M.I.T. Media Laboratory Vision and Modeling Group, Technical Report No. 263, February 1994.
Appears in *Proceedings of the SPIE: Digital Video Compression on Personal Computers: Algorithms and Technologies, vol. 2187, San Jose, February 1994.*

Applying Mid-level Vision Techniques for Video Data Compression and Manipulation

John Y. A. Wang†, Edward H. Adelson‡, and Ujjaval Desai†

†Department of Electrical Engineering and Computer Science
‡Department of Brain and Cognitive Sciences
The MIT Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139

ABSTRACT

Most image coding systems rely on signal processing concepts such as transforms, VQ, and motion compensation. In order to achieve significantly lower bit rates, it will be necessary to devise encoding schemes that involve mid-level and high-level computer vision. Model-based systems have been described, but these are usually restricted to some special class of images such as head-and-shoulders sequences. We propose to use mid-level vision concepts to achieve a decomposition that can be applied to a wider domain of image material. In particular, we describe a coding scheme based on a set of overlapping layers. The layers, which are ordered in depth and move over one another, are composited in a manner similar to traditional "cel" animation. The decomposition (the vision problem) is challenging, but we have attained promising results on simple sequences. Once the decomposition has been achieved, the synthesis is straightforward.

1 INTRODUCTION

Vision systems and image coding systems face similar problems: how to represent images in ways that are efficient and useful. Current image coding systems use representations such as DCT's and pyramids, which are similar to those used in low-level vision systems. Someday we may have image coders that utilize high-level vision, including 3-D object recognition, but today this is impossible except for very restricted domains. A third approach is to develop an image coding systems based on "mid-level" vision, utilizing such concepts as global motion, grouping, surfaces, and regions. These concepts are sophisticated enough to produce powerful representations, and yet are simple enough to be computed. We have investigated a representation based on a set overlapping layers, where each layer contains information about color, intensity, transparency, and motion. The layered representation captures much of the visual coherence of the sequence, and it may allow for better data compression than standard techniques. The approach may be categorized with object-based methods.[10,13] Earlier descriptions of layered video coding were presented by Adelson and Wang .[1,19,20]

In the compositing methods of computer graphics, or in the production techniques used by traditional "cel" animators, the image representation includes the concept of 2-D images that are placed over one another as layers. Each layer occludes the one beneath it according to its transparency or opacity at each point. At the same time, each layer adds its own color and intensity. The final image is the composition of all the operations of attenuation and addition. In computer graphics the attenuation is represented by the "alpha" channel.

Figure 1 illustrates the concept with a hypothetical image sequence of a car moving against a background. A traffic light in the foreground occludes the car. This sequence can be decomposed into three layers shown in figure 2. These layers are: the traffic light in the foreground; the car in the mid-ground; and the building and tree in the background. Given the layers, a single frame of the sequence is generated by overlaying the layers and compositing according to their alpha maps. For simple occlusions, the alpha map takes on a value of 1 where the layer occludes the layers below it and 0 where the layers below are visible. When the alpha map takes on intermediate values, we can describe phenomena such as transparency and motion blur. Figure 3 illustrates how an image can be synthesized from the layers. A sequence of images is generated by applying the motion map to the intensity and alpha maps before compositing.

We propose a video coding system, shown in figure 4, whereby video is decomposed into the layered representation. A layer encoding module compresses the layer maps before transmission or storage. The compressed data is easily converted back into the layered representation for generating the original images or for creating a set of new images by video editing. We present an implementation of a layer analysis system based on affine motion segmentation and discuss compression of layer maps for coding applications. Video compression, background recovery, and video special effects are demonstrated by examples.

2 LAYER ANALYSIS

In block-based coding techniques, an image is divided into an arbitrary array of blocks and within each block a simple translational motion is estimated. Data compression is achieved by representing motion data by a blockwise description, resulting in transmission of only a small amount of motion data. Given the motion data, an image can be constructed from the available data in previous frames. However, this image model cannot accurately describe the scene. Objects do not usually fall within these block and motion coherence usually extends beyond these blocks. Systems that combine blocks with similar motion have demonstrated improvements in video compression.[11,14]

In our layer analysis, we perform motion segmentation on the image to determine the coherent motion regions. In contrast, segmentation in the block-based techniques is implicit by nature of the block processing. We also extend the simple translation model to an affine motion model to support more complex motions including: translation, rotation, zoom, and shear. The affine model is attractive for layer analysis because it is defined by only six parameters and the affine motion regions roughly correspond to moving 3-D planar surfaces. This model is described as follows:

$$V_x(x,y) = a_{x0} + a_{xx}x + a_{xy}y \quad (1)$$

$$V_y(x,y) = a_{y0} + a_{yx}x + a_{yy}y \quad (2)$$

where Vx and Vy are the x and y components of velocity, and the $a's$ are the parameters of the transformation. From the standpoint of motion data compression, affine motion segmentation represents the motion field with piecewise linear motion regions whereas block-based techniques represents motion with piecewise constant regions that are pre-defined. A number of authors have described methods for achieving piecewise affine decompositions.[3,6,9]

In addition to reducing data in the spatial dimensions, layer analysis also reduces video data in the temporal dimension. We assume the intensity profile of moving surfaces remains constant over time and the images in different frames of the sequence are simply "warped" by an affine transformation. By tracking these surfaces over time, we can identify the corresponding points for each regions to derive a single intensity map describing these surfaces.

The difficult task in layer analysis is segmentation, in particular, determining the coherent motion regions. Our segmentation framework consists of: local motion estimation; motion model generation; and region classification. Additional constraints on region size and connectivity are introduced in an iterative framework to improve stability and robustness. Our implementation of the segmentation algorithm is outlined in figure 5.

In this algorithm, motion segmentation is obtained by classifying each image location based on a set of affine motion models. We hypothesize these models by sampling the dense motion field, which is estimated by the optic flow estimator. The analysis begins at the region generator. Initially, the region generator segments the image into an array of square regions. Subsequently, the region generator creates new regions from unassigned regions. The model estimator calculates the affine parameters associated with each of the regions. Model merger combines multiple models having similar parameters to produce a single model. Region classifier assigns each pixel to the model that best describes its motion. Iteratively motion models are refined by estimating models within coherent motion regions. Constraints on region size and connectivity are

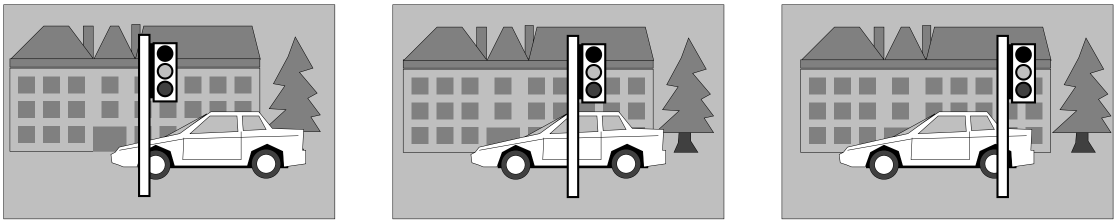


Figure 1: Three frames of a hypothetical scene.

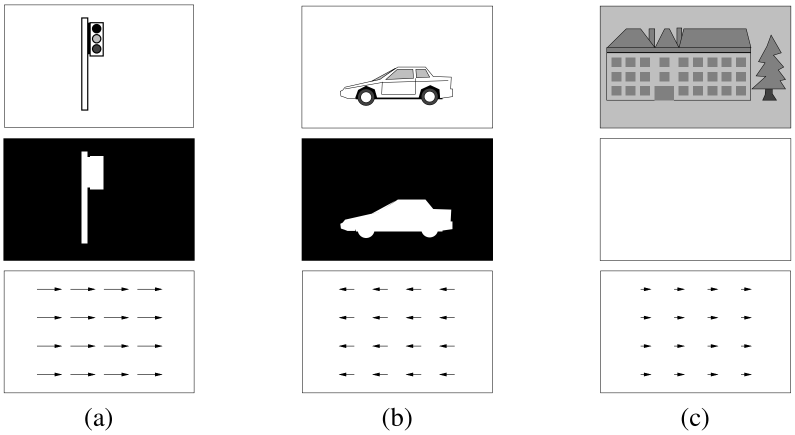


Figure 2: The layers involved in the hypothetical scene. Each layer consists of: an intensity map, an alpha map, and motion map. The scene is can be decomposed into: (a) a foreground traffic light layer; (b) A mid-ground moving car layer; and (c) a the background building and tree layer.

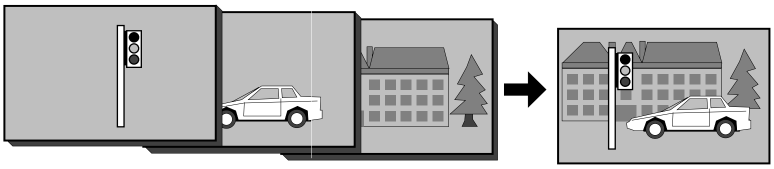


Figure 3: Three layers are composited according to their alpha map and their depth ordering to form a single image of the hypothetical sequence. Other frames of the sequence are generated by applying the motion transforms to each layer and compositing.

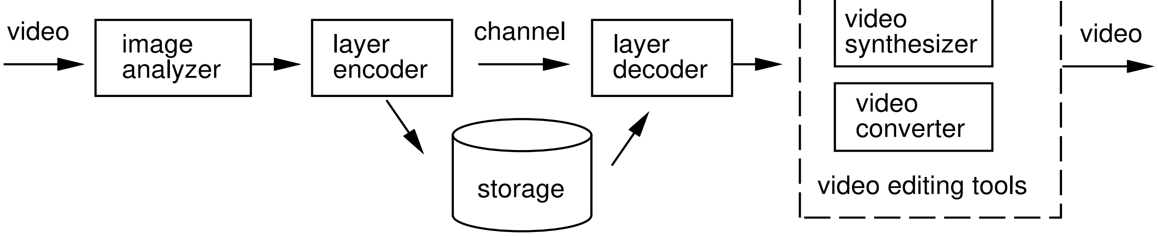


Figure 4: In concept in layer coding is illustrated in this diagram. The video data is decomposed into the layered representation by the video analyizer. The layers are compressed by the layer encoder before transmission or storage to further reduce bit rate.

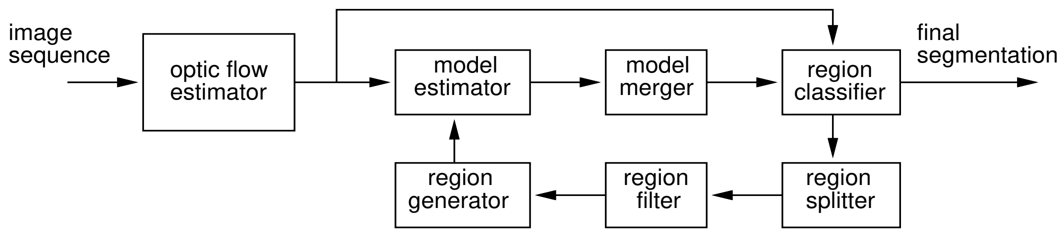


Figure 5: This figures shows the block diagram of the motion segmentation algorithm. Segmentation of the motion data is obtained by iteratively refining a set of motion hypotheses and classifying pixels based on this set.

enforced by the region splitter and region filter to provide added stability.

After the first iteration, the algorithm produces a set of coherent motion regions. By re-estimating the affine parameters within the new regions, we obtain better model estimates. At each iteration, the assignment improves because the models are estimated within coherent 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.

2.1 Optic flow estimation

Our local motion estimate is obtained with a multi-scale coarse-to-fine algorithm based on a gradient approach described by other authors.[2,12,15] The optic flow map describes the motion at each pixel location. The gradient approach uses the image intensity profile to determine the most likely displacement in a given analysis window. Like with block-matching, the underlying criterion for image motion is to find a displacement of an image region such that the sum of square difference (SSD) with that of the corresponding image is minimized:

$$\min_{V_x, V_y} SSD = \min_{V_x, V_y} \sum [I_t(x - V_x(x,y), y - V_y(x,y)) - I_{t+1}(x,y)]^2 \quad (3)$$

where I_t and I_{t+1} are the images at time t and $t+1$, respectively. Unlike in block-matching algorithm, which requires costly search for image displacement, in the gradient approach image displacement is obtained by linear estimation techniques. A linear least-squares solution for motion is obtained by first linearizing the right side of equation 3, then minimizing with respect to $V_x(x,y)$ and $V_y(x,y)$.

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum -I_x I_t \\ \sum -I_y I_t \end{bmatrix} \quad (4)$$

where, I_x, I_y, and I_t are functions of x, y, and t, and correspond to the partial derivatives of the image intensity with respect to x, y, and t, respectively. Non-integral displacement estimates are obtained for every image location (x,y). The summation is taken over a small region around the point (x,y). We use a multi-scale implementation to allow for estimation of large motions while maintaining low computational requirements.

2.2 Motion segmentation

The model estimator calculates affine parameter values for each region by applying standard linear regression on the motion data. This estimation 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 $\mathbf{a}_i^T = [a_{x0i}\ a_{xxi}\ a_{xy_i}\ \ a_{y0_i}\ a_{yx_i}\ a_{yy_i}]$ be the i^{th} hypothesis vector in the 6 dimensional affine parameter space with $\mathbf{a}_{x_i}{}^T = [a_{x0i}\ a_{xxi}\ a_{xy_i}]$ 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 1 and 2 can be simply written as:

$$V_x(x, y) = \phi^T \mathbf{a}_{x_i} \tag{5}$$

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

A linear least squares estimate of $\mathbf{a}_i$ of the motion field is obtained as follows:

$$[\mathbf{a}_{y_i} \;\; \mathbf{a}_{x_i}] = [\sum_{R_i} \phi \; \phi^T]^{-1} \sum_{R_i} (\phi [V_y(x, y) \;\; V_x(x, y)]) \tag{7}$$

The summation is taken over R_i the i^{th} motion region. Initially, model estimation is performed over an array of square regions generated by the region generator.

Next, the model merger finds similar models and produces a single representative model for each similar group, thus regions undergoing similar motion are assigned to the same motion model. The model merging process is carried out in the parameter space with a k-means clustering algorithm.[18] In k-means clustering, each model is treated as vector in parameter space. Initially a set of centers are chosen. Then each model vector is assigned to the nearest center forming clusters. A mean vector is calculated for each cluster and used as the new center. Model clustering allows us to describe multiple model with a few representative ones.

The region classifier identifies the coherent motion regions by assigning each pixel location to one of the given motion models. A motion map for each affine model is created, and each location assigned to the model that best describes the motion at that location. Even though motion models are initially estimated within blocks, pixel level resolution on segmentation is obtained by model testing at each image location. In the classification, we minimize the following cost function, $C(i(x, y))$:

$$\min_i \; C(i(x, y)) = \min(\mathbf{V}(x, y) - \mathbf{V}_{\mathbf{a}_i}(x, y))^2 \tag{8}$$

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 i^{th} affine motion hypothesis.

2.3 Temporal data reduction

The layer 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, segmentation of consecutive frames will be similar resulting in fewer iterations for convergence. 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. When the motion segmentation on the entire sequence is completed, each affine motion region will be identified along with its affine motion parameters.

After calculating the affine motions and identifying the regions for the entire sequence, an intensity map is produced for each coherent motion region. For each pixel in the intensity map, corresponding intensity values for each frame in the sequence are collected and the median value is obtained for the intensity map. In this processing, the video data is temporally reduce into a few intensity maps. In addition to data compression, earlier studies have shown that motion compensated median filtering can enhance noisy images[8] and temporal accumulation of data can produce higher resolution images.[9]

The reliability of intensity map can be obtained by calculating the variance in the median operation or by simply using the number of points in the operation as an indicator. Furthermore, we assume a simple occlusion model so that alpha maps simply take on a value of 1 in regions where intensity values are reliable and 0 where they are not.

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. 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.

3 LAYER ENCODING

Video compression is achieved when the video data is decomposed into the layered representation. However, further data compression can be obtained by efficiently encoding the layer maps, shown in figure 2, in particular, the intensity map and the alpha map. The motion map requires no additional encoding because each map is described by only six numbers per layer per frame.

We use standard still image coding techniques, for example JPEG,[16] for layer encoding. The JPEG algorithm uses discrete cosine transform[17] (DCT) on 8 x 8 blocks to achieve good energy compaction. The DCT has an orthogonal basis, which is composed of cosines of different frequencies. Usually a few DCT coefficients contain most of the input energy. Therefore high image compression can be achieved by retaining only these coefficients.

A simple approach to compressing the layer maps with JPEG is to derive a single alpha-intensity image obtained by simply setting all unsupported regions to a known constant value, for instance zero. However, the boundary between the supported and unsupported regions creates artificial discontinuities in intensity. The presence of these intensity edges in the alpha-intensity image causes the JPEG algorithm to perform poorly because these edges result in energy spread in the frequency domain, thus considerably reducing the DCT energy compaction. We have developed a better method that separately compresses the intensity and alpha maps.

3.1 Intensity map encoding

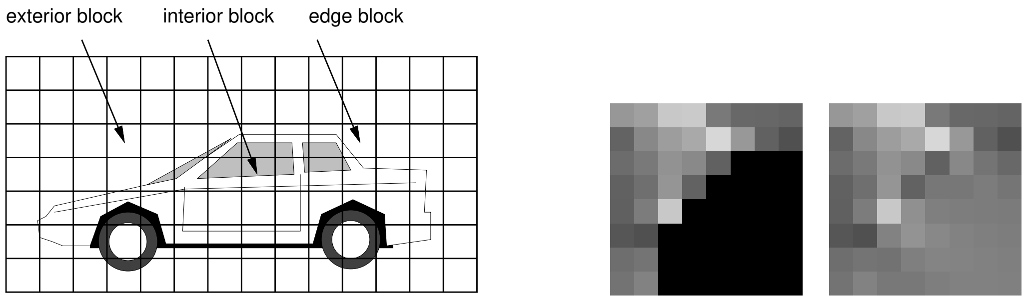
In our encoding algorithm, we first classify all possible 8 x 8 blocks of an intensity map into three categories: interior blocks, in which all pixels are supported; edge blocks, in which a fraction of pixels are supported; and exterior blocks, in which no pixels are supported. Figure 6(a) shows these blocks for a typical intensity map.

The JPEG algorithm can be directly used for the interior blocks because these blocks do not have artificial intensity edges. Since the exterior blocks contain no information about the layer, they can be coded with minimal bit rate. For better compression, the edge blocks require a special encoding algorithm.

Consider an edge block shown in figure 6(b) where unsupported pixels are shown in white. In our algorithm, we reduce the edges by filling the unsupported locations with values that result in a smooth transition from the supported regions. Each unsupported pixel is recursively assigned the average intensity of its neighboring pixels. The result of this filling algorithm is shown in figure 6(c). When edge blocks are modified by smoothly filling the unsupported regions, the distortion introduced by JPEG quantization is reduced as compared with filling these regions with arbitrary values. In the JPEG algorithm, high frequencies are quantized more coarsely than low frequencies, therefore introducing low frequency energy by smooth filling results in less distortion. This reduction in distortion is most noticeable when compressing these blocks at low JPEG quality factors.

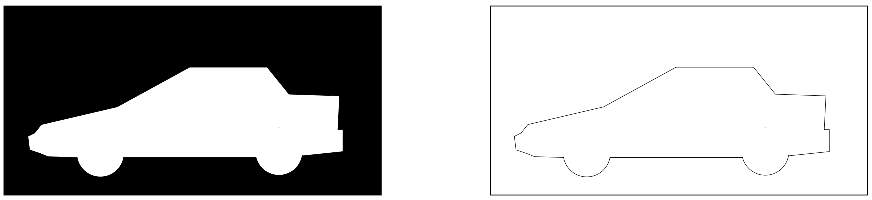
3.2 Alpha map encoding

Figure 7(a) shows an alpha map. The alpha map is completely represented by its boundary contour, see figure 7(b), hence our alpha map encoding algorithm is simply a contour encoding algorithm. In this figure we show a single contour, however, more complex layers may have multiple contours. We use a chain coding algorithm to encode a contour. In this algorithm



(a) (b) (c)

Figure 6: (a)The exterior, interior, and edge blocks of an intensity map; (b) an edge block; (c) the modified edge block.



(a) (b)

Figure 7: (a) An alpha map; (b) a contour description of the alpha map.

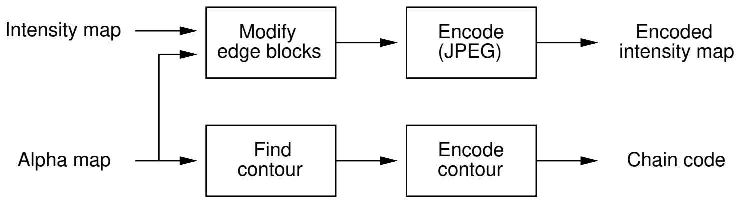


Figure 8: This figure outlines the block diagram of the layer encoder. The intensity map is encoded by first modifying the edge blocks and then performing JPEG compression on the modified map. The alpha map is encoded by first extracting its contour and then applying the contour encoding algorithm.

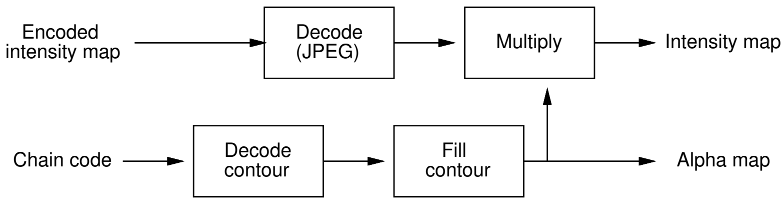


Figure 9: The decoder block diagram is shown here.

a contour is represented by an initial point and a sequence of directionals indicating the position of the next point on the contour. There are only 8 distinct direction vectors, one corresponding to each of the nearest neighbors, thus requiring only 3 bits per directional. However, the chain codes can be further compressed if there are long runs of a particular directional vector. We used the Lempel-Ziv compression algorithm[5] on the chain codes to produce a very compact description of the alpha map.

The complete encoding and decoding processes are outlined in figure 8 and figure 9, respectively. Standard JPEG decompression algorithm is used to decode the compressed intensity map. Using a simple contour decoding algorithm, the transmitted chain codes are decoded to obtain contours of the alpha map. The contours are then filled to get the required alpha map.

More complex algorithms, such as region-oriented transform coding,[4,7] can be used to obtain better compression. However, we are able to achieve good results with our faster algorithm.

4 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 10. 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 is in the distance.

4.1 layer analysis results

Optic flow obtained with a multi-scale coarse-to-fine gradient method on a pair of frames is shown in figure 11(a). The initial segmentation map for the first frame is shown in figure 11(b). The block dimensions are 20 x 20 pixels and the minimum allowable size of regions is 400 pixels. Maximum velocity error tolerated between the optic flow and the affine motion model in the region classifier is 1 pixel/frame. Segmentation converges after 12 iterations resulting in 5 coherent motion regions as shown in figure 11(c). 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.

The layered representation for this sequence is shown in figure 12. We used Frame 15 as the reference frame for the intensity map extraction. 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 intensity maps, alpha maps, and the affine motion parameters. Figure 13[a] shows one frame of synthesized sequence from layers. The objects are ordered and placed in their respective positions with the correct occlusions.

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 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.

4.2 Compression results

Using our layer encoding algorithm, we encoded 30 frames of the flower sequence, 1 second of color video, at JPEG quality factors of 80 and 45 with only 1.06 Mbits and 660 Kbits, respectively. Applying JPEG, with quality factors of 80 and 45, directly to the composite alpha-intensity images resulted in bit rates of 1.54 Mbits and 1.04 Mbits, respectively. These numbers clearly indicate the improvement in compression obtained by our encoding algorithm. Other edge block modification methods, that provide better compression with JPEG, are currently being studied.

The alpha maps, which are encoded losslessly, required only about 70 Kbits. If further reduction in the bit rate is desired, the alpha map contours can be simplified by using morphological operators such as erosion and dilation. Motion parameters for the sequence required 40 Kbits.

The average distortions, in terms of root mean squared error(RMSE), for our algorithm at quality factors of 80 and 45 were 2.865 and 3.77, respectively, whereas for the composite alpha-intensity encoding method the corresponding distortions were 3.345 and 5.04, respectively. We use a simple RMSE distortion measure: $RMSE = \frac{1}{N}\{\sum[I(x,y) - \hat{I}(x,y)]^2\}^{\frac{1}{2}}$; where I denotes the original image with N pixels and $\hat{I}$ denotes the reconstructed image.

It is evident from experimental results that our layer encoding algorithm not only gives improvement in compression, but also provides higher quality.

5 OTHER APPLICATIONS

Video compression is achieved by the layered representation because the representation exploits the spatial and temporal coherences in the data. Unlike block-based coders, video data is decomposed into meaningful object-like primitives with interesting motions much like those used in computer graphics, thereby facilitating manipulation of video data. In figure 13(b), we show an example of video editing. This image is generated by compositing all the layers of the flower sequence except the ones corresponding to the tree. Block-based techniques cannot synthesize this image without the tree. The occluded regions are correctly recovered because in layer analysis we build a description for these regions by collecting data over many frames.

Our examples of the flower sequence also show that background and foreground recovery is easily accomplished by the layer analysis. Note that an ordinary background memory could not achieve the effect shown in figure 13(b) because the various regions of the scene are undergoing different motions.

The layered representation also provides frame rate flexibility. 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 achieved by using the layered format. Likewise, our analysis can decompose sequences in raster interlaced or progressively scanned formats into the layered representation, and subsequently, the video synthesizer can generate the sequence in either format.

6 CONCLUSIONS

We present a layered image representation that utilizes mid-level vision concepts for compression of video data. These concepts, which include motion estimation, image segmentation, and representation of coherent motion surfaces, exploit the spatial and temporal coherences of moving images. In this representation, an image sequence is described by a set of layers consisting of an intensity map, an alpha map, and motion information.

We describe an algorithm for layer analysis based on a robust motion segmentation framework. In this framework, we iteratively determine the coherent motion regions from a dense motion field. A set of affine motion models are derived by sampling the motion field. Segmentation is obtained by assigning each location to an affine motion model. Coherent motion regions are tracked across many frames and a single layer intensity map is derived for each region.

The layer analysis allows a sequence to be represented by a few layers resulting in compression of video data. We propose a simple algorithm for encoding the layer maps. The algorithm separately encodes the intensity and alpha maps to reduce both bit rate and distortion.

Because the layered representation is similar to representations utilized in computer graphics, layer analysis facilitates the manipulation of video data such as background recovery, video editing and special effects, and frame rate conversion. We present results of layer analysis on a real sequence.

Acknowledgments

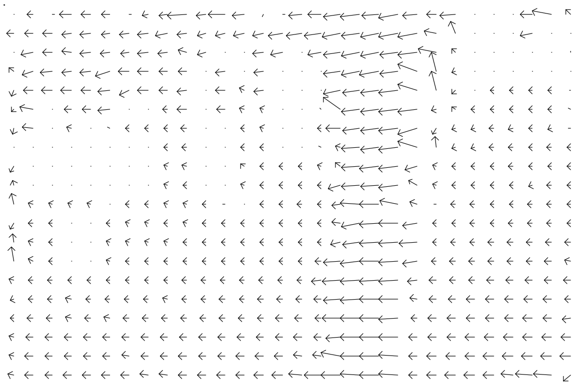
This research was supported in part by contracts with Television of Tomorrow Program, SECOM Co., and Goldstar Co.

(a) Frame 1

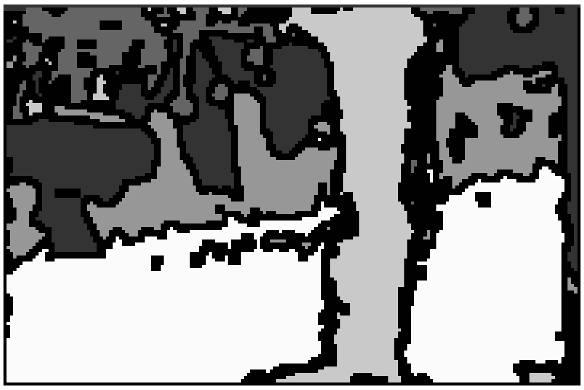
(b) Frame 15

(c) Frame 30

Figure 10: Three frames of MPEG flower garden sequence.



(a) motion field



(b) motion segmentation

Figure 11: Results of dense motion field estimation on first pair of frames is shown in (a). Segmentation of the motion field with affine motion models is shown in (b). Optic flow

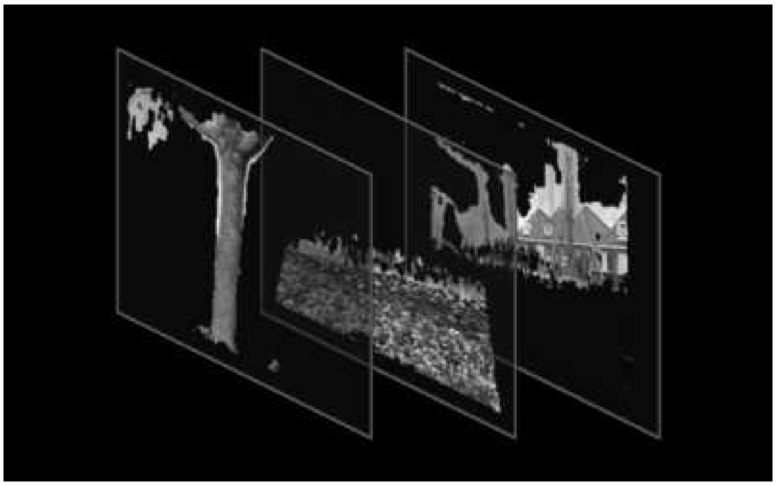


Figure 12: This figure shows layered representation of the flower sequence. Three intensity maps corresponding to the tree, flower bed, and house are shown here along with their depth ordering. Unsupported regions are shown here in black.

(a)

(b)

Figure 13: (a) Frames 1 as reconstructed from the layered representation. (b) Frames 1 reconstructed from layers without the tree.

7 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. Biggar, O. J. Morris, and A. G. Constantinides. Segmented-image coding : performance comparison with the discrete cosine transform. *IEE Proc. Part F*, 135(2):121–132, April 1988.

[5] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., New York, 1991.

[6] 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.

[7] M. Gilge, T. Engelhardt, and R. Mehlan. Coding of arbitrarily shaped image segments based on a generalized orthogonal transform. *Signal Processing:* Image Communication *1*, pages 153–180, 1989.

[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] S. Liu and M. Hayes. Segmentation-based coding of motion difference and motion field images for low bit-rate video compression. In *IEEE ICASSP*, volume 3, pages 525–528, San Francisco, California, April 1992.

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

[13] 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.

[14] H. Nicolas and C. Labit. Region-based motion estimation using deterministic relaxation schemes for image sequence coding. In *IEEE ICASSP*, volume 3, pages 265–268, San Francisco, California, April 1992.

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

[16] M. Rabbani and P. W. Jones. *Digital Image Compression Techniques*, pages 113–128. SPIE - The International Society for Optical Engineering, Washington, 1991.

[17] K. R. Rao and P. Yip. *Discrete Cosine Transform*. Academic Press Inc, 1990.

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

[19] 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.

[20] 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.