Practical guide to Optical Flow in "Vision with Direction"

Stefan Karlsson

November 2013

1 Introduction

This text is a brief ***practical*** guide to introduce optical flow. Its based on chapter 12 of Biguns "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. It is intended as supplementary to the exercises in real-time optical flow.

Optical flow is the apparent 2D motion in an image sequence. Every position, in every video-frame will have a 2D vector associated with it (a bit like the 2D gradient calculation in that sense). The vector tells the best estimate of motion of that position, as it moves from one frame to the next. Any region Ω where optical flow is to be estimated reliably must contain "nice texture"[1]. In this context "bad textures" are regions that have either...

- constant gray value(no information), or..
- regions of linear symmetry(information in only one direction).

We can say that "nice textures" are those with linearly independent 2D gradients $\nabla I(\vec{x_i})$. 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).

Most of the time, the question of whether a region is "nice" or "bad", depends on how big we make it. Making a region bigger, makes it more likely to gather observations from the image that provide new directional information. Making a region bigger will have the drawback of reducing the resolution of the resulting optical flow, so a comprimise is necessary:

- big region $\rightarrow$ better data,
- small region $\rightarrow$ better resolution.

[1] with "nice" we will mean textures with well distributed spatial structure. If that does not make sense to you right now, just read on

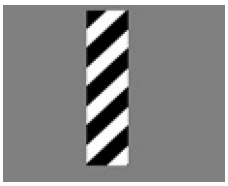

Figure 1: The barberpole illusion. A pattern of linear symmetry is wrapped around a cylinder, and rotated. The true motion is rotation left or right but the perceived motion by the observer is up or down

This phenomenon is often referred to as the aperture problem (the aperture meaning the shape of the region we do the estimate over).

This guide will explain the Lucas and Kanade(LK) method as well the method of the 3D structure tensor(TS) in practical terms. We will go through the LK method first, starting with a brief review of the 2D structure tensor.

2 Brief review:2D Structure Tensor

Whether the region is "nice" or "bad" is given by the 2D structure tensor, defined 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: $m_{11} = \int\int D_x f D_y f$. A region is "nice"(well distributed structure) if both eigenvalues are large. This is equivalent to saying that S_{2D} is "well conditioned". In practical terms, it means that S_{2D} can be inverted with no problems. If S_{2D} can *not* be inverted, its because we have a "bad" texture, and the 2 cases are distinguished naturally as:

- constant gray value(both eigenvalues of S_{2D} are zero)
- linear symmetry(one eigenvalue of S_{2D} is zero).

In practice, we will always consider a local neighborhood 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 (e.g. Gaussian) 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 Lucas and Kanade algorithm can be phrased entirely in terms of moments[2] as:

$$\vec{v} = \begin{pmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{pmatrix}^{-1} \begin{pmatrix} m_{101} \\ m_{011} \end{pmatrix}$$

For clarity, we can introduce a vector $\vec{b}$,

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

and remind ourselves what the 2D structure tensor is:

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

yielding the expression:

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

There are cases when S_{2D} can not be inverted! This happens for the "bad textures" as described earlier (the case when we have the "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 altogether). Another approach (a much better way) is to apply some form of spatial "regularization"[3], but that is beyond the scope of this guide.

4 3D Structure Tensor

While many algorithms fail to deal with the so-called aperture 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.

[2]The Lucas and Kanade algorithm is derived properly in this fashion in section 12.9 in the book[1], and it leads up to Equation 12.100, which contains the moments we use here

[3]simplest way would be to use "Tikhnov regularization", but much more advanced so called variational approaches exist

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). Without theoretical underpinnings[4] one algorithm is given 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:

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

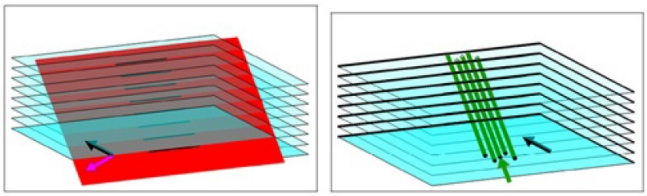

Figure 2: The distinction of motion of lines (left), and motion of points (right), as they appear going through the spatio-temporal volume. The y-axis in the figures (up-down) is time. z-axis(in-out) and x-axis(left-right) are the spatial coordinates of the frame (taken from [1])

[4]How to interpret this matrix is elaborated extensively in[1]. However, a practical algorithm for using it for optical flow is not collected in any one place, but is rather spread out over chapter 12.

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.