

# Two-Frame Motion Estimation Based on Polynomial Expansion

Gunnar Farneback

Computer Vision Laboratory, Linköping University,  
SE-581 83 Linköping, Sweden

gf@isy.liu.se

<http://www.isy.liu.se/cvl/>

**Abstract.** This paper presents a novel two-frame motion estimation algorithm. The first step is to approximate each neighborhood of both frames by quadratic polynomials, which can be done efficiently using the polynomial expansion transform. From observing how an exact polynomial transforms under translation a method to **estimate displacement fields from the polynomial expansion coefficients** is derived and after a series of refinements leads to a **robust algorithm**. Evaluation on the Yosemite sequence shows good results.

## 1 Introduction

In previous work we have developed **orientation tensor based algorithms** to estimate motion, with excellent results both with respect to accuracy and speed [1, 2]. A **limitation** of those, however, is that the estimation of the **spatiotemporal orientation tensors** requires the motion field to be temporally consistent. This is often the case but turned out to be a problem in the **WITAS project** [3], where **image sequences are obtained by a helicopter-mounted camera**. Due to **high frequency vibrations from the helicopter affecting the camera system**, there are **large, quickly varying, and difficult to predict displacements between successive frames**.

A natural solution is to **estimate the motion, or displacement, field from only two frames and try to compensate for the background motion**. This paper presents a novel method to estimate displacement. It is related to our orientation tensor methods in that the first processing step, a signal transform called polynomial expansion, is common. Naturally this is only done **spatially** now, **instead of spatiotemporally**. Another common theme is the inclusion of parametric motion models in the algorithms.

## 2 Polynomial Expansion

The idea of polynomial expansion is to approximate some neighborhood of each pixel with a polynomial. Here we are only interested in quadratic polynomials, giving us the local signal model, expressed in a local coordinate system,

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

where  $\mathbf{A}$  is a symmetric matrix,  $\mathbf{b}$  a vector and  $c$  a scalar. The coefficients are estimated from a weighted least squares fit to the signal values in the neighborhood. The weighting has two components called certainty and applicability. These terms are the same as in normalized convolution [4–6], which polynomial expansion is based on. The certainty is coupled to the signal values in the neighborhood. For example it is generally a good idea to set the certainty to zero outside the image. Then neighborhood points outside the image have no impact on the coefficient estimation. The applicability determines the relative weight of points in the neighborhood based on their position in the neighborhood. Typically one wants to give most weight to the center point and let the weights decrease radially. The width of the applicability determines the scale of the structures which will be captured by the expansion coefficients.

While this may sound computationally very demanding it turns out that it can be implemented efficiently by a hierarchical scheme of separable convolutions. Further details on this can be found in [6].

### 3 Displacement Estimation

Since the result of polynomial expansion is that each neighborhood is approximated by a polynomial, we start by analyzing what happens if a polynomial undergoes an ideal translation. Consider the exact quadratic polynomial

$$f_1(\mathbf{x}) = \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1^T \mathbf{x} + c_1 \quad (2)$$

and construct a new signal  $f_2$  by a global displacement by  $\mathbf{d}$ ,

$$\begin{aligned} f_2(\mathbf{x}) &= f_1(\mathbf{x} - \mathbf{d}) = (\mathbf{x} - \mathbf{d})^T \mathbf{A}_1 (\mathbf{x} - \mathbf{d}) + \mathbf{b}_1^T (\mathbf{x} - \mathbf{d}) + c_1 \\ &= \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + (\mathbf{b}_1 - 2\mathbf{A}_1 \mathbf{d})^T \mathbf{x} + \mathbf{d}^T \mathbf{A}_1 \mathbf{d} - \mathbf{b}_1^T \mathbf{d} + c_1 \\ &= \mathbf{x}^T \mathbf{A}_2 \mathbf{x} + \mathbf{b}_2^T \mathbf{x} + c_2. \end{aligned} \quad (3)$$

Equating the coefficients in the quadratic polynomials yields

$$\mathbf{A}_2 = \mathbf{A}_1, \quad (4)$$

$$\mathbf{b}_2 = \mathbf{b}_1 - 2\mathbf{A}_1 \mathbf{d}, \quad (5)$$

$$c_2 = \mathbf{d}^T \mathbf{A}_1 \mathbf{d} - \mathbf{b}_1^T \mathbf{d} + c_1. \quad (6)$$

The key observation is that by equation (5) we can solve for the translation  $\mathbf{d}$ , at least if  $\mathbf{A}_1$  is non-singular,

$$2\mathbf{A}_1 \mathbf{d} = -(\mathbf{b}_2 - \mathbf{b}_1), \quad (7)$$

$$\mathbf{d} = -\frac{1}{2} \mathbf{A}_1^{-1} (\mathbf{b}_2 - \mathbf{b}_1). \quad (8)$$

We note that this observation holds for any signal dimensionality.

### 3.1 Practical Considerations

Obviously the assumptions about an entire signal being a single polynomial and a global translation relating the two signals are quite unrealistic. Still the basic relation (7) can be used for real signals, although errors are introduced when the assumptions are relaxed. The question is whether these errors can be kept small enough to give a useful algorithm.

To begin with we replace the global polynomial in equation (2) with local polynomial approximations. Thus we start by doing a polynomial expansion of both images, giving us expansion coefficients  $\mathbf{A}_1(\mathbf{x})$ ,  $\mathbf{b}_1(\mathbf{x})$ , and  $c_1(\mathbf{x})$  for the first image and  $\mathbf{A}_2(\mathbf{x})$ ,  $\mathbf{b}_2(\mathbf{x})$ , and  $c_2(\mathbf{x})$  for the second image. Ideally this should give  $\mathbf{A}_1 = \mathbf{A}_2$  according to equation (4) but in practice we have to settle for the approximation

$$\mathbf{A}(\mathbf{x}) = \frac{\mathbf{A}_1(\mathbf{x}) + \mathbf{A}_2(\mathbf{x})}{2}. \quad (9)$$

We also introduce

$$\Delta \mathbf{b}(\mathbf{x}) = -\frac{1}{2}(\mathbf{b}_2(\mathbf{x}) - \mathbf{b}_1(\mathbf{x})) \quad (10)$$

to obtain the primary constraint

$$\mathbf{A}(\mathbf{x})\mathbf{d}(\mathbf{x}) = \Delta \mathbf{b}(\mathbf{x}), \quad (11)$$

where  $\mathbf{d}(\mathbf{x})$  indicates that we have also replaced the global displacement in equation (3) with a spatially varying displacement field.

### 3.2 Estimation Over a Neighborhood

In principle equation (11) can be solved pointwise, but the results turn out to be too noisy. Instead we make the assumption that the displacement field is only slowly varying, so that we can integrate information over a neighborhood of each pixel. Thus we try to find  $\mathbf{d}(\mathbf{x})$  satisfying (11) as well as possible over a neighborhood  $I$  of  $\mathbf{x}$ , or more formally minimizing

$$\sum_{\Delta \mathbf{x} \in I} w(\Delta \mathbf{x}) \|\mathbf{A}(\mathbf{x} + \Delta \mathbf{x})\mathbf{d}(\mathbf{x}) - \Delta \mathbf{b}(\mathbf{x} + \Delta \mathbf{x})\|^2, \quad (12)$$

where we let  $w(\Delta \mathbf{x})$  be a weight function for the points in the neighborhood. The minimum is obtained for

$$\mathbf{d}(\mathbf{x}) = \left( \sum w \mathbf{A}^T \mathbf{A} \right)^{-1} \sum w \mathbf{A}^T \Delta \mathbf{b}, \quad (13)$$

where we have dropped some indexing to make the expression more readable. The minimum value is given by

$$e(\mathbf{x}) = \left( \sum w \Delta \mathbf{b}^T \Delta \mathbf{b} \right) - \mathbf{d}(\mathbf{x})^T \sum w \mathbf{A}^T \Delta \mathbf{b}. \quad (14)$$

In practical terms this means that we compute  $\mathbf{A}^T \mathbf{A}$ ,  $\mathbf{A}^T \Delta \mathbf{b}$ , and  $\Delta \mathbf{b}^T \Delta \mathbf{b}$  pointwise and average these with  $w$  before we solve for the displacement. The

minimum value  $e(\mathbf{x})$  can be used as a reversed confidence value, with small numbers indicating high confidence. The solution given by (13) exists and is unique unless the whole neighborhood is exposed to the aperture problem.

Sometimes it is useful to add a certainty weight  $c(\mathbf{x} + \Delta\mathbf{x})$  to (12). This is most easily handled by scaling  $\mathbf{A}$  and  $\Delta\mathbf{b}$  accordingly.

### 3.3 Parameterized Displacement Fields

We can improve robustness if the displacement field can be parameterized according to some motion model. This is straightforward for motion models which are linear in their parameters, like the affine motion model or the eight parameter model. We derive this for the eight parameter model in 2D,

$$\begin{aligned} d_x(x, y) &= a_1 + a_2x + a_3y + a_7x^2 + a_8xy, \\ d_y(x, y) &= a_4 + a_5x + a_6y + a_7xy + a_8y^2. \end{aligned} \quad (15)$$

We can rewrite this as

$$\mathbf{d} = \mathbf{S}\mathbf{p}, \quad (16)$$

$$\mathbf{S} = \begin{pmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{pmatrix}, \quad (17)$$

$$\mathbf{p} = (a_1 \ a_2 \ a_3 \ a_4 \ a_5 \ a_6 \ a_7 \ a_8)^T. \quad (18)$$

Inserting into (12) we obtain the weighted least squares problem

$$\sum_i w_i \|\mathbf{A}_i \mathbf{S}_i \mathbf{p} - \Delta\mathbf{b}_i\|^2, \quad (19)$$

where we use  $i$  to index the coordinates in a neighborhood. The solution is

$$\mathbf{p} = \left( \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{S}_i \right)^{-1} \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \Delta\mathbf{b}_i. \quad (20)$$

We notice that like before we can compute  $\mathbf{S}^T \mathbf{A}^T \mathbf{A} \mathbf{S}$  and  $\mathbf{S}^T \mathbf{A}^T \Delta\mathbf{b}$  pointwise and then average these with  $w$ . Naturally (20) reduces to (13) for the constant motion model.

### 3.4 Incorporating A Priori Knowledge

A principal problem with the method so far is that we assume that the local polynomials at the same coordinates in the two signals are identical except for a displacement. Since the polynomial expansions are local models these will vary spatially, introducing errors in the constraints (11). For small displacements this is not too serious, but with larger displacements the problem increases. Fortunately we are not restricted to comparing two polynomials at the same coordinate. If we have a priori knowledge about the displacement field, we can compare

the polynomial at  $\mathbf{x}$  in the first signal to the polynomial at  $\mathbf{x} + \tilde{\mathbf{d}}(\mathbf{x})$  in the second signal, where  $\tilde{\mathbf{d}}(\mathbf{x})$  is the a priori displacement field rounded to integer values. Then we effectively only need to estimate the relative displacement between the real value and the rounded a priori estimate, which hopefully is smaller.

This observation is included in the algorithm by replacing equations (9) and (10) by

$$\mathbf{A}(\mathbf{x}) = \frac{\mathbf{A}_1(\mathbf{x}) + \mathbf{A}_2(\tilde{\mathbf{x}})}{2}, \quad (21)$$

$$\Delta \mathbf{b}(\mathbf{x}) = -\frac{1}{2}(\mathbf{b}_2(\tilde{\mathbf{x}}) - \mathbf{b}_1(\mathbf{x})) + \mathbf{A}(\mathbf{x})\tilde{\mathbf{d}}(\mathbf{x}), \quad (22)$$

where

$$\tilde{\mathbf{x}} = \mathbf{x} + \tilde{\mathbf{d}}(\mathbf{x}). \quad (23)$$

The first two terms in  $\Delta \mathbf{b}$  are involved in computing the remaining displacement while the last term adds back the rounded a priori displacement. We can see that for  $\tilde{\mathbf{d}}$  identically zero, these equations revert to (9) and (10), as would be expected.

### 3.5 Iterative and Multi-scale Displacement Estimation

A consequence of the inclusion of an a priori displacement field in the algorithm is that we can close the loop and iterate. A better a priori estimate means a smaller relative displacement, which in turn improves the chances for a good displacement estimate. We consider two different approaches, iterative displacement estimation and multi-scale displacement estimation.

In both the approaches we iterate with the estimated displacements from one step used as a priori displacement in the next step. The a priori displacement field in the first step would usually be set to zero, unless actual knowledge about it is available.

In the first approach the same polynomial expansion coefficients are used in all iterations and need only be computed once. The weak spot of this approach is in the first iteration. If the displacements (relative the a priori displacements) are too large, the output displacements cannot be expected to be improvements and iterating will be meaningless.

The problem of too large displacements can be reduced by doing the analysis at a coarser scale. This means that we use a larger applicability for the polynomial expansion and/or lowpass filter the signal first. The effect is that the estimation algorithm can handle larger displacements but at the same time the accuracy decreases.

This observation points to the second approach with multiple scales. Start at a coarse scale to get a rough but reasonable displacement estimate and propagate this through finer scales to obtain increasingly more accurate estimates. A drawback is that we need to recompute the polynomial expansion coefficients for each scale, but this cost can be reduced by subsampling between scales.

## 4 Experimental Results

The algorithm has been implemented in Matlab, with certain parts in the form of C mex files. Source code for the implementation is available from <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 [7], figure 1. 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,  $\arccos(\hat{\mathbf{v}}_{\text{est}}^T \hat{\mathbf{v}}_{\text{true}})$  [8]. 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.

We have estimated the displacement from the center frame and the frame before. The averaging over neighborhoods is done using a  $39 \times 39$  Gaussian weighting function ( $w$  in equation (19)) with standard deviation 6. The polynomial expansion is done with an  $11 \times 11$  Gaussian applicability with standard deviation 1.5. In order to reduce the errors near the borders, the polynomial expansions have been computed with certainty set to zero off the border. Additionally pixels close to the borders have been given a reduced weight (see section 3.2) because the expansion coefficients still can be assumed to be less reliable there. The constant and affine motion models have been used with a single iteration and with three iterations at the same scale.

The results and a comparison with other methods can be found in table 1. Clearly this algorithm cannot compete with the most accurate ones, but that is to be expected since those take advantage of the spatio-temporal consistency over several frames. Still these results are good for a two-frame algorithm. A more thorough evaluation of the algorithm can be found in [6].

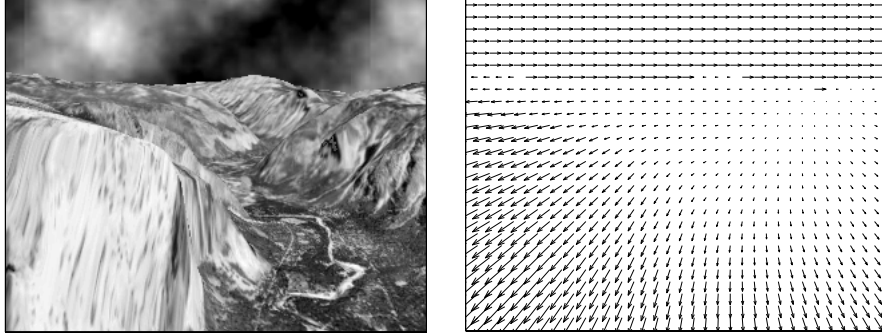
The main weakness of the algorithm is the assumption of a slowly varying displacement field, causing discontinuities to be smoothed out. This can be solved by combining the algorithm with a simultaneous segmentation procedure, e.g. the one used in [2].

## Acknowledgements

The work presented in this paper was supported by WITAS, the Wallenberg laboratory on Information Technology and Autonomous Systems, which is gratefully acknowledged.

## References

1. Farnebäck, G.: Fast and Accurate Motion Estimation using Orientation Tensors and Parametric Motion Models. In: Proceedings of 15th International Conference on Pattern Recognition. Volume 1., Barcelona, Spain, IAPR (2000) 135–139



**Fig. 1.** One frame of the Yosemite sequence and the corresponding true velocity field (subsampled).

**Table 1.** Comparison with other methods, Yosemite sequence. The sky region is excluded for all results.

Technique	Average error	Standard deviation	Density
Lucas & Kanade [9]	2.80°	3.82°	35%
Uras <i>et al.</i> [10]	3.37°	3.37°	14.7%
Fleet & Jepson [11]	2.97°	5.76°	34.1%
Black & Anandan [12]	4.46°	4.21°	100%
Szeliski & Coughlan [13]	2.45°	3.05°	100%
Black & Jepson [14]	2.29°	2.25°	100%
Ju <i>et al.</i> [15]	2.16°	2.0°	100%
Karlholm [16]	2.06°	1.72°	100%
Lai & Vemuri [17]	1.99°	1.41°	100%
Bab-Hadiashar & Suter [18]	1.97°	1.96°	100%
Mémin & Pérez [19]	1.58°	1.21°	100%
Farnebäck, constant motion [1, 6]	1.94°	2.31°	100%
Farnebäck, affine motion [1, 6]	1.40°	2.57°	100%
Farnebäck, segmentation [2, 6]	1.14°	2.14°	100%
Constant motion, 1 iteration	3.94°	4.23°	100%
Constant motion, 3 iterations	2.60°	2.27°	100%
Affine motion, 1 iteration	4.19°	6.76°	100%
Affine motion, 3 iterations	2.08°	2.45°	100%

2. Farnebäck, G.: Very High Accuracy Velocity Estimation using Orientation Tensors, Parametric Motion, and Simultaneous Segmentation of the Motion Field. In: Proceedings of the Eighth IEEE International Conference on Computer Vision. Volume I., Vancouver, Canada (2001) 171–177
3. URL: <http://www.ida.liu.se/ext/witas/>.
4. Knutsson, H., Westin, C.F.: 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, IEEE (1993) 515–523
5. Westin, C.F.: 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.
6. Farnebäck, G.: Polynomial Expansion for Orientation and Motion Estimation. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden (2002) Dissertation No 790, ISBN 91-7373-475-6.
7. Heeger, D.J.: Model for the extraction of image flow. *J. Opt. Soc. Am. A* **4** (1987) 1455–1471
8. Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Performance of optical flow techniques. *Int. J. of Computer Vision* **12** (1994) 43–77
9. Lucas, B., Kanade, T.: An Iterative Image Registration Technique with Applications to Stereo Vision. In: Proc. Darpa IU Workshop. (1981) 121–130
10. Uras, S., Girosi, F., Verri, A., Torre, V.: A computational approach to motion perception. *Biological Cybernetics* (1988) 79–97
11. Fleet, D.J., Jepson, A.D.: Computation of Component Image Velocity from Local Phase Information. *Int. Journal of Computer Vision* **5** (1990) 77–104
12. Black, M.J., Anandan, P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding* **63** (1996) 75–104
13. Szeliski, R., Coughlan, J.: Hierarchical spline-based image registration. In: Proc. IEEE Conference on Computer Vision Pattern Recognition, Seattle, Washington (1994) 194–201
14. Black, M.J., Jepson, A.: Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18** (1996) 972–986
15. Ju, S.X., Black, M.J., Jepson, A.D.: Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In: Proceedings CVPR’96, IEEE (1996) 307–314
16. Karlholm, J.: 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.
17. Lai, S.H., Vemuri, B.C.: Reliable and efficient computation of optical flow. *International Journal of Computer Vision* **29** (1998) 87–105
18. Bab-Hadiashar, A., Suter, D.: Robust optic flow computation. *International Journal of Computer Vision* **29** (1998) 59–77
19. Mémin, E., Pérez, P.: Hierarchical estimation and segmentation of dense motion fields. *International Journal of Computer Vision* **46** (2002) 129–155