Image Understanding – Assignment #5

Optical Flow

1. Part I- Mathematical derivation

$$F(u,v) = \sum_{pixels\ p} \left\| I_x(p)u(p) + I_y(p)v(p) + I_t(p) \right\|^2 + \lambda \sum_{p,q\ neighbors} (u(p)-u(q))^2 + (v(p)-v(q))^2$$

1.1 $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}(\boldsymbol{p})}$

$$\frac{\partial F}{\partial u(p)} = \mathbf{2}\left[I_x(p)u(p) + I_y(p)v(p) + I_t(p)\right]I_x(p) + \lambda \sum_{q\ neighbors} \mathbf{2}[u(p)-u(q)]$$

In this implementation, 8 neighboring pixels are used. Let the coordinate of p is *(x, y)* , then the right part (smoothing part) of equation above becomes:

$$\lambda \sum_{q\ neighbors} \mathbf{2}[u(p)-u(q)]$$
$$= 2\lambda[8 * u(x,y) - [u(x-1,y-1) + u(x-1,y) + u(x-1,y+1) + u(x,y-1) + u(x,y+1) + u(x+1,y-1) + u(x+1,y) + u(x+1,y+1)]]$$

Let $\bar{u} = [u(x-1,y-1) + u(x-1,y) + u(x-1,y+1) + u(x,y-1) + u(x,y+1) + u(x+1,y-1) + u(x+1,y) + u(x+1,y+1)]/8$

Then, $\lambda \sum \mathbf{2}[u(p)-u(q)] = 16\lambda(u(p)-\bar{u})$

Therefore,

$$\frac{\partial F}{\partial u(p)} = \mathbf{2}\left[I_x(p)u(p) + I_y(p)v(p) + I_t(p)\right]I_x(p) + 16\lambda(u(p)-\bar{u})$$

1.2 $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{v}(\boldsymbol{p})}$

$$\frac{\partial F}{\partial v(p)} = \mathbf{2}\left[I_x(p)u(p) + I_y(p)v(p) + I_t(p)\right]I_y(p) + \lambda \sum_{q\ neighbors} \mathbf{2}[v(p)-v(q)]$$

Similarly, 8 neighboring pixels are used, the equation above can be rewritten as:

$$\frac{\partial F}{\partial v(p)} = \mathbf{2}\left[I_x(p)u(p) + I_y(p)v(p) + I_t(p)\right]I_y(p) + 16\lambda(v(p)-\bar{v})$$

$$\bar{v} = [v(x-1,y-1) + v(x-1,y) + v(x-1,y+1) + v(x,y-1) + v(x,y+1) + v(x+1,y-1) + v(x+1,y) + v(x+1,y+1)]/8$$

1.3 **Solving** $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}(\boldsymbol{p})} = \mathbf{0}$ and $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{v}(\boldsymbol{p})} = \mathbf{0}$

Let $\frac{\partial F}{\partial u(p)} = 0$ and $\frac{\partial F}{\partial v(p)} = 0$, then:

$$\begin{cases} \mathbf{2}\left[I_x(p)u(p) + I_y(p)v(p) + I_t(p)\right]I_x(p) + 16\lambda(u(p)-\bar{u}) = 0 \\ \mathbf{2}\left[I_x(p)u(p) + I_y(p)v(p) + I_t(p)\right]I_y(p) + 16\lambda(v(p)-\bar{v}) = 0 \end{cases}$$

Let λ = 8λ, the equations above can be written as:

$$\begin{cases} \left[I_x(p)u(p) + I_y(p)v(p) + I_t(p)\right]I_x(p) + \lambda(u(p)-\bar{u}) = 0 \\ \left[I_x(p)u(p) + I_y(p)v(p) + I_t(p)\right]I_y(p) + \lambda(v(p)-\bar{v}) = 0 \end{cases}$$

Rewrite those functions as the equations of ***u*** and ***v***.

$$\begin{cases} [I_x^2(p)+\lambda]u(p)+I_x(p)I_y(p)v(p)=\lambda\bar{u}-\ I_x(p)I_t(p) \\ [I_y^2(p)+\lambda]v(p)+I_x(p)I_y(p)u(p)=\lambda\bar{v}-\ I_y(p)I_t(p) \end{cases}$$

Solve ***u*** and ***v*** out of those functions:

$$\begin{cases} u(p)=\dfrac{[I_y^2(p)+\lambda]\bar{u}-I_x(p)I_y(p)\bar{v}-I_x(p)I_t(p)}{I_x^2(p)+I_y^2(p)+\lambda} \\ v(p)=\dfrac{[I_x^2(p)+\lambda]\bar{v}-I_x(p)I_y(p)\bar{u}-I_y(p)I_t(p)}{I_x^2(p)+I_y^2(p)+\lambda} \end{cases}$$

Rewrite the above functions:

$$\begin{cases} u(p)=\bar{u}-\dfrac{I_x^2(p)\bar{u}+I_x(p)I_y(p)\bar{v}+I_x(p)I_t(p)}{I_x^2(p)+I_y^2(p)+\lambda} \\ v(p)=\bar{v}-\dfrac{I_y^2(p)\bar{v}+I_x(p)I_y(p)\bar{u}+I_y(p)I_t(p)}{I_x^2(p)+I_y^2(p)+\lambda} \end{cases}$$

Jacobi Method will be used to solve these two equations. *u(p)* and *v(p)* will be set as zero matrices initially. Then in each iteration, $\bar{u}$ and $\bar{v}$ will be calculated. Then the value of each pixel in *u* and *v* will be updated by using equations above.

2. Part II- Implementation

2.1 Organization of the code

The main function is HSOpticalFlow.m. The information of this function is shown in Table 2.1.

Table 2.1 Description of HSOpticalFlow Function

Name	HSOpticalFlow		
Description	Calculate the optical flow based on 2 input images and the given parameter lambda		
	Name	Type	Description
Input	*frame1*	matrix	the matrix storing the first input image
	frame2	matrix	the matrix storing the second input image
	lambda	double	parameter for smoothing part
Output	*u*	matrix	the matrix storing the x component of optical flow
	v	matrix	the matrix storing the y component of optical flow

Also, there are several helper functions which are called by the main function. Below is the description of those functions.

Table 2.2 Description of Derivatives Function

Name	Derivatives		
Description	Calculate the derivatives Ix, Iy and It of two input images		
Input	Name	Type	Description
	frame1	matrix	the matrix storing the first input image
	frame2	matrix	the matrix storing the second input image
Output	*Ix*	matrix	the matrix storing the x derivatives of each pixel
	Iy	matrix	the matrix storing the y derivatives of each pixel
	It	matrix	the matrix storing the t derivatives of each pixel

The given input image is smoothed by gaussian filter. The GaussianSmooth function and Convolution Function are the exactly same functions implemented in Assignment 1.

2.2 Derivative Calculation

Based on Horn and Schunck`s paper, I_x, I_y and I_t are calculated by using neighboring pixels from 2 input images. The formulas used in this implementation is shown below:

$$I_x(i,j,t) = \frac{1}{4}[I(i,j+1,t) - I(i,j,t) + I(i+1,j+1,t) - I(i+1,j,t) + I(i,j+1,t+1) - I(i,j,t+1) + I(i+1,j+1,t+1) - I(i+1,j,t+1)]$$

$$I_y(i,j,t) = \frac{1}{4}[I(i+1,j,t) - I(i,j,t) + I(i+1,j+1,t) - I(i,j+1,t) + I(i+1,j,t+1) - I(i,j,t+1) + I(i+1,j+1,t+1) - I(i,j+1,t+1)]$$

$$I_t(i,j,t) = \frac{1}{4}[I(i,j,t+1) - I(i,j,t) + I(i+1,j+1,t+1) - I(i+1,j+1,t) + I(i,j+1,t+1) - I(i,j+1,t) + I(i+1,j,t+1) - I(i+1,j,t)]$$

2.3 Optical Flow Calculation Method

The basic idea is to use the equations in section 1.3 and solve them by Jacobi method iteratively.

Step 1: Set the initial value of matrix ***u*** and ***v*** by zero matrices.

Step 2: In each iteration, calculate $\bar{u}$ and $\bar{v}$ by convolving ***u*** and ***v*** with a mean matrix. In this implementation, 8 neighboring pixels are used and the mean matrix is shown as below based on Horn and Schunck`s paper:

$$\text{Mean} = \begin{bmatrix} 1/12 & 1/6 & 1/12 \\ 1/6 & 0 & 1/6 \\ 1/12 & 1/6 & 1/12 \end{bmatrix}$$

Step 3: Update ***u*** and ***v*** by using the equations in section 1.3.

Step 4: After several iterations, ***u*** and ***v*** will be stabled.

2.4 Result of Optical Flow Calculation

Test code:

```
I = imread('seq1-image1.pgm');       %read the first input image and store it in matrix I
J = imread('seq1-image2.pgm');       %read the second input image and store it in matrix J
HSOpticalFlow (I,J,1);               %calculate the optical flow with given parameter
```

Results of Optical Flow of seq1 with different parameters are shown as below.

Fig 2.1 Magnitude of optical flow with lambda = 0.1

Fig 2.2 Magnitude of optical flow with lambda = 1

Fig 2.3 Magnitude of optical flow with lambda = 10

Fig 2.4 Magnitude of optical flow with lambda = 100

Results of Optical Flow of seq2 with different parameters are shown as below.

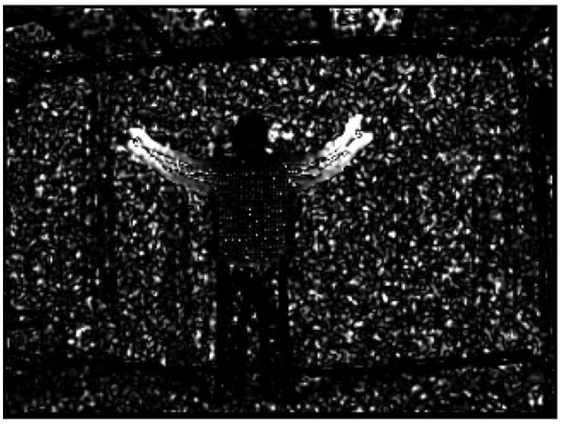

Fig 2.5 Magnitude of optical flow with lambda = 0.1

. Fig 2.6 Magnitude of optical flow with lambda = 1

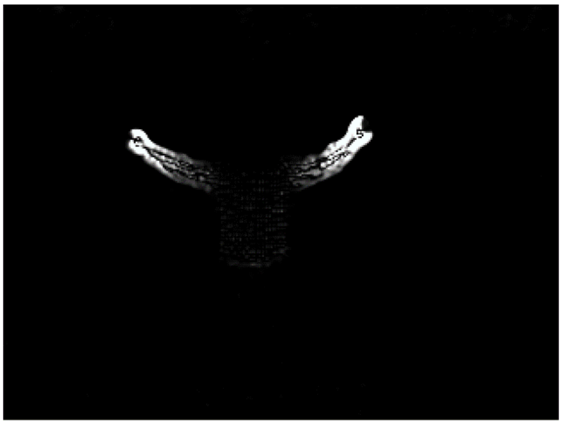

Fig 2.7 Magnitude of optical flow with lambda = 10

. Fig 2.8 Magnitude of optical flow with lambda = 100

2.5 Influence of Parameter Lambda

Parameter Lambda effects the weight of the smoothing part in $F(u, v)$. Bigger lambda leads to smaller weight of gradient of $\boldsymbol{u}$ and $\boldsymbol{v}$.

When lambda becomes bigger, the magnitude of optical flow of each pixel becomes smaller, but the direction becomes more accurate. When lambda keeps increasing, the optical flow of pixels with small movement will be wiped out.

I think lambda = 1 seems a good parameter, it can preserve details and wipe out some noise.