Proceedings of the International Conference on Pattern Recognition,
vol. III, pages 303-308, Quebec City, Canada, August 2002.

4-D Voting for Matching, Densification and Segmentation into Motion Layers

Mircea Nicolescu and Gérard Medioni
Integrated Media Systems Center
University of Southern California
Los Angeles, CA 90089-0273
{mnicoles, medioni}@iris.usc.edu

Abstract*

We present a novel approach for grouping from motion, based on a 4-D Tensor Voting computational framework. From sparse point tokens in two frames we recover the dense velocity field, motion boundaries and regions, in a non-iterative process that does not involve initialization or search in a parametric space, and therefore does not suffer from local optima or poor convergence problems. We encode the image position and potential velocity for each token into a 4-D tensor. A voting process then enforces the smoothness of motion while preserving motion discontinuities, selecting the correct velocity for each input point, as the most salient token. By performing an additional dense voting step we infer velocities at every pixel location, which are then used to determine motion boundaries and regions. We demonstrate our contribution with synthetic and real images, by analyzing several difficult cases – opaque and transparent motion, rigid and non-rigid motion.

1 Introduction

Given two or more image frames, the goal of the problem of grouping from motion is to determine three types of information – a *dense velocity field*, *motion boundaries*, and *regions*. From a computational point of view, the analysis can be decomposed in three processes – *matching*, *densification* and *segmentation*. The *matching* process identifies the elements (tokens) in successive views that represent the same physical entity, thus producing a possibly sparse velocity field. The *densification* process infers velocity vectors at every image location, and the *segmentation* process groups tokens into regions separated by motion boundaries.

In this work we focus on the problem of motion analysis from sparse sets of point tokens in two frames. Two examples of such input are shown in Fig. 1 and Fig. 2. If the frames in each pair are presented in a properly timed succession, a certain motion of image regions is perceived from one frame to the other. However, while in one case the regions can be detected even without motion, only from monocular cues (here, different densities of points), in the other case no monocular information is available. This example shows that analysis is possible even from motion cues *only*. Another interesting aspect is the fact that the human vision system not only establishes point correspondences, but also perceives *regions* in motion, although the input consists of sparse points only.

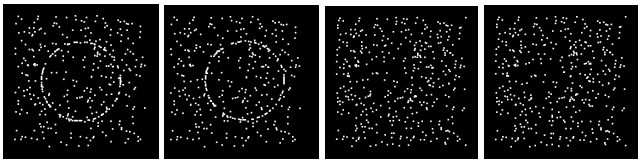


Fig. 1. Translating circle Fig. 2. Translating disk

Optical flow techniques [1] rely on local, raw estimates of the flow field to produce a partition of the image. However, the flow estimates are very poor at motion boundaries and cannot be obtained in uniform areas. Past approaches have also used Markov Random Fields [2] or regularization techniques to handle discontinuities [3].

Significant improvements have been achieved by using layered representations [4][5]. The difficulties range from a severe restriction in motion representation (as rigid or planar), to over-fitting and instability due to high-order parameterizations.

A computational framework that successfully enforces the smoothness constraint in a unified manner, while preserving smoothness discontinuities is Tensor Voting [6]. The first to propose using Tensor Voting for motion analysis were Gaucher and Medioni [7]. They employ

* This 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 by National Science Foundation Grant 9811883.

successive steps of voting, first to determine the boundary points as tokens with maximal motion uncertainty, then to locally refine velocities on each side of the boundary. However, their voting communication is essentially a 2-D process that does not inhibit neighboring elements with different velocities from influencing each other.

In this paper we propose a novel approach based on a *layered 4-D representation* of data, and a *voting scheme* for token communication. Our methodology is formulated as a 4-D Tensor Voting computational framework.

In the next section we give an overview of our method. In Section 3 we present the voting framework and we discuss how the voting concepts are generalized and extended to the 4-D case. In Sections 4, 5 and 6 we describe our approach for matching, densification and segmentation. Section 7 presents our experimental results, while Section 8 summarizes our contribution.

2 Overview of the method

In any method that seeks to solve the motion analysis problem, each token is characterized by four attributes – its image coordinates (x,y) and its velocity with the components (v_x,v_y). We encapsulate them into a (x,y,v_x,v_y) tuple in the 4-D space, this being a natural way of expressing the spatial separation of tokens according to *both* velocities and image coordinates. In general, there may be several candidate velocities for each point (x,y), so each tuple (x,y,v_x,v_y) represents a potential match.

Both matching and densification are based on a process of communicating the affinity between tokens. In our representation, this affinity is expressed as the token preference for being incorporated into a *smooth surface layer* in the 4-D space. A necessary condition is to enforce strong support between tokens in the same layer, and weak support across layers, or at isolated tokens.

In our Tensor Voting framework, the affinities between tokens are embedded in the concept of surface saliency exhibited by the data. By letting the tokens propagate their information through voting, wrong matches are eliminated as they receive little support, and distinct moving regions are extracted as salient smooth layers.

3 Tensor voting

3.1 Overview

The use of a voting process for feature inference from sparse and noisy data was formalized into a unified tensor framework by Medioni, Lee and Tang [6]. The input data is encoded as tensors, then support information (including proximity and smoothness of continuity) is propagated by voting. The only free parameter is the scale of analysis, which is indeed an inherent property of visual perception.

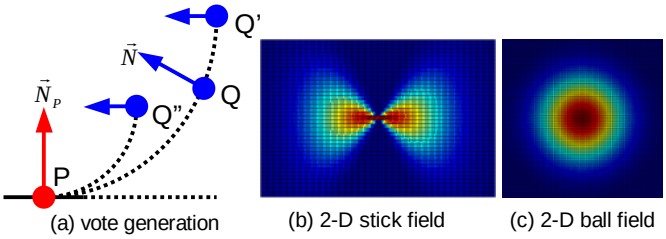


Fig. 3. Voting in 2-D

In the 2-D case, the salient features to be extracted are points and curves. Each token is encoded as a second order symmetric 2-D tensor, geometrically equivalent to an ellipse. It is described by a 2×2 eigensystem, where eigenvectors e_1 and e_2 give the ellipse orientation and eigenvalues λ_1 and λ_2 are the ellipse size. The tensor is represented as a matrix $S = \lambda_1 \cdot e_1 e_1^T + \lambda_2 \cdot e_2 e_2^T$.

An input token that represents a curve element is encoded as a *stick tensor*, where e_2 represents the curve tangent and e_1 the curve normal, while λ_1=1 and λ_2=0. An input point element is encoded as a *ball tensor*, with no preferred orientation, while λ_1=1 and λ_2=1.

The communication between tokens is performed through a voting process, where each token casts a vote at each site in its neighborhood. The size and shape of this neighborhood, and the vote strength and orientation are encapsulated in predefined voting fields (kernels), one for each feature type – there is a stick voting field and a ball voting field in the 2-D case. The fields are generated based only on the scale factor σ. Vote orientation corresponds to the smoothest local curve continuation from voter to recipient, while vote strength $VS(\vec{d})$ decays with distance $|\vec{d}|$ between them, and with curvature ρ:

$$VS(\vec{d}) = e^{-\left(\frac{|\vec{d}|^2+\rho^2}{\sigma^2}\right)} \qquad (1)$$

Fig. 3(a) shows how votes are generated to build the 2-D stick field. A tensor P where curve information is locally known (illustrated by curve normal $\vec{N}_P$) casts a vote at its neighbor Q. The vote orientation is chosen so that it ensures a smooth curve continuation through a circular arc from voter P to recipient Q. To propagate the curve normal $\vec{N}$ thus obtained, the vote $V_{stick}(\vec{d})$ sent from P to Q is encoded as a tensor according to:

$$V_{stick}(\vec{d}) = VS(\vec{d}) \cdot \vec{N}\vec{N}^T \qquad (2)$$

Fig. 3(b) shows the 2-D stick field, with its color-coded strength. When the voter is a ball tensor, with no information known locally, the vote is generated by

rotating a stick vote in the 2-D plane and integrating all contributions. The 2-D ball field is shown in Fig. 3(c).

At each receiving site, the collected votes are combined through simple tensor addition, producing generic 2-D tensors. During voting, tokens that lie on a smooth curve reinforce each other, and the tensors deform according to the prevailing orientation. Each tensor encodes the local orientation of geometric features (given by the tensor orientation), and their saliency (given by the tensor shape and size). For a generic 2-D tensor, its curve saliency is given by $(\lambda_1-\lambda_2)$, the curve normal orientation by e_1, while its point saliency is given by λ_2. Therefore, the voting process infers curves and junctions simultaneously, while also identifying outlier noise (tokens that receive very little support). The 3-D case is similar, where salient features are points, curves and surfaces [6].

3.2 Extension to 4-D

The issues to be addressed here are the *tensorial representation* of the features in the 4-D space, the generation of *voting fields*, and the *data structures* used for vote collection. Table 1 shows all the geometric features that appear in a 4-D space and their representation as *elementary* 4-D tensors, where *n* and *t* represent normal and tangent vectors, respectively. Note that a surface in the 4-D space can be characterized by two normal vectors, or by two tangent vectors. From a *generic* 4-D tensor that results after voting, the geometric features are extracted as shown in Table 2.

The 4-D voting fields are obtained as follows. First the 4-D stick field is generated in a similar manner to the 2-D stick field (see Fig. 3(a)). Then, the other three voting fields are built by integrating all the contributions obtained by rotating a 4-D stick field around appropriate axes. In particular, the 4-D ball field – the only one directly used here – is generated according to:

$$V_{ball}(\vec{d}) = \int_0^{2\pi}\int\int R\, V_{stick}(R^{-1}\vec{d})\, R^T \, d\theta_{xy} d\theta_{xu} d\theta_{xv} \qquad (3)$$

where x, y, u, v are the 4-D coordinates axes and R is the rotation matrix with angles θ_{xy}, θ_{xu}, θ_{xv}.

The data structure used to store the tensors is an *approximate nearest neighbor (ANN) k-d tree* [8]. The space complexity of is $O(n)$, where n is the input size. The average time complexity of the voting process is $O(\mu n)$ where μ is the average number of tokens in the neighborhood. Therefore, in contrast to other voting techniques, such as the Hough Transform, both time and space complexities of the Tensor Voting methodology are *independent* of the dimensionality of the desired feature. The running time for an input of size 700 is about 20 seconds on a Pentium III (600 MHz) processor.

Feature	λ_1 λ_2 λ_3 λ_4	e_1 e_2 e_3 e_4	Tensor
point	1 1 1 1	Any 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 t_1 t_2 t_3	Stick

Table 1. Elementary tensors in 4-D

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

Table 2. A generic tensor in 4-D

4 Matching

We take as input two frames containing identical point tokens in a sparse configuration. For illustration purposes, we give a description of our approach by using a specific example – the point tokens represent an opaque **translating disk** (Fig. 2) against a static background.

Candidate matches are generated as follows: in a pre-processing step, for each token in the first frame we simply create a potential match with every point in the second frame that is located within a neighborhood (whose size is given by the scale factor) of the first token. The resulting candidates appear as a cloud of (x,y,v_x,v_y) points in the 4-D space. In our translation example we have 400 input points, and by using the procedure described above we generate an average of 5.3 candidate matches per point, among which at most one is correct. Fig. 4(a) shows the candidate matches. In order to display 4-D results, the last component of each 4-D point has been dropped – the 3 dimensions shown are x and y (in the horizontal plane), and v_x (the height). The correct matches can be already perceived as they are grouped in two parallel layers surrounded by noisy matches.

Since no information is initially known, each potential match is encoded into a 4-D *ball tensor*. Then each token casts votes in a sparse voting process (only at input token locations). Votes are generated by using the 4-D *ball voting field*. During voting there is strong support between tokens that lie on a smooth surface (layer), while communication between layers is reduced by the spatial

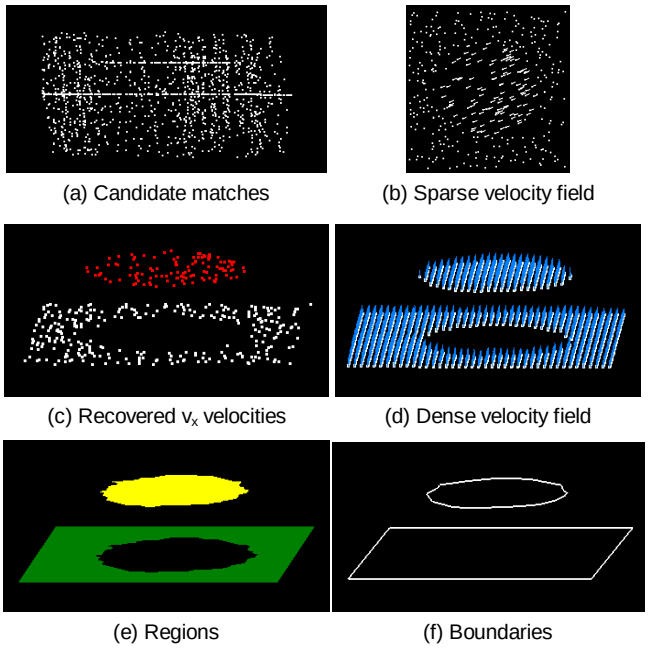


Fig. 4. Translating disk

separation in the 4-D space of both image coordinates and pixel velocities.

Wrong matches appear as isolated points, which receive little or no support, and are rejected as outliers. A measure of this support is given by the surface saliency (λ_2-λ_3). For the translating disk example, matching was 100% accurate - all 400 matches have been recovered correctly, despite the large amount of approximately 500% noise present. Fig. 4(b) shows the recovered sparse velocity field, while Fig. 4(c) shows a 3-D view of the recovered matches (the height represents the v_x velocity component).

5 Densification

We first need to obtain an estimation of the layer orientations as accurate as possible. Although local layer orientations have already been determined as a by-product during the matching process (after voting, e_1 and e_2 give the normals to layers), they may have been corrupted by the presence of wrong matches. Therefore, we perform an orientation refinement through another sparse voting process, but now with the correct matches only. The layer orientations are then found at each token as e_1 and e_2.

In order to recover boundaries and regions as *continuous* curves and surfaces, we need to first infer velocities and layer orientations at *every* image location.

Therefore we must obtain appropriate tensor values at every pixel (x,y). There may be several tensors with the same (x,y) but with different (v_x,v_y), since overlapping layers are present in the case of transparent motion. For each pixel (x,y) we try to find the best (v_x,v_y) locations at which to place the newly generated tokens. The candidates considered are all the discrete points (v_x,v_y) between the minimum and maximum velocities in the sparse tokens set, within a neighborhood of the (x,y) point. At each candidate position (x,y,v_x,v_y) we accumulate votes from the sparse tokens, according to the same Tensor Voting framework that we have used so far. After voting, the candidate tokens whose surface saliencies (λ_2-λ_3) are locally maximal are retained, and their (v_x,v_y) coordinates represent the most likely velocities at (x,y). By following this procedure at every (x,y) image location we generate a *dense velocity field*. Note that in this process, along with velocities we simultaneously infer layer orientations. Fig. 4(d) shows a 3-D view of the dense set of tokens and their associated layer orientations (only one normal shown).

6 Segmentation

The next step is to group tokens into *regions* (Fig. 4(e)), by using again the smoothness constraint. We start from an arbitrary point in the image, assign a region label to it, and try to recursively propagate this label to all its image neighbors. In order to decide whether the label must be propagated, we use the smoothness of both velocity and layer orientation as a grouping criterion. Finally, we have implemented a method to extract the *motion boundary* for each region (Fig. 4(f)), as a "partially convex hull". The process is controlled by the scale factor only, that determines the perceived level of detail (the departure from the actual convex hull).

7 Results

The case illustrated so far may be considered too simple since the only motion involved is translation. However, no assumption – such as translational, planar, or rigid motion – has been made. The *only* criterion used is the smoothness of *image* motion. To support this argument, we show next that our approach also performs very well for several other configurations.

7.1 Using motion cues only

Expanding disk (Fig. 5). The input consists of two sets of 400 points each, representing an opaque disk in expansion against a static background. After processing, only 1 match among 400 is wrong. This example demonstrates that, without special handling, our framework easily accommodates non-rigid image motion.

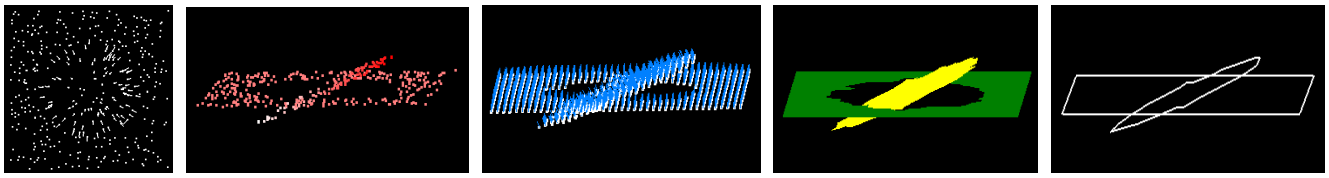


Fig. 5. Expanding disk

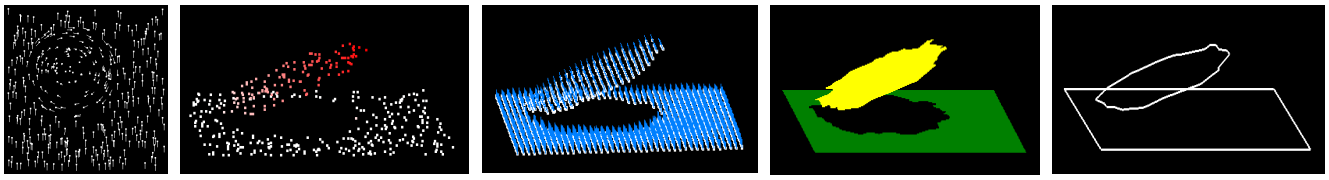


Fig. 6. Rotating disk – translating background

Rotating disk – translating background (Fig. 6). The input consists of two sets of 400 points each, representing an opaque rotating disk against a translating background. After processing, only 2 matches among 400 are wrong. This is a very difficult case even for human vision, due to the fact that around the left extremity of the disk the two motions are almost identical. In that part of the image there are points on different moving objects that are not separated, even in the 4-D space. In spite of this inherent ambiguity, our method is still able to accurately recover velocities, regions and boundaries. The key fact is that we rely not only on the 4-D positions, but also on the local layer orientations that are still different and therefore provide a good affinity measure.

7.2 Incorporating intensity information

So far we have only presented cases where no monocular information (such as intensity) is available, and the entire analysis has been performed based on motion cues *only*. Human vision is able to handle these cases remarkably well, and their study is fundamental for understanding the motion analysis process. Nevertheless they are very difficult from a computational perspective – most existing methods cannot handle such examples in a consistent and unified manner.

To further validate our approach we have also analyzed several standard image sequences, where both monocular and motion cues are available. In order to incorporate monocular information into our framework, we only needed to change the pre-processing step where candidate matches are generated. We ran a simple intensity-based cross-correlation procedure, and we retained all peaks of correlation as candidate matches. The rest of our framework remains unchanged.

Yosemite sequence (Fig. 7). We analyzed the motion from two frames of the Yosemite sequence (without the sky) to quantitatively estimate the performance of our approach. The average angular error obtained is 3.74° ± 4.3° for 100% field coverage, result which is comparable with those in the literature [1]. Also note that our method successfully recovers non-planar motion layers.

Flower Garden sequence (Fig. 8). For a qualitative estimation, we also analyzed the motion from two frames of the Flower Garden sequence. It is worth mentioning that wrong candidates generated due to occlusion are corrected during the densification step.

7.3 Handling reflections and transparency

Since our framework allows for overlapping motion layers, it can successfully handle images containing reflections and transparency. Here we consider the image $I(x,y,t)$ at time t as a combination of two patterns A and B, which have independent motions a and b:

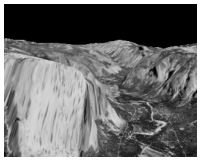
$$I(x,y,t) = A^{ta} + B^{tb} \qquad (4)$$

where A^{ta} denotes pattern A transformed by motion ta.

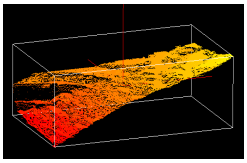
In order to obtain the dominant motion (assume it is a), we run a cross-correlation procedure, followed by a step of voting as described in the Matching section, to eliminate noisy matches. Next we use a "nulling" method [9][10], to estimate the remaining motion b. The pattern component A with velocity a is removed by moving each frame with a, then subtracting it from the following frame. The resulting difference images are:

$$\begin{aligned} D_k &= I(x,y,k+1) - I^a(x,y,k) \\ &= (A^{(k+1)a} + B^{(k+1)b}) - (A^{(k+1)a} + B^{kb+a}) \\ &= (B^b - B^a)^{kb} \end{aligned} \qquad (5)$$

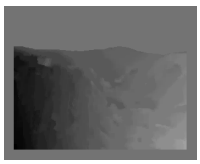
Assuming that we have three frames, the difference images are $D_0 = (B^b - B^a)$ and $D_1 = (B^b - B^a)^b$, which show a pattern $(B^b - B^a)$ moving with a single motion b. We use the same method – cross-correlation followed by voting – to determine motion b from frames D_0 and D_1.



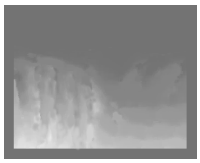
(a) an input frame



(b) motion layer (x-velocities)



(c) x-velocities

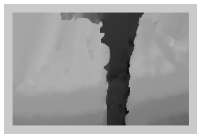


(d) y-velocities

Fig. 7. Yosemite

(a) an input frame

(b) regions



(c) x-velocities

(d) y-velocities

Fig. 8. Flower Garden

(a) an input frame

(b) registered background

(c) registered foreground

Fig. 9. Transparent motion

Finally, we put together the two sets of 4-D tokens with velocities *a* and *b*, and run a step of dense voting and grouping (as described in Sections 5 and 6) on the entire set. This process also fills any holes in the layers, which may have been produced by the noisy matches elimination. Note that the entire procedure recovers the motions without separating the patterns.

Transparent motion sequence (Fig. 9). We analyzed the motion from three frames captured with a moving camera, showing a face reflected in a framed picture. In order to show the accuracy of our results, we compute two "temporal average" images after registering the input frames using the two recovered motions (Fig. 9(b) and (c)). In each of these, the registered pattern is sharp, while the other one is blurred due to the image motion.

8 Conclusions

We have presented a novel approach for the problem of perceptual grouping from motion cues, based on a *layered 4-D representation* of data, and a *voting scheme* for token communication. Our methodology is formulated as a 4-D Tensor Voting computational framework.

Using a 4-D space for our approach is essential, since it allows for a spatial separation of the points according to both velocities and image coordinates. Consequently, the proposed framework allows tokens from the same layer to strongly support each other, while inhibiting influence from other layers or from isolated tokens.

Despite the high dimensionality, our voting scheme is both time and space efficient. It is non-iterative and the only free parameter is scale, which is an inherent characteristic of human vision.

We demonstrated the contributions of this work by analyzing several cases – opaque and transparent motion, rigid and non-rigid motion. We showed that our method successfully addresses the difficult problem of grouping from motion cues *only*, and is also able to incorporate the use of monocular cues that are present in real images.

We plan to extend our approach for real image sequences by using a more elaborate procedure for generating the initial candidates, rather than a simple cross-correlation technique. Other research directions include studying the occlusion relationships and incorporating information from multiple frames.

References

[1] J. Barron, D. Fleet, S. Beauchemin, "Performance of Optical Flow Techniques", *IJCV*, 1994, 12:1, pp. 43-77.

[2] F. Heitz, P. Bouthemy, "Multimodal Estimation of Discontinuous Optical Flow Using Markov Random Fields", *PAMI*, December 1993, 15: 12, pp. 1217-1232.

[3] S. Ghosal, "A Fast Scalable Algorithm for Discontinuous Optical Flow Estimation", *PAMI*, 1996, 18:2, pp. 181-194.

[4] S. Hsu, P. Anandan, S. Peleg, "Accurate Computation of Optical Flow by Using Layered Motion Representations", *ICPR*, 1994, pp. 743-746.

[5] Y. Weiss, "Smoothness in Layers: Motion Segmentation Using Nonparametric Mixture Estimation", *CVPR*, 1997, pp. 520-526.

[6] G. Medioni, Mi-Suen Lee, Chi-Keung Tang, "A Computational Framework for Segmentation and Grouping", *Elsevier Science*, 2000.

[7] L. Gaucher, G. Medioni, "Accurate Motion Flow Estimation with Discontinuities", *ICCV*, 1999, pp. 695-702.

[8] S. Arya, D. Mount, N. Netanyahu, R Silverman, A. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions", *Journal of the ACM*, 1998, 45:6, pp. 891-923.

[9] J. Bergen, P. Burt, R. Hingorani, S. Peleg, "A Three-Frame Algorithm for Estimating Two-Component Image Motion", *IEEE Trans. PAMI*, 1992, 14:9, pp. 886-896.

[10] R. Szeliski, S. Avidan, P. Anandan, "Layer Extraction from Multiple Images Containing Reflections and Transparency", *CVPR*, 2000, pp. 246-253.