TP3: Optical Flow

Loic Simon Chaohui Wang Panagiotis Koutsourakis

16 May 2008

1 Theory

The aim of this lab is to compute the optical flow of an image under the assumption that the intensity of the moving pixels remains constant. In other words, we are considering a sequence of images $(I(t)_{\{t=1\cdots T\}})$ taken from a deformable scene which color doesn't change along time. Because of this deformation, pixels are moving along the sequence, and we are trying to figure out what is the path of each pixel $\mathbf{x}_0(t) = (x_0(t), y_0(t))$ given its initial position $\mathbf{x}_0$. The hypothesis of constant intensity can be expressed as follows:

$$I(\mathbf{x}_0(t), t) = I(\mathbf{x}_0, 0) \tag{1}$$

Taking the derivative of the former equation we get,

$$0 = \frac{d}{dt} I(x_0(t), t) = \nabla I.u + I_t \tag{2}$$

where $\mathbf{u} = (u_x, u_y) = \frac{d\mathbf{x}_0}{dt} = d\mathbf{x}_0$ is the displacement of the pixel. Note that equation **??** is not sufficient to determine $\mathbf{u}$ (which is the unknown quantity we're trying to find out). In this kind of under-constrained problem, a common way to remove the indeterminacy is to express this problem as an optimization one. One possible choice (among others) is to minimize the following energy:

$$E(\mathbf{u}) = \sum_{\mathbf{x}} g_{\mathbf{x}_0}(\mathbf{x}) \left[\nabla I(\mathbf{x}, t).\mathbf{u} + I_t(\mathbf{x}, t)\right]^2 (\mathbf{x}) \tag{3}$$

(Here $g_{\mathbf{x}_0}$ is a Gaussian filter centered at $\mathbf{x}_0$).
The minimum is achieved when the derivative of E vanishes :

$$\frac{\partial E(u_x, u_y)}{\partial u_x} = \sum_{\mathbf{x}} g_{\mathbf{x}_0}(\mathbf{x}) \left[u_x I_x^2 + u_y I_x I_y + I_x I_t\right] (\mathbf{x}) = 0 \tag{4}$$

$$\frac{\partial E(u_x, u_y)}{\partial u_y} = \sum_{\mathbf{x}} g_{\mathbf{x}_0}(\mathbf{x}) \left[u_y I_y^2 + u_x I_x I_y + I_y I_t\right] (\mathbf{x}) = 0 \tag{5}$$

where, I_x, I_y and I_t represent respectively the derivative of the image sequence relatively to x, y and t. This linear system can be expressed as follows:

$$M_{\mathbf{x}_0}\mathbf{u} = b_{\mathbf{x}_0} \tag{6}$$

where,

$$M_{\mathbf{x}_0} = \begin{pmatrix} \sum g_{\mathbf{x}_0} I_x^2 & \sum g_{\mathbf{x}_0} I_x I_y \\ \sum g_{\mathbf{x}_0} I_x I_y & \sum g_{\mathbf{x}_0} I_y^2 \end{pmatrix} \quad (7)$$

$$b_{\mathbf{x}_0} = -\begin{pmatrix} \sum g_{\mathbf{x}_0} I_x I_t \\ \sum g_{\mathbf{x}_0} I_y I_t \end{pmatrix} \quad (8)$$

$$(9)$$

2 Computation

You have all the following files.

- opticalflow.cpp: main function.
- libEcp.h: auxiliary functions among which the display functions and those computing the flow. You will have to modify the function computeOpticalFlow.
- EcpException.h

The data sets are available in the directories named Dataset* and you can load them using the loadImages instruction called in the main function.

The computations you'll have to do consist in :

1. evaluating the derivative images I_x,I_y and I_t for each image of the sequence.
2. computing their pairwise product $(I_x^2, I_x I_y, \dots)$.
3. convolving them with a Gaussian filter (explain where this convolution is involved).
4. use the previous results to compute $M_{\mathbf{x}_0}$ and $b_{\mathbf{x}_0}$ for each pixel $\mathbf{x}_0$ and for all t.

2.1 derivative and convolution

Every function in this lab is defined on a discrete grid (pixel location and time are discrete variables). Therefore, derivative have to be approximated with a finite difference scheme.

For instance I_t can be computed in the simple manner that follows

$$I_t(x, y, t) = \frac{I(x, y, t+1) - I(x, y, t+)}{1} \quad (10)$$

or in a more sophisticated way.

For the same reason you have to use the discrete convolution formula:

$$f * g(x_0, y_0) = \sum_{x,y=-\infty}^{\infty} f(x, y) g(x_0 - x, y_0 - y) \quad (11)$$

(in fact this is already implemented in CImg)

2.2 Optical flow computation

To finish, after you computed $M_{\mathbf{x}_0}$ and $b_{\mathbf{x}_0}$, you have to invert the system **??** in order to obtain $u, \forall \mathbf{x}_0$.

2.3 Visualization

You can eventually visualize the optical flow using the visualizeOpticalFlow functions.