M.I.T. Media Laboratory Vision and Modeling Group, Technical Report No. 228 (revised), November 1993.

Representing Moving Images with Layers

Edward H. Adelson
The MIT Media Laboratory
Dept. Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139

John Y. A. Wang
The MIT Media Laboratory
Dept. Elec. Eng. and Comp. Sci.
Massachusetts Institute of Technology
Cambridge, MA 02139

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.

1 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.

Figure 1 illustrates the concept with an image sequence of a waving hand (figure 1[a]) moving against a moving background (figure 1[b]). Suppose that both the hand and the background execute simple motions as shown. The resultant image sequence is shown in figures 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 consisted of stable objects undergoing smooth motions, our decomposition should also contain stable objects undergoing smooth motions.

In the representation that we will describe below, 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.

Figure 2 shows the layered decomposition of the hand sequence. The hand and background layers are shown in figure 2[a] and [b], respectively. The resynthesized sequence is shown in figure 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. But 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 are 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 [10, 12]. An earlier description of layered formats is found in [1].

2 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 figure 3. Each layer occludes the one beneath it, according to the equation:

$$I_1(x,y) = E_0(x,y)(1-\alpha_1(x,y)) + E_1(x,y)\alpha_1(x,y). \tag{1}$$

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.

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 figure 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 non-unique: 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 mid-level 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 maps are restricted to affine transformations. 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 [17, 18].

3 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 screen 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.

Figure 5[a] shows a sequence of positions that the viewing frame might take during a pan over the background. Figure 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 [9].

To build up the extended background we may consider the background to be a continuous function of unlimited spatial extent. 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 figure 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. Figure 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 figure 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, at least with the test sequences we have used.

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 composite in resynthesizing the scene. Figure 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.

4 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 with overlapping regions.

We consider that regions undergoing common affine motion are likely to arise from the same surface in the world, and so we seek out such regions and group them together. Affine motion is defined by the equations:

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

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

where Vx and Vy 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.

The concept is easier to visualize in one dimension, as shown in figure 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 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 problems, etc. The result is a smooth function as shown in figure 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 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 [7, 13]. The velocity field is now modeled as a set of smooth regions interrupted by discontinuities. The result looks like that shown in figure 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.

Figure 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 non-contiguous. This represents the first phase of our motion analysis.

A number of authors have described methods for achieving piecewise affine decompositions [3, 5, 9]. We have developed a method based on robust estimation and K-means clustering.

5 Motion estimation

Our analysis of an image sequence into layers consists of three stages: 1) local motion estimation; 2) motion-based segmentation and; 3) object image recovery. Because the task of layer extraction consists of decomposing the sequence into a set of moving layers, the critical processing consists of motion estimation and segmentation.

In many multiple motion estimation techniques, an iterative algorithm is used is to detect multiple motion regions in the scene. 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. Estimation of these dominant regions involves global estimation using a single motion model, and thus, often results in accumulating data from multiple objects.

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 models that describe the motion fields within subregions of the image.

Finally, the task of segmentation consists of assigning regions to a given set of models. Unlike the recursive algorithms where only one region is estimated at each step, our framework allows for multiple region segmentation at each step. By allowing for multiple models, the segmentation algorithm is not limited to scenes where a single dominant motion region is present.

Our implementation of the multiple motion estimation algorithm is related to the robust techniques presented by [4, 5, 6], and is outlined in figure 10. Our segmentation algorithm is divided into two primary steps: 1) local motion estimation and, 2) affine motion segmentation.

5.1 Optic flow estimation

Our local motion estimate is obtained with a multi-scale coarse-to-fine algorithm based on a gradient approach described by [2, 11, 14]. For two consecutive frames the motion at each point in the image can be described by equation 4 and the linear least-squares solution for motion by equation 5:

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

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

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 [15] may be suitable. However, in most natural scenes, the simple optic flow model provides a good starting point for our segmentation algorithm.

5.2 Motion segmentation

Given the optic flow field, we estimate the affine parameters within a set of subregions in an image by standard linear regression techniques. This estimation can be seen as a plane-fitting algorithm in the 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 $\mathbf{a}_i^T = [a_{x0i} \;\; a_{xxi} \;\; a_{xyi} \quad a_{y0i} \;\; a_{yxi} \;\; a_{yyi}]$ be the i^{th} hypothesis vector in the 6 dimensional affine parameter space with $\mathbf{a}_{x_i}{}^T = [a_{x0i} \;\; a_{xxi} \;\; a_{xyi}]$ 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 6 and 7 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:

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

The summation is taken over a spatial patch corresponding to the i^{th} subregion in the image.

The segmentation is initialized by dividing the image into an array of square regions. By estimating motion within each of the subregions, we obtain a set of affine motions that are likely to be exhibited by moving objects in the image. The size of the initial square regions should be kept to a minimum to avoid estimating motion across object boundaries while maintaining enough data for a stable analysis.

Regardless of the initial subregion size, many of the affine hypotheses will be incorrect because the initial square subregions may contain object boundaries. These hypotheses are identified by their large residual error, σ_i^2, which can be calculated as follows:

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

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 estimates from patches that cover the same object will have similar parameters. These are grouped in the affine motion parameter space with a k-means clustering algorithm described in [16]. In the clustering process, we derive a representative model for each group of similar models. The model clustering produces a set of likely affine motion models that are exhibited by objects in the scene.

A scaled distance, $D_m(\mathbf{a}_1, \mathbf{a}_2)$, is used in the parameter clustering in order to scale the distance of the different components in the parameter space. The scale factor is 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.

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

$$M = diag(1 \quad \text{dim} \quad \text{dim} \quad 1 \quad \text{dim} \quad \text{dim}) \tag{11}$$

where dim is the dimensions of the image.

In the 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. Iteratively, the centers are 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. Thus the k-means clustering strives at describing 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.

5.3 Region assignment by hypothesis testing

Next, we use hypothesis testing with the derived motion models to reassign the regions. In the hypothesis testing, we want to assign the regions within the image in a way that minimizes the cost function, G(i(x,y)), described by:

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

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

Since each location is assigned to only one hypothesis, we obtain the minimum total cost by minimizing the cost at each location. Thus, each location is assigned to the model that best describes the motion at that location. We summarize the assignment in the following equation:

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

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

However, not all the pixels receive an assignment because the 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 described using affine motion models. These regions are unassigned when the error between the expected and observed motion is greater than a prescribed threshold. We find by experiment that a value of 1 pixel motion provides reasonable threshold. These unassigned regions are assigned in a refinement stage of the segmentation. We assign these regions by warping the images according to the affine motion models and selecting the model that minimizes the error in intensity between the pair of images.

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

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

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.

5.4 Iterative algorithm

In the initial segmentation step, we use an array of square subregions. These will not provide 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 subregions undergoing affine motion. By estimating the affine parameters within the new subregions, we obtain parameters that more accurately describe the motion of the region. At each iteration, the segmentation becomes more accurate because the affine motion estimation is performed within single motion regions. 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. Models that have small support regions are eliminated because their affine parameters will be more sensitive to noise.

The analysis maintains temporal coherence and stability of the segmentation by using the current motion segmentation results as the initial segmentation map for the next pair of frames. Since an object's shape and motion change slowly from frame to frame, the segmentation of consecutive frames will be similar resulting in fewer iterations for convergence. When the motion segmentation on the entire sequence is completed, each affine motion region will be identified in the sequence with its support maps and affine motion parameters.

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

6 Layer analysis

The images of the corresponding regions in the different frames differ only by an affine transformation. By applying these transformations to all the frames in the original sequence, corresponding regions in different frames can be motion compensated with an inverse warp. 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 represensative 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].

$$\mathbf{Layer}_i(x,y) = \\ \mathbf{median}\{M_{i,1}(x,y), M_{i,2}(x,y), ..., M_{i,n}(x,y)\} \qquad (15)$$

where $\mathbf{Layer}_i$ is the $i^t h$ 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 equation 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 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.

7 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 figure 6. Dimensions of the image are 720 x 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 in the distance.

Optic flow obtained with a multi-scale coarse-to-fine gradient method on a pair of frames is shown in figure 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 figure 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, shown in figure 11[c].

We used an initial segmentation map consisting of 75 square subregions. However, we found that the exact number of initial subregions 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 5 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 minutes.

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.

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

Figures 15, 16, 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 is pans right and zooms out.

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.

8 Compression

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

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 6 numbers per layer per frame.

We are able to code a 30-frame (1 second) video sequence of the flower garden using 300 kbits for a resolution of 360 x 240, and 700 kbits for a resolution of 720 x 480. Thus, the data rates are 300 and 700 kbit/sec respectively. 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.

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. But 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 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.

9 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. Figure 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 becaause 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.

10 Conclusions

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 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 efficient image coding. One can code 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.

This approach may provide useful tools in image analysis, frame-rate conversion, and special effects.

Acknowledgments

This research was supported in part by contracts with Television of Tomorrow Program, SECOM Co., and Goldstar Co. We thank Ujjaval Desai for compressing the layer images.

References

[1] E. H. Adelson. Layered representation for image coding. Technical Report 181, The MIT Media Lab, 1991.

[2] J. Bergen, P. Anandan, K. Hana, and R. Hingorini al. Hierarchial model-based motion estimation. In *Proc. Second European Conf. on Comput. Vision*, pages 237–252, 1992.

[3] J. Bergen, P. Burt, R. Hingorini, and S. Peleg. Computing two motions from three frames. In *Proc. Third Int'l Conf. Comput. Vision*, pages 27–32, Osaka, Japan, December 1990.

[4] M. J. Black and P. Anandan. Robust dynamic motion estimation over time. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, pages 296–302, Maui, Hawaii, June 1991.

[5] T. Darrell and A. Pentland. Robust estimation of a multi-layered motion representation. In *Proceedings IEEE Workshop on Visual Motion*, pages 173–178, Princeton, New Jersey, October 1991.

[6] R. Depommier and E. Dubois. Motion estimation with detection of occlusion areas. In *IEEE ICASSP*, volume 3, pages 269–273, San Francisco, California, April 1992.

[7] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of imgaes. *IEEE Trans. PAMI*, 6(6):721–741, November 1984.

[8] T. S. Huang and Y. P. Hsu. Image sequence enhancement. In T. S. Huang, editor, *Image sequence analysis*, pages 289–309. Springer-Verlag, 1981.

[9] M. Irani and S. Peleg. Image sequence enhancement using multiple motions analysis. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, pages 216–221, Champaign, Illinois, June 1992.

[10] R. Lenz and A. Gerhard. Image sequence coding using scene analysis and spatio-temporal interpolation. In T. S. Huang, editor, *NATO ASI Series, Vol. F2, Image sequence processing and dynamic scene analysis.* Springer-Verlag Berlin Heidelberg, 1983.

[11] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Image Understanding Workshop*, pages 121–130, 1981.

[12] H. G. Mussman, M. Hotter, and J. Ostermann. Object-oriented analysis-synthesis coding of moving images. *Signal Processing:* Image Communication *1*, pages 117–138, 1989.

[13] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985.

[14] L. H. Quam. Hierarchial warp stereo. In *Proc. DARPA Image Understing Workshop*, pages 149–155, New Orleans, Lousiana, 1984. Springer-Verlag Berlin Heidelberg.

[15] 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 Pattern Recog.*, pages 289–295, Maui, Hawaii, June 1991.

[16] C. W. Therrien. *Decision estimation and classification.* John Wiley and Sons, New York, 1989.

[17] J. Y. A. Wang and E. H. Adelson. Layered representation for image sequence coding. In *IEEE ICASSP*, volume 5, pages 221–224, Minneapolis, Minnesota, April 1993.

[18] J. Y. A. Wang and E. H. Adelson. Layered representation for motion analysis. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, pages 361–366, New York, June 1993.

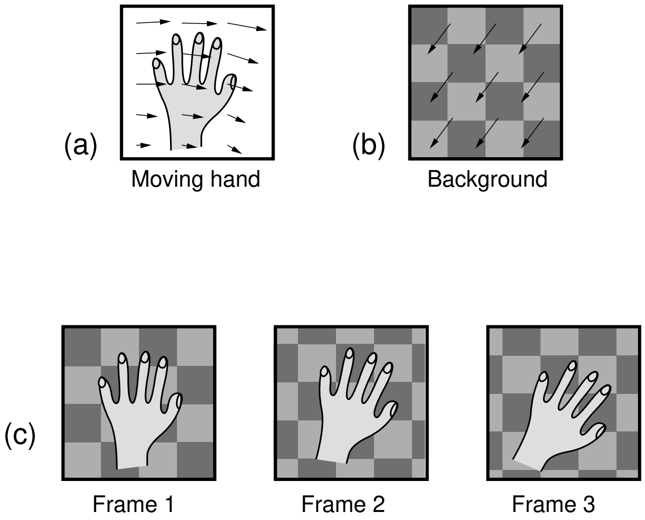


Figure 1: The objects involved in a hypothetical scene of a moving hand. (a) The hand, which undergoes a simple motion. (b) The background, which is translating down and to the left. (c) The observed image sequence.

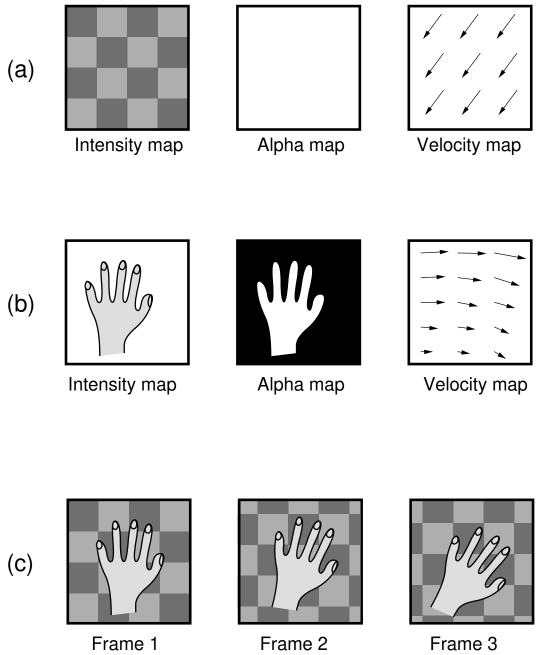


Figure 2: The desired decomposition of the hand sequence into layers. (a) The background layer. The intensity map corresponds to the checkerboard pattern; the alpha map is unity everywhere (since the background is assumed to be opaque); the velocity map is a constant. (b) The hand layer. The alpha map 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.

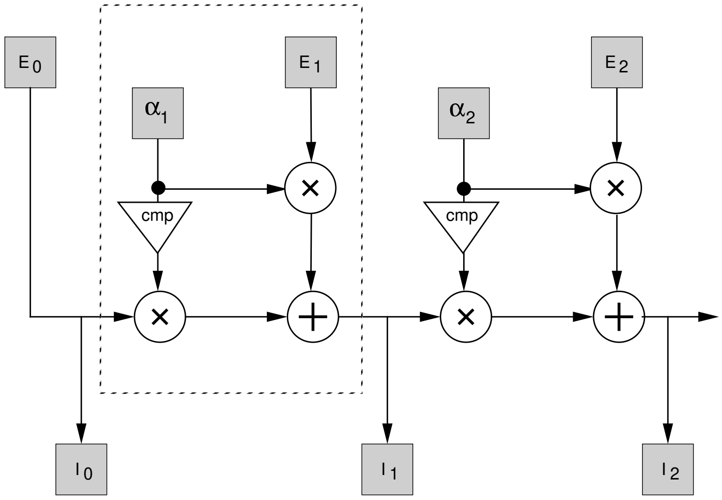


Figure 3: A flow chart for compositing a series of layers. The box labled "cmp" generates the complement of alpha, $(1-\alpha)$

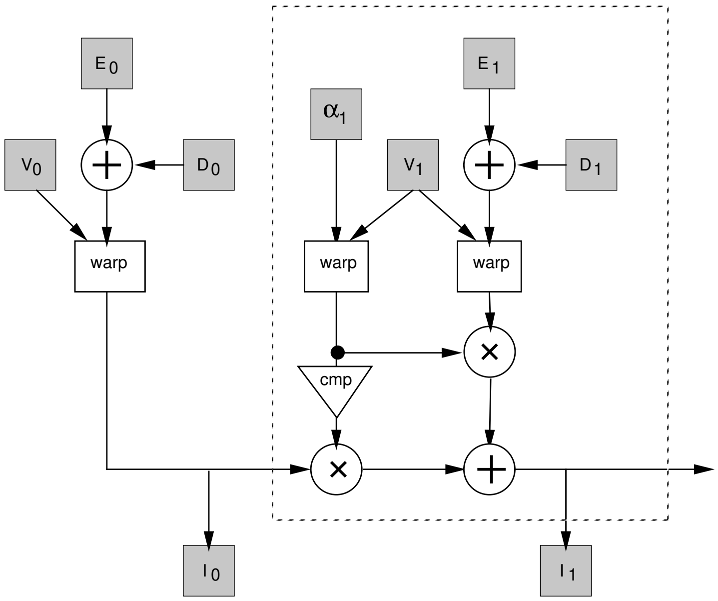
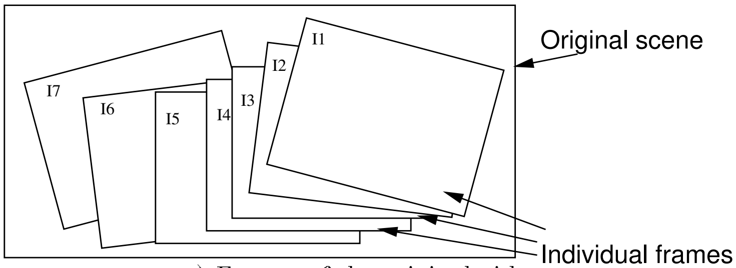
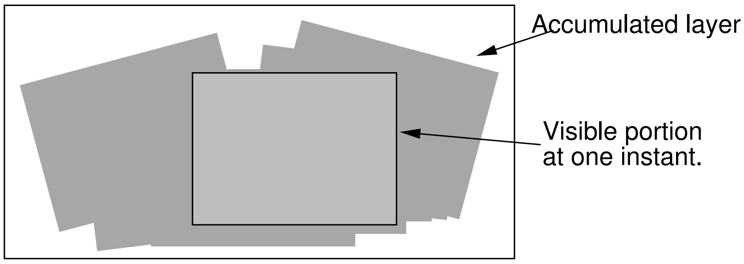


Figure 4: A flow chart for compositing that incorporates velocity maps, V, and delta maps, D.



a) Frames of the original video



b) Layer image accumulated from many frames

Figure 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.

a. Frame 0 of MPEG Flower garden sequence.

b. Frame 15 of MPEG Flower garden sequence.

c. Frame 30 of MPEG Flower garden sequence.

Figure 6: Frames 0, 15 and 30, of MPEG flower garden sequence.

a) Frame 1 aligned | b) Frame 15 as reference | c) Frame 30 aligned

Figure 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.

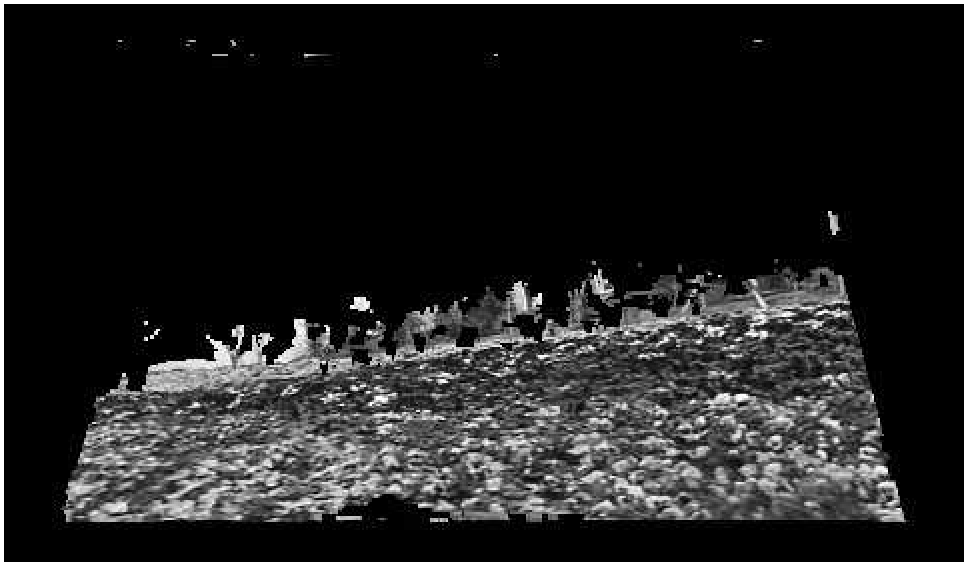


Figure 8: Accumulation of the flowerbed. Image intensities are obtained from a temporal median operation on the motion compensated images. Only the regions belonging to the flowerbed layer is accumulated in this image. Note also occluded regions are correctly recovered by accumulating data over many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

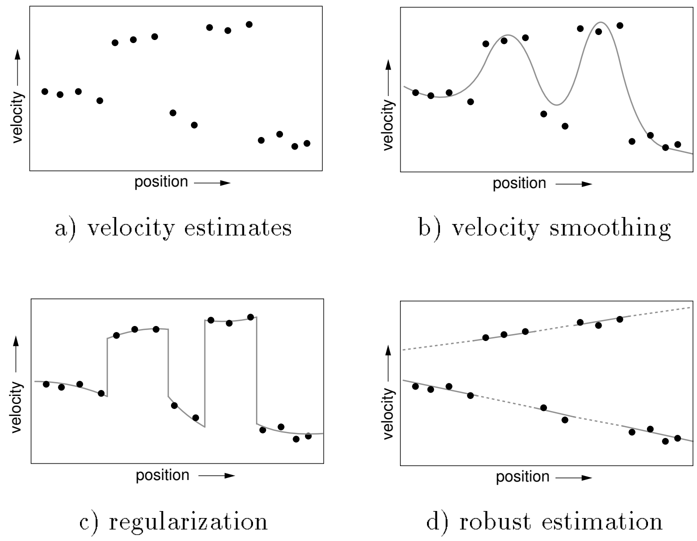


Figure 9: a) Velocity estimates appear as samples in space. b) Standard optic flow algorithms impose some smoothing on the velocity estimates in order to deal with noise. c) Regularization algorithms model the velocity field as a set of smooth regions while allowing for sharp breaks at model boundaries. 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.

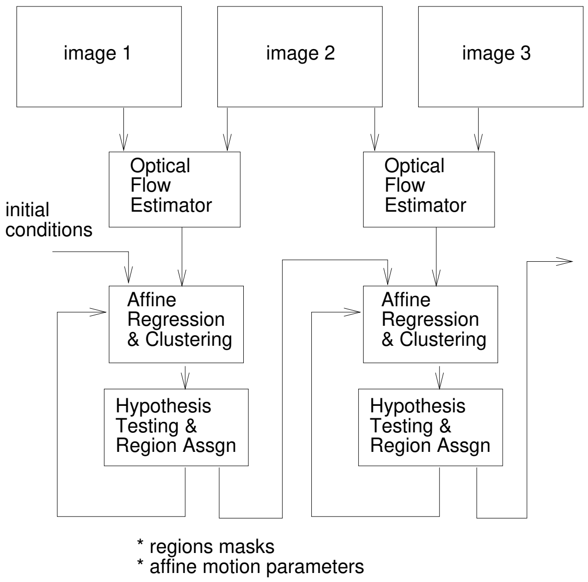
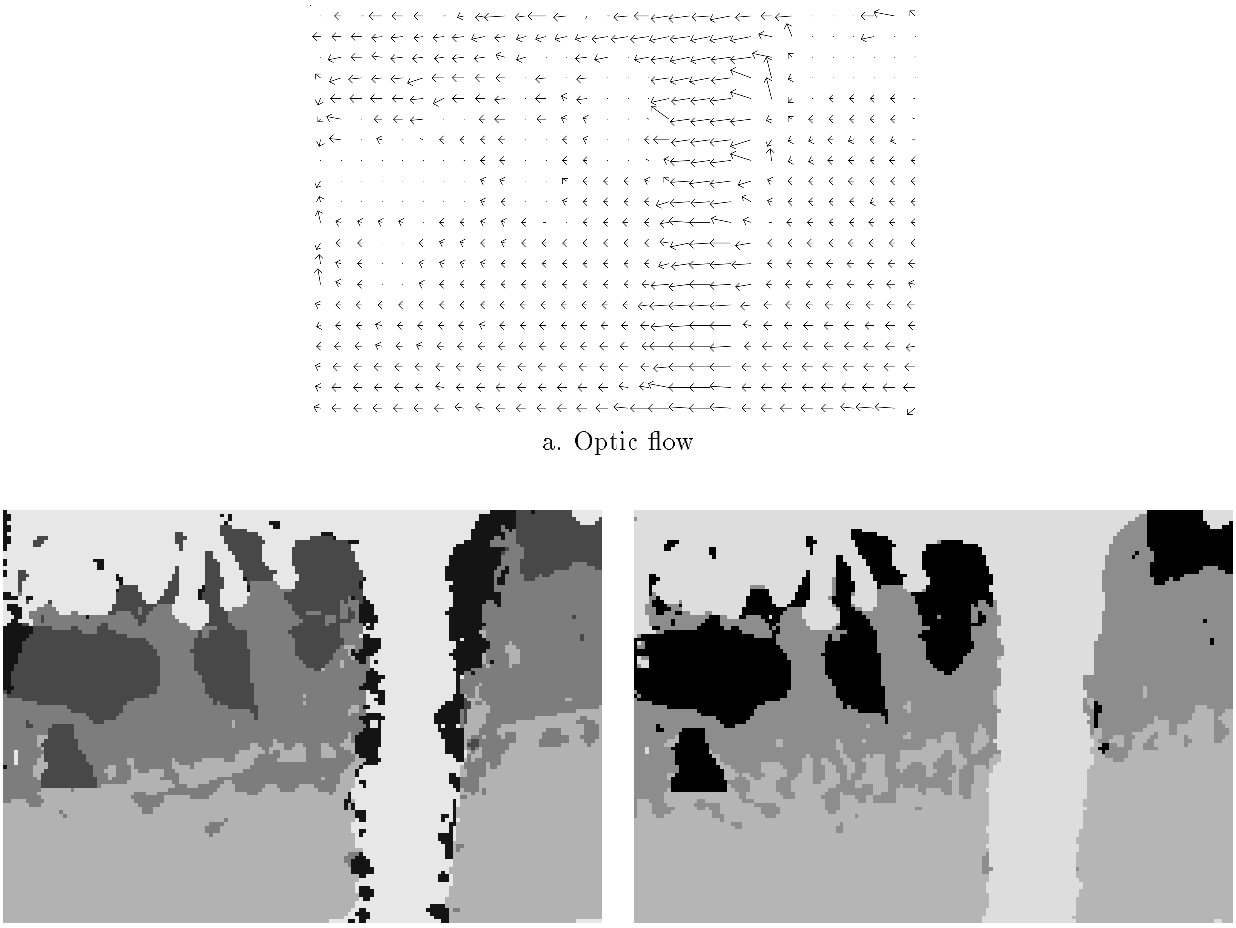


Figure 10: This figures shows the technique used in motion segmentation. Affine motion models are determined by regression on the dense motion fields and the regions are assigned to minimizes the error between the motion expected by the models and the estimated dense motion.

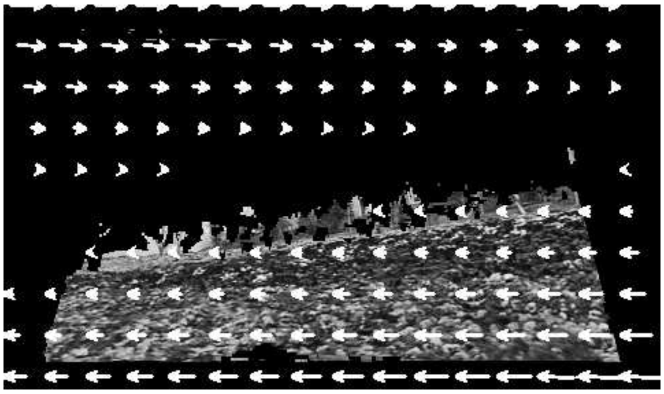


a. Optic flow

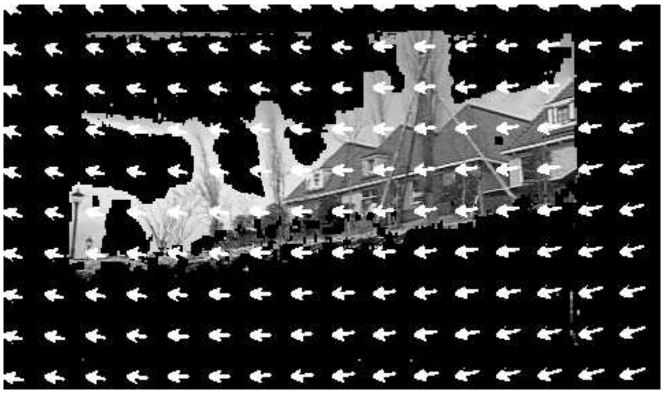
b. Motion segmentation

c. Refined motion segmentation

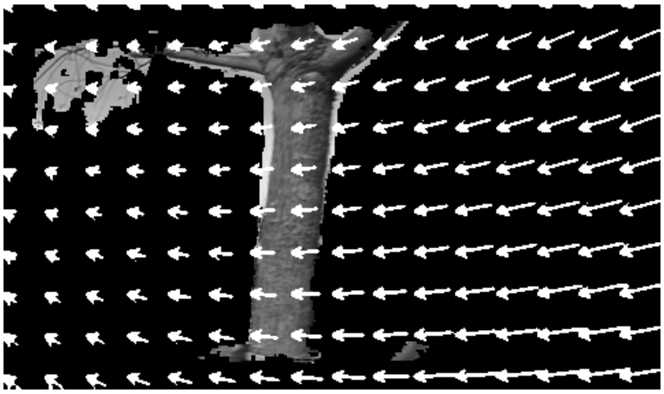
Figure 11: (a) The optic flow from multi-scale gradient method. (b) Segmentation obtained by clustering optic flow into affine motion regions. (c) Segmentation from consistancy checking by image warping. Representing moving images with layers.



a. Flower layer



b. House layer



c. Tree layer

Figure 12: The layers corresponding to the tree, the house, and the flower bed. The affine flow field for each layer is superimposed.

a. Synthesized frame 0

Synthesized frame 15

c. Synthesize frame 30

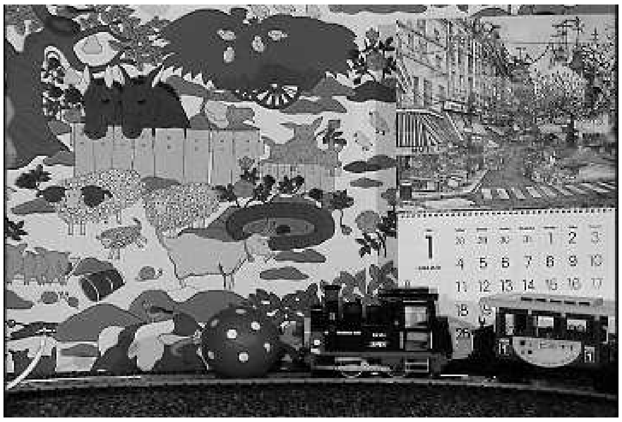
Figure 13: Frames 0, 15, and 30 as recontructed from the layered representation.

a. Frame 0

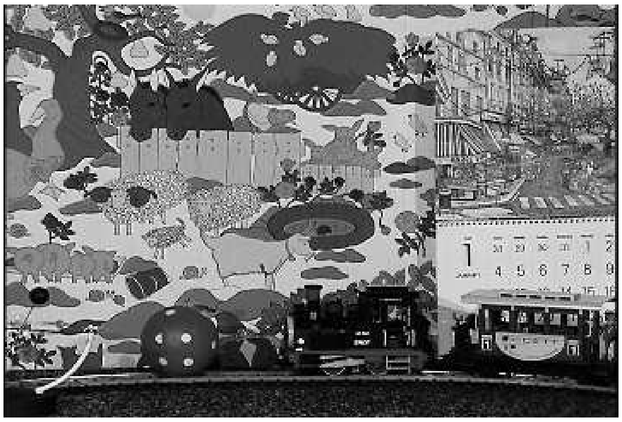
b. Frame 15

c. Frame 30

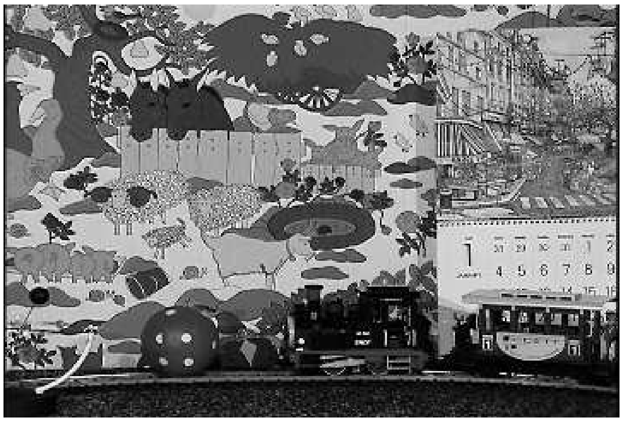
Figure 14: The sequence reconstructed without the tree layer.



a. Frame 1 of MPEG Calendar sequence.

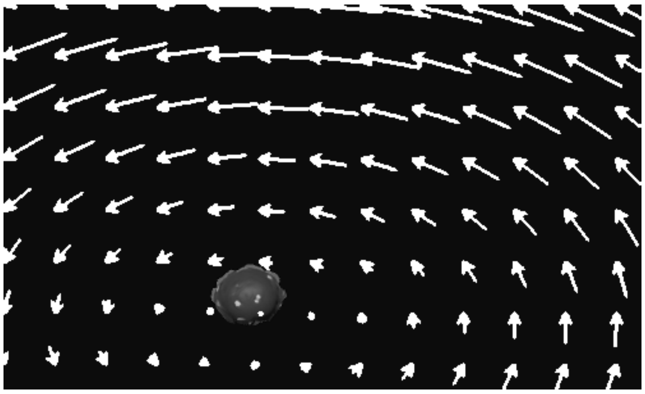


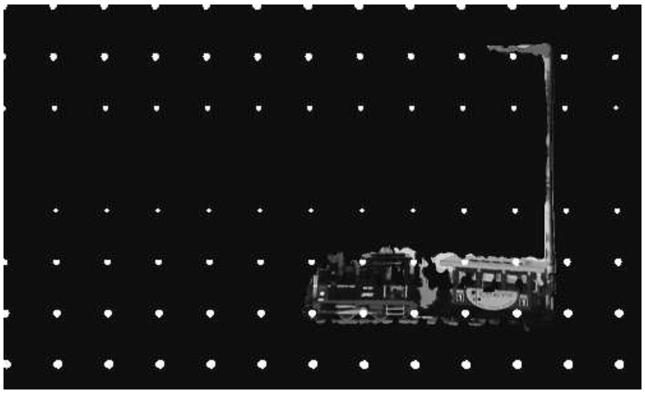
b. Frame 15 of MPEG Calendar sequence.



c. Frame 30 of MPEG Calendar sequence.

Figure 15: Frames 0, 15 and 30, of MPEG Calendar sequence.


a. ball layer


b. train layer

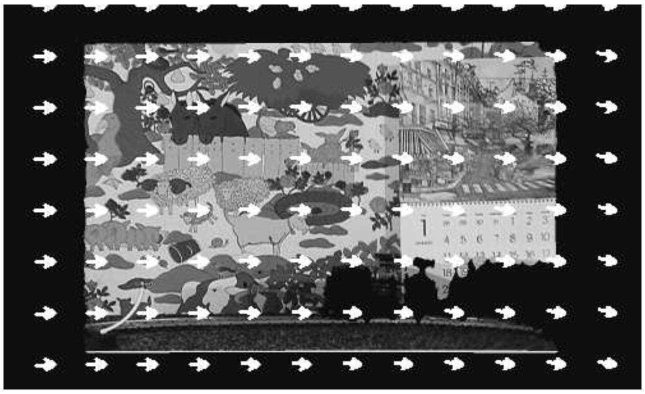
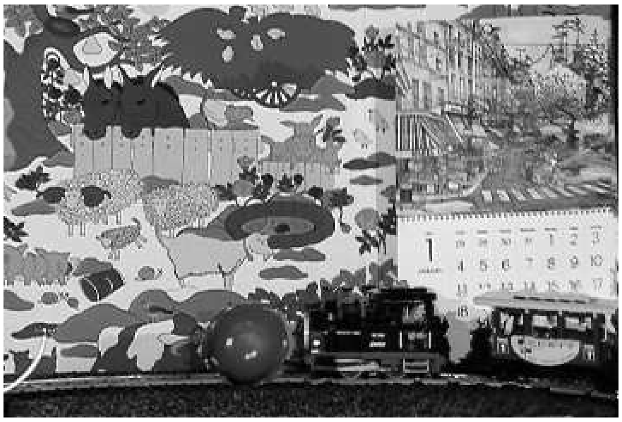
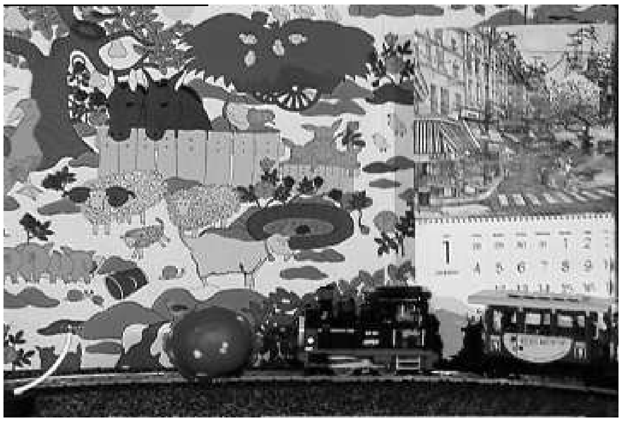

c. background layer

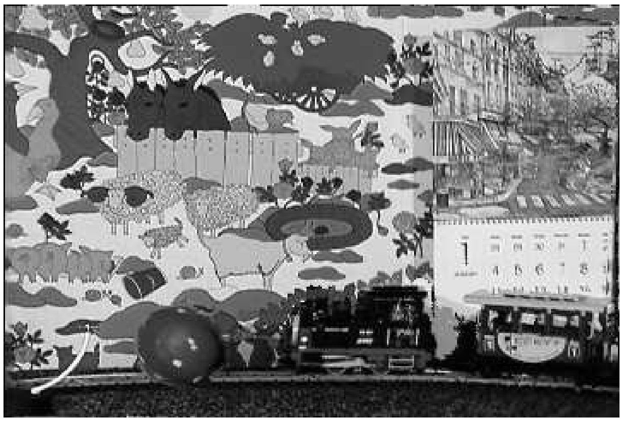
Figure 16: The layers corresponding to the ball, the train, and the background.



a. Synthesize frame 1 MPEG Calender sequence.



b. Synthesize frame 15 MPEG Calender sequence.



c. Frame 30 of MPEG Calendar sequence.

Figure 17: Synthesize frames 1, 15 and 30, of MPEG Calendar sequence. These Images are generated from the layer images in figure 16.