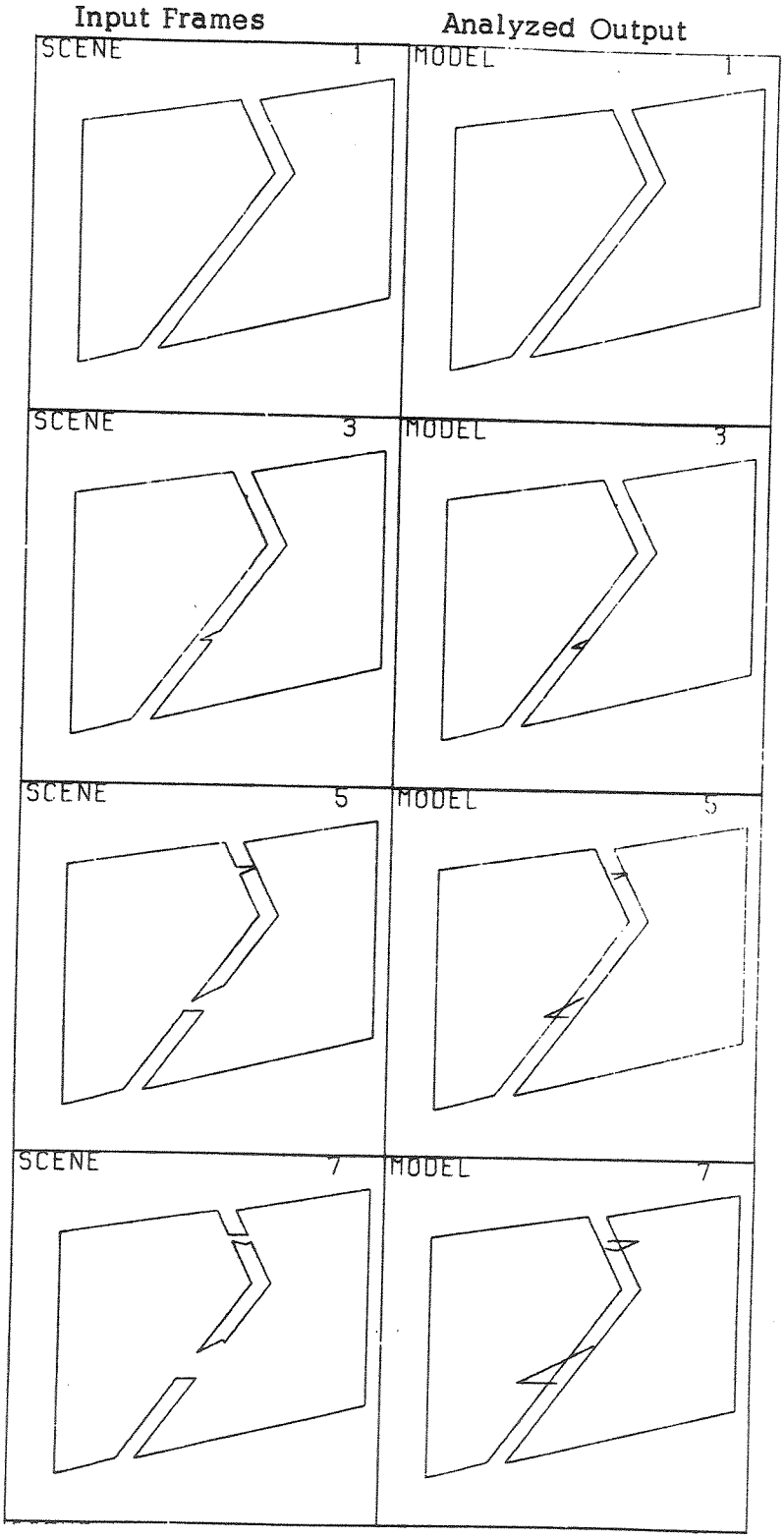


Figure 6. Example of the Analysis of a Complex Image Formed by Occlusion of Objects.

the intensity values of the frame pixels. If the intensity value of a pixel passes this threshold, then its x,y coordinates are recorded; then a simple clustering procedure groups the recorded points and replaces all the points of a cluster by the centroid of that group. Thus, each frame is represented by a list containing the x,y coordinates computed for the centroids of the clusters detected in the threshold video image.

The next step is to form the "bugpaths". A single "bugpath" is the list of centroids which represent a particular organism at each instance of time (i.e., one in each frame). To form a "bugpath", the centroid representing a "bug" in one frame must be matched to the representative centroid in the adjacent frames. A particular centroid in one frame is matched to another in the next frame by a simple smallest-Euclidean-distance criterion. When all the "bugpaths" have been formed, the dynamic scene is completely described by two sets of lists: the elements of the first set are lists which range through the spatial domain, but each list is fixed in time by a particular frame; the second set contains lists which range through time as well as space, but each is associated with a particular "bug". From this data structure an entire repertoire of velocity (and in this case behavioral) analysis functions can readily be applied, and the majority of the paper involves the description of these functions.

3. A NEW APPROACH

3.1 Acquisition of the Images

The investigation undertaken here has been to analyze a sequence of frames taken from an image-dissector camera. The camera is controlled by an XDS930 computer and responds in 32 levels of gray, at 256 x 256 pixels. The sequence is formed by directing the camera at a homogenously dark background upon which white planar objects move with various velocities. Due to the time required by the camera to scan the scene, each frame is processed as a still shot, then the objects are moved and another frame is scanned. It is then assumed that the frames represent systematic samples of the visual scene, with a constant time interval between consecutive frames. At this stage each frame is passed through a preprocessor which locates the edges of the objects [37]. This preprocessor yields a 112 x 112 binary map of each frame, with edges marked by a one and all other points marked by a zero.

The fact that white planar objects are used has several implications on the output of the edge-finder: first, the only edges in a frame are the boundaries of the objects; secondly, the scale of the images remains constant because the objects move in a single plane perpendicular to the camera's line of sight; finally, when two or more objects overlap,the boundary between the objects is not discernable to the camera, thus the overlapping objects appear as a single object. These facts along with

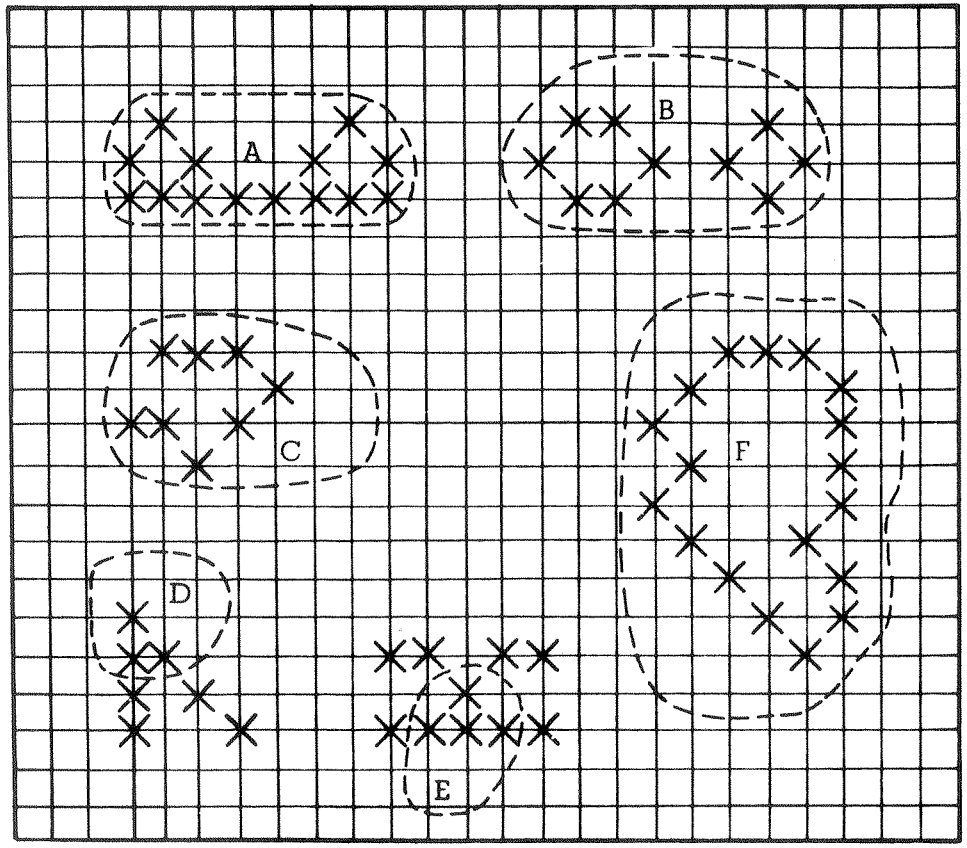
the properties of the preprocessor, insure that the binary images of the objects are closed connected chains of points.

To understand that description, consider the following definitions for grid points of the binary map: the neighbors of a given grid point are the eight grid points nearest the given point; a set of points is connected if for every pair of points in the set there exists a sequence of set points such that one of the pair points is a neighbor to the first element of the sequence, the other pair point is a neighbor to the last element of the sequence, and consecutive elements of the sequence are neighbors; a chain is a set of marked points such that every element of the set has no more than two marked neighbors[†]; a set of points is closed if every point has at least two neighbors in the set. Taken together, these definitions specify that the preprocessed image of an object is a set of marked points on the grid, such that every point has exactly two marked neighbors, and that by going from neighbor to neighbor within the set, one can get from any set point to any other set point. Figure 7 shows several sets exemplifying various combinations of these definitions.

Thus, the input to the system described in the remainder of this report is a sequence of frames each of which is a two-dimensional binary map

---

[†]The exception to this rule is corners. The points adjacent to the corner points may have more than two neighbors, however taken as a set (the corner point and its adjacent points), the points will not have more than two marked neighbors (see Fig. 7, sets D and E).



A - a closed connected set of points which is not a chain.

B - a closed chain which is not connected.

C - a connected chain which is not closed.

D - an allowable corner on a chain.

E - a corner which is not allowable on a chain.

F - a closed connected chain of points.

Figure 7. Binary Image Types.

representation of the field-of-view of the camera, such that each object viewed by the camera is represented by a closed connected chain of points in the map (see Fig. 8). The system immediately transforms this sequence of frames into a data base where each object image is described by two complementary methods. The data base itself is described in the Appendix. The first descriptive method is a simple list of the x,y coordinates of the object image points, starting with an arbitrary point then proceeding in a clockwise manner about the image until the original point is encountered again. This description can easily be obtained from the binary map and is mainly used as the basis of the second descriptive method, chain-coding.

3.2 Encoding the Images

Freeman [47] was one of the first to discuss in detail chain-coding of digital images. The chain code for an image is normally formed by defining an eight-direction orientation, choosing an arbitrary starting point on the image, and then proceeding in a clockwise manner from neighbor to neighbor recording the direction between neighbors according to the orientation: Fig. 9 gives an example of this. Freeman also noted that if one forms a graph of the chain code versus arc length from the starting point, then essentially horizontal lines in the graph will correspond to straight lines in the image. McKee [48] observed that if one slightly modifies the chain code so that its graph against arc length is "continuous", then circular arcs of the image result in straight graph lines of slope proportional

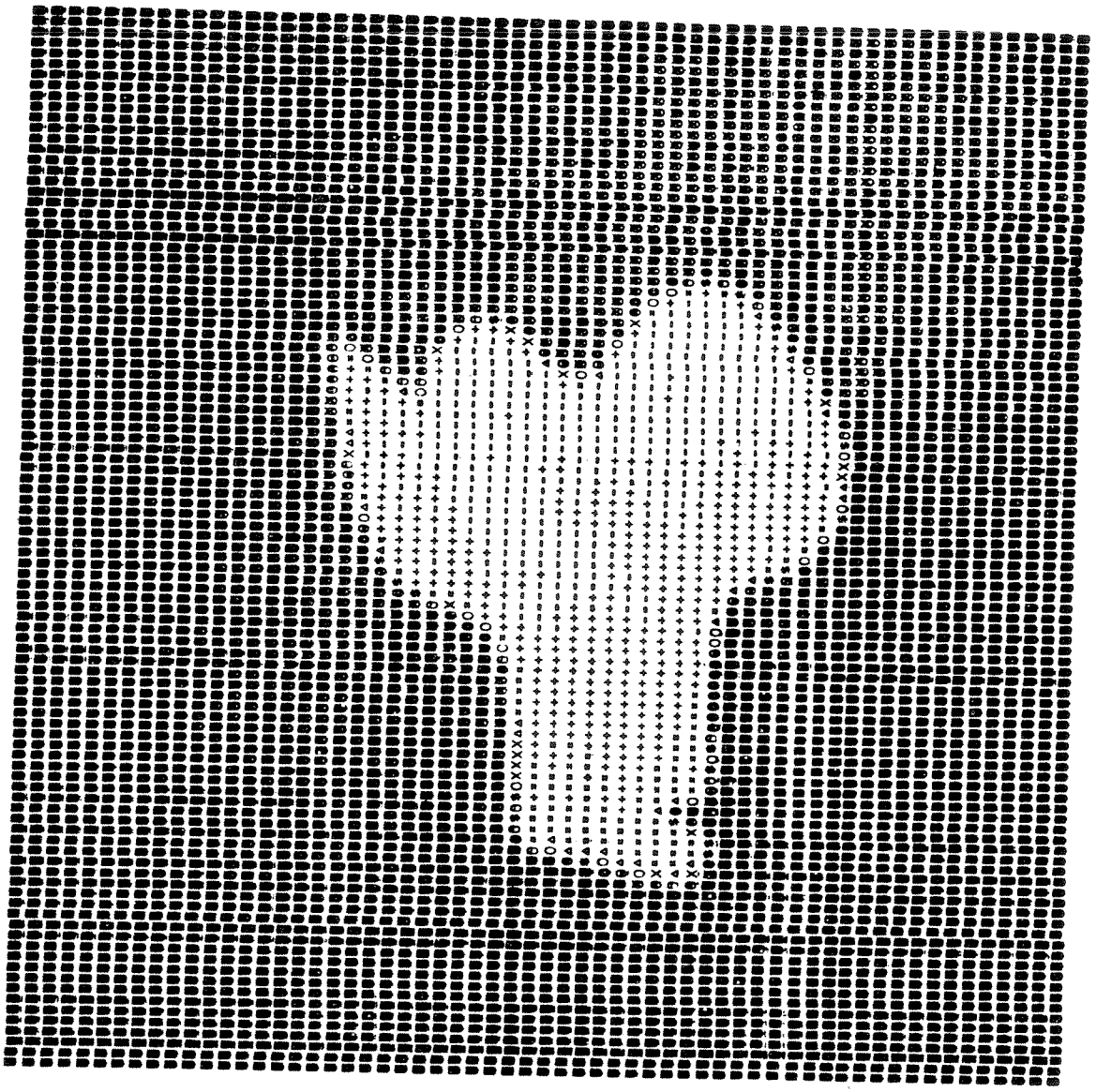


Figure 8a. Gray-Scale Overprint of Input Image.

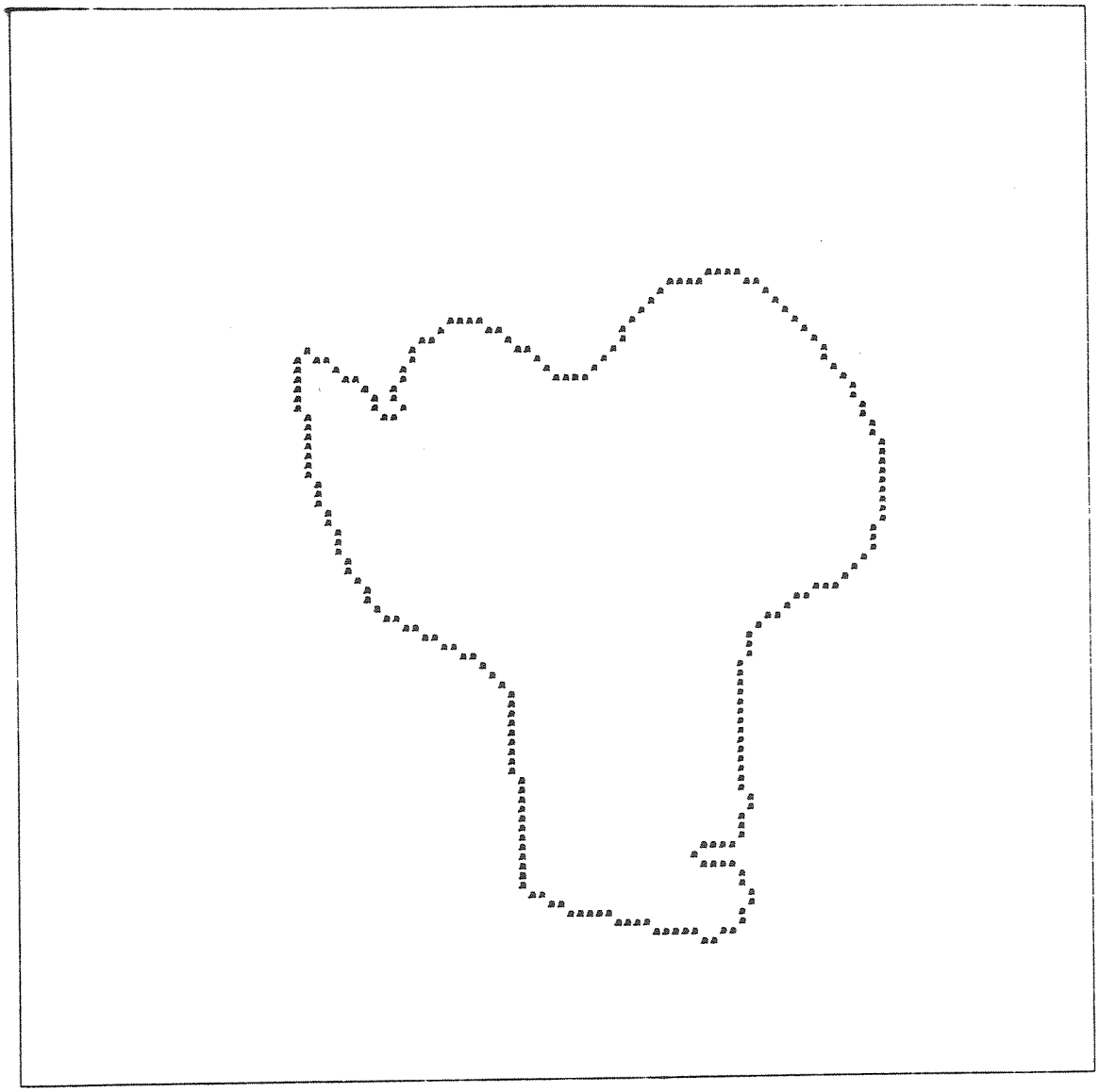
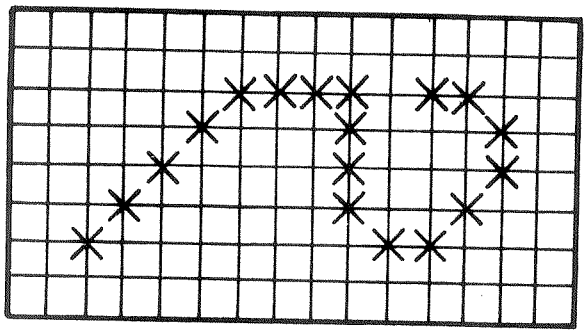
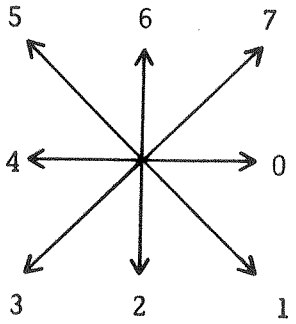


Figure 8b. Binary Map from the Image of Fig. 8a.



input image



given orientation

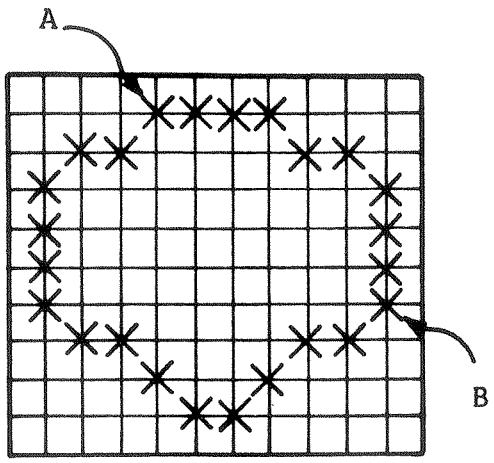
77770002221077654
normal chain code starting
from the left-most point

7777888 10 10 10 9877654
McKee's "continuous" chain code

Figure 9. Chain Code Example.

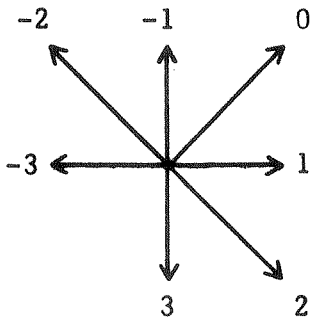
to the curvature of the arc. This immediately suggests that one could represent an arbitrary curve with a series of straight line segments and circular arcs by fitting lines to the modified chain-code graph. In this way, McKee described images by a list of lines where each line is given by its starting point, length, and slope as found in the chain-code graph.

Here, instead of obtaining the code and then modifying it, an equivalent method is used whereby the total angle subtended since the starting point is graphed against arc length. To form this graph an arbitrary starting point is chosen, the counter for the total subtended angle is set to zero, and the image is traced in a clockwise manner. For each point, a temporary orientation is defined so that the 0 direction is the same as the direction of the vector from the given point's counter-clockwise neighbor to the point itself. The directions to the right of 0 are the positive directions 1, 2, and 3, while the directions to the left are negative, -1, -2, and -3; of course, since the images are closed curves the exactly opposite direction is not possible. Now, the direction of the given point's clockwise neighbor is determined according to this temporarily defined orientation. This direction, the discrete angular change, is added to the total angle counter and this intermediate total is recorded as the code for the given point (see Fig. 10). This process is continued until the starting point is encountered again. It should be noted that since these closed curves are being transversed in a clockwise manner and the right-side directions are positive, the final value will always be 8; which is to say that the image subtends

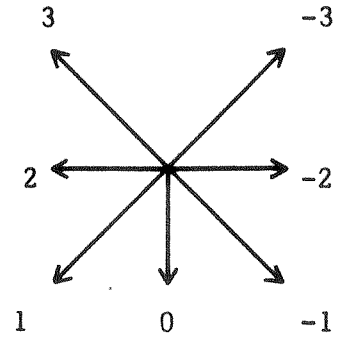


input image

1112123334544566567778 98
subtended angle code derived
from input starting at point A.



orientation at point A



orientation at point B

Figure 10. Example of Subtended Angle Chain Code.

in total an angle of 2π radians.

The rectilinear nature of the grid causes this code to be distorted because the arc length between diagonal neighbors is $\sqrt{2}$ units, while other neighbors are one unit apart. This distortion can be reasonably rectified by specifying that the distance between diagonal neighbors is three units, while other neighbors are two units apart: this is easily done by repeating the code for points whose clockwise neighbor is on a diagonal three times, while repeating the other codes only twice.

The graph of the code versus arc length would now be ready for the line-fitting process except that the process would be greatly affected by both the arbitrary choice of the starting point and the noisy nature of the images. The effect of the starting point choice on the line-fitting process can be nullified by using the graph formed by completely traversing the image twice. Equivalently, the codes for the first traversal can be repeated while adding eight to each code value; then the resulting lines at the beginning and end of the graph, which are not repeated in the body of the graph, can be discarded along with the unnecessary repetitions. The unrepeated lines in the middle of the graph are the correct lines representing the arcs in the vicinity of the starting point.

The effects of noise can be minimized by averaging each code value over a nine-point window, which includes the given code value and the four values on either side of it. With these modifications, the graph is finally ready to have straight lines fitted to it. A standard mean-square-error

technique is used to actually fit the straight lines which are referred to here as code lines. The erroneous and unnecessary code lines are then discarded, as described above, and the final result is a description of the image in terms of a list of straight lines in the subtended angle versus arc length domain.

Careful attention must be paid throughout this process to retain the connections between the spatial coordinates of the image points and their corresponding code points. This system does that, and when each code line is inserted into its respective object description in the data base, it is associated with the spatial description by pointers to the first and last corresponding image points in the originally formed list of coordinates. The ability to retain this connection is a major advantage of this descriptive method. The length and slope of the code lines, as well as the order in which the code lines appear, efficiently and effectively contain the necessary information for discimination of shape regardless of translation or rotation.‡ The coordinate list complements this by retaining the information for spatial discrimination, such as area, perimeter, and location of the image. But most importantly, the combination of both descriptions conveniently segments an image, using features intrinsic to that image. Figure 11

---

‡ This statement must be qualified because translations of integral units and rotations of integer multiple of 90° are the only transformations on a discrete grid which do not modify the shape of the images on that grid. The differences in the shape of an image due to translation and rotation are considered part of the noise inherent in the digitizing process.

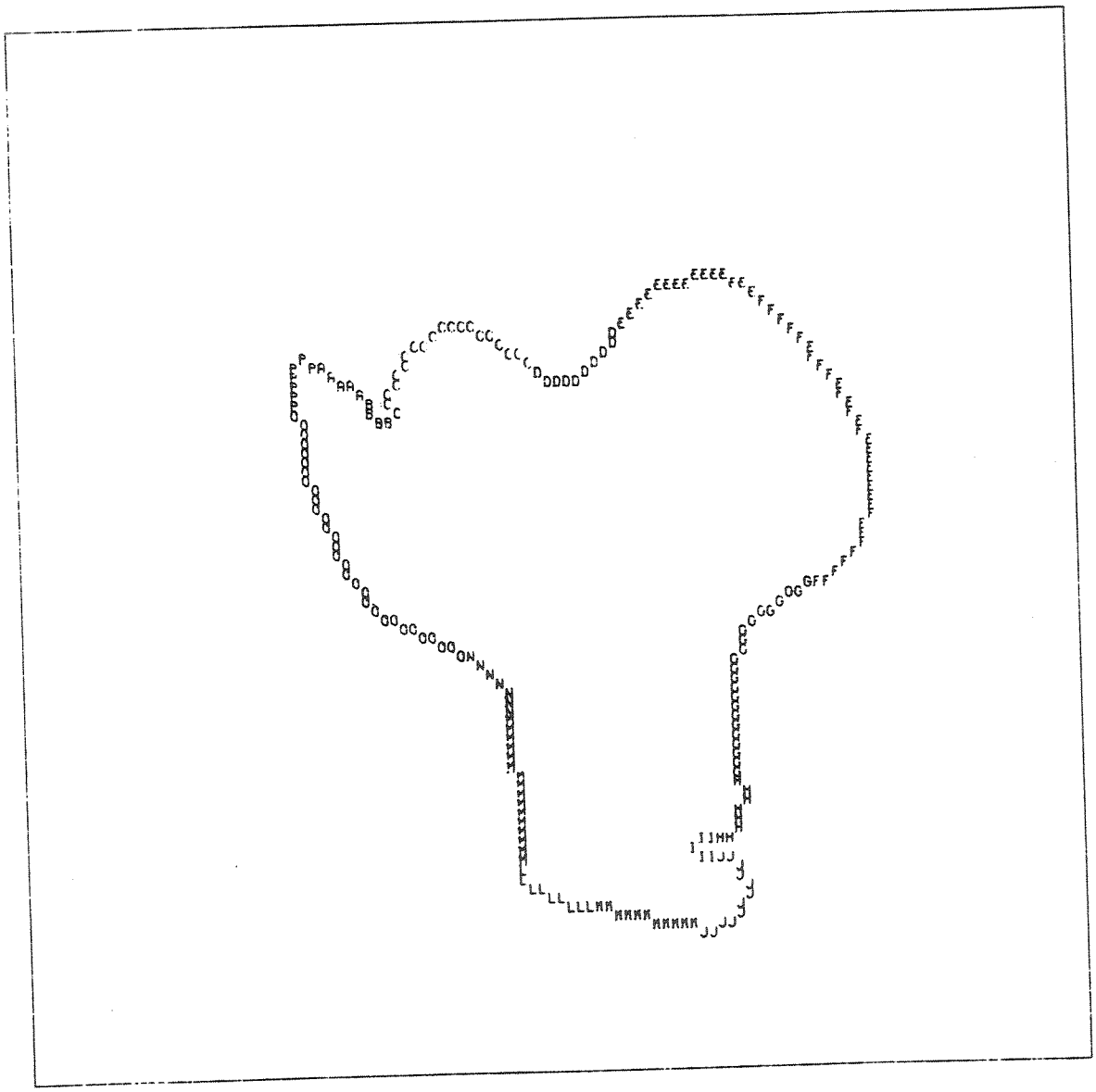


Figure 11. Image Segmented into its Intrinsic Features, i.e., its Code Lines.

shows the derived features of an example frame.

Two images described by a list of such code lines can be compared for similarity of shape in the following manner. Choose one of the images and designate one of its code lines as the beginning. Then using an initial value of zero, the slope of the given line, and the code line's length, generate the first section of the code graph for this image. The remaining sections of the graph are generated by taking the next code line in the list for this image and using its slope and length to extend the graph from the point that the last section ended. When all the code lines for this image have been used, then form another graph for the other image also starting at zero. The area between these two graphs is a good measure of the similarity of the shape of the two images as oriented to their starting points. In this case, the starting point of each image is the spatial point which corresponds to the start of the code line chosen to begin the graph of the respective image. Two images are oriented to their starting points if the images are translated so that their starting points are at the origin and are rotated about the origin so that the "tangents" of the images at their starting points are the same. Here the "tangent" of an image at a given point is the vector from the counter-clockwise neighbor of the given point to the point itself. Thus,comparing the code graphs is equivalent to physically orienting the images to their starting points and visually comparing the similarity of their shapes.

It is clear that this form of shape comparison is severely limited by

the choice of starting line, i.e., the beginning code line on the generated graph. Two images could have the same shape, yet if different starting lines were specified for each image then the area between the generated graphs could be substantial. Intuitively, if two copies of the same image were oriented to different starting points,then even a visual comparison would declare their shapes, as oriented, to be different. Circles and certain pairs of starting points on other axially symmetric images would visually be judged to be similar; but in these cases the area between the code graphs would also be small. McKee [48] presents this problem, as well as discussing the problem of scaling, in somewhat different settings not dealt with by this system. The next section presents a discussion of how this system solves the problem of arbitrarily choosing the starting line for shape comparison.

3.3 Segment Matching

At this point, the system has taken the input frames, extracted the object images of each frame, and then analyzed each object image into its components, i.e., its straight lines and circular arcs. Now, from these analyzed images the system must attempt to synthesize representations of the actual objects in the scene. An actual object is to be distinguished from an apparent object in that the apparent object is the image formed by the occlusion of two or more actual objects. The first step in this synthesis is to find features (i.e., edge segments) common to consecutive frames. The second step, discussed in the next section, is to group these features

according to common motion vectors.

The common features are detected on the basis of the shape of the images in the current and next frames. The terms "current" and "next" will be used throughout this discussion to refer to the first and second frames of any pair of consecutive frames. If a portion of an image in the current frame is found to match a portion of an image in the next frame, then they form an edge segment and are called the current aspect and the next aspect of that edge segment; but since each frame can contain several images, the system must make rough pairings of the images in the current frame to images in the next frame before the portions can be matched. The rough pairings are based on features global to the images, such as area, perimeter, and position of the centroids.

The paired images are then compared to find any common edge segments. Remember, the images have been analyzed into their components (i.e., code lines), so the system begins by finding code lines with similar slopes and lengths. The similarity measure is a relative difference defined for two numbers u and v by the following equation: $RD = (2*|u-v|)/|u+v|$. If the relative difference of the slopes of two code lines is less than 0.25 and the relative difference of their lengths is less than 0.5, then the components are declared similar. Two code lines are the maximally matched components of a set of matched code lines if the sum of their relative differences is minimal over the set. This measure gives preference to arcs with large curvature or long lengths, over either short flat arcs or straight lines.

The system goes on to look for groups of contiguous code lines of the current image which are similar to contiguous code lines in the next image. These groups are listed by the order of their maximally matched code lines; then the remaining singularly matched code lines are added to the end of the list under the same ordering. Starting with the first group of contiguous components, then progressing through the rest of the list, the system tries to "grow" common edge segments. The system "grows" the edge segments in the sense that it begins with a "seed" segment and generates a code graph for each aspect of the segment as the segment is extended in both directions around the image. The "seed" is the edge segment whose aspects are represented by the maximally matched components of the given group. The segment is first grown incrementally in the clockwise direction until either the shape similarity measure between the two aspects is greater than some preset threshold,or a previously grown edge is encountered; then the orientation is reversed and the segment is grown incrementally in the counter-clockwise direction. The shape similarity measure is the code-graph difference as described in the previous section.

Besides meeting certain criteria while the edge segment is being grown, the resulting segment must also pass a length and orientation test. If the edge segment passes these tests, it is accepted as a feature common to both images and recorded as such in a data base of common segments. A data base retains information about the matched code lines of the segments and their connections to the spatial coordinates list. A complete description

of this data base is given in the Appendix. Figure 12 shows two consecutive frames and the resulting common segments.

The image pairing, component matching, and common edge segment growing are continued until each image in the consecutive frames is appropriately matched to one or more images in the other frame. An object image must be allowed to match more than one other image because the current frame might contain an apparent object which will split into two or more of its constituent actual objects in the next frame; conversely, two actual objects of the current frame might occlude one another in the next frame, creating a single apparent object.

3.4 Motion Analysis

Having found the edge segments which are common to consecutive frames, the system is now ready to group the segments into object models according to motion measurements. The motion of a segment is determined by comparing the location and orientation of the segment's two aspects. Remember that the aspects of an edge segment are the image segments from the current and next frames, which were matched to form the edge segment. The horizontal, vertical, and angular velocities of a given edge segment are determined from the displacement of the next frame aspect, with respect to the current frame aspect. The noise inherent in the digitizing process, as well as the effect of different orientations on the binary grid, preclude the point-to-point mapping of the two aspects: this means that the displacement cannot be directly calculated from the list of coordinates of the aspects.

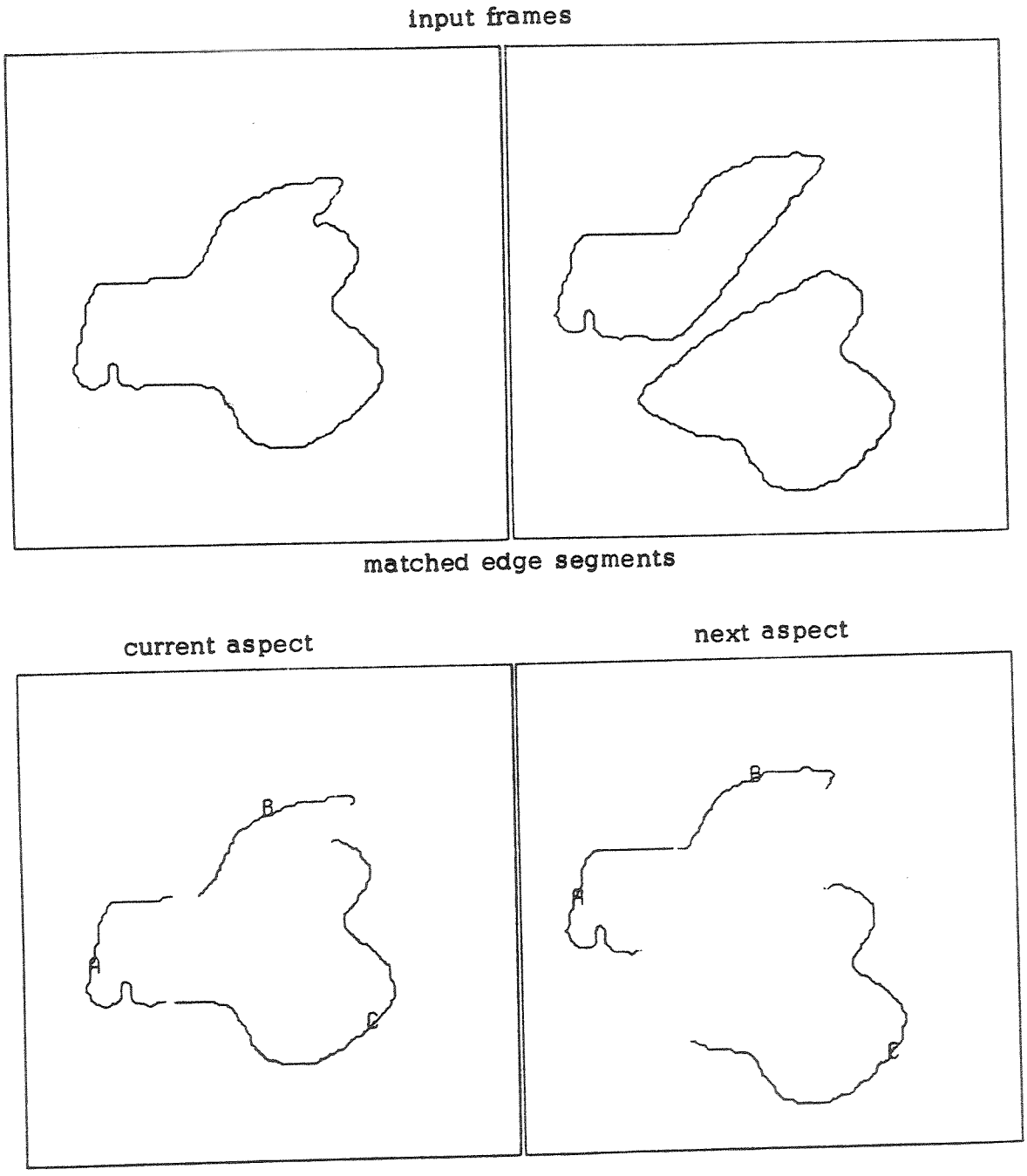


Figure 12. The Matched Edge Segments for a Pair of Consecutive Frames.

Instead, the system forms a triangular complex of points for each aspect of the given edge segment and calculates the displacement based on the location and orientation of these point complexes.

A triangular complex is formed, for a given aspect of an edge segment, by finding the first, last, and midpoint coordinates of the aspect. Connecting these points in the given order forms a triangle from which the system calculates the centroid of the triangle and the midpoints of the three sides. This process yields seven points which can be mapped to the similar seven points formed for the matching aspect. The system then computes the average horizontal displacement and the average vertical displacement for the mapped points. The angular velocity is estimated by forming six vectors from each triangle and averaging the angular change of the mapped vectors. The six vectors used are the three sides of the triangle and the three lines that connect the vertices to the centroid.

Each edge segment common to the current and next frames is processed in the above manner to derive estimates of the three velocities; then the system groups the segments into object models. Now it is clear that two edge segments with their current aspects taken from different object images of the current frame cannot be edges of the same actual object. Thus, for each pair of images which have been mapped together by the segment matching process, the following procedure is applied only to the set of edge segments that are common to a given pair of images. From such a set of edge segments a list of all possible distinct unordered pairs of

segments is formed. For each pair on the list, the velocity estimates are contrasted. If the velocity differences are greater than some preset thresholds, then the two segments definitely represent different actual objects; if the differences are less than some other thresholds, then the two segments definitely represent the same actual object. Finally, if the differences are between the thresholds, then the two segments possibly represent the same object. Each of the three velocities have separate thresholds, although the horizontal and vertical thresholds have the same value.

Essentially, this contrasting and thresholding procedure defines two mutually exclusive relations over the set of unordered pairs of edge segments: the "same object" relation; and the "different object" relation. The final object models are formed from the "equivalence classes" of the "same object" relation. An "equivalence class" for that relation is a subset of the original set of edge segments, such that for every segment in the class there is at least one other segment in the class, which has been designated as definitely representing the same object as the given segment. In addition, for every segment in the class, all segments of the original set which have the "same object" relation to the given segment must be elements of the class. To each such "equivalence class" is added any segment which has been designated as possibly representing the same object as some segment in the class, yet is not in the "different object" relation to any element of the class. Thus, the "same object" relation is used to form the object models, while the "different object" relation is used to clarify the ambiguous

in-between cases.

An object model is formed for each augmented "equivalence class", and the Appendix gives a complete description of this data structure. The motion analysis process given above is applied to the set of edge segments of each of the paired images until all of the images in the current frame have been analyzed. Figure 13 shows the edges of the object models of an example as analyzed by this process. It should be noted that the models are named according to information obtained from previous frames as well as current motion measurements. The name of an object model is determined by interrogating the image data base to determine if any of the image segments of the current aspect of the object model were matched to an image in the previous frame. This procedure works well, except when two occluding objects exhibit joint motion through a few frames, before having a relative velocity. In addition, if occluding objects do not exhibit a relative angular velocity, then the portion of the image where the actual object boundaries intersect may retain its shape through several frames.

In both of these cases, a previously supposed actual object is found to be only apparent and must be split into its constituents. The system checks for these cases by looking for current object models which have been given the same name. The system first assumes that if the naming procedure assigns the same name to two models, this is evidence that they should be parts of one model. With this assumption, the system again performs a motion analysis of the segments in the involved models, only under less

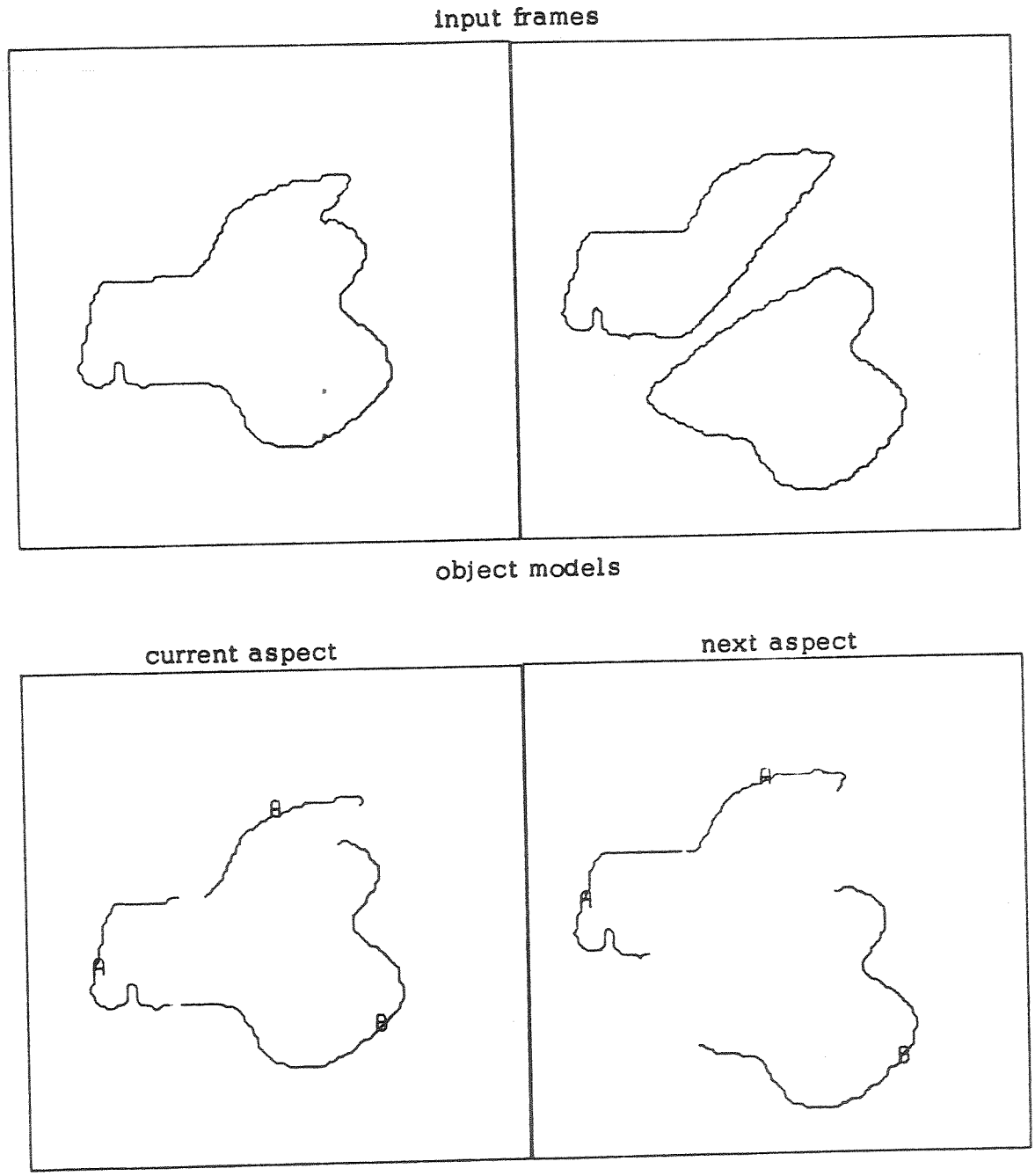


Figure 13. The Object Models Derived by Motion Analysis.

stringent thresholds for the same object designation. If the velocity differences of all of the segments in the models pass these thresholds,then the models are merged; otherwise, one of the models is arbitrarily given a new name.

When the entire motion analysis is completed, a list of the object models of the current frame will have been generated and the system can go on to the next frame. In this way, each frame becomes the next frame for the segment matching and motion analysis procedures, then it is analyzed directly as the current frame, and finally it becomes a past frame as part of the data base. The next section will discuss several examples which this system has analyzed.

3.5 Examples

Presented in this section are three example scenes which the system has analyzed into object models. The input frames are shown by plots reconstructed from the coordinate list of each frame. For each pair of consecutive frames the derived object models are shown below the input frame, with the edges of each model labeled by the system-defined name of the object model. It should be remembered that an object model is associated with two sets of image points, i.e., the current and next aspects of the edge segments of the object. So,for a pair of frames, the image points from the first frame form the current aspect and the image points from the second frame form the next aspect of the object model. To highlight this, the frames of each example scene are presented in pairs, with the object model aspects

below their respective input frames. Thus, the edge segments associated with the current frame, which are labeled by the same name as edges of the next frame, were first matched by shape and then grouped into object models by motion measurements.

The first example, Fig. 14, is a scene containing two occluding objects shown through three frames. The stationary central object is labeled A, while the rotating object is labeled B. The sections which could not be matched by shape are not plotted: these sections include small noisy edges, as well as the junction points of the occluding objects which are indeed changing shape.

The second example, Fig. 15, shows a large object uncovering a smaller object. Note that as the small object first appears (frames 2 and 3) the edge is changing too radically to be matched by shape, so no model is formed for the small object until 3 is the current frame.

We would like to present the final example, Fig. 16, in the spirit of the old adage, that many times one learns the most from his failures. Although the main portions of this example have been analyzed correctly, the three minor errors bring to light some of the inner workings of the system. Each of the three errors is the result of the system matching edge segments which are locally similar in shape. Remember that the images of the input frames are initially partitioned into intrinsic features by the shape description, i.e., into code lines. Then the "seed" partitions from which the edge segments are "grown" are the code lines which most

nearly match (in slope and length) a code line from the next frame. At this level the analysis is strictly on the basis of shape, so that if an input image contains several portions of similar shape the system may become confused and match the wrong portions.

This is precisely what has happened in this example. With frame 2 as the current input frame, the system mistakenly matches the edge segments to form object model B and object model C. New object models were formed because of the large velocity derived from comparing the location of the current aspect of each model to the location of its next aspect. With frame 3 as the current frame, the same problem occurs for object model B.

The errors also show the arbitrary nature of the object-naming process. When it is found necessary to divide the segments of a previously single object model into several new models (see Section 3.4), the new names are assigned arbitrarily. Thus, when the models for objects B and E of the current aspect of frame 2 were formed from the previous model B, either of them could have been given the name E. Although the naming procedure somewhat clouds the issue, one can still see from the example that the large object named A is consistently tracked throughout the scene as are the major portions of the central object, after they receive the new name. The smallest object undergoes extensive change, but at least some part of it is always tracked correctly through each frame.

The errors discussed above are primarily a result of the localness of the segment growing procedure. It seems that if some global

information could be brought into the decision, then the problem could be solved; however, it is not clear at this point what this global information might be. Such things as global orientation or connectivity of the segments are rendered ineffectual by the fact that the images are moving, merging, and splitting. It is clear that more work is needed on this problem, possibly making more extensive use of the information from previous frames when analyzing the current frame.

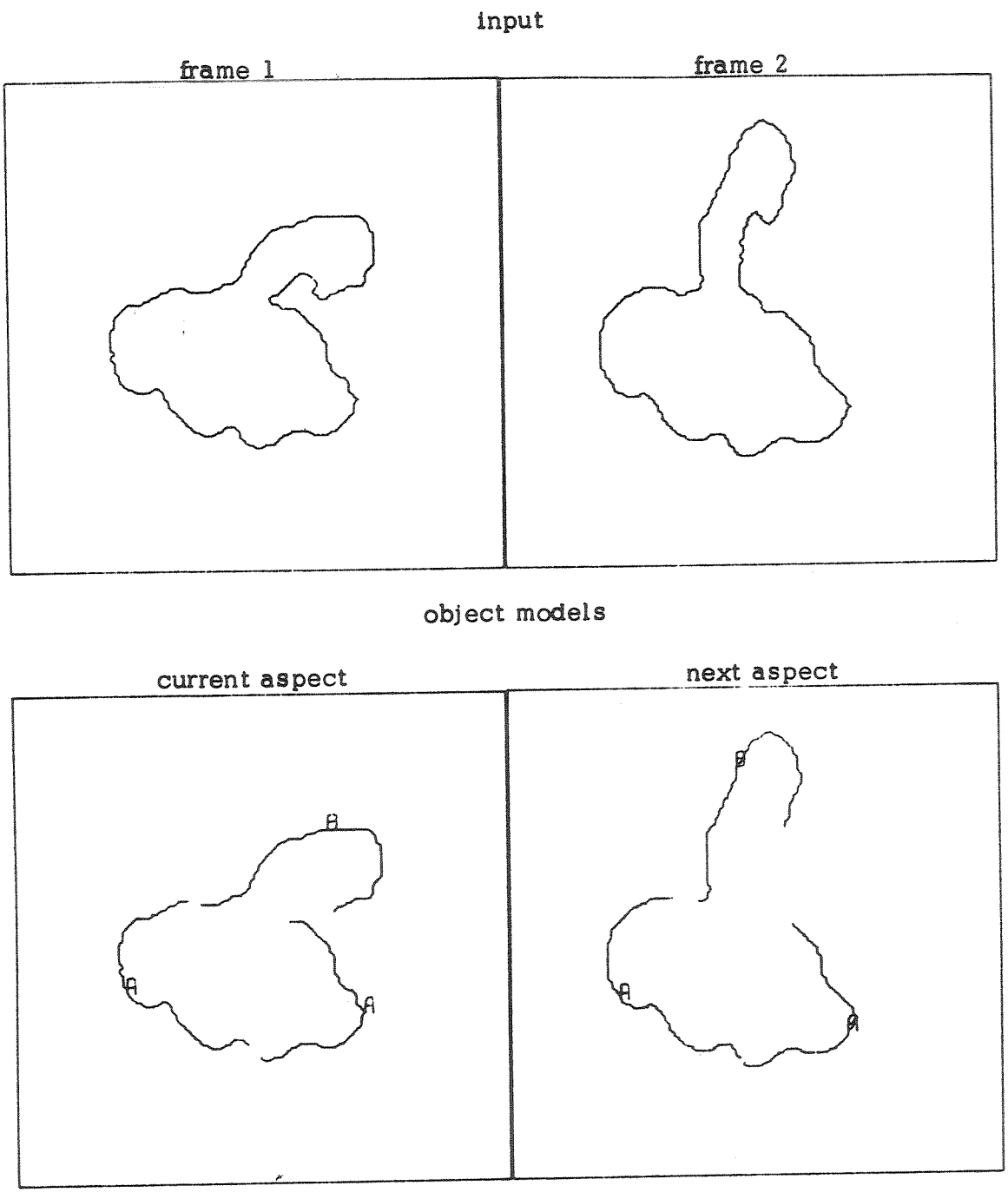


Figure 14. Example 1: Two Occluding Objects.

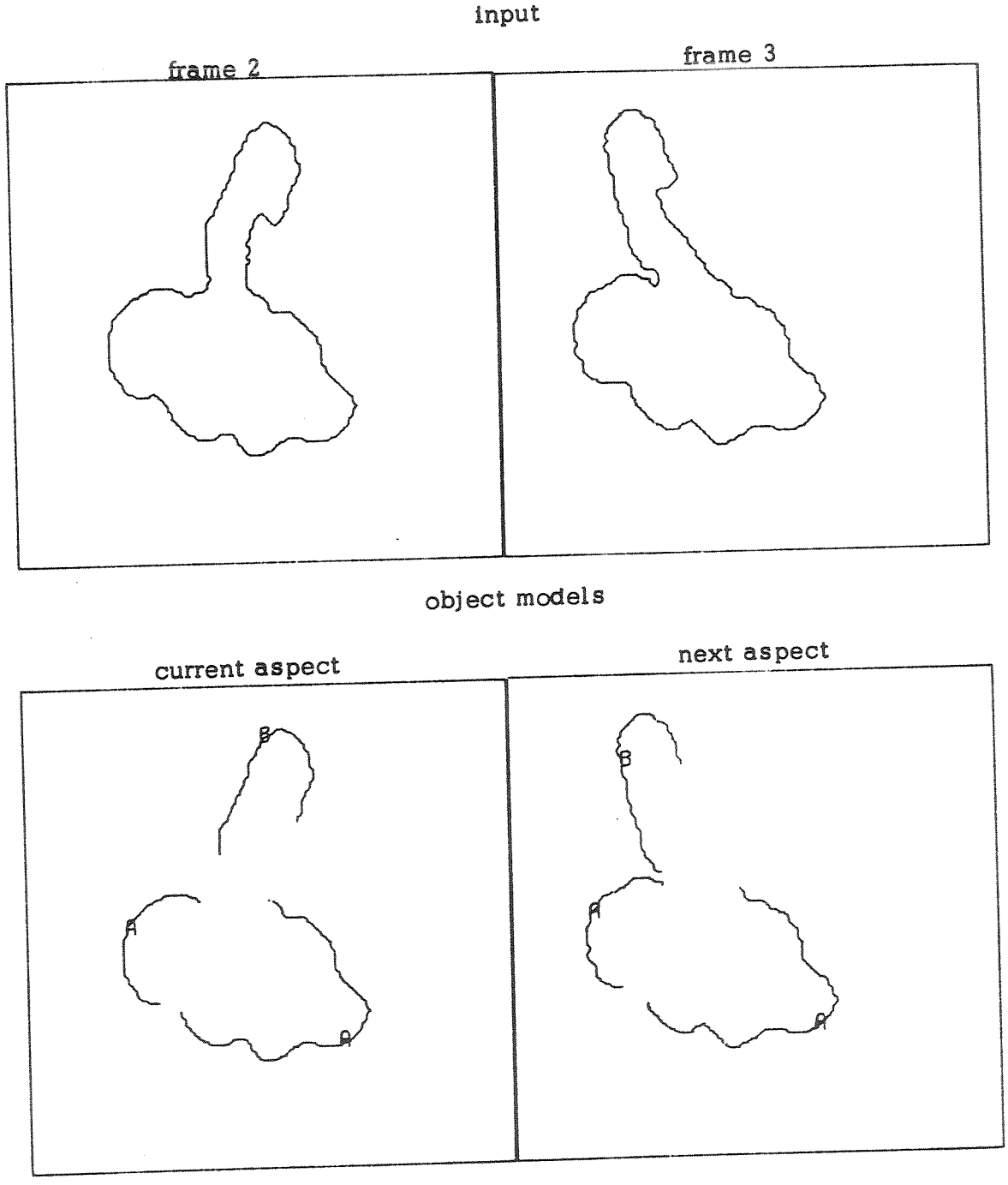


Figure 14. (continued)

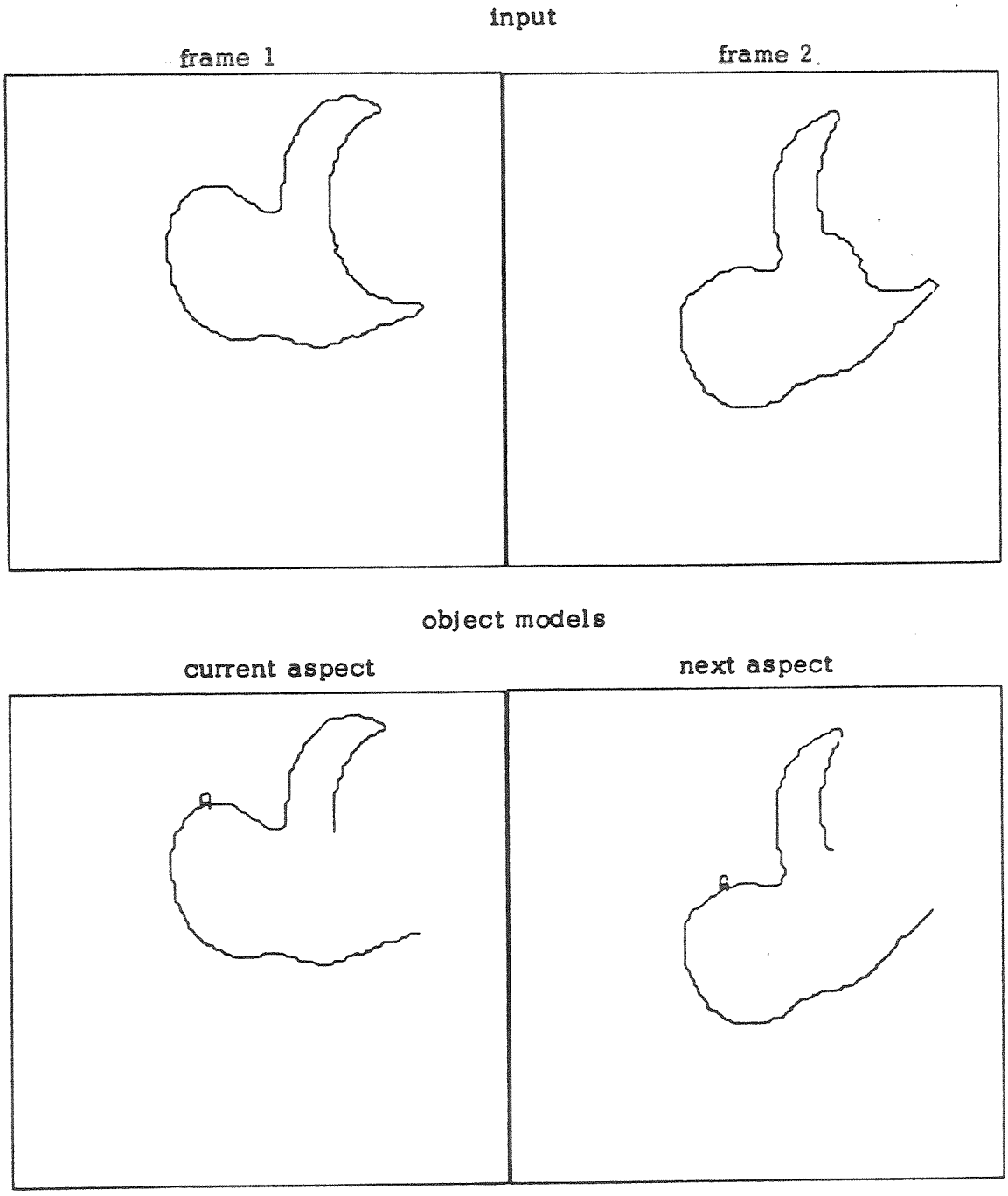


Figure 15. Example 2: Two Objects Separating.

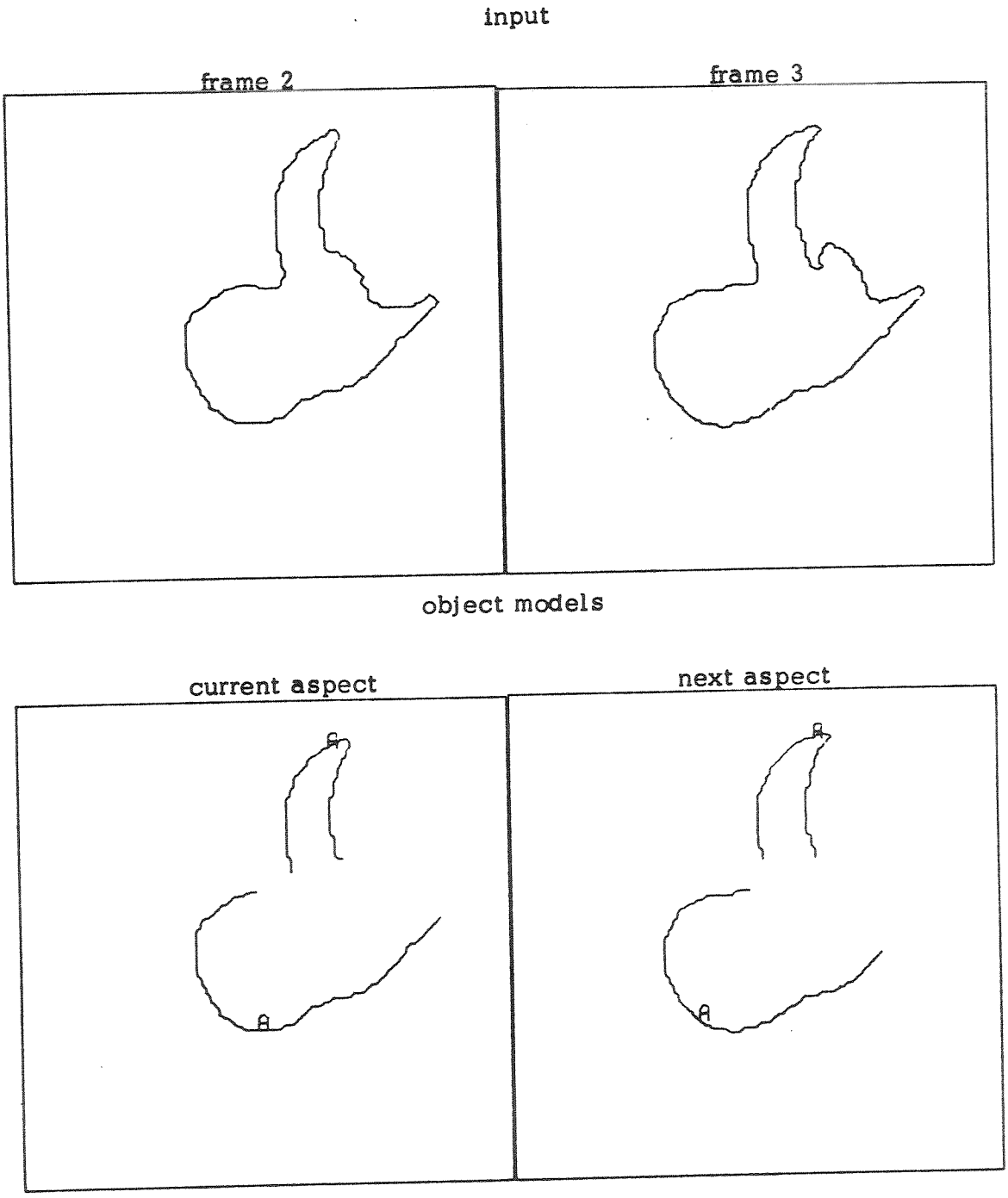


Figure 15. (continued)

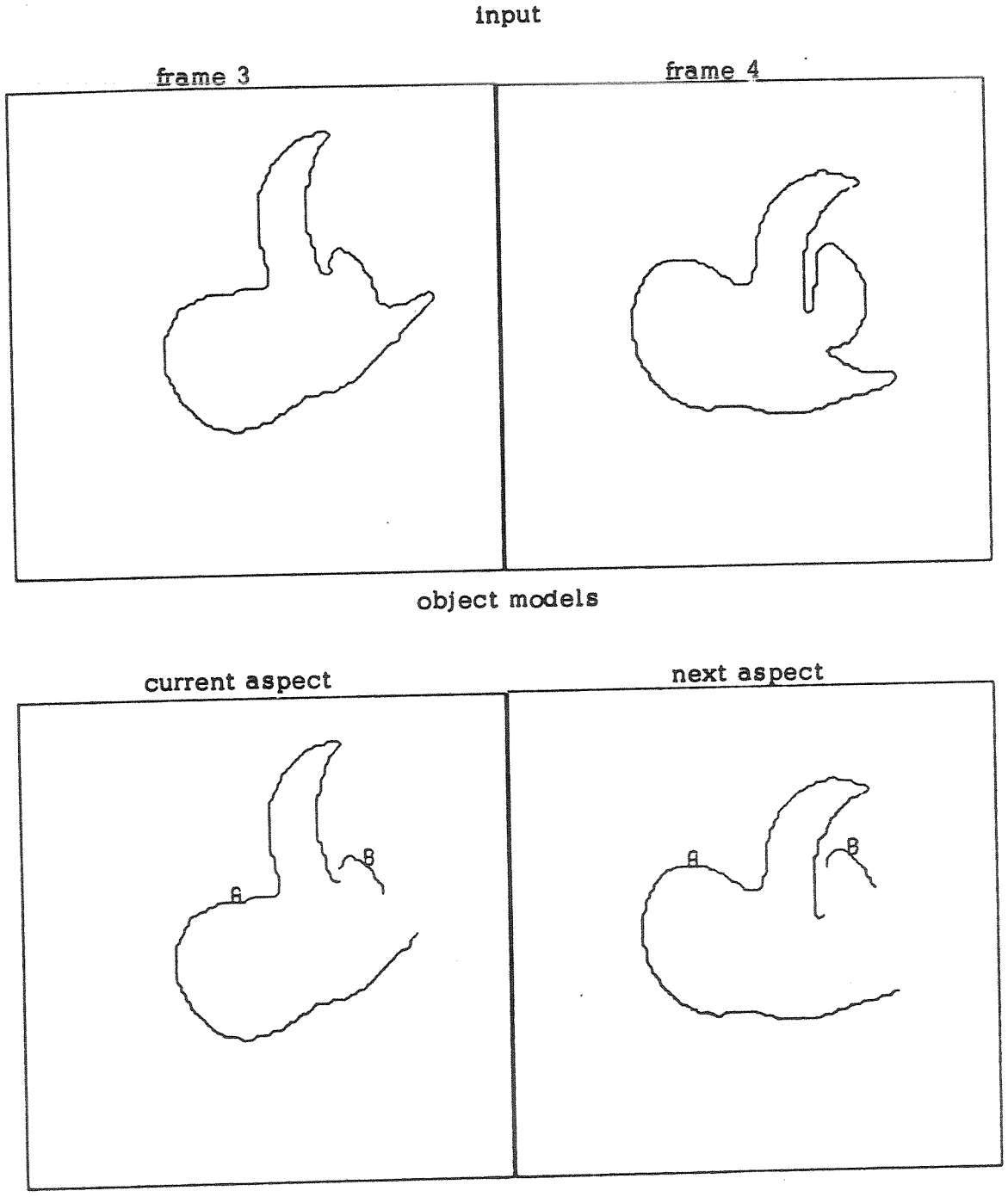


Figure 15. (continued)

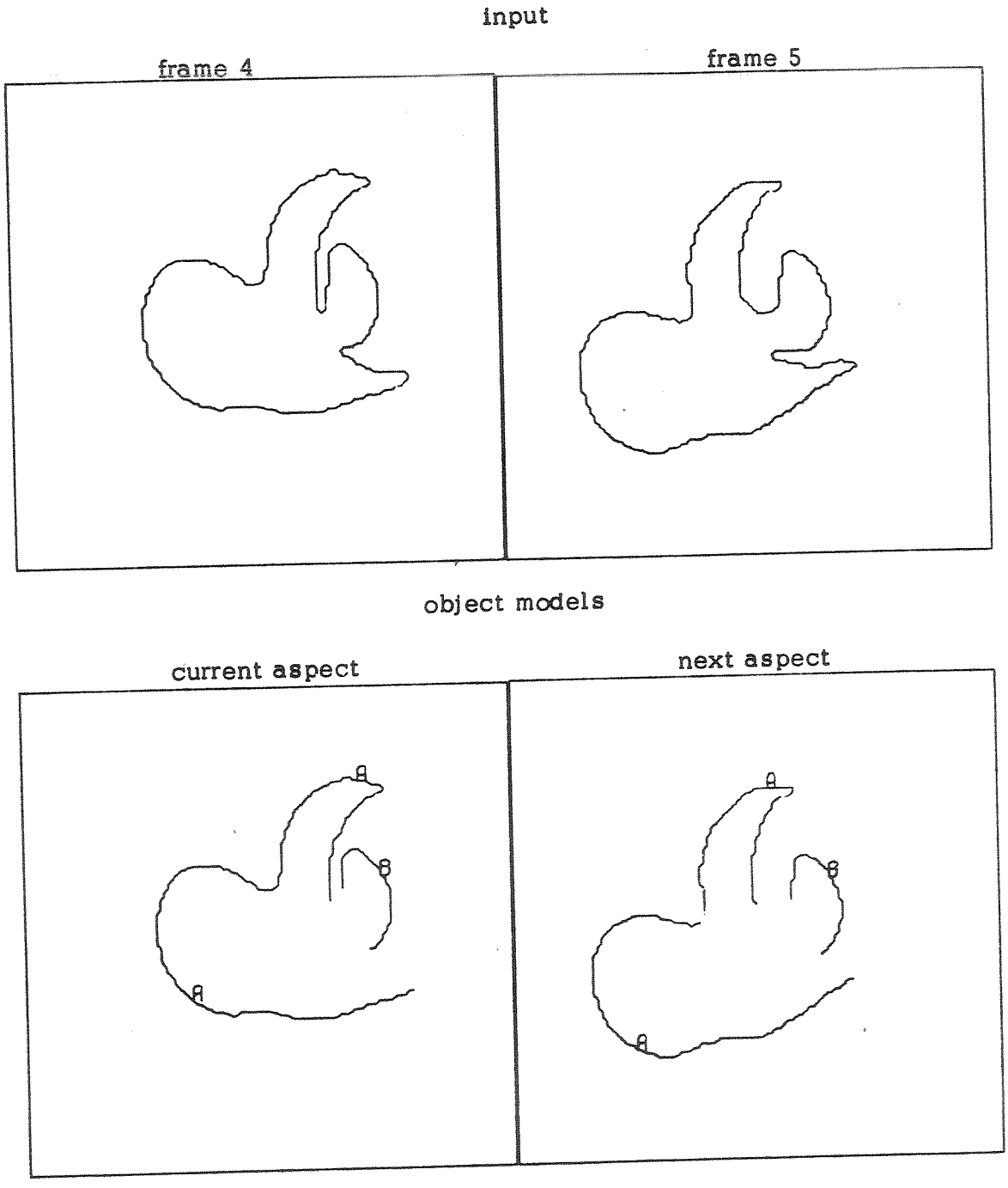


Figure 15. (continued)

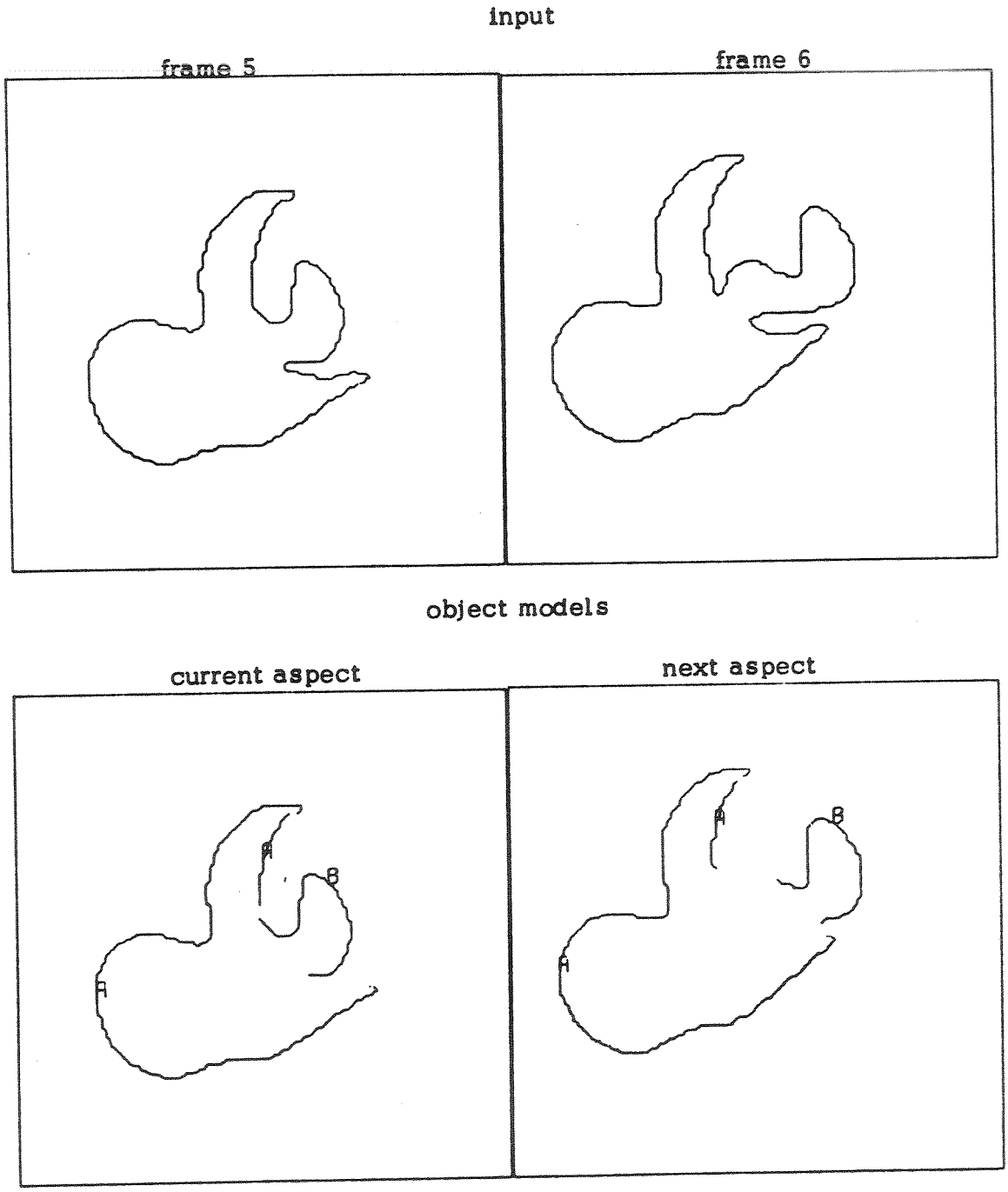


Figure 15. (continued)

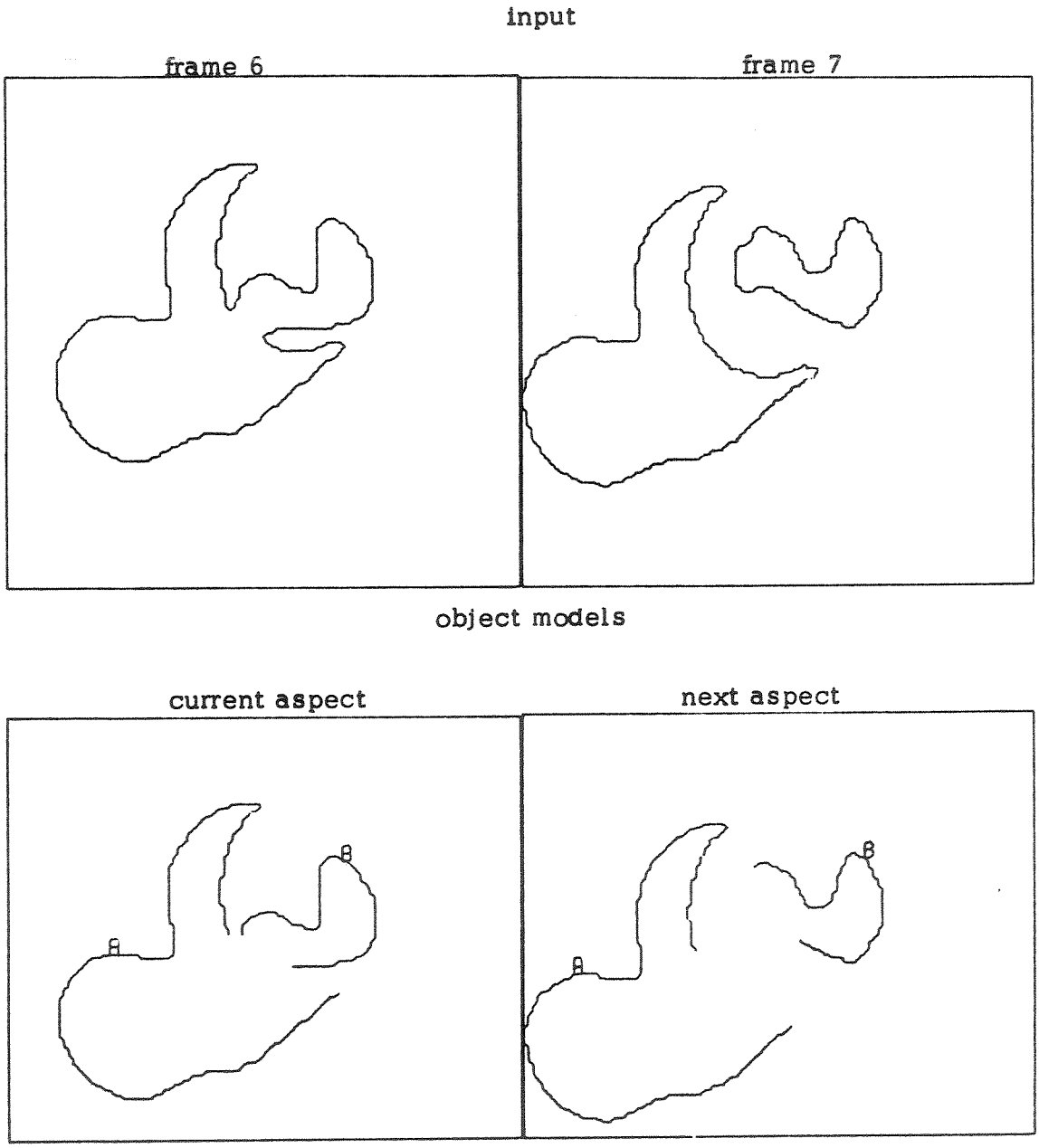


Figure 15. (continued)

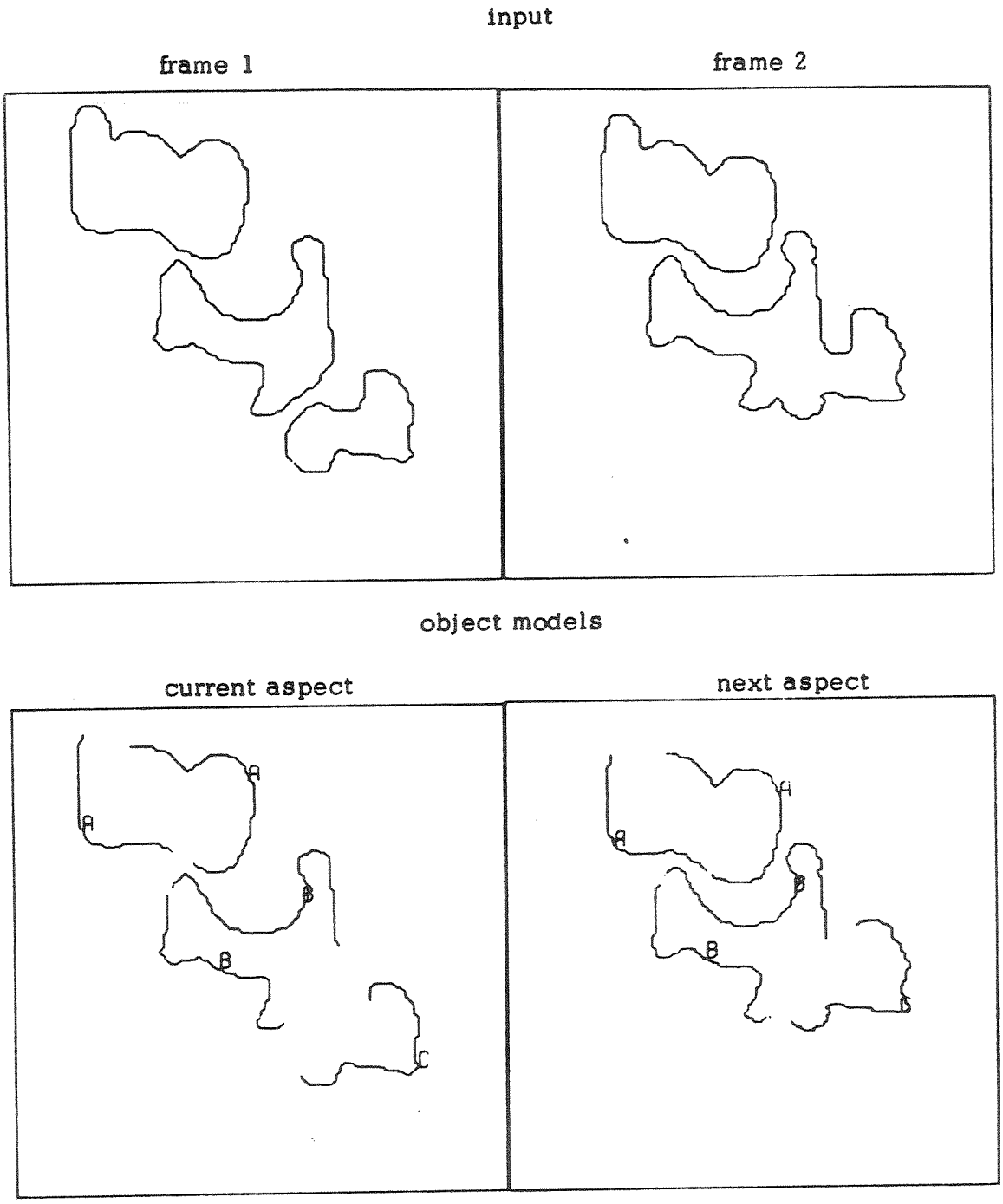


Figure 16. Example 3: Three Objects With Various Motions.

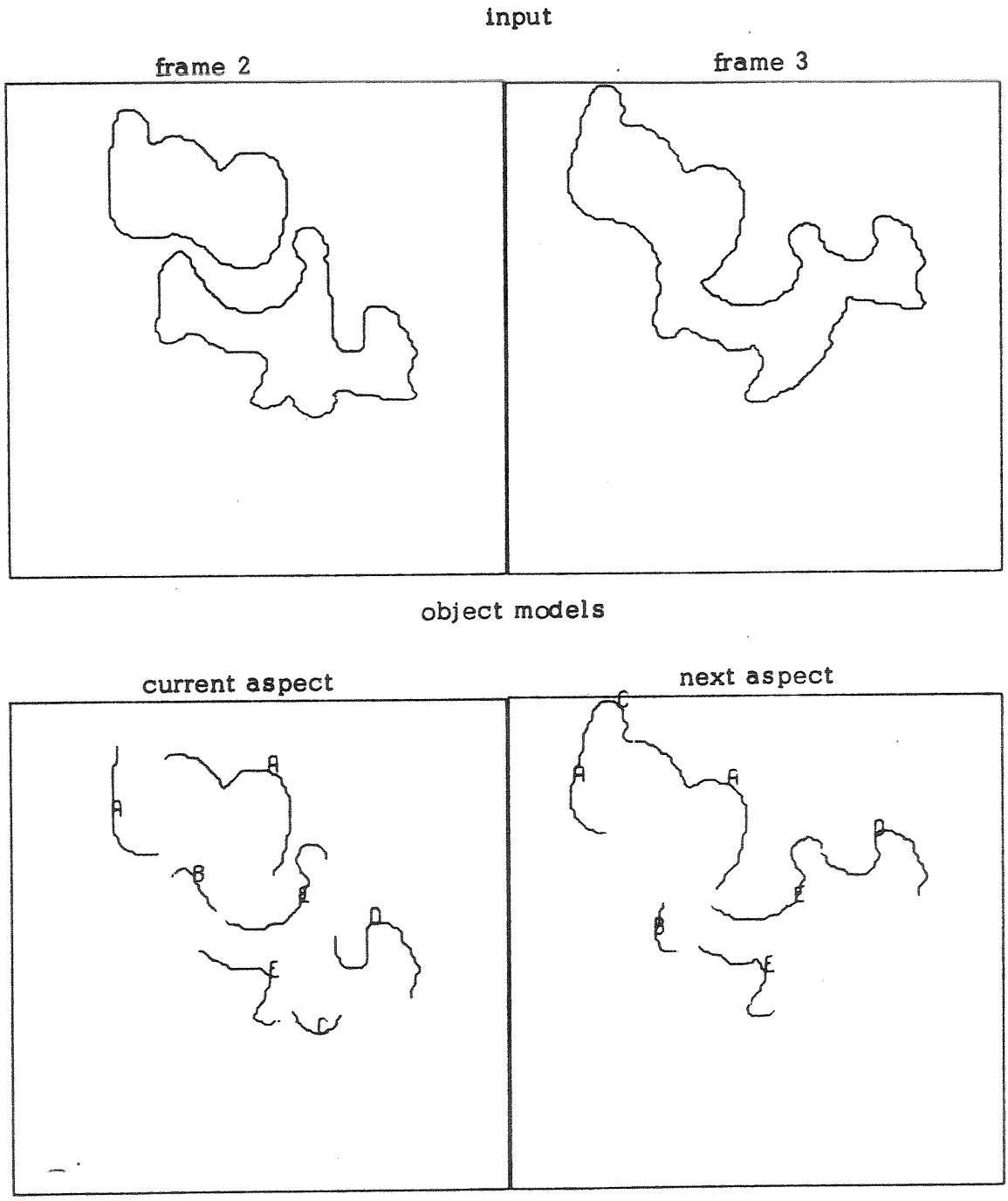


Figure 16. (continued)

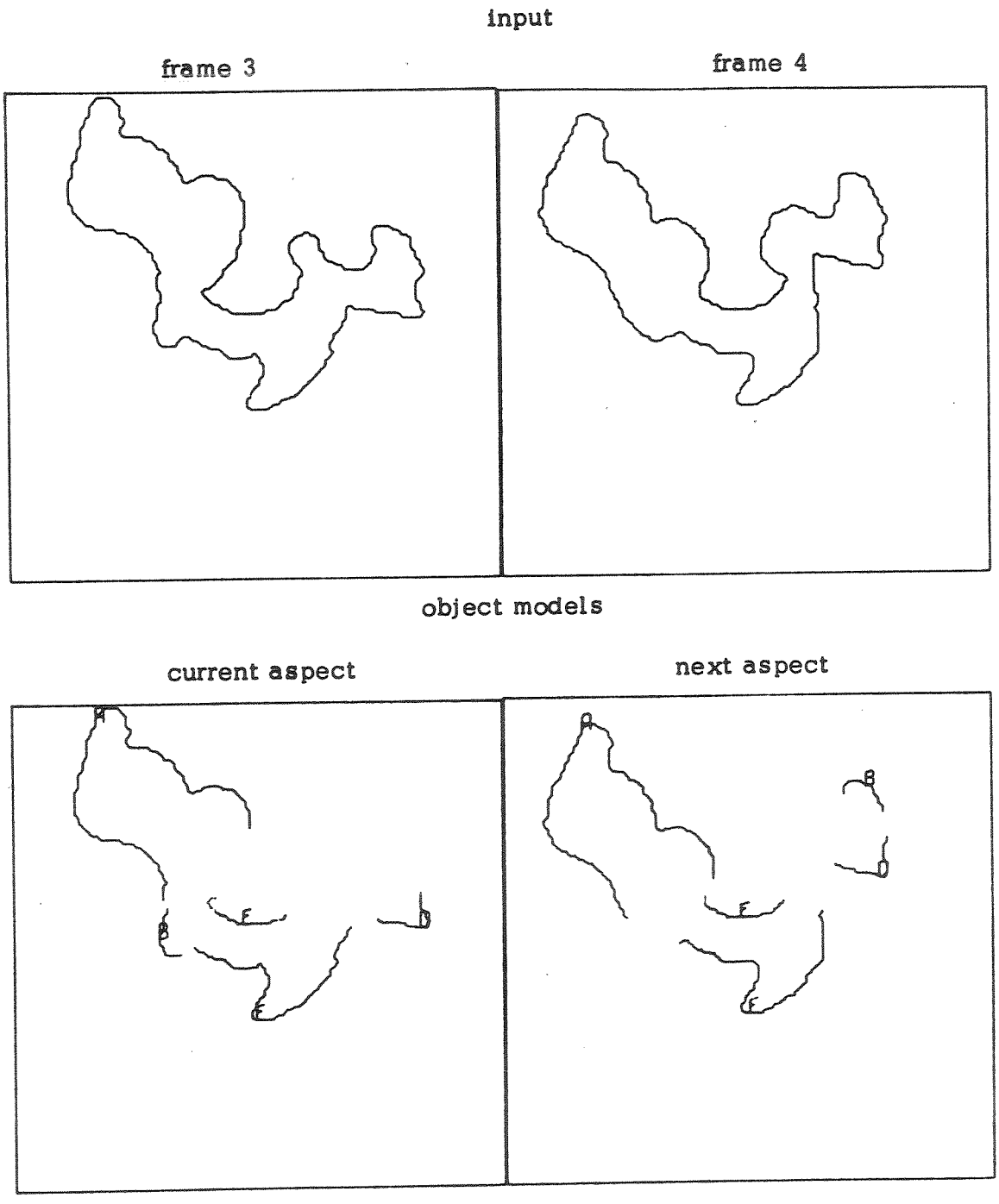


Figure 16. (continued)

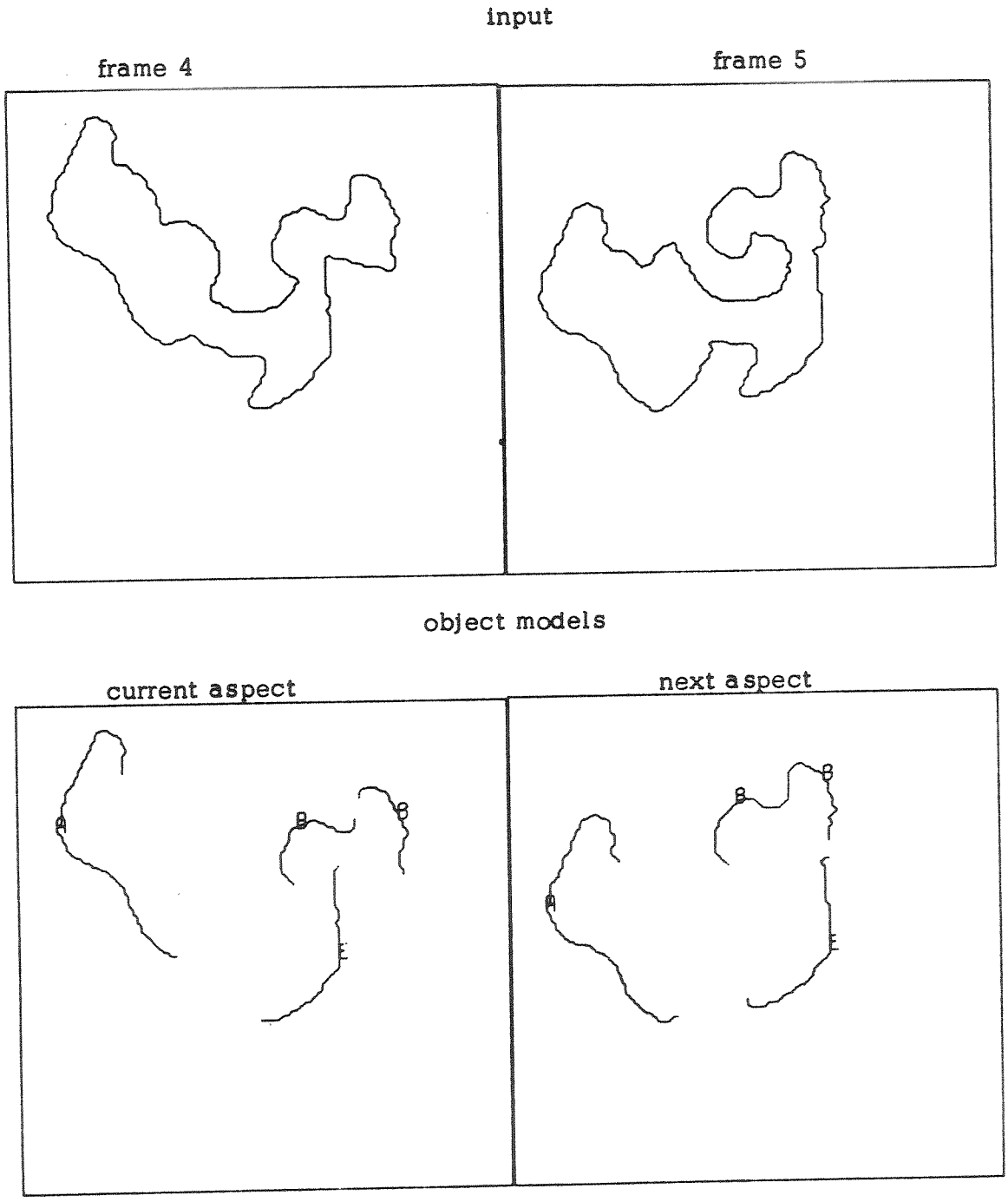


Figure 16. (continued)

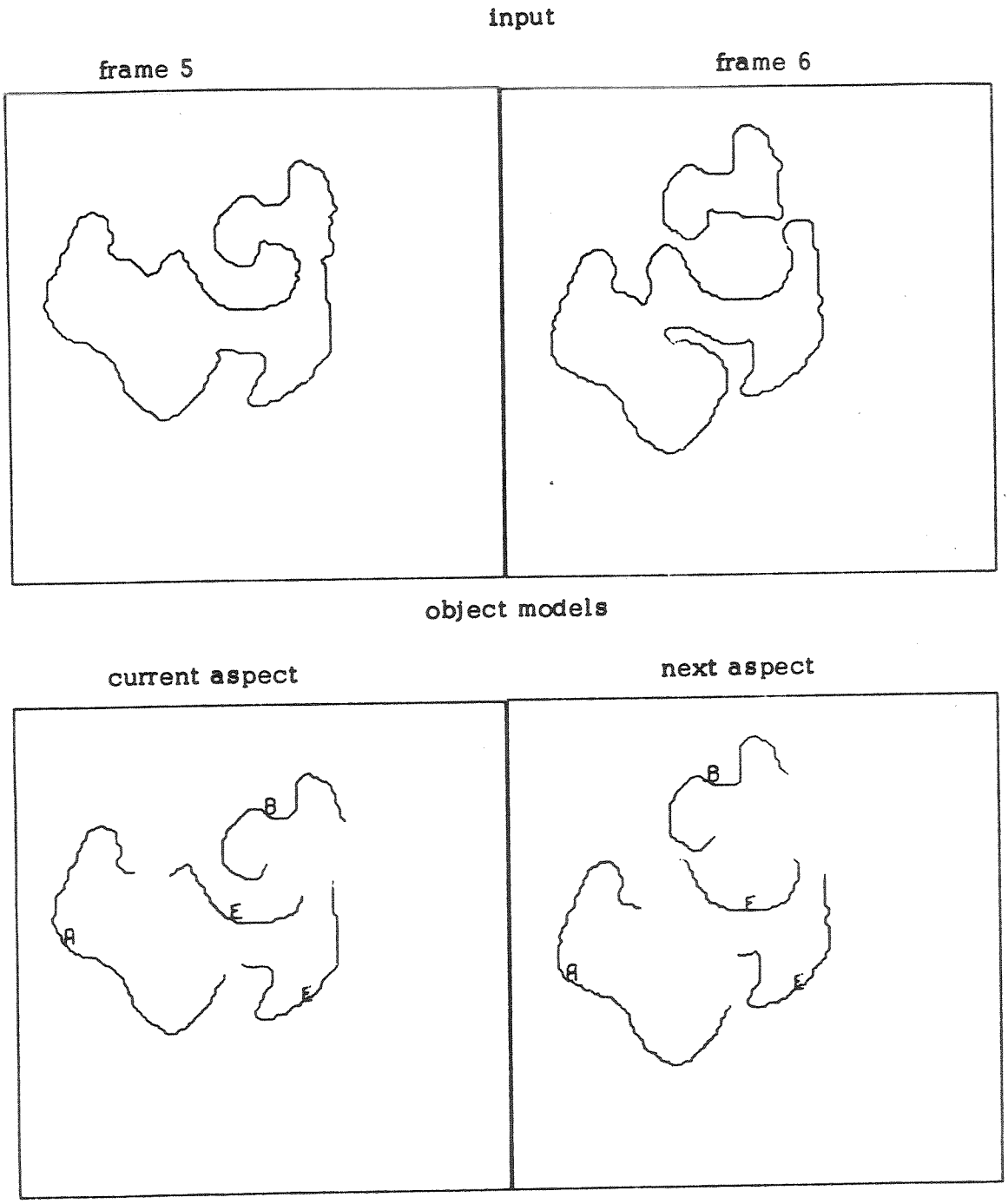


Figure 16. (continued)

4. CONCLUSION

In the Introduction we described a multilevel visual system suitable for perceiving dynamic scenes. Then, in Chapter 2 we discussed the problems inherent in trying to develop a computer system of this sort, and presented several previous attempts to develop the "peripheral" and "attentive" visual processes. It was shown that the first major problem of dynamic scene analysis, to associate "semantically identical" images although they appear different, has been solved in many ways. The motion-detecting processes of Section 2.2 solved it by identifying locally different frames by global techniques. In Refs. 43 and 46 the objects are reduced to centroids, making spatial location the only feature of the objects. This reduction makes the solution to the first major problem rather simple, while it destroys the features needed to solve the second major problem, occlusion.

The two most successful systems for analyzing occluding objects (static [16] and dynamic [17] scenes), both rely on local feature analysis and the polygonal nature of their inputs. Chow and Aggarwal [36] analyzed the occlusion of curvilinear objects by features global to the objects, but only under a rather heavy-handed restriction on the motion of the occluding objects. The system described in Chapter 3 utilizes local features to analyze the curvilinear objects without severely restricting the movement of the objects. Indeed, this system can be seen as the complementary attentive process to Chow and Aggarwal's peripheral process. The

combination of the two systems might yield an overall system much like that described in the Introduction. However, it must be noted that scenes analyzed by these systems are still essentially artificial scenes: the objects are rigid and the images are rather clean. For the more complex and noisy everyday scenes, it appears that the extracted objects must be described by a "general-purpose model" as mentioned in Ref. 49, or by some form of "rubber mask" as presented in Ref. 50. Widrow [50] mentions in passing that the "rubber mask" can be used to keep track of the distortions as they are encountered and although he rejects the idea for his application, it is a facility fundamental to dynamic scene analysis.

Discussing two levels of analysis implies that a general computer vision system will integrate the use of both levels. The general system would have procedures to "watch" the peripheral regions of the scene. These procedures would alert the system to areas which were undergoing extensive change. The system would then decide (possibly directed by a goal-oriented procedure) which areas merit the full attentive-level processes. From these remarks it is clear that biological vision systems still serve as the main model for generalized computer vision systems. The extreme efficiency of biological systems validate their use as models; however, the extensive parallelism in the biological processes remains problematic for the essentially serial computer processes.

At the present state of dynamic scene analysis the "peripheral" vision problem, global motion detection, has been solved in a number of

satisfactory ways, while more research is needed on the "attentive" vision problem of locally analyzing general shapes. In addition, research is needed to derive systems which use both levels of analysis and are able to exploit the parallelism inherent to the visual process.

APPENDIX

Information about a given dynamic scene is retained in four interrelated structures: the coordinate list; the encoded image list; the matched edge segment list; and the object model list. The coordinate list is a simple linear array, each element of which contains the x and y coordinates of a given marked point in a given frame of the scene. Due to the large size of the CDC6600 word, the x coordinate is packed, right justified in the left half of the word, while the y coordinate is packed into the right half of the word. Throughout this appendix the components of such packed words will be referred to as the "left" and "right". The coordinate list has no inherent structure; instead structure is imposed by references to the list from the other structures.

Encoded Image List

The encoded image list is structured by frames, images, and code lines. The first word contains the number of frames and is immediately followed by the block for the first frame. Each frame has a block of words with the following structure.

FRAME BLOCK

index	meaning of element
1	pointer to the next frame block
2	miscellaneous matching information
3	number of images in this frame
4	pointer to the last image block in this frame

A frame block is immediately followed by the first image block of this frame. Image blocks have the following structure and are connected into a doubly

linked list structure.

IMAGE BLOCK

index	meaning of element
1	pointer to the next image block
2	pointer to the previous image block
3 left	first spatial point of this image (points into the coordinate list)
right	last spatial point of this image
4 left	total area of the image
right	total perimeter of the image
5 left	the x coordinate of the centroid of the image
right	the y coordinate of the centroid of the image
6	miscellaneous matching information
7	miscellaneous matching information
8	number of code lines for this image

An image block is immediately followed by the linear list of its code line blocks, each having the following structure.

CODE LINE BLOCK

index	meaning of element
1	length of code line (in code graph units)
2	slope of the code line
3	initial value of the code line
4	if zero then the code line is not yet matched otherwise the left is the index of the edge segment which contains the code line, i.e., the edge segment for which this code line is the current aspect; and the right is the image pair of the edge segment (a pointer into the matched edge segment list)
5	the same as 4 except the code line is the next aspect
6 left	the first spatial point of this code line (points into the coordinate list)
right	the last spatial point of this code line

Matched Edge Segment List

The matched edge segment list is structured by frame pairs, image pairs, and edge segments. Each frame pair has a block of the following structure.

FRAME PAIR BLOCK

index	meaning of element
1 left	current frame (a pointer into the encoded image list)
right	next frame
2	pointer to next frame pair block

A frame pair block is immediately followed by its first image pair block.

Each image block has the following structure.

IMAGE PAIR BLOCK

index	meaning of element
1 left	image of the current aspect (a pointer into the encoded image list)
right	image of the next aspect
2	pointer to next image pair block for this frame pair
3	number of edge segments for this image pair

An image pair block is immediately followed by the linear list of its

edge segment blocks each having the following structure.

EDGE SEGMENT BLOCK

index	meaning of element
1	orientation of the seed code line (a boolean flag which is true when the segment was originally grown in a clockwise direction, starting from the counter-clockwise end of the seed code line)
2 left	previous model (if a portion of the edge segment matched to the previous frame then this points to the model which contains the portion)
right	current model
3 left	index of the seed code line for the current aspect
right	index of the seed code line for the next aspect
4 left	percent of the first code line actually matched by the segment
right	index of the first code line matched
5 left	same as 4 but for the last code line
right	same as 4 but for the last code line
6 left	first spatial point of the current aspect of the segment (a pointer into the coordinate list)
right	last spatial point of the current aspect
7	same as 4 but for the next aspect
8	same as 5 but for the next aspect
9	same as 6 but for the next aspect

Object Model List

The object model list is structured by object frames, model lists, and segment lists.

OBJECT FRAME BLOCK

index	meaning of element
1	pointer to next object frame block
2	pointer to the first model in this object frame

MODEL BLOCK

1	pointer to next model block
2	a header word
3 left	pointer to parent object frame block
right	print image of the modeled object
4	pointer to the first segment in this model

SEGMENT BLOCK

1	pointer to next segment block in this model
2 left	index of the segment
right	image pair of the segment (a pointer into the matched edge segment list)
3 left	x velocity of the segment
right	y velocity of the segment
4	angular velocity of the segment

References

1. Gibson, J.J., The Perception of the Visual World, Houghton Mifflin Co., Boston, 1950.

2. Cornsweet, T., Visual Perception, Academic Press, New York, 1970.

3. Kolers, P.A., Aspects of Motion Perception, Pergammon Press, Oxford, 1972.

4. Duda, R.O. and Hart, P.E., Pattern Classification and Scene Analysis, A. Wiley Interscience, New York, 1973.

5. Rosenfeld, A., Picture Processing by Computer, Academic Press, New York, 1969.

6. Winston, P.H., The Psychology of Computer Vision, McGraw-Hill, 1975.

7. Johansson, G. "Visual Motion Perception," Scientific American, Vol. 232, No. 6, June 1975, pp. 76-89.

8. Hubel, D.H. and Wiesel, T.N., "Receptive Fields in the Cat's Striate Cortex," J. Physiology, Vol. 148, 1959.

9. Lettvin, J.Y., Maturana, H.E., McCulloch, W.S., and Pitts, W.H., "What the Frog's Eye Tells the Frog's Brain," Proc. IRE, Vol. 47, 1959, pp. 1940-1951.

10. Barlow, H.B. and Hill, R.M., "Selective Sensitivity to Direction of Motion in Ganglion Cells of the Rabbit's Retina," Science, Vol. 139, 1963.

11. MacKay, D.M., "Interactive Processes in Visual Perception," Sensory Communication, Rosenbleth, W.A. (Ed.), MIT Press and Wiley, 1961.

12. Schouten, J.F., "Subjective Stroboscopy and a Model of Visual Movement Detectors," Models for the Perception of Speech and Visual Form, Wathen-Dunn (Ed.), MIT Press, 1967.

13. Price, K., "A Comparison of Human and Computer Vision Systems," SIGART Newsletter, No. 50, Feb. 1975, pp. 5-10.

14. Chien, R.T. and Jones, V.C., "Acquisition of Moving Objects and Hand-Eye Coordination," Advance Papers of the 4th International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, U.S.S.R., Sept. 3-8, 1975, pp. 737-740.

15. Futrelle, R.P. and Potel, M.J., "The System Design for GALATEA, An Interactive Real-time Computer Graphics System for Movie and Video Analysis," Computers and Graphics, Vol. 1, No. 1, May 1975, pp. 115-121.

16. Waltz, D., "Generating Semantic Descriptions from Drawings of Scenes with Shadows," AITR-271, MIT AI Lab, Nov. 1972. See also Winston [1] above, pp. 19-91.

17. Aggarwal, J.K. and Duda, R.O., "Computer Analysis of Moving Polygonal Images," IEEE Transactions on Computers, Vol. C-24, No. 10, Oct. 1975, pp. 966-976.

18. Kasvand, T. "Experiments with an On-Line Picture Language," Frontiers of Pattern Recognition, Watanabe (Ed.), Academic Press, 1972.

19. Martin, W. and Aggarwal, J.K., "Further Computer Analysis of Moving Images," Texas Biannual of Electronic Research No. 22, Electronics Research Center, The University of Texas at Austin, November 15, 1975, pp. 25-42.

20. Leese, J.A., Novak, C.S.,and Taylor, V.R., "The Determination of Cloud Pattern Motions from Geosynchronous Satellite Image Data," Pattern Recognition, Vol. 2, Dec. 1970, pp. 279-292.

21. Moore, G.A., "Application of Computers to Quantitative Analysis of Microstructures," National Bureau of Standards Report No. 9428, 1966.

22. Bristor, C.L., Frankel, M., and Kendall, E., "Some Advances in Automatic Determination of Cloud Motion and Growth from Digitized ATS-I Picture Pairs," Manuscript submitted to Weather Motions from Space ATS-I, University of Wisconsin Press, Madison, 1970.

23. Leese, J.A., Novak, C.S., and Clark, B.B., "An Automated Technique for Obtaining Cloud Motion from Geosynchronous Satellite Data Using Cross-Correlation," J. Applied Meteorology, Vol. 10, Feb. 1971, pp. 118-132.

24. Arking, A.A., Lo, R.C.,and Rosenfeld, A., "An Evaluation of Fourier Transform Techniques for Cloud Motion Estimation," Computer Science Technical Report TR-351, University of Maryland, Jan. 1975.

25. Lo, R.C. and Parikh, J.A., "A Study of the Application of Fourier Transforms to Cloud Movement Estimation from Satellite Photographs," TR-242, University of Maryland, 1973.

26. Lo, R.C., Mohr, J., and Parikh, J.A., "Applications of Fourier Transform Methods of Cloud Movement Estimation to Simulated and Satellite Photographs," TR-292, University of Maryland, 1974.

27. Lo, R.C. and Mohr, J., "Applications of Enhancement and Thresholding Techniques to Fourier Transform Cloud Motion Estimates," TR-326, University of Maryland, 1974.

28. Potter, J.L., "Motion as a Cue to Segmentation," Milwaukee Symposium on Automatic Control, 1974, pp. 100-104.

29. Potter, J.L., "Scene Segmentation by Velocity Measurements Obtained with a Cross-Shaped Template," 4IJCAI, Tbilisi, Georgia, U.S.S.R., Sept. 3-8, 1975, pp. 803-808.

30. Lillestrand, R.L., "Techniques for Change Detection," IEEE Transactions on Computers, Vol. C-21, July 1972, pp. 654-659.

31. Ulstad, M.S., "An Algorithm for Estimating Small Scale Differences Between Two Digital Images," Pattern Recognition, Vol. 5, 1973, pp. 323-333.

32. Limb, J.O. and Murphy, J.A., "Estimating the Velocity of Moving Images in Television Signals," Computer Graphics and Image Processing, Vol. 4, 1975, pp. 311-327.

33. Nagel, H.H., "Formation of an Object Concept by Analysis of Systematic Time Variations in the Optically Perceptible Environment," Bericht Nr. 27, University of Hamburg, July 1976.

34. Nagel, H.H., "Experience with Yakimovsky's Algorithm for Boundary and Object Detection in Real World Images," Bericht Nr. 23, University of Hamburg, March 1976.

35. Yakimovsky, Y., "Boundary and Object Detection in Real World Images," 4IJCAI, Tbilisi, Georgia, U.S.S.R., Sept. 1975, p. 695.

36. Chow, W.K. and Aggarwal, J.K., "Computer Analysis of Planar Curvilinear Moving Images," to appear in IEEE Transactions on Computers.

37. McKee, J. and Aggarwal, J.K., "Finding the Edges of the Surfaces of Three-Dimensional Curved Objects by Computer," Pattern Recognition, Vol. 7, 1975, pp. 25-52.

38. Fenton, T.R. and Vas, R., "Application of Moiré Topography to the Measure of 3-Dimensional Body Motion," Medical and Biological Engineering, Vol. 12, No. 4, July 1974, pp. 569, 572.

39. Will, P.M. and Pennington, K.S., "Grid Coding: A Preprocessing Technique for Robot and Machine Vision," Artificial Intelligence, Vol. 2, 1971, pp. 319-329.

40. Lappalainen, P. and Tervonen, M., "Instrumentation of Movement Analysis by Raster-Scanned Image Source," IEEE Trans. on Instrumentation and Measurement, Vol. IM-23, No. 3, Sept. 1975, pp. 217-221.

41. Nevatia, R., "Depth Measurement by Motion Stereo," Computer Graphics and Image Processing, Vol. 5, 1976, pp. 203-214.

42. Smith, E.A. and Phillips, D.R., "Automated Cloud-Tracking Using Precisely Aligned Digital ATS Pictures," IEEE Trans. on Computers, Vol. C-21, No. 7, July 1972, pp. 715-729.

43. Endlich, R.M., Wolf, D.E., Hall, D.J.,and Brain, A.E., "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.

44. Ball, G.H. and Hall, D.J., "A Clustering Technique for Summarizing Multivariate Data," Behavioral Science, Vol. 12, 1967, pp. 153-155.

45. Peterman, R., "Computer Analysis of Planar Motion of Polygons," Master's Thesis, University of Texas at Austin, Jan. 1975.

46. Greaves, J.O.B., "The Bugsystem: The Software Structure for the Reduction of Quantized Video Data of Moving Organisms," Proc. IEEE, Vol. 63, No. 10, Oct. 1975, pp. 1415-1425.

47. Freeman, H., "Computer Processing of Line-Drawing Images," Computing Surveys, Vol. 6, No. 1, March 1974.

48. McKee, J.W. and Aggarwal, J.K., "Computer Recognition of Partial Views of Curved Objects," to be published in the IEEE Transactions on Computers.

49. Zucker, S.W., Rosenfeld, A.,and Davis, L.S., "General-Purpose Models: Expectations about the Unexpected," 4IJCAI, Tbilisi, Georgia, U.S.S.R., Sept. 1975, pp. 716-721.

50. Widrow, B., "The 'Rubber-Mask' Technique. I. Pattern Measurement and Analysis; II. Pattern Storage and Recognition," Pattern Recognition, Vol. 5, pp 175-211.