ECE4580 Homework #11

Due: Apr. 20, 2017

Problem 1. (25 pts) Implement the gradient descent algorithm discussed in class, which uses the energy functional from Horn and Schunck. Recall that the Horn and Schunck optical flow energy is

$$\mathcal{E} = \int_D \left[(\nabla I \cdot X + I_t)^2 + \alpha^2 \left(||\nabla u||^2 + ||\nabla v||^2 \right) \right] \overrightarrow{\mathrm{d}x} \tag{1}$$

whose gradient descent minimization is the discretization of:

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}t} &= -\left((\nabla I \cdot X + I_t) I_x - \alpha^2 \Delta u \right), \\ \frac{\mathrm{d}v}{\mathrm{d}t} &= -\left((\nabla I \cdot X + I_t) I_y - \alpha^2 \Delta v \right), \end{aligned} \tag{2}$$

where X is the vector field given by (u, v). It should be possible to use the Horn and Schunck discretization of the differentials, just be careful to define the Laplacian, $\Delta(\cdot)$, correctly (much like in the earlier homework where the boundary conditions mattered).

Code stubs and data files are included with the homework assignment.

Problem 2. (40 pts) Implement the Horn and Schunck optical flow solver. This is essentially the Gauss-Siedel iterative solver as discussed in class:

$$\begin{aligned} u^{n+1} &= \bar{u} - I_x \left(I_x \bar{u} + I_y \bar{v} + I_t \right) / \left(\kappa \alpha^2 + I_x^2 + I_y^2 \right), \\ v^{n+1} &= \bar{v} - I_y \left(I_x \bar{u} + I_y \bar{v} + I_t \right) / \left(\kappa \alpha^2 + I_x^2 + I_y^2 \right), \end{aligned}$$

where $\bar{u}$, $\bar{v}$ are the averaging terms from the discrete Laplacian operator, and I_x, I_y, and I_t are the partial derivatives. Make sure to use the convolutions that Horn and Schunck utilized. Below, will be a summary of the main convolution definitions. If you seek more understanding, then please refer to the implementation in the Horn book if you are unsure, or in the original paper of Horn and Schunck (see Handouts section of t-square).

Supposing that I_1 and I_2 were the two sequential images from animage sequence, then the different partial derivatives are defined by:

$$\begin{aligned} I_t = \frac{\partial I}{\partial t} &= K_t * I_2 - K_t * I_1, \quad \text{where } K_t = \frac{1}{4} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \\ I_x = \frac{\partial I}{\partial x} &= K_x * I_1 + K_x * I_2, \quad \text{where } K_x = \frac{1}{4} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \\ I_y = \frac{\partial I}{\partial y} &= K_y * I_1 + K_y * I_2, \quad \text{where } K_y = \frac{1}{4} \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \end{aligned}$$

where the convolution kernels are `imfilter` kernels and the y-coordinate is presume to increase downwards (to mirror the way image axes are done when an image is plotted in Matlab). Also for u (and likewise for v), the Laplacian Horn and Schunck utilized is:

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \kappa u + \kappa A * u = \kappa u + \kappa \bar{u}, \text{where } \kappa = 3, \text{ and } A = \begin{bmatrix} \frac{1}{12} & \frac{1}{6} & \frac{1}{12} \\ \frac{1}{6} & 0 & \frac{1}{6} \\ \frac{1}{12} & \frac{1}{6} & \frac{1}{12} \end{bmatrix}.$$

The main thing to note is that the partials are actually the partials of both images averaged together. The differentials for each image can be performed either as shifts or as convolutions, with the appropriate boundary conditions. The paper gives the convolution stencils for different pieces. Furthermore, please be sure to have the appropriate boundary conditions (of zero) as required by the derivation (see lecture notes).

Code stubs are included with the homework assignment to assist with the organization and structure of the code. The data files for the image sequences should also be available with the homework assignment.

Problem 3. (5 pts) Comment on the differences between the linear iterative solver and the gradient descent solver. What do you notice about the parameter sensitivity of the two methods? Recall that they are both trying to minimize the same energy functional.

Problem 4. (20 pts) Move on to the next activity of the learning module. The group submission should reflect the work of the group, and should also be submitted individually with the name of your partner in the document. If submitting video or links to video for the pair, then only one member need to do so, while the other member should just note as much. The prior expectaction for deliverables continues to hold.

Optical Flow. There are many assumptions that go into the optical flow derivation, meaning that you will have to be careful about the implementation. Below are a few insights into the process to help you get on your way.

The optical flow constraint relies on the total time derivative of the brightness constancy assumption, which is essentially like linearizing a system. Consequently, there is a preferred operational range for the image and its derivatives. You will have to play around with the dynamic range of the image to get the optical flow code to work out correctly (it will have to be reduced). Plus, there are the parameters of the iterative solver and the gradient descent algorithm.

The fact that the image is discrete and subject to noise also makes things difficult. In order to regularize the data, it is common to smooth the images prior to computing the optical flow. In your code, you will need to implement this smoothing. Convolution with a Gaussian is the typical approach, but you can use any other smoothing technique of your choice so long as you document it in the code.

You will have to play around with the algorithm parameters and the dynamic range of the images to find the right settings. Also, I know I mentioned it once before, but don't forget the boundary conditions on the optical flow vector field.

Auxiliary Files. There is an m-file called `playmatrix` that will animate a 3D matrix representing a grayscale image sequence. There is also an m-file called `optflow_check` that applies the optical flow transformation to a given (source) image and compares it to the target image. If the optical flow is roughly in the right direction, then the transformed source image should approximate the target image. The Matlab data file consists of short grayscale image sequences to be processed. The `flowmeXX` m-files are scripts to invoke the gradient descent and the Horn and Schunk methods.