

Two-Frames Accurate Motion Segmentation Using Tensor Voting and Graph-Cuts

Thang Dinh* and Gérard Medioni
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089, USA
{thangdin, medioni}@usc.edu

Abstract

Motion segmentation and motion estimation are important topics in computer vision. Tensor Voting is a process that addresses both issues simultaneously; but running time is a challenge. We propose a novel approach which can yield both the motion segmentation and the motion estimation in the presence of discontinuities. This method is a combination of a non-iterative boosted-speed voting process in sparse space in a first stage, and a Graph-Cuts framework for boundary refinement in a second stage. Here, we concentrate on the motion segmentation problem. After initially choosing a sparse space by sampling the original image, we represent each of these pixels as 4-D tensor points and apply the voting framework to enforce local smoothness of motion. Afterwards, the boundary refinement is obtained by using the Graph-Cuts image segmentation. Our results attained in different types of motion show that the method outperforms other Tensor Voting approaches in speed, and the results are comparable with other methodologies in motion segmentation.

1. Introduction

Motion estimation and motion segmentation are complementary processes: motion segmentation infers knowledge of boundaries, as discontinuities in the dense flow generated by an accurate motion estimation scheme; and accurate motion estimation requires the knowledge of boundaries, as generated by a motion segmentation scheme. There are many proposed approaches to solve these two problems.

Optical flow techniques [11] rely on local estimations of the flow field to provide the segmented regions. However, the flow estimations often lack accuracy at motion boundaries. To obtain the motion estimation, several

state-of-the-art methods have been proposed in [3][4][9][14][23]. Simultaneously, in motion segmentation, many approaches have been invoked [12][17][25]. Some approaches are based on Markov Random Field and EM algorithm [2][28]. More recently, there are approaches using Level Set Methods [8][21][22] and Graph-Cuts [16][20][24].

Tensor Voting, proposed by Medioni et al. [10], is an approach that successfully enforces the smoothness constraint in a consistent and unified way while preserving discontinuities. This method is a non-iterative technique and does not depend on any strict threshold. However, it is only based on pixel velocities, which are not reliable, especially at the motion boundaries, to obtain the segmentation. Nicolescu and Medioni [15] have successfully overcome these issues with the 4-D Tensor Voting framework for matching, densification, and segmentation by applying another 2-D voting process for boundary refinement. Even though this method is usually accurate in most cases, it fails if the regions along the boundaries have sophisticated textures with many edges to be chosen as boundaries. Min and Medioni [6] presented a 5-D Tensor Voting framework to utilize the rich temporal information of motion; though, this approach uses naive color oversegmentation at initialization without any prior knowledge, which easily leads to irrelevant final results.

Another drawback of Tensor Voting is that it suffers from a lengthy processing time. The 4-D voting framework of Nicolescu and Medioni often takes over 3 hours to process a pair of images with a resolution of 192x200 for only the matching step in their approach. Even when using GPU implementation [6], the 5-D framework also takes about an hour to finish the whole task with a sequence of 5 frames. Moreover, Tensor Voting only uses motion information and misses other rich information such as color, and intensity.

To address the above issues, we propose a novel approach for segmentation from two images, a two-stage framework using the 4-D voting process in sparse space, combined with Graph-Cuts for refining the motion boundaries.

* Vietnam Education Foundation Fellow

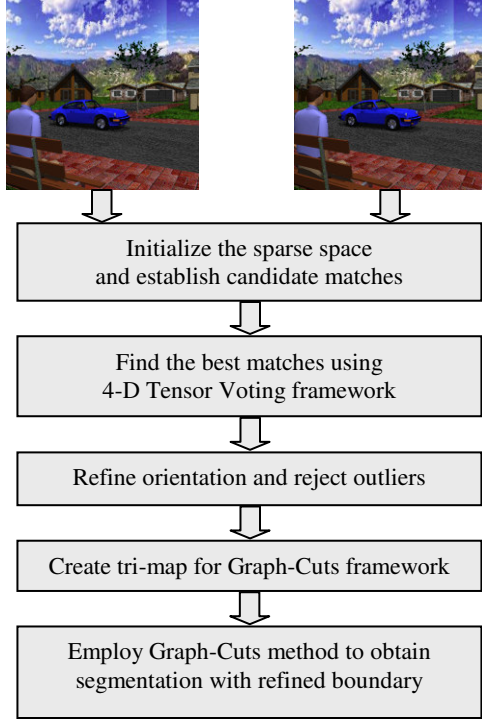


Figure 1. Outline for our proposed approach.

The rest of the paper is organized as follows: in section 2 is the overview of our method; a brief introduction about the 4-D voting process is in section 3; in section 4, we discuss in details how to employ the 4-D Tensor Voting framework in the matches selection, orientation refinement and outlier rejection; tri-map establishment is described in section 5, followed by the description of the Graph-Cuts framework in section 6; section 7 shows experimental results together with the comparison between our approach and others; and our conclusion and future directions are outlined in section 8.

2. Overview of our method

In this section, we describe the outline of our proposed approach, as shown in Figure 1.

Initialization. To overcome the lengthy processing time, a major drawback of Tensor Voting, we set up a sparse space for voting process to take place. It reduces the running time exponentially based on the chosen sparseness, which is examined carefully in section 4. For each square region, we select the center pixel as the nominee. After obtaining the collection of all these pixels, we establish candidate matches using multi-scale normalized cross-correlation windows.

Tensor Voting stage. Each of these candidates is represented as a (x, y, v_x, v_y) point in 4-D space where x and y are the pixel coordinates in the first image, and v_x and v_y are the velocities obtained after candidates matching. We

then enforce the smoothness of motion using 4-D Tensor Voting framework to extract motion layers.

Tri-map establishment. In order to run the Graph-Cuts efficiently, a tri-map is organized in which we indicate the seeds for foreground, background, and undefined regions. From the output layers of the Tensor Voting stage, we determine the moving regions, get the boundaries, then acquire the inbound and outbound, and clarify boundary regions with a defined thickness.

Graph-Cuts stage. The color/intensity information is essentially mocular, thus independent of the motion information generated by the Tensor Voting process. The combination of both sources of information, as proposed here, significantly improves the results.

Note that the Graph-Cuts stage is essentially binary (in/out) and must be repeated for any pair of adjacent regions, which can then be merged.

3. 4-D Tensor Voting framework

The Tensor Voting framework can be extended to any dimension with some implementation changes for efficiency [5]. However, in each voting space, we have to define the tensor representations of the features, the voting fields, and the data structures for vote collection.

Tensor representation. Extended from the 2-D voting process proposed by Medioni et al. in [15], in this framework, points, curves, surfaces and volumes are the four geometric features which should be extracted. The features and their representation as elementary 4-D tensors are presented in Table 1 by giving the values of eigenvectors and eigenvalues. The mentioned n and t are the normal and tangent vectors. Geometric features extracted after the voting process from generic 4-D tensors are shown in Table 2.

Table 1. 4-D Elementary Tensors.

Feature	$\lambda_1 \lambda_2 \lambda_3 \lambda_4$	$e_1 e_2 e_3 e_4$	Tensor
Point	1 1 1 1	Any orthonormal basis	Ball
Curve	1 1 1 0	$n_1 n_2 n_3 t$	C-Plate
Surface	1 1 0 0	$n_1 n_2 t_1 t_2$	S-Plate
Volume	1 0 0 0	$n_1 t_1 t_2 t_3$	Stick

Table 2. A 4-D Generic Tensor.

Feature	Saliency	Normals	Tangents
Point	λ_4	none	none
Curve	$\lambda_3 - \lambda_4$	$e_1 e_2 e_3$	e_4
Surface	$\lambda_2 - \lambda_3$	$e_1 e_2$	$e_3 e_4$
Volume	$\lambda_1 - \lambda_2$	e_1	$e_2 e_3 e_4$

Voting fields. Determining the voting fields is the most important part when applying the Tensor Voting framework because they encode all characteristics i.e. the

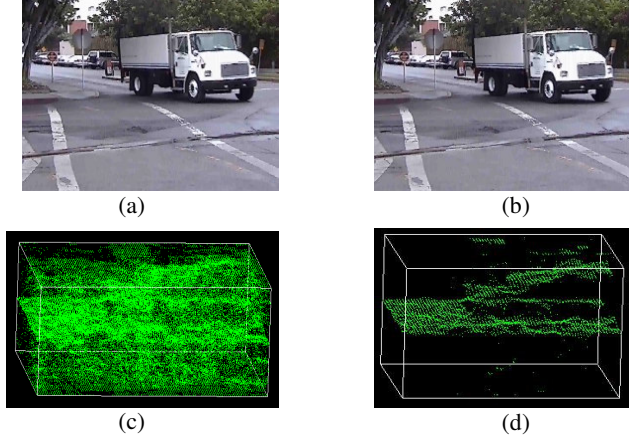


Figure 2. (a),(b) are the two input frames of truck sequence; (c) 3-D view of the candidate matches generated after cross-correlation (v_x component only); (d) 3-D view of the selected matches (v_x component only).

size, the shape of the neighborhood where the votes are cast. The 4-D stick is the fundamental field, using to generate all other voting fields. This construction is done by getting the integration of all the results gained from rotating a 4-D stick field around applicable axes. In this framework, we use only the 4-D ball field which is generated as following:

$$V_{ball}(\vec{d}) = \iiint_0^{2\pi} RV_{stick}(R^{-1}\vec{d})R^T d\theta_{xy}\theta_{xu}\theta_{xv} \quad (1)$$

Where x,y,u,v are the 4-D coordinates axes, θ_{xy} , θ_{xu} , θ_{xv} are rotation angles in each plane.

Data structure. We continue to use approximate nearest neighbor (ANN) k - d tree [19] to store the tensors as Niculescu and Medioni employed in [15]. This data structure is very efficient, since the space complexity of the algorithm equals to $O(n)$, where n is the input size. The voting process has an average time complexity $O(\mu n)$ where μ is the average number of tokens in the neighborhood. Hence, Tensor Voting is independent from the dimensionality of the feature.

Although the optimization in selecting data structure is significant, Tensor Voting framework [15] is still time consuming. Therefore, the sparse space proposed in the next section is necessary.

4. Moving regions segmentation

4.1. Generating potential matches in sparse space

In our experiments, we take as input a pair of images in a sequence, in which both the objects and the camera can be moving. In order to find the candidate matches between these two images, we use a simple normalized cross-

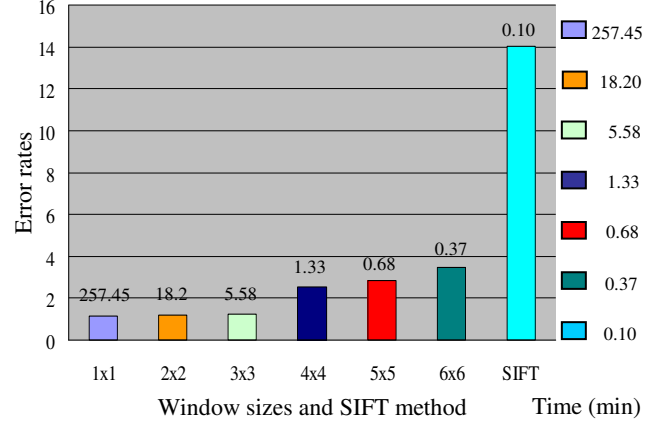


Figure 3. Time vs. error rates; different colors for different window sizes and the SIFT descriptors method; the caption and the right column descriptions indicate the processing time of each method; more details are described in Table 3.

correlation process to produce the matches based on the peaks of correlation. From the correlation, we can also obtain the velocity (v_x , v_y) between the two images. However, unlike the approach proposed by Niculescu and Medioni [15], we only collect a representative for each region in the image. We have investigated several approaches of this idea such as: SIFT descriptors, and hierarchy of sampling windows. However, SIFT produces a too sparse space which does not contain enough information for Tensor Voting to succeed, as described in Figure 3 and Table 3; whereas the hierarchy of sampling windows method yields good results. Nevertheless, in this approach, the lowest layer, the smallest window in the pyramid, can cover all of the contributions gained from others. Therefore, we apply the sampling method only one time by collecting the center of each square region as the candidate.

The larger the size of the window, the faster our process runs, but the less information can be utilized, which leads to poor results. After performing the experiment with many different sizes of windows shown in Figure 3, we found that the best results are obtained by using 3x3 windows, which produce good segmentations with an acceptable speed.

Table 3. Time and error rates when applying different window sizes and the SIFT descriptors.

Window Sizes/SIFT	Time (min.)	Error rates (%)
1x1	257.45	1.16
2x2	18.20	1.19
3x3	5.58	1.26
4x4	1.33	2.53
5x5	0.68	2.85
6x6	0.37	3.46
SIFT	0.10	14.01



Figure 4. The raw segmentation by Tensor Voting in truck sequence; green dots are the obtained result after outlier refinement step.

According to the work in [15], taking advantage of the voting framework’s robustness regardless of noise, we use different sizes of the correlation windows to repeat this process at multiple scales. This effort helps increase the probability of having the correct match among the candidates. All of the experiments are also carried out using three correlation window sizes: 3x3, 5x5, and 7x7.

After finding all potential candidates, we represent each of them as a (x, y, v_x, v_y) point in 4-D space, illustrated in Figure 2(c).

4.2. Voting process

Matches selection. Since there is no provided information about the orientation, we initially encode all of the candidate matches as 4-D ball tensors. The voting process is then applied to the 4-D ball voting where each token propagates its information to the neighborhood with a pre-defined size σ ($\sigma=10$ in our experiments). After that, the candidate with highest surface saliency is considered as the best match while others are rejected as outliers.

Orientation refinement. This is another voting process applied only on tensors selected by the previous step. After this phase, we obtain a more accurate estimation of the local layer orientations.

Outlier rejection. Although we can get all the best matches after the first step, there may still be some pixels with incorrect assigned velocity, such as those in low texture areas. If a token is a part of a layer, it receives strong support from its neighbors, which is in the surface saliency (λ_2 - λ_3). In other words, if expressed any incorrect match, it should be isolated in the 4-D space. Hence, we can easily reject all of those with a surface saliency of less than 10% of the average saliency of the entire set proposed in [15].

5. Tri-map establishment

After the Tensor Voting process, the boundaries of the

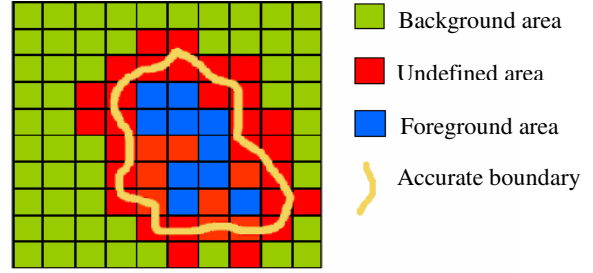


Figure 5. An illustration of Tri-map including background, foreground, and undefined areas.

moving regions are not accurate, as shown in Figure 4. It happens because what has been used until now is the voting framework in sparse space, and the pixel velocities information around motion boundaries is not reliable. Graph-Cuts are employed in order to delineate the accurate boundaries based on the color information of the images. However, we need to prepare the graph with the result obtained from the previous Tensor Voting step.

The Graph-Cuts framework is initialized with the seeds for the foreground and the background; while all others are considered as undefined pixels. We create a tri-map with label 0 for background, 1 for foreground and 2 for undefined pixels, as illustrated in Figure 5.

From the sparse set of points which are divided into smooth regions based on their motions, we can estimate the current boundaries provided by the voting process. Consequently, we assume that the boundaries have thickness; thus, they need to be refined, and we use the Graph-Cuts framework. All of the inside and outside pixels are foreground and background seeds, respectively. In order to reduce noise, which may occur after previous steps, we also put a threshold θ ($\theta=8$ in our experiments) on how large a moving region should be. It helps to avoid assigning wrong seeds. Note that the unit used to calculate the threshold here, is not in pixels but in points which are in sparse space.

Moreover, because the Graph-Cuts framework produces a binary segmentation, we process one pair of adjacent regions at a time. Once all pairs are processed, we can merge the intermediate results.

6. Graph-Cuts framework

We apply the framework as proposed in the Lazy Snapping Cut-out tool by Li et al. [27]. Our segmentation problem can now be considered as a binary labeling problem. The image is presented as a graph $G = \{V, E\}$ where V is the set of all nodes and E is the set of all arcs. We try to label every node in the graph as x_i where $x_i = 0$ if node i is on background and $x_i = 1$ if it is on foreground. This labeling problem can be solved by

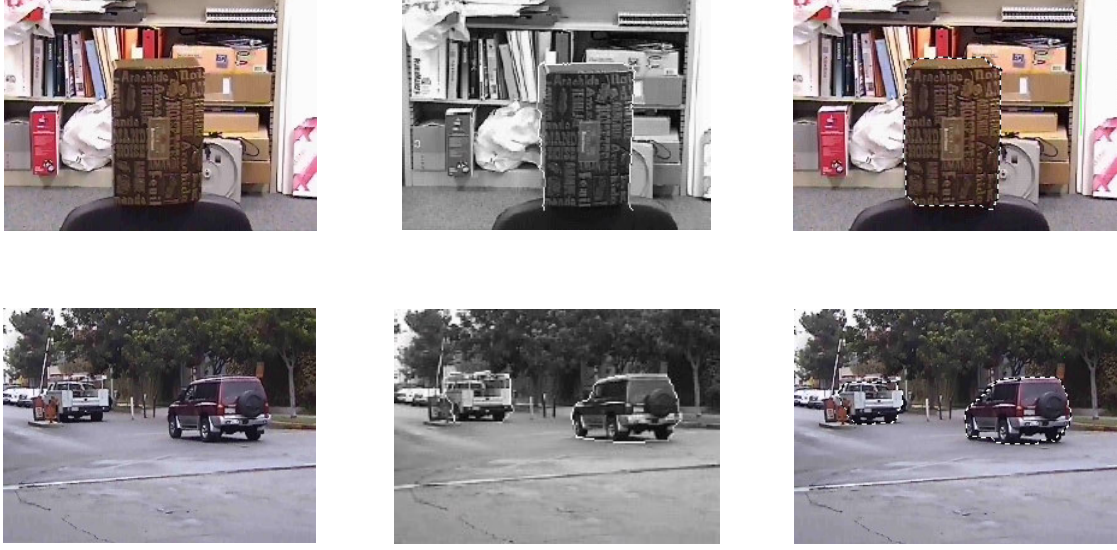


Figure 6. Results of the bombon (top) and barrier (bottom) sequence; the first column images is the input; the second column is the results obtained by Nicolescu and Medioni[15]; the last column is our results.

minimizing the Gibbs energy $E(X)$:

$$E(X) = \sum_{i \in V} E_1(x_i) + \lambda \sum_{(i,j) \in E} E_2(x_i, x_j) \quad (2)$$

Where $E_1(x_i)$ is likelihood energy denoting the cost when labeling node i as x_i and $E_2(x_i, x_j)$ is the prior energy denoting the cost when two adjacent nodes i and j have labels x_i and x_j , respectively.

Here, applying Li et al. method [27], we also adopt the formulation of energy minimization as a graph cut problem proposed by Boykov and Jolly [26]. This algorithm has been successfully applied in many approaches in graphics [7][27] and in computer vision [13][24].

Likelihood energy. In equation (2), the energy term E_1 denotes the color similarity of a node when it is foreground or background. We cluster the seeds of two sets Foreground (F) and Background (B), which have been classified by Tri-map, using the K-means method of Duda et al.[18]. In our experiments, we choose the number of clusters as 64 for foreground set and 196 for background set. This is because the moving regions are usually much smaller than the background.

Denote U as undefined region, K_n^F , K_m^B as the mean colors of the foreground and background clusters respectively, C_i is the color of node i , we compute E_1 as follow:

$$\begin{cases} E_1(x_i = 1) = 0 & E_1(x_i = 0) = \infty & \forall i \in F \\ E_1(x_i = 1) = \infty & E_1(x_i = 0) = 0 & \forall i \in B \\ E_1(x_i = 1) = \frac{d_i^F}{d_i^F + d_i^B} & E_1(x_i = 0) = \frac{d_i^B}{d_i^F + d_i^B} & \forall i \in U \end{cases} \quad (3)$$

Where $d_i^F = \min_n \|C_i - K_n^F\|$ is the minimum distance from the color of node i to Foreground clusters and $d_i^B = \min_m \|C_i - K_m^B\|$ is the minimum distance from the color of that node i to Background clusters.

The first two equations guarantee that the label of all nodes in F and B provided in the tri-map are consistent, while the third one encourages the label of undefined node to be the same as the nodes having similar colors in Foreground or Background.

Prior energy. E_2 is defined as a function of the color gradient between two connecting nodes i and j :

$$E_2(x_i, x_j) = |x_i - x_j| \cdot \frac{1}{C_{ij}} \quad (4)$$

Where $C_{ij} = \|C_i - C_j\|^2$ is the $L2$ -Norm of the difference in RGB color space between two pixels i and j . So E_2 will represent the gradient information along the boundary of the object. The more similarity in colors of two nodes i and j , the larger E_2 is; thus, the less likely the edge connecting i and j is to be on the boundary.

7. Experimental results

We have evaluated various sequences and we provide the results here. In all of our experiments, we extracted the center pixel of every 3x3 square region to set up the sparse space for voting process. In the finding candidates step, we used 3 different size correlation windows which are 3x3, 5x5, 7x7. During the voting process, we used the scale factor $\sigma=10$ for every step. When creating a tri-map to

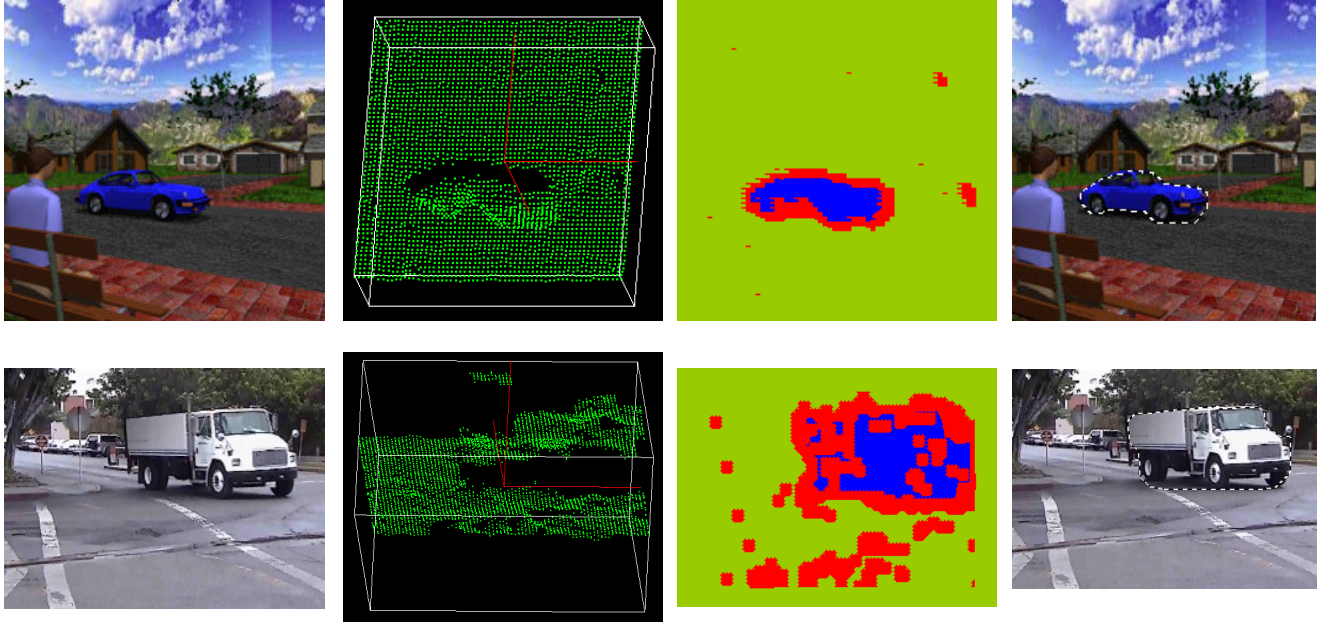


Figure 7. Results of our method in the “street with car” (top) and the truck (bottom) sequence; the first column is one frame of the input; the second column is the 3D view of component v_x after matches selection; the established tri-map is in column 3; column 4 is the final segmented result with accurate boundary.

initialize Graph-Cuts framework, we investigated the region around the boundary within the thickness 10. All of the experiments were tested on a Pentium IV 3.20Ghz with 2GB of memory.

Barrier sequence (Figure 6). This sequence is a real input data which has two vehicles moving away from the camera. It is challenging because of the low texture ground and the relatively small size of the objects compared to the size of the image. Note that the white vehicle is occluded by the security bar as well and the velocities of the two vehicles are totally different. However, Tensor Voting only requires the smoothness of motion without any assumption about the type of motion such as translational, planar, or rigid one.

Bombon sequence (Figure 6). This is a moving camera sequence. However, the chair where the box put on should be considered as static object with a little moving action observed from the frames. Hence, the box is the only moving object which needs to be segmented from the image.

Street with car sequence (Figure 7). The moving car is very small relative to the size of the image. Although the color of the car is quite distinct from the color of the background, there are background areas which can be seen through the car glass which could make it difficult for Graph-Cuts to obtain the boundary. However, Tensor Voting has covered it by filling almost every point inside, which strongly helps Graph-Cuts to refine the boundary.

Truck sequence (Figure 7). In this sequence, we adopt frame 7 and frame 9 as an input of our framework. The truck in the sequence is moving close to the camera. Because of the mirror and the exhaust pipe of the truck are too small, our framework could not obtain it while matching. A part of the shadow under the truck on the left is nearly merged into the ground; thus, our method also trims it while processing.

To quantify the efficiency of our work, we calculate the error rates by computing the percentage of misplaced foreground and background pixels, as shown in Table 4, where FE, BE and TT are the foreground errors, the background errors and the errors in total, respectively. Because this is real data, we generate the ground-truth by using the magnetic lasso tool of Adobe Photoshop 9.0.

Moreover, we include the comparison of our method with the work of Nicolescu and Medioni [15] about accuracy in Table 4 and processing time in Table 5. Our approach can surpass their proposed framework in accuracy within approximate 30 times faster.



Figure 8. The result of car sequence; (a) and (b) are the two input frames; (c) is our result; (d) is the result obtained by MMSWC method in [1].

Table 4. Accuracy comparison table in several sequences.

Sequence	Our method			4-D Tensor Voting		
	FE	BE	TT	FE	BE	TT
Bombon	0.32	0.40	0.72	0.39	0.53	0.91
Street w. car	0.51	0.18	0.69	0.43	1.05	1.49
Truck	0.83	0.43	1.26	0.26	2.48	2.73
Barrier	0.64	0.42	1.06	0.65	0.45	1.10

We also perform our algorithm on the sequence “Car” evaluated in [1] with the significant result shown in Figure 8. In this case, our boundary is more accurate except the misplaced small background region at the bottom right.

Table 5. Processing time comparison table in several sequences.

Sequence	Our method (min.)	4-D Tensor Voting (min.)
Bombon	7.23	242.17
Street w car	8.38	231.25
Truck	5.58	269.27
Barrier	9.32	257.43

8. Conclusion

We have presented a novel approach for the segmentation of moving objects based on Tensor Voting framework and Graph-Cuts. In our method, Tensor Voting enforces the smoothness of motion for matching, orientation refinement, and outlier rejection first; later, a tri-map is developed from the result obtained from Tensor Voting. Graph-Cuts is the final step to attain the accurate boundaries. Experimental results and comparisons show that this is an efficient method in both running time and precision. However, our method fails in low texture background sequence because of wrong candidate

matches. Hence, we intend extend our framework into multiple frames effective way with GPU implementation in order to strengthen this framework with temporal information in an acceptable running time. We will also try new candidate matches generation methods to overcome the issue of wrong matches addressed above.

Acknowledge

This work was supported by grants from MURI-ARO W911NF-06-1-0094.

This work was funded by a grant from the Vietnam Education Foundation (VEF). The opinions, findings, and conclusions stated herein are those of the authors and do not necessarily reflect those of VEF.

References

- [1] A. Barbu and S.C. Zhu. Multigrid and Multi-level Swendsen-Wang Cuts for Hierarchic Graph Partition. CVPR 2004.
- [2] A. Jepson and M. J. Black. Mixture Models for Optic Flow Computation. CVPR 1993.
- [3] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with An Application To Stereo Vision. IJCAI 1981. pp 674–679.
- [4] B. K. P. Horn and B. G. Schunck. Determining Optical Flow. AI, 17:185–203, 1981.
- [5] C. K. Tang, G. Medioni, and M. S. Lee. Epipolar Geometry Estimation by Tensor Voting in 8D. ICCV 1999.
- [6] C. Min and G. Medioni. Motion Segmentation by Spatiotemporal Smoothness Using 5D Tensor Voting. POCV 2006.
- [7] C. Rother, A. Blake, and V. Kolmogorov. “Grabcut” - Interactive Foreground Extraction Using Iterated Graph Cuts. SIGGRAPH, 23(3):309–314, 2004.
- [8] D. Cremers and S. Soatto. Motion Competition: A Variational Framework for Piecewise Parametric Motion Segmentation. IJCV, 62(3):249–265, 2005.
- [9] E. Mémin and P. Pérez. A Multigrid Approach for Hierarchical Motion Estimation. ICCV 1998.
- [10] G. Medioni, M. Lee, and C. Tang. A Computational Framework for Segmentation and Grouping. 1st edition. Elsevier, 2000.
- [11] J. Barron, D. Fleet, S. Beauchemin. Performance of Optical Flow Techniques. IJCV, 12(1):43–77, 1994.
- [12] J. Wang and E. Adelson. Representating Moving Images with Layers. TIP, 3(5):625–638, 1994.
- [13] J. Xiao and M. Shah. Motion Layer Extraction in The Presence of Occlusion Using Graph Cut. CVPR 2004.
- [14] L. Alvarez, J. Weickert, and J. Sánchez. Reliable Estimation of Dense Optical Flow Fields with Large Displacements. IJCV, 39(1):41–56, 2000.
- [15] M. Nicolescu and G. Medioni. Motion Segmentation with Accurate Boundaries – A Tensor Voting Approach. CVPR 2003.
- [16] O. Juan. On Some Extensions of Level Sets and Graph Cuts & Their Applications to Image and Video Segmentation. PhD thesis, École Nationale des Ponts et Chaussées, 2006.

- [17] P. Bouthemy and E. François. Motion Segmentation and Qualitative Dynamic Scene Analysis from An Image Sequence. *IJCV*, 10(2):157–182, 1993.
- [18] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification* (2nd Edition). Wiley Press, 2000.
- [19] S. Arya, D. Mount, N. Netanyahu, R Silverman, and A. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *JACM*, 45(6):891–923, 1998.
- [20] S. Birchfield and C. Tomasi. Multiway Cut for Stereo and Motion with Slanted Surfaces. *ICCV* 1999.
- [21] T. Amiaz and N. Kiryati. Piecewise-Smooth Dense Optical Flow via Level Sets. *IJCV*, 68(2):111–124, 2006.
- [22] T. Brox, A. Bruhn, and J. Weickert. Variational Motion Segmentation with Level Sets. *ECCV* 2006.
- [23] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High Accuracy Optical Flow Estimation Based on A Theory For Warping. *ECCV* 2004.
- [24] T. Schoenemann and D. Cremers. Near Real-time Motion Segmentation Using Graph Cuts. *DAGM* 2006.
- [25] W. B. Thompson. Combining Motion and Contrast for Segmentation. *PAMI*, 2(6):543–549, 1980.
- [26] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *PAMI*, 23(11):1222–1239, 2001.
- [27] Y. Li, J. Sun, C. K. Tang, and H. Y. Shum. Lazy Snapping. *SIGGRAPH*, 23(3):303–308, 2004.
- [28] Y. Weiss. Smoothness in Layers: Motion Segmentation Using Nonparametric Mixture Estimation. *CVPR* 1997.