

A Direct Geometric Algebra Optical Flow Solution

Lee Streeter

University of Waikato, Department of Physics and Electronic Engineering, Hamilton

Email: lvs2@phys.waikato.ac.nz

Abstract

Optical flow is the vector inverse problem of estimation motion through an image sequence. Geometric Algebra is an appropriate mathematical language for describing and solving vector problems. In this paper we apply Geometric Algebra to optical flow and pose a direct solution using a simple smoothness constraint. Implementational considerations are given and comparative assessment is made to two seminal optical flow methods, namely the Horn-Schunck and Lucas-Kanade algorithms. The new method is demonstrated to be fast and to give results comparable to these seminal methods.

Keywords: Geometric Algebra, Optical Flow

1 Introduction

Optical flow is the inverse problem of estimating motion in an image sequence without tracking object. An image sequence of time varying brightness can be described by

$$I(x(t), t) = f(t)$$

where $x(t)$ describes motion through I . Taking the derivative with respect to t gives

$$\nabla I \cdot \mathbf{u} + I_t = f_t(t)$$

where (using the partial derivative notation $y_x = \frac{\partial y}{\partial x}$) $\mathbf{u} = \mathbf{x}_t$ is the two-vector of motion flow or optical flow through I . Typically (but not always) $f(t)$ is assumed constant with time from which we get the “classical” optical flow constraint equation

$$\nabla I \cdot \mathbf{u} = -I_t \quad (1)$$

Finding \mathbf{u} involves finding an appropriate inversion to equation 1 which involves two unknowns in one equation, leading to the aperture problem, having two parts:

1. The optical flow constraint equation only allows a direct solution along the direction of image gradient
2. An indirect solution can be found by applying an ‘aperture function’ (small window) and finding an local least squares solution, but with a tradeoff between localisation and accuracy.

The least squares solution is known as the Lucas-Kanade optical flow algorithm [1], where the constraint is posed as a matrix problem with \mathbf{u} the two

element flow vector, \mathbf{I}_t the $N \times 1$ time difference vector and $\nabla \mathbf{I}$ the $N \times 2$ matrix of image gradient values of N image points within the aperture region

$$\nabla \mathbf{I} \mathbf{u} = -\mathbf{I}_t$$

which by applying the psudeo inverse gives

$$\mathbf{u} = -((\nabla \mathbf{I})^T \nabla \mathbf{I})^{-1} (\nabla \mathbf{I})^T \mathbf{I}_t$$

Often a weighting function is applied to $\nabla \mathbf{I}$ and \mathbf{I}_t so that data closer to the point of interest are given greater weighting. In the Lucas-Kanade algorithm the aperture problem is accepted and the size of the aperture is selected to give the best tradeoff between localisation and accuracy given the particular image sequence under examination.

The Horn-Schunck algorithm [2] applies the Laplacian operator as a smoothness constraint and combines this with the optical flow constraint equation by using a Lagrange multiplier to be solved by calculus of variations, viz,

$$E = \iint (\nabla I \cdot \mathbf{u} - I_t + \lambda \nabla^2 (u_x + u_y)) dx dy$$

where u_x and u_y are the components of \mathbf{u} . The Lagrange multiplier λ controls the degree of smoothness enforced by the energy functional E which is minimised. λ has much the same influence as the window function in the Lucas-Kanade algorithm, essentially increasing the size of the aperture of influence on \mathbf{u} , apparent in the over-smoothing of sharp motion boundaries. The Horn-Schunck algorithm proceeds by iteratively minimising E which is typically slow.

A third general class of optical flow algorithms use correlative searching to find the shift of a point in one image to the next. The Camus algorithm [3]

is one such method, which explores search patterns from one pixel into neighbouring pixels in subsequent frames to minimise a pixel intensity correlation constraint.

This paper presents a new direct approach to the problem of optical flow using the geometric algebra. In section 2 the aspects of geometric algebra used are outlined, the optical flow constraint equation is reposed in terms of geometric algebra and a direct solution is proposed. Section 3 discusses implementational issues of the proposed algorithm and outlines steps taken to compare the new method with the above which are of a similar fundamental approach. Section 4 gives results and some observations are made.

2 Geometric Formulation

2.1 Geometric Algebra

The geometric product of Geometric (or Clifford) Algebra (GA) [4] of vectors is formed by adding the inner product of two vectors a and b (in GA it is conventional to not use bold face for vectors) with the outer product, that is.

$$ab = a \cdot b + a \wedge b$$

where the inner product is

$$a \cdot b = \sum_i a_i b_i$$

the outer product (or bivector) term is

$$a \wedge b = \sum_{i,j,i \neq j} e_i e_j (a_i b_j - b_i a_j)$$

where e_i defines an orthonormal (in this case) basis common to a and b . The term $e_i e_j$ is called the pseudoscalar and generates the bivector equation because

$$e_i e_j = -e_j e_i, \quad i \neq j$$

Incidentally the bivector is directly related to the cross product in 3D, but provides an extension into other dimensions. The geometric product has the vector inverse

$$b = \frac{a}{a^2}(ab) = \frac{a}{a \cdot a} ab$$

which holds because the geometric product is associative and $a^2 = a \cdot a$. Lastly note that the result of a geometric product is called a multivector and can consist of any combination of a scalar, vector and bivector terms and define points in multivector space. For vectors of two dimensions this multivector space has a scalar product term, two vector terms and one bivector term.

2.2 Geometric Optical Flow Constraint

We begin by considering equation 1 as being half the picture of the true geometric optical flow constraint and remembering that if we have an inner product then there will be an associated outer product $\nabla I \wedge u = b(x)$ where $x = (x, y)$ is the spatial position vector (here non-mathtype symbols refer to scalar terms). Adding the inner and outer terms gives the multivector geometric optical flow constraint equation

$$\nabla I u = \nabla I \cdot u + \nabla I \wedge u = -I_t + b \hat{x} \hat{y}$$

which when inverted gives the two components of u as

$$\begin{aligned} u_1 &= \frac{1}{|\nabla I|^2} (-I_x I_t - I_y b) \\ u_2 &= \frac{1}{|\nabla I|^2} (-I_x b + I_y I_t) \end{aligned} \quad (2)$$

thus by mathematically bringing out the intrinsic geometric nature of optical flow (a two dimensional vector problem) the number of independent unknown variables is reduced from two to one. Any selection of b will solve equation 1 and varying b effectively varies the angle between u and ∇I , providing a solution to the first aspect of the aperture problem. To solve the second aspect of the aperture problem we seek the image $b(x)$ which provides a smooth solution for u without using the regression or convex descent type approaches commonly utilised in computing optical flow.

2.3 Direct Solution

A simple direct solution for estimating b can be found by enforcing smoothness directly on u (equation 2). To this end we pose the diffusion problem

$$U_j = \sigma^2 \nabla^2 U \quad (3)$$

where $U(x, y, t, j) = u_1 + u_2$ and j is an introduced ‘diffusion time’, a separate entity to the video sequence time. To solve this directly the Fourier transform is applied which gives

$$\hat{U}_j = -\sigma^2 |k|^2 \hat{U} \quad (4)$$

where k is the spatial frequency vector in Fourier space. The solution is

$$U = \frac{1}{\sigma^2 j} U_0 * e^{-\frac{|x|^2}{4\sigma^2 j}} \quad (5)$$

Where $*$ denotes spatial convolution. To find a minimising initial solution we choose $\hat{U}_j(j=0) = 0$ which with equation 4 gives

$$-\sigma^2 |k|^2 \hat{U}_0 = 0 \Rightarrow U_0 = 0 \quad (6)$$

where $k_i \neq 0$ and assuming $\hat{U}(k=0) = 0$. Substituting equation 2 into 6 yields

$$b = \frac{I_x + I_y}{I_x - I_y} I_t \quad (7)$$

which gives the initial (unsmoothed) solution

$$\begin{aligned} u_{10} &= \frac{1}{|\nabla I|^2} (-I_x I_t - I_y \frac{I_y + I_x}{I_x - I_y} I_t) \\ u_{20} &= \frac{1}{|\nabla I|^2} (I_x \frac{I_y + I_x}{I_x - I_y} I_t - I_y I_t) \end{aligned} \quad (8)$$

Clearly in equation 8 there will be singularities where $|\nabla I|^2 = 0$ and $I_x - I_y = 0$. To deal with these singularities the following piecewise robustness operator is acted individually upon b and $1/|\nabla I|^2$

$$\rho(x, \alpha) = \begin{cases} -\alpha & x < -\alpha \\ x & -\alpha \leq x \leq \alpha \\ \alpha & x > \alpha \end{cases} \quad (9)$$

thus singularities are simply truncated to the limit set by our choice of α .

Smoothing is performed by utilising the convolution with a Gaussian operation given in equation 5 with $j = 1$. Temporal smoothness in a computed optical flow sequence is easily achieved by incorporating the temporal dimension into equation 5 as

$$U(x, t) = \frac{1}{\sigma^2 \sigma_t} U_0 * e^{-\frac{|x|^2}{4\sigma^2} - \frac{t^2}{4\sigma_t^2}} \quad (10)$$

3 Experimentation

Applying the above method is simply a matter of implementing equations 8, 9 and 10 and choosing the free parameters σ , σ_t , α_b and $\alpha_{\nabla I}$. The impact of each free parameter is discussed here. σ controls the degree of spatial coherence and thus helps to ‘fill in’ areas of small flow due to image homogeneity much like the iterative diffusion of the Horn-Schunck algorithm. However selecting a large value for σ will clearly blur sharp flow boundaries, thus the aperture problem is not completely solved. Likewise σ_t controls the degree of temporal coherence, motion is increasingly blurred with increasing σ_t . To select α_b consider $b = \nabla I \wedge u$. In our implementation we set the range of I to $[0, 1]$ so ∇I has the range $[-1, 1]$ and we assume u likewise. Therefore the range of b is $[-2, 2]$ from which we set $\alpha_b = 2$. In equation 8 the inverse gradient magnitude term serves to increase flow in areas of image homogeneity but such areas typically produce unreliable flow values, thus in selecting $\alpha_{\nabla I}$ there is a tradeoff between boosting flow in homogenous regions and increasing flow uncertainty.

The method optical of flow presented here is direct and thus clearly faster than other non-direct methods, however it is instructive to test the accuracy of our method compared to others. The Horn-Schunck and Lucas-Kanade algorithms are based on solving the same fundamental approximation to image motion that we has used, thus we compare our algorithm to them. We do not consider the correlative search (area matching) algorithms such as in [3] as they utilise a fundamentally different approach to posing the problem. Comparative assessment of such methods can be found in [5]. The Horn-Schunck algorithm is applied with Lagrangian multiplier $\lambda = 50$ for 100 iterations. The Lucas-Kanade algorithm is applied using singular value decomposition psudeo inverse linear regression over a box profile window of size 5×5 pixels. For our algorithm we choose $\sigma = 3$, $\sigma_t = 2$, $\alpha_t = 2$ and $\alpha_{\nabla I} = 100$. Filter dimensions were 3 times that of the corresponding σ values.

To test the accuracy of each method we use the techniques presented in [5] and the test sequences offered for use by the same¹. Angle error is computed as

$$E_A = \cos^{-1}(\hat{c} \cdot \hat{e}) \quad (11)$$

where c is the correct motion vector, e is the optical flow estimate and the hats here denote vector normalisation. For angle error a small number δ is added as to each vector as a third vector component to ensure that the error in angle for small valued flows is diminished as these flows are typically unreliable. For our experiments we choose $\delta = 0.1$. Magnitude error is computed as

$$E_M = \begin{cases} \frac{\|c-e\|}{\|c\|}, & \|c\| \geq T \\ \frac{\|e-T\|}{T}, & \|c\| < T \text{ and } \|e\| \geq T \\ 0, & \|c\| < T \text{ and } \|e\| < T \end{cases} \quad (12)$$

where T is used to diminish error where flow magnitude is small, that is like in [5] we don’t expect flow to be accurate were magnitude is small. Here we choose $T = 1$. The test sequences include three real life scenes and three synthetic scenes. The real life scenes are the MoreBlocks sequence, the Sinusoid grid sequence and the ZoomBox sequence. The synthetic sequences are the Office sequence the Street sequence and the Sphere sequence.

4 Results and Observations

Figure 1 shows two frames from the Street sequence along with the horizontal flow computed by our method. The motion of most objects in the scene is to the left which is clearly indicated by the dark

¹<http://www.cs.otago.ac.nz/research/vision/Research/OpticalFlow/opticalflow.html>, date accessed 15/03/2005.

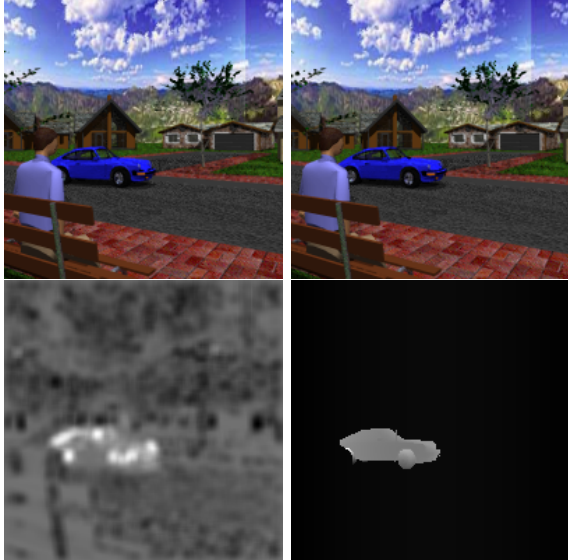


Figure 1: (top) Two consecutive frames from the Street sequence, (below left) Horizontal motion as computed by optical flow and (below right) the true horizontal motion.

regions. The motion of the car to the right is indicated by bright areas. Inspection of the car region and person region in the flow image shows the difficulty the new method has with estimating flow in areas of low image gradient, a common difficulty in optical flow that is typically solved by multi-resolution analysis.

Figure 2 shows a frame from the SR1 tree sequence (not one of the test-bed sequences) and the corresponding estimated horizontal motion. This sequence was generated by a camera panning past some trees. The motion of the foreground tree is apparent in the flow image, as well as the motion of the foreground grass and the mid-ground tree in the left of the scene. There is a log behind the foreground tree which is not so obvious in the flow image.

Figure 3 shows the cumulative error plots for the geometric method, the Horn-Schunck algorithm and the Lucas-Kanade algorithm. The error curves for the new method is very similar to the Horn-Schunck method while the Lucas-Kanade method out performs both at low error margins but is inferior at higher error margins, likely because of the ‘local averaging’ by regression used by the Lucas-Kanade method accounting for image noise, flow boundaries and ‘filling in’ regions of low image gradient by virtue of least-mean-square averaging, but with the trade off of error where ‘textural’ flow variation is smaller than the regression window. The similarity to the Horn-Schunck method is of no surprise as both methods attempt to solve very similar algebraic constraints under the same smoothness criterion.



Figure 2: (left) A frame from the SR1 Tree sequence, the camera is panning to the left and (right) the corresponding flow estimate.

The average cumulative angle error plot for the new method shows that almost all flow is within 90 degrees of the true flow direction, verifying that the new method is not estimating flow to be in completely the wrong direction(!) and thus has some merit. It is not very accurate as evidenced by the sharp decline in accuracy at lower angle error margins. The average cumulative magnitude error plot shows that 80% of flow is within flow magnitude of 1 from the true flow, meaning that 20% of all flow magnitude error in estimation is greater than 1. Furthermore the drop in flow magnitude error less than 1 is very sharp indicating low precision in flow magnitude estimation.

The major source of error with this new method is that flow boundaries are not addressed by the smoothness constraint. A nonlinear smoothness

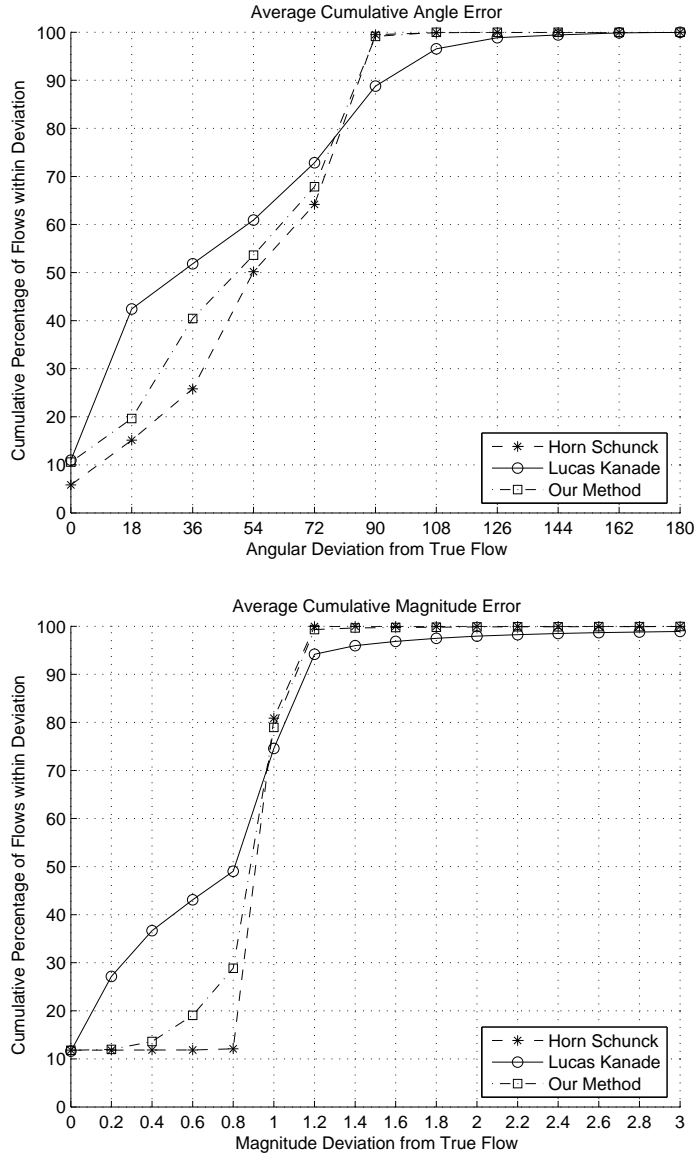


Figure 3: (top) Cumulative plot of the average angle error and (below) Cumulative plot of the average angle error

constraint such as that posed in [6] may improve results but such methods are difficult to solve directly, potentially nullifying the main strength of the method presented here. Alternatively a linear anisotropic smoothness constraint where strength of smoothing in a given direction at a given point is dependent on spatial-temporal gradient information may improve the result, again as long as the posed smoothness constraint allows for a direct mathematical solution.

This optical flow method presented here implemented in MatLab using an AMD 2.4 GHz desktop computer with 512Mbytes of RAM takes approximately 0.3-0.5 seconds to find the optical flow for two frames of a video sequence of size 512×512

or less pixels. In contrast the 100 iterations of the Horn-Schunck algorithm takes about 6 seconds in MatLab with the same computer. The Lucas-Kanade algorithm is abnormally slow in MatLab as it is an interpreted language. However it is reasonable to expect the direct geometric method to be faster than the Lucas-Kanade method which involves matrix inversion by singular value decomposition, which in turn involves requires eigenvector calculation. Correctly implemented with parallel hardware it is reasonable to expect that this geometric optical flow algorithm would approach if not reach realtime video speeds, especially if the convolution step (the slowest in the algorithm) is sped up by such hardware.

5 Acknowledgements

Useful discussion and proof reading by Michael Cree and the use of University of Waikato Physics computer resources is gratefully acknowledged.

References

- [1] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *Proceedings of Imaging Understanding Workshop*, pp. 121–130, 1981.
- [2] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” tech. rep., Massachusetts Institute of Technology, 1980. AI Memo 572.
- [3] T. Camus, “Real-time quantized optical flow,” *Journal of Real-Time Imaging*, vol. 3, pp. 71–86, 1997.
- [4] D. Hestenes, *New Foundations for Classical Mechanics (Fundamental Theories of Physics)*. Springer, 1999.
- [5] B. McCane, K. Novins, D. Crannitch, and B. Galvin, “On benchmarking optical flow,” *Computer Vision and Image Understanding*, vol. 84, no. 1, pp. 126–143, 2001.
- [6] M. Black, *Robust Incremental Optical Flow*. PhD thesis, Yale University, Department of Computer Science, 1992.