TP7: Optical Flow

Panagiotis Koutsourakis Mickael Savinaud

May 5, 2009

1 Introduction

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer (an eye or a camera) and the scene. 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.

2 Theory

We consider a sequence of images $(I(t)_{\{t=1\cdots T\}})$ taken from a real 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,

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

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 2 is not sufficient to determine $\mathbf{u}$ (which is the unknown quantity we're trying to compute). In this kind of under-constrained problems, a common way to remove the indeterminacy is to express the 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 :

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

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

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:

$$M_{\mathbf{x}_0}\mathbf{u} = b_{\mathbf{x}_0} \quad (6)$$

where,

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

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

$$(9)$$

3 Work To Do

You should send as back a report containing the code you wrote, the results you obtained (stating explicitly what parameters you used) and your observations. You should perform your experiments in the two sequences that is provided to you. The report should be handed back by 18/05. Afterwards, the solutions are going to be available and no report will be accepted. We divise your work in three parts.

3.1 Optical Flow between 2 images

In this section, the computations you'll have to do consist in:

1. Evaluate the derivative images I_x,I_y and I_t for each image of the sequence. (cf previous Labs)
2. Compute their pairwise product ($I_x^2, I_x I_y, \ldots$).
3. Convolve them with a Gaussian filter. (cf. lab1).
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.
5. inverting the system 6 in order to obtain $u, \forall \mathbf{x}_0$.

You'll have to verify your results with a method that applies the optical flow to I_t and compare with I_{t+1}. To compare we give you some methods in the visualization section. You must modify value of the gaussian filter to observe the impact of this parameter of the optical flow. Include in your report, 2 or 3 examples values.

3.2 Optical Flow on a sequence of images

You must expand the previous computation for a list of images provided by the loadImages methods. To display your results, you should use the visualizeOpticalFlow method. Use the taxi sequence and optical flow methods to show the displacement of the objects in the scene.

3.3 Implementations Outlines

You have all the following files.

- opticalflow.cpp: main function with a skeleton. It is strongly recommended to use the scheme provided in this file for the solution.
- libEcp.h: auxiliary functions among which the display functions and those computing the flow. You will have to modify the function *computeOpticalFlow()* which must compute optical flow for a sequence of images.
- EcpException.h

The data sets are available in the directories named taxi et taxi2 and you can load them using the loadImages instruction called in the main function.

3.4 Visualization

You can visualize your results with the above functions:

```
1) visualizeOpticalFlow()
2) displayImageSequence()
3) displayAndCompare2Images()
4) warpImage()
```

4 Bonus

If you use the taxi2 sequence, you can observe that large displacements are not well recovered by you optical flow method. Therefore you can use a gaussian pyramid to recover each different level of velocity in the optical flow.

The Gaussian pyramid is computed as follows. The original image is convolved with a Gaussian kernel. The resulting image is a low pass filtered version of the original image. The cut-off frequency can be controlled using the parameter σ.

In this section, the computations you'll have to do consist in:

1. Compute images of the gaussian pyramid at one level.
2. Compute optical flow for this level with your optical flow method.

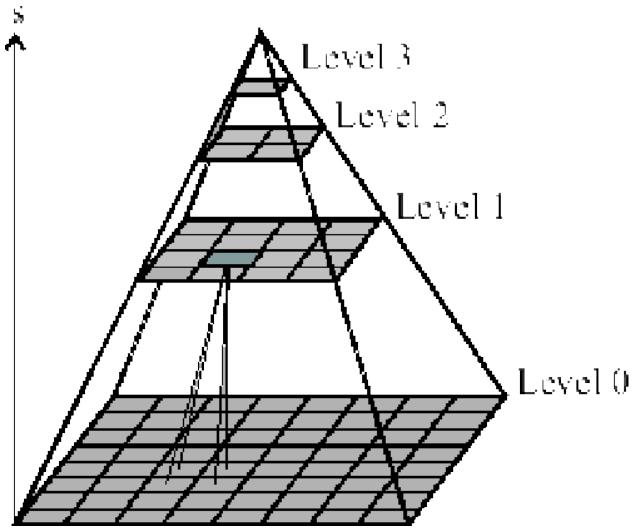

Figure 1: The filtered images stacked one on top of the other form a tapering pyramid structure, hence the name.

3. Update images for the next level of the pyramid with a methods that use your warpImage method.

4. Do this previous step for 3 or 4 level of the pyramid.

5. Visualize the final optical flow for the sequence.