Optical Flow Algorithms

Jacky Baltes

University of Manitoba

Email: jacky@cs.umanitoba.ca

Web: www.cs.umanitoba.ca/~jacky

October 21, 2003

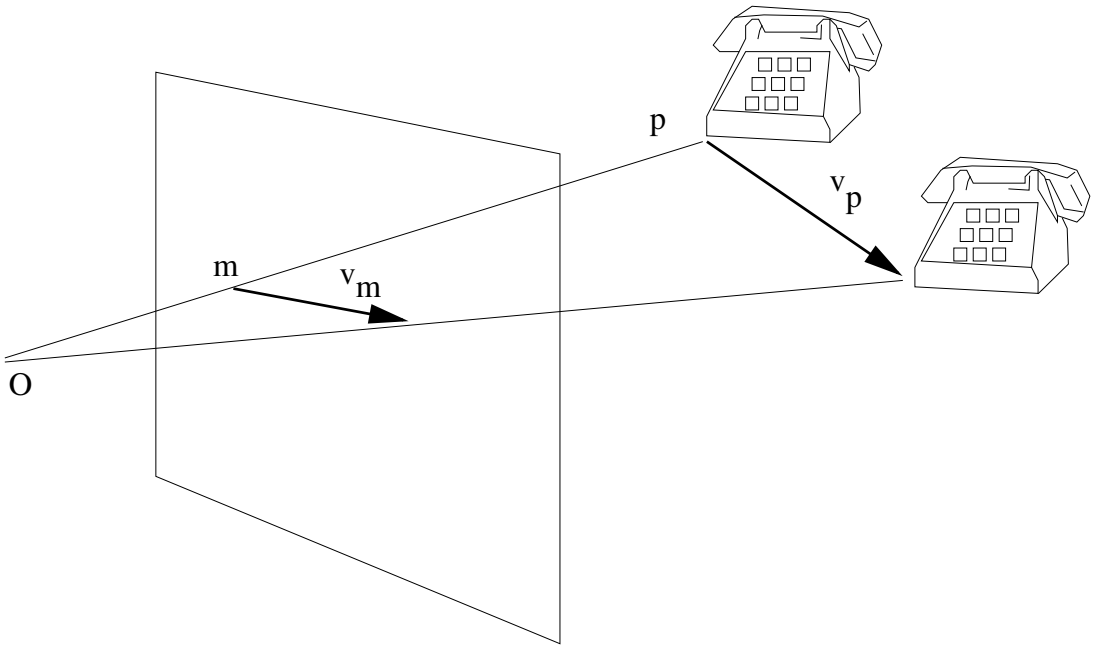
Motion Fields

- Motion is an important queue in computer vision. It is used in tracking, structure from motion, 3D reconstruction, video compression.
- Several issues
  1. 2D and 3D motion representations
  2. calculating 2D motion
  3. **inferring/estimating 3D motion**
  4. structure and motion

2D Motion Fields

- A velocity vector is associated with every point in the image
- Collection of these vectors is a 2D motion field

Image Projection



- P is a 3D point with coordinates $(X, Y, Z)^T$
- m is the image projection $(x, y)^T$
- $P = Z\hat{m}$
- differentiate with respect to time (to get velocities)

$$\frac{dp}{dt} = \frac{dZ}{dt}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + Z\begin{bmatrix} dx/dt \\ dy/dt \\ 0 \end{bmatrix}$$

- that is

$$\dot{p} = \dot{Z}\hat{m} + Z\dot{\hat{m}}$$

2D Motion

- replace with velocities

$$V_p = (V_p^T k)\hat{m} + Z v_m$$

- solve for v_m

$$v_m = \frac{1}{Z}(V_p - (V_p^T k)\hat{m})$$

Optical Flow

- We can not observe the motion field directly
- We can observe images only and see how points in the image move
- Optical flow != motion field
- Example: a lighted sphere moves, light around the sphere moves
- Underlying assumption: optical flow is similar to motion field

Optical Flow Constraint Equation (OFCE)

- $I(x, y, t)$ is the image
- $m = (x, y)^T$ is a image pixel
- velocity of an image pixel

$$v_m = \dot{m} = [v_x, v_y]^T = \begin{bmatrix} dx/dt \\ dy/dt \end{bmatrix}$$

- Assumption: Intensity of pixel m is the same during dt

$$I(x + v_x dt, y + v_y dt, t + dt) = I(x, y, t)$$

Optical Flow Constraint Equation (OFCE)

- Use Taylor expansion to approximate differential

$$I(x,y,t)+\frac{\delta I}{\delta x}v_x dt+\frac{\delta I}{\delta y}v_y dt+\frac{\delta I}{\delta t}dt+... = I(x,y,t)$$

- Cancel $I(x,y,t)$, we have

$$\frac{\delta I}{\delta x}v_x + \frac{\delta I}{\delta y}v_y + \frac{\delta I}{\delta t} = 0$$

- this is the optical flow equation

$$\nabla I \cdot v_m + \frac{\delta I}{\delta t} = 0$$

- where $\nabla I = [\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}]^T$ is the image gradient at pixel m.

Image Gradient

- where $\nabla I = [\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}]^T$ is the image gradient at pixel m.
- This can be calculated from the image
- Also, $\frac{\delta I}{\delta t}$ can be calculated from the image
- However, we need to determine v_x and v_y and only have a single equation
- Aperture problem

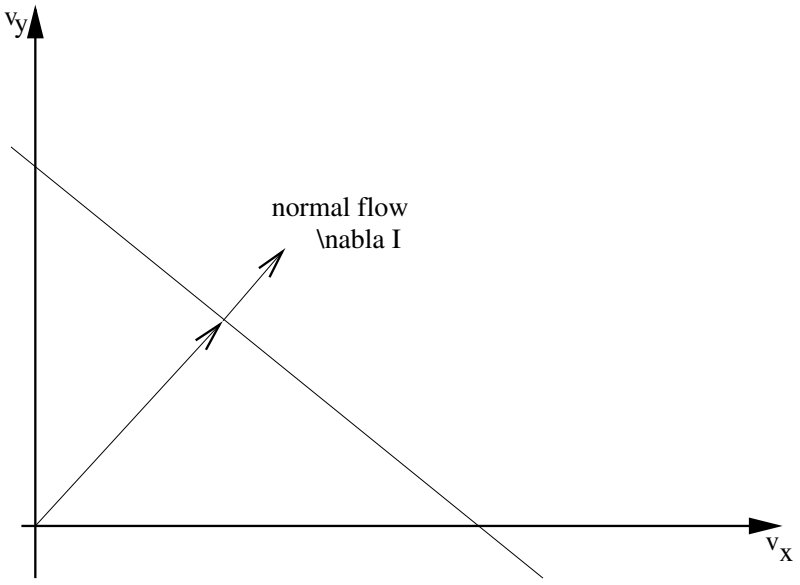


Image Gradient

- How to calculate the image gradient ∇I?
- Later on, we need $\frac{\delta I}{\delta x}$ and $\frac{\delta I}{\delta y}$
- Use discrete approximation of the gradient from the image
- Simple: $\frac{\delta I(x,y)}{\delta x} = (I(x+1,y) - I(x-1,y))/2$
- Simple: $\frac{\delta I(x,y)}{\delta y} = (I(y+1,y) - I(y-1,y))/2$
- Implemented as convolution masks: $[-1, 0, +1]$

Image Gradient

- These derivatives are usually quite noisy, since they are based and emphasize the difference between neighboring pixels.
- Average a number of rows for $\frac{\delta I(x,y)}{\delta x}$
- Convolution mask

$$\begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$$

Common Convolution Masks

- Roberts $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
- Prewitt $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- Sobel $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

Laplacian Methods

- Method for edge detection. Calculate gradient and threshold. In effect we are looking for extremas in I'.
- Another method for determining edges and the gradient it to look for 0 crossings in I".
- Need to apply a blur mask for noise reduction first, since otherwise much too noisy.
- Approximate 2nd order derivative in x direction

$$\nabla^2 I(x,y) = (\nabla I(x+1,y) - \nabla I(x-1,y))/2$$

- Some examples of Laplace operators

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Horn Schunck Method

- A global method.
- Assumes that the optical flow field varies smoothly. Variation of the optical flow field can not be too big.
- Use a global smoothness constraint. Based on the gradient of the optical flow ∇v_x and ∇v_y.
- This technique is called regularization
- Isotropic stabilizer $||\nabla v_x||^2 + ||\nabla v_y||^2$
- Measure of un-smoothness (Smoothness error)

$$\begin{aligned} e_s &= \int\int (||\nabla v_x||^2 + ||\nabla v_y||^2) dxdy \\ &\int\int (\frac{\delta v_x}{\delta x})^2 + (\frac{\delta v_x}{\delta y})^2 + (\frac{\delta v_y}{\delta x})^2 + (\frac{\delta v_y}{\delta y})^2 dxdy \end{aligned}$$

- Error of the optical flow $e_c = \int\int (\nabla I \cdot v_m + \frac{\delta I}{\delta t}^2 dxdy$
- Minimize the error $e = e_c + \lambda e_s$

$$\begin{aligned} e &= e_c + \lambda e_s \\ &= \int\int (\nabla I \cdot v_m + \frac{\delta I}{\delta t})^2 + \lambda(||\nabla v_x||^2 + ||\nabla v_y||^2) dx \end{aligned}$$

Horn Schunck Method Discretization

- Given the four neighbors of a pixel, the smoothness function can be written as:

$$
\begin{aligned}
s(i,j) \quad = \quad & \frac{1}{4}[(v_x(i,j) - v_x(i-1,j))^2 \\
& +(v_x(i+1,j) - v_x(i,j))^2 \\
& +(v_x(i,j+1) - v_x(i,j))^2 \\
& +(v_x(i,j) - v_x(i,j-1))^2 \\
& +(v_y(i,j) - v_y(i-1,j))^2 \\
& +(v_y(i+1,j) - v_y(i,j))^2 \\
& +(v_y(i,j+1) - v_y(i,j))^2 \\
& +(v_y(i,j) - v_y(i,j-1))^2]
\end{aligned}
$$

Horn Schunck Method Discretization

- The error of the optical flow is
  $c(i,j) = [\frac{\delta I}{\delta x} v_x(i,j) + \frac{\delta I}{\delta y} v_y(i,j) + \frac{\delta I}{\delta t}]^2$
- So, we want to minimize
  min$E = \sum_i \sum_j [c(i,j) + \lambda s(i,j)]$

Horn Schunck Method

- To minimize the error, we differentiate the error equation with respect to $v_x(i,j)$ and $v_y(i,j)$
- $$\begin{aligned}\frac{\delta e}{\delta v_x(i,j)} &= 2\left(\frac{\delta I}{\delta x}v_x(i,j) + \frac{\delta I}{\delta y}v_y(i,j) + \frac{\delta I}{\delta t}\right)\frac{\delta I}{\delta x} \\ &\quad +2\lambda(v_x(i,j) - \bar{v}_x(i,j)) = 0 \\ \frac{\delta e}{\delta v_y(i,j)} &= 2\left(\frac{\delta I}{\delta x}v_x(i,j) + \frac{\delta I}{\delta y}v_y(i,j) + \frac{\delta I}{\delta t}\right)\frac{\delta I}{\delta y} \\ &\quad +2\lambda(v_y(i,j) - \bar{v}_y(i,j)) = 0\end{aligned}$$

  where $\bar{v}_x$ and $\bar{v}_y$ are local averages of v_x and v_y respectively.

Horn Schunck Method

- We can write

$$\left[\lambda + \left(\frac{\delta I}{\delta x}\right)^2\right] v_x + \frac{\delta I}{\delta x}\frac{\delta I}{\delta y} v_y = \lambda \bar{v}_x - \frac{\delta I}{\delta x}\frac{\delta I}{\delta t}$$

$$\frac{\delta I}{\delta x}\frac{\delta I}{\delta y} v_x + \left[\lambda + \left(\frac{\delta I}{\delta x}\right)^2\right] v_y = \lambda \bar{v}_y - \frac{\delta I}{\delta y}\frac{\delta I}{\delta t}$$

Horn Schunck Method

- Iterative scheme (Gauss-Seidel equations)

$$v_x^{k+1} = \bar{v}_x^k - \left[\frac{\left(\frac{\delta I}{\delta x}\right) \bar{v}_x^k + \left(\frac{\delta I}{\delta y}\right) \bar{v}_y^k + \frac{\delta I}{\delta t}}{\lambda + \left(\frac{\delta I}{\delta x}\right)^2 + \left(\frac{\delta I}{\delta y}\right)^2} \right] \frac{\delta I}{\delta x}$$

$$v_y^{k+1} = \bar{v}_y^k - \left[\frac{\left(\frac{\delta I}{\delta x}\right) \bar{v}_x^k + \left(\frac{\delta I}{\delta y}\right) \bar{v}_y^k + \frac{\delta I}{\delta t}}{\lambda + \left(\frac{\delta I}{\delta x}\right)^2 + \left(\frac{\delta I}{\delta y}\right)^2} \right] \frac{\delta I}{\delta y}$$

- More concisely: $\mathbf{v}^{k+1} = \bar{\mathbf{v}}^k - \alpha(\nabla \mathbf{I})$

Horn Schunck Method Implementation

- Usually use the following three masks to compute spatio temporal differences $M_x = \frac{1}{4}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, M_y = \frac{1}{4}\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, M_t = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$
- Use the following equations to compute $\frac{\delta I}{\delta x}$, $\frac{\delta I}{\delta y}$, and $\frac{\delta I}{\delta t}$

$$\begin{aligned} \frac{\delta I}{\delta x} &= M_x * (I_1 + I_2) \\ \frac{\delta I}{\delta y} &= M_y * (I_1 + I_2) \\ \frac{\delta I}{\delta t} &= M_t * (I_2 - I_1) \end{aligned}$$