

# Motion Segmentation by Spatiotemporal Smoothness Using 5D Tensor Voting

Changki Min and Gérard Medioni  
Integrated Media Systems Center  
University of Southern California  
Los Angeles, CA 90089, USA  
{cmin,medioni}@usc.edu

## Abstract

*Our goal is to recover temporal trajectories of all pixels in a reference image for the given image sequence, and segment the image based on motion similarities. These trajectories can be visualized by observing the 3D  $(x, y, t)$  spatiotemporal volume. The mathematical formalism describing the evolution of pixels in time is that of fiber bundles, but it is difficult to implement directly. Instead, we express the problem in a higher dimensional 5D space, in which pixels with coherent apparent motion produce smooth 3D layers. The coordinates in this 5D space are  $(x, y, t, v_x, v_y)$ . It is initially populated by peaks of correlation. We then enforce smoothness both in the spatial and temporal domains simultaneously, using the tensor voting framework. Unlike the previous 4D approach which uses only two frames, we fully take advantage of the temporal information through multiple images, and it significantly improves the motion analysis results. The approach is generic, in the sense that it does not make restrictive assumptions on the observed scene or on the camera motion. We present some results on real data sets, and they are very good on even challenging image sequences such as serious occlusion.*

## 1. Introduction

Motion segmentation is one of the most important aspects of video sequence analysis, and serves various computer vision applications such as object tracking, surveillance, robot navigation, structure from motion and event detection. The main goal of motion segmentation is to divide an image into regions based on the motion of individual pixels. Here, the most important constraint which also defines the 'meaningful regions' is the *coherence* of the motion of pixels in a region. In our proposed approach, we observe the motion across multiple frames so that each pixel is associated with a long temporal trajectory. Each region with coherent motion constitutes a layer. The lay-

ered representation for motion analysis was introduced by Wang and Adelson [17], and adopted by other researchers [4][9][3][18]. Most of them use an affine motion model, which tends to fail when the scene is close to the camera. Our approach, however, does not require any assumptions regarding camera model, motion model, the number of objects, or scene structure except a spatiotemporal smoothness constraint. Many algorithms for two frame motion segmentation have been proposed. For instance, Ayer and Sawhney [4] estimated motion parameters based on an ML estimation of mixture models and the MDL encoding principle, and Kolmogorov and Zabih [12] proposed a graph cut approach to minimize an energy function for the correspondence problem. However, generalizing the above methods to multiple frames is not trivial. Nicolescu and Medioni [14] first proposed to use a layered representation within the tensor voting framework in a 4D space. Although their motion segmentation results are usually accurate, it often fails when the motion boundary regions have complex textures. Also it does not take advantage of rich temporal information of motion. There are many approaches which utilize multiple frames for motion analysis, and the factorization method is one of the most popular techniques. It was introduced by Tomasi and Kanade [15], and has been significantly improved by other researchers [8][11][16][19]. For instance, Costeira and Kanade [8] introduced the shape interaction matrix for multibody factorization, and some solutions for its limitations have been presented in [16][19]. Most of factorization approaches, however, are sensitive to noises, and assume linear models such as affine. A convenient representation of a video sequence is a spatiotemporal 3D  $(x, y, t)$  volume as illustrated in Figure 1. The left figure shows the given image sequence, and the right figure shows their  $(x, y, t)$  spatiotemporal volume representation. This was very well exploited by Bolles et al [6], in the case of translation. Extensions to more general motions are more difficult [5]. While it is easy to visualize the motion in this space, the constraints are difficult to implement.

The appropriate mathematical formalism to describe

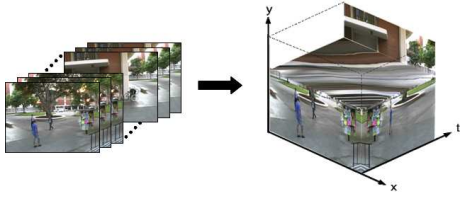


Figure 1. Spatiotemporal volume representation

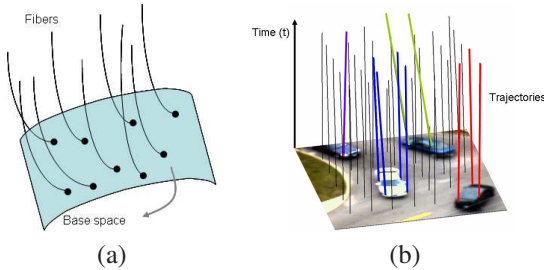


Figure 2. Fiber bundle representation. (a) fiber bundle concept, (b) fiber bundle representation for motion analysis

these structures is that of fiber bundles [10]. A fiber bundle is a map  $p : E \rightarrow B$ , where  $B$  is the *base space*,  $E$  is the *total space*, and the map  $p$  is called the *projection* of the bundle. For each point  $b \in B$  in the base space, the space  $p^{-1}(b)$  is called *fiber* of the bundle over  $b \in B$ . This is shown in Figure 2(a). For the motion problem, we assign the reference image as the base space. Then, the temporal trajectory of each pixel in the reference image forms a fiber as can be seen in (b). Although the fiber bundle formalism gives a good representational framework, it does not provide any tools to solve the stated motion problem. Therefore, we convert the fiber bundle representation to a 5D space,  $(x, y, t, v_x, v_y)$ , in which  $(x, y)$  represent the pixel coordinates in the reference image,  $t$  represents time (frame number), and  $(v_x, v_y)$  represent the velocity of a pixel in the reference image at each time instance  $t$ . In this 5D space, each moving object is represented as a smooth layer on the  $(x, y, t)$  domain, and we enforce our motion-related constraint to the layer. The final smoothed layers are then converted back to the fiber bundle representation to obtain temporal trajectories of the pixels in the reference frame.

In our proposed approach, we assume only a single motion property: spatiotemporal smoothness. In other words, we assume that each pixel in the reference frame moves smoothly in time (i.e., temporal smoothness constraint), and a set of pixels which belong to a single object moves in a similar way (i.e., spatial smoothness constraint). Those two constraints are enforced simultaneously in the 5D space. It is important to note that the spatiotemporal smoothness of motion is represented as the *smoothness* of layers in the 5D space, thus, our main tasks are smoothing the layers, and identifying individual layers each of which corresponds to

an independent moving object. To enforce the spatiotemporal smoothness constraint in the 5D space, we use the tensor voting framework [13].

This paper is organized as follows. In the following section 2, we present the structure of the 5D space and the 5D tensor voting framework, and then the details of our proposed algorithm are explained in section 3. Section 4 shows the results of some sequences after processing with our algorithm, followed by our conclusion in section 5.

## 2. Data representation in 5D

### 2.1. Analogy in 1D image

In order to provide some intuition for the readers, and to justify the embedding into a higher dimensional space, let us consider a 1D image motion. Figure 3(a) shows a synthetic 2D image sequence, where the background moves to the right side and the car moves to the left side. The pixels in every frame have only horizontal motion with constant speeds so that observing only a single scan line is enough to analyze the motion of the sequence. The red lines in (a) indicate the scan lines we used for analysis, and they can be regarded as a set of 1D images. (b) is the vertical stack of all the 1D images, and it is analogous to the spatiotemporal volume of 2D image sequences. The red line in (b) indicates the reference 1D image, and the numbers from 1 to 3 indicate independent objects. The objects 1 and 3 are in fact a single background which is occluded by the foreground object 2 (i.e., car). By finding all correspondences of the pixels in the reference image through other images, we can generate temporal trajectories (or fibers) as can be seen in (c). Due to the occlusion between the foreground and background objects and horizontal camera motion, the initial trajectories are so much noisy. Since the spatiotemporal smoothing for fibers and the pixel grouping based on similar motion are difficult to implement in the representation, we convert the fiber bundle representation to the proposed 3D space (5D space in the case of 2D images). As can be seen in (d), the 3D space has three components: image domain  $x$ , time domain  $t$ , and velocity domain  $v$ . Thus, a point in the 3D space indicates the velocity  $v$  of a pixel  $x$  in the reference frame at a given moment  $t$ . The initial layers in the 3D space include large number of outliers since they were generated from noisy trajectories. Through the tensor voting stage, those outliers can be easily detected and removed, and their correct positions in the space are estimated via the densification stage resulting in very smooth layers as shown in (e). The smoothness of each layer along the time axis represents the temporal smoothness constraint of motion, while the smoothness of each layer along the image axis represents the spatial smoothness constraint of motion. These facts explain how we enforce both spatial and temporal smoothness constraints simultaneously through ten-

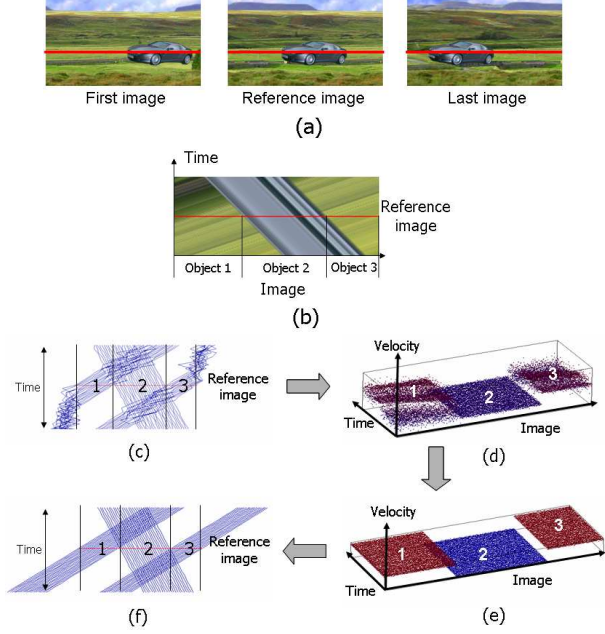


Figure 3. 1D image example. (a) original 2D synthetic images, (b) vertical stack of all 1D images, (c) initial noisy temporal trajectories, (d) initial noisy layers, (e) smoothed layers by tensor voting, (f) spatiotemporally smoothed trajectories.

sor voting. The numbers on layers in (d) and (e) indicate their associated objects in (b) and (c). Hence, it is obvious that finding independently moving objects is equivalent to identifying each layer in (e). The final temporal trajectories computed from the smooth layers in (e) are shown in (f). Through the spatiotemporal smoothness constraint, we can smooth all the trajectories of the pixels in the reference frame. This also means that the full continuous motion of objects which are partially or fully occluded in some frames can be recovered as shown in (f).

## 2.2. Description of 5D space

The previous 1D image example is a special case where all the pixels have only horizontal motion. In general sequences, however, each pixel can move in any directions meaning that it has both vertical and horizontal velocity components. Thus, the layered representation of 2D image sequences can be described in a 5D space instead of the 3D space for 1D image sequences. In fact, the 5D space has the same components as the 3D space: image domain, time domain, and velocity domain. However, the image domain is now  $(x, y)$  instead of a single  $x$ , and the velocity domain is  $(v_x, v_y)$  instead of a single  $v$ , resulting in the 5D space  $(x, y, t, v_x, v_y)$ . Analogously, a point in the space represents the velocity  $(v_x, v_y)$  of a pixel  $(x, y)$  in the reference frame at the given time  $t$ .

More specifically, we define a velocity vector function

$V$  over the 3D spatiotemporal domain,  $D = I \times T$ , where  $I \subset \mathbb{R}^2$  represents the pixels in the reference frame, and  $T \subset \mathbb{R}$  represents time, i.e.,

$$V : I \times T \rightarrow \mathbb{R}^2, (x, y, t) \mapsto (v_x, v_y).$$

In practice, we have to deal with discrete image coordinates and time. However, the velocities are still real numbers due to subpixel accuracy computation. In our approach, all the initial mappings from  $(x, y, t)$  to  $(v_x, v_y)$  are basically obtained from measurements (i.e., finding correspondences), and they are represented in the 5D space altogether forming layers. Each 5D token now communicates with its neighboring tokens through the voting process, and the tokens finally generate smooth layers. Note that our framework does not try to compute an analytic form of the function  $V$ . Rather, each final  $(v_x, v_y)$  at  $(x, y, t)$  is obtained by voting which is a spatiotemporal process in the 5D space.

## 2.3. Tensor voting in 5D space

Due to limited space, we do not present all the details of the tensor voting framework here. Rather, we refer readers to [13] for the general tensor voting theory, and [14] for the 4D framework for two frame motion segmentation. In this section, we simply explain the basic concept of the 5D tensor voting.

In the tensor voting framework, each input point is encoded as a tensor which is a symmetric nonnegative definite matrix whose eigenvalues are  $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$  in descending order, and the corresponding eigenvectors are  $e_1, e_2, e_3, e_4, e_5$  respectively. If the orientation of a point is unknown, then the point is initialized as the identity matrix, which is the case in this paper. Otherwise, the given orientation such as surface normal vectors is properly encoded in its tensor. After the encoding step, each token (5D point with its associated tensor) casts votes to its neighboring tokens based on predefined voting kernels. Each voting kernel is a tensor field, and it encodes constraints such as smoothness. During the voting process, each input token collects votes from its neighbors by tensor addition, and the resulting tensor  $T$  at the point is decomposed as in:

$$\begin{aligned} T &= \lambda_1 e_1 e_1^T + \lambda_2 e_2 e_2^T + \lambda_3 e_3 e_3^T + \lambda_4 e_4 e_4^T + \lambda_5 e_5 e_5^T \\ &= (\lambda_1 - \lambda_2) e_1 e_1^T + \\ &\quad (\lambda_2 - \lambda_3) (e_1 e_1^T + e_2 e_2^T) + \\ &\quad (\lambda_3 - \lambda_4) (e_1 e_1^T + e_2 e_2^T + e_3 e_3^T) + \\ &\quad (\lambda_4 - \lambda_5) (e_1 e_1^T + e_2 e_2^T + e_3 e_3^T + e_4 e_4^T) + \\ &\quad \lambda_5 (e_1 e_1^T + e_2 e_2^T + e_3 e_3^T + e_4 e_4^T + e_5 e_5^T) \end{aligned}$$

Table 1 shows the relation between geometric features in the 5D space at each dimension and their tensor representations. The *saliency* of each feature is defined as shown in the

Dim	Saliency	Normal vectors	Tangent vectors
0	$\lambda_5$	none	none
1	$\lambda_4 - \lambda_5$	$e_1, e_2, e_3, e_4$	$e_5$
2	$\lambda_3 - \lambda_4$	$e_1, e_2, e_3$	$e_4, e_5$
3	$\lambda_2 - \lambda_3$	$e_1, e_2$	$e_3, e_4, e_5$
4	$\lambda_1 - \lambda_2$	$e_1$	$e_2, e_3, e_4, e_5$

Table 1. Geometric features in 5D space

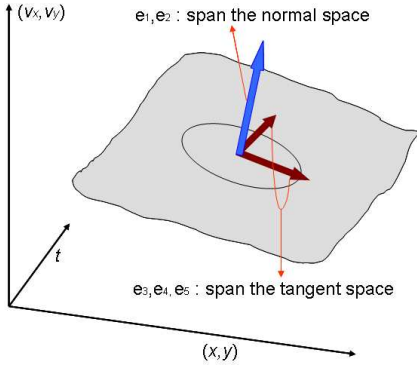


Figure 4. The target motion layer in 5D space

table, and it represents the confidence of each feature. Since the layers we are trying to find are of dimension 3 (i.e., 3D variety), the five eigenvectors of the tensor at each token are interpreted as two normal vectors ( $e_1, e_2$ ), and three tangent vectors ( $e_3, e_4, e_5$ ). This is illustrated in Figure 4. Therefore, the 3D variety saliency,  $\lambda_2 - \lambda_3$ , is maximum for all points in layers.

### 3. Our approach

In this section, we provide the details of our proposed approach, and the overall algorithm is shown in Figure 5.

#### 3.1. Initial color oversegmentation

In the tensor voting framework, each token in the 5D space casts its orientation information to the neighboring tokens, and then smooth layers which correspond to independent moving objects are extracted based on the 3D variety saliency at each token. Basically, a layer is formed due to the affinity between nearby tokens in space and time with consistent velocities. The tokens around motion boundaries, however, may collect votes from tokens which belong to different layers resulting in wrong overall tensors. Therefore, the obtained motion boundaries are not exact, so we may need a post processing step to find the correct ones. In [14], the authors utilize 2D image gradient information around the motion boundaries to correct them. In this paper, we propose to use color oversegmentation before the following tensor voting processes, so that each token collects votes only from the neighboring tokens which belong

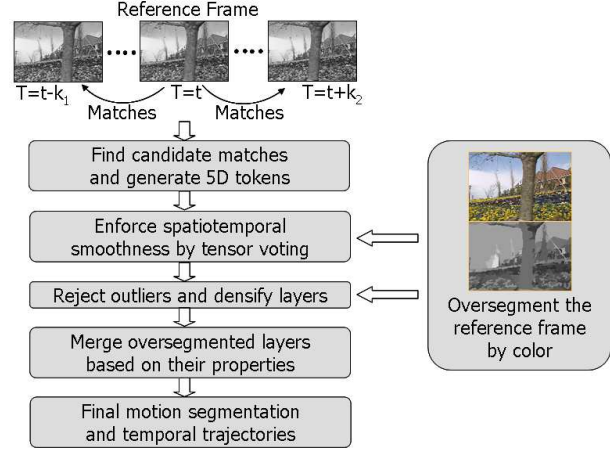


Figure 5. Overall proposed algorithm

to the same segment. By assuming that each segment contains pixels of only a single object, we can not only avoid voting between different layers, but safely assume that each segment satisfies the spatiotemporal smoothness constraint. For our experiments, we use the mean shift based image segmentation (EDISON) of [7] and [1] with color reference images, and it usually generates 100 to 200 segments.

#### 3.2. Initializing tensors and voting

The image sequences are gray scale, and we use intensity based cross-correlation with a 11x11 window to find potential matches between the reference image and other images. For instance, assume that 4 images are given  $(I_0, I_1, I_2, I_3)$ , and the first one,  $I_0$ , is the reference image. From the correlation computation, we obtain the velocities,  $(\mathbf{V}_{01}(p), \mathbf{V}_{02}(p), \mathbf{V}_{03}(p))$ , where  $\mathbf{V}_{ij}(p)$  is the velocity of a pixel  $p$  in the reference image between  $I_i$  and  $I_j$ . Since the appropriate velocity in our approach is the velocity of the pixel  $p$  at each time  $t$ , (i.e.,  $\mathbf{V}_{01}(p), \mathbf{V}_{12}(p), \mathbf{V}_{23}(p)$ ), we need to convert the previous obtained velocities into those forms using the simple equation:  $\mathbf{V}_{t,t+1}(p) = \mathbf{V}_{0,t+1}(p) - \mathbf{V}_{0,t}(p)$ . A sequential search which finds  $\mathbf{V}_{01}(p), \mathbf{V}_{12}(p), \mathbf{V}_{23}(p)$  directly from the given images is not appropriate because any wrong match at  $t$  results in wrong matches for all the following frames after  $t$ .

After finding all correspondences of each pixel in the reference frame across other frames, we generate 5D points of the pixel for each time  $t$  (here, discrete frame numbers). Since the initial matches do not provide any orientation, we encode them as ball tensors. After all points are encoded, each of them propagates its information to the neighborhood, the size of which is defined by  $\sigma$ , through the tensor voting process. To prohibit possible communication between different layers, we restrict votes only within the same image segment.

### 3.3. Outlier rejection and densification

Through the first tensor voting step, each token collects votes from its neighbors within the same segment. As we have mentioned in the previous section, the geometric feature that represents desired smooth layers in the 5D space is the 3D variety, and its saliency can be computed by  $\lambda_2 - \lambda_3$ . If a token is part of a layer, then its neighbors strongly support the token resulting in high 3D variety saliency. On the other hand, if a token is a wrong match point, then it is more likely an isolated point in the 5D space so that it collects weaker votes from its neighbors than tokens in layers. Thus, we can locate those outliers by observing the 3D variety saliency at each token.

After the outlier rejection process, we need to estimate correct positions of the removed points to obtain dense velocity fields. For this densification process, the smoothness constraint is also applied. At each removed position,  $(x, y, t)$ , we setup multiple candidate velocities,  $(v_x, v_y)$ , based on the minimum and maximum velocities of its neighbors within the same segment, and generate corresponding candidate 5D tokens. Each token now collects votes from its existing neighbors, and then we select the one which has the highest 3D variety saliency among the candidates for the  $(x, y, t)$  position.

### 3.4. Merging motion layer segments

Since we started with an oversegmentation of the image, now we need to merge segments of the same object based on similarity of layers (each image segment is associated with its layer in the 5D space). In our approach, we compare all adjacent two layers, and to measure the similarity between them we use the following two properties of layers:

- Average of  $(v_x, v_y)$  components (i.e., average velocity of the pixels) of the 5D tokens around the boundary between two layers.
- Normal space which those tokens span (Figure 4)

Roughly speaking, the first property examines if two layers are connected to each other, and the second property examines if the connection is smooth. When a segment is large, non-flat, and not parallel to  $(x, y, t)$  space, then the average velocity components and normal space of all the tokens in the layer do not correctly characterize the layer. Therefore, we consider only tokens around the boundary where two layers meet as shown in Figure 6(a). For the tokens in each square window, we compute the average velocity components and normal spaces for both layer 1 and layer 2, and then compute the similarity of the two layers based on them. This process is repeated for other square windows, and the final similarity measures between two layers are computed by averaging the errors from each square window. The following equations are used for error computation.

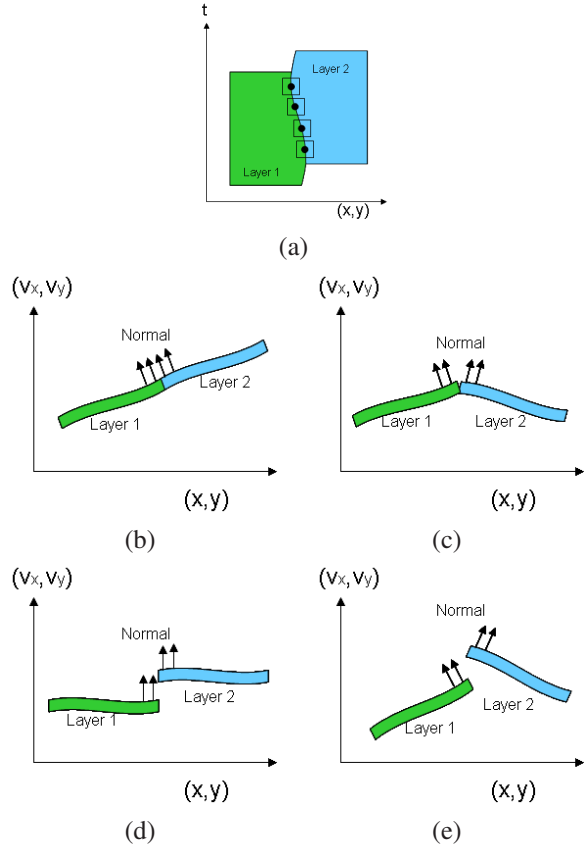


Figure 6. Merging process and various cases: (a) Top view of two adjacent layers.  $(v_x, v_y)$  is orthogonal to this paper, (b) Both velocity and normal spaces are similar - merge, (c) Similar velocity but different normal spaces - do not merge, (d) Different velocity but similar normal spaces - do not merge, (e) Both velocity and normal spaces are different - do not merge

- Average velocity error :  $\frac{1}{N} \sum_n \left| \bar{V}_{layer1}^n - \bar{V}_{layer2}^n \right|$
- Normal space error :  $S_3/S_2$

In the first equation,  $N$  is the total number of square windows, and  $\bar{V}$  is the average velocity component of the tokens in each layer at the given window  $n$ . In the second equation,  $S_i$  are singular values ( $S_1$  is the largest one) of the matrix  $E$  which is the stack of two normal vectors ( $e_1$  and  $e_2$ ) from each token in both layer 1 and layer 2. If two normal spaces are compatible, then the merged layer will have small  $S_3$  (i.e., close to rank 2). Otherwise, the third singular value,  $S_3$ , will be large meaning that the merged layer is close to rank 3.

Note that the two similarity measures, average velocity and normal space, are independent measures, thus, both of them should be satisfied at the same time to merge layers. Figure 6(b)-(e) show various situations. If two layers have similar velocities and normal spaces around their bound-

ary, then they can be merged (shown in (b)). However, if any of two similarity measures fails, then two layers cannot be merged (shown in (c)-(e)). Consequently, all two adjacent layers are tested for merging, and based on those pair-wise results we build the final motion segmentation for the given reference image. Since we are merging layers based on pair-wise comparison, it is possible that layers A and B are merged, and layers A and C are merged, while the similarity measure tells that the layers B and C are different. For this case, we can merge all three layers together without any problems because the layer A works as a bridge which smoothly connects the layer B and C, and they still satisfy the spatiotemporal smoothness constraint.

When we decide if two layers are merged together or not, their similarity measures are compared with certain thresholds. Here, those thresholds are empirically obtained. However, for general sequences, a constant threshold for the average velocity error can be used because we roughly normalize the maximum velocity of objects when we initially generate 5D points. Also the threshold for the normal space error does not significantly affect the merging results so that a constant threshold value obtained empirically can be used as well. In some cases, however, we might need to try several thresholds to get reasonable result. For instance, if the given sequence has many moving objects with difference sizes and speeds, and the quality of the initial matching is poor, then it is hard to distinguish which layers are moving objects and which layers are sets of wrong matches.

#### 4. Experimental results

In this section, we test various sequences and show their segmentation results. The first three sequences, *Blue car*, *Truck*, and *Flower garden*, demonstrate the accuracy of motion segmentation results processed by our proposed method. The next sequence, *Two pedestrians*, explains how motion layers of occluded objects can be recovered by the spatiotemporal smoothness constraint. The last two sequences, *Teddy bear* and *Cones*, show the accuracy of our matching results by comparing them with ground-truth data.

##### Various sequences

First, we show the basic motion segmentation outputs from three different sequences: *Blue car*, *Truck*, and *Flower garden*, which are shown in Figure 8 (a), (d), and (g), respectively. The first two sequences have 5 frames and the last *Flower garden* sequence has 7 frames. The scale factor  $\sigma$  is set to 7 for all three sequences. In the case of *Blue car* and *Truck* sequences, the shadows below the cars move together with the cars, thus, they are included in the ground-truth car regions (Figure 8 (b) and (e)). Figure 8 (c)(f)(i) show our motion segmentation results for each sequence, and the overall error rates are presented in Table 2. If the shadows below the cars are excluded from the ground-truth car re-

Sequences	Error rates
<b>Blue car</b>	0.45%
<b>Truck</b>	0.95%
<b>Flower garden</b>	5.17%

Table 2. Error rates of the segmentation results in Figure 8.

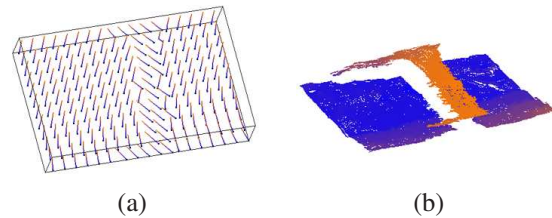


Figure 7. More results of the *Flower garden* sequence. (a) temporal trajectories of the pixels in the reference image, (b) one of the motion layers ( $v_x$  component only).

gions, then the error rates increase to 1.89% and 3.14% for the *Blue car* and *Truck* sequences, respectively. The segmented tree region in (i) does not include the thin branches because they were not correctly segmented in the initial oversegmentation stage. Also (i) shows the case of 'under-segmentation': some of the initial small patches around the tree violate our assumption that each patch includes pixels of only a single object because of the complex textures around the tree. As a result, the tree boundary is not smooth. Figure 7 shows the computed temporal trajectories and one of the motion layers of the *Flower garden* sequence.

##### Two Pedestrians

From this sequence, we present the advantage of temporal smoothness constraint in the presence of serious occlusion. As can be seen in Figure 9(a), there are two people walking in opposite directions, so that the one in the back suffers from serious occlusions in the frame 4-6. Therefore, the matches of him between the reference frame (2<sup>nd</sup> with blue box) and those three frames create significant amount of wrong matches, and they are mostly removed during the outlier rejection process due to their random distribution in the 5D space. (b) shows the result of the outlier rejection for the frame 5. We see from the figure that most of the matches of the person in the back have been removed because he is not visible in the frame 5. On the other hand, he is mostly visible in other frames, and the matches at those frames create smooth layers in the 5D space. During the densification process, those smooth layers which consist of good matches are extended to the positions where wrong matches are removed. This is possible due to the 'temporal smoothness constraint', and the final densification results of frame 5 is represented in (c). Figure 9(d) and (e) show temporal trajectories of the pixels in a single scan line for initial noisy matches and spatiotemporally smoothed matches, respectively. In (d), the pixels of the person in the back lose their

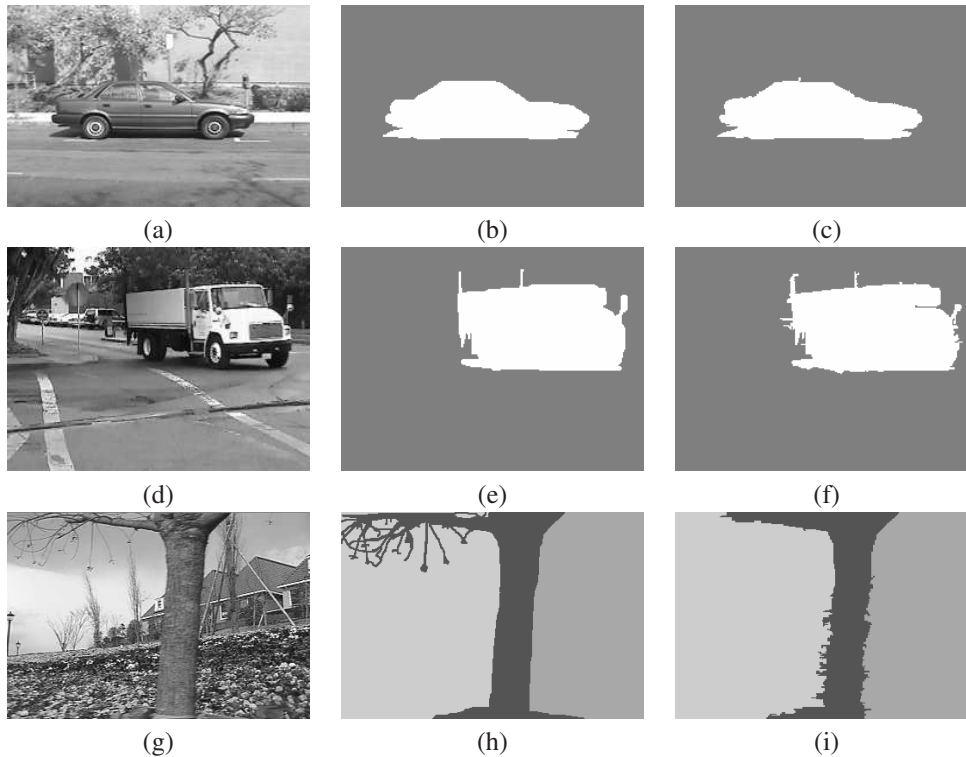


Figure 8. Motion segmentation examples of various sequences. (a)(d)(g): reference images of the input sequences, (b)(e)(h): ground-truth segmentation, (c)(f)(i) our motion segmentation results. The ground-truth car regions in (b) and (e) include shadows.

temporal tracks due to occlusions, but they are recovered by spatiotemporal smoothness constraint as can be seen in (e).

### Middlebury sequences

In order to compare our proposed approach with others in terms of the accuracy of pixel-matching, we tested the *Teddy bear* and *Cones* sequences because they come with their ground-truth disparity maps [2]. Figure 10(a) and (b) show the reference images of the sequences, (c) and (d) show the given ground-truth disparity maps, and (e) and (f) show our computed disparity maps. Each sequence has 4 frames and the scale factor  $\sigma$  is set to 5 for both. Considering all pixels in the images and setting the error threshold to 1, the reported error rates were 11.5% and 8.47% for the *Teddy bear* and *Cones*, respectively (they ranked 4<sup>th</sup> and 1<sup>st</sup> when the tests were run). Unlike other methods which are optimized to handle stereo pairs only, our approach is more generic, in that it can handle all kinds of motion, not just horizontal disparity.

## 5. Conclusion

In this paper, we have presented a novel 5D representation for the motion segmentation task. The most important idea is the smoothing process which enforces both spatial and temporal smoothness simultaneously. Of course, our temporal smoothness assumption may not hold if objects or

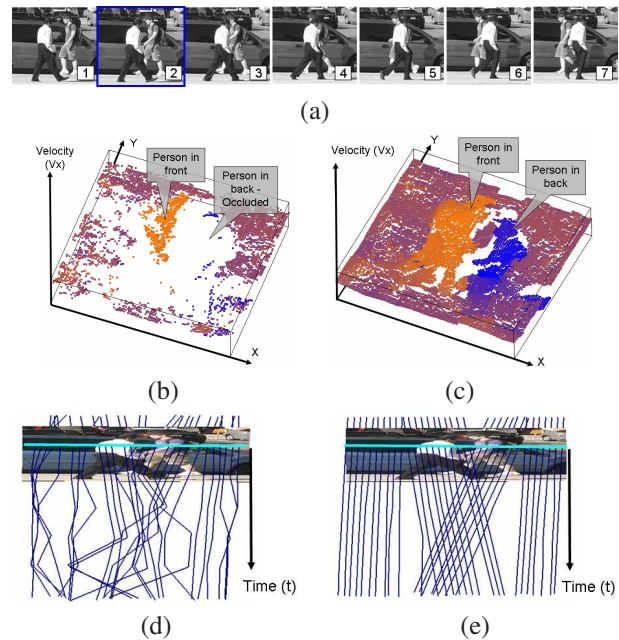


Figure 9. Results of the *Two Pedestrians* sequence. (a) input sequence, (b) motion layer after outlier rejection between frame 2 and 5, (c) after densification, (d) initial noisy temporal trajectories, (e) spatiotemporally smoothed temporal trajectories.

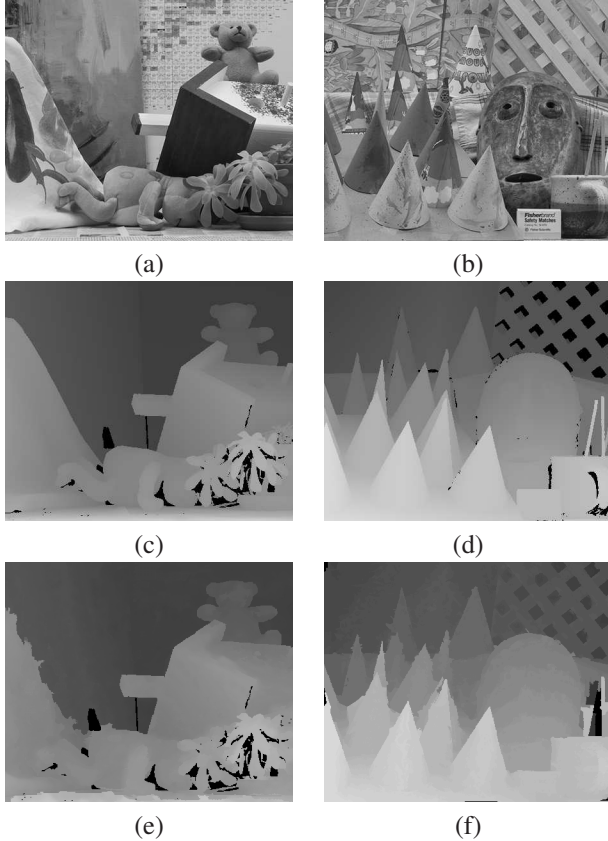


Figure 10. Results of the *Teddy bear* and *Cones* sequences. (a)(b): reference images of the sequences, (c)(d): ground-truth disparity maps, (e)(f): our disparity map results.

the camera move abruptly at some of the images. Such a case is not in our scope, and real video sequences with high enough frame rate usually do not have this problem. In fact, the proposed 5D representation has the same amount of information as the fiber bundle representation. However, by converting the complex spatiotemporal smoothness property in the fiber bundle to simple smooth layers in the 5D space, we could efficiently solve the motion segmentation problem with dense temporal trajectories. Current framework can process only a single reference frame at a time, so that, in near future, we are going to study how to extend our current framework to process multiple reference frames sequentially in an efficient way.

## Acknowledgment

The research has been funded in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152, and U.S. National Science Foundation grant IIS 03 29247. Any opinions, findings and conclusions or recommendations expressed in this material are those of the

authors and do not necessarily reflect those of the National Science Foundation.

## References

- [1] <http://www.caip.rutgers.edu/riul/research/code/EDISON/>. 4
- [2] <http://cat.middlebury.edu/stereo/>. 7
- [3] E. Adelson and Y. Weiss. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *CVPR*, pages 321–326, 1996. 1
- [4] S. Ayer and H. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In *ICCV*, pages 777–784, 1995. 1
- [5] H. Baker and R. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *IJCV*, 3(1. May 1989):33–49, May 1989. 1
- [6] R. Bolles, H. Baker, and D. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1(1):7–56, 1987. 1
- [7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619, May 2002. 4
- [8] J. Costeira and T. Kanade. A multibody factorization method for independently moving-objects. *IJCV*, 29(3):159–179, September 1998. 1
- [9] S. Hsu, P. Anandan, and S. Peleg. Accurate computation of optical flow by using layered motion representations. In *ICPR*, pages A:743–746, 1994. 1
- [10] D. Husemoller. *Fibre Bundles*. Springer-Verlag, 3rd edition, 1993. 2
- [11] K. Kanatani. Motion segmentation by subspace separation and model selection. In *ICCV*, pages II: 586–591, 2001. 1
- [12] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. In *ICCV*, pages II: 508–515, 2001. 1
- [13] G. Medioni, M. Lee, and C. Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier, 1st edition, 2000. 2, 3
- [14] M. Nicolescu and G. Medioni. Layered 4d representation and voting for grouping from motion. *PAMI*, 25(4):492–501, April 2003. 1, 3, 4
- [15] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137–154, November 1992. 1
- [16] R. Vidal and R. Hartley. Motion segmentation with missing data using powerfactorization and gpca. In *CVPR*, pages II: 310–316, 2004. 1
- [17] J. Wang and E. Adelson. Layered representation for motion analysis. In *CVPR*, pages 361–366, 1993. 1
- [18] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cuts. *PAMI*, 27(10):1644–1659, October 2005. 1
- [19] L. Zelnik-Manor and M. Irani. Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. In *CVPR*, pages II: 287–293, 2003. 1