Very High Accuracy Velocity Estimation using Orientation Tensors, Parametric Motion, and Simultaneous Segmentation of the Motion Field

Gunnar Farnebäck
Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden
gf@isy.liu.se

Abstract

In [10] we presented a new velocity estimation algorithm, using orientation tensors and parametric motion models to provide both fast and accurate results. One of the tradeoffs between accuracy and speed was that no attempts were made to obtain regions of coherent motion when estimating the parametric models. In this paper we show how this can be improved by doing a simultaneous segmentation of the motion field. The resulting algorithm is slower than the previous one, but more accurate. This is shown by evaluation on the well-known Yosemite sequence, where already the previous algorithm showed an accuracy which was substantially better than for earlier published methods. This result has now been improved further.

1. Introduction

The algorithm presented in this paper has three distinct components; estimation of orientation tensors, estimation of parametric motion models, and simultaneous segmentation of the motion field.

The estimation of orientation tensors involves spatiotemporal filtering of the volume obtained by stacking the frames of an image sequence onto each other. The filter responses are combined pointwise into 3D orientation tensors, which give a powerful representation of the oriented structures in the volume, and correspondingly of the local constraints on the motion that can be inferred from the intensity variations in the sequence. In particular this representation is well suited to deal with the aperture problem.

To improve the robustness of the velocity estimation it is assumed that the motion locally conforms to a parametric motion model. In this paper we only discuss the affine model, but the framework works for a larger class of motion models. The model parameters can in a very straightforward manner be computed from the orientation tensors in the region.

The weak point of the previous paragraph is that we would want the region to contain a coherent motion (with respect to the affine motion model). In particular we do not want it to span discontinuities in the motion field, since this would clearly degrade the estimated velocity field. On the other hand, in order to segment out regions of coherent motion, we would first need to know the velocities. The solution to this problem is to do a simultaneous segmentation and velocity estimation. This is accomplished with the help of a region growing based segmentation algorithm.

Spatiotemporal filtering and parametric motion models, in particular affine motion, are today standard components of motion estimation algorithms. The use of orientation tensors is, however, less common. The basic relations between 3D orientation tensors and motion have been explored in e.g. [4, 15, 14, 25, 13]. A more sophisticated tensor based algorithm has been presented by Karlholm [19]. A survey of some other tensor based approaches can be found in [17]. The idea to do simultaneous segmentation and velocity estimation is also well-known, see e.g. [7] for an overview.

The first two components of this algorithm are common with the algorithm presented in [10]. More details on the orientation tensor estimation can be found in [11]. An early prototype of the segmentation algorithm used here, with an emphasis on segmentation rather than velocity estimation, can be found in [8]. More details on all aspects of the algorithm presented here can be found in [9].

2. Preliminaries

Before we can discuss the segmentation part of the algorithm, we need to recapitulate the algorithm presented in [10]. Obviously this will have to be done briefly. For further

details the reader is referred to [9, 10, 11].

2.1. Orientation tensors

By stacking the frames of an image sequence onto each other we obtain a spatiotemporal image volume with two spatial dimensions and a third temporal dimension. It can be verified that motion in the sequence is directly related to oriented structures in the volume.

A powerful representation of local orientation is the orientation tensor [13, 20]. In 3D this tensor takes the form of a 3×3 symmetric positive semidefinite matrix $\mathbf{T}$ and the quadratic form $\hat{\mathbf{u}}^T\mathbf{T}\hat{\mathbf{u}}$ can be interpreted as a measure of how much the signal locally varies in the direction given by $\hat{\mathbf{u}}$.

To estimate 3D orientation tensors we start by locally, for each neighborhood, projecting the signal onto a second degree polynomial, according to the signal model

$$f(\mathbf{x}) \sim \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c. \tag{1}$$

The parameters $\mathbf{A}$, $\mathbf{b}$, and c are computed by a Gaussian weighted least squares approximation of the signal. This can be implemented very efficiently by a hierarchical scheme of separable convolutions. From the model parameters, the orientation tensor is constructed by

$$\tilde{\mathbf{T}} = \mathbf{A}\mathbf{A}^T + \gamma\mathbf{b}\mathbf{b}^T, \tag{2}$$

where γ is a non-negative weight factor between the even and the odd parts of the signal. As a further preprocessing step we compute the isotropy compensated tensor

$$\mathbf{T} = \tilde{\mathbf{T}} - \lambda_{\min}\mathbf{I}, \tag{3}$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\tilde{\mathbf{T}}$.

2.2. Estimating parametric motion models

A 2D velocity vector $(v_x \; v_y)^T$, measured in pixels per frame, can be extended to a 3D spatiotemporal directional vector $\mathbf{v}$ and a unit directional vector $\hat{\mathbf{v}}$ by

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}, \quad \hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}. \tag{4}$$

The orientation tensor computed above has the useful property that for a translating neighborhood it will satisfy the relation $\mathbf{v}^T\mathbf{T}\mathbf{v} = 0$. In the presence of the aperture problem, all velocity vectors with the correct normal velocity component will satisfy the relation. In principle we could now compute velocity vectors from the orientation tensors point for point. This would, however, have the drawback of yielding rather noisy estimates and of course we would have the usual problems with the aperture problem.

Instead we assume that we have a region where the motion is coherent with respect to the affine motion model,

$$\begin{aligned} v_x(x,y) &= ax + by + c, \\ v_y(x,y) &= dx + ey + f, \end{aligned} \tag{5}$$

where x and y are image coordinates. This can be rewritten in terms of the spatiotemporal vector (4) as

$$\mathbf{v} = \mathbf{S}\mathbf{p}, \quad \text{where} \tag{6}$$

$$\mathbf{S} = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \tag{7}$$

$$\mathbf{p} = \begin{pmatrix} a & b & c & d & e & f & 1 \end{pmatrix}^T. \tag{8}$$

The parameters of the model can be estimated directly from the orientation tensors in the region by using $\mathbf{v}^T\mathbf{T}\mathbf{v}$ as a cost function. Summing over the region and applying the motion model we get

$$d_{\text{tot}}(\mathbf{p}) = \sum_i \mathbf{v}_i^T\mathbf{T}_i\mathbf{v}_i = \sum_i \mathbf{p}^T\mathbf{S}_i^T\mathbf{T}_i\mathbf{S}_i\mathbf{p} = \mathbf{p}^T\mathbf{Q}_{\text{tot}}\mathbf{p}, \tag{9}$$

where

$$\mathbf{Q}_{\text{tot}} = \sum_i \mathbf{Q}_i = \sum_i \mathbf{S}_i^T\mathbf{T}_i\mathbf{S}_i. \tag{10}$$

We obtain the motion model parameters by minimizing (9) under the constraint that the last element of $\mathbf{p}$ be 1. In order to do this we partition $\mathbf{p}$ and $\mathbf{Q}_{\text{tot}}$ as

$$\mathbf{p} = \begin{pmatrix} \bar{\mathbf{p}} \\ 1 \end{pmatrix}, \quad \mathbf{Q}_{\text{tot}} = \begin{pmatrix} \bar{\mathbf{Q}} & \mathbf{q} \\ \mathbf{q}^T & \alpha \end{pmatrix}, \tag{11}$$

turning (9) into

$$d_{\text{tot}}(\mathbf{p}) = \bar{\mathbf{p}}^T\bar{\mathbf{Q}}\bar{\mathbf{p}} + \bar{\mathbf{p}}^T\mathbf{q} + \mathbf{q}^T\bar{\mathbf{p}} + \alpha, \tag{12}$$

which is minimized by

$$\bar{\mathbf{p}} = -\bar{\mathbf{Q}}^{-1}\mathbf{q}. \tag{13}$$

Compared to first estimating the velocity vectors on a point by point basis from the orientation tensors and then fitting them to the motion model, this approach has two important advantages. The first advantage is that the effects of noise and inaccuracies in the tensor estimation are reduced significantly. The second advantage is that even if the aperture problem is present in some part of the region, information obtained from other parts can help to fill in the missing velocity component. There does remain a possibility that the motion field cannot be uniquely determined, but that requires the signal structures over the whole region to be oriented in such a way that the motion becomes ambiguous; a generalized aperture problem. Additionally this approach inherently incorporates all information we have at each point, whether it is a full velocity vector or only the normal component.

3. Simultaneous segmentation and velocity estimation

For the best results, the estimation of the affine motion field should be done over a region with coherent motion. In [10] we ignored this completely, as a speed-accuracy tradeoff, instead using weighted neighborhoods around each pixel as regions.

In this section we present an efficient algorithm for simultaneous segmentation and velocity estimation, only given an orientation tensor field for one frame. The goal of the segmentation is to partition the image into a set of disjoint regions, so that each region is characterized by a coherent motion, with respect to the affine motion model. In this section a region R is defined to be a nonempty, connected set of pixels. The segmentation algorithm is based on a competitive region growing approach. The basic algorithm is first presented in abstract form.

3.1. The competitive algorithm

To each region R is associated a cost function $C_{\mathsf{R}}(\mathbf{x})$, which is defined for all pixels in the image. Regions are extended by adding one pixel at a time. To preserve connectivity the new pixel must be adjacent to the region, and to preserve disjointedness it must not already be assigned to some other region. The new pixel is also chosen as inexpensive as possible. The details are as follows.

Let the border $\Delta\mathsf{R}$ of region R be the set of unassigned pixels in the image which are adjacent to some pixel in R. For each region R, the possible candidate, $N(\mathsf{R})$, to be added to the region is the least expensive pixel bordering to R, i.e.

$$N(\mathsf{R}) = \arg\min_{\mathbf{x}\in\Delta\mathsf{R}} C_{\mathsf{R}}(\mathbf{x}). \tag{14}$$

The corresponding minimum cost for adding the candidate to the region is denoted $C_{\min}(\mathsf{R})$. In the case of an empty border, $N(\mathsf{R})$ is undefined and $C_{\min}(\mathsf{R})$ is infinite.

Assuming that a number of regions $\{\mathsf{R}_n\}$ in some way have been obtained, the rest of the image is partitioned as follows.

1. Find the region R_i for which the cost to add a new pixel is the least, i.e. let $i = \arg\min_n C_{\min}(\mathsf{R}_n)$.
2. Add the least expensive pixel $N(\mathsf{R}_i)$ to R_i.
3. Repeat the first two steps until no unassigned pixels remain.

Notice that it does not matter what the actual values of the cost functions are. It is only relevant which of them is smallest. Hence the algorithm is called competitive.

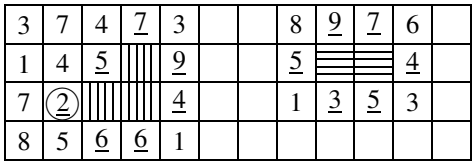

Figure 1. Illustration of the competitive algorithm

An illustration of this algorithm is provided in figure 1. We have two regions, one to the left marked with vertical bars and one to the right marked with horizontal bars. Around each region are shown the values of the cost function $C_{\mathsf{R}}(\mathbf{x})$ for respective region. The underlined values belong to each border $\Delta\mathsf{R}$, assuming four-connectivity. The minimum cost $C_{\min}(\mathsf{R})$ is 2 for the left region and 3 for the right region. Hence the circled pixel is added to the left region in this iteration. In the next iteration the pixel with an underlined 3 will be added to the right region because the minimum cost for the left region is increased to 4 after the addition of the circled pixel.

3.2. Candidate regions

A fundamental problem with the simultaneous segmentation and velocity estimation approach is that we typically need a segmentation in order to compute the motion model parameters, and we need velocity estimates in order to partition the image into regions. Since we assume no a priori knowledge about the segmentation of the image, we use the concept of *candidate regions* to introduce preliminary regions into the algorithm.

To begin with we arbitrarily fill the image with a large number of overlapping rectangular candidate regions, e.g. squares of the size 21×21, with a distance between the center points of 4 pixels. The exact numbers are not critical. For each candidate region we then compute the optimal motion model parameters as described in section 2.2. Obviously these rectangular regions are not at all adapted to the motion field of the frame and as a consequence the computed motion models are likely to be suboptimal. In order to improve the candidate regions we use a procedure called *regrowing*.

The regrowing procedure is the first application of the competitive algorithm. Regrowing is performed for one candidate region at a time, which means that there is no competition between different regions but rather between the pixels. To begin with the candidate region contains only one pixel, its *starting point*, which was also the center point of the initial rectangle. The cost function used is $\frac{\hat{\mathbf{v}}\mathbf{T}\hat{\mathbf{v}}}{\operatorname{tr}\mathbf{T}}$, where $\mathbf{v}$ is the velocity given by the candidate region's current motion model. The competitive algorithm is then run

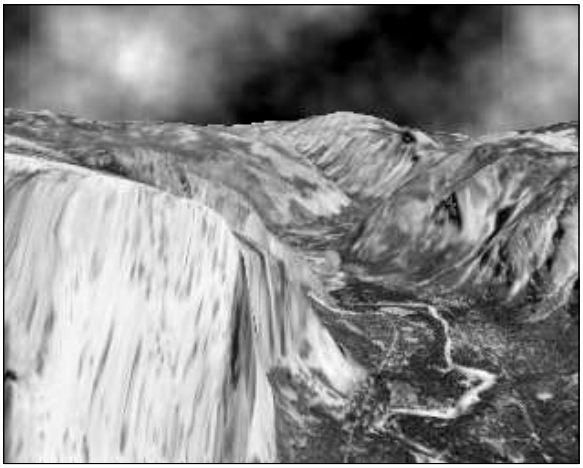

Figure 2. One frame of the Yosemite sequence.

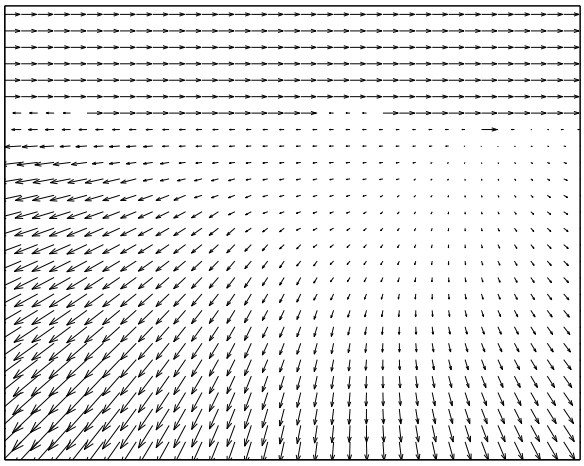

Figure 3. Corresponding true velocity field (subsampled).

until the candidate region has grown to a specified size. This size is called the *candidate region size*, m_0 and is a design parameter of the segmentation algorithm. The effect of the regrowing procedure is that the candidate region now consists of the m_0 connected pixels, starting from a fixed point, that are most consistent with the candidate region's motion model. When the candidate region has been regrown, new optimal parameters are computed. Each candidate region is regrown twice, a number which seems to be sufficient to obtain reasonably coherent regions.

The choice of m_0 is a somewhat complex tradeoff. A small value yields higher speed and the possibility to detect smaller coherent regions, while a higher value gives more robust parameter estimation and reduces tendencies to oversegment.

3.3. Segmentation algorithm

Having obtained candidate regions, the rest of the segmentation algorithm is a matter of alternately converting candidate regions into real regions and letting the latter grow. In contrast to the candidate regions, the real regions are not allowed to overlap but have to be disjoint. While the candidate regions are allowed to overlap each other they must not overlap the real regions, which means that they have to be regrown from time to time, taking this restriction into account. To accomodate the inclusion of new regions, the competitive algorithm is extended to have the following steps, to be iterated as long as there are nonassigned pixels left:

1. Regrow the candidate regions which are currently overlapping a real region. If a candidate region cannot be regrown to its full size m_0, it is removed. The same thing happens when a candidate region's starting point becomes occupied by a real region. The cost of the most expensive included pixel is called the *maximum cost* of the candidate region.

2. Find the candidate region with the least maximum cost. This is the aspirant for inclusion among the real regions.

3. As in the competitive algorithm, find the least expensive pixel that may be added to one of the already existing real regions.

4. Compare the least maximum cost from step 2 with the cost of the least expensive pixel in step 3.

   (a) If the least maximum cost is smallest, raise the corresponding candidate region to the status of a real region.

   (b) Otherwise, add the least expensive pixel to the corresponding region.

In the first iteration there are no real regions yet, so the first thing that happens is that the best candidate region is transformed into the first real region.

When a candidate region is converted into a real region in step 4a, all its m_0 pixels are immediately included in the real region.

To see how the segmentation algorithm works, one frame of the Yosemite sequence, figures 2 and 3, has been segmented. In figure 4 we can see how the regions develop and how new regions are successively added.

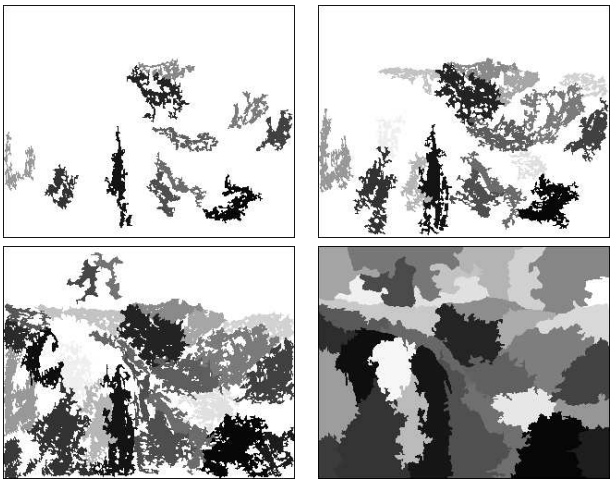

Figure 4. Development of regions.

While the comparison in step 4 can be made directly between the given values, it is beneficial to introduce a design parameter λ, with which the least maximum cost is multiplied before the comparison is made. The effect of λ is that for a large value, new regions are added only if it would be very expensive to enlarge the existing ones. This may be desired e.g. if the segmentation is intended for a video coding application, where excessive fragmentation into many small regions can be costly. A small λ value means that existing regions are enlarged only if there are pixels available that are very consistent with the motion models, which we in this paper prefer since we are more interested in the velocity field than in the segmentation itself.

The regrowing of candidate regions in step 1 of the algorithm may seem prohibitively computationally expensive. In practice though, a reasonable shortcut is to assume that the maximum cost always increases when a candidate region has to be regrown. Therefore it is sufficient to regrow candidate regions only when the least maximum cost is smaller than the least expensive pixel and also only a few of the top candidate regions need to be regrown.

An algorithm with some basic elements in common with the competitive algorithm can be found in [1], being applied to grayscale segmentation. The algorithm presented here was developed later but independently.

4. Evaluation

The algorithm has been implemented in Matlab, with certain parts in the form of C mex files. Typical running times for the algorithm on a 360 MHz SUN Ultra 60, computing the velocity for one frame of the 252×316 Yosemite sequence, is in the order of a minute. As a comparison, the faster algorithm presented in [10] takes about 16 seconds using the affine motion model and about 3.5 seconds using the simpler constant motion model.

Source code for the implementation is available at `http://www.isy.liu.se/~gf`.

The algorithm has been evaluated on a commonly used test sequence with known velocity field, Lynn Quam's Yosemite sequence [16], figures 2 and 3. This synthetic sequence was generated with the help of a digital terrain map and therefore has a motion field with depth variation and discontinuities at occlusion boundaries.

The accuracy of the velocity estimates has been measured using the average spatiotemporal angular error, $\arccos(\hat{\mathbf{v}}_{\text{est}}^T \hat{\mathbf{v}}_{\text{true}})$ [3]. The sky region is excluded from the error analysis because the variations in the cloud textures induce an image flow that is quite different from the ground truth values computed solely from the camera motion.

The orientation tensors have been computed using spatiotemporal filters of effective size $9 \times 9 \times 9$ and a standard deviation $\sigma = 1.4$ for the Gaussian weighting function. The tensor computation also involves the parameter γ, equation (2), which has been set to $\frac{1}{8}$. With the design parameters for the segmentation algorithm, λ and m_0, set to 0.06 and 500 respectively, we obtain an average angular error of 1.30° with standard deviation 2.29°.

An interesting question is how the design parameters affect the results. This has been studied in detail in [9]. It turns out that the result is not very sensitive to changes in the γ and λ values. The value of σ is more critical, but this is predictable since it controls the scale of the structures for which the orientation is estimated. The dependence on m_0 is more troublesome, since the results fluctuate quite unpredictably between 1.25° and 1.45° when m_0 is varied, as shown in figure 5.

While the sensitivity to the value of m_0 is disturbing, it points out a possible strategy for improving the algorithm. By computing the velocity for a number of different m_0 values and averaging the results, we can reduce the overall errors and stabilize the estimates. This does of course come at the cost of more computations, but since we in this paper have settled for accuracy being more important than speed, we find the addition worthwhile and include it in the algorithm. Using 11 evenly spaced values $400, 420, \ldots, 600$ of m_0 we get an average angular error of 1.14° and a standard deviation of 2.14°. Picking 11 m_0 values randomly in the same interval, we consistently get average angular errors between 1.13° and 1.18°. The running time increases to about 5.5 minutes with this modification.

Statistics on the distribution of errors are given in table 1. Comparison with previously published results, table 2, shows that the algorithm presented here gives excellent accuracy, clearly improving on the results presented in [10].

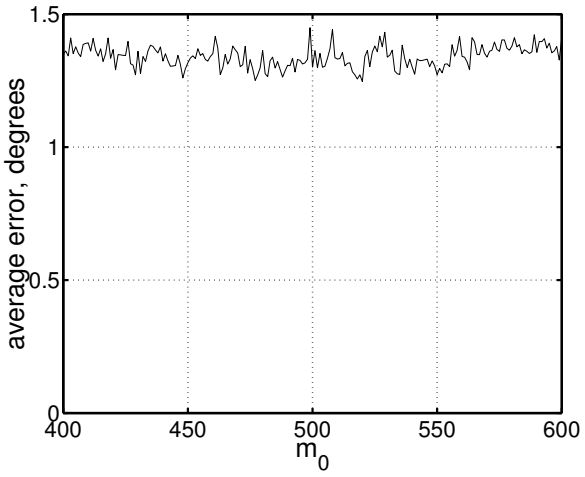

Figure 5. Average angular errors while varying $\mathbf{m_0}$.

		affine
Average error		$1.14°$
Standard deviation		$2.14°$
Proportion of estimates with errors below:	$< 0.5°$	32.0%
	$< 1°$	64.4%
	$< 2°$	87.8%
	$< 3°$	94.0%
	$< 5°$	98.0%
	$< 10°$	99.7%

Table 1. Distribution of errors.

Technique	Average error	Standard deviation	Density
Lucas & Kanade [22]	$2.80°$	$3.82°$	35%
Uras *et al.* [24]	$3.37°$	$3.37°$	14.7%
Fleet & Jepson [12]	$2.97°$	$5.76°$	34.1%
Black & Anandan [5]	$4.46°$	$4.21°$	100%
Szeliski & Coughlan [23]	$2.45°$	$3.05°$	100%
Black & Jepson [6]	$2.29°$	$2.25°$	100%
Ju *et al.* [18]	$2.16°$	$2.0°$	100%
Karlholm [19]	$2.06°$	$1.72°$	100%
Lai & Vemuri [21]	$1.99°$	$1.41°$	100%
Bab-Hadiashar & Suter [2]	$1.97°$	$1.96°$	100%
Farnebäck, constant motion [10]	$1.94°$	$2.31°$	100%
Farnebäck, affine motion [10]	$1.40°$	$2.57°$	100%
This algorithm	$1.14°$	$2.14°$	100%

Table 2. Comparison with other methods, Yosemite sequence. The sky region is excluded for all results.

5. Summary and conclusions

We have shown how the algorithm presented in [10] can be combined with a region growing segmentation algorithm to provide a simultaneous segmentation and velocity estimation method. The result is improved accuracy at the cost of speed. By averaging multiple runs with slightly different parameters, the velocity estimates can be improved even further. The final algorithm, while being somewhat slow, gives very high accuracy and clearly outperforms previously published methods, at least on the Yosemite sequence.

Acknowledgments

The author wants to acknowledge the financial support of WITAS: The Wallenberg Laboratory for Information Technology and Autonomous Systems.

References

[1] R. Adams and L. Bischof. Seeded region growing. 16(6):641–647, June 1994.

[2] A. Bab-Hadiashar and D. Suter. Robust optic flow computation. *International Journal of Computer Vision*, 29(1):59–77, August 1998.

[3] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *Int. J. of Computer Vision*, 12(1):43–77, 1994.

[4] J. Bigün. *Local Symmetry Features in Image Processing.* PhD thesis, Linköping University, Sweden, 1988. Dissertation No 179, ISBN 91-7870-334-4.

[5] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, Jan. 1996.

[6] M. J. Black and A. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. 18(10):972–986, 1996.

[7] F. Dufaux and F. Moscheni. Segmentation-based motion estimation for second generation video coding techniques. In L. Torres and M. Kunt, editors, *Video Coding: The Second Generation Approach*, chapter 6, pages 219–263. Kluwer Academic Publishers, 1996.

[8] G. Farnebäck. Motion-based Segmentation of Image Sequences. Master's Thesis LiTH-ISY-EX-1596, Computer Vision Laboratory, SE-581 83 Linköping, Sweden, May 1996.

[9] G. Farnebäck. Spatial Domain Methods for Orientation and Velocity Estimation. Lic. Thesis LiU-Tek-Lic-1999:13, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, March 1999. Thesis No. 755, ISBN 91-7219-441-3.

[10] G. Farnebäck. Fast and Accurate Motion Estimation using Orientation Tensors and Parametric Motion Models. In *Proceedings of 15th International Conference on Pattern Recognition*, volume 1, pages 135–139, Barcelona, Spain, September 2000. IAPR.

[11] G. Farnebäck. Orientation Estimation Based on Weighted Projection onto Quadratic Polynomials. In B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2000: proceedings*, pages 89–96, Saarbrücken, November 2000.

[12] D. J. Fleet and A. D. Jepson. Computation of Component Image Velocity from Local Phase Information. *Int. Journal of Computer Vision*, 5(1):77–104, 1990.

[13] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.

[14] L. Haglund. *Adaptive Multidimensional Filtering*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, October 1992. Dissertation No 284, ISBN 91-7870-988-1.

[15] L. Haglund, H. Bårman, and H. Knutsson. Estimation of Velocity and Acceleration in Time Sequences. In *Proceedings of the 7th Scandinavian Conference on Image Analysis*, pages 1033–1041, Aalborg, Denmark, August 1991. Pattern Recognition Society of Denmark.

[16] D. J. Heeger. Model for the extraction of image flow. *J. Opt. Soc. Am. A*, 4(8):1455–1471, 1987.

[17] B. Jähne, H. Haussecker, and P. Geissler, editors. *Handbook of Computer Vision and Applications*. Academic Press, 1999. ISBN 0-12-379770-5.

[18] S. X. Ju, M. J. Black, and A. D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *Proceedings CVPR'96*, pages 307–314. IEEE, 1996.

[19] J. Karlholm. *Local Signal Models for Image Sequence Analysis*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 1998. Dissertation No 536, ISBN 91-7219-220-8.

[20] H. Knutsson. Representing Local Structure Using Tensors. In *The 6th Scandinavian Conference on Image Analysis*, pages 244–251, Oulu, Finland, June 1989. Report LiTH–ISY–I–1019, Computer Vision Laboratory, Linköping University, Sweden, 1989.

[21] S.-H. Lai and B. C. Vemuri. Reliable and efficient computation of optical flow. *International Journal of Computer Vision*, 29(2):87–105, August/September 1998.

[22] B. Lucas and T. Kanade. An Iterative Image Registration Technique with Applications to Stereo Vision. In *Proc. Darpa IU Workshop*, pages 121–130, 1981.

[23] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *Proc. IEEE Conference on Computer Vision Pattern Recognition*, pages 194–201, Seattle, Washington, 1994.

[24] S. Uras, F. Girosi, A. Verri, and V. Torre. A computational approach to motion perception. *Biological Cybernetics*, pages 79–97, 1988.

[25] C-F. Westin. *A Tensor Framework for Multidimensional Signal Processing*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 1994. Dissertation No 348, ISBN 91-7871-421-4.