Practical guide to Optical Flow in "Vision with Direction"

Stefan Karlsson

November 2012

1 Introduction

This text is a brief practical guide to explaining the contents of the optical flow and motion estimation chapter 12 of Prof. Josef Biguns book, "Vision with Direction"[1] . It has been written for the benefit of the students of Halmstad University, and is in response to the questions that have arisen. Optical flow is the apparent 2D motion in an image sequence, a $R^3 \rightarrow R^2$ flow-field. This just means that every pixel in every video-frame will have a 2D vector associated with it(a bit like the gradient calculation in that sense). The vector tells the appearant motion of that pixel, as it moves from one frame to the next. Any region Ω where optical flow is to be estimated with full degrees of freedom, must contain linearly independent $\nabla I(\vec{x_i})$. In practical language, we must be sure that we do not have a region of the kind we find in the barber pole illusion (see fig 1 or google "barber pole illusion" for a proper animated version).

Figure 1: The barberpole illusion. A pattern of linear symmetry is wrapped around a cylinder, and rotated. The true motion is rotation left or right but the perceived motion by the observer is up or down

While many algorithms fail to deal with this so-called apperture problem, the method of the 3D structure tensor (TS) takes a rather unique approach. Instead of trying to always estimate two degrees of freedom of motion, the method of the TS detects first what kind of motion is present. It distinguishes between point motions, and line motions. If a textured region Ω, containing corners and well distributed structure is in motion, the TS method estimates 2 degrees of freedom of motion (displacement in x and y). If, on the other hand, Ω contains only linear symmetry (as is the case in the barber pole) then only

one degree of freedom is estimated: namely the degree of motion perpendicular to the edges. Most importantly: the TS method labels each flow vector in the image as belonging to one or the other type : Line motion or Point motion. The difference is visualized in fig 2, taken from the book.

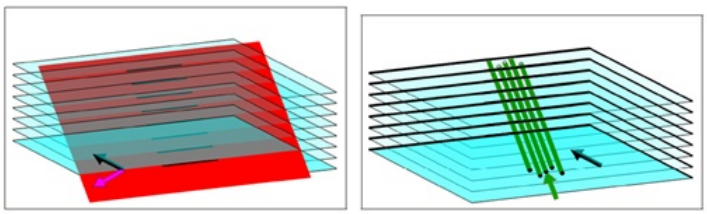

Figure 2: The distinction of motion of lines (left), and motion of points (right), as they appear going through the spatio-temporal volume. The y-axis in the figures (up-down) is time. z-axis(in-out) and x-axis(left-right) are the spatial coordinates of the frame.

Like the book, this guide will explain the Lucas and Kanade(LK) method (as they it is fundamentally connected to TS). Unlike the book, however, we will go through the LK method first.

2 Brief review:2D Structure Tensor

As we have already learnt from the book[1], chapter 10, the amount of linear symmetry is given by the eigenvalues of the 2D structure tensor. The 2D structure tensor is found in its matrix form as Eq. 10.32 in the book, and we write it here as:

$$S_{2D} = \begin{pmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{pmatrix}$$

where the elements are so-called "spectral moments", for example: $m_{11} = \int\int D_x f D_y f$. In practice, we will always consider a local neighbourhood Ω, and will always have discrete images. Therefore we can write here $m_{11} = \sum w I_x I_y$, where w is a smooth window function(Gaussian preferrably) covering the region Ω, I_x is the x-derivative of the discrete image I, and I_y is its y derivative. In general, we will write $m_{ijk} = \sum w I_x^i I_y^j I_t^k$. Assuming that we can move the smooth window function w around, we can consider different regions Ω, differing only where they have their window functions centered (at some $\vec{x}$, say). We will therefore write $m_{11}(\vec{x})$ to indicate positioning of Ω at $\vec{x}$. likewise we write:

$$S_{2D}(\vec{x}) = \begin{pmatrix} m_{20}(\vec{x}) & m_{11}(\vec{x}) \\ m_{11}(\vec{x}) & m_{02}(\vec{x}) \end{pmatrix}$$

3 Lucas and Kanade, using the 2D structure tensor for optical flow

The section on the Lucas and Kanade algorithm is 12.9 in the book, and it leads up to Equation 12.100, which is (in the notation of the book):

$$\mathrm{D}^T d = -\mathrm{D}^T \mathrm{D}\vec{v}$$

It is pointed out in text, but not written out explicitly that this can be interpreted as:

$$\begin{pmatrix} m_{101}(\vec{x}) \\ m_{011}(\vec{x}) \end{pmatrix} = -\begin{pmatrix} m_{20}(\vec{x}) & m_{11}(\vec{x}) \\ m_{11}(\vec{x}) & m_{02}(\vec{x}) \end{pmatrix} \vec{v}(\vec{x})$$

for $m_{101} = \sum wI_xI_t$ and $m_{011} = \sum wI_yI_t$. Writing

$$\vec{b}(\vec{x}) = \begin{pmatrix} m_{101}(\vec{x}) \\ m_{011}(\vec{x}) \end{pmatrix}$$

we have

$$\vec{b} = -S_{2D}\vec{v}(\vec{x})$$

The quantity $\vec{v}(\vec{x})$ is precisely what we want: the motion of pixels from one frame to the next. To get it, we invert the matrix S_{2D}, and get the full equation of the lucas and kanade method:

$$\vec{v}(\vec{x}) = -S_{2D}^{-1}\vec{b}$$

On the right hand side, we have a pure moments based expression, i.e. there is nothing there but different kinds of m_{ijk}. It is not always the case that S_{2D} can be inverted however. This is the case when we have "barber pole" for example. We must somehow deal with this, and one way is to check the determinant of S_{2D} before inverting(thereby simply skipping those regions alltogether). Another approach (a much better way) is to apply some form of "regularization" approach, but that is beyond the scope of this guide.

4 3D Structure Tensor

The 3D structure tensor first introduced in 1991 [2], is given as the straightforward generalization of the 2D version $S_{2D}(\vec{x})$:

$$S_{3D}(\vec{x}) = \begin{pmatrix} m_{200}(\vec{x}) & m_{110}(\vec{x}) & m_{101}(\vec{x}) \\ m_{110}(\vec{x}) & m_{020}(\vec{x}) & m_{011}(\vec{x}) \\ m_{101}(\vec{x}) & m_{011}(\vec{x}) & m_{002}(\vec{x}) \end{pmatrix}$$

where from now on $(\vec{x})$ denotes 3D coordinates (x, y, t). How to interpret this matrix is elaborated extensively in the book, and the algorithm for using it

for optical flow is not collected in any one place, but is rather spread out over the chapter 12.

Without giving any theoretical underpinnings, I will simply describe the algorithm practically here.

For every Ω of interest(indexed by $\vec{x}$):

Gather its eigen system: $\{\vec{v_i}, \lambda_i\}$ for $\lambda_1 \leq \lambda_2 \leq \lambda_3$. Where v_{ix} is the x component of the ith eigenvector, and $\gamma \in [0, 1)$ is a threshold).

- if $\lambda_1 + \lambda_2 + \lambda_3 = 0$, then Ω contains no structure (close to a constant gray-value).
- else if$\frac{\lambda_3 - \lambda_2}{\lambda_3 + \lambda_2} > \gamma$, then Ω contains line motion given by:

$$\vec{u_l} = \frac{v_{3t}}{v_{3x}^2 + v_{3y}^2} \begin{pmatrix} v_{3x} \\ v_{3y} \end{pmatrix}$$

- else if $\frac{\lambda_2 - \lambda_1}{\lambda_2 + \lambda_1} > \gamma$ then Ω contains point motion given by:

$$\vec{u_p} = \frac{1}{v_{1t}} \begin{pmatrix} v_{1x} \\ v_{1y} \end{pmatrix}$$

References

[1] J. Bigun. *Vision with Direction.* Springer, Heidelberg, 2006.

[2] J. Bigun, G.H. Granlund, and J. Wiklund. Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE-PAMI*, 13(8):775–790, 1991.