

Object Tracking based on the Orientation Tensor Concept

Jörgen Karlholm

Carl-Johan Westelius
Hans Knutsson

Carl-Fredrik Westin

ISSN 1400-3902
LiTH-ISY-R-1658
1994-09-02

Object tracking based on the orientation tensor concept

Jörgen Karlholm Carl-Johan Westelius Carl-Fredrik Westin
Hans Knutsson

Computer Vision Laboratory, Dept. of EE
Linköping University, S-581 83 Linköping, Sweden
email: `jorgen@isy.liu.se`

Abstract

We apply the 3D-orientation tensor representation to construct an object tracking algorithm. 2D-line normal flow is estimated by computing the eigenvector associated with the largest eigenvalue of 3D (two spatial dimensions plus time) tensors with a planar structure. Object's true 2D velocity is computed by averaging tensors with consistent normal flows, generating a 3D line representation that corresponds to a 2D point in motion. Flow induced by camera rotation is compensated for by ignoring points with velocity consistent with the ego-rotation. A region-of-interest growing process based on motion consistency generates estimates of object size and position.

1 Introduction

The literature on optical flow estimation is vast. Descriptions and performance studies of a number of different techniques are given in [3] and the monographs by Fleet [5] and Jähne [10]. We will only briefly describe the particular methods used in the present study. Details on the tensor field representation and filtering methods are found in [14, 15, 18, 19].

In the language of [3], the optical flow estimation method used is an *energy method*, see also [1, 7]. It is related to the gradient methods based on the motion-constraint equation [8, 17],

$$(\nabla f)^T \mathbf{v} = 0 \tag{1}$$

with ∇f denoting the spatio-temporal gradient $(f_x, f_y, f_t)^T$, and $\mathbf{v} = (u, v, 1)^T$, a 3D representation of the image velocity. Suppose we want to find the best least square estimate of \mathbf{v} , given gradient estimates from N points in a translating object, with w_i the weight given to estimate i . This gives us an equation

$$\mathbf{W} \mathbf{F} \mathbf{v} = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & & \\ \vdots & & \ddots & \\ 0 & & & w_N \end{pmatrix} \begin{pmatrix} \nabla f_1^T \\ \vdots \\ \nabla f_N^T \end{pmatrix} \mathbf{v} = \mathbf{0}$$

It is straightforward to show that the solution that minimises $\|\mathbf{W}\mathbf{F}\mathbf{v}\|_2$ is given by the eigenvector corresponding to the smallest eigenvalue of $\mathbf{G} = \sum_i w_i^2 \nabla f_i \nabla f_i^T$. We may interpret \mathbf{G} as a result of *averaging* local outer products $\mathbf{G}_i = \nabla f_i \nabla f_i^T$. See also [4], and compare this to the naive approach of averaging local optical flow estimates, Figure 1.

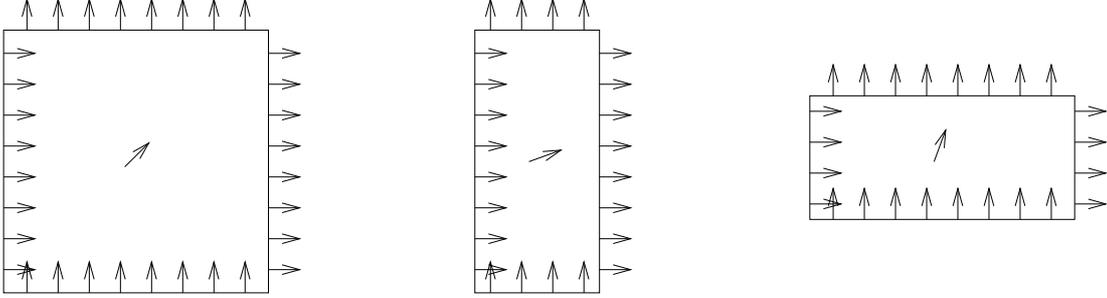


Figure 1: Naive averaging of local optical flow estimates. All three rectangles move with the same velocity, but the averaged flow vectors (inside rectangles) are different, only the one for the square pointing in the correct direction.

2 Local structure tensor

The gradient method discussed above uses estimates of the local structure of the 3D spatio-temporal space to extract optical flow. We use a related method, introduced by Knutsson [12, 13]. The main difference is the use of *quadrature* filters [11], rather than gradient estimation. Quadrature filters capture both first and second order variations, and this has the important consequence of allowing a structured and easily interpretable representation of the local spatio-temporal neighbourhood. In this method, the *dominant orientation* of a neighbourhood is represented as a dyadic product

$$\mathbf{T} \equiv A\hat{\mathbf{x}}\hat{\mathbf{x}}^T$$

where $A > 0$ is an arbitrary number and $\hat{\mathbf{x}}$ is a vector pointing in the direction of maximum signal variation.

A dyadic product accurately describes the local structure of a *simple* neighbourhood varying in just one direction $\hat{\mathbf{x}}$, so that if we denote the local coordinates by ξ ,

$$\mathbf{S}(\xi) = \mathbf{G}(\xi^T \mathbf{x})$$

where \mathbf{S} and \mathbf{G} are arbitrary signal functions.

In [13] it is shown that \mathbf{T} can be constructed by combining the outputs of polar separable quadrature filters, and [14] discusses an efficient implementation of this method, using 1D filters.

In neighbourhoods which are not simple, the estimated \mathbf{T} will not be a simple dyadic product, but have a more complex structure, being a sum of such products, and we refer

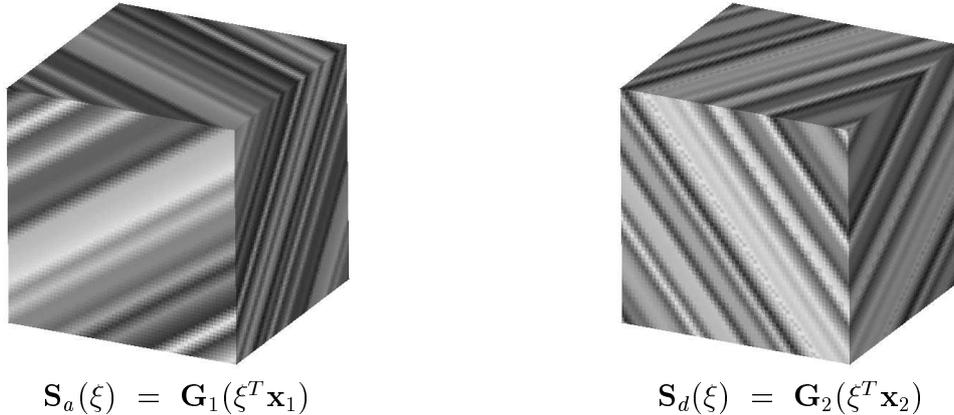


Figure 2: Two different three-dimensional simple neighbourhoods. The neighbourhoods are constructed using two different signal functions (\mathbf{G}_1 and \mathbf{G}_2) and two different signal orienting vectors (\mathbf{x}_1 and \mathbf{x}_2).

to it as the *local structure tensor*. Given a base $\{\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y, \hat{\mathbf{e}}_t\}$ we may obtain the eigenvalue decomposition of the corresponding matrix and, henceforth concentrating on the 3D case, write

$$\mathbf{T} = \lambda_1 \hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T + \lambda_2 \hat{\mathbf{e}}_2 \hat{\mathbf{e}}_2^T + \lambda_3 \hat{\mathbf{e}}_3 \hat{\mathbf{e}}_3^T$$

Letting $\lambda_1 \geq \lambda_2 \geq \lambda_3$, it is found that certain elementary but important local structures are revealed by means of an eigenvalue analysis of \mathbf{T} .

Plane case: A rank one or simple neighbourhood where $\lambda_1 \gg \lambda_2 \simeq \lambda_3 \geq 0$.

$$\mathbf{T} \simeq \lambda_1 \mathbf{T}_1 = \lambda_1 \hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T$$

This case corresponds to a neighbourhood that is approximately *planar*, i.e. is constant on planes in a given orientation. The orientation of the *normal vectors* to the planes is given by $\hat{\mathbf{e}}_1$.

Line case: A rank two neighbourhood where $\lambda_1 \simeq \lambda_2 \gg \lambda_3 \geq 0$.

$$\mathbf{T} \simeq \lambda_1 \mathbf{T}_2 = \lambda_1 (\hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T + \hat{\mathbf{e}}_2 \hat{\mathbf{e}}_2^T)$$

This case corresponds to a neighbourhood that is approximately constant on *lines*. The orientation of the lines is given by the eigenvector corresponding to the smallest eigenvalue, $\hat{\mathbf{e}}_3$.

Isotropic case: A rank three neighbourhood where $\lambda_1 \simeq \lambda_2 \simeq \lambda_3 \geq 0$.

$$\mathbf{T} \simeq \lambda_1 \mathbf{T}_3 = \lambda_1 (\hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T + \hat{\mathbf{e}}_2 \hat{\mathbf{e}}_2^T + \hat{\mathbf{e}}_3 \hat{\mathbf{e}}_3^T)$$

This case corresponds to an approximately *isotropic* neighbourhood, meaning that there exists energy in the neighbourhood but no dominant orientation, e.g. in the case of noise.

In general, \mathbf{T} will be a linear combination of these cases, i.e. \mathbf{T} can be expressed as:

$$\mathbf{T} = (\lambda_1 - \lambda_2) \mathbf{T}_1 + (\lambda_2 - \lambda_3) \mathbf{T}_2 + \lambda_3 \mathbf{T}_3 \quad (2)$$

where $(\lambda_1 - \lambda_2)$, $(\lambda_2 - \lambda_3)$ and λ_3 may be viewed as coordinates of \mathbf{T} in the tensor basis \mathbf{T}_i .

3 Motion estimation

It is now straightforward to obtain optical flow estimates from the tensor, if we interpret the 3D plane case as a moving 2D line/edge segment, and the 3D line case as a moving 2D point [4, 9]. In the 3D plane case we can only estimate the line’s *normal* flow (due to the “aperture problem”), but in the 3D line case the true flow is available.

Let $\mathbf{P}_{xy} = \hat{\mathbf{e}}_x \hat{\mathbf{e}}_x^T + \hat{\mathbf{e}}_y \hat{\mathbf{e}}_y^T$ be a projection operator onto the xy-plane. The velocities are then computed as follows:

$$\mathbf{v} = \begin{cases} -\frac{\hat{\mathbf{e}}_t^T \hat{\mathbf{e}}_1}{\|\mathbf{P}_{xy} \hat{\mathbf{e}}_1\|_2} \cdot \frac{\mathbf{P}_{xy} \hat{\mathbf{e}}_1}{\|\mathbf{P}_{xy} \hat{\mathbf{e}}_1\|_2} & \text{normal velocity, moving line case} \\ \frac{\mathbf{P}_{xy} \hat{\mathbf{e}}_3}{\hat{\mathbf{e}}_t^T \hat{\mathbf{e}}_3} & \text{true velocity, moving point case} \end{cases}$$

Here $\|\mathbf{x}\|_2 = \sqrt{\sum x_i^2}$, the Euclidean norm.

By averaging the tensor field over all points of a translating object we obtain a resultant tensor which ideally is of rank 2, and subsequently extract the true velocity from the eigenvector $\hat{\mathbf{e}}_3 = k(u, v, 1)^T$ corresponding to the smallest eigenvalue, in total analogy with the gradient method discussed above, see Figure 3.

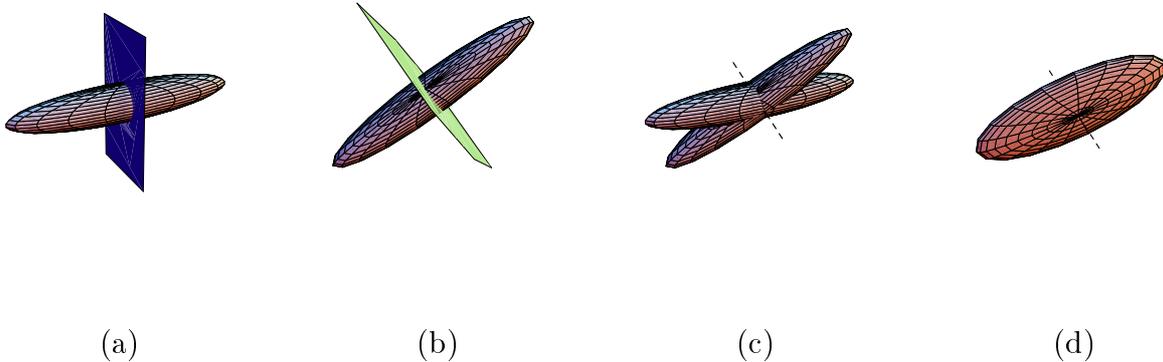


Figure 3: Tensor averaging. (a), (b): Two edges in common motion, creating planes in the 3D spatiotemporal space. Orientation tensors (ideally of rank 1) visualised as ellipsoids with eigenvectors forming principal axes. The eigenvector corresponding to the largest eigenvalue indicates orientation of plane. (c), (d): Averaging of tensors (‘symbolically’ shown in (c) gives as result an estimate of the true motion, now from the eigenvector corresponding to the *smallest* eigenvalue of the tensor (d), which ideally is of rank 2.

In general it is more robust to estimate normal flow than true flow [2, 6], and due to phase interference this is true also when using quadrature filters. We consequently do not want to use points outside line/edge segments. Since our method uses a rather costly eigenvalue decomposition we need a way to eliminate points which are not part of oriented

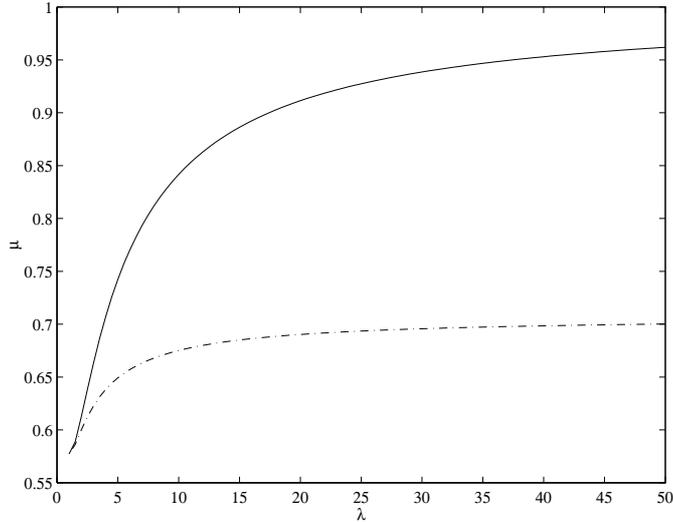


Figure 4: Estimating degree of anisotropy of local neighbourhood. *Solid line*: Plane-like anisotropy. *Dotted line*: Line-like anisotropy.

structures. A simple way to obtain an estimate of the degree of orientation is to compute

$$\mu = \frac{\|\mathbf{T}\|_F}{\text{Tr}(\mathbf{T})}$$

Here $\|\mathbf{T}\|_F = \sqrt{\sum_{i,j} \mathbf{T}_{ij}^2} = \sqrt{\sum_k \lambda_k^2}$, the Frobenius norm, and $\text{Tr}(\mathbf{T}) = \sum_k \mathbf{T}_{kk} = \sum_k \lambda_k$. It is easily seen that $1 \geq \mu \geq 1/\sqrt{3}$.

We perform the eigenvalue decomposition only at points where μ , the eigenvalue asymmetry, is greater than some threshold μ_0 . Figure 4 illustrates how μ varies with different degrees of neighbourhood anisotropy. The solid line shows $\mu = \sqrt{\lambda^2 + 1 + 1} / (\lambda + 1 + 1)$, plane-like anisotropy. The dotted line shows $\mu = \sqrt{2\lambda^2 + 1} / (2\lambda + 1)$, line-like anisotropy.

The anisotropy estimation also gives us a means to get rid of low-energy noise. The signal energy is proportional to the Frobenius norm of the tensor. Instead of thresholding separately on the tensor norm, we add a small amount of isotropic energy

$$\tilde{\mathbf{T}} = \mathbf{T} + \alpha \mathbf{I}$$

which, followed by the asymmetry thresholding, quenches low-energy asymmetries. The constant α is chosen as a small fraction of the largest tensor element in a frame, which is taken as an estimate of the maximum signal energy.

4 Object tracking

The motion estimation algorithm has been employed in a system for object tracking, implemented on a simulated robot with a movable camera head. In the actual implementation,

the goal of the tracking is to keep the moving object in the centre of the field of view. We do this by detecting the centre of gravity (CoG) of the object’s motion field, and placing a region-of-interest (ROI) at that point. If the CoG deviates too much from the centre of the camera image, a saccade is generated. Consequently tracking is achieved through a combination of smooth pursuit and saccades.

Once a candidate for the object’s CoG has been generated (e.g. by a preattentive motion detection system), we iteratively increase the radius of the ROI until the motion estimate within the ROI becomes inconsistent with the hypothesis of a single translating object. This is done by means of an eigenvalue analysis of the averaged tensor. As mentioned above, averaging of tensors generated from a single translating object will ideally produce a rank 2 resultant tensor. If the averaging is done over points with multiple velocities, the resultant tensor will be of rank 3. Consequently we introduce a new threshold μ_1 such that if $\lambda_3 / (\lambda_1 + \lambda_2) > \mu_1$, the ROI radius is judged to be too large.

The isotropy thresholding does not work when the estimates have been obtained from a region with a single dominant orientation (aperture problem), since in this case the tensor will never become isotropic, no matter how many velocities are blended. Also, when there is just one orientation, we do not obtain the correct velocity from the eigenvector corresponding to the smallest eigenvalue. The best we can do is to compute the *normal* velocity from the dominant eigenvector. It is simple to detect the aperture problem; if there is a dominant orientation and a single velocity, the resultant averaged tensor has a dominant eigenvalue, i.e. $(\lambda_2 + \lambda_3) / \lambda_1 \ll 1$. If there is a dominant orientation, but multiple velocities, the tensor will pass the isotropy test ($\lambda_3 / (\lambda_1 + \lambda_2) \ll 1$), but the eigenvector \hat{e}_3 corresponding to λ_3 will lie in the xy-plane and give rise to an absurd velocity, so this case is easily detectable.

Having extracted the “true” velocity of the object, this is converted into camera joint velocities in a simple control algorithm. In subsequent iterations processing is restricted to the ROI, whose size and position are continually updated using the consistency approach. Figure 5 illustrates the extraction of valid data points.

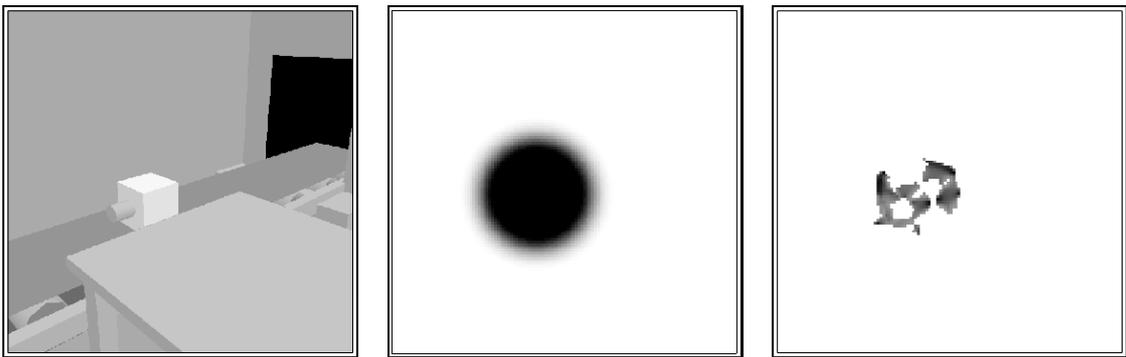


Figure 5: Conveyor belt tracking scene. *Left*: Current data input. A box is transported on a conveyor belt. *Middle*: Gaussian region-of-interest grown around object. *Right*: Points used in tensor averaging.

5 Increasing robustness of tracking

The algorithm as given above is very fragile. If the tracked object has a shape such that the CoG is outside its boundary, or if the camera rotation fails to compensate for the object motion, again leading to a situation with the predicted CoG outside the object boundary, the ROI growing process may stick to camera-induced motion of the background or to some other object moving by. It turns out that there is an elegant way of getting around these problems.

Knowing the camera rotation velocity and the geometry of the camera we may compute an approximation of the camera-induced motion field. In the 3D spatiotemporal space this motion is represented by a vector $\mathbf{v}_{ego} = (u_{ego}, v_{ego}, 1)^T$. We note that this vector will lie in the 3D planes generated by all linear structures moving with this velocity. This gives us a means to eliminate all self-induced motion by taking the scalar product between \mathbf{v}_{ego} and the normal vectors of the planes, and neglecting points with too small scalar product. The normal vectors are identical with the eigenvector $\hat{\mathbf{e}}_1$ corresponding to the *largest* eigenvalue of the tensor. Figure 6 (*Left*) shows that the disallowed planes will lie inside a cone centred at \mathbf{v}_{ego} . If the camera geometry is unknown, it may still be possible to compensate for ego-rotation by averaging tensors over the whole image. This would give an estimate of self-induced rotation if the static background dominates the image.

By monitoring the quality of the tracking, storing a moving average of the most recent velocity corrections (or “retinal slip”), we may in fact eliminate even more of the signal, by letting through only points with a velocity within a window around $(0, 0, 1)^T$, the window size being proportional to the average slip. Of course there must be a minimum attainable size of the window to manage sudden accelerations, the size being inversely proportional to the sampling rate, see Figure 6 (*Right*).

6 Conclusion and Extensions

A new tensor-based tracking algorithm has been presented. The algorithm uses estimates of line/edge motion to grow a region-of-interest around positions with consistent motion. The true optical flow is extracted by means of a tensor averaging procedure and subsequent eigenvalue decomposition. A procedure for ego-rotation compensation and adaptive elimination of irrelevant data has been described. The algorithm has been implemented and tested in a robot simulation environment, Figures 7 and 8.

A straightforward improvement of tracking may be achieved by including depth information obtained from disparity measurements using a stereo camera head. This enables us to restrict the ROI to points at a specific depth. In fact, it makes it possible to grow a 3D ROI around the object.

The present implementation of the tracker follows a single object and keeps it centred in the field of view. It should, however, be possible to generalise this to tracking of multiple objects. We simply attach a ROI to each object and make the same computations as above within each ROI, updating the velocities of the ROIs instead of the camera joints. Using a



Figure 6: Eliminating irrelevant motion. *Left*: Cone centred around vector representing camera-induced motion. The line segment (which becomes a plane in the spatiotemporal space) moves with respect to a static background, since it does not intersect with the cone. *Right*: Retinal slip cone centred at $(0, 0, 1)^T$. Opening angle represents the current quality of tracking.

filtering system that detects local motions not covered by ROI's, we may spawn new ROIs as new objects appear, and delete others as objects disappear.

A significant detail that we have not mentioned, is related to spatiotemporal filtering in general, namely the problem with *edge-effects*. When the camera moves around, new objects will enter the field of view. Due to the temporal buffer depth of the filtering system the local structure estimates will not be correct until the object has been visible for a while. Analogously there will be problems with saccades, since the temporal buffers will contain sequences from two unrelated spatial positions. A simple remedy for the saccade problem is to neglect all data until the old data has been shifted out of the buffers (as in frame 5 of Figure 8), but this may cause problems for the tracking controller. A systematic way to get around edge-effects is to use *normalised convolution* [16, 18], which is a technique where *certainty* labels are attached to all extracted data and least-square sense optimal filtering estimates are computed based on these. When a saccade has been produced the data from the “pre-saccadic” gaze direction is given zero certainty (or validity), and in the edge case we set zero certainty to data outside the image boundary, as opposed to the usual procedure of putting a frame with zeros (or any other value) around the image. Normalised convolution can be used whenever the camera is accelerating fast, saccades being just a special case. In all such situations the local linear structure approximation has a short temporal validity and old data is useless and should be discarded. The result

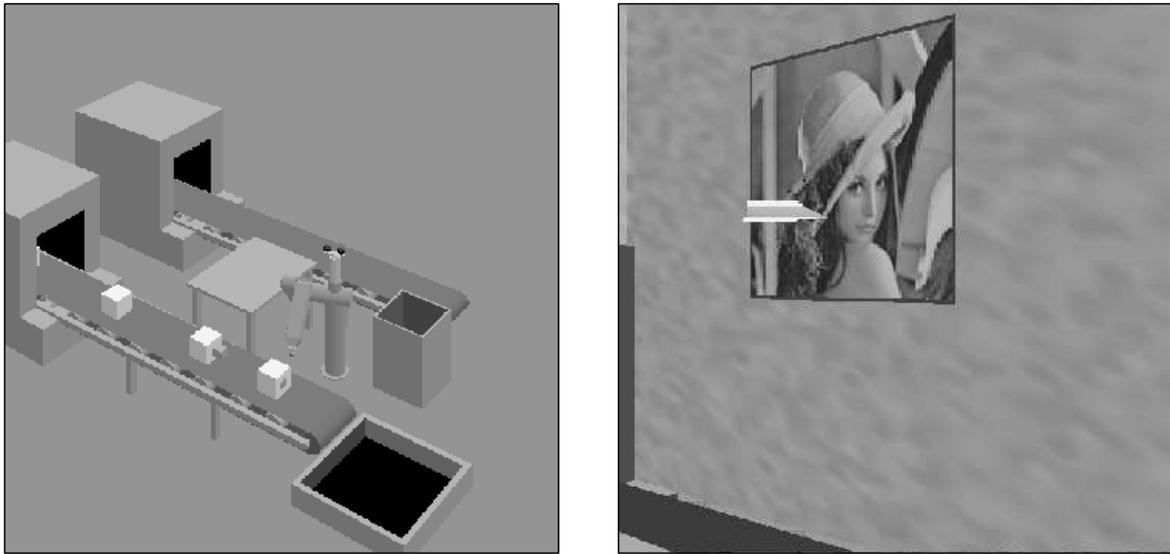


Figure 7: Applications of tracking algorithm. *Left:* Conveyor belt tracking and reaching. *Right:* “Aeroplane” tracked against a textured background.

of acceleration can be observed in Figure 8, frame 4, where the ego-rotation compensation does not quite eliminate all background motion.

7 Acknowledgement

This work has been done within the ESPRIT project “Vision as Process” BRA 7108.

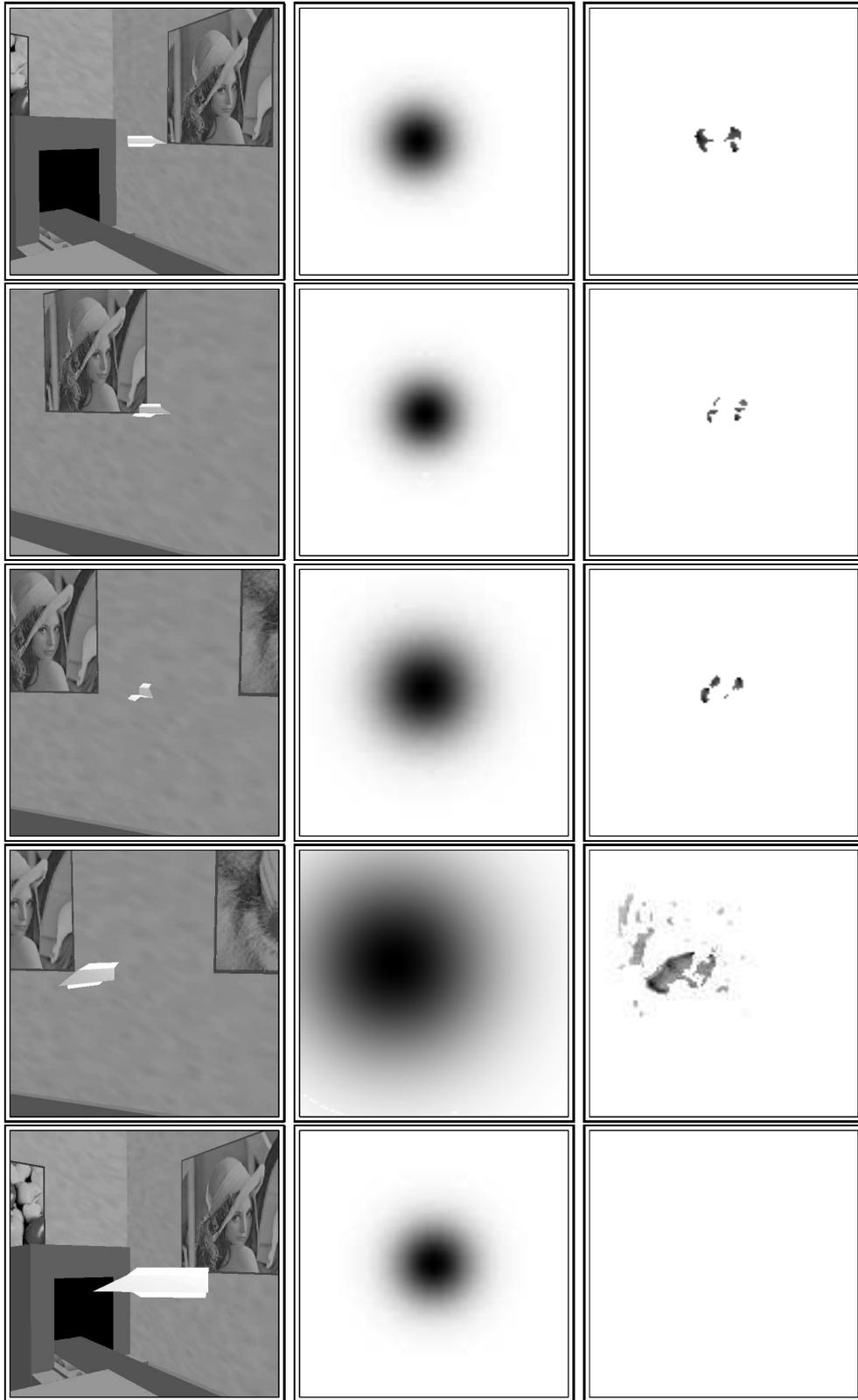


Figure 8: Fragments of tracking test sequence showing an “aeroplane” moving in a circular orbit with constant angular velocity, $\dot{\phi} = 1.0$ rad/s. Orbit radius $r = 1.0$ m, and sampling rate $f_s = 25$ frames/s. The columns show current image, Gaussian region-of-interest and points used in tensor averaging, respectively. Frame 5 shows the result of a saccadic movement to recentre the object. After a saccade, no tensor data is extracted until old frames have been shifted out of the temporal buffer.

References

- [1] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Jour. of the Opt. Soc. of America*, 2:284–299, 1985.
- [2] J. Aloimonos. Purposive and qualitative active vision. In *DARPA Image Understanding Workshop*, Philadelphia, Penn., USA, September 1990.
- [3] J.L. Barron, D. J. Fleet, S. S. Beauchemin, and T. A. Burkitt. Performance of optical flow techniques. In *Proc. of the CVPR*, pages 236–242, Champaign, Illinois, USA, 1992. IEEE. Revised report July 1993, TR-299, Dept. of Computer Science, University of Western Ontario, London, Ontario, Canada N6A 5B7.
- [4] J. Bigün, G. H. Granlund, and J. Wiklund. Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):775–790, August 1991. Report LiTH-ISY-I-1148, Linköping University, Sweden, 1990.
- [5] D. J. Fleet. *Measurement of image velocity*. Kluwer Academic Publishers, 1992. ISBN 0-7923-9198-5.
- [6] L. Haglund. *Adaptive Multidimensional Filtering*. PhD thesis, Linköping University, Sweden, S-581 83 Linköping, Sweden, October 1992. Dissertation No 284, ISBN 91-7870-988-1.
- [7] D. J. Heeger. Optical Flow Using Spatio-Temporal Filters. *Int. Journal of Computer Vision*, 2(1):279–302, 1988.
- [8] B. K. P. Horn and B.G. Schunk. Determining optical flow. *AI*, pages 185–204, 1981.
- [9] B. Jähne. Motion determination in space-time images. In O. Faugeras, editor, *Computer Vision-ECCV90*, pages 161–173. Springer-Verlag, 1990.
- [10] B. Jähne. *Digital Image Processing: Concepts, Algorithms and Scientific Applications*. Springer Verlag, Berlin, Heidelberg, 1991.
- [11] H. Knutsson. *Filtering and Reconstruction in Image Processing*. PhD thesis, Linköping University, Sweden, 1982. Diss. No. 88.
- [12] H. Knutsson. Producing a continuous and distance preserving 5-D vector representation of 3-D orientation. In *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management - CAPAIDM*, pages 175–182, Miami Beach, Florida, November 1985. IEEE. Report LiTH-ISY-I-0843, Linköping University, Sweden, 1986.

- [13] 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.
- [14] H. Knutsson and M.T. Andersson. N-dimensional orientation estimation using quadrature filters and tensor whitening. In *Proceedings of IEEE International Conference on Acoustics, Speech, & Signal Processing*, Adelaide, Australia, April 1994. IEEE.
- [15] H. Knutsson and H. Bårman. Robust orientation estimation in 2D, 3D and 4D using tensors. In *Proceedings of International Conference on Automation, Robotics and Computer Vision*, September 1992.
- [16] H. Knutsson and C-F Westin. Normalized and differential convolution: Methods for interpolation and filtering of incomplete and uncertain data. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York City, USA, June 1993. IEEE.
- [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] C-F Westin. *A Tensor Framework for Multi-dimensional Signal Processing*. PhD thesis, Linköping University, Sweden, S-581 83 Linköping, Sweden, 1994.
- [19] C-F Westin and H. Knutsson. Estimation of Motion Vector Fields using Tensor Field Filtering. In *Proceedings of IEEE International Conference on Image Processing*, Austin, Texas, November 1994. IEEE.