Fast and Accurate Motion Estimation using Orientation Tensors and Parametric Motion Models

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

Abstract

Motion estimation in image sequences is an important step in many computer vision and image processing applications. Several methods for solving this problem have been proposed, but very few manage to achieve a high level of accuracy without sacrificing processing speed.

This paper presents a novel motion estimation algorithm, which gives excellent results on both counts. The algorithm starts by computing 3D orientation tensors from the image sequence. These are combined under the constraints of a parametric motion model to produce velocity estimates. Evaluated on the well-known Yosemite sequence, the algorithm shows an accuracy which is substantially better than for previously published methods. Computationally the algorithm is simple and can be implemented by means of separable convolutions, which also makes it fast.

1 Introduction

Motion estimation algorithms always involve a trade-off between speed and accuracy. The method presented here is primarily intended to be accurate but some compromises have been made in order to keep it fast as well.

The primary measurements on the image sequence are made by spatiotemporal filtering, in the form of estimation of 3D orientation tensors. This gives a powerful representation of the local constraints on the motion that can be inferred from the intensity variations. In neighborhoods where the aperture problem only allows direct detection of the normal velocity component, this fact is accurately reflected in the tensor. The tensor field is also well suited for estimation of parametric motion models over larger regions. The algorithm uses this technique to improve the accuracy; for each point the velocity is computed under the constraint of affine motion in a neighborhood. Other parametric motion models than the affine can be used as well. The constant motion model (pure translation) in particular gives less accuracy but higher speed.

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. [3, 10, 9, 20, 8]. A more sophisticated tensor based algorithm has been presented by Karlholm [14]. His algorithm has some elements in common with the one presented here, but differs in the method for estimation of tensors and in that it uses a maximum likelihood approach to estimate motion model parameters. A survey of some other tensor based approaches can be found in [12].

2 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 is easy to see that movement in the image sequence induces structures with certain orientations in the volume. A translating point, e.g., is transformed into a line whose direction in the volume directly corresponds to its velocity.

A powerful representation of local orientation is the orientation tensor [8, 15]. 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 use a new method by Farnebäck, described in [6]. The basis for this method is to locally, for each neighborhood, project 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 weighted least squares approximation of the signal, where the weighting function normally is a Gaussian. (As is shown in [6], the weighting function must be chosen with care. Unweighted projection is *not* to be recommended.) It turns out that this in practice can be implemented very efficiently by a hierarchical scheme of separable convolutions. From the model parameters, the orientation tensor is constructed by

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

where γ is a non-negative weight factor between the even and the odd parts of the signal. The value of γ, as well as the standard deviation σ of the Gaussian weighting function, depend on the scale of the structures for which we wish to estimate orientation and are available as design parameters.

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

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}, \quad \hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}. \tag{3}$$

Ideally, if we have exact translation and no noise, there would locally be no variation in the volume along the direction of $\hat{\mathbf{v}}$. Thus we would have $\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}} = 0$ and the velocity could be recovered from the eigenvector of $\mathbf{T}$ with eigenvalue zero. In the case of a moving line we have the aperture problem, i.e. only the normal velocity component can be detected, which causes $\mathbf{T}$ to have a double eigenvalue zero. Then all velocities with the correct normal velocity satisfy $\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}} = 0$. Further details on how to compute point velocity and normal velocity from tensors can be found in [8].

In practice we cannot require $\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}}$ to be zero, but it suffices to minimize the expression. The minimum value is given by the smallest eigenvalue λ_3 and by writing $\mathbf{T} = \tilde{\mathbf{T}} + \lambda_3\mathbf{I}$ we can see that $\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}} = \hat{\mathbf{v}}^T\tilde{\mathbf{T}}\hat{\mathbf{v}} + \lambda_3\hat{\mathbf{v}}^T\mathbf{I}\hat{\mathbf{v}} = \hat{\mathbf{v}}^T\tilde{\mathbf{T}}\hat{\mathbf{v}} + \lambda_3$. The conclusion is that we can remove $\lambda_3\mathbf{I}$, the isotropic part, from the tensor without changing the solution to the minimization problem. For computational reasons we will later prefer to minimize $\mathbf{v}^T\mathbf{T}\mathbf{v}$ rather than $\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}}$ but then we have $\mathbf{v}^T\mathbf{T}\mathbf{v} = \mathbf{v}^T\tilde{\mathbf{T}}\mathbf{v} + \lambda_3\mathbf{v}^T\mathbf{v}$, which would be clearly biased against large velocities. Therefore we replace the original tensor $\mathbf{T}$ by the isotropy compensated tensor $\tilde{\mathbf{T}}$. To simplify the notation we drop the tilde in the rest of the presentation.

3 Estimating a parameterized velocity field

Rather than estimating the velocity from the tensors point for point we assume that the velocity field over a region can be parameterized according to some motion model and use all the tensors in the region to compute the parameters.

To present the method we use the affine motion model

$$\begin{aligned} v_x(x,y) &= ax + by + c, \\ v_y(x,y) &= dx + ey + f, \end{aligned} \tag{4}$$

where x and y are image coordinates. This can be rewritten in terms of the spatiotemporal vector (3) as

$$\mathbf{v} = \mathbf{S}\mathbf{p}, \quad \text{where} \tag{5}$$

$$\mathbf{S} = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \tag{6}$$

$$\mathbf{p} = \begin{pmatrix} a & b & c & d & e & f & 1 \end{pmatrix}^T. \tag{7}$$

In order to estimate the parameters $\mathbf{p}$ of the motion model we introduce the cost functions

$$d(\mathbf{v},\mathbf{T}) = \mathbf{v}^T\mathbf{T}\mathbf{v}, \quad \text{and} \tag{8}$$

$$d_{\text{tot}} = \sum_i d(\mathbf{v}_i, \mathbf{T}_i), \tag{9}$$

where the summation is over all points of the region. Applying the affine motion model we obtain from (5) and (8)

$$d(\mathbf{v},\mathbf{T}) = \mathbf{v}^T\mathbf{T}\mathbf{v} = \mathbf{p}^T\mathbf{S}^T\mathbf{T}\mathbf{S}\mathbf{p} = \mathbf{p}^T\mathbf{Q}\mathbf{p}, \tag{10}$$

where $\mathbf{Q} = \mathbf{S}^T\mathbf{T}\mathbf{S}$ is a positive semidefinite quadratic form. Summing these over the region transforms (9) into

$$\begin{aligned} d_{\text{tot}}(\mathbf{p}) &= \sum_i d(\mathbf{v}_i, \mathbf{T}_i) = \sum_i \mathbf{p}^T\mathbf{S}_i^T\mathbf{T}_i\mathbf{S}_i\mathbf{p} \\ &= \mathbf{p}^T\left(\sum_i \mathbf{Q}_i\right)\mathbf{p} = \mathbf{p}^T\mathbf{Q}_{\text{tot}}\mathbf{p}, \end{aligned} \tag{11}$$

which should be minimized 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

$$\mathbf{p} = \begin{pmatrix} \bar{\mathbf{p}} \\ 1 \end{pmatrix}, \quad \mathbf{Q}_{\text{tot}} = \begin{pmatrix} \bar{\mathbf{Q}} & \mathbf{q} \\ \mathbf{q}^T & \alpha \end{pmatrix}, \tag{12}$$

turning (11) into

$$d_{\text{tot}}(\mathbf{p}) = \bar{\mathbf{p}}^T\bar{\mathbf{Q}}\bar{\mathbf{p}} + \bar{\mathbf{p}}^T\mathbf{q} + \mathbf{q}^T\bar{\mathbf{p}} + \alpha, \tag{13}$$

which is minimized by

$$\bar{\mathbf{p}} = -\bar{\mathbf{Q}}^{-1}\mathbf{q} \tag{14}$$

giving the minimum value

$$\alpha - \mathbf{q}^T\bar{\mathbf{Q}}^{-1}\mathbf{q}. \tag{15}$$

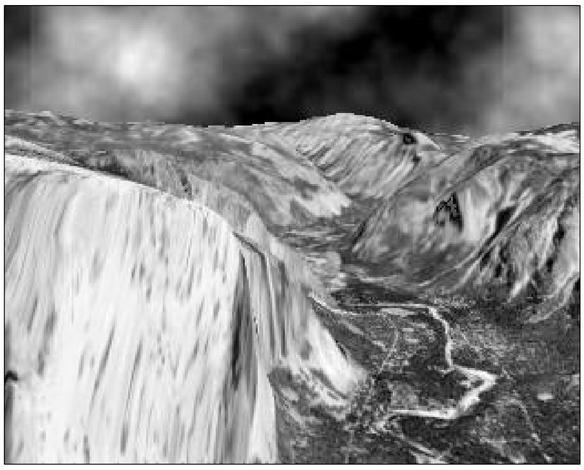

Figure 1. One frame of the Yosemite sequence.

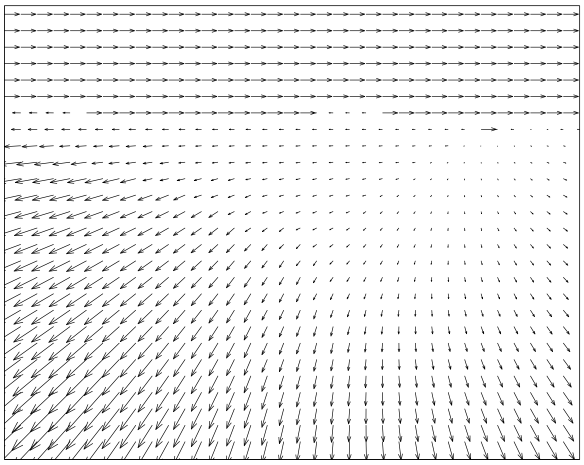

Figure 2. Corresponding true velocity field (subsampled).

In order to use this method of parameter estimation, the necessary and sufficient property of the motion model is that it is linear in its parameters, i.e. that it can be expressed in the form (5). See [6] for further examples.

There are two important advantages to estimating the velocity over a whole region rather than point by point. 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.

A disadvantage with velocity estimation over a whole region is that it is assumed that the true velocity field is at least reasonably consistent with the chosen motion model. Thus we would like to somehow find such regions. This can be done, e.g. by simultaneous segmentation and velocity estimation as in [6], but at the cost of performance. Here we choose to give priority to speed, ignoring this problem.

4 A fast velocity estimation algorithm

Instead of trying to obtain regions with a coherent motion, we minimize a weighted distance measure for a motion model around each point, i.e. (11) is replaced by

$$\begin{aligned} d_{\text{tot}}(\mathbf{p}) &= \sum_i w_i d(\mathbf{v}_i, \mathbf{T}_i) = \mathbf{p}^T \left(\sum_i w_i \mathbf{Q}_i \right) \mathbf{p} \\ &= \mathbf{p}^T \mathbf{Q}_{\text{tot}} \mathbf{p}, \end{aligned} \qquad (16)$$

where the sum is taken over a neighborhood of the current point and the weights w_i are given by a Gaussian. In effect this means that we convolve the quadratic forms $\mathbf{Q}_i$ over the image with the weight function, and this operation can be efficiently computed by separable convolution. We can also view this as a weighted averaging of the quadratic forms.

The optimal parameters are computed at each point according to (14) by solving a linear equation system and the corresponding velocity estimates are computed by (5). It also turns out that the minimum value (15) can be used as a confidence measure, with small values indicating high confidence in the estimated velocity, since it indicates how well the local neighborhood is consistent with the motion model in use.

The algorithm has been implemented in Matlab, with convolution and solution of multiple equation systems in the form of C mex files. In addition to the affine motion model, the simpler constant motion model (with $\mathbf{S} = \mathbf{I}$, implying $\mathbf{Q} = \mathbf{T}$, and $\mathbf{p} = (a \;\; b \;\; 1)^T$) has also been implemented. Typical running times for the algorithm on a 360 MHz SUN Ultra 60 are given below and relate to the computation of the velocity for one frame of the 252×316 Yosemite sequence.

With the affine motion model, tensors computed using filters of effective size $11 \times 11 \times 11$, and a 41×41 averaging kernel takes about 16 seconds. Of these, 1.8 seconds are spent on tensor estimation, 0.5 seconds on isotropy

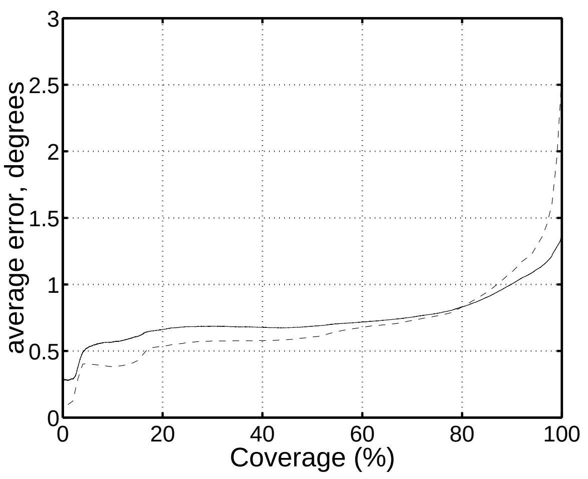

Figure 3. Angular errors for different levels of coverage. Affine motion model.

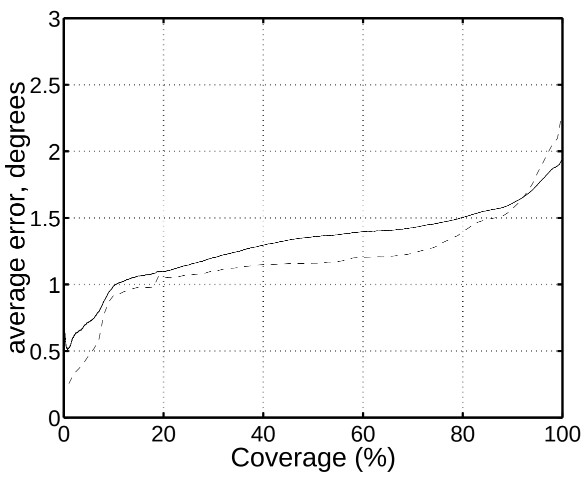

Figure 4. Angular errors for different levels of coverage. Constant motion model.

compensation, 0.3 seconds on computation of the quadratic forms $\mathbf{Q}_i$, 8.6 seconds on averaging these, and 4.8 seconds on solving for the velocity.

With the constant velocity motion model, $9 \times 9 \times 9$ tensors, and 15×15 averaging, the running time is about 3.5 seconds, with 1.6 seconds for tensor estimation, 0.5 seconds for isotropy compensation, 1.2 seconds for averaging, and 0.2 seconds for solving for the velocity.

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

5 Evaluation

The velocity estimation algorithm has been evaluated on a commonly used test sequence with known velocity field, Lynn Quam's Yosemite sequence [11], figures 1 and 2. This synthetic sequence is 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, $\arccos(\hat{\mathbf{v}}_{\text{est}}^T \hat{\mathbf{v}}_{\text{true}})$ [2]. 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.

Using the affine motion model, with filter sizes as in the discussion on running times and $\sigma = 1.6$, $\gamma = \frac{1}{256}$, and $\sigma_{\text{avg}} = 6.5$, yields an average angular error of $1.40°$ with standard deviation $2.57°$. Using the constant motion model, $\sigma = 1.4$, $\gamma = \frac{1}{32}$, and $\sigma_{\text{avg}} = 3.5$, the angular error increases to $1.94°$ with standard deviation $2.31°$. Figures 3 and 4 show the results at different levels of coverage, using the confidence values (15) to choose which points to include. The dashed lines give the corresponding standard deviations.

		affine	constant
Average error		$1.40°$	$1.94°$
Standard deviation		$2.57°$	$2.31°$
Proportion of estimates with errors below:	$< 0.5°$	35.8%	14.1%
	$< 1°$	65.0%	39.7%
	$< 2°$	82.1%	70.5%
	$< 3°$	89.7%	83.4%
	$< 5°$	95.4%	92.8%
	$< 10°$	98.8%	98.6%

Table 1. Distribution of errors.

Statistics on the distribution of errors for the two motion models are given in table 1. Comparison with previously published results, table 2, shows that the algorithm presented here gives excellent accuracy, with the results for the affine motion model being substantially better than previously published methods. A more detailed evaluation can be found in [6].

6 Conclusions

We have presented a novel motion estimation algorithm, which is more accurate than other methods and still fast.

If one is willing to sacrifice some of the speed, the accuracy can be improved further. Using the same basic ideas, but replacing (16) by a simultaneous segmentation and motion estimation algorithm, we can achieve an average angular error on the Yosemite sequence of $1.14°$ with a standard

Technique	Average error	Standard deviation	Density
Lucas & Kanade [17]	2.80°	3.82°	35%
Uras *et al.* [19]	3.37°	3.37°	14.7%
Fleet & Jepson [7]	2.97°	5.76°	34.1%
Black & Anandan [4]	4.46°	4.21°	100%
Szeliski & Coughlan [18]	2.45°	3.05°	100%
Black & Jepson [5]	2.29°	2.25°	100%
Ju *et al.* [13]	2.16°	2.0°	100%
Karlholm [14]	2.06°	1.72°	100%
Lai & Vemuri [16]	1.99°	1.41°	100%
Bab-Hadiashar & Suter [1]	1.97°	1.96°	100%
constant motion	1.94°	2.31°	100%
constant motion	1.43°	1.24°	70%
affine motion	1.40°	2.57°	100%
affine motion	0.75°	0.73°	70%

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

deviation of $2.14°$. The details of this algorithm are beyond the scope of this paper, but can be found in [6].

Acknowledgments

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

References

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

[2] 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.

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

[4] 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.

[5] M. J. Black and A. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.