Chapter 14

Dynamic Vision

Most biological vision systems have evolved to cope with the changing world. Machine vision systems have developed in the same way. For a computer vision system, the ability to cope with moving and changing objects, changing illumination, and changing viewpoints is essential to perform several tasks. Although early computer vision systems were concerned primarily with static scenes, computer vision systems for analyzing dynamic scenes are being designed for different applications.

The input to a dynamic scene analysis system is a sequence of image frames taken from a changing world. The camera used to acquire the image sequence may also be in motion. Each frame represents an image of the scene at a particular instant in time. The changes in a scene may be due to the motion of the camera, the motion of objects, illumination changes, or changes in the structure, size, or shape of an object. It is usually assumed that the changes in a scene are due to camera and/or object motion, and that the objects are either rigid or quasi-rigid; other changes are not allowed. The system must detect changes, determine the motion characteristics of the observer and the objects, characterize the motion using high-level abstraction, recover the structure of the objects, and recognize moving objects. In applications such as video editing and video databases, it may be required to detect *macro* changes in a sequence. These changes will partition the segment into many related segments exhibiting similar camera motion or a similar scene in a sequence.

A scene usually contains several objects. An image of the scene at a given time represents a projection of the scene, which depends on the position of the camera. There are four possibilities for the dynamic nature of the camera and world setup:

1. Stationary camera, stationary objects (SCSO)
2. Stationary camera, moving objects (SCMO)
3. Moving camera, stationary objects (MCSO)
4. Moving camera, moving objects (MCMO)

For analyzing image sequences, different techniques are required in each of the above cases. The first case is simply static-scene analysis.

In many applications, it may be necessary to process a single image to obtain the required information. This type of analysis has been the topic of discussion in the earlier chapters in this book.

Some applications require information extracted from a dynamic environment; in some cases a vision system must understand a dynamic process from a single viewpoint. In applications such as mobile robots or autonomous vehicles, a vision system must analyze an image sequence acquired while in motion. As we will see, recovering information from a mobile camera requires different techniques than those useful when the camera remains stationary.

A sequence of image frames offers much more information to aid in understanding a scene but significantly increases the amount of data to be processed by the system. The application of static-scene analysis techniques to each frame of a sequence requires an enormous amount of computation, in addition to all of the difficulties of static-scene analysis. Fortunately, research in dynamic-scene analysis has shown that the recovery of information in many cases is easier in dynamic scenes than in static scenes.

In dynamic-scene analysis, SCMO scenes have received the most attention. In analyzing such scenes, the goal is usually to detect motion, to extract masks of moving objects for recognizing them, and to compute their motion characteristics. MCSO and MCMO scenes are very important in navigation applications. MCMO is the most general and possibly the most difficult situation in dynamic scene analysis, but it is also the least developed area of computer vision.

Dynamic scene analysis has three phases:

- Peripheral
- Attentive
- Cognitive

The peripheral phase is concerned with extraction of approximate information which is very helpful in later phases of analysis. This information indicates the activity in a scene and is used to decide which parts of the scene need careful analysis. The attentive phase concentrates analysis on the active parts of the scene and extracts information which may be used for recognition of objects, analysis of object motion, preparation of a history of events taking place in the scene, or other related activities. The cognitive phase applies knowledge about objects, motion verbs, and other application-dependent concepts to analyze the scene in terms of the objects present and the events taking place.

The input to a dynamic scene analysis system is a frame sequence, represented by $F(x, y, t)$ where x and y are the spatial coordinates in the frame representing the scene at time t. The value of the function represents the intensity of the pixel. It is assumed that the image is obtained using a camera located at the origin of a three-dimensional coordinate system. The projection used in this observer-centered system may be either perspective or orthogonal.

Since the frames are usually taken at regular intervals, we will assume that t represents the tth frame of the sequence, rather than the frame taken at absolute time t.

14.1 Change Detection

Detection of changes in two successive frames of a sequence is a very important step for many applications. Any perceptible motion in a scene results in some change in the sequence of frames of the scene. Motion characteristics can be analyzed if such changes are detected. A good quantitative estimate of the motion components of an object may be obtained if the motion is restricted to a plane that is parallel to the image plane; for three-dimensional motion, only qualitative estimates are possible. Any illumination change in

a scene will also result in changes in intensity values, as will scene changes in a TV broadcast or a movie.

Most techniques for dynamic-scene analysis are based on the detection of changes in a frame sequence. Starting with frame-to-frame changes, a global analysis of the sequence may be performed. Changes can be detected at different levels: pixel, edge, or region. Changes detected at the pixel level can be aggregated to obtain useful information with which the computational requirements of later phases can be constrained.

In this section, we will discuss different techniques for change detection. We will start with one of the simplest, yet one of the most useful change detection techniques, *difference pictures*, and then discuss change detection for edges and regions.

14.1.1 Difference Pictures

The most obvious method of detecting change between two frames is to directly compare the corresponding pixels of the two frames to determine whether they are the same. In the simplest form, a binary difference picture $DP_{jk}(x, y)$ between frames $F(x, y, j)$ and $F(x, y, k)$ is obtained by:

$$DP_{jk}(x, y) = \begin{cases} 1 & \text{if } |F(x, y, j) - F(x, y, k)| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{14.1}$$

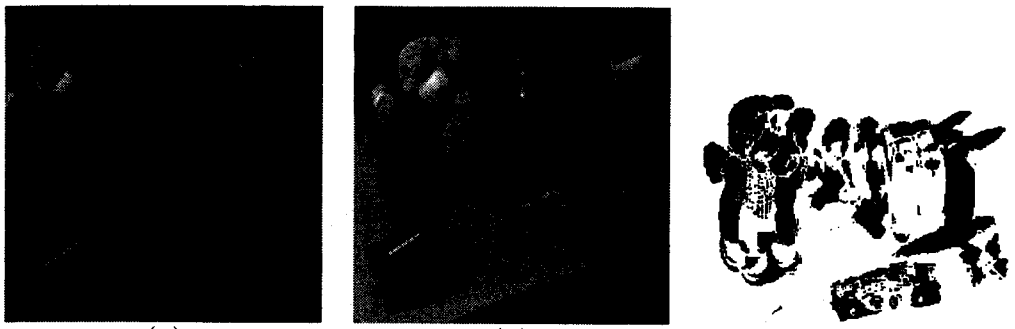
where τ is a threshold.

In a difference picture, pixels which have value 1 may be considered to be the result of object motion or illumination changes. This assumes that the frames are properly registered. In Figures 14.1 and 14.2 we show two cases of change detection, one due to illumination changes in a part of image and the other due to motion of an object.

A concept discussed in Chapter 3, thresholding, will play a very important role here also. Slow-moving objects and slowly varying intensity changes may not be detected for a given threshold value.

Size Filter

A difference picture obtained using the above simple test on real scenes usually results in too many noisy pixels. A simple size filter is effective in elimi-



(a) (b) (c)

Figure 14.1: Two frames from a sequence, (a) and (b), and their difference picture, (c). Notice that the changed areas (shown in black) are due to the motion of objects. We used $\tau = 25$.

(a) (b) (c)

Figure 14.2: Two frames from a sequence, (a) and (b), and their difference picture, (c). Notice that the changed areas are due to the changes in the illumination in the part of the scene. We used $\tau = 25$.

nating many noisy areas in the difference picture. Pixels that do not belong to a connected cluster of a minimum size are usually due to noise and can be filtered out. Only pixels in a difference picture that belong to a 4-connected (or 8-connected) component larger than some threshold size are retained for further analysis. For motion detection, this filter is very effective, but unfortunately it also filters some desirable signals, such as those from slow or small moving objects. In Figure 14.3 we show the difference picture for the frames shown in Figure 14.1 with $\tau = 10$ and the result of the size filtering.

(a) (b)

Figure 14.3: The difference picture for the frames shown in Figure 14.1 with $\tau = 10$ and the result of the size filtering are shown in (a) and (b), respectively. Notice that size filtering has removed many noisy regions in the image. All regions below 10 pixels were filtered out.

Robust Change Detection

To make change detection more robust, intensity characteristics of regions or groups of pixels at the same location in two frames may be compared using either a statistical approach or an approach based on the local approximation of intensity distributions. Such comparisons will result in more reliable change detection at the cost of added computation.

A straightforward domain-independent method for comparing regions in images is to consider corresponding areas of the frames. These corresponding areas may be the superpixels formed by pixels in nonoverlapping rectangular areas comprising m rows and n columns. The values of m and n are selected to compensate for the aspect ratio of the camera. Thus, a frame partitioned into disjoint superpixels, as shown in Figure 14.4(a), may be considered. Another possibility is to use a local mask, as in all convolutions, and compare the intensity distributions around the pixel, as shown in Figure 14.4(b).

One such method is based on comparing the frames using the likelihood

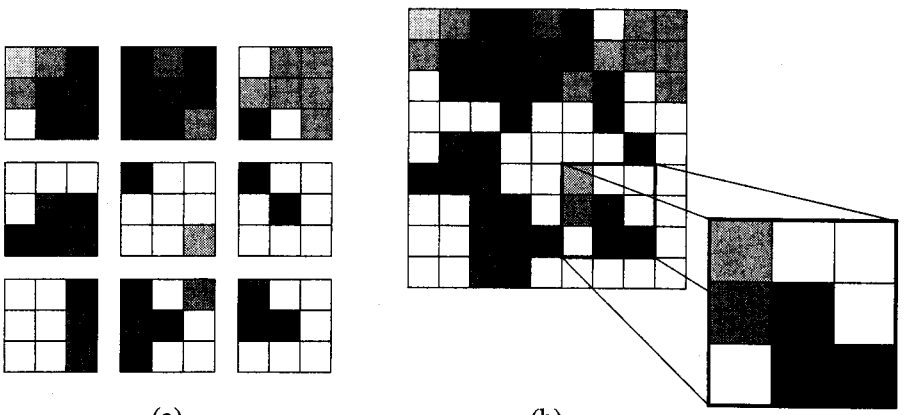


Figure 14.4: Partitioning frames for applying the likelihood ratio test. In (a) we show nonoverlapping areas, called superpixels, and (b) shows regular masks representing the local area of a pixel.

ratio. Thus, we may compute

$$\lambda = \frac{\left[\frac{\sigma_1^2+\sigma_2^2}{2} + \left(\frac{\mu_1-\mu_2}{2}\right)^2\right]^2}{\sigma_1^2 * \sigma_2^2} \tag{14.2}$$

(where μ and σ^2 denote the mean gray value and the variance for the sample areas from the frames) and then use

$$DP_{jk}(x,y) = \begin{cases} 1 & \text{if } \lambda > \tau \\ 0 & \text{otherwise} \end{cases} \tag{14.3}$$

where τ is a threshold. The likelihood ratio test combined with the size filter works quite well for many real world scenes. In Figure 14.5, we show the results of change detection using the likelihood ratio test.

The likelihood test discussed above was based on the assumption of uniform second-order statistics over a region. The performance of the likelihood ratio test can be improved significantly by using facets and quadratic surfaces to approximate the intensity values of pixels belonging to superpixels. These-higher order approximations allow for better characterization of intensity values and result in more robust change detection.

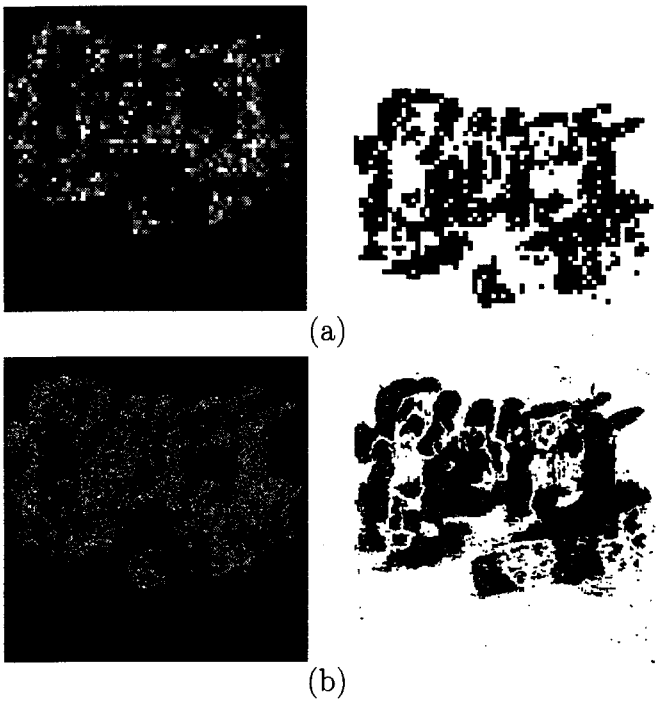


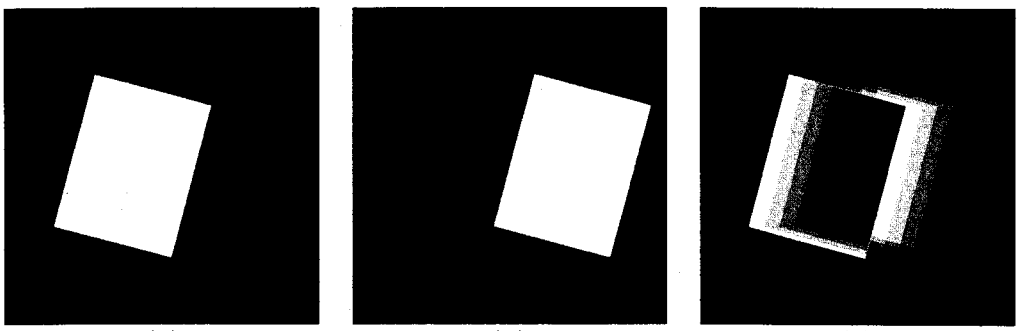
Figure 14.5: The results of the likelihood ratio test on the image pair in Fig. 14.1 for (a) superpixels and (b) regular masks.

Note that the likelihood ratio test results in detecting dissimilarities at the superpixel level. Since the tests use the likelihood ratio, they can only determine whether or not the areas under consideration have similar gray-level characteristics; information about the relative intensities of the areas is not retained. As shown later, the sign of the changes can also provide useful information for the analysis of motion.

Accumulative Difference Pictures

Small or slow-moving objects will usually result in a small number of changes using differencing approaches. A size filter may eliminate such pixels as noise. This problem of detecting small and slow-moving objects is exacerbated when using robust differencing approaches since superpixels effectively raise the size threshold for the detection of such objects.

By analyzing the changes over a sequence of frames, instead of just be-



(a) (b) (c)

Figure 14.6: Results for change detection using accumulative difference pictures. In (a) and (b) we show the first and last frames, respectively, of a synthetic object in motion. The accumulative difference picture is given in (c).

tween two frames, this problem may be solved. An accumulative difference picture may be used to detect the motion of small and slow-moving objects more reliably. An accumulative difference picture is formed by comparing every frame of an image sequence to a reference frame and increasing the entry in the accumulative difference picture by 1 whenever the difference for the pixel, or some measure for the superpixel, exceeds the threshold. Thus, an accumulative difference picture ADP_k is computed over k frames by

$$ADP_0(x,y) = 0 \tag{14.4}$$

$$ADP_k(x,y) = ADP_{k-1}(x,y) + DP_{1k}(x,y). \tag{14.5}$$

The first frame of a sequence is usually the reference frame, and the accumulative difference picture ADP_0 is initialized to 0. Small and slow-moving objects can be detected using an ADP. In Figure 14.6, results of detecting changes using accumulative difference pictures are shown.

Difference Pictures in Motion Detection

The most attractive aspect of the difference picture for motion detection is its simplicity. In its simplest form, the difference picture is noise-prone. Changes in illumination and registration of the camera, in addition to electronic noise of the camera, can result in many false alarms. A likelihood ratio in conjunction with a size filter can eliminate most of the camera noise. Changes in

illumination will create problems for all intensity-based approaches and can be handled only at a symbolic level. Misregistration of frames results in the assignment of false motion components. If the misregistration is not severe, accumulative difference pictures can eliminate it.

It should be emphasized that measuring dissimilarities at the pixel level can only detect intensity changes. In dynamic scene analysis, this is the lowest level of analysis. After such changes have been detected, other processes are required to interpret these changes. Experience has shown that the most efficient use of the difference picture is to have peripheral processes direct the attention of interpretation processes to areas of the scene where some *activity* is taking place. Approximate information about events in a scene may be extracted using some features of difference pictures.

14.1.2 Static Segmentation and Matching

Segmentation is the task of identifying the semantically meaningful components of an image and grouping the pixels belonging to such components. Segmentation need not be performed in terms of objects; some predicates based on intensity characteristics may also be used. Predicates based on intensity characteristics are usually called *features*. If an object or feature appears in two or more images, segmentation may be necessary in order to identify the object in the images. The process of identifying the same object or feature in two or more frames is called the correspondence process.

Static-scene analysis techniques can be used to segment, or at least partially segment, each frame of a dynamic sequence. Matching can then be used to determine correspondences and detect changes in the location of corresponding segments. Cross-correlation and Fourier domain features have been used to detect cloud motion. Several systems have been developed which segment each frame of a sequence to find regions, corners, edges, or other features in the frames. The features are then matched in consecutive frames to detect any displacement. Some restriction on the possible matches for a feature can be achieved by predicting the new location of the feature based on the displacements in previous frames.

The major difficulty in the approaches described above is in segmentation. Segmentation of an image of a static scene has been a difficult problem. In most cases, a segmentation algorithm results in a large number of features in each frame, which makes the correspondence problem computationally

quite expensive. Moreover, it is widely believed that motion detection can be used to produce better segmentation, as opposed to the techniques of segmentation and matching used to determine motion above.

14.2 Segmentation Using Motion

The goal of many dynamic-scene analysis systems is to recognize moving objects and to find their motion characteristics. If the system uses a stationary camera, segmentation generally involves separating moving components in the scene from stationary components. Then the individual moving objects are identified based either on their velocity or on other characteristics. For systems using a moving camera, the segmentation task may be the same as above or may include further segmentation of the scene's stationary components by exploiting the camera motion. Most research efforts for the segmentation of dynamic scenes have assumed a stationary camera.

Researchers in perception have known for a long time that motion cues are helpful for segmentation. Computer vision techniques for segmenting SCMO scenes perform well compared to techniques for segmenting stationary scenes. Segmentation into stationary and nonstationary components, in a system using a moving camera, has only recently received attention. One problem in segmenting moving-observer scenes is that every surface in the scene has image plane motion. This is precisely what can be used to aid the separation of moving and stationary objects in stationary-camera scenes. For segmenting moving-camera scenes, the motion assigned to components in the images due to the motion of the camera should be removed. The fact that the image motion of a surface depends both on the surface's distance from the camera and on the structure of the surface complicates the situation.

Segmentation may be performed using either region-based or edge-based approaches. In this section, some approaches for the segmenting of dynamic scenes are discussed.

14.2.1 Time-Varying Edge Detection

As a result of the importance of edge detection in static scenes, it is reasonable to expect that time-varying edge detection may be very important in dynamic-scene analysis. In segment-and-match approaches, efforts are

(a) (b) (c)

Figure 14.7: In (a) and (b), two frames of a sequence are shown. In (c), edges are detected using the time-varying edge detector.

wasted on attempting to match static features to moving features. These static features are obstacles to extracting motion information. If only moving features are detected, the computation needed to perform matching may be substantially reduced.

A moving edge in a frame is an edge, *and* moves. Moving edges can be detected by combining the temporal and spatial gradients using a logical AND operator. This AND can be implemented through multiplication. Thus, the time-varying edginess of a point in a frame $E(x, y, t)$ is given by

$$E_t(x, y, t) = \frac{dF(x, y, t)}{dS} \cdot \frac{dF(x, y, t)}{dt} \tag{14.6}$$

$$= E(x, y, t) \cdot D(x, y) \tag{14.7}$$

where $dF(x, y, t)/dS$ and $dF(x, y, t)/dt$ are, respectively, the spatial and temporal gradients of the intensity at point (x, y, t). Various conventional edge detectors can be used to compute the spatial gradient, and a simple difference can be used to compute the temporal gradient. In most cases, this edge detector works effectively. By applying a threshold to the product, rather than first differencing and then applying an edge detector or first detecting edges and then computing their temporal gradient, this method overcomes the problem of missing slow-moving or weak edges. See Figures 14.7 and 14.8.

As shown in Figure 14.8, this edge detector will respond to slow-moving edges that have good edginess and to poor edges that are moving with appreciable speed. Another important fact about this detector is that it does

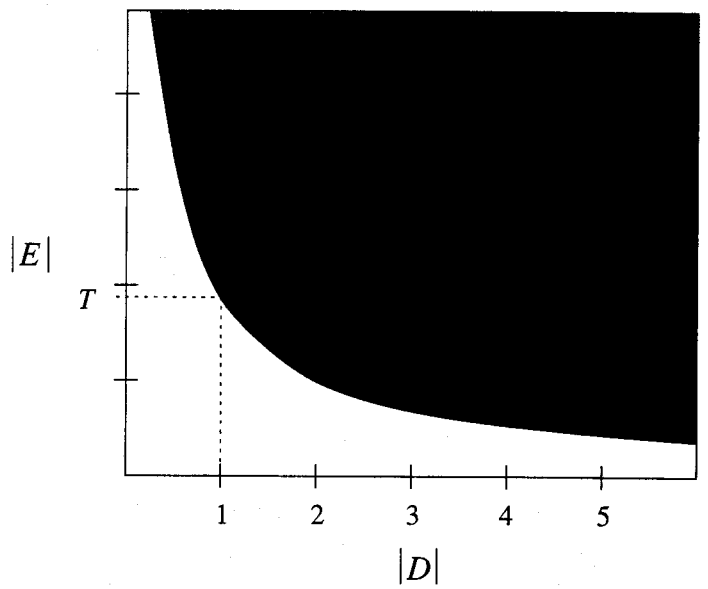


Figure 14.8: A plot showing the performance of the edge detector. Note that slow-moving edges will be detected if they have good contrast, and that poor-contrast edges will be detected if they move well.

not assume any displacement size. The performance of the detector is satisfactory even when the motion of an edge is very large.

14.2.2 Stationary Camera

Using Difference Pictures

Difference and accumulative difference pictures find the areas in a scene which are changing. An area is usually changing due to the movement of an object. Although change detection results based on difference pictures are sensitive to noise, the areas produced by a difference picture are a good place from which to start segmentation. In fact, it is possible to segment a scene with very little computation using accumulative difference pictures. In this section, such an approach is discussed.

Let us define absolute, positive, and negative difference pictures and accumulative difference pictures as follows:

$$DP_{12}(x,y) = \begin{cases} 1 & \text{if } |F(x,y,1) - F(x,y,2)| > T \\ 0 & \text{otherwise} \end{cases} \quad (14.8)$$

$$PDP_{12}(x,y) = \begin{cases} 1 & \text{if } F(x,y,1) - F(x,y,2) > T \\ 0 & \text{otherwise} \end{cases} \quad (14.9)$$

$$NDP_{12}(x,y) = \begin{cases} 1 & \text{if } F(x,y,1) - F(x,y,2) < T \\ 0 & \text{otherwise} \end{cases} \quad (14.10)$$

$$AADP_n(x,y) = AADP_{n-1}(x,y) + DP_{1n}(x,y) \quad (14.11)$$

$$PADP_n(x,y) = PADP_{n-1}(x,y) + PDP_{1n}(x,y) \quad (14.12)$$

$$NADP_n(x,y) = NADP_{n-1}(x,y) + NDP_{1n}(x,y). \quad (14.13)$$

Depending on the relative intensity values of the moving object and the background being covered and uncovered, PADP and NADP provide complementary information. In either the PADP or NADP, the region due to the motion of an object continues to grow after the object has been completely displaced from its projection in the reference frame, while in the other, it stops growing. The area in the PADP or NADP corresponds to the area covered by the object image in the reference frame. The entries in this area continue to increase in value, but the region stops growing in size. Accumulative difference pictures for a synthetic scene are shown in Figure 14.9. A test to determine whether or not a region is still growing is needed in order to obtain a mask of an object. The mask can then be obtained from the accumulative difference picture where the object's area stops growing after the object is displaced from its projection. In its simplest form, this approach has one obvious limitation. Masks of moving objects can be extracted only after the object has been completely displaced from its projection in the reference frame. However, it appears that properties of difference and accumulative difference pictures can be used to segment images in complex situations, such as running occlusion. To prevent running occlusions from disrupting segmentation, the segmentation process should not wait for an object's current position to be completely displaced from its projection in the reference frame. Regions in the accumulative difference pictures can be monitored as opposed to monitoring the reference frame projections of objects. Simple tests on the rate of region growth and on the presence of *stale* entries allow a system to determine which regions are eventually going to mature and result

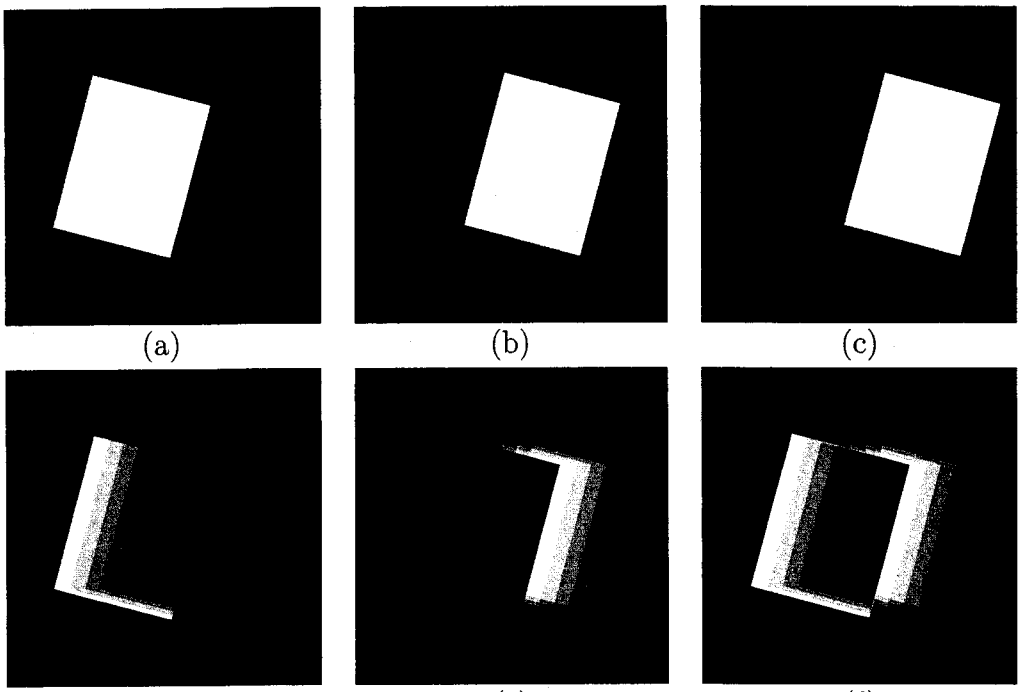


Figure 14.9: (a)–(c) show frames 1, 5, and 7 of a scene containing a moving object. The intensity-coded positive, negative, and absolute ADPs are shown in parts (d), (e), and (f), respectively.

in a mask for an object in the reference frame. Early determination of reference frame positions of objects, and hence, extraction of masks for objects, allows a system to take the action necessary to prevent running occlusion.

14.3 Motion Correspondence

Given two frames of a sequence, one can analyze them to determine features in each frame. To determine the motion of objects, one can establish the correspondence among these features. The correspondence problem in motion is similar to the correspondence problem in stereo. In stereo, the major constraint used is the epipolar constraint. However, in motion, other constraints must be used. In the following we describe a constraint propaga-

tion approach to solve the correspondence problem. Since this approach is similar to the problem for stereo, and for historical reasons, we present the formulation similar to that for stereo.

Relaxation Labeling

In many applications, we are given a set of labels that may be assigned to objects that may appear in the "world." Possible relationships among different objects in this world and the conditions under which a certain set of labels may or may not be applied to a set of objects is also known. The relationships among the objects in the image may be found using techniques discussed in earlier chapters. Now, based on the knowledge about the labels in the domain, proper labels must be assigned to the objects in the image. This problem is called the *labeling problem.*

The labeling problem may be represented as shown in Figure 14.10. Each node represents an object or entity which should be assigned a label. The arcs connecting nodes represent relations between objects. This figure represents the observed entities and relations among them in a given situation. We have to assign labels to each entity based on the label set and the constraints among the labels for the given domain.

Assume that we have a processor at each node. We define sets R, C, L, and P for each node. The set R contains all possible relations among the nodes. The set C represents the compatibility among these relations. The compatability among relations helps in constraining the relationships and labels for each entity in an image. The set L contains all labels that can be assigned to nodes, and the set P represents the set of possible levels that can be assigned to a node at any instant in computation. Each processor knows the label of its node and all nodes which are connected to it. It also knows all relations involving its node and the sets R and C. Assume that in the first iteration the possible label set P_i^1 of node i is L for all i. In other words, all nodes are assigned all possible labels initially. The labeling process should then iteratively remove invalid labels from P_i^k to give P_i^{k+1}. Since at any stage labels are discarded considering only the current labels of the node, its relations with other nodes, and the constraints, each processor has sufficient information to refine its label set P_i^k. Thus, it is possible for all processors to work synchronously. Note that at any time a processor uses only information directly available to it, that is, information pertaining

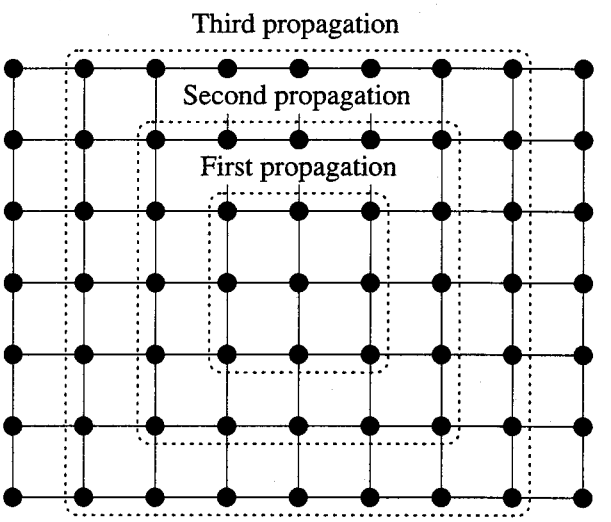


Figure 14.10: Parallel propagation in graphs.

only to its object. Each iteration, however, propagates the effect through its neighbor or related nodes, to other nodes that are not directly related. The circle of influence of a node increases with each iteration. This propagation of influence results in global consistency through direct local effects.

In most applications, some knowledge about the objects is available at the start of the labeling process. Segmentation, or some other process which takes place before labeling, often gives information that can be used to refine the initial set P_i for a node. This knowledge can be used to refine the initial label sets for objects. The labeling process is then used to further refine these sets to yield a unique label for each object. The labeling problem now may be considered in a slightly different form than the above. Based on some unary relations, a label set P_i^1 can be assigned to an object. The correct label is uncertain. However, a confidence value can be assigned to each label $l_k \in P_i^1$. The confidence value, like a subjective probability, indicates a belief that the entity may be assigned this label based on the available evidence in the image. Thus, for each element $l_k \in P_i$, a nonnegative probability p_{ik} represents the confidence that the label l_k is the correct label for node i. This confidence value may be considered the membership value if approaches from fuzzy set theory are used in constraint propogation.

The task of the labeling process is to use the constraints to refine the

confidence value for each label. The confidence value p_{ik} is influenced by the confidence values in the labels of the connected nodes. Thus, in the tth iteration, the confidence value p_{ik}^t for the label l_k at node i is the function of the confidence value p_{ik}^{t-1} and the confidence values of the labels of all directly related nodes. In each iteration a node looks at the labels of all its related nodes and then uses the known constraints to update the confidence in its labels. The process may terminate either when each node has been assigned a unique label or when the confidence values achieve a steady state. Note that in place of just the presence or absence of a label, there is now a continuum of confidence values in a label for an object.

The above process, commonly called the *relaxation labeling process*, attempts to decide which of the possible interpretations is correct on the basis of local evidence. Interestingly, though, the final interpretation is globally correct. In each iteration, the confidence in a label is directly influenced only by directly related nodes. However, this influence is propagated to other nodes in later iterations. The sphere of influence increases with the number of iterations. In relaxation labeling, the constraints are specified in terms of compatibility functions. Suppose that objects O_i and O_j are related by R_{ij}, and under this relation labels L_{i_k} and L_{j_l} are highly "likely" to occur. The knowledge about the likelihood of these labels can be expressed in terms of a function that will increase the confidence in these labels for the objects under consideration. In such a situation, the presence of L_{i_k} at O_i encourages the assignment of L_{j_l} to O_j. It is also possible that the incompatibility of certain labels can be used to discourage labels by decreasing their confidence values.

In the following, we discuss an algorithm that uses relaxation labeling to determine disparity values in images. The algorithm to determine optical flow, discussed later in this chapter, also is an example of relaxation labeling.

Disparity Computations as Relaxation Labeling

The matching problem is to pair a point $p_i = (x_i, y_i)$ in the first image with a point $p_j = (x_j, y_j)$ in the second image. The disparity between these points is the displacement vector between the two points:

$$d_{ij} = (x_i - x_j, y_i - y_j). \tag{14.14}$$

The result of matching is a set of conjugate pairs.

In any kind of matching problem, there are two questions that must be answered:

- How are points selected for matching? In other words, what are the features that are matched?
- How are the correct matches chosen? What constraints, if any, are placed on the displacement vectors?

Three properties guide matching:

Discreteness, which is a measure of the distinctiveness of individual points

Similarity, which is a measure of how closely two points resemble one another

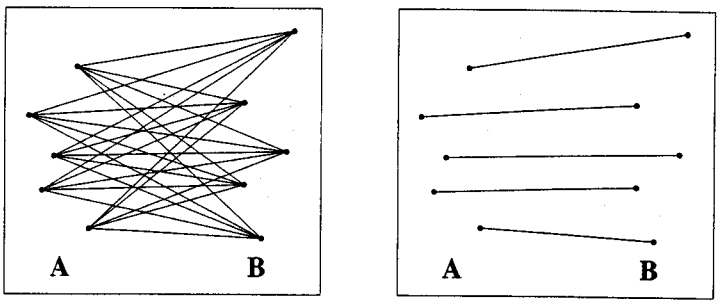
Consistency, which is a measure of how well a match conforms to nearby matches

The property of discreteness means that features should be isolated points. For example, line segments would not make good features since a point can be matched to many points along a line segment. Discreteness also minimizes expensive searching by reducing the problem of analyzing image disparities to the problem of matching a finite number of points.

The set of potential matches form a bipartite graph, and the matching problem is to choose a (partial) covering of this graph. As shown in Figure 14.11, initially each node can be considered as a match for each node in the other partition. Using some criterion, the goal of the correspondence problem is to remove all other connections except one for each node. The property of similarity indicates how close two potential matching points are to one another; it is a measure of affinity. Similarity could be based on any property of the features that are selected to implement discreteness.

The property of consistency is implied by the spatial continuity of surfaces in the scene and assumes that the motion is well behaved. Consistency allows the obvious matches to improve the analysis of more difficult matches. Some points are sufficiently distinct and similar that it is easy to match them; this match can assist in matching nearby points.

The discrete feature points can be selected using any corner detector or a feature detector. One such feature detector is the Moravec interest operator.



(a) (b)

Figure 14.11: (a) A complete bipartite graph. Here each node in group A has a connection with each node in group B. Using a characteristic of nodes (points) and some other knowledge, a correspondence algorithm must remove all but one connection for each node, as shown in (b).

This operator detects points at which intensity values are varying quickly in at least one direction. This operator can be implemented in the following steps:

1. Compute sums of the squares of pixel differences in four directions (horizontal, vertical, and both diagonals) over a 5×5 window.
2. Compute the minimum value of these variances.
3. Suppress all values that are not local maxima.
4. Apply a threshold to remove weak feature points.

Any feature detector can be used in place of the above operator. One can use a corner detector or computed curvature values at every point and select high curvature points as features.

Next one must pair each feature point in the first image with all points in the second image within some maximum distance. This will eliminate many connections from the complete bipartite graph. The connections removed are those that are between points far away in two images and hence unlikely to be candidate matches. Each node a_i has position (x_i, y_i) in the first image and a set of possible labels (disparity vectors). The disparity labels are displacement vectors or the undefined disparity, which allows some feature points to remain unmatched.