Tina Memo No. 2004-012
Internal

Tutorial: Computing 2D and 3D Optical Flow.

J.L.Barron and N.A.Thacker.

Last updated
20 / 1 / 2005

Imaging Science and Biomedical Engineering Division,
Medical School, University of Manchester,
Stopford Building, Oxford Road,
Manchester, M13 9PT.

1 Introduction to Optical Flow

Optical flow is an approximation of the local image motion based upon local derivatives in a given sequence of images. That is, in 2D it specifies how much each image pixel moves between adjacent images while in 3D in specifies how much each volume voxel moves between adjacent volumes. The 2D image sequences used here are formed under perspective projection via the relative motion of a camera and scene objects. The 3D volume sequences used here were formed under orthographic projection for a stationary sensor and a moving/deforming object. In both cases, the moving patterns cause temporal varieties of the image brightness. It is assumed that all temporal intensity changes are due to motion only. This is usually true but there are many exceptions (see below). 2D/3D derivatives are usually computed by repeated application of lowpass and highpass filters, for example the filters proposed by Simoncelli [2]. Thus the computation of differential optical flow is, essentially, a two-step procedure:

1. measure the spatio-temporal intensity derivatives (which is equivalent to measuring the velocities normal to the local intensity structures) and
2. integrate normal velocities into full velocities, for example, either locally via a least squares calculation [4, 28] or globally via a regularization [3, 28].

Such algorithms are generally designed to work on a specific form of image [1]. There can be no occlusion (one object moving in front of/or behind another object), again unless this is modelled for. Similarly we assume there are no specularities in the scene (otherwise the light source(s) and sensor(s) positions would have to be explicitly modelled).

Finally, all objects in the scene are rigid, no shape changes allowed. This assumption is often relaxed to local rigidity. This assumption assures that optical flow actually captures real motions in a scene rather than expansions, contractions, deformations and/or shears of various scene objects.

In general, the use of optical flow in a generic machine vision system will probably require a sophisticated analysis of image content and motion in order to determine that all of the algorithmic assumptions are likely to be met. Quantitative use of the data will also require quantitative predictions of accuracy. The algorithms described here have been made available as a project within the Tina system, see [1].

2 The 2D and 3D Motion Constraint Equation

The basis of differential optical flow is the **motion constraint equation** which we derive below in 2D and 3D.

2.1 The 2D Motion Constraint Equation

Assume $I(x, y, t)$ is the center pixel in a $n \times n$ neighbourhood and moves by δx, δy in time δt to $I(x+\delta x, y+\delta y, t+\delta t)$. Since $I(x, y, t)$ and $I(x + \delta x, y + \delta y, t + \delta t)$ are the images of the same point (and therefore the same) we have:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t). \tag{1}$$

This assumption forms the basis of the **2D Motion Constraint Equation** and is illustrated in Figure 1 below. The assumption is true to a first approximation (small local translations) provided δx, $\delta y, \delta t$ are not too big. We can perform a 1^{st} order Taylor series expansion about $I(x, y, t)$ in equation (1) to obtain:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + H.O.T., \tag{2}$$

[1] Consider in the 2D case, a rotating metal sphere under a lighting source fixed in position and intensity. Then there will be no changing intensity and hence no optical flow even though the sphere is in motion. Alternatively, if the sphere is stationary but the light source's position and/or intensity changes, optical flow can be measured even though there is no motion. Of course, this example ignores the difficulties of measuring image derivatives on textureless surfaces or in the presence of specularities.

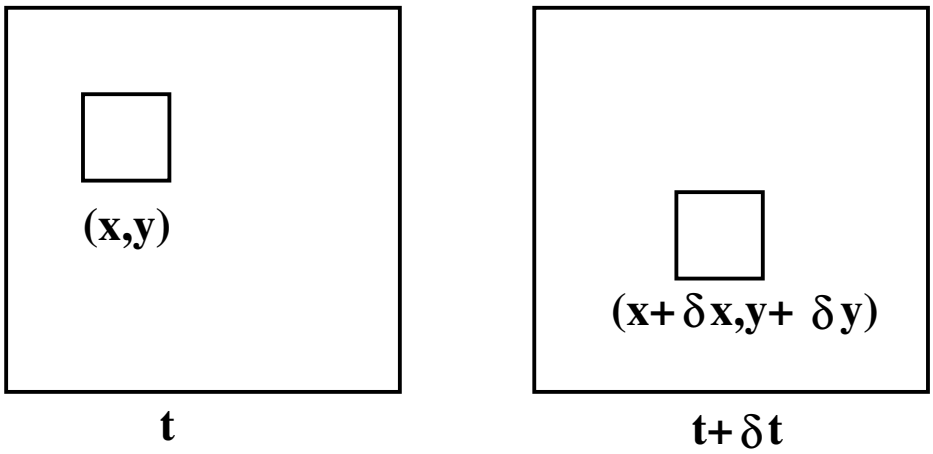

Figure 1: The image at position (x, y, t) is the same as at $(x + \delta x, y + \delta y, t + \delta t)$

where *H.O.T.* are the Higher Order Terms, which we assume are small and can safely be ignored. Using the above two equations we obtain:

$$\begin{aligned} \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t &= 0 \text{ or} \\ \frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\underbrace{\frac{\delta t}{\delta t}}_{=1} &= 0 \text{ and finally :} \\ \frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} &= 0. \end{aligned} \tag{3}$$

Here $v_x = \frac{\delta x}{\delta t}$ and $v_y = \frac{\delta y}{\delta t}$ are the x and y components of **image velocity** or **optical flow**[2] and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are image intensity derivatives at (x, y, t). We normally write these partial derivatives as:

$$I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y} \text{ and } I_t = \frac{\partial I}{\partial t}. \tag{4}$$

Note the difference between (v_x, v_y) which are the x and y components of optical flow and (I_x, I_y, I_t) which are intensity derivatives. This equation can be rewritten more compactly[3] as:

$$(I_x, I_y) \cdot (v_x, v_y) = -I_t \tag{5}$$

or as:

$$\nabla I \cdot \vec{v} = -I_t, \tag{6}$$

where $\nabla I = (I_x, I_y)$ is the **spatial intensity gradient** and $\vec{v} = (v_x, v_y)$ is the image velocity or optical flow at pixel (x, y) at time t. $\nabla I \cdot \vec{v} = -I_t$ is called the **2D Motion Constraint Equation** and is 1 equation in 2 unknowns (a line) as shown in Figure 2b. This is a consequence of the **aperture** problem: there is usually insufficient local image intensity structure to measure full image velocity, but sufficient structure to measure the component normal to the local intensity structure. Figure 2a shows one example of the aperture problem, where a line moving up and to the right is viewed through a circular aperture. In this case, it is impossible to recover the correct full image velocity, but only the image velocity normal to the line. The problem of computing full image velocity then becomes finding an additional constraint that yields a second different equation in the same unknowns.

Normal velocity then is a local phenomenon and occurs when there is insufficient local intensity structure to allow a full image velocity to be recovered. In this case, only the component of velocity normal to the local intensity structure (for example, an edge), $\vec{v}_n$, can be recovered. The tangential component of the velocity, $\vec{v}_t$, cannot be recovered. As shown in Figure 2b, the 2D Motion Constrain Equation yields a line in $\vec{v} = (v_x, v_y)$ space. One velocity on this line is the correct velocity. The velocity with the smallest magnitude on that line is the normal velocity $\vec{v_n}$. The magnitude and direction of the normal velocity, $\vec{v}_n = v_n\hat{n}$ can be computed solely in terms of the

[2] We use these terms interchangeably.

[3] Note that we ignore the transpose symbol T when we can to keep our equations "cleaner". Thus $\nabla I \cdot \vec{v}^T$ becomes $\nabla I \cdot \vec{v}$. Both ∇I and $\vec{v}$ are column vectors. The meaning should be obvious from the context.

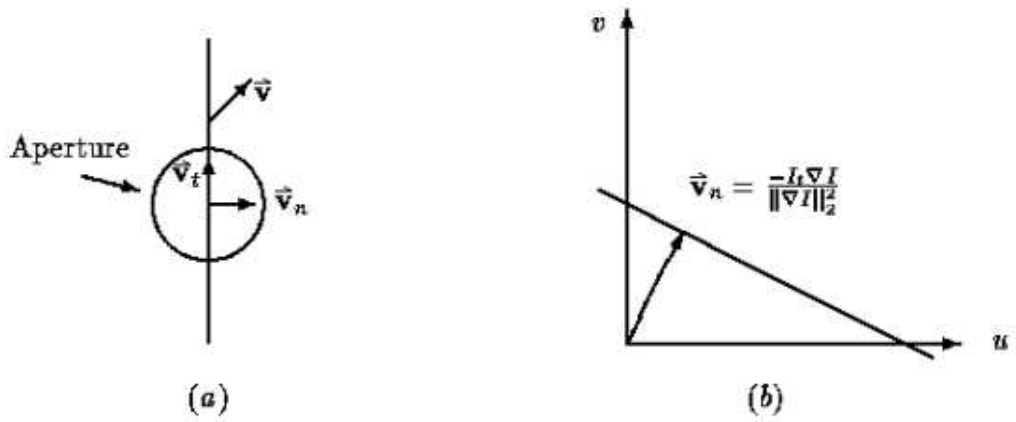

Figure 2: (a) The aperture problem: only normal velocity $\vec{v}_n$ can be recovered but tangential velocity $\vec{v}_t$ cannot. (b) The **motion constraint** equation yields a line in $\vec{v} = (v_x, v_y)^T$ space. One of the velocities on the line is the correct one. The velocity with the smallest magnitude on that line is $\vec{v_n}$.

intensity derivatives, I_x, I_y and I_t as:

$$v_n = \frac{-I_t}{||\nabla I||_2} \quad \text{and} \quad \hat{n} = \frac{(I_x, I_y)}{||\nabla I||_2}. \tag{7}$$

v_n and $\hat{n}$ are the **raw** normal velocity magnitude and the **raw** normal velocity unit direction, respectively, i.e.:

$$\vec{v}_n = v_n \hat{n} = \frac{-I_t(I_x, I_y)}{||\nabla I||_2^2} \tag{8}$$

is the **raw** normal velocity and $\nabla I = (I_x, I_y)$ is the spatial intensity gradient. For completeness purposes, we include a discussion of normal velocity in our write-up. The Lucas and Kanade optical flow algorithm allows the computation of normal velocity but the Horn and Schunck optical flow algorithm does not.

Note that the 2D motion constraint equation can be re-written as:

$$\vec{v} \cdot \hat{n} = v_n, \tag{9}$$

which is equivalent to equation (6), since the unit direction of normal velocity is $\hat{n} = \frac{(I_x, I_y)}{||(I_x, I_y)||_2}$ and the magnitude of normal velocity is $v_n = \frac{-I_t}{||(I_x, I_y)||_2}$ as in equation (7), see Barron et al. [5] for more details.

2.2 The 3D Motion Constraint Equation

The **3D Motion Constraint Equation** is a simple extension of the 2D motion constraint equation. Figure 3 shows a small 3D $n \times n \times n$ block at (x, y, z) at time t moving by $(\delta x, \delta y, \delta z)$ to $(x + \delta x, y + \delta y, z + \delta z)$ over time δt. Since $I(x, y, z, t) = I(x + \delta x, y + \delta y, z + \delta z, t + \delta t)$ we can perform a 1^{st} order Taylor series expansion (as in the 2D case) and obtain:

$$I_x V_x + I_y V_y + I_z V_z + I_t = 0, \tag{10}$$

where $\vec{V} = (V_x, V_y, V_z) = \left(\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}, \frac{\delta z}{\delta t}\right)$ is the **3D volume velocity** (or **3D optical flow**) and I_x, I_y, I_z and I_t are the 3D spatio-temporal derivatives. Equation (10) can be written more compactly as:

$$\nabla I \cdot \vec{V} = -I_t, \tag{11}$$

where $\nabla I = (I_x, I_y, I_z)$ is the 3D spatial intensity gradient, I_t is the temporal intensity derivative and $\vec{V} = (V_x, V_y, V_z)$ is the 3D velocity. This equation describes a plane in 3D space. It can be rewritten as:

$$\vec{V} \cdot \hat{n} = V_n, \tag{12}$$

where $\vec{V_n} = V_n \hat{n}$ is a 3D plane normal velocity (see the explanation below), $\hat{n}$ is now the 3D normal direction (the surface normal of a plane) and V_n is the 3D plane normal velocity magnitude:

$$\vec{V_n} = \frac{-I_t(I_x, I_y, I_z)}{||(I_x, I_y, I_z)||_2^2}. \tag{13}$$

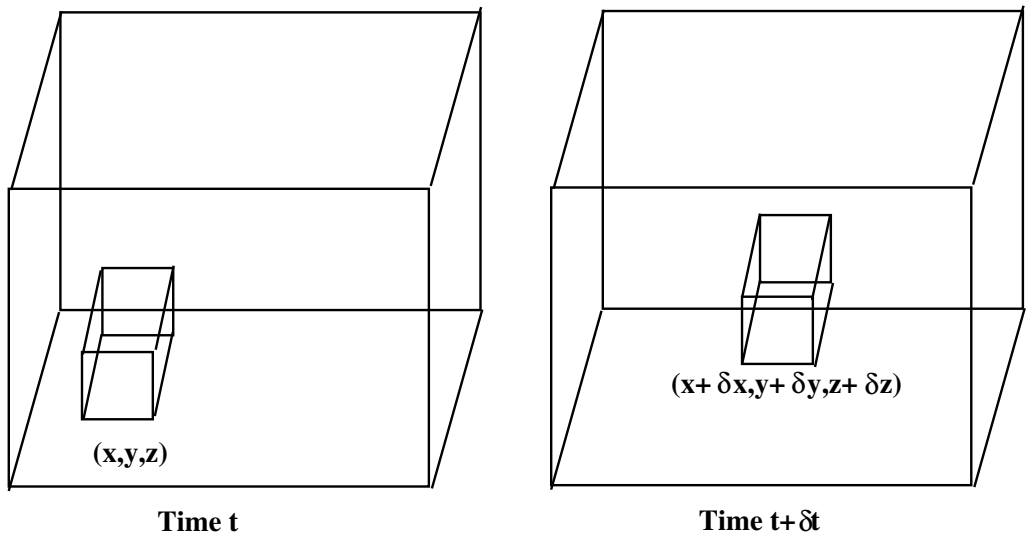

Figure 3: A small $n \times n \times n$ 3D neighbourhood of voxels centered at (x, y, z) at time t moving to $(x+\delta x, y+\delta y, z+\delta z)$ at time $t + \delta t$.

The aperture problem in 3D actually yields two types of normal velocity: **plane normal** velocity (the velocity normal to a local intensity planar structure) and **line normal** velocity (the velocity normal to a local line intensity structure causes by the intersection of 2 planes) and full details can be found in other papers [6, 7][4]. Plane and line normal velocities are illustrated in Figure 4 and explained briefly below. If the spatio-temporal derivative data

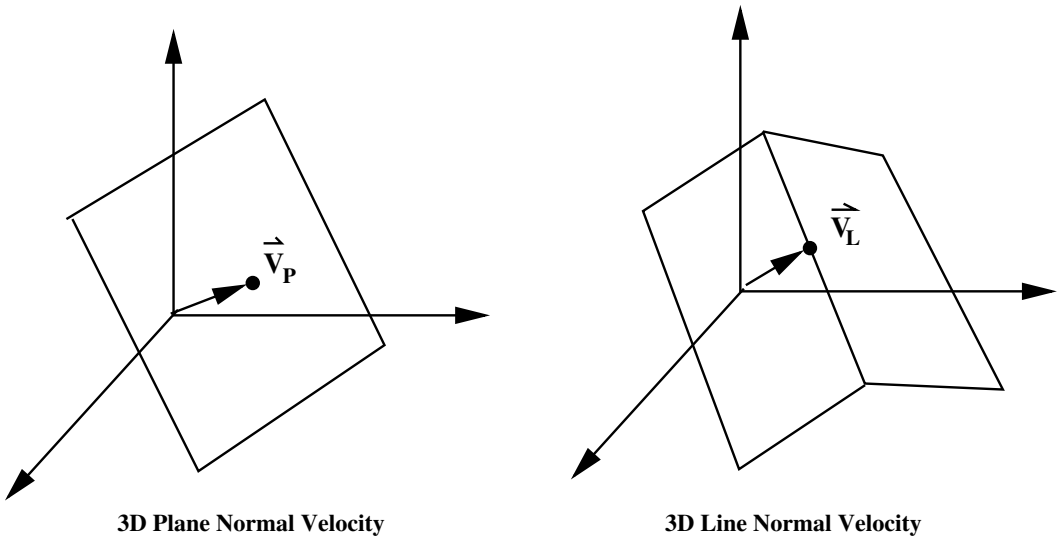

Figure 4: Graphical illustrations of the 3D plane and line normal velocities.

best fits a single plane only, we have plane normal velocity. The point on the plane closest to the origin $(0, 0, 0)$ is its magnitude and the plane surface normal is its direction. If the spatio-temporal derivative data best fits two separate planes (perhaps found by an EM calculation), then the point on their intersection line closest to the origin $(0, 0, 0)$ is the line normal velocity. Of course, if the spatio-temporal derivative data fits 3 or more planes we have a full 3D velocity calculation (the best intersection point of all the planes via a (total) least squares fit). We are concerned only with the computation of full 3D velocity for the programs described here.

[4]These papers describe the 3D aperture problems and the resulting types of normal velocity and their computation for 3D range flow (which is just 3D optical flow for points on a moving surface) and, with only minor changes, applies to 3D (volumetric) optical flow, which we are interested in here.

3 2D/3D Differentiation

Regardless of the optical flow method used, we need to compute image intensity derivatives. Differentiation was done using Simoncelli's [2] matched balanced filters for low pass filtering (blurring) [p_5 in Table 1] and high pass filtering (differentiation) [d_5 in Table 1]. Matched filters allow comparisons between the signal and its derivatives as the high pass filter is simply the derivative of the low pass filter and, from experimental observations, yields more accurate derivative values.

Before performing Simoncelli's filtering we use the simple averaging filter suggested by Simoncelli, $[\frac{1}{4}, \frac{1}{2}, \frac{1}{4}]$, to slightly blur the images. Simoncelli claims that, because both of his filters were derived from the same principles, more accurate derivatives result and he demonstrated this on the Yosemite Fly-Through sequence [2, 5]. We have independently verified these results for the same data.

n	p_5	d_5
0	0.036	-0.108
1	0.249	-0.283
2	0.431	0.0
3	0.249	0.283
4	0.036	0.108

Table 1: Simoncelli's 5-point Matched/Balanced Kernels

To compute I_x in 2D, we first convolve the smoothing kernel, p_5, in the t dimension to reduce the 5 images to 1 image, then convolve the smoothing kernel p_5 on that result in the y dimension and then convolve the differentiation kernel, p_5, on that result in the x dimension to obtain I_x. To compute I_t in 2D for frame i we first convolve p_5 in the x dimension and on that result in the y dimension for frames $i-2$, $i-1$, i, $i+1$ and $i+2$. This yields 5 smoothed images in x and y. We then differentiate these images in the t dimension using d_5 to get I_t. Figure 5 graphically illustrates the application of these kernels in 2D.

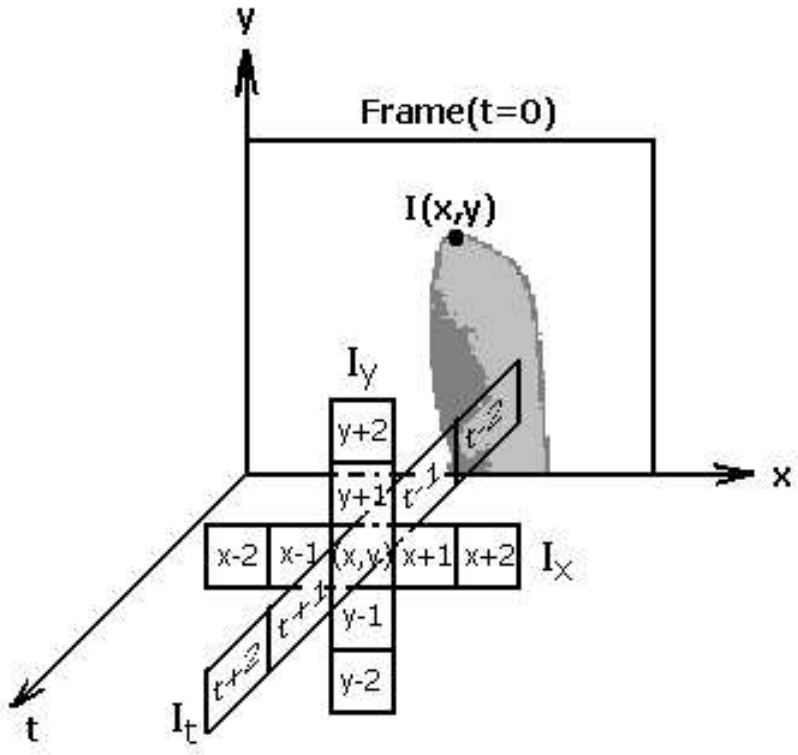

Figure 5: I_x, I_y, and I_t are computed using the 5 tap lowpass or highpass Simoncelli filters [2] with 1D coefficients at ($i-2$, $i-1$, i, $i+1$,$i+2$) in one of the x, y, or t dimensions, as given in Table 1.

To compute I_x in 3D, we first smooth in the t dimension first using p_5 (to reduce the 5 volumes of 3D data to 1 volume of 3D data), then smooth that result in the y dimension using p_5 and then smooth that new result in

the z dimension, again using p_5, and finally differentiate the fully smoothed result in the x dimension using d_5. Similar operations are performed to compute I_y and I_z. To compute I_t in 3D, we smooth each of the 5 volumes, first in the x dimension, then that result in the y dimension and finally that new result in the z dimension, using p_5 (theoretically the order is not important). Lastly, we differentiate the 5 volumes of smoothed derivative data using d_5 in the t dimension (this computation is a CPU intensive operation).

4 2D Optical Flow

Below, we describe the 2D optical flow algorithms attributed to Lucas and Kanade [4] (a local least squares calculation) and Horn and Schunck [3] (a global regularization calculation).

4.1 2D Lucas and Kanade

Following Lucas and Kanade [4], we implemented a weighted least-squares (LS) fit of local first-order constraints (6) to a constant model for $\vec{v}$ in each small spatial neighbourhood Ω by minimizing:

$$\sum_{x,y\in\Omega} W^2(x,y)[\nabla I(x,y,t)\cdot\vec{v}+I_t(x,y,t)]^2, \tag{14}$$

where $W(x,y)$ denotes a window function that gives more influence to constraints at the centre of the neighbourhood than those at the periphery. [W are typically 2D Gaussian coefficients]. The solution to (14) is given by:

$$\vec{v}=[A^TW^2A]^{-1}A^TW^2\vec{b}, \tag{15}$$

where, for N pixels (for a $n\times n$ neighbourhoods $N=n^2$), $(x_i,y_i)\in\Omega$ at a single time t:

$$\begin{aligned} A &= [\nabla I(x_1,y_1),...,\nabla I(x_N,y_N)],\\ W &= \mathrm{diag}[W(x_1,y_1),...,W(x_N,y_N)],\\ \vec{b} &= -(I_t(x_1,y_1),...,I_t(x_N,y_N)). \end{aligned} \tag{16}$$

The solution to equation (15) can can be solved in closed form when A^TW^2A is a nonsingular matrix. A^TW^2A is the 2×2 matrix:

$$A^TW^2A=\begin{bmatrix} \sum W^2(x,y)I_x^2(x,y) & \sum W^2(x,y)I_x(x,y)I_y(x,y)\\ \sum W^2(x,y)I_y(x,y)I_x(x,y) & \sum W^2(x,y)I_y^2(x,y)\end{bmatrix}, \tag{17}$$

where all sums are taken over pixels (x,y) in the neighbourhood Ω. We use $n=5$ neighbourhoods in the algorithms implemented here. We set all the weights in W to 1.0 as experimentation showed that using Gaussian weights had little effect on the accuracy of the result [5].

We perform an eigenvalue/eigenvector decomposition on A^TW^2A. We obtain two eigenvalues, $\lambda_2\geq\lambda_1\geq 0$, and their corresponding orthogonal unit eigenvectors, $\hat{e}_1$ and $\hat{e}_2$. If the smallest eigenvalue, $\lambda_1\geq\tau_D$, where τ_D is a user specified threshold (we use 1.0 typically), then the $\vec{v}$ computed by equation (15) is accepted as a reliable full velocity. If $\lambda_1<\tau_D$ but $\lambda_2\geq\tau_D$ then we can compute a least squares normal velocity by projecting $\vec{v}$ in the direction of the larger eigenvalue:

$$\vec{v}_n=(\vec{v}\cdot\hat{e}_2)\hat{e}_2. \tag{18}$$

Although we do not display normal velocities, our 2D Lucas and Kanade code actually computes both raw normal velocity as given in equation (7) and least squares normal velocity as given above in equation (18) [5].

4.2 2D Horn and Schunck

Horn and Schunck [3] combined the gradient constraint (6) with a global smoothness term to constrain the estimated velocity field $\vec{v}=(v_x,v_y)$, minimizing:

$$\int_D (\nabla I\cdot\vec{v}+I_t)^2+\lambda^2\left[\left(\frac{\partial v_x}{\partial x}\right)^2+\left(\frac{\partial v_x}{\partial y}\right)^2+\left(\frac{\partial v_y}{\partial x}\right)^2+\left(\frac{\partial v_y}{\partial y}\right)^2\right]dxdy \tag{19}$$

[5] They are in the *norm_vels*1 and *norm_vels*2 Imrects parameters of the **compute_lucas_optical_flow_2D** function in the **calc2Dflow.c** file.

defined over a domain D (the image), where the magnitude of λ reflects the relative influence of the smoothness term. We typically used $\lambda = 1.0$ instead of $\lambda = 10.0$ as suggested in [3] because it produced better results in most of our test cases. Iterative equations are used to minimise (19) and the image velocity can be obtained from the Gauss Seidel equations that solve the appropriate Euler-Lagrange equations:

$$\begin{aligned} v_x^{k+1} &= \bar{v}_x^k - \frac{I_x[I_x\bar{v}_x^k + I_y\bar{v}_y^k + I_t]}{\alpha^2 + I_x^2 + I_y^2}, \\ v_y^{k+1} &= \bar{v}_y^k - \frac{I_y[I_x\bar{v}_x^k + I_y\bar{v}_y^k + I_t]}{\alpha^2 + I_x^2 + I_y^2}, \end{aligned} \tag{20}$$

where k denotes the iteration number, v_y^0 and v_y^0 denote initial velocity estimates which are typically set to zero[6] and $\bar{v}_x^k$ and $\bar{v}_y^k$ denote neighbourhood averages of v_x^k and v_y^k. We typically use about 100 iterations in all testing below (although the **2D flowtool** allows you to set the number of iterations). Let $\vec{v}_{2Dof}$ be the $N \times M \times 2$ vector constructed from the the $N \times M$ optical flow vectors computed for an image of size $N \times M$, whose components are the consecutive 2D velocity components, laid out side by side. A test to stop the iterations is when the flow field has converged, i.e. when

$$||\vec{v}_{2Dof}^k - \vec{v}_{2Dof}^{k-1}||_2 \leq TOL, \tag{21}$$

at iteration k, where TOL is some preset tolerance level. Similarly if the difference between consecutive flow field sizes is increasing, this would indicate that the flow field is diverging. We did not implement convergence or divergence tests. Our practical experience shows that these circumstances never arise.

5 3D Optical Flow

In this section, we describe the simple extensions for 3D optical flow using the Lucas and Kanade [4] and Horn and Schunck [3] approaches.

3D Lucas and Kanade

Using the 3D motion constraint equation:

$$I_xV_x + I_yV_y + I_zV_z = -I_t, \tag{22}$$

where I_x, I_y, I_z and I_t are the 3D intensity derivatives in a $n \times n \times n$ neighbourhood centered at voxel (x, y, z), we assume a constant velocity (V_x, V_y, V_z) in that neighbourhood and minimize:

$$\sum_{x,y,z\in\Omega} W^2(x,y,z)\left[\nabla I(x,y,z,t)\cdot\vec{V}^T + I_t(x,y,z,t)\right]^2, \tag{23}$$

where $W(x, y, z)$ denotes a Gaussian windowing function. Equation (23) can be solved by setting up a system of equations:

$$\vec{V} = [A^TW^2A]^{-1}A^TWB, \tag{24}$$

where the diagonal elements of W are the $N = n \times n \times n$ 3D Gaussian coefficients, the N rows of A consist of I_x, I_y and I_z for each (x, y, z) position and the N rows of B consist on the $-I_t$ values for those (x, y, z) positions. That is:

$$\begin{aligned} A &= [\nabla I(x_1,y_1,z_1), ..., \nabla I(x_N,y_N,z_N)], \\ W &= \text{diag}[W(x_1,y_1,z_1), ..., W(x_N,y_N,z_N)], \\ B &= -(I_t(x_1,y_1,z_1), ..., I_t(x_N,y_N,z_N)). \end{aligned} \tag{25}$$

A^TW^2A is computed as:

$$A^TW^2A = \begin{pmatrix} \sum W^2(x,y,z)I_x(x,y,z)^2 & \sum W^2(x,y,z)I_x(x,y,z)I_y(x,y,z) \\ \sum W^2(x,y,z)I_y(x,y,z)I_x(x,y,z) & \sum W^2(x,y,z)I_y(x,y,z)^2 \\ \sum W^2(x,y,z)I_z(x,y,z)I_x(x,y,z) & \sum W^2(x,y,z)I_z(x,y,z)I_y(x,y,z) \end{pmatrix}$$

[6] v_x^0 and v_y^0 could also be the optical flow computed by another method, for example Lucas and Kanade.

$$\left.\begin{array}{l}\sum W^2(x,y,z)I_x(x,y,z)I_z(x,y,z)\\ \sum W^2(x,y,z)I_y(x,y,z)I_z(x,y,z)\\ \sum W^2(x,y,z)I_z(x,y,z)^2\end{array}\right). \tag{26}$$

Again, we set W entries to 1.0 as in the 2D case and use $n = 5$. We perform eigenvalue/eigenvector analysis of A^TW^2A to compute eigenvalues $\lambda_3 \geq \lambda_2 \geq \lambda_1 \geq 0$ and accept as reliable full 3D velocities those velocities with $\lambda_1 > \tau_D$. The default value for τ_D is 1.0. We don't compute line normal (when $\lambda_1 < \tau_D$ and $\lambda_3 \geq \lambda_2 \geq \tau_D$) or plane normal (when $\lambda_1 \leq \lambda_2 < \tau_D$ and $\lambda_3 \geq \tau_D$) velocities (we did this earlier using an eigenvector/eigenvalue analysis in a total least square framework for range flow[6, 7]).

3D Horn and Schunck

We extend the Horn and Schunck regularization to:

$$\sum_R (I_xV_x + I_yV_y + I_zV_z + I_t) + \alpha^2 \left[\left(\frac{\partial V_x}{\partial x}\right)^2 + \left(\frac{\partial V_x}{\partial y}\right)^2 + \left(\frac{\partial V_x}{\partial z}\right)^2 + \right.$$
$$\left.\left(\frac{\partial V_y}{\partial x}\right)^2 + \left(\frac{\partial V_y}{\partial y}\right)^2 + \left(\frac{\partial V_y}{\partial z}\right)^2 + \left(\frac{\partial V_z}{\partial x}\right)^2 + \left(\frac{\partial V_z}{\partial y}\right)^2 + \left(\frac{\partial V_z}{\partial z}\right)^2\right]. \tag{27}$$

Iterative Gauss Seidel equations that minimize the Euler-Lagrange equations based on this integral are:

$$V_x^{k+1} = \bar{V}_x^n - \frac{I_x\left[I_x\bar{V}_x + I_y\bar{V}_y + I_z\bar{V}_z + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}, \tag{28}$$

$$V_y^{k+1} = \bar{V}_y^k - \frac{I_y\left[I_x\bar{V}_x + I_y\bar{V}_y + I_z\bar{V}_z + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)} \text{ and} \tag{29}$$

$$V_z^{k+1} = \bar{V}_z^k - \frac{I_z\left[I_x\bar{V}_x + I_y\bar{V}_y + I_z\bar{V}_z + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}, \tag{30}$$

where $\bar{V}_x^k$, $\bar{V}_y^k$ and $\bar{V}_z^k$ are $5 \times 5 \times 5$ averages of neighbourhoods of velocities at iteration k. We perform a fixed number of iterations as specified by the user (the default is 50 but we usually used 100 or 200). [We did not check for convergence or divergence in this implementation.]

6 Testing and Theory

How is Testing Currently Performed? Optical flow can be evaluated either qualitatively and/or qualitatively. Quantitatively, error can be measured as average error in magnitude or direction [7] or an angle error measure capturing both the magnitude and direction deviation from ground truth [8]. Qualitative flow evaluation is only useful for a general judgement and as a proceed/don't proceed measure. It is usually performed in the absence of ground truth and does not produce results which can be compared in any meaningful way to other work. Synthetic or real data must be accompanied by ground truth to perform quantitative error analysis.

Are there Image Sequences for which the Correct Answers are Known? We can compute optical flow for a real image sequence made using an optical bench with known camera motion and scene structure and use it in a motion and structure calculation [16]. Similarly, one can measure optical flow for synthetic image sequences where true dense depth is known and perform a quantitative error analysis on the computed depth maps [9]. One example of 2D and 3D synthetic data is sinusoidal images/volumes (which are perfectly differentiable) and thus might be considered as **gold standard** data. 2D/3D sinusoidal image sequences were used in [5, 28]. Tagged MRI data may supply accurate 3D motion information for 3D MRI analysis but is not yet commonly available.

Are there Datasets in Common Use? Barron et al. [5] performed a quantitative analysis for 9 optical flow algorithms using the Translating/Diverging tree sequence made by David Fleet [8], the Yosemite Fly-Through sequence made by Lynn Quam at SRI and a number of translating sinusoidal/square image sequences. Otte and Nagel [17] have made a calibrated real image sequence. These data have known ground truth and are publically available[7]. Though these datasets have been available for a considerable length of time it could probably not be said that these datasets are accepted as any sort of de-facto standard.

[7] anonymous ftp to ftp.csd.uwo.ca, cd to pub/vision and http://i21www.ira.uka.de/image_sequences.

Are there Experiments which Show that the Algorithms are Stable and Work as Expected? A number of researchers have derived covariances for optic flow estimates (see for example, [18, 19, 20]). Simoncelli used a Bayesian framework assuming input Gaussian errors for the image values and multi-scale Gaussian priors for optic flow estimates and computed the optic flow estimates along with uncertainties. Comaniciu and his colleagues [20] also used a multi-scale framework and estimated the uncertainties for the optic flow estimate by utilizing the variable bandwidth meanshift estimation framework. The main significance of their work is that a non-parametric density representation for the local optic flow distribution allows for multiple motion regions in a local patch. The mode estimate of the density function, and the covariance around that mode (obtained via a variable-bandwidth meanshift filter) are used as the final refined estimate for optic flow. Other researchers have performed covariance propagation for optic flow [21, 22]. Here we are interested in the propagation of covariance matrices for random input perturbations to the covariances associated with final computed results, in this case, optical flow. There may be an inherent bias in many optical flow estimators because of the fact that the regularization assumption (e.g. the Horn and Schunck smoothness assumption in the flow field [3]) is not necessarily correct with all datasets. In other words, the true underlying smoothness constraint is an unknown and the estimation framework naturally has biases. More recent work was done by Fermüller and Aloimonos and colleagues [24, 23, 25, 10] and seeks to explain perceptual illusions through the estimation framework bias.

Ye and Haralick [11, 12] propose a 2 stage optical flow algorithm, using 'least trimmed squares' followed by weighted least square estimators. The 1^{st} stage takes into account poor derivative quality. Nestares et al. [13] use an estimate of optical flow and its covariance at each pixel [14] in a likelihood function framework to extract confidence measures of the translational sensor parameters.

Are there any Strawman Algorithms? Some of the "old" optical flow algorithms are still pretty good. Of course, there are now better algorithms, but algorithms such as Lucas and Kanade [4] and Horn and Schunck [3], and, to a lesser extent, Nagel [15] and Uras et al. [27] are pretty good, readily accessible to researchers and the code is available [5][8]. Lots of new algorithms have appeared in the literature (one only has to scan the main computer vision journals and conferences since 1994) and all claim to give better optical flow results compared to those in [5]. Still, often the results are only marginally better and the codes not generally available. Some of these classical 2D algorithms also allow simple extensions into 3D. For example, 3D optical flow on gated MRI data has been computed using simple extensions of 2D Lucas and Kanade and 2D Horn and Schunck [28].

Is there a Quantitative Methodology for the Design of Algorithms? Every algorithm has some assumptions from which the method could be derived using quantitative statistics (generally likelihood). The task therefore falls into the category of a constrained estimation problem. Until now most of the following assumptions have been implicitly made for differential optical flow.

1. The data has to be appropriately sampled to avoid aliasing, i.e. the Nyquist sampling conditions are satisfied (no aliasing). In other words, the data must allow the calculation of good derivatives! For large motions, hierarchical structures, such as a Gaussian pyramid [29] may allow differentiation. An implementation of Bergen et al.'s hierarchical approach is described in [26].

2. We assume the input noise in the images is mean zero IID Gaussian error, $N(0, \sigma^2)$. Most but not all sensors satisfy this assumption. Algorithms using least squares/total least squares then give an optimal solution.

3. We assume local translation. If the sensor motion has a rotational component we assume that it can be approximated by a number of small local translations.

4. For 1^{st} order derivatives the data should fit a straight line: the deviation from a straight line (a residual) could be used as a measure of the "goodness" of a derivative and these goodness values could be use in the subsequent optical flow calculation. Spies [30] showed that the sum of the normalized squared errors follows a χ^2 distribution [31].

What Should We be Measuring to Quantify Performance? Optical flow algorithms should provide not only an estimate of flow but also a confidence measure of how good the flow is. We believe it is better that, if an optical flow algorithm provides a poor flow field it should also provide indicative confidence measures. An optical flow field with no confidence measures is difficult to confidently use as input to other applications. Covariance

[8] The algorithms of Nagel and Uras et al. use 2^{nd} order intensity derivatives which are often difficult to measure accurately. Indeed, Horn and Schunck's use of 1^{st} order intensity derivatives is effectively a 2^{nd} order method because their smoothness constraint uses derivatives of image velocities, themselves constrained by 1^{st} order intensity derivatives.

matrices are one good candidate for such a confidence measure and Haralick's propagation framework is a good way to integrate such information in a larger vision system. There are also a number of approaches which allow us to adress the fundamental problem of constructing a gold standard for quantitative testing.

- We can use reconstruction error: with the computed flow and the current image, generate the image at the next time and compare that constructed image to the next image [32]. If the metric adopted is small (in comparison to the expected errors) then both the optical flow and the reconstruction method are good, otherwise you probably don't know with certainity which/or both is not working.
- Given good flow, it should correctly predict a future event. For example, the velocity of a Doppler storm represented as a 3D ellipsoid should be able to predict the storm(s) in the next image. We can compute the intersection of predicted and actual storm ellipsoids in the next image [33] as a measure of the 3D velocity accuracy. Quantitative comparison requires the error covariances.

  Can the correct 3D motion parameters and dense depth maps be accurately computed? Again, if the error measures are good, then both the optical flow and reconstruction are good, if the error measures are poor, this could be a result of problems in one or both calculations (and this is unknown).

References

[1] J. Barron, Incorporating Optical Flow into Tinatool., Tina Memo 2004-013.

[2] E.P. Simoncelli, "Design of multi-dimensional derivative filters", IEEE Int. Conf. Image Processing", Vol. 1, pp790-793, 1994.

[3] B.K.P. Horn and B.G. Schunck, "Determining Optical Flow", *Artificial Intelligence*, **17**, 1981, pp185–204.

[4] B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", DARPA *Image Understanding Workshop*, 1981, pp121–130 (see also *IJCAI'81*, pp674–679).

[5] J.L. Barron, D.J. Fleet and S.S. Beauchemin, "Performance of Optical Flow Techniques", *Int. Journal of Computer Vision*, **12**, 1994, pp. 43–77.

[6] H. Spies, H. Haußecker, B. Jähne, and J.L.Barron (1999), "Differential Range Flow Estimation", 21.Symposium fur Mustererkennung (DAGM1999), Springer, Bonn, September 15-17th, pp309-316.

[7] H. Spies, B. Jähne, and J. L. Barron (2002), "Range Flow Estimation" Computer Vision Image Understanding (CVIU2002), Volume 85, Number 3, March, pp209-231.

[8] D. Fleet. *Measurement of Image Velocity*. Kluwer Academic Publishers, Norwell, 1992.

[9] B. Tian, J. Barron, W. K. J. Ngai, and Spies H. A comparison of 2 methods for recovering dense accurate depth using known 3D camera motion. In *Vision Interface*, pages 229–236, 2003.

[10] C. Fermüller, Y. Aloimonos, and H. Malm. Bias in visual motion processes: A theory predicting illusions. In *Statistical Methods in Video Processing*. (in conjunction with European Conference on Computer Vision), 2002.

[11] M. Ye and R. M. Haralick. Optical flow from a least-trimmed squares based adaptive approach. In *ICPR*, pages Vol III: 1052–1055, 2000.

[12] M. Ye and R. M. Haralick. Two-stage robust optical flow estimation. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2623–2028, 2000.

[13] O. Nestares, D. J. Fleet, and D. J. Heeger. Likelihood functions and confidence bounds for total-least-squares problems. In *Proceedings of Conference on Computer Vision and Pattern Recognition CVPR 2000*, pages 1523–1530, Hilton Head, SC, USA, June 2000. IEEE Computer Society.

[14] O. Nestares and R. Navarro. Probabilistic multichannel optical flow analysis based on a multipurpose visual representation of image sequences. In *IS&T/SPIE 11th Intl. Symposium on Electronic Imaging*, pages 429–440, 1999.

[15] H.-H Nagel. Constraints for the estimation of displacement vector fields from image sequences. In *International Joint Conference on Artificial Intelligence*, pages 945–951, 1983.

[16] J. L. Barron and R. Eagleson. Recursive estimation of time-varying motion and structure parameters. *Pattern Recognition*, 29(5):797–818, May 1996.

[17] M. Otte and H.-H. Nagel. Optical flow estimation: Advances and comparisons. In *European Conference on Computer Vision*, pages 51–60, 1994.

[18] E. P. Simoncelli. *Bayesian Multi-scale Differential Optical Flow*, volume 2, chapter 14, pages 397–422. Academic Press, 1999.

[19] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean-shift. In *Proceedings of Conference on Computer Vision and Pattern Recognition CVPR 2000*, pages II: 142–149, Hilton Head, SC, USA, June 2000. IEEE Computer Society.

[20] V. Ramesh D. Comaniciu and P. Meer. Kernel-based optical tracking. *IEEE Transactions PAMI*, 25(5):564–577, May 2003.

[21] R. M. Haralick. Covariance propagation in computer vision. In *Proceedings of the ECCV Workshop on Performance Characteristics of Vision Algorithms*, Cambridge, UK, April 1996. http://www.vision.auc.dk/ hic/perf-proc.html.

[22] R. Marik, J. Kittler, and M. Petrou. Error sensitivity assessment of vision algorithms based on direct error propagation. In *Proceedings of the ECCV Workshop on Performance Characteristics of Vision Algorithms*, Cambridge, UK, April 1996. http://www.vision.auc.dk/ hic/perf-proc.html.

[23] C. Fermüller, R. Pless, and Y. Aloimonos. Statistical biases in optical flow. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 561–566, 1999.

[24] C. Fermüller and Y. Aloimonos. The statistics of optical flow: Implications for the processes of correspondence in vision. In *ICPR*, volume 1, pages 119–126, 2000.

[25] C. Fermüller, H. Malm, and Y. Aloimonos. Uncertainty in visual processes predicts geometrical optical illusions. Technical Report CR-TR-4251, Computer Vision and Mathematics at Lund Institute of Technology, Sweden, May 2001.

[26] J. L. Barron and M. Khurana. Determining optical flow for large motions using parametric models in a hierarchical framework. In *Vision Interface*, pages 47–56, May 1997.

[27] S. Uras, F. Girosi A. Verri, and V. Torre. A computational approach to motion perception. *Biological Cybernetics*, 60:79–97, 1988.

[28] J. L. Barron. Experience with 3D optical flow on gated MRI cardiac datasets. In 1^{st} *Canadian Conference on Computer and Robot Vision*, pages 370–377, 2004.

[29] J. H. Bergen, P. Anandan, K. J. Hamma, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, pages 237–252, May 1992.

[30] H. Spies. Certainties in low-level operators. In *Vision Interface*, pages 257–262, 2003.

[31] H. Spies and J. L. Barron. Evaluating certainties for image intensity differentiation for optical flow. In 1^{st} *Canadian Conference on Computer and Robot Vision*, pages 408–416, 2004.

[32] T. Lin and J. L. Barron. *Image Reconstruction Error for Optical Flow*, pages 269–290. Scientific Publishing Co., Singapore, (C. Archibald and P. Kwok, (eds.)), 1995.

[33] X. Tang, J. L. Barron, R. E. Mercer, and P. Joe. Tracking weather storms using 3D doppler radial velocity information. In *13th Scandinavian Conference on Image Analysis*, pages 1038–1043, 2003.