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:

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

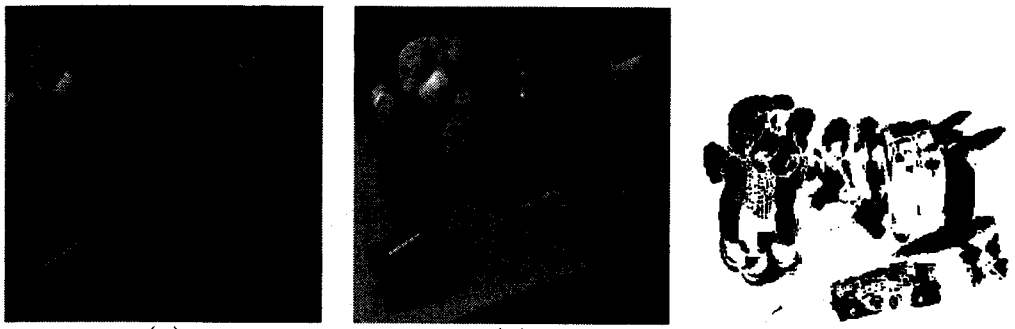
where τ is a threshold.

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

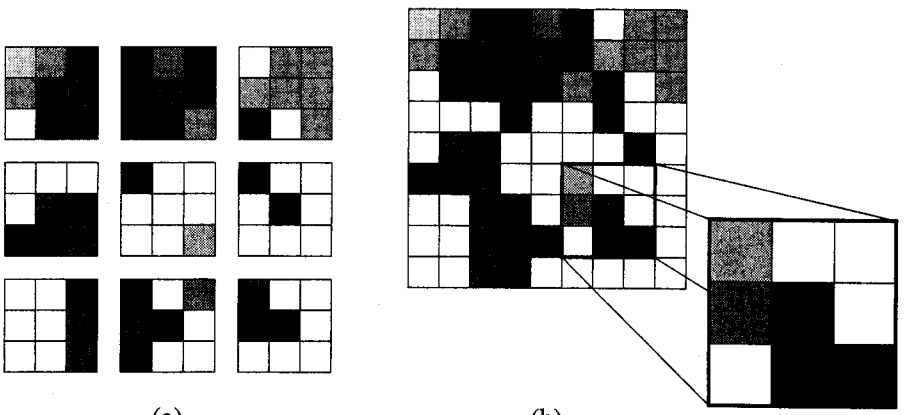


Figure 14.4: Partitioning frames for applying the likelihood ratio test. In (a) we show nonoverlapping areas, called superpixels, and (b) shows regular masks representing the local area of a pixel.

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.

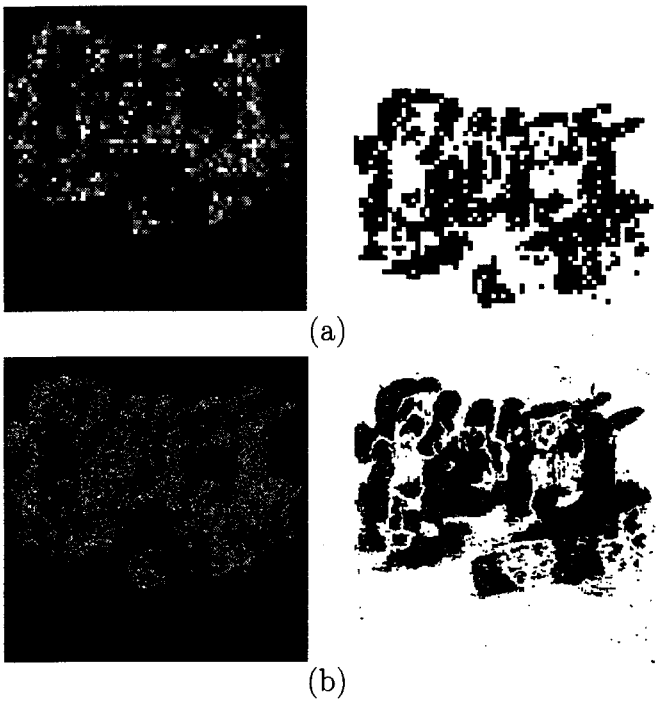


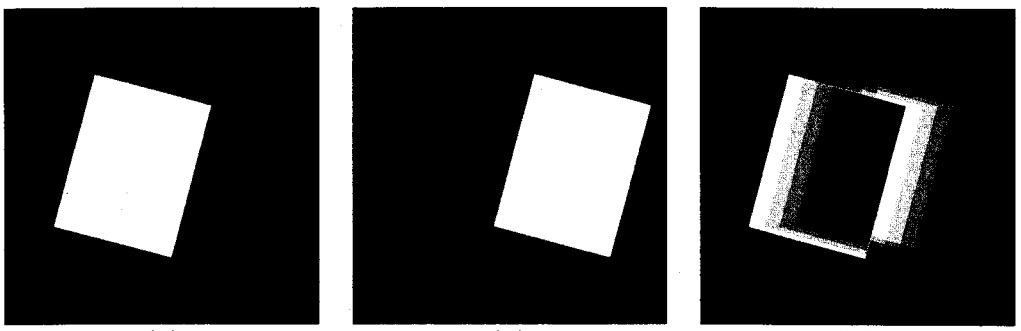
Figure 14.5: The results of the likelihood ratio test on the image pair in Fig. 14.1 for (a) superpixels and (b) regular masks.

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

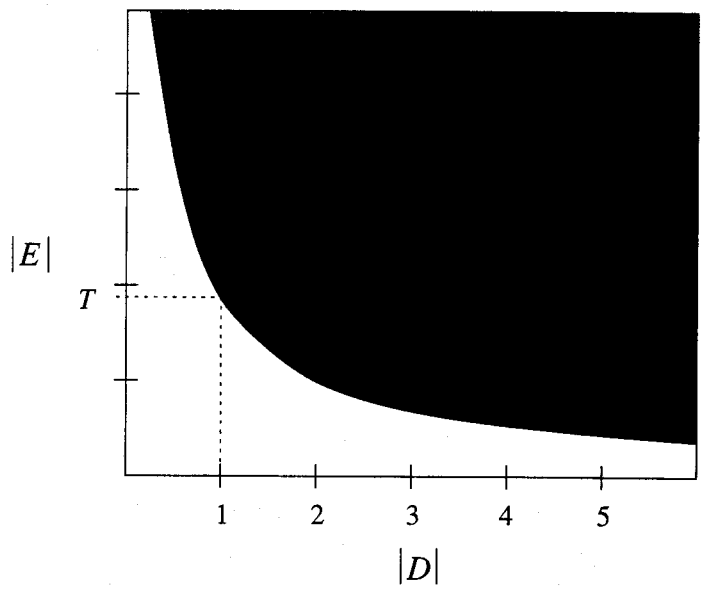


Figure 14.8: A plot showing the performance of the edge detector. Note that slow-moving edges will be detected if they have good contrast, and that poor-contrast edges will be detected if they move well.

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

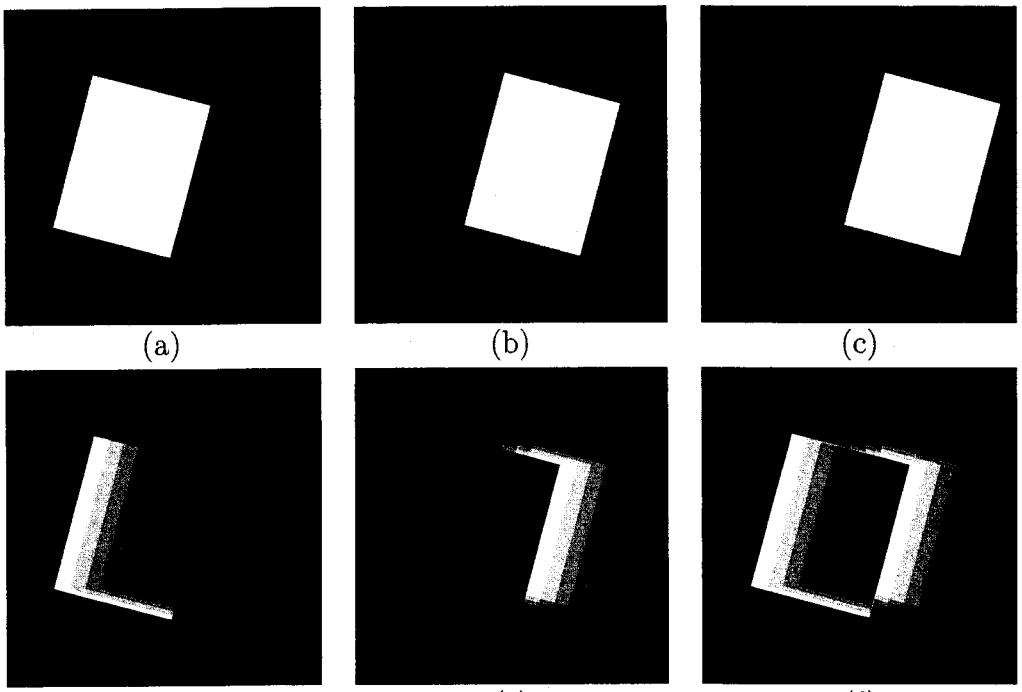


Figure 14.9: (a)–(c) show frames 1, 5, and 7 of a scene containing a moving object. The intensity-coded positive, negative, and absolute ADPs are shown in parts (d), (e), and (f), respectively.

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

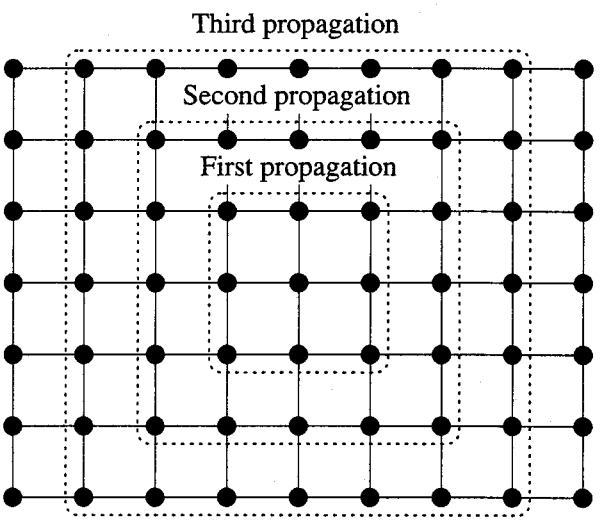


Figure 14.10: Parallel propagation in graphs.

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

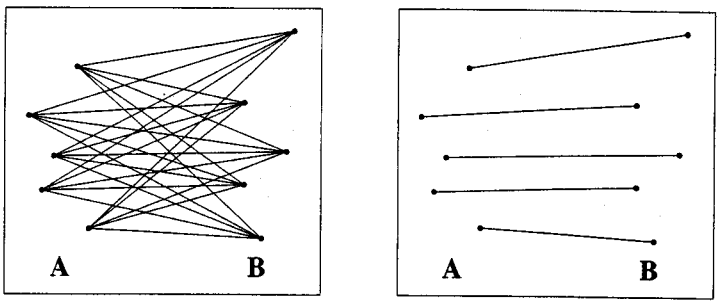
Consistency, which is a measure of how well a match conforms to nearby matches

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.

The initial probabilities of a match are computed using a measure of similarity between the feature points in the two images. A good measure is the sum of the squares of the pixel differences, s_i, in corresponding windows. The following approach may be used for assigning these probabilities.

Let l be a candidate label at a point. This label represents a disparity vector at the point. First we compute $w_i(l)$, which represents the similarity between the point (x_i, y_i) and its potential match at disparity l.

$$w_i(l) = \frac{1}{1 + cs_i(l)}, \tag{14.15}$$

where $s_i(l)$ is the sum of the squared differences corresponding to label l, and c is some positive constant. The probability that this point has undefined disparity is obtained by first defining

$$p_i^0(\text{undefined}) = 1 - \max(w_i(l)). \tag{14.16}$$

This probability is determined based on the strength of the most similar point for (x_i, y_i). If there are no strongly similar points, then it is likely that the point has no match in this image. The probabilities of the various matches (labels) are

$$p_i(l|i) = \frac{w_i(l)}{\sum w_i(l')}, \tag{14.17}$$

where $p_i(l|i)$ is the conditional probability that a_i has label l given a_i is matchable, and the sum is over all labels l' excluding the "undefined" label. The probability estimates are refined using the consistency property and iterative relaxation algorithm. In this approach, the labels at each node are strengthened or weakened based on the labels of the neighboring nodes in that iteration. The most important property used here is that all disparities should be similar in a given neighborhood. Thus, similar disparities of nodes in a neighborhood should strengthen each other and dissimilar ones should be weakened. This is accomplished using the following approach.

Let us consider probability for disparity vectors of all neighbors of a_i. For each neighbor, sum the probability of labels (disparities) that are close to, or similar to, the disparity of a_i:

$$q_i^k(l) = \sum_{\substack{\text{Neighbors} \\ \text{of } a_i}} \sum_{\substack{\text{Nearby} \\ \text{disparities}}} p_j^k(l'). \tag{14.18}$$

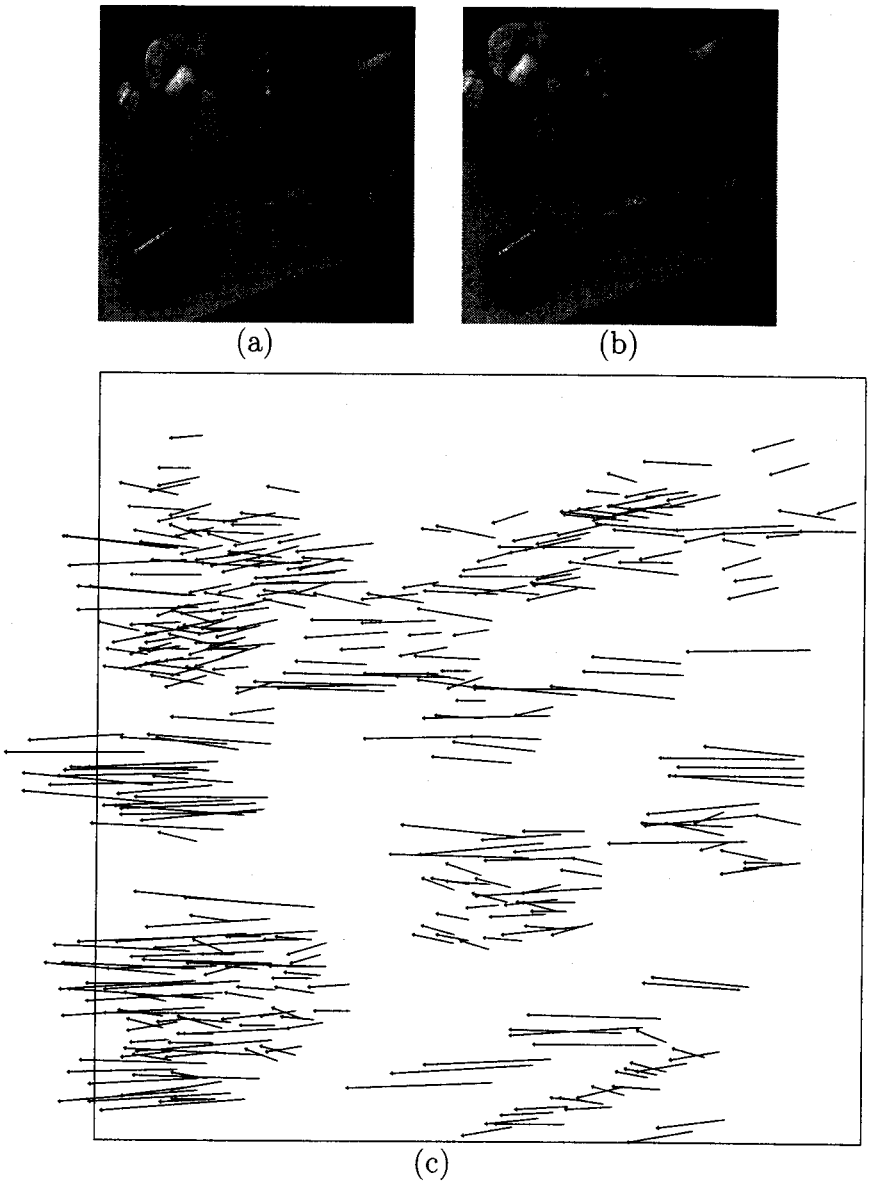


Figure 14.12: This figure shows two frames of a sequence and the disparities of the matched feature points (shown magnified by a factor of 5) after applying the relaxation labeling algorithm.

The probabilities are now refined with the iterative calculation:

$$p_i^{k+1}(l) = p_i^k(A + Bq_i^k(l)) \tag{14.19}$$

for constants A and B. Constants A and B are selected to control the rate of convergence of the algorithm. The updated probabilities must be normalized. Usually, a good solution is obtained after only a few iterations. To speed up the algorithm, matches with low probability are removed.

In Figure 14.12, we show two frames of a sequence and disparities found using the above algorithm. Interested readers should see [21] for an in-depth analysis of disparity calculations.

14.4 Image Flow

Image flow is the distribution of velocity, relative to the observer, over the points of an image. Image flow carries information which is valuable for analyzing dynamic scenes. Several methods for dynamic-scene analysis have been proposed which assume that image flow information is available. Unfortunately, however, although image flow has received a significant amount of attention from researchers, the techniques developed for computing image flow do not produce results of the quality which will allow the valuable information to be recovered. Current methods for computing image flow, information which is critical in optical flow, and the recovery of such information are discussed in this section.

Definition 14.1 *Image flow is the velocity field in the image plane due to the motion of the observer, the motion of objects in the scene, or apparent motion which is a change in the image intensity between frames that mimics object or observer motion.*

14.4.1 Computing Image Flow

Image flow is determined by the velocity vector of each pixel in an image. Several schemes have been devised for calculating image flow based on two or more frames of a sequence. These schemes can be classified into two general categories: feature-based and gradient-based. If a stationary camera is used, most of the points in an image frame will have zero velocity. This is assuming

that a very small subset of the scene is in motion, which is usually true. Thus, most applications for image flow involve a moving camera.

14.4.2 Feature-Based Methods

Feature-based methods for computing image flow first select some features in the image frames and then match these features and calculate the disparities between frames. As discussed in an earlier section, the correspondence may be solved on a stereo image pair using relaxation. The same approach may be used to solve the correspondence problem in dynamic scenes. However, the problem of selecting features and establishing correspondence is not easy. Moreover, this method only produces velocity vectors at sparse points. This approach was discussed above as disparity analysis.

14.4.3 Gradient-Based Methods

Gradient-based methods exploit the relationship between the spatial and temporal gradients of intensity. This relationship can be used to segment images based on the velocity of points.

Suppose the image intensity at a point in the image plane is given as $E(x, y, t)$. Assuming small motion, the intensity at this point will remain constant, so that

$$\frac{dE}{dt} = 0. \tag{14.20}$$

Using the chain rule for differentiation, we see that

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0. \tag{14.21}$$

Using

$$u = \frac{dx}{dt} \tag{14.22}$$

and

$$v = \frac{dy}{dt}, \tag{14.23}$$

the relationship between the spatial and temporal gradients and the velocity components is:

$$E_x u + E_y v + E_t = 0. \tag{14.24}$$

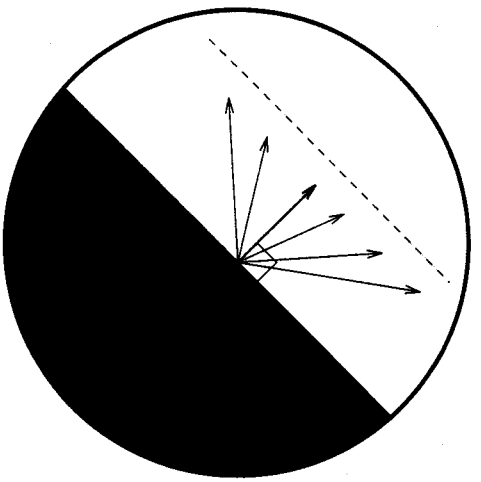


Figure 14.13: If one sees a point using a tube such that only one point is visible, then motion of the point cannot be determined. One can only get the sense of the motion, not the components of the motion vector. This problem is commonly called the *aperture problem.*

In the above equation, E_x, E_y, and E_t can be computed directly from the image. Thus, at every point in an image, there are two unknowns, u and v, and only one equation. Using information only at a point, image flow cannot be determined. This can be explained using Figure 14.13. This is known as the *aperture problem.* The velocity components at a point cannot be determined using the information at only one point in the image without making further assumptions.

It can be assumed that the velocity field varies smoothly over an image. Under this assumption, an iterative approach for computing image flow using two or more frames can be developed. The following iterative equations are used for the computation of image flow. These equations can be derived using the variational approach discussed below.

$$u = u_{\text{average}} - E_x \frac{P}{D} \tag{14.25}$$

$$v = v_{\text{average}} - E_y \frac{P}{D} \tag{14.26}$$

where

$$P = E_x u_{\text{average}} + E_y v_{\text{average}} + E_t \tag{14.27}$$

and

$$D = \lambda^2 + E_x^2 + E_y^2. \tag{14.28}$$

In the above equations E_x, E_y, E_t, and λ represent the spatial gradients in the x and y directions, the temporal gradient, and a constant multiplier, respectively. When only two frames are used, the computation is iterated over the same frames many times. For more than two frames, each iteration uses a new frame.

An important fact to remember about gradient-based methods is that they assume a linear variation of intensities and compute the point-wise velocities under this assumption. It is typically expected that this assumption is satisfied at edge points in images and, hence, the velocity can be computed at these points. The smoothness constraint is not satisfied at the boundaries of objects because the surfaces of objects may be at different depths. When overlapping objects are moving in different directions, the constraint will also be violated. These abrupt changes in the velocity field at the boundaries cause problems. To remove these problems, some other information must be used to refine the optical flow determined by the above method.

14.4.4 Variational Methods for Image Flow

Recall that the aperture problem says that the image-flow velocity at a point in the image plane cannot be computed by only using the changes in the image at that point without using information from other sources. Image flow can be computed using variational methods that combine the image flow constraint equation with an assumption about the smoothness of the image-flow velocity field. The image-flow constraint equation is

$$E_x u + E_y v + E_t = 0 \tag{14.29}$$

where u and v are the x and y components of the image-flow, respectively, and E_x, E_y, and E_t are the spatial and temporal derivatives of the image intensity. For a smoothness measure, use the sum of the squared magnitudes of each image flow component as the integrand in the regularization term:

$$\int\int \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right] dx\, dy. \tag{14.30}$$

Combine the smoothness measure with a measure of deviations from the problem constraint weighted by a parameter that controls the balance between deviations from the image-flow constraint and deviations from smoothness:

$$\int\int\left\{(E_x u+E_y v+E_t)^2+\nu^2\left[\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial u}{\partial y}\right)^2+\left(\frac{\partial v}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2\right]\right\}dx\,dy. \tag{14.31}$$

Use the calculus of variations to transform this norm into a pair of partial differential equations

$$\nu^2\nabla^2 u = E_x^2 u + E_x E_y v + E_x E_t \tag{14.32}$$

$$\nu^2\nabla^2 v = E_x E_y u + E_y^2 v + E_y E_t. \tag{14.33}$$

Use finite difference methods to replace the Laplacian in each equation with a weighted sum of the flow vectors in a local neighborhood, and use iterative methods to solve the difference equations.

14.4.5 Robust Computation of Image Flow

The motion information can be unreliable at motion boundaries, since the process of occluding or disoccluding the background does not obey the image-flow constraints of Equation 14.24. The incorrect motion constraints are outliers. Robust methods avoid the problems caused by incorrect motion constraints at boundaries.

Image flow can be computed using robust regression with least-median-squares regression. The least-median-squares algorithm is applied over successive neighborhoods. Within each neighborhood, the algorithm tries all possible pairs of constraint lines. The intersection of each pair of constraint lines is computed and the median of the square of the residuals is computed to assign a cost to each estimate. Each intersection and its cost are stored. After all possible pairs have been tried, the intersection corresponding to the minimum cost is used as the estimate for the image-flow velocity for the center of the neighborhood.

There are several steps in computing the intersection of the constraint lines and the residuals. The constraint lines are represented in polar form using the distance d of the constraint line from the origin in velocity space and the angle α of the image gradient:

$$d = \rho \cos(\alpha - \beta), \tag{14.34}$$

where $\rho(x, y)$ and $\beta(x, y)$ are the speed and direction of motion, respectively. Let the coordinates of the first constraint line be d_1 and α_1, and the coordinates of the second constraint line be d_2 and α_2. The position of the intersection in rectangular coordinates is

$$x = \frac{d_1 \sin \alpha_2 - d_2 \sin \alpha_1}{\sin(\alpha_1 - \alpha_2)} \tag{14.35}$$

$$y = \frac{d_2 \cos \alpha_1 - d_1 \cos \alpha_2}{\sin(\alpha_1 - \alpha_2)}. \tag{14.36}$$

The fit of the model to the constraint lines is the median of the squared residuals:

$$\underset{i}{\text{med}}\,(r_i^2). \tag{14.37}$$

The r residual between the motion estimate and each constraint line is the perpendicular distance of the constraint line from the estimate x and y. The residual is given by

$$r = x \cos \alpha + y \sin \alpha. \tag{14.38}$$

The position of the intersection of the pair of constraint lines, given by Equation 14.36, is a candidate solution. The median of the squared residuals of the constraint lines with respect to the candidate is computed and saved, along with the candidate, as a potential solution. The median of the squared residuals is the median of the square of the perpendicular distance of each constraint line in the neighborhood from the candidate.

The typical neighborhood size is 5×5. An $n \times n$ neighborhood contains n^2 constraint lines. The number of possible pairs of constraint lines in an $n \times n$ neighborhood would be

$$\frac{n^2(n^2 - 1)}{2}. \tag{14.39}$$

A 5×5 neighborhood would yield 300 pairs. It is not necessary to try all possible pairs if computation time is restricted. Rousseeuw and Leroy [207, p. 198] provide a table showing the number of trials that must be run to fit a model with p parameters and 95% confidence to data sets with various percentages of outliers. Assume that at most 50% of the constraints

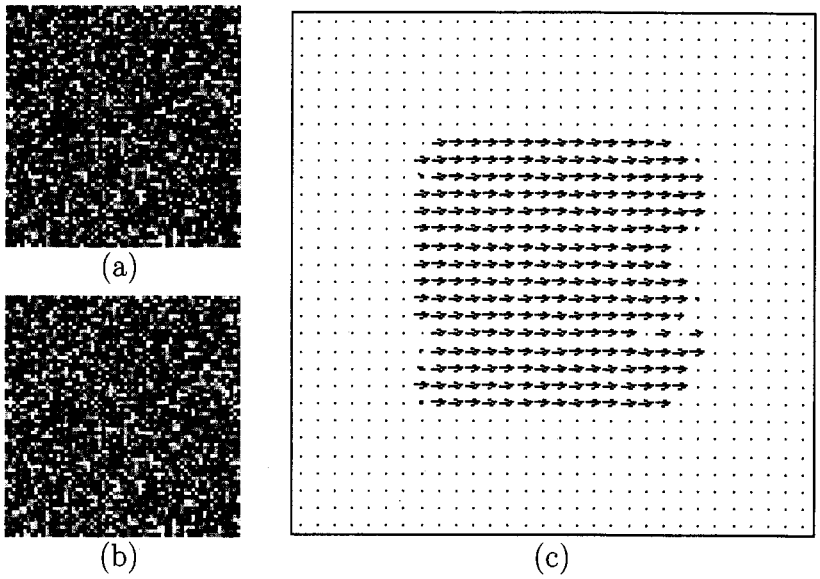


Figure 14.14: Image flow computed using the least-median-squares algorithm. Two frames of a synthetic image sequence were computed by filling a 64×64 background image and a 32×32 foreground image with pixels from a uniform random number generator. The foreground image was overlayed on the center of the background image to create the first frame and overlayed one pixel to the right to create the second frame.

in the neighborhood will be outliers. The local estimate of the image-flow velocity field requires only two constraint lines. From the table published by Rousseeuw and Leroy, only 11 pairs of constraints would have to be tried to provide a consistent estimate with 95% confidence. Using more pairs would increase the odds of finding a consistent estimate. If fewer than all possible pairs of constraint lines are used, the pairs should be selected so that the constraints in each pair are far apart. This reduces the problems with ill-conditioning caused by intersecting constraint lines that have nearly the same orientation. A preprogrammed scheme could be used for selecting the constraint line pairs in each neighborhood. The results using this approach are shown in Figure 14.14.

14.4.6 Information in Image Flow

Many researchers have studied the types of information that can be extracted from an image-flow field, assuming that high-quality image flow has been computed. Let us assume an environment which contains rigid, stationary surfaces at known depths, and that the observer, the camera, moves through this world. The image flow can be derived from the known structure. Thus, the structure of the environment can be obtained, in principle, from the computed image-flow field.

Areas with smooth velocity gradients correspond to single surfaces in the image and contain information about the structure of the surface. Areas with large gradients contain information about occlusion and boundaries, since only different objects at different depths can move at different speeds relative to the camera. Using an observer-based coordinate system, a relationship between the surface orientation and the smooth velocity gradients can be derived. The orientation is specified with respect to the direction of motion of the observer.

The translational component of object motion is directed toward a point in the image called the focus of expansion (FOE) when the observer is approaching or the focus of contraction when the observer is receding; see Figure 14.15. This point is the intersection of the direction of object motion in the image plane. Surface structure can be recovered from the first and second spatial derivatives of the translational component. The angular velocity is fully determined by the rotational component.

The importance of the FOE for recovering structure from the translational components of image flow encouraged several researchers to develop methods for determining the FOE. If the FOE is correctly determined, it may be used for computing the translational components of image flow. Since all flow vectors meet at the FOE, their direction is already known and only their magnitude remains to be computed. Thus, the two-dimensional problem of the computation of image flow is reduced to a one-dimensional problem. This fact has been noted by many researchers. However, it has not been applied, possibly due to the uncertainty in the proposed approaches for locating the FOE in real scenes.

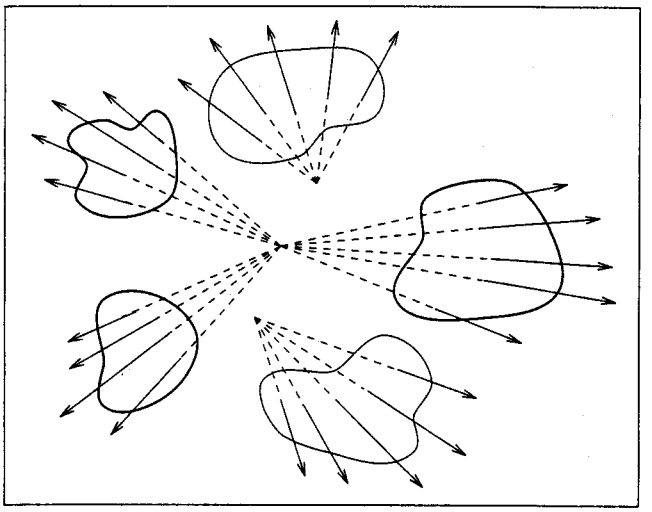


Figure 14.15: The velocity vectors for the stationary components of a scene, as seen by a translating observer, meet at the focus of expansion (FOE).

14.5 Segmentation Using a Moving Camera

If the camera is moving, then every point in the image has nonzero velocity relative to it.[1] The velocity of points relative to the camera depends both on their own velocity and on their distance from the camera. Difference picture–based approaches may be extended for segmenting moving camera scenes. Additional information will be required, however, to decide whether the motion at a point is due solely to its depth or is due to a combination of its depth and its motion. Gradient-based approaches will also require additional information.

If the camera's direction of motion is known, then the FOE with respect to the stationary components in a scene can easily be computed. The FOE will have coordinates

$$x'_f = \frac{dx}{dz} \tag{14.40}$$

and

$$y'_f = \frac{dy}{dz} \tag{14.41}$$

[1]With the exception of the pathological case where object points are moving with the same velocity as the camera.

in the image plane, where dx, dy, dz is the camera displacement between frames. As discussed in a later section, the velocity vectors of all the stationary points in a scene project onto the image plane so that they intersect at the FOE. A transformation with respect to the FOE may be used to simplify the task of segmentation. The ego-motion polar (EMP) transformation of an image transforms a frame $F(x', y', t)$ into $E(r', \theta, t)$ using

$$E(r', \theta, t) = F(x', y', t) \tag{14.42}$$

where

$$r' = \sqrt{(x' - x'_f)^2 + (y' - y'_f)^2} \tag{14.43}$$

and

$$\theta = \tan^{-1}\left(\frac{y' - y'_f}{x' - x'_f}\right). \tag{14.44}$$

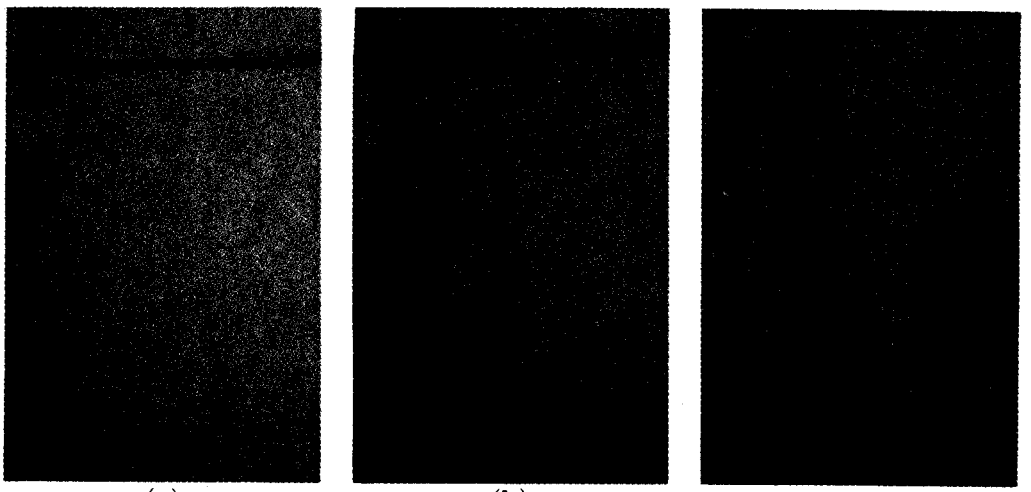
In EMP space, stationary points are displaced only along the θ axis between the frames of an image sequence, while points on moving objects are displaced along the r' axis as well as the θ axis. Thus, the displacement in the EMP space may be used to segment a scene into its stationary and non-stationary components. In Figure 14.16, three frames of a sequence acquired by a moving camera are shown. The results of the segmentation are shown in Figure 14.17.

A method for representing object motion in images, acquired by a moving camera, is derived from the fact that all of the velocity vectors of stationary objects in a scene acquired by a translating observer intersect at the FOE. An image frame may be transformed, with respect to the FOE, to a second frame in which the abscissa is r' and the ordinate is θ. Under this transformation, it is possible to segment a dynamic scene into its moving and stationary components, as discussed earlier.

14.5.1 Ego-Motion Complex Log Mapping

More information about moving, as well as stationary, objects may be extracted using a complex logarithmic mapping (CLM) rather than the simple polar mapping. Let us define

$$\omega = \log \alpha, \tag{14.45}$$



(a) (b) (c)

Figure 14.16: Three frames of a scene acquired by a moving camera are shown in (a), (b), and (c).

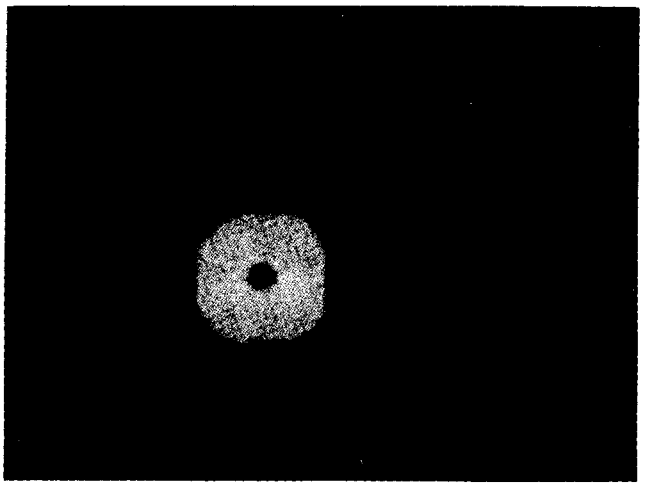


Figure 14.17: Moving objects are segmented from stationary objects, in the frames shown in Figure 14.16, using EMP segmentation. Moving objects appear brighter.

where ω and α are complex variables:

$$\alpha = x' + jy' = r'(\cos\theta + j\sin\theta) = r'e^{j\theta} \tag{14.46}$$

and

$$\omega = u(z) + jv(z). \tag{14.47}$$

Under this transformation, it can be shown that

$$u(r', \theta) = \log r' \tag{14.48}$$

$$v(r', \theta) = \theta. \tag{14.49}$$

The above results state that if the observer is moving, the horizontal displacement of a stationary point in CLM space depends only on the depth of the point and, furthermore, the vertical displacement is zero. This fact is very useful, not only in segmenting dynamic scenes into moving and stationary components, but also in determining the depth of stationary points. It has been shown that the retino-striate mapping can be approximated by a complex log function. This mapping is considered to be responsible for size, rotation, and projection invariances in biological systems. In the following discussion it is shown that if the mapping is obtained with respect to the FOE, rather than the center of the image, some other advantages can be achieved.

14.5.2 Depth Determination

First assume a camera is translating along its optical axis in a static world. For a stationary point in the environment, with real-world coordinates (x, y, z) relative to the observer at a time instant, the perspective projection, (x', y'), of this point onto the image plane is given by

$$x' = \frac{x}{z} \tag{14.50}$$

$$y' = \frac{y}{z} \tag{14.51}$$

assuming that the projection plane is parallel to the x–y plane at $z = 1$. The latter assumption simplifies the derivation without loss of generality. For translational motion along the direction of the gaze of the observer, the

relationship between the distance r' of the projection of the point from the center of the image and the distance z of the scene point from the observer is

$$\frac{dr'}{dz} = \frac{d\sqrt{x'^2 + y'^2}}{dz} = -\frac{r'}{z}. \tag{14.52}$$

By the chain rule for differentiation,

$$\frac{du}{dz} = \frac{du}{dr'}\frac{dr'}{dz}, \tag{14.53}$$

and from Equation 14.48,

$$\frac{du}{dr'} = \frac{1}{r'}. \tag{14.54}$$

Therefore, we have

$$\frac{du}{dz} = \frac{1}{r'} \cdot \left(-\frac{r'}{z}\right) = -\frac{1}{z}. \tag{14.55}$$

Similarly, to find dv/dz,

$$\frac{d\theta}{dz} = \frac{d(\tan^{-1}\frac{y'}{x'})}{dz} = 0 \tag{14.56}$$

and

$$\frac{dv}{dz} = \frac{dv}{d\theta}\frac{d\theta}{dz} = 0. \tag{14.57}$$

In Equation 14.55 we see that the depth, z, of a point can be determined from the horizontal displacement, du, in CLM space for that point and from the velocity, dz, of the observer. This is a formalization of the observable phenomenon that near objects appear to get bigger faster than far objects, as you approach them. Furthermore, the axial movement of the observer will result only in a horizontal change in the mapping of the image points, since $dv/dz = 0$. There will be no vertical movement of the mapped points. Thus, the correspondence of points between the two mapped images will become easier, since there is only a horizontal change in the points' locations over time. Now, assuming that there is sufficient control of the camera to be able to determine the amount of its movement, both variables necessary to determine image depths are readily available. Thus, it is possible to recover depth, in principle, if the camera motion is along its optical axis.

In real-life applications of machine vision it is not always possible to constrain the motion of the camera to be along the optical axis. Thus, the approach must be extended to arbitrary camera motion. To see that the depth can be recovered for an arbitrary translational motion of the camera, let us assume that the transform is taken with respect to the point (a, b) in the image plane. Then

$$r' = \sqrt{(x'-a)^2 + (y'-b)^2} \tag{14.58}$$

$$u = \log r' = \log \sqrt{(x'-a)^2 + (y'-b)^2}. \tag{14.59}$$

Now

$$\frac{du}{dz} = \frac{d}{dz} \log r' = \frac{1}{r'} \frac{dr'}{dz}. \tag{14.60}$$

Let us substitute for x' and y' from Equations 14.50 and 14.51, and evaluate dr'/dz.

$$\frac{dr'}{dz} = \frac{d\sqrt{(\frac{x}{z}-a)^2 + (\frac{y}{z}-b)^2}}{dz} \tag{14.61}$$

$$= \frac{1}{2\sqrt{(\frac{x}{z}-a)^2 + (\frac{y}{z}-b)^2}} \cdot \left[2\left(\frac{x}{z}-a\right) \cdot \frac{z\frac{dx}{dz} - z}{z^2} + 2\left(\frac{y}{z}-b\right) \cdot \frac{z\frac{dy}{dz} - y}{z^2}\right] \tag{14.62}$$

$$= \frac{1}{\sqrt{(\frac{x}{z}-a)^2 + (\frac{y}{z}-b)^2}} \cdot \frac{1}{z} \cdot \left[\left(\frac{x}{z}-a\right)\left(\frac{dx}{dz} - \frac{x}{z}\right) + \left(\frac{y}{z}-b\right)\left(\frac{dy}{dz} - \frac{y}{z}\right)\right]. \tag{14.63}$$

Hence

$$\frac{du}{dz} = \frac{1}{(\frac{x}{z}-a)^2 + (\frac{y}{z}-b)^2} \cdot \frac{1}{z} \cdot \left[\left(\frac{x}{z}-a\right)\left(\frac{dx}{dz} - \frac{x}{z}\right) + \left(\frac{y}{z}-b\right)\left(\frac{dy}{dz} - \frac{y}{z}\right)\right]. \tag{14.64}$$

If (a, b) is an arbitrary point in the image, the above equations are complicated and require detailed knowledge of the relationship between the camera

and the objects in the scene. However, if we let (a, b) be the focus of expansion, the equations become greatly simplified. The focus of expansion (FOE) is an important point on the image plane. If a camera moves toward the objects in the scene, the objects appear to get bigger. If the vectors that represent this expansion of the objects in the image plane are extended, they will all meet in a single point, the FOE. If the camera is moving away from the scene, the objects will seem to get smaller. In this case the point at which the extended vectors meet is called the focus of contraction, the FOC. For either case, it is the pixel where the path of the translating camera pierces the image plane. If (a, b) is the FOE, then

$$a = \frac{dx}{dz} \text{ and } b = \frac{dy}{dz}. \tag{14.65}$$

Substituting for dx/dz and dy/dz

$$\frac{du}{dz} = \frac{1}{(\frac{x}{z}-a)^2+(\frac{y}{z}-b)^2} \cdot \frac{1}{z} \cdot \left[-\left(\frac{x}{z}-a\right)^2 - \left(\frac{y}{z}-b\right)^2\right] \tag{14.66}$$

$$= -\frac{1}{z}. \tag{14.67}$$

Now let us examine dv/dz, when v is calculated with respect to the FOE (a, b):

$$v = \theta = \tan^{-1}\left(\frac{y'-b}{x'-a}\right) \tag{14.68}$$

$$\frac{dv}{dz} = \frac{1}{1+\left(\frac{y'-b}{x'-a}\right)^2} \cdot \frac{d}{dz}\left(\frac{y'-b}{x'-a}\right). \tag{14.69}$$

Considering only the second factor of this equation, and substituting for x' and y',

$$\frac{d}{dz}\left(\frac{y'-b}{x'-a}\right) = \frac{d}{dz}\frac{(\frac{y}{z}-b)}{(\frac{x}{z}-a)} \tag{14.70}$$

$$= \frac{(\frac{x}{z}-a)(z\frac{dy}{dz}-y)\frac{1}{z^2} - (\frac{y}{z}-b)(z\frac{dx}{dz}-x)\frac{1}{z^2}}{(\frac{x}{z}-a)^2} \tag{14.71}$$

$$= \frac{(\frac{x}{z}-a)(\frac{dy}{dz}-\frac{y}{z}) - (\frac{y}{z}-b)(\frac{dx}{dz}-\frac{x}{z})}{z(\frac{x}{z}-a)^2}. \tag{14.72}$$

Remembering that $dx/dz = a$ and $dy/dz = b$,

$$\frac{d}{dz}\left(\frac{y'-b}{x'-a}\right) = \frac{(\frac{x}{z}-a)(b-\frac{y}{z})-(\frac{x}{z}-a)(b-\frac{y}{z})}{z(\frac{x}{z}-a)^2} \tag{14.73}$$

$$= 0. \tag{14.74}$$

Therefore,

$$\frac{dv}{dz} = 0. \tag{14.75}$$

Note that when the mapping is done with respect to the FOE, then the displacement in the u direction depends only on the z coordinate of the point. For other values of (a, b) the above property will not be true. Thus, if the FOE is known or can be computed, the approach can be applied to an arbitrarily translating camera, as long as there is some movement in depth (dz is in the denominator and so cannot be zero). This extension is called the ego-motion complex logarithmic mapping (ECLM) since it is based on the motion parameters of the camera itself.

14.6 Tracking

In many applications, an entity, a feature or an object, must be tracked over a sequence of frames. If there is only one entity in the sequence, the problem is easy to solve. In the presence of many entities moving independently in a scene, tracking requires the use of constraints based on the nature of objects and their motion. Due to inertia, the motion of a physical entity cannot change instantaneously. If a frame sequence is acquired at a rate such that no dramatic change takes place between two consecutive frames, then for most physical objects, no abrupt change in motion can be observed. The projection of a smooth three-dimensional trajectory is also smooth in the two-dimensional image plane. This allows us to make the smoothness assumption in images. This property is used to formulate *path coherence*. Path coherence implies that the motion of an object at any point in a frame sequence will not change abruptly.

We can combine the solution of the correspondence problem for stereopsis and motion. The following three assumptions help in formulating an approach to solve the correspondence problem:

- The location of the given point will be relatively unchanged from one frame to the next frame.
- The scalar velocity of a given point will be relatively unchanged from one frame to the next.
- The direction of motion of a given point will be relatively unchanged from one frame to the next frame.

We can also use the smoothness of image motion in monocular image sequences. To formalize the idea of the smoothness of motion, let us first formalize the idea of path coherence which states that the motion of an object at any time instant cannot change abruptly.

14.6.1 Deviation Function for Path Coherence

To use the above properties in an algorithm, we formulate a function to implement the above ideas concretely and use it to evaluate motion properties in a frame sequence. The path coherence function should follow these four guiding principles:

1. The function value is always positive.
2. It should consider the amount of angular deviation without any effect of the sign of the direction of motion.
3. The function should respond equally to the incremental speed.
4. The function should be normalized in the range 0.0 to 1.0.

Trajectories of two point-tokens are shown in Figure 14.18. Let the trajectory be represented as

$$T_i = \left\langle P_i^1, P_i^2, P_i^3, \ldots, P_i^n \right\rangle \tag{14.76}$$

where T_i is trajectory and P_i^k represents a point in the kth image. If the coordinates of the point are given by the vector X_i in the frame, the coordinates in the kth frame can be represented as X_{ik}. Representing the trajectory in vector form,

$$T_i = \langle X_{i1}, X_{i2}, X_{i3}, \ldots, X_{in} \rangle . \tag{14.77}$$

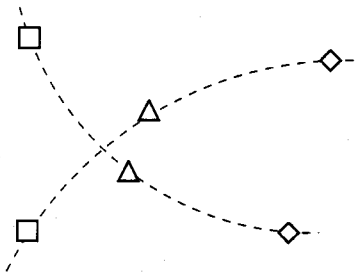


Figure 14.18: The trajectories of two points. The points in the first, second, and third frames are labeled □, △, and ◇, respectively.

Now let us consider the deviation ${d_i}^k$ in the path of the point in the kth frame. The deviation in the path is a measure of path coherence, given by

$$ {d_i}^k = \phi(\overline{X_{ik-1}X_{ik}}, \overline{X_{ik}X_{ik+1}}), \tag{14.78} $$

where ϕ is a path coherence function. The deviation for the complete trajectory is defined as

$$ D_i = \sum_{k=2}^{n-1} {d_i}^k. \tag{14.79} $$

If there are m points in a sequence of n frames resulting in m trajectories, the deviation of all trajectories should be considered, which is given by

$$ D(T_1, T_2, T_3, \ldots, T_m) = \sum_{i=k}^{m} \sum_{k=2}^{n-1} {d_i}^k. \tag{14.80} $$

Thus, the correspondence problem is solved by maximizing the smoothness of motion; that is, the total deviation D is minimized to find the set of correct trajectories.

14.6.2 Path Coherence Function

After understanding the trajectory function, let us define a constraint function for path coherence. If the sampling rate of the camera is high enough, then change in the direction and velocity of any moving point in consecutive time frames is smooth.

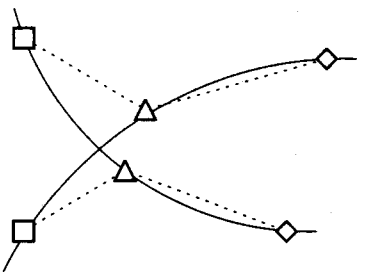


Figure 14.19: Deviations in a trajectory.

This is described by the deviation function:

$$\phi(P_i^{k-1}, P_i^k, P_i^{k+1}) = w_1(1 - \cos\theta) + w_2\left(1 - 2\frac{\sqrt{d_1 d_2}}{d_1 + d_2}\right). \qquad (14.81)$$

In the vector form it can represented as

$$\phi(P_i^{k-1}, P_i^k, P_i^{k+1}) = w_1\left(1 - \frac{\overline{X_{ik-1}X_{ik}} \cdot \overline{X_{ik}X_{ik+1}}}{\|\overline{X_{ik-1}X_{ik}}\|\|\overline{X_{ik}X_{ik+1}}\|}\right) + w_2\left(1 - 2\,\frac{\sqrt{\|\overline{X_{ik-1}X_{ik}}\|\|\overline{X_{ik}X_{ik+1}}\|}}{\|\overline{X_{ik-1}X_{ik}}\| + \|\overline{X_{ik}X_{ik+1}}\|}\right) \qquad (14.82)$$

where w_1 and w_2 are weights that are selected to assign differing importance to direction and velocity changes (see Figure 14.19).

Note that the first term is the dot product of displacement vectors and the second considers the geometric and arithmetic mean of the magnitude. The first term in the above expression can be considered as *direction coherence*, and the second term can be considered as *speed coherence*. The weight can be selected in the range 0.00 to 1.00 such that their sum is 1.

One of the main difficulties with the use of multiple frames is the problem of occlusion. When working with a large sequence of frames, it is possible that some objects may disappear totally or partially. Similarly, some new objects may appear in the frame sequence from some intermediate frame onward. In addition, the changing geometry of objects due to motion and the changes in scene illumination over the frame sequence can cause significant changes in the feature point data that is to be matched over the frames.

All these changes in the feature point data set lead to incorrect correspondence, if such a correspondence is obtained by performing minimization to extract a globally smooth set of complete trajectories. By forcing the trajectories to satisfy some local constraints and allowing them to be incomplete, if necessary, trajectories in the presence of occlusion can be obtained.

14.6.3 Path Coherence in the Presence of Occlusion

Two limitations of the path coherence algorithm are:

- It assumes that the same number of feature points are available in every frame.
- It assumes that the same set of feature points are being extracted in every frame.

In practice, the number of feature points can drastically change from frame to frame. Even when the feature point count is the same over all the frames, it may not necessarily be the same set of feature points that was extracted in the earlier frames. Thus, while searching for a smooth set of trajectories, we must allow the possibility of obtaining incomplete trajectories indicating either occlusion, appearance of a new object, or simply the absence of the corresponding feature points in the subsequent or earlier frames due to poor feature detection. Moreover, some constraints in the form of maximum possible displacement and maximum local smoothness deviation must be placed on the acceptable trajectories to avoid chance groupings of distant feature points that may happen to form a smooth path.

Given a set P^j of m_j feature points for each of the n frames, the algorithm below finds the maximal set of complete or partially complete trajectories that minimizes the sum of local smoothness deviations for all the trajectories obtained subject to the conditions that the local smoothness deviation for none of the trajectories exceeds $\phi_{\max}$ and the displacement between any two successive frames for any trajectory is always less than $d_{\max}$. The above local constraints limit the acceptable location of a feature point in the next frame given its location in two previous frames.

14.6.4 Modified Greedy Exchange Algorithm

To account for missing points due to occlusion, *phantom points* are used. Phantom feature points are hypothetical points which are used as fillers to extend all trajectories over the given frame set. These points are introduced during the trajectory initialization phase as described later. The notion of phantom feature points serves two purposes: it allows us to satisfy the local constraints, and it provides a way to deal with incomplete trajectories. For the notion of phantom feature points to be useful, we must define the displacement and local smoothness deviation values for a trajectory that has some phantom feature points assigned to it. For computing the frame-to-frame displacement for a trajectory T_i, a displacement function $\text{Disp}(P_i^k, P_i^{k+1})$ is defined as follows:

$$\text{Disp}(P_i^k, P_i^{k+1}) = \begin{cases} \text{Euclidean Disp}(P_i^k, P_i^{k+1}) & \text{if both points are true feature points} \\ d_{\max} & \text{otherwise.} \end{cases}$$

This definition of frame-to-frame displacement for a trajectory implies that a phantom feature point always moves by a fixed amount $d_{\max}$. For computing the local smoothness deviation for a trajectory T_i in the kth frame function, $\text{DEV}(P_i^{k-1}, P_i^k, P_i^{k+1})$ is defined as follows:

$$\text{DEV}(P_i^{k-1}, P_i^k, P_i^{k+1}) = \begin{cases} 0 & \text{if } P_i^{k-1} \text{ is a phantom point} \\ \phi(P_i^{k-1}, P_i^k, P_i^{k+1}) & \text{if all three are true feature points} \\ \phi_{\max} & \text{otherwise.} \end{cases}$$

The above definition of the local smoothness deviation function is equivalent to the path coherence function if all three feature points are true feature points. It introduces a penalty of $\phi_{\max}$ for not having a true feature for T_i in the subsequent frame or missing a true feature point in the current frame, the kth frame. There is no penalty if a trajectory begins from the current frame under consideration. However, as the greedy exchange algorithm is applied alternately in the forward and backward directions, it is clear that the assignment of phantom feature points in the subsequent frames or earlier frames is equally discouraged. With the above definitions for computing the displacement and local smoothness deviation values, the steps of the modified greedy exchange algorithm are as follows:

Initialization:

1. For each of the m_k true feature points in $P^k, k = 1, 2, \ldots, n-1$, determine the nearest neighbor in P^{k+1} that is within the distance $d_{\max}$. Resolve arbitrarily in case of multiple choices.

2. Form initial trajectories by linking the nearest neighbors in the successive frames. Extend all incomplete trajectories using phantom feature points to span n frames.

3. For every trajectory of step 2 above, form an additional trajectory consisting only of phantom feature points.

Exchange Loop:

forward-flag = on;
backward-flag = on;
For each frame index $k = 2$ to $n-1$:
  while (forward-flag == on or backward-flag = on)
  do
    if (forward-flag == on) then
    begin
      for $i = 1$ to $m-1$
        for $j = i+1$ to m
          if within constraints of $d_{\max}$
            calculate:

$$G_{ij}{}^k = [\phi(P_i{}^{k-1}, P_i{}^k, P_i{}^{k+1}) + \phi(P_j{}^{k-1}, P_j{}^k, P_j{}^{k+1})] - [\phi(P_i{}^{k-1}, P_i{}^k, P_j{}^{k+1}) + \phi(P_j{}^{k-1}, P_j{}^k, P_i{}^{k+1})]. \quad (14.83)$$

      Pick the ij pair with maximum gain.
      if (gain is greater than 0)
      begin
        Exchange the point in $(k+1)$th frame.
        Set backward-flag on.
      end (if)
      else
        Set forward-flag off.
    end (if)

```
        else if (backward-flag == on) then
        begin
            for i = 1 to m − 1
                for j = i + 1 to m
                    if within constraints of d_max
                        calculate G_ij^k from Equation 14.83.
            Pick the ij pair with maximum gain.
            if (gain is greater than 0)
            begin
                Exchange the point in (k + 1)th frame.
                Set backward-flag on.
            end (if)
            else
                Set forward-flag off.
        end (if)
    end (while)
end (for)
```

Termination:

Repeat the exchange loop until there are no more frames.

The exchange operation of the above algorithm must be applied alternately in the forward and backward directions. The number of trajectories formed during step 2 of the initialization depends on the quality of the data. In an ideal case, where the same set of m feature points are consistently present in all the frames, only m trajectories will be formed and none of these will have any phantom points. In the worst case, when the feature points of any one frame do not have any correlation with the feature points of any other frame, the number of trajectories formed can be as large as the total count of true feature points from all the frames. In general, the number of trajectories formed will be at least $m_{\max}$, where $m_{\max} = \max(m_1, m_2, \ldots, m_n)$, and different trajectories will have different numbers of phantom feature points.

It should be noted that the introduction of phantom feature points does not require the locations of these points. Also, the introduction of trajectories with only phantom feature points during step 3 of the initialization phase does not affect the overall sum of local smoothness deviation values,

the criterion being minimized to determine the correspondence. The introduction of these trajectories ensures that no final trajectory will have a local smoothness deviation value greater than $\phi_{\max}$ in any frame. The checking of the ij pairs in the exchange loop ensures that the $d_{\max}$ constraint will not be violated.

14.7 Shape from Motion

One of the major goals of dynamic scene analysis is to get three-dimensional information about the structure of objects in the scene and their three-dimensional motion characteristics. By the structure of an object, we mean the relative locations of points on the object. Thus, if we know the locations of points on an object with respect to a *reference point*, with a scale factor, then we say that the structure of the object is known. To get this information, however, image plane information about objects must be converted to scene information. As we saw in stereo, by using multiple views of an object, or by using multiple locations of a camera, this information may be recovered. The interpretation of two-dimensional displacements in terms of three-dimensional motion is complicated.

An assumption about the *rigidity* of objects is helpful in recovering the structure of objects. The rigidity assumption states that any set of elements undergoing a two-dimensional transformation, which has a unique interpretation as a rigid body moving in space, should be so interpreted. This assumption provides a much-needed constraint in the scene that can be used to determine the relative locations of points. The *structure-from-motion* theorem states that given three distinct orthographic projections of four noncoplanar points in a rigid configuration, the structure and motion compatible with the three views are uniquely determined up to a reflection about the image plane.

The structure-from-motion theorem provides a precise statement about constraints and possible solutions. By changing constraints on the types of objects and types of projections, and by using different mathematical approaches, many different formulations of the above problem are possible. Approaches to recovering three-dimensional structures from image sequences may be divided into the following two general classes.

Token-Based Methods

Suppose that we have several views of an object available and that these views can be considered orthographic. An interest operator is applied to consecutive frames of this sequence and some interesting points or tokens, such as corners, are extracted. Also suppose that the correspondence problem between interesting points has been solved, using a method discussed earlier. If token correspondence has been established, the structure-from-motion theorem states that it is possible to recover the three-dimensional location of four noncoplanar points and their motion from their orthogonal projections. This gives an implicit three-dimensional structure for an object. Here we will consider an approach to recover this structure.

Suppose that we have four points in space and their orthographic projections are known in three frames. Let the points be P_0, P_1, P_2, and P_3 and the frames be taken at time instants t_1, t_2, and t_3. We represent three-dimensional coordinates of points as (x_{ij}, y_{ij}, z_{ij}) and two-dimensional coordinates as (x'_{ij}, y'_{ij}), where i and j represent the point number, $i = 0, 1, 2, 3$, and frame number, $j = 1, 2, 3$, respectively. Now we can write the transformation of three-dimensional points as

$$\begin{bmatrix} x_{i,2} \\ y_{i,2} \\ z_{i,2} \end{bmatrix} = R_{12} \begin{bmatrix} x_{i,1} \\ y_{i,1} \\ z_{i,1} \end{bmatrix} + T_{12} \tag{14.84}$$

and

$$\begin{bmatrix} x_{i,3} \\ y_{i,3} \\ z_{i,3} \end{bmatrix} = R_{23} \begin{bmatrix} z_{i,2} \\ y_{i,2} \\ z_{i,2} \end{bmatrix} + T_{23}. \tag{14.85}$$

In the above, R_{ij} and T_{ij} are rotation matrices and translation vectors from time t_i to time t_j. We want to determine rotations matrices, translation vectors, and three-dimensional coordinates of points to determine the motion and structure. To solve for these quantities, we have image plane coordinates of four points in three frames. Now since we are using orthographic projections, we know that

$$x'_{ij} = x_{ij} \quad \text{and} \quad y'_{ij} = y_{ij}. \tag{14.86}$$

Upon substituting this information in the above equations, we see that this is an underconstrained situation; we have too few equations to determine the required variables.

Many different approaches have been proposed to solve this underconstrained problem. Most of these methods introduce an assumption that allows the formulation of the problem in such a way that one of the known techniques for determining the unknowns may be applied. We do not discuss any specific method here. Interested readers may want to study the references given in the Further Readings section.

Feature-based methods for the recovery of three-dimensional structure or for the estimation of motion parameters require two steps: determination of the precise location of tokens or points, and the correspondence between tokens or points. If interest operators are applied based on small neighborhoods, then the number of tokens extracted in a real image is very large, making correspondence a difficult problem. Interest operators based on a large neighborhood and higher-order gray-level characteristics result in a more reasonable number of tokens for determining correspondences, but the location of tokens may not be precise.

Trajectory-Based Methods

The above methods depend on a set of points in two or three frames. If a token is tracked over several frames by solving correspondences, a two-dimensional trajectory of the token is obtained. Methods for recovering three-dimensional structure and motion from trajectories are more reliable than methods based on sets of features in new frames. A trajectory may be interpolated to obtain better resolution of the two-dimensional path by using curve fitting techniques. Moreover, the correspondence problem may be simplified by considering more than two frames and extending relaxation across the frames.

Further Reading

Nagel [174] proposed the use of the likelihood ratio for motion detection. Much of the work presented here on difference and accumulative difference pictures was done by Jain with other researchers [129, 130, 122, 125].

Fennema and Thompson proposed use of the gradient method to compute optical flow for use in segmentation based on motion [79]. Horn and Schunck [114] developed the optical flow determination algorithm. Schunck

[212] developed robust approaches to compute optical flow. Nagel and Enkelmann [176] formulate and evaluate an oriented smoothness constraint for computing optical flow. Determination of optical flow has been a very active research area. For some recent approaches see [4, 106, 169, 175, 9, 222]. Clocksin [58] and Longuet-Higgins and Prazdny [157] describe the significance of optical flow and its potential in extracting information about surface orientation and structure. Subbarao [229] has presented a rigorous approach for recovering information from optical flow.

Barnard and Thompson [21] introduced a feature-based matching algorithm to estimate geometrical disparity between images. Recovery of three-dimensional motion parameters and the three-dimensional structure of objects has been an active research area. Ullman published a book containing an extensive discussion of the correspondence problem from the viewpoint of natural vision [241]. He popularized the *structure-from-motion* problem. Later Huang, with his students, did pioneering research in recovering motion characteristics of objects under various conditions [237, 77, 76, 249]. Jerian and Jain showed the sensitivity of various approaches to noise and initial estimates and provided a polynomial systems solution [134, 133]. Chellappa and Broida developed a framework for estimating motion in a sequence of frames using Kalman filters [47, 48].

Ego-motion complex logarithmic mapping was developed by Jain [124, 123, 127]. Schwartz [214, 213, 215, 216] and Cavanaugh [54, 55] have studied this mapping in the context of biological vision systems.

Jenkin [132] proposed a novel approach for combining the solution of the correspondence problem for stereopsis and motion. Sethi and Jain [219] developed a tracking algorithm based on smoothness of motion. The algorithm for path coherence in the presence of occlusion was developed by by Salari and Sethi [209]. A large number of image frames taken at short intervals helps minimize the correspondence problem since the amount of change in successive images is expected to be very small. This concept has led to the so-called epipolar plane image analysis in which the images are acquired using a moving camera. Explicit representation of both the spatial and temporal structure of such image sequences is captured in a spatiotemporal surface. Tracking mechanisms that operate locally on these evolving surfaces to obtain three-dimensional scene reconstruction are described by Baker and Bolles in [15]. In the last few years, there have been several other approaches based on the spatiotemporal image solid. Some of these are [106, 247, 156].

Determination of optical flow and solving the correspondence problem have been two difficult problems in dynamic vision. In the last few years, several techniques have been proposed for direct computation of motion properties, bypassing optical flow and correspondence [126, 7, 6, 184]. These approaches appear promising.

Zhang, Faugeras, and Ayache [262] approach the problem of motion determination as a stereo matching and motion estimation problem. Stereo and motion are also used by Grosso, Sandini, and Tistarelli in their system described in [94]. However, in their system, the objects remain stationary, and multiple views are obtained by moving the cameras around the objects while maintaining the direction of gaze fixed toward a point in space. A survey of many of the methods for dynamic-scene analysis is given by Aggarwal and Nandhakumar [2]. A framework for the abstraction of motion concepts from image sequences is described by Tsotos et al. [238]. It includes semantic nets for knowledge representation and associated algorithms operating in a competing and cooperating feedback mode.

Exercises

14.1 What are the three phases in a dynamic-scene analysis system? Consider the task of a driver on a freeway. What are the operations performed in the three phases of dynamic-scene analysis for this task?

14.2 Define a difference picture. How would you select an appropriate threshold in a particular application of a difference picture?

14.3 How can you form a difference picture using a statistical approach to determine dissimilarity of intensity values in an image? How is this approach better than the one based on the straightforward difference picture approach?

14.4 What is an accumulative difference picture? What limitations of difference pictures does it overcome? Does sign of difference help in motion analysis? How?

14.5 What is the difference between a time-varying edge and a moving edge? How will you detect moving edges in a dynamic scene? Can

you use three-dimensional edge detectors to detect motion in an image sequence?

14.6 Define the correspondence problem in motion analysis. How can you solve this problem? List various approaches to solve this problem. What do you think is the problem that makes determination of corresponding points so difficult?

14.7 To obtain the feature point locations to subpixel resolution, compute the x and y moments from the intermediate image of the minimum variance obtained in step 2 of the Moravec interest operator.

14.8 What is image flow? Where can it be used?

14.9 Derive the motion constraint equation. How can you use this equation to segment a dynamic scene into moving and stationary objects?

14.10 Define the aperture problem. What problems does it cause in computing image flow? How can you overcome the aperture problem? Suggest at least two different approaches.

14.11 Define focus of expansion. Where and how it can be used? Is it related to vanishing points?

14.12 What is complex logarithmic mapping? What properties of this mapping make it attractive in dynamic-scene analysis?

14.13 When a camera is moving, all points in the environment are getting displaced in an image. How can you segment a dynamic scene to determine moving objects in case of a moving camera?

14.14 What is path coherence? How can you use it for tracking objects in a scene?

14.15 What is structure-from-motion? Under what conditions can you realistically determine the structure of an object using three frames?

14.16 The top left corner of a 6×8-pixel object is initially located at $(0, 0)$. It moves to the location $(4, 8)$ after four frames of uniform velocity. Find the absolute, positive, and negative accumulative difference pictures.

14.17 Consider a point object located at *world* coordinates $(10, 0, 10)$ at time $t = 0$. A camera system with a focal length of 2 is located such that its *lens center* is at the origin of the world coordinate system and its optical axis is looking directly at the object at time $t = 0$. The object is moving with a uniform velocity of $(5, 0, 0)$.

a. What are the coordinates of the point object in the image plane at time $t = 0$?

b. Find the image coordinates of the point at $t = 1$ if

(i) The camera is stationary.

(ii) The camera translates with a uniform velocity of $(0, 0, 5)$.

14.18 A family decides to enjoy a nice Spring weekend outdoors along with their pets, Cilly the cat and Billy the dog. Just when they had their new video camera set up on a tripod and leveled so that its axis is horizontal, Cilly decides to go after a songbird feasting on some bread crumbs. Immediately, Billy decides to chase Cilly. After four seconds of running at constant speed, both Billy and Cilly reach their destination only to find that the bird just took off in the direction $(0, 0, 200)$. Initial positions of all animals and the camera are as shown in Figure P14.18. The camera height is 5 feet and the focal length of its lens is 0.25 feet. Assume that the animals are small enough so that they can be represented as points on the ground. Also, assume that Billy's instantaneous direction of travel is always toward the current position of Cilly and that he also travels at constant speed. Find the image plane coordinates and the direction of optical flow for all animals at time instances $t = 0^+$ (just after start), $t = 2$ (an estimate is enough for Billy's position), and $t = 4^-$ (just before end) seconds. Also mark the direction of optical flow for the bird after it starts its flight. Mark all distances in the image plane. Note that all distances are measured from the center of the lens for convenience.

Computer Projects

14.1 Design a system to count how many people entered the coffee room and what percentage took coffee. Assume that you can use multiple

cameras if required and you are free to determine the location and orientation of cameras.

14.2 Implement an accumulative difference picture–based approach for motion analysis. Use this approach to extract images of all people who came to your coffee room and took coffee.

14.3 Develop a program to establish correspondence. Use this to determine all corresponding points in a scene containing at least three objects moving in different directions.

14.4 Implement a tracking algorithm using path coherence. Test it using a basketball sequence to track the ball. Modify this algorithm to work with a mobile camera. Assume that you want to design a mobile robot that will track a particular moving object (pointed to it interactively). Apply your tracking algorithm to this problem.

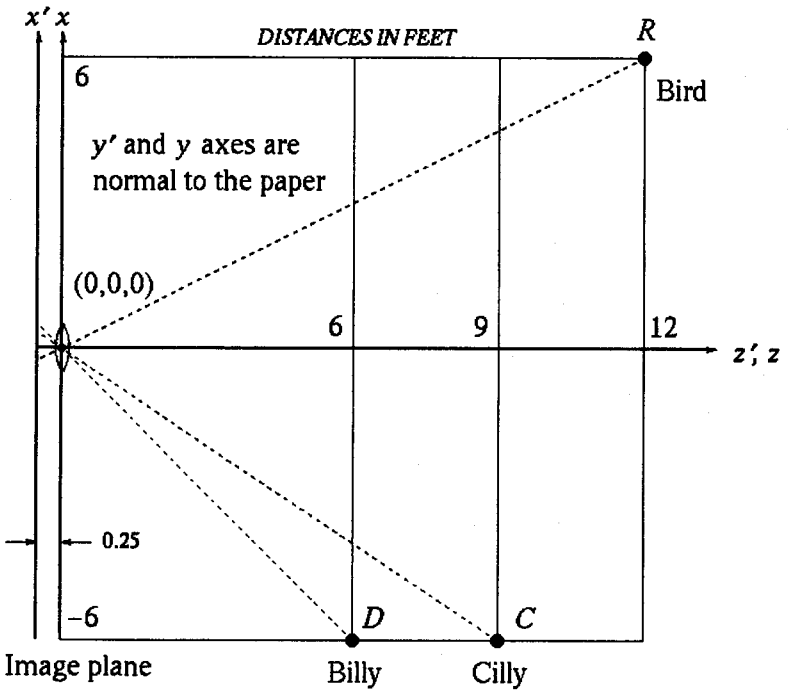


Figure P14.18