

## 1 Review

- *edges*: points with  $\max \|\nabla g\|$ , always the direction of  $\nabla g$ .
- *corners*: points with maximum cornerness.

$$\text{cornerness} = \det M - K \text{tr}^2(M) \text{ (by Harris), where } M = \sum_{i \in \mathbb{N}} \begin{pmatrix} I_{x,i}^2 & I_{x,i}I_{y,i} \\ I_{x,i}I_{y,i} & I_{y,i}^2 \end{pmatrix}$$

Particularly,  $\det M = \lambda_1 \lambda_2$ ,  $\text{tr}(M) = \lambda_1 + \lambda_2$ .

## 2 Feature Tracking

### 1. *Kanade-Lucas-Tomasi (KLT)* tracker

In two images with object motion, we can easily have  $I_1(x - u, y - v) = I_2(x, y)$ . The intensity are the same, which is called *Brightness Constancy*. However, the same brightness is not SUFFICIENT! Neighborhood should match and we should not only focus on one pixel.

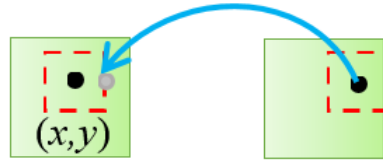


Figure 1: Feature Tracking between two pictures

Here,  $(u, v) = \arg \min_{(u,v)} \varepsilon(u, v)$  is a displacement or optical flow, we can define:

$$E(u, v) = I_1(x - u, y - v) - I_2(x, y)$$

And the tracking operator is:

$$\varepsilon(u, v) = \iint E(u, v)^2 dx dy \tag{1}$$

Because  $(u, v)$  is assumed to be constant within pixel's neighbourhood  $\mathbb{N}$  as pure local translation, we can have:

$$\frac{\partial \varepsilon}{\partial u}(u, v) = 0, \quad \frac{\partial \varepsilon}{\partial v}(u, v) = 0$$

$$\text{Hessian}(\varepsilon) = \begin{pmatrix} \frac{\partial^2 \varepsilon}{\partial u^2} & \frac{\partial \varepsilon}{\partial u} \frac{\partial \varepsilon}{\partial v} \\ \frac{\partial \varepsilon}{\partial u} \frac{\partial \varepsilon}{\partial v} & \frac{\partial^2 \varepsilon}{\partial v^2} \end{pmatrix}$$

Apparently, the  $\text{Hessian}(\varepsilon)$  is positive definite, so we can find the minimum for  $\varepsilon$ . By writing  $\vec{d} = (u, v)^T$ , we can have:

$$\frac{\partial \varepsilon}{\partial \vec{d}} = \frac{\partial}{\partial \vec{d}} \left( \iint E(u, v) dx dy \right) = \iint 2E(\vec{d}) \frac{\partial E}{\partial \vec{d}} dx dy \quad (2)$$

Considering back about  $E(u, v)$ , we know:

$$\begin{aligned} E(u, v) &= I_1(x - u, y - v) - I_2(x, y) \\ &= I_1(x, y) - \nabla I_1^T \begin{pmatrix} u \\ v \end{pmatrix} - I_2(x, y) \\ &= I_1(x, y) - I_2(x, y) - \nabla I_1^T \vec{d} \\ &= -\Delta I(x, y) - \nabla I_1^T \vec{d}, \end{aligned}$$

where  $\Delta I(x, y) = I_2(x, y) - I_1(x, y)$ , and it is the same pixel here not shifted one. The result contains  $\vec{d}$ , which is good to find  $(u, v)$  by setting  $\frac{\partial \varepsilon}{\partial \vec{d}} = 0$ . Now we can find  $\frac{\partial E}{\partial \vec{d}}$ :

$$\frac{\partial E}{\partial \vec{d}} = \frac{\partial I_1(\vec{x} - \vec{d})}{\partial \vec{d}} = \frac{\partial (I_1(\vec{x}) - \vec{d}^T \nabla I_1(\vec{x}))}{\partial \vec{d}} = -\nabla I_1 \vec{x}$$

Put this into equation (2) again, we can have:

$$\begin{aligned} \frac{\partial}{\partial \vec{d}} \varepsilon(u, v) &= 2 \iint (\Delta I(x, y) + \nabla I^T \vec{d}) \nabla I dx dy = 0 \\ \Rightarrow (\iint \nabla I \nabla I^T) \vec{d} &= - \iint \Delta I \nabla I dx dy \end{aligned}$$

## 2. Discretized Version

The previous feature tracker is in continuous version, and here we can also write everything in discrete as:

$$\begin{aligned} \sum_{i \in \mathbb{N}} \nabla I_i \nabla I_i^T \vec{d} &= - \sum_{i \in \mathbb{N}} \nabla I \Delta I \\ \sum \begin{pmatrix} I_{x,i}^2 & I_{x,i} I_{y,i} \\ I_{x,i} I_{y,i} & I_{y,i}^2 \end{pmatrix} \vec{d} &= - \sum \nabla I \Delta I = - \sum \begin{pmatrix} \Delta I I_{x,i} \\ \Delta I I_{y,i} \end{pmatrix} \end{aligned} \quad (3)$$

## 3. Implementation

We have two images as matrices  $I_1$  and  $I_2$ . In MATLAB, we can have these following implementations (Figure 2).

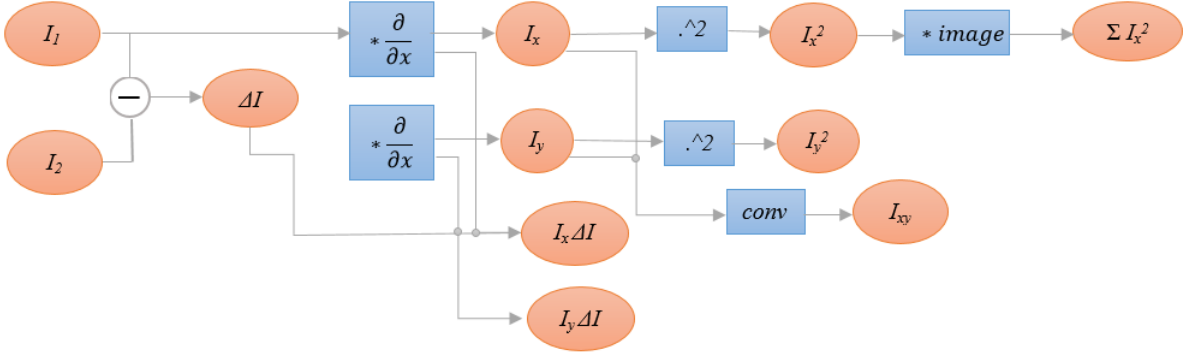


Figure 2: Implementation Procedures in MATLAB.

#### 4. Existence and Uniqueness

- Existence

We can rewrite the equation (3) in discrete form as:

$$M_{2 \times 2} \vec{d}_{2 \times 1} = \vec{m}_{2 \times 1}$$

Inputting  $\vec{m}$ , if  $(M, \vec{m})$  has a rank of 2, which means  $\vec{m}$  belongs to the range of  $M$ , the solution exists, when:

- (1) The intensity changed.
- (2)  $(u, v)$  is not constant.

- Uniqueness

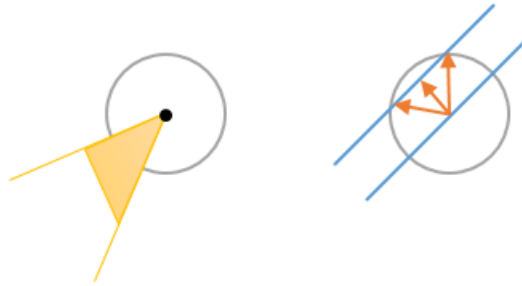


Figure 3: Uniqueness of KLT.

Solution is unique, if and only if  $\text{rank}(M) = 2$  or  $\det(M) \neq 0$  or  $\lambda_{\min} > 0$ , because the  $\text{Hessian}(M)$  is positive definite.

Note: *The uniqueness of KLT is the same as cornerness.* Figure 3 shows two examples, and in the second case there exists solution but it is not the unique solution.

## 5. Motion

All we have discussed above is based on  $\vec{d} = (u, v)^T$  is displacement. However, when  $(u, v)$  is a velocity, which means the images moves fast, how can we track the feature?

In previous part, when we compute the gradient of image, we always use:

$$\frac{\partial I}{\partial x} = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Here, we can use another kind of representation:

$$\frac{\partial I}{\partial x} = \frac{f(x + \Delta x/2) - f(x - \Delta x/2)}{\Delta x}$$

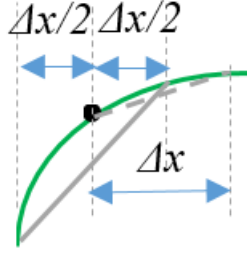


Figure 4: Gradient when image moves.

It is like the convolution with  $[-\frac{1}{2} \ 0 \ \frac{1}{2}]$  in the time domain as a discrete form.

Previously, in the equation  $M\vec{d} = \vec{m}$ , both  $M = \sum \nabla I_1 \Delta I_1$  and  $\vec{m} = \nabla I_1 (I_2 - I_1)$  are referring to the first image. Now, the first image is moving fast. Considering the symmetric definition of  $\varepsilon(u, v)$ , we can rewrite the equation (1) as:

$$\varepsilon(u, v) = \iint \left( I_1(\vec{x}) - \frac{\vec{d}}{2} - I_2(\vec{x} + \frac{\vec{d}}{2}) \right)^2 d\vec{x}$$

And instead of  $\nabla I_1$ , we can compute:

$$\begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} \left( \frac{I_1 + I_2}{2} \right)$$

Drawbacks: It will add noise to the original signal.

## 3 Optical Flow Equation

Take a back look at that  $I_1(x - u, y - v) = I_2(x, y)$ , we can rewrite it as:

$$I_1(x, y) - (u, v) \nabla I_1(x, y) = I_2(x, y)$$

$$\Rightarrow \nabla I_1^T \begin{pmatrix} u \\ v \end{pmatrix} + \Delta I = 0$$

For each pixel, we can also input the time:

$$I_x u + I_y v + I_t = 0,$$

where  $I_t = \frac{\partial I}{\partial t}$ , and  $(u, v)$  is a constant. If we put this to the whole image, we can have:

$$\begin{pmatrix} I_{x,1} & I_{y,1} \\ \vdots & \vdots \\ I_{x,n} & I_{y,n} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_{t,1} \\ \vdots \\ -I_{t,n} \end{pmatrix} \quad (4)$$

It has the form as  $A \begin{pmatrix} u \\ v \end{pmatrix} = b$ , where  $A$  is  $n \times 2$ ,  $\begin{pmatrix} u \\ v \end{pmatrix}$  is  $2 \times 1$  and  $b$  is  $n \times 1$ . Here,  $A$  is non-singular, because gradients are not all parallel. Hence,  $A^T A$  is non-singular as well. Then the equation can be written as:

$$(A^T A) \begin{pmatrix} u \\ v \end{pmatrix} = -A^T b \Rightarrow M \begin{pmatrix} u \\ v \end{pmatrix} = \vec{m}$$

It is the relaxation of constancy (*Tomas-Shi: 93*). Note: *Optical Flow is locally affine*, like:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}$$

The matrix  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$  here can be a rotation matrix as  $\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$ , which represents the rotation or dilating of original plane.



Figure 5: Affine Transforms with different matrices.

And when the matrix is diagonal, the image is square. while when it is an upper-triangle, the image is a Parallelogram, like Figure 5.

By setting  $\vec{d} = (u, v)$ , we can have:

$$\begin{pmatrix} u \\ v \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix} + \vec{d}$$

$$(I_x, I_y)(A\vec{x} + \vec{d}) + I_t = 0 \Rightarrow \nabla I^T A\vec{x} + \nabla I^T \vec{d} + I_t = 0,$$

$$\text{where } A\vec{x} + \vec{d} = \begin{pmatrix} ax + by + u_0 \\ cx + dy + v_0 \end{pmatrix}.$$

Here, we can do some simplifications by using:

$$F^T = (xI_x \ yI_x \ xI_y \ yI_y)_{1 \times 4}$$

And,

$$F^T(a \ b \ c \ d)^T = \nabla I^T A\vec{x}$$

The equation (4) can be rewritten as:

$$\underbrace{\begin{pmatrix} F_1^T & \nabla I_1^T \\ \vdots & \vdots \\ F_n^T & \nabla I_n^T \\ n \times 4 & n \times 2 \end{pmatrix}}_K \begin{pmatrix} a \\ b \\ c \\ d \\ u_0 \\ v_0 \end{pmatrix} = - \begin{pmatrix} I_{t,1} \\ \vdots \\ I_{t,n} \end{pmatrix}$$

We have already known the matrix  $K$ , here we want to solve  $(a \ b \ c \ d \ u_0 \ v_0)^T$ . Now we might have a trade-off between the  $rank(K)$  and the size of neighborhood. Definitely, when the image is small, then this kind of computation is OK. However, if the size of image is big, it will bring huge computation. Hence, practically, we track with  $(u_0, v_0)^T$  only, and if  $M\vec{d} = \vec{m}$  is not satisfied then we compute affine.

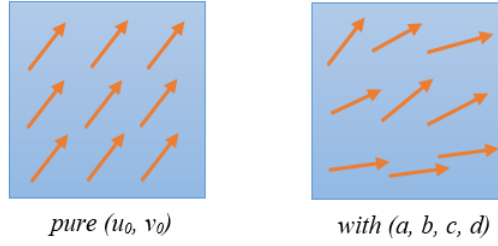


Figure 6: Effect of using  $F^T$  and  $(a \ b \ c \ d \ u_0 \ v_0)^T$ .