Representing Moving Images with Layers

John Y. A. Wang and Edward H. Adelson, *Member, IEEE*

***Abstract*— We describe a system for representing moving images with sets of overlapping layers. Each layer contains an intensity map that defines the additive values of each pixel, along with an alpha map that serves as a mask indicating the transparency. The layers are ordered in depth and they occlude each other in accord with the rules of compositing. Velocity maps define how the layers are to be warped over time. The layered representation is more flexible than standard image transforms and can capture many important properties of natural image sequences. We describe some methods for decomposing image sequences into layers using motion analysis, and we discuss how the representation may be used for image coding and other applications.**

I. Introduction

An image coding system involves three parts: the encoder, the representation, and the decoder. The representation is the central determinant of the overall structure of the coding system. The most popular image coding systems today are based on "low-level" image processing concepts such as DCT's, subbands, etc. It may ultimately be possible to encode images using "high-level" machine vision concepts such as 3-D object recognition, but it will be many years before such techniques can be applied to arbitrary images. We believe that a fruitful domain for new image coding lies in "mid-level" techniques, which involve concepts such as segmentation, surfaces, depth, occlusion, and coherent motion. We describe one such representation based on "layers" and show how it may be applied to the coding of video sequences.

Consider the language used by a traditional cel animator. First a background is painted and then a series of images are painted on sheets of clear celluloid (the "cels"). As a cel is moved over the background it occludes and reveals different background regions. A similar representation is used in computer graphics where a set of images can be composited with the aid of "alpha channels" that serve as masks to indicate the transparency and opacity of the overlying layers.

In traditional cel animation one is restricted to rigid motions of the backgrounds and the cels. In a digital system, however, it is easy to use more complex motions such as affine transformations, which include all combinations of translation, rotation, dilation, and shear. Such representations will be successful insofar as the image model offers an adequate description of the motions found in the original sequence.

Manuscript received April 17, 1992; revised December 28, 1993. This work was supported in part by contracts with the Television of Tomorrow Program, SECOM Co., and Goldstar Co. The associate editor coordinating the review of this paper and approving it for publication was Dr. M. Ibrahim Sezan.

J. Y. A. Wang is with the Department of Electrical Engineering and Computer Sciences and the MIT Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

E. H. Adelson is with the Department of Brain and Cognitive Sciences and the MIT Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

IEEE Log Number 9402957.

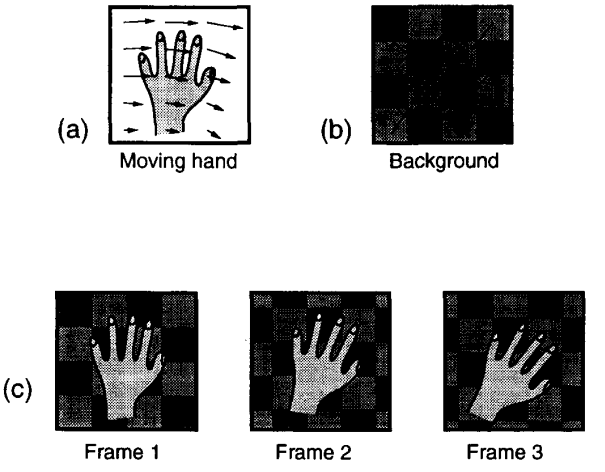


Fig. 1. The objects involved in a hypothetical scene of a moving hand: (a) the hand, which undergoes a simple motion; (b) the background, which is translating down and to the left; (c) the observed image sequence.

Fig. 1 illustrates the concept. Fig. 1(a) shows an image sequence of a waving hand moving against a moving background, shown in Fig. 1(b). Suppose that both the hand and the background execute simple motions as shown. The resultant image sequence is shown in Fig. 1(c).

Given this sequence, we wish to invert the process by which it was generated. Thus we would like to decompose the sequence into a set of layers which can be composited so as to generate the original sequence. Since the world consists of stable objects undergoing smooth motions, our decomposition should also contain stable objects undergoing smooth motions.

In the representation that we use, following Adelson [1], each layer contains three different maps: (1) the intensity map, (often called a "texture map" in computer graphics); (2) the alpha map, which defines the opacity or transparency of the layer at each point; and (3) the velocity map, which describes how the map should be warped over time. In addition, the layers are assumed to be ordered in depth.

Fig. 2 shows the layered decomposition of the hand sequence. The hand and background layers are shown in Figs. 2(a) and 2(b), respectively. The resynthesized sequence is shown in Fig. 2(c).

Let us note that traditional motion analysis methods fall short of what is needed. Optic flow techniques typically model the world as a 2-D rubber sheet that is distorted over time.

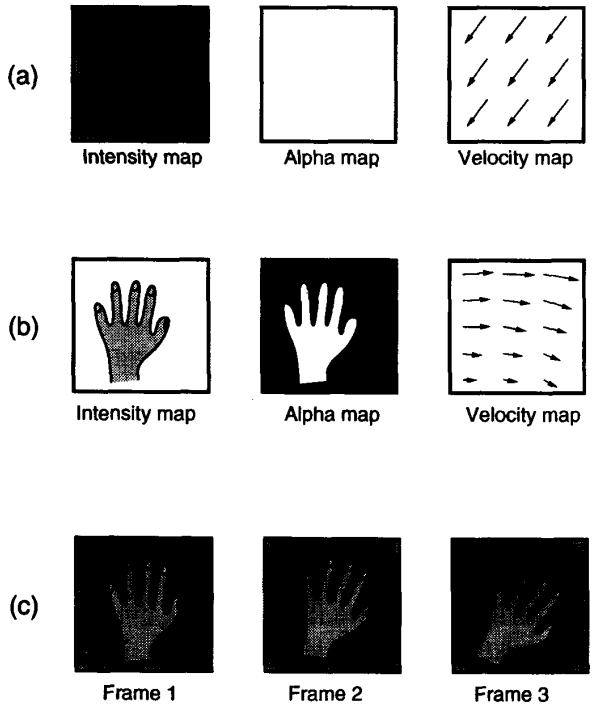


Fig. 2. The desired decomposition of the hand sequence into layers. (a) The background layer. The intensity map corresponds to the checkerboard pattern; the alpha map is unity everywhere (since the background is assumed to be opaque); the velocity map is a constant. (b) The hand layer. The alpha map is unity where the hand is present and zero where the hand is absent; the velocity map is smoothly varying. (c) The resynthesized image sequence based on the layers.

However, when one object moves in front of another, as moving objects generally do, the rubber sheet model fails. Image information appears and disappears at the occlusion boundaries and the optic flow algorithm has no way to represent this fact. Therefore such algorithms tend to give extremely bad motion estimates near boundaries.

In many image coding systems the motion model is even more primitive. A fixed array of blocks is assumed to translate rigidly from frame to frame. The model cannot account for motions other than translations, nor can it deal with occlusion boundaries.

The layered representation that we propose is also an imperfect model of the world but it is able to cope with a wider variety of phenomena than the traditional representations. The approach may be categorized with object-based methods [11], [15].

II. The Layered Representation

As discussed above, a layer contains a set of maps specifying its intensity, opacity, and motion. Other maps can be defined as well, but these ones are crucial.

In order to deal with transparency, motion blur, optical blur, shadows, etc., it is useful to allow the alpha channel to take on any value between 0 and 1. A flow chart for the compositing process is shown in Fig. 3. Each layer occludes the one beneath it, according to the equation

$$I_1(x,y) = E_0(x,y)(1-\alpha_1(x,y)) + E_1(x,y)\alpha_1(x,y) \quad (1)$$

where α_1 is the alpha channel of layer E_1 and E_0 is the background layer. Any number of stages can be cascaded, allowing for any number of layers.

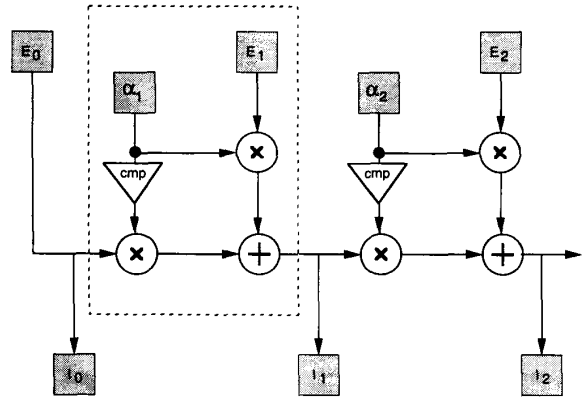


Fig. 3. A flow chart for compositing a series of layers. The box labeled "cmp" generates the complement of alpha, $(1-\alpha)$.

For dealing with image sequences, we allow a velocity map to operate on the layers over time. It is as if the cel animator were allowed to apply simple distortions to his cels. The intensity map and the alpha map are warped together so that they stay registered. Since the layered model may not adequately capture all of the image change in each layer, we also allow for a "delta map," which serves as an error signal to update the intensity map over time. The resulting system is shown in Fig. 4. Only one full stage is shown, but an unlimited series of such stages may be cascaded.

Once we are given a description in terms of layers, it is straightforward to generate the image sequence. The difficult part is determining the layered representation given the input sequence. In other words, synthesis is easy but analysis is hard. The representation is nonunique: there will be many different descriptions that lead to the same synthesized image sequence. Indeed it is always possible to represent an image sequence with a single layer, where the delta map does all the work of explaining the change over time. As our expertise in midlevel vision improves we can achieve better representations that capture more of the underlying structure of the scene. Thus, a layered representation can serve to specify the decoding process, while the encoding process remains open to further improvement by individual users.

We will describe some analysis techniques that we have applied to a set of standard video sequences. In our present implementations we have simplified the representation in several ways. The alpha channels are binary, i.e., objects are completely transparent or completely opaque. The velocity

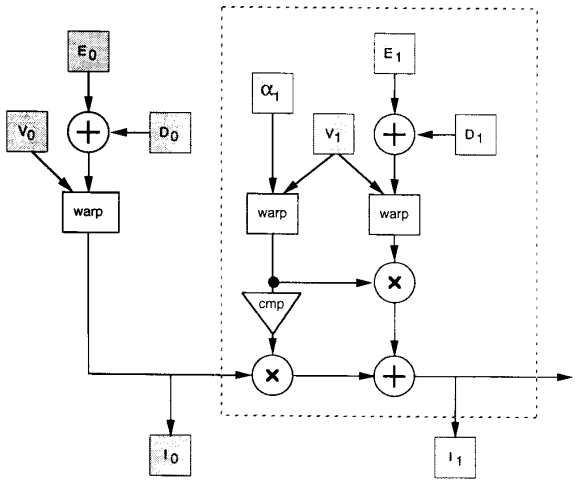


Fig. 4. A flow chart for compositing that incorporates velocity maps, V, and delta maps, D.

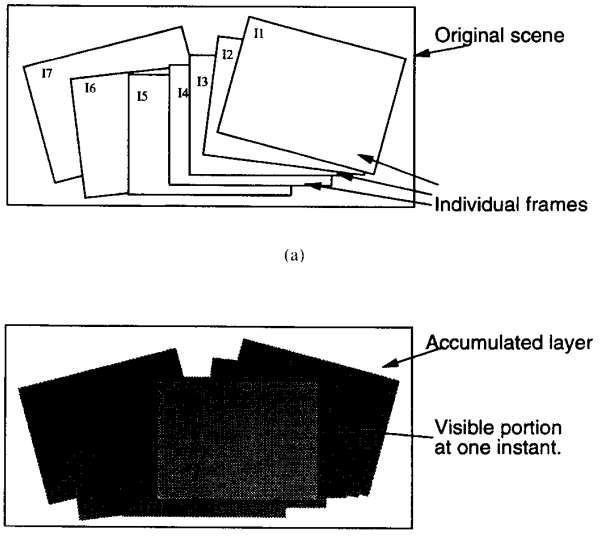


Fig. 5(a). The frames in the sequence are taken from an original scene that is larger than any individual frame. (b) The information from all the frames may be accumulated into a single large layer. Each frame is then a glimpse into this layer as viewed through a window.

maps are restricted to affine transformations, which others have found to be successful models for segmenting multiple motions [3], [10]. There are no delta maps used. In spite of these simplifications the representation is able to capture much of the desired image information. Earlier discussions of this work are contained in [21] and [22].

III. Analysis into Layers

Let us begin by considering the first layer: the background. Sometimes the background is stationary but often it is undergoing a smooth motion due, for example, to a camera pan. Background information appears and disappears at the edges of the viewing frame in the course of the pan. It is, of course, possible to send updating information on each frame, adding and subtracting the image data at the edges. However, with layers it is preferable to represent the background as it really is: an extended image that is larger than the viewing frame. Having built up this extended background, one can simply move the viewing frame over it, thereby requiring a minimal amount of data.

Fig. 5(a) shows a sequence of positions that the viewing frame might take during a pan over the background. Fig. 5(b) shows the image information that can be accumulated from that sequence. At any given moment the camera is selecting one part of the extended background [10], [19].

To build up the extended background we may consider the background to be a continuous function of unlimited spatial extent [12], [14]. Each image in a sequence captures a set of discrete samples from that function within the bounds specified by the viewing frame. The viewing parameters may change over time; for example the background image may undergo affine or higher-order distortions. If we estimate these parameters we can continue to map the image data into our extended background image.

It is also possible that foreground objects will obscure the background from time to time. If there are background regions that are always obscured we cannot know their true content. On the other hand, since these regions are never seen, we do not need to know them in order to resynthesize the sequence. Any regions that are revealed can be accumulated into the background if we correctly identify them.

To illustrate how an extended view of a layer can be obtained, consider the MPEG *Flower Garden* sequence, three frames of which are shown in Fig. 6. Because the camera is translating laterally, the flower bed image undergoes a shearing motion, with the nearer regions translating faster than the farther regions. The distortion can be approximated by an affine transformation. Fig. 7 shows the sequence after the motion of the flower bed has been cancelled with the inverse affine transformation. When the inverse warp is applied, the flower bed is stabilized. The remainder of the image appears to move; for example, the tree seems to bend.

We can generate the entire warped sequence and accumulate the pixels with stable values into a single image. This image will contain pixels that belong to the flower bed, including those that were sometimes hidden by the tree and those that were sometimes hidden by the right or left boundary of the viewing frame. Such an accumulated image is shown in Fig. 8.

In the ideal case, it will be possible to account for all the visible flower bed pixels in every image in the sequence by warping and sampling this single accumulated image. And in practice we find the process can be fairly successful in cases where the region is undergoing a simple motion.

The same process can be used for any region whose motion has been identified. Thus we may build up separate images for several regions. These become the layers that we will

(a)

(b)

(c)

Fig. 6. The MPEG *Flower Garden* sequence: (a) Frame 0; (b) frame 15; (c) frame 30.

(a)

(b)

(c)

Fig. 7(a). Frame 1 warped with an affine transformation to align the flowerbed region with that of frame 15; (b) original frame 15 used as reference; (c) frame 30 warped with an affine transformation to align the flowerbed region with that of frame 15.

composite in resynthesizing the scene. Fig. 12 shows the other layers that we extract: one for the house region; one for the tree. We will now describe how we use motion analysis to perform the layered decomposition.

IV. Motion Analysis

The *Flower Garden* sequence contains a number of regions that are each moving in a coherent fashion. We can use the motion to segment the image sequence. In traditional image processing systems, segmentation means that each pixel is assigned to a region and the image is cut apart like a jigsaw puzzle without overlap. We do perform such a segmentation initially, but our ultimate goal is a layered representation whereby the segmented regions are tracked and processed over many frames to form layers of surface intensity that have overlapping regions ordered in depth. Thus, our analysis of an image sequence into layers consists of two stages: 1) robust motion segmentation and 2) synthesis of the layered representation.

We consider that regions undergoing common affine motion are likely to arise from the same surface in the world,

Fig. 8. Accumulation of the flowerbed. Image intensities are obtained from a temporal median operation on the motion-compensated images. Only the regions belonging to the flowerbed layer are accumulated in this image. Note also that occluded regions are correctly recovered by accumulating data over many frames.

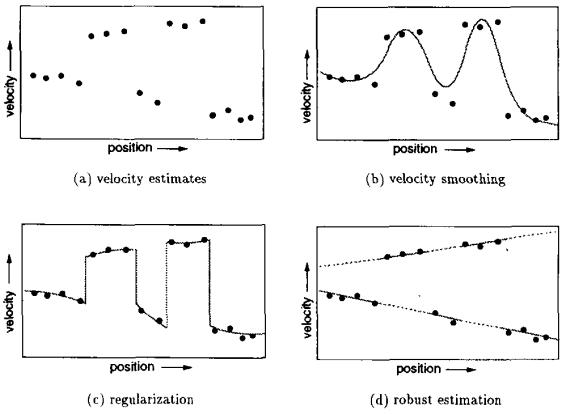


Fig. 9(a). Velocity estimates appear as samples in space; (b) standard optic flow algorithms impose some smoothing on the velocity estimates in order to deal with noise; (c) regularization algorithms model the velocity field as a set of smooth regions while allowing for sharp breaks at model boundaries; (d) shows the representation that we wish to attain. The velocity samples are explained by two affine models (straight lines) and discontinuities are explained by the occlusion of one object by the other.

and so we seek out such regions and group them together. Affine motion is defined by the equations

$$V_x(x, y) = a_{x0} + a_{xx}x + a_{xy}y \quad (2)$$

$$V_y(x, y) = a_{y0} + a_{yx}x + a_{yy}y \quad (3)$$

where V_x and V_y are the x and y components of velocity, and the $a's$ are the parameters of the transformation. If these components are plotted as a function of position, then affine motions become planes. When we use affine models for analyzing motion, we are proposing that the optic flow in an image can be described as a set of planar patches in velocity space.

The concept is easier to visualize in one dimension, as shown in Fig. 9(a). Suppose that we have a background undergoing one affine motion and a foreground object undergoing another affine motion. In the illustration the foreground object contains two parts, both of which are undergoing the same common motion. The velocity field will consist of five different patches: three corresponding to the background, and two corresponding to the foreground.

In a standard optic flow algorithm the velocity field will be smoothed. Such algorithms generally impose some smoothing on the velocity estimates in order to deal with noise, the aperture problem, etc. The result is a smooth function, as shown in Fig. 9(b). Regions along the boundaries are assigned velocities that are intermediate between the foreground and background velocities. This is plainly incorrect. It arises from the fact that the optic flow algorithm is implicitly using something like a rubber-sheet model of the world. The velocity field is forced to be single-valued and smooth, and so the mixing of the two motions is inevitable.

The analysis can be improved by allowing for sharp breaks in the velocity field, as is often done with regularization [8], [16]. The velocity field is now modeled as a set of smooth regions interrupted by discontinuities. The result looks like that shown in Fig. 9(c). The representation still has problems. It does not allow for multiple motion estimates to coexist at a point, which is needed for dealing with transparency. And it does not tell us anything about the way the regions should be grouped; it does not know, for example, that all of the background segments belong together.

Fig. 9(d) shows the representation that we wish to attain. Since we are modeling the motions with affine transformations, we are seeking to explain the data with a set of straight lines. In this case we need two straight lines, one for the background and one for the foreground. There are no motion discontinuities as such; the lines of motion continue smoothly across space. The discontinuities are to be explained by the occlusions of one object by another—that is, there are discontinuities in visibility. The motions themselves are considered to extend continuously whether they are seen or not. In this representation the background pixels all share a common motion, so they are assigned to a single segment, in spite of being spatially noncontiguous. This represents the first phase of our motion analysis.

V. Robust Motion Segmentation

Our robust segmentation consists of two stages: 1) local motion estimation; 2) segmentation by affine model fitting. Critical processing involves motion estimation and segmentation.

A number of authors have described methods for achieving multiple affine motion decompositions [3], [5], [10]. Our method is based on robust estimation and k-means clustering in affine parameter space.

In many multiple motion estimation techniques, a recursive algorithm is used is to detect multiple motion regions in the scene [3], [10]. At each iteration, these algorithms assume that a dominant motion region can be detected. Once the dominant region is identified and the motion within the region is estimated, it is eliminated and the next dominant motion is estimated from the remaining portion of the image. Such methods can have difficulties with scenes containing

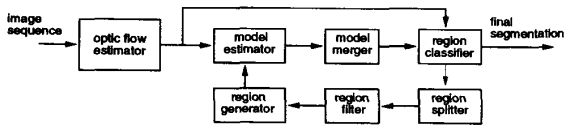


Fig. 10. The motion segmentation algorithm. Motion models are estimated within regions specified by the region generator. Similar models are merged by the merger and segmentation obtained by motion classification. Region splitter and region filter enforces local connectivity and provides robustness to the system.

several strong motions, since the estimated parameters reflect a mixture of sources.

To avoid the problems resulting from estimating a single global motion, we use a gradual migration from a local representation to a global representation. We begin with a conventional optic flow representation of motion. Because optic flow estimation is carried out over a local neighborhood, we can minimize the problems of mixing multiple motions within a given analysis region. From optic flow, we determine a set of affine motions that are likely to be observed. Segmentation is obtained by classifying regions to the motion model that provides the best description of the motion within the region.

The segmentation framework is outlined in Fig. 10. At each iteration, our multiple model framework identifies multiple coherent motion regions simultaneously. Iteratively, the motion model parameters are calculated within these coherent regions and segmentation is refined.

Several authors have presented robust techniques for multiple motion estimation. Black and Anandan [4] described a multiple motion estimation technique that applies image constraints via influence functions that regulate the effects of outliers in a simulated annealing framework. Darrell and Pentland [5] described a multimodel regularization network whereby image constraints are applied separately to each model. Depommier and Dubois [6] employed line models to detect motion discontinuities. These techniques, however, require intensive computation.

We use a simpler technique for imposing image constraints and rejecting outliers. In our algorithm, motion smoothness is imposed only within the layer and by having multiple layers we can describe the discontinuities in motion. In addition, we impose constraints on coherent region size and local connectivity by applying simple thresholds to reject outliers at different stages in the algorithm, thus providing stability and robustness.

A. Optic Flow Estimation

Our local motion estimate is obtained with a multiscale coarse-to-fine algorithm based on a gradient approach described by [2], [13], and [17]. For two consecutive frames the motion at each point in the image can be described by (4) and the linear least-squares solution for motion by (5)

$$I_t(x - V_x(x,y), y - V_y(x,y)) = I_{t+1}(x,y) \tag{4}$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum -I_x I_t \\ \sum -I_y I_t \end{bmatrix} \tag{5}$$

where I_x, I_y, and I_t are the partial derivatives of the image intensity at position (x, y) with respect to x, y, and t, respectively. The summation is taken over a small neighborhood around the point (x, y). The multi-scale implementation allows for estimation of large motions. When analyzing scenes exhibiting transparent phenomena, the motion estimation technique described by Shizawa and Mase [18] may be suitable. However, in most natural scenes, the simple optic flow model provides a good starting point for our segmentation algorithm.

B. Motion Segmentation

Given the optic flow field, the task of segmentation is to identify the coherent motion regions. When a set of motion models is known, classification based on motion can directly follow to identify the corresponding regions. However, the corresponding motion models for the optic flow field are unknown initially. One simple solution is to generate a set of models that can describe all the motions that might generally be encountered in sequences. This results in a large hypothesis set, which is undesirable because classification with a large number of models will be computationally expensive and unstable.

In our framework, we sample the motion data to derive a set of motion hypotheses that are likely to be observed in the image. Referring to Fig. 10, the region generator initially divides the image into several arbitrary regions. Within each region, the model estimator calculates the model parameters producing one motion hypothesis for each region. An ideal configuration for these regions is one that corresponds to the actual coherent regions. However, this configuration is unknown and is ultimately what we want to obtain in segmentation.

We use an array of nonoverlapping square regions to derive an initial set of motion models. The size of the initial square regions are kept to a minimum to localize the estimation and to avoid estimating motion across object boundaries. Larger regions will provide better parameter estimation under noisy situations.

The affine parameters within these regions are estimated by standard linear regression techniques. This estimation can be seen as a plane-fitting algorithm in velocity space since the affine model is a linear model of local motion. The regression is applied separately on each velocity component because the x affine parameters depend only on the x component of velocity and the y parameters depend only on the y component of velocity. If we let $\mathbf{a}_i^T = [a_{x0_i} a_{xx_i} a_{xy_i} a_{y0_i} a_{yx_i} a_{yy_i}]$ be the ith hypothesis vector in the 6-D affine parameter space with $\mathbf{a}_{x_i}^T = [a_{x0_i} a_{xx_i} a_{xy_i}]$ and $\mathbf{a}_{y_i}^T = [a_{y0_i} a_{yx_i} a_{yy_i}]$ corresponding to the x and y components , and $\phi^T = [1\ x\ y]$ be the regressor, then the motion fields equations (2) and (3) can be simply written as

$$V_x(x,y) = \phi^T \mathbf{a}_{x_i} \tag{6}$$

$$V_y(x,y) = \phi^T \mathbf{a}_{y_i} \tag{7}$$

and a linear least squares estimate of $\mathbf{a}_i$ for a given motion

field is as follows

$$[\mathbf{a}_{x_i}\mathbf{a}_{y_i}] = \left[\sum_{P_i}\phi\phi^T\right]^{-1}\sum_{P_i}(\phi[V_x(x,y)V_y(x,y)]) \quad (8)$$

where each region is indexed with the variable i, and the summation is applied within each region.

Regardless of the initial region size, many of the affine hypotheses will be incorrect because the initial square regions may contain object boundaries. The reliability of a hypothesis is indicated by its residual error, σ_i^2, which can be calculated as follows

$$\sigma_i^2 = \frac{1}{N_i}\sum_{x,y}(\mathbf{V}(x,y) - \mathbf{V}_{\mathbf{a}_i}(x,y))^2 \quad (9)$$

where N is the number of pixels in the analysis region. Hypotheses with a residual greater than a prescribed threshold are eliminated because they do not provide a good description of motion within the analysis region.

Motion models from regions that cover the same object will have similar parameters. These are grouped in the affine motion parameter space with a k-means clustering algorithm [20], which is modified to allow k to be determined adaptively. In the clustering process, we derive a representative model for each group of similar models. This model clustering produces a set of likely affine motion models that are exhibited by objects in the scene. In addition, it provides more stability in the segmentation by preventing multiple models from representing a single coherent motion region with a single model instead of with various similar models.

A scaled distance, $D_m(\mathbf{a}_1,\mathbf{a}_2)$, is used in the parameter clustering process in order to scale the distance of the different components in the parameter space. Scale factors are chosen so that a unit distance along any component in the parameter space corresponds to roughly a unit displacement at the boundaries of the image.

$$D_m(\mathbf{a}_1,\mathbf{a}_2) = [(\mathbf{a}_1 - \mathbf{a}_2)^T M(\mathbf{a}_1 - \mathbf{a}_2)]^{\frac{1}{2}} \quad (10)$$

$$M = \mathrm{diag}(1\ r^2\ r^2\ 1\ r^2\ r^2) \quad (11)$$

where r is the roughly the dimensions of the image.

In our adaptive k-means algorithm, a set of cluster centers that are separated by a prescribed distance are initially selected. Each of the affine hypotheses is then assigned to the nearest center. Following the assignment, the centers are updated with mean position of the cluster. The centers are iteratively modified until cluster membership, or equivalently, the cluster centers, are unchanged. During these iterations, some centers may approach other centers. When the distance between any two centers is less than a prescribed distance, the two clusters are merged into a single cluster, reducing the two centers into a single center. In this way the adaptive k-means clustering strives to describe the data with a small number of centers while minimizing distortion. By selecting the clusters with the largest membership, we can expect to represent a large portion of the image with only a few motion models.

C. Region Assignment by Hypothesis Testing

We use the derived motion models in a hypothesis testing framework to identify the coherent regions. In the hypothesis testing, we assign the regions within the image in a way that minimizes the motion distortion. We use the distortion function, $G(i(x,y))$, described by

$$G(i(x,y)) = \sum_{x,y}(\mathbf{V}(x,y) - \mathbf{V}_{\mathbf{a}_i}(x,y))^2 \quad (12)$$

where $i(x,y)$ indicates the model that location (x,y) is assigned to, $\mathbf{V}(x,y)$ is the estimated local motion field, and $\mathbf{V}_{\mathbf{a}_i}(x,y)$ is the affine motion field corresponding to the ith affine motion hypothesis. From (12) we see that $G(i(x,y))$ reaches a minimum when the affine models exactly describe the motion within the regions. However, this is often not the case when dealing with real sequences, so instead we settle for an assignment that minimizes the total distortion.

Since each pixel location is assigned to only one hypothesis, we obtain the minimum total distortion by minimizing the distortion at each location. This is achieved when each pixel location is assigned to the model that best describes the motion at that location. Coherent motion regions are identified by processing each location in this manner. We summarize the assignment in the following equation

$$i_0(x,y) = \arg\ \min[\mathbf{V}(x,y) - \mathbf{V}_{\mathbf{a}_i}(x,y)]^2 \quad (13)$$

where $i_0(x,y)$ is the minimum distortion assignment.

However, not all the pixels receive an assignment because some of motion vectors produced by optic flow estimation may not correctly describe image motion. In regions where the assumptions used by the optic flow estimation are violated, usually at object boundaries, the motion estimates are typically difficult to describe using affine motion models. Any region where the error between the expected and observed motion is greater than a prescribed threshold remains unassigned, thus preventing inaccurate data from corrupting the analysis. We find by experiment that a value of 1 pixel motion provides a reasonable threshold.

We now define the binary support maps that describe the regions for each of the affine hypotheses as

$$P_i(x,y) = \begin{cases} 1 & \text{if } i_0(x,y) = i \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

Thus, for each model, its corresponding region support map takes on the value of 1 in regions where it best describes the motion, and a value of 0 elsewhere. These maps allow us to identify the model support regions and to refine our affine motion estimates in the subsequent iterations.

D. Iterative Algorithm

In the initial segmentation step, we use an array of square regions. These will not produce the best set of affine models as discussed above. However, after the first iteration, the algorithm produces a set of affine models and the corresponding regions undergoing affine motion. By estimating the affine parameters within the estimated regions, we obtain parameters that more accurately describe the motion of the region. At each

iteration, the segmentation becomes more accurate because the parameter estimation of an affine motion model is performed within a single coherent motion region.

Additional stability and robustness in the segmentation is obtained by applying additional constraints on the coherent regions. These constraints are applied to the segmentation before the models are reestimated. Because the affine motion classification is performed at every point in the image with models that span globally, a model may coincidentally support points in regions undergoing a different coherent motion or it may support points that have inaccurate motion estimates. Typically, this results in a model supporting sporadic points that are disjoint from each other.

The region splitter and filter as shown in Fig. 10 identify these points and eliminate them. When two disjoint regions are supported by a single model, the region splitter separates them into two independent regions so that a model can be produced for each region. Following the splitter is the region filter, which eliminates small regions because model estimation in small regions will be unstable. Subsequently, the model estimator produces a motion model for each reliably connected region. Thus the region splitter in conjunction with the region filter enforces the local spatial connectivity of motion regions.

Sometimes the splitting process produces two regions even though they can be described by a single model. This, however, is not a problem because the parameters estimated for each of these regions will be similar. The models corresponding to these regions will be clustered into a single model in the model merging step.

Convergence is obtained when only a few points are reassigned or when the number of iterations reaches the maximum allowed. This is typically less than 20 iterations. Regions that are unassigned at the end of the motion segmentation algorithm are reassigned in a refinement step of the segmentation. In this step, we assign these regions by warping the images according to the affine motion models and selecting the model that minimizes the intensity error between the pair of images.

The analysis maintains temporal coherence and stability of segmentation by using the current motion segmentation results to initialize the segmentation for the next pair of frames. The affine model parameters and the segmentation of a successive frame will be similar because an object's shape and motion change slowly from frame to frame. In addition to providing temporal stability, analysis that is initialized with models from previous segmentation results in fewer iterations for convergence. When the motion segmentation on the entire sequence is completed, each affine motion region will be identified and tracked in the sequence with corresponding support maps and affine motion parameters.

Typically, processing on the subsequent frames requires only two iterations for stability, and the parameter clustering step becomes trivial. Thus, most of the computational complexity is in the initial segmentation, which is required only once per sequence.

VI. Layer Synthesis

The robust segmentation technique described above provides us with a set of nonoverlapping regions, which fit together like pieces of a jigsaw puzzle. Regions that are spatially separate may be assigned a common label since they belong to the same motion model. However, robust segmentation does not generate a layered representation in and of itself. The output of a segmenter does not provide any information about depth and occlusion; the segments all lie in a single plane.

In order to generate a true layered representation we must take a second step. The information from a longer sequence must be combined over time, so that the stable information within a layer can be accumulated. Moreover, the depth ordering and occlusion relationships between layers must be established. This combined approach—robust segmentation followed by layer accumulation—is central to the present work. Previous robust segmenters have taken the first step but not the second. (For example, [5] use a "multilayer" neural network in the robust segmenter, but the neural layers contain no information about depth ordering or occlusion, nor do they contain overlapping regions.)

In deriving the intensity and alpha maps for each layer, we observe that the images of the corresponding regions in the different frames differ only by an affine transformation. By applying these transformations to each of the frames in the original sequence, corresponding regions in different frames can be motion-compensated with an inverse warp. We use bicubic interpolation in the motion compensation.

Thus, when the motion parameters are accurately estimated for each layer, the corresponding regions will appear stationary in the motion-compensated sequence. The layer intensity maps and alpha maps are derived from these motion-compensated sequences.

Some of the images in the compensated sequence, however, may not contain a complete image of the object because of occlusions. Additionally, an image may have small intensity variations due to different lighting conditions. In order to recover the complete representative image and boundary of the object, we collect the data available at each point in the layer and apply a median operation on the data. This operation can be easily seen as a temporal median filtering operation on the motion compensated sequence in regions defined by the support maps. Earlier studies have shown that motion compensation median filter can enhance noisy images and preserve edge information better than a temporal averaging filter [8].

$$\text{Layer}_i(x,y) = \text{median}\{M_{i,1}(x,y), M_{i,2}(x,y), \ldots, M_{i,n}(x,y)\} \quad (15)$$

where Layer_i is the ith layer, and $M_{i,k}(x,y)$ is the motion-compensated image obtained by warping frame k of the original sequence by the estimated affine transform for layer i. Note that the median operation ignores regions that are not supported by the layer. These regions are indicated by a 0 in the support maps described in (14).

Finally, we determine occlusion relationships. For each layer, we generate a map corresponding to the number of points available for constructing the layer intensity map. A point in the intensity map generated from more data

is visible in more frames and its derived intensity in the layer representation is more reliable. These maps and the corresponding layer intensity maps are warped to their respective positions in the original sequence according to the inverse affine transformation. By verification of intensities and of the reliability maps, the layers are assigned a depth ordering. A layer that is derived from more points occludes an image that is derived from fewer points, since an occluded region necessarily has fewer corresponding data points in the motion compensation stage. Thus, the statistics from the motion segmentation and temporal median filtering provide the necessary description of the object motion, texture pattern, opacity, and occlusion relationship.

VII. Experimental Results

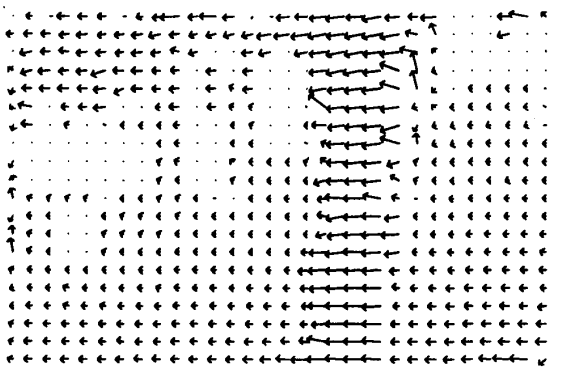
We have implemented the image analysis technique in C on a Hewlett Packard 9000 series 700 workstation. We illustrate the analysis, the representation, and the synthesis with the first 30 frames of the MPEG *Flower Garden* sequence, of which frames 1, 15, and 30 are shown in Fig. 6. Dimensions of the image are 720×480. In this sequence, the tree, flower bed, and row of houses move towards the left but at different velocities. Regions of the flower bed closer to the camera move faster than the regions near the row of houses, which are in the distance.

Optic flow obtained with a multiscale coarse-to-fine gradient method on a pair of frames is shown in Fig. 11(a). Notice the incorrect motion estimates along the occlusion boundaries of the tree as shown by the different lengths of the arrows and the arrows that point upwards.

The segmentation map for the first frame is shown in Fig. 11(b). Each affine motion region is depicted by a different gray level. The darkest regions along the edges of the tree correspond to regions where the motion could not be easily described by affine models. Region assignment based on warping the images and minimizing intensity error reassigns these regions, as shown in Fig. 11(c).

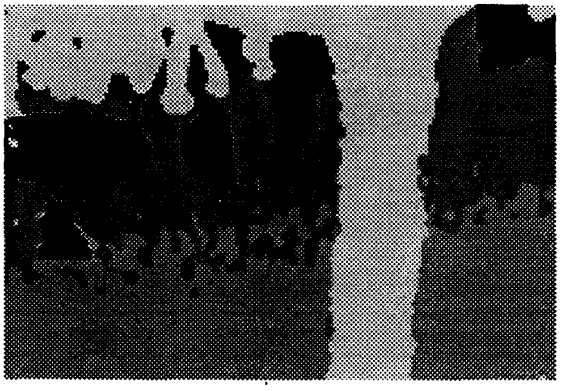
We used an initial segmentation map consisting of 75 nonoverlapping square regions to derive a set of affine motion models. However, we found that the exact number of initial regions is not critical in the segmentation. The ten models with the lowest residual errors were selected from the k-means affine parameter clustering for model testing in the assignment stage. After about five iterations, segmentation remained stable with four affine models. We find that the number of models initially selected can vary from five to 15 with similar results. This initial segmentation required about two minutes of computation time. Total computation time for motion analysis on the 30 frames including the median operation is about 40 min.

We used frame 15 as the reference frame for the image alignment. With the motion segmentation, we are able to remove the tree from the flower bed and house layers and recover the occluded regions. The sky layer is not shown. Regions with no texture, such as the sky, cannot be readily assigned to a layer since they contain no motion information. We assign these regions to a single layer that describes stationary textureless objects. Note that the overlap regions in house and flower bed layer are undergoing similar motion, and therefore, these regions can be supported by either layer without introducing annoying artifacts in the synthesis.



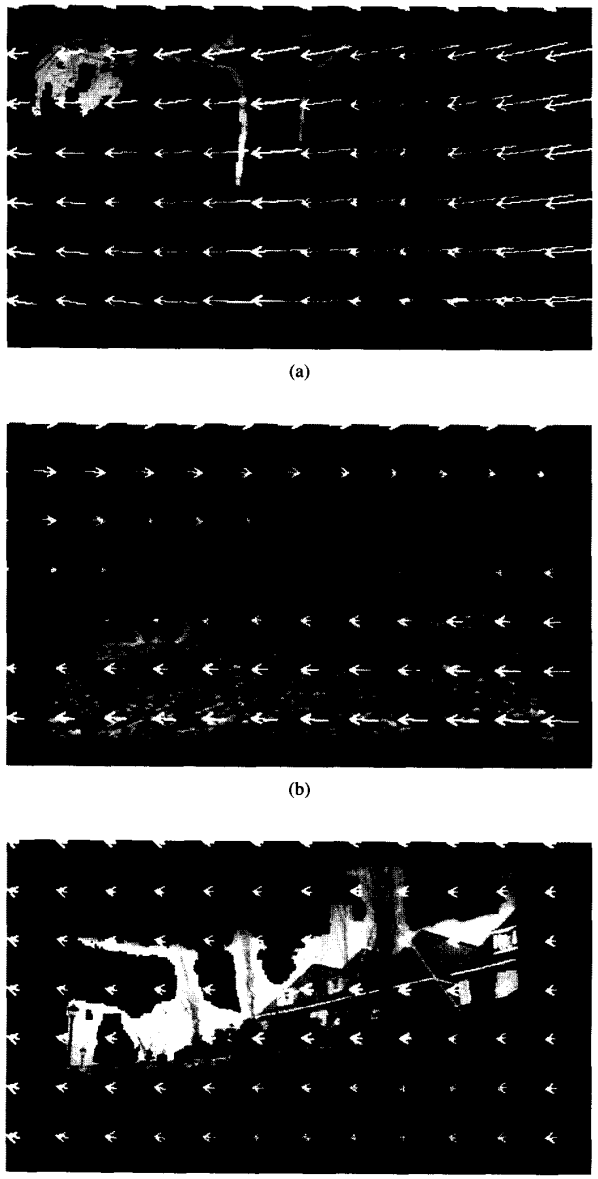
(a)

(b)



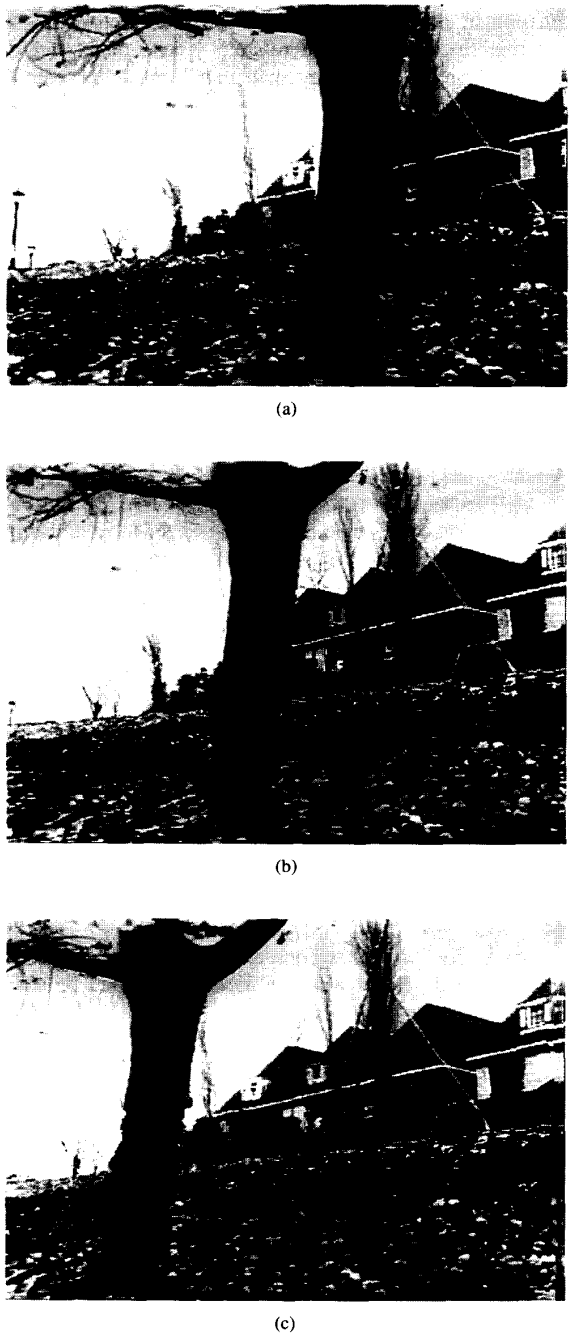
(c)

Fig. 11(a). The optic flow from multi-scale gradient method; (b) segmentation obtained by clustering optic flow into affine motion regions; (c) segmentation from consistency checking by image warping, representing moving images with layers.



(a) (b) (c)

Fig. 12. The layers corresponding to (a) the tree, (b) the flower bed, and (c) the house. The affine flow field for each layer is superimposed.

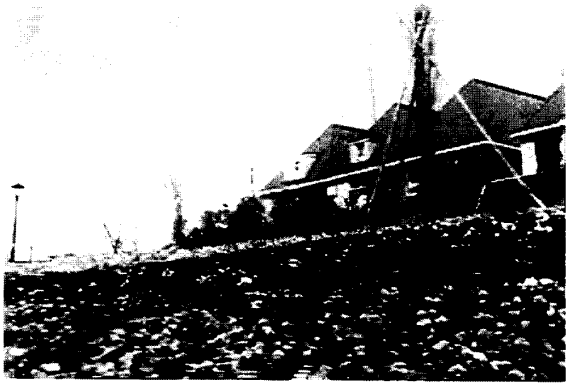


(a) (b) (c)

Fig. 13(a). Frame 0; (b) frame 15; (c) frame 30, as reconstructed from the layered representation.

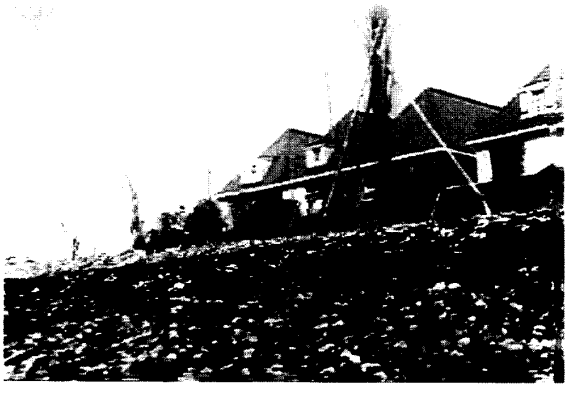
We can recreate an approximation of the entire image sequence from the layer maps of Fig. 12, along with the occlusion information, the affine parameters that describe the object motion, and the stationary layer. Fig. 13 shows three synthesized images corresponding to the three images in Fig. 6. The objects are placed in their respective positions and the occlusion of background by the tree is correctly described by the layers.

Figs. 15–17 show results of layer analysis on the MPEG *Calendar* sequence. In this sequence, a toy train pushes a ball along the track, while the camera pans right and zooms out. A small rotating toy in the lower left corner of the image did not produce a layer because its motion could not be estimated. In addition, the calendar, which is moving downward with respect to the wallpaper, is combined into a single layer with the wallpaper. These regions are represented with a single layer because motion in these regions as estimated from two consecutive frames can be described with a single affine motion within specified motion tolerances.



(a)

(b)

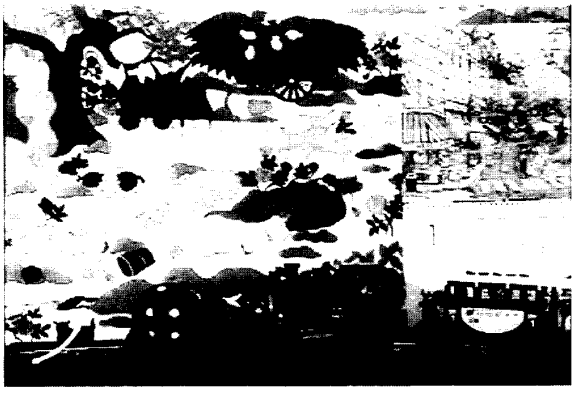


(c)

Fig. 14(a)-(c). The sequence reconstructed without the tree layer.

(a)

(b)



(c)

Fig. 15(a). Frame 0; (b) frame 15; and (c) frame 30, of the MPEG *Calendar* sequence.

To handle this problem, we need to consider motion analysis and segmentation over different temporal scales.

Currently, our motion analysis technique performs well when motion regions in the image can be easily described by the affine motion model. Because our analysis is based on motion, regions must be sufficiently textured and large in size for stability in the segmentation and layer extraction. Therefore, scenes with few foreground objects undergoing affine motion are suitable for our analysis. Sequences with complicated motions that are not easily described by the layered model require special treatment.

VIII. Compression

The layered decomposition can be used for image compression. In the case of the *Flower Garden* sequence, we are able to

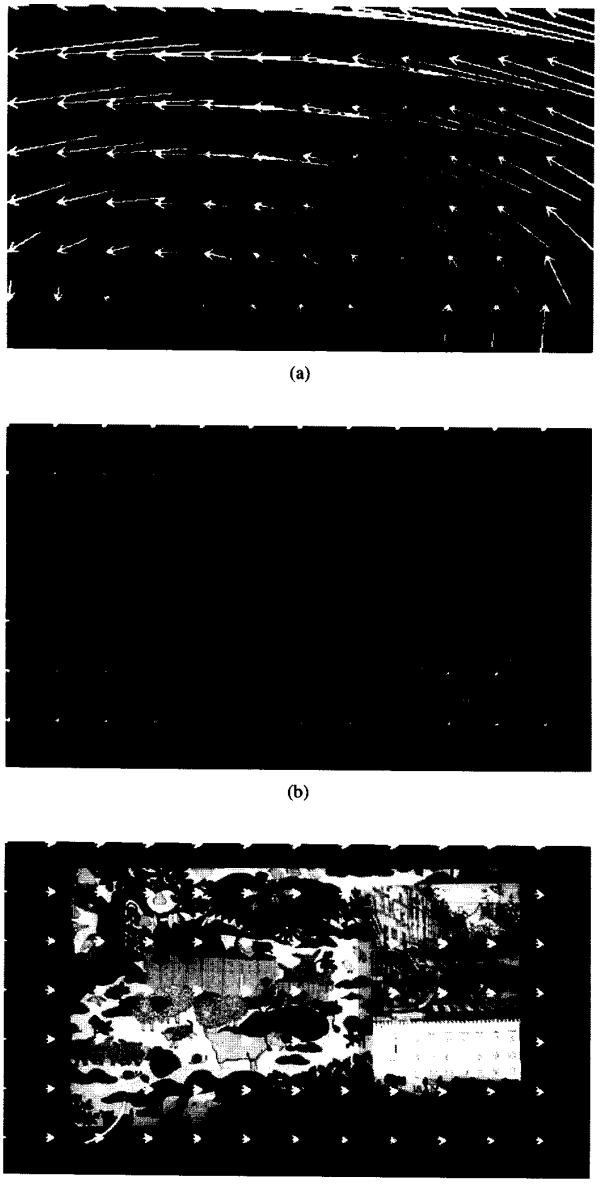


Fig. 16. The layers corresponding to (a) the ball, (b) the train, and (c) the background.

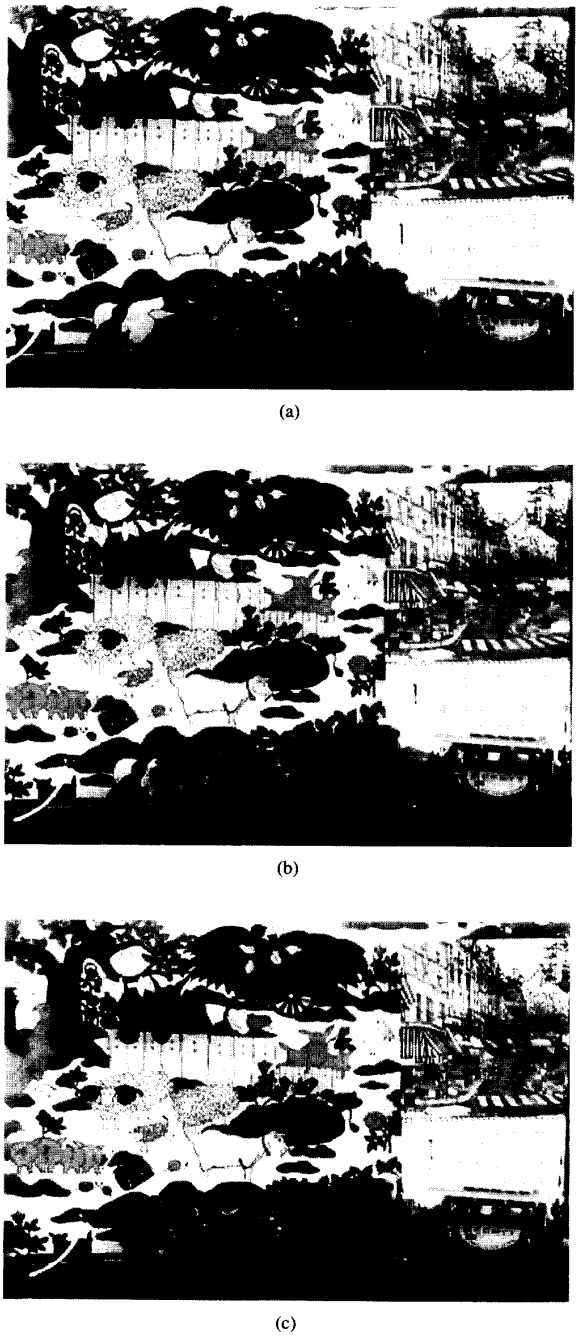


Fig. 17. Synthesis frames (a) 1, (b) 15, and (c) 30 of the MPEG *Calendar* sequence. These images are generated from the layer images in Fig. 16.

synthesize a full 30 frame sequence starting with only a single still image of each of the layers. The layers can be compressed using conventional still image compression schemes. While we have not yet done extensive studies of compression we can describe some preliminary results [7], [23].

The data that must be sent to reconstruct the *Flower Garden* sequence includes the intensity maps, the alpha maps, and the motion parameters for each layer. To compress the intensity map we used a JPEG coder. Some blocks in the map are empty, being outside the valid region of support for the layer; these are coded with a symbol indicating an empty block. Other blocks are partially empty, and these were filled in smoothly so that JPEG did not devote bits to coding the sharp (spurious) edge. To compress the alpha map we used a chain code, which is possible because we are representing alpha as a binary image. The motion parameters can be sent without compression because they only represent six numbers per layer per frame.

We are able to code a 30 frame (1 s) color video sequence of the *Flower Garden* using 300 kbits for a resolution of 360

$\times$ 240, and 700 kbits for a resolution of 720 $\times$ 480. Thus, the data rates are 300 and 700 kb/s, respectively. The affine motion parameters for the sequence required 40 kb. For the full resolution sequence, the alpha maps were encoded losslessly with chain codes at 60 kb and the color intensity maps encoded with JPEG at 600 kb.

The only artifacts added by the data compression are those associated with the JPEG coder, and these are fixed artifacts that are rigidly attached to the moving layers. Thus they do not "twinkle" or "shimmer" the way temporal artifacts often do with traditional coders. At the data rates noted above, the JPEG artifacts are only slightly visible with an SNR of 30 dB.

The layered coding process itself can add artifacts regardless of data compression. For example, since the layered segmentation is imperfect there is a bit of the sky attached to the edge of the tree, and this becomes visible when the tree moves over the houses. Also, the affine approximation of the motion is imperfect so that the synthesized motions of the layers do not perfectly match the actual motions in the original scene. And the temporal accumulation process can introduce blurring into regions where the affine warp is incorrect. We believe that all of these artifacts can be reduced by improvements in the analysis process. In any case, the reconstructed sequence will not generally be a perfect replica of the original sequence even if the layers are transmitted without data compression. Error images could be sent in order to correct for the imperfections of the reconstructed images at a cost of additional data.

It is interesting to note that some of the artifacts associated with the layered coding can be severe in the MSE sense and yet be invisible to the human viewer. For example, the naive observer does not know that the flower bed's motion is not truly affine. The reconstructed motion is slightly wrong—and in the squared error sense it is quite wrong—but it looks perfectly acceptable.

IX. Other Applications

Because the layered representation breaks the sequence into parts that are meaningfully related to the scene, it becomes possible to achieve some interesting special effects. Fig. 14 shows the *Flower Garden* sequence synthesized without the tree layer. This shows what the scene would have looked like had the tree not been present; it is a synthetic sequence that has never actually existed. Occluded regions are correctly recovered because our representation maintains a description of motion in these regions. Note that an ordinary background memory could not achieve the effect because the various regions of the scene are undergoing different motions.

The layered representation also provides frame-rate independence. Once a sequence has been represented as layers it is straightforward to synthesize the images corresponding to any instant in time. Slow-motion and frame-rate conversion can be conveniently done by using the layered format.

X. Conclusion

We employ a layered image representation that provides a useful description of scenes containing moving objects. The image sequence is decomposed into a set of layers, each layer describing a region's motion, intensity, shape, and opacity. Occlusion boundaries are represented as discontinuities in a layer's alpha map (opacity), and there is no need to represent explicit discontinuities in velocity.

To achieve the layered description, we use a robust motion segmentation algorithm that produces stable image segmentation and accurate affine motion estimation over time. We deal with the many problems in motion segmentation by appropriately applying the image constraints at each step of our algorithm. We initially estimate the local motion within the image, then iteratively refine the estimates of an object's shape and motion. A set of likely affine motion models exhibited by objects in the scene are calculated from the local motion data and used in a hypothesis testing framework. Layers are accumulated over a sequence of frames.

The layered representation can be used for image coding. One can represent a 30–frame sequence of the MPEG *Flower Garden* sequence using four layers, along with affine motion parameters for each layer; each layer is represented by a single still image. The layers can be coded using traditional still frame coding techniques.

The synthesized sequence provides an approximate reconstruction of the original sequence. One can achieve substantial data compression in sequences that lend themselves to layered coding. The method is more successful for sequences that are easily represented by the layered model, i.e., sequences containing a few regions undergoing simple motions. We are currently investigating extensions to more complex sequences.

The layered decomposition also provides useful tools in image analysis, frame-rate conversion, and special effects.

References

[1] E. H. Adelson, "Layered representation for image coding, *Tech. Rep. 181*, MIT Media Lab, 1991.

[2] J. Bergen *et al.*, "Hierarchial model-based motion estimation.," in *Proc. Second Europ. Conf. Comput. Vision*, 1992, pp. 237–252.

[3] J. Bergen, P. Burt, R. Hingorini, and S. Peleg, "Computing two motions from three frames," in *Proc. Third Int. Conf. Comput. Vision* (Osaka, Japan), Dec. 1990, pp. 27–32.

[4] M. J. Black and P. Anandan, "Robust dynamic motion estimation over time," in *Proc. IEEE Conf. Comput. Vision Patt. Recog.* (Maui, HI), June 1991, pp. 296–302.

[5] T. Darrell and A. Pentland, "Robust estimation of a multi-layered motion representation," in *Proc. IEEE Workshop Visual Motion* (Princeton, NJ), Oct. 1991, pp. 173–178.

[6] R. Depommier and E. Dubois, "Motion estimation with detection of occlusion areas," in *IEEE ICASSP* (San Francisco), Apr. 1992, pp. 269–273, vol. 3.

[7] U. Desai, "Coding of segmented image sequences," M.S.E.E. dissertation, Mass. Inst. Tech., Cambridge, June 1994.

[8] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.

[9] T. S. Huang and Y. P. Hsu, "Image sequence enhancement," in *Image Sequence Analysis* (T. S. Huang, Ed.). New York: Springer-Verlag, 1981, pp. 289–309.

[10] M. Irani and S. Peleg, "Image sequence enhancement using multiple motions analysis," in *Proc. IEEE Conf. Comput. Vision Patt. Recog.* (Champaign, IL), June 1992, pp. 216–221.

[11] R. Lenz and A. Gerhard, "Image sequence coding using scene analysis and spatio-temporal interpolation," in *Image Sequence Processing and Dynamic Scene Analysis* (T. S. Huang, Ed.). Berlin: Springer-Verlag, 1983, NATO ASI Series, vol. F2.

[12] A. Lippman and R. Kermode, "Generalized predictive coding of movies," in *Picture Encoding Symp.* (Lausanne, Switzerland), 1993,

pp. 17–19.
[13] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Image Understanding Workshop*, 1981, pp. 121–130.
[14] P. C. McLean, "Structured video coding," for Master of Science in media arts and sciences, Media Lab, Mass. Inst. of Technol., Cambridge, June 1991.
[15] H. G. Mussman, M. Hotter, and J. Ostermann, "Object-oriented analysis—synthesis coding of moving images," *Signal Processing: Image Commun.*, vol. 1, pp. 117–138, 1989.
[16] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, 1985.
[17] L. H. Quam, "Hierarchial warp stereo," in *Proc. DARPA Image Understanding Workshop* (New Orleans, LA), 1984, Berlin: Springer-Verlag.
[18] M. Shizawa and K. Mase, "A unified computational theory for motion transparency and motion boundaries based on eigenenergy analysis," in *Proc. IEEE Conf. Comput. Vision Patt. Recog.* (Maui, HI), June 1991, pp. 289–295.
[19] L. Teodosio and W. Bender, "Salient video stills," in *Proc. ACM Multimedia Conf.* (Anaheim, CA), Aug. 1993.
[20] C. W. Therrien, *Decision Estimation and Classification*. New York: Wiley, 1989.
[21] J. Y. A. Wang and E. H. Adelson, "Layered representation for image sequence coding," in *IEEE ICASSP* (Minneapolis, MN), Apr. 1993, pp. 221–224, vol. 5.
[22] J. Y. A. Wang and E. H. Adelson, "Layered representation for motion analysis," in *Proc. IEEE Conf. Comput. Vision Patt. Recog.* (New York), June 1993, pp. 361–366.
[23] J. Y. A. Wang, E. H. Adelson, and U. Desai, "Applying mid-level vision techniques for video data compression and manipulation," to appear in *Proc. SPIE Digital Video Compression Personal Comput.: Algorithms Technol.* (San Jose, CA), 1994, vol. 2187.

John Y. A. Wang received the S.B. and M.S. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1987.

His research experience includes work in circuit design at MIT, Texas Instruments, and Hughes Aircraft. Currently, he is a research assistant at the MIT Media Laboratory Vision and Modeling Group, working towards the Ph.D. degree in electrical engineering. His research interests are in computer vision, image understanding, and motion analysis. His recent research has been centered on developing image representations and algorithms for image coding.

Mr. Wang is a member of Sigma Xi, Tau Beta Pi, and Eta Kappa Nu.

Edward H. Adelson (M'84) received the B.A. degree in physics and philosophy from Yale University and the Ph.D. in experimental psychology from the University of Michigan.

He is an Associate Professor of Vision Science at the Media Laboratory and the Department of Brain and Cognitive Science, Massachusetts Institute of Technology, Cambridge. He has published numerous papers in the fields of human visual perception, visual neurophysiology, computational vision, image processing, and image coding. His recent research includes problems in mid-level vision including motion perception and the analysis of reflectance, shading, and transparency.

Dr. Adelson received the Adolph Lomb medal in 1984 and the Rank Prize in 1992.