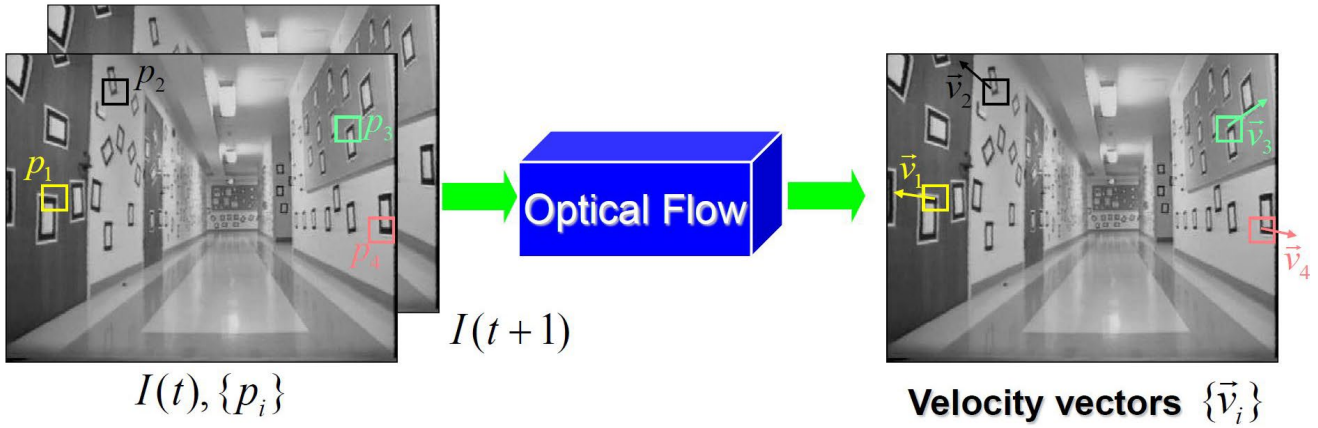
Optical Flow & Visual Tracking

Yulai Weng

November 18, 2019

Optical Flow

What is Optical Flow (OF)?



Optical flow is the relation of the motion field:

- *the 2D projection of the physical movement of points relative to the observer* to 2D displacement of pixel patches on the image plane.

Common assumption:

The appearance of the image patches do not change (brightness constancy)

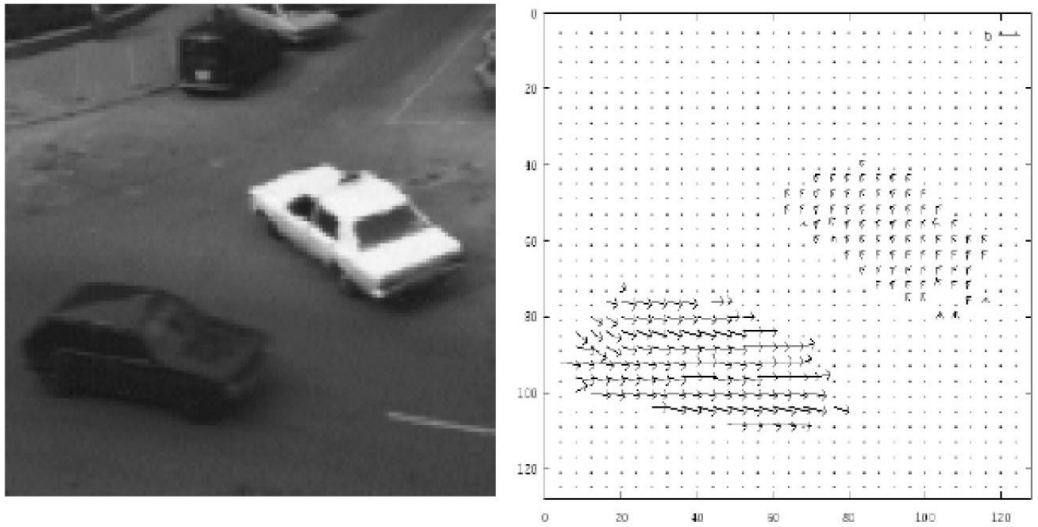
$$I(p_i, t) = I(p_i + \vec{v}_i, t+1)$$

Note: more elaborate tracking models can be adopted if more frames are process all at once

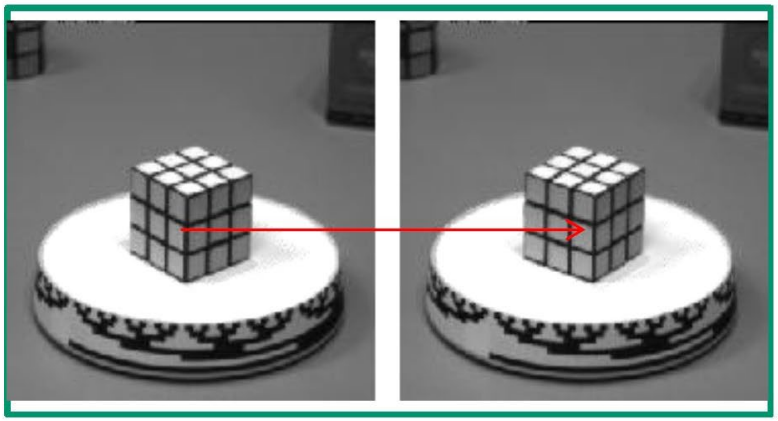
Optical Flow

OPTICAL FLOW = apparent motion of brightness patterns

Ideally, the optical flow is the projection of the three-dimensional velocity vectors on the image



Brightness Constancy

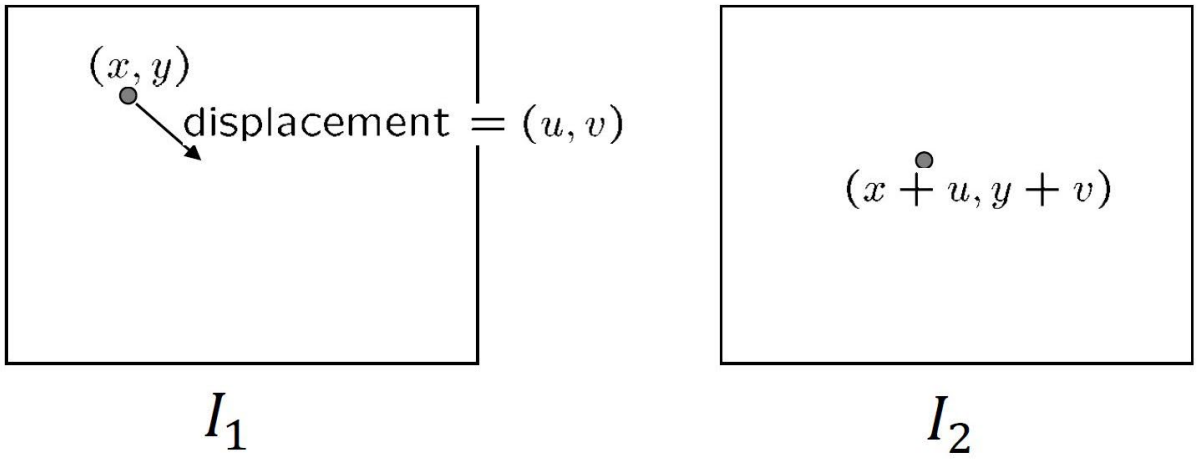


Time: t Time: t + dt

Point moves (small), but its brightness remains constant:

$$I_{t1}(x, y) = I_{t2}(x + u, y + v)$$

$$I = constant \rightarrow \frac{dI}{dt} = 0$$



Mathematical Formulation

$I(x(t),y(t),t)$ = brightness at (x,y) at time t

Brightness constancy assumption (shift of location but brightness stays same):

$$I(x+\frac{dx}{dt}\delta t, y+\frac{dy}{dt}\delta t, t+\delta t) = I(x,y,t)$$

Optical flow constraint equation (chain rule):

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

Aperture Problem

$$u = \frac{dx}{dt}, \qquad v = \frac{dy}{dt}$$

$$I_x = \frac{\partial I}{\partial y}, \qquad I_y = \frac{\partial I}{\partial y}, \qquad I_t = \frac{\partial I}{\partial t}$$

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

Horn and Schunck optical flow equation

1 equation in 2 unknowns

Optical Flow: 1D Case

Brightness Constancy Assumption:

$$f(t) \equiv I(\underbrace{x(t), t}) = I(x(t+dt), t+dt)$$

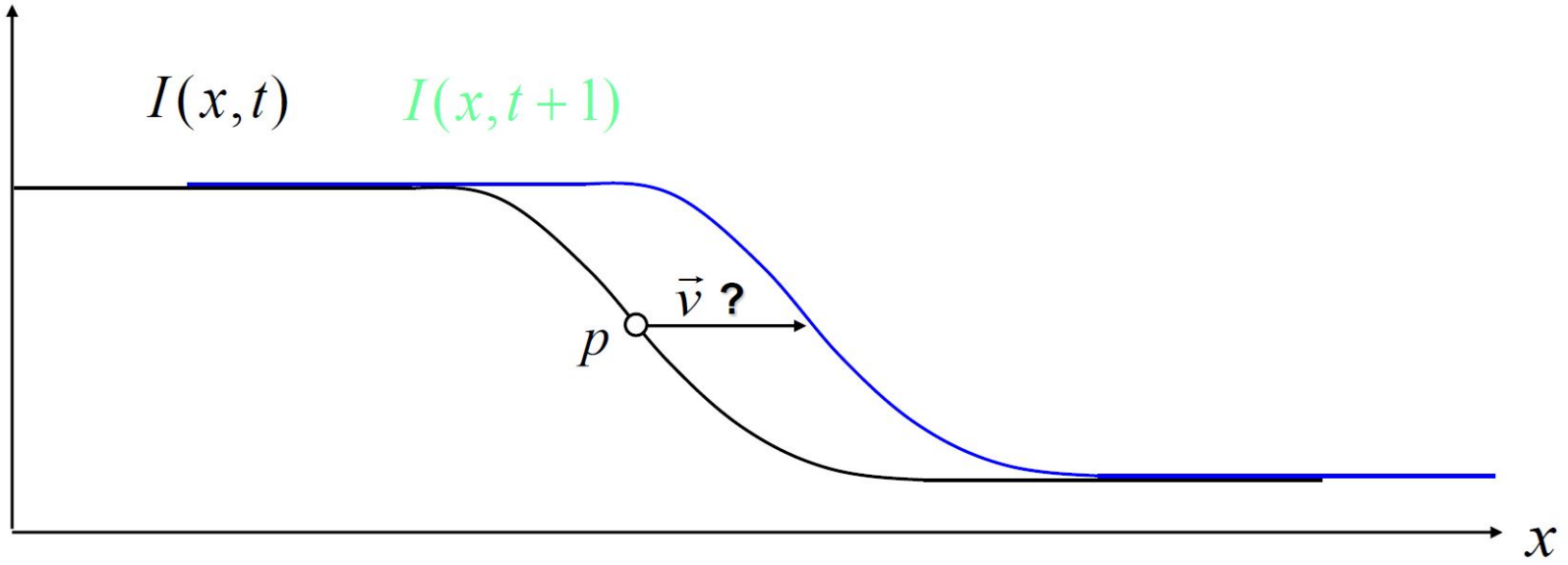
$$\frac{\partial f(x)}{\partial t} = 0$$ Because no change in brightness with time

$$\left.\frac{\partial I}{\partial x}\right|_t \left(\frac{\partial x}{\partial t}\right) + \left.\frac{\partial I}{\partial t}\right|_{x(t)} = 0$$

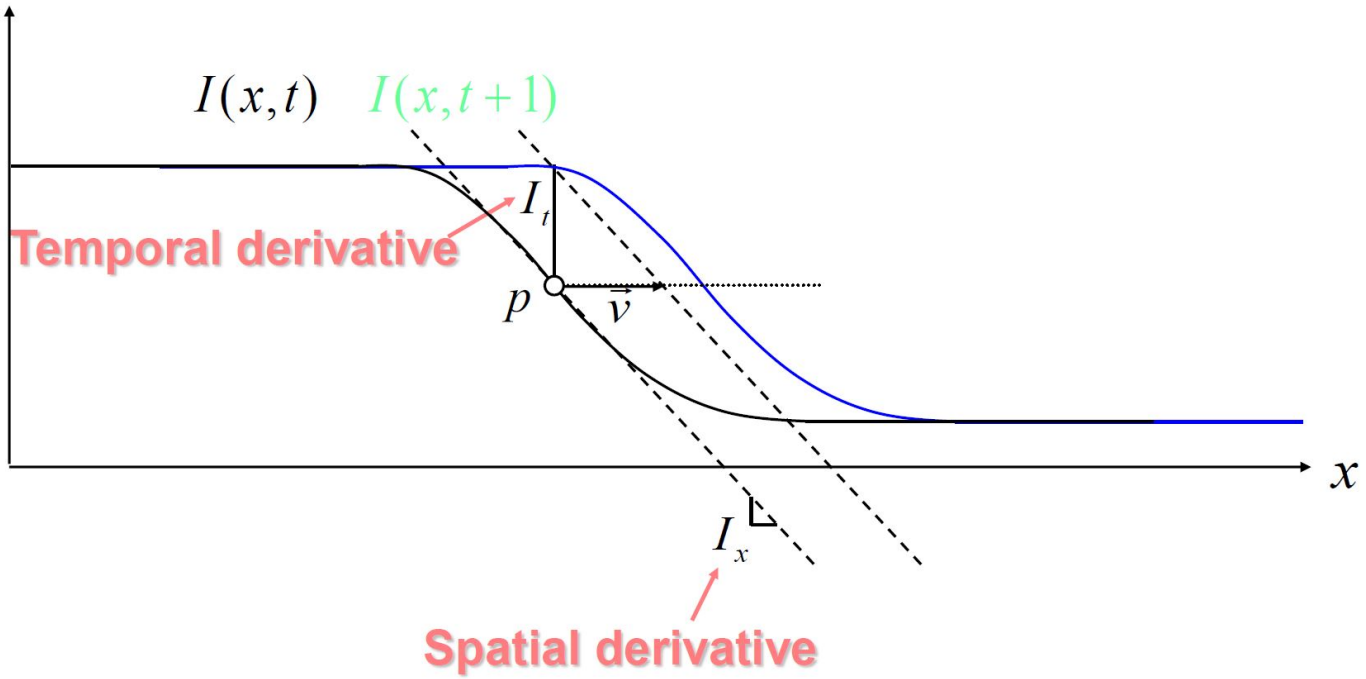
I_x v I_t

$$\Rightarrow v = -\frac{I_t}{I_x}$$

Tracking in 1D Case



Tracking in 1D Case



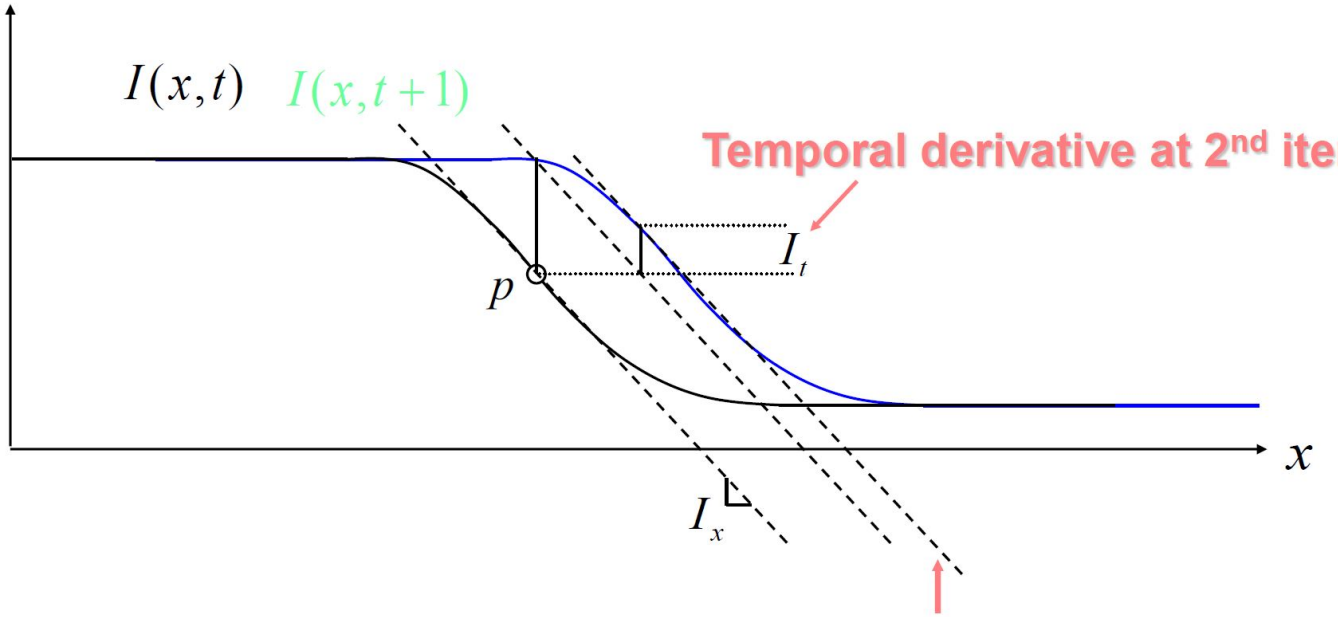
$$I_x = \left.\frac{\partial I}{\partial x}\right|_t \qquad I_t = \left.\frac{\partial I}{\partial t}\right|_{x=p} \qquad \Rightarrow \qquad \vec{v} \approx -\frac{I_t}{I_x}$$

Assumptions:

- **Brightness constancy**
- **Small motion**

Tracking in 1D Case

Iterating helps refining the velocity vector



$$\vec{v} \leftarrow \vec{v}_{previous} - \frac{I_t}{I_x}$$

Converges in about 5 iterations

From 1D to 2D Tracking

1D: $$\left.\frac{\partial I}{\partial x}\right|_t \left(\frac{\partial x}{\partial t}\right) + \left.\frac{\partial I}{\partial t}\right|_{x(t)} = 0$$

2D: $$\left.\frac{\partial I}{\partial x}\right|_t \left(\frac{\partial x}{\partial t}\right) + \left.\frac{\partial I}{\partial y}\right|_t \left(\frac{\partial y}{\partial t}\right) + \left.\frac{\partial I}{\partial t}\right|_{x(t)} = 0$$

$$\left.\frac{\partial I}{\partial x}\right|_t u + \left.\frac{\partial I}{\partial y}\right|_t v + \left.\frac{\partial I}{\partial t}\right|_{x(t)} = 0$$

One equation, two velocity *(u,v)* unknowns…

Normal Flow

Notation

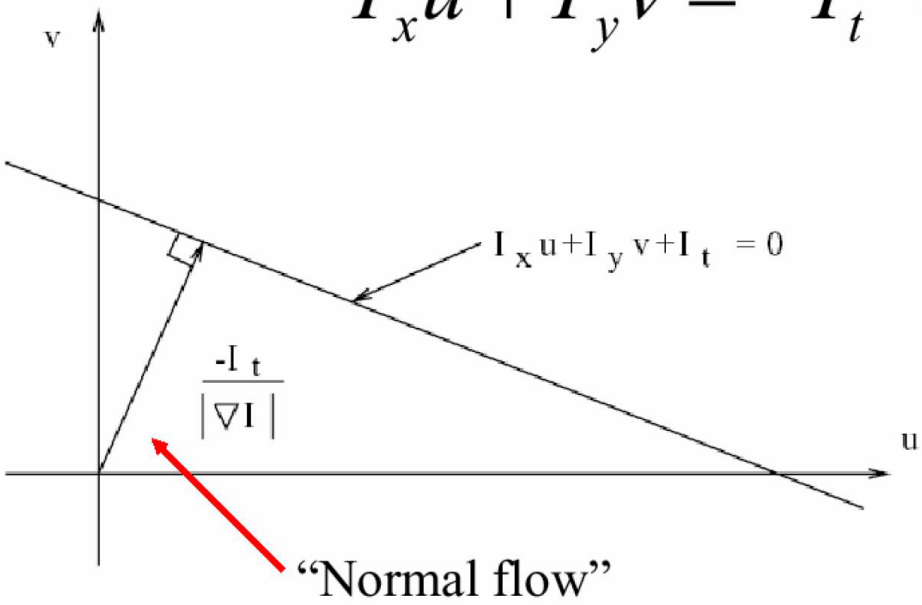
$$I_x u + I_y v + I_t = 0$$

$$\nabla I^T \mathbf{u} = -I_t$$

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} \quad \nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

At a single image pixel, we get a line:

$$I_x u + I_y v = -I_t$$



We get at most "Normal Flow" – with one point we can only detect movement perpendicular to the brightness gradient. Solution is to take a patch of pixels Around the pixel of interest.

Solving Apeture Problem

- How to get more equations for a pixel?
  - Basic idea: impose additional constraints
    - most common is to assume that the flow field is smooth locally
    - one method: pretend the pixel's neighbors have the same (u,v)
      - If we use a 5x5 window, that gives us 25 equations per pixel!

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u\ v]$$

$$\underbrace{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}}_{A\ 25\times2} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d\ 2\times1} = - \underbrace{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}_{b\ 25\times1}$$

Lucas-Kanade Flow

$$\underset{25\text{x}2}{A}\ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

- We have more equations than unknowns: solve least squares problem. This is given by:

$$\underset{2\text{x}2}{(A^T A)}\ \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

  - Summations over all pixels in the KxK window

Condition for Solvability

– Optimal (u, v) satisfies Lucas-Kanade equation

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

When is This Solvable?

- $\mathbf{A^T A}$ should be invertible
- $\mathbf{A^T A}$ should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of $\mathbf{A^T A}$ should not be too small
- $\mathbf{A^T A}$ should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

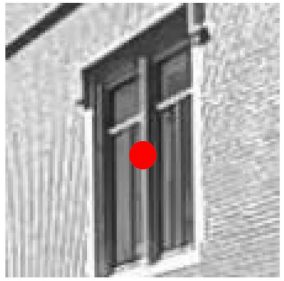
Errors in Lucas Kanade

- What are the potential causes of errors in this procedure?
  - Suppose A^TA is easily invertible
  - Suppose there is not much noise in the image
- When our assumptions are violated
  - Brightness constancy is **not** satisfied
  - The motion is **not** small
  - A point does **not** move like its neighbors
    - window size is too large
    - what is the ideal window size?
- A^TA is singular if all gradient vectors point in the same direction
  -- e.g., along an edge
  -- of course, trivially singular if the summation is over a single pixel
  -- i.e., only *normal flow* is available (aperture problem)
- Corners and textured areas are OK

Iterative Refinement

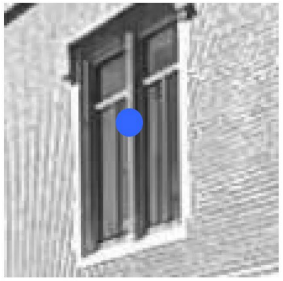
- Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation
- Warp one image toward the other using the estimated flow field

  (easier said than done)
- Refine estimate by repeating the process



$\mathbf{I}(\mathbf{x})$

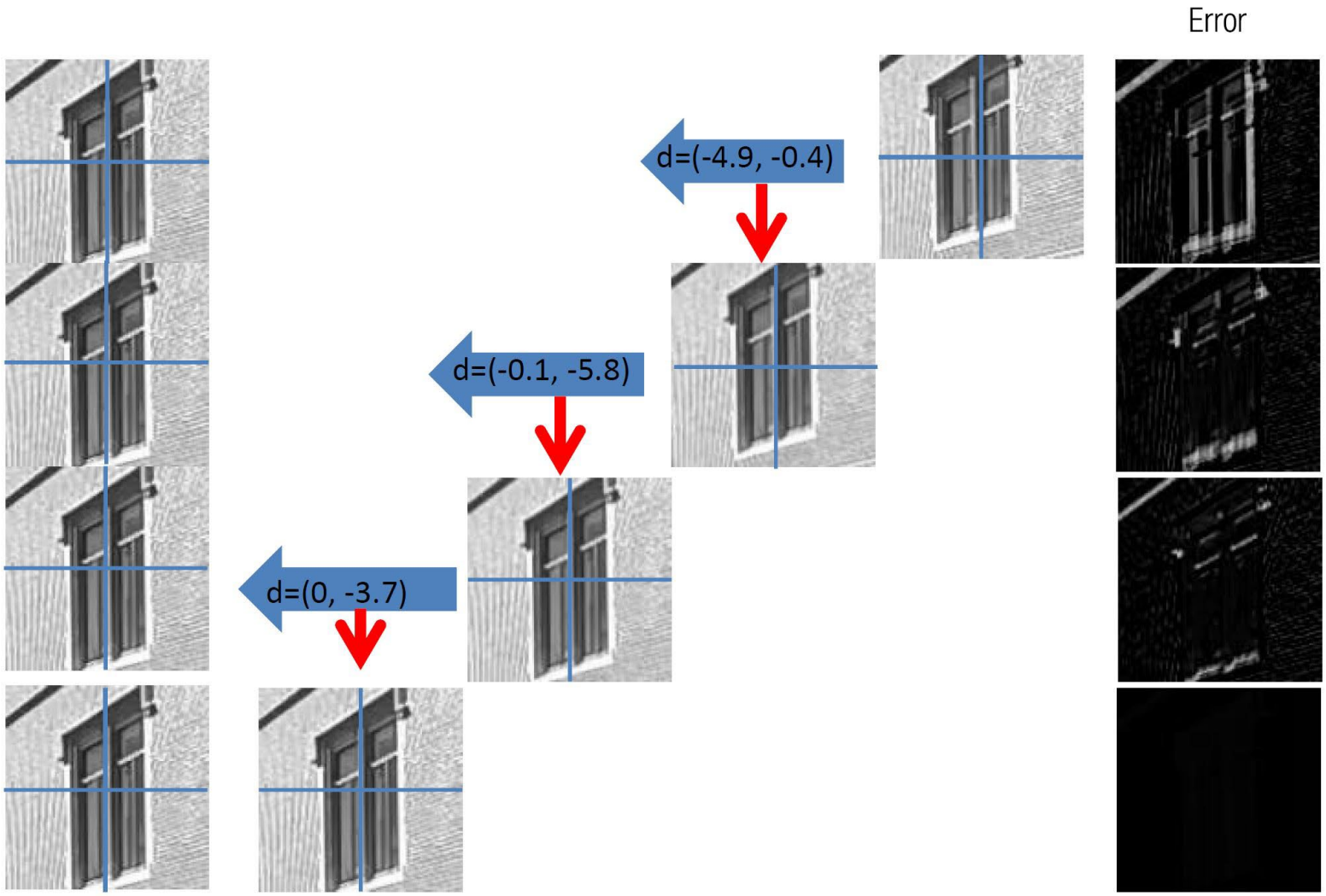
$t = 0$



$\mathbf{J}(\mathbf{x})$

$t = 1$

Iterative Refinement



Iterative Refinement

The Inverse Compositional Algorithm

Original (x, y) position

1. **Initialize** $(x', y') = (x, y)$
2. **Compute** (u, v) **by**

$$I_t = J(x', y') - I(x, y)$$

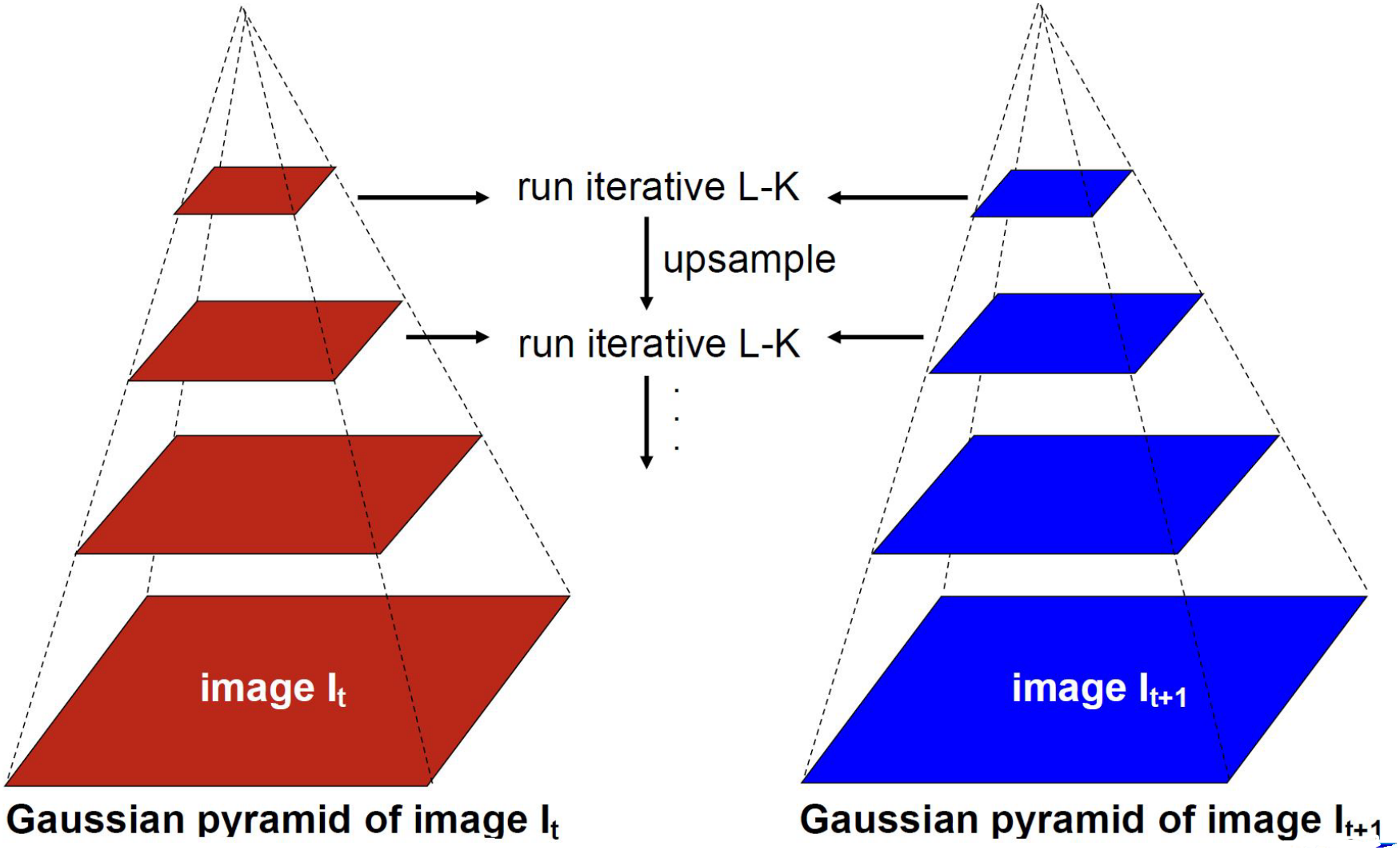
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

2^{nd} moment matrix for feature patch in first image

displacement

3. **Shift window by** (u, v): $x' := x' + u$; $y' := y' + v$;
4. **Recalculate** I_t
5. **Repeat steps 2-4 until small change**
   - Use interpolation for subpixel values

Coarse to Fine Image Pyramid



Build Image Pyramid

$$
\begin{aligned}
I^L(x,y) \quad = \quad & \frac{1}{4} I^{L-1}(2x,2y) + \\
& \frac{1}{8}\left(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)\right) + \\
& \frac{1}{16}\left(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)\right)
\end{aligned}
$$

For simplicity in the notation, let us define dummy image values one pixel around the image I^{L-1} (for $0 \le x \le n_x^{L-1}-1$ and $0 \le y \le n_y^{L-1}-1$):

$$
\begin{aligned}
I^{L-1}(-1,y) &\doteq I^{L-1}(0,y), \\
I^{L-1}(x,-1) &\doteq I^{L-1}(x,0), \\
I^{L-1}(n_x^{L-1},y) &\doteq I^{L-1}(n_x^{L-1}-1,y), \\
I^{L-1}(x,n_y^{L-1}) &\doteq I^{L-1}(x,n_y^{L-1}-1), \\
I^{L-1}(n_x^{L-1},n_y^{L-1}) &\doteq I^{L-1}(n_x^{L-1}-1,n_y^{L-1}-1).
\end{aligned}
$$

Then, equation 2 is only defined for values of x and y such that $0 \le 2x \le n_x^{L-1}-1$ and $0 \le 2y \le n_y^{L-1}-1$. Therefore, the width n_x^L and height n_y^L of I^L are the largest integers that satisfy the two conditions:

$$
\begin{aligned}
n_x^L &\le \frac{n_x^{L-1}+1}{2}, \\
n_y^L &\le \frac{n_y^{L-1}+1}{2}.
\end{aligned}
$$

Tracking in Image Pyramid

Goal: Let $\mathbf{u}$ be a point on image I. Find its corresponding location $\mathbf{v}$ on image J

Build pyramid representations of I *and* J: $\{I^L\}_{L=0,\dots,L_m}$ and $\{J^L\}_{L=0,\dots,L_m}$

Initialization of pyramidal guess: $\mathbf{g}^{L_m} = [g_x^{L_m} \;\; g_x^{L_m}]^T = [0 \;\; 0]^T$

for $L = L_m$ **down to** 0 **with step of** -1

Location of point $\mathbf{u}$ *on image* I^L: $\mathbf{u}^L = [p_x \;\; p_y]^T = \mathbf{u}/2^L$

Derivative of I^L *with respect to* x: $I_x(x,y) = \dfrac{I^L(x+1,y) - I^L(x-1,y)}{2}$

Derivative of I^L *with respect to* y: $I_y(x,y) = \dfrac{I^L(x,y+1) - I^L(x,y-1)}{2}$

Spatial gradient matrix: $$G = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)\,I_y(x,y) \\ I_x(x,y)\,I_y(x,y) & I_y^2(x,y) \end{bmatrix}$$

Initialization of iterative L-K: $\overline{\nu}^0 = [0 \;\; 0]^T$

Tracking in Image Pyramid

for $k = 1$ **to** K **with step of** 1 (or until $\|\overline{\eta}^k\| <$ accuracy threshold)

Image difference: $\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + \nu_x^{k-1}, y + g_y^L + \nu_y^{k-1})$

Image mismatch vector: $\overline{b}_k = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y)\, I_x(x,y) \\ \delta I_k(x,y)\, I_y(x,y) \end{bmatrix}$

Optical flow (Lucas-Kanade): $\overline{\eta}^k = G^{-1}\, \overline{b}_k$

Guess for next iteration: $\overline{\nu}^k = \overline{\nu}^{k-1} + \overline{\eta}^k$

end of for-loop on k

Final optical flow at level L: $\mathbf{d}^L = \overline{\nu}^K$

Guess for next level $L-1$: $\mathbf{g}^{L-1} = [g_x^{L-1} \;\; g_y^{L-1}]^T = 2\,(\mathbf{g^L} + \mathbf{d}^L)$

end of for-loop on L

Final optical flow vector: $\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0$

Location of point on J: $\mathbf{v} = \mathbf{u} + \mathbf{d}$

Solution: The corresponding point is at location $\mathbf{v}$ on image J

Affine Warp

- Optical Flow: Translation Warp

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix}$$

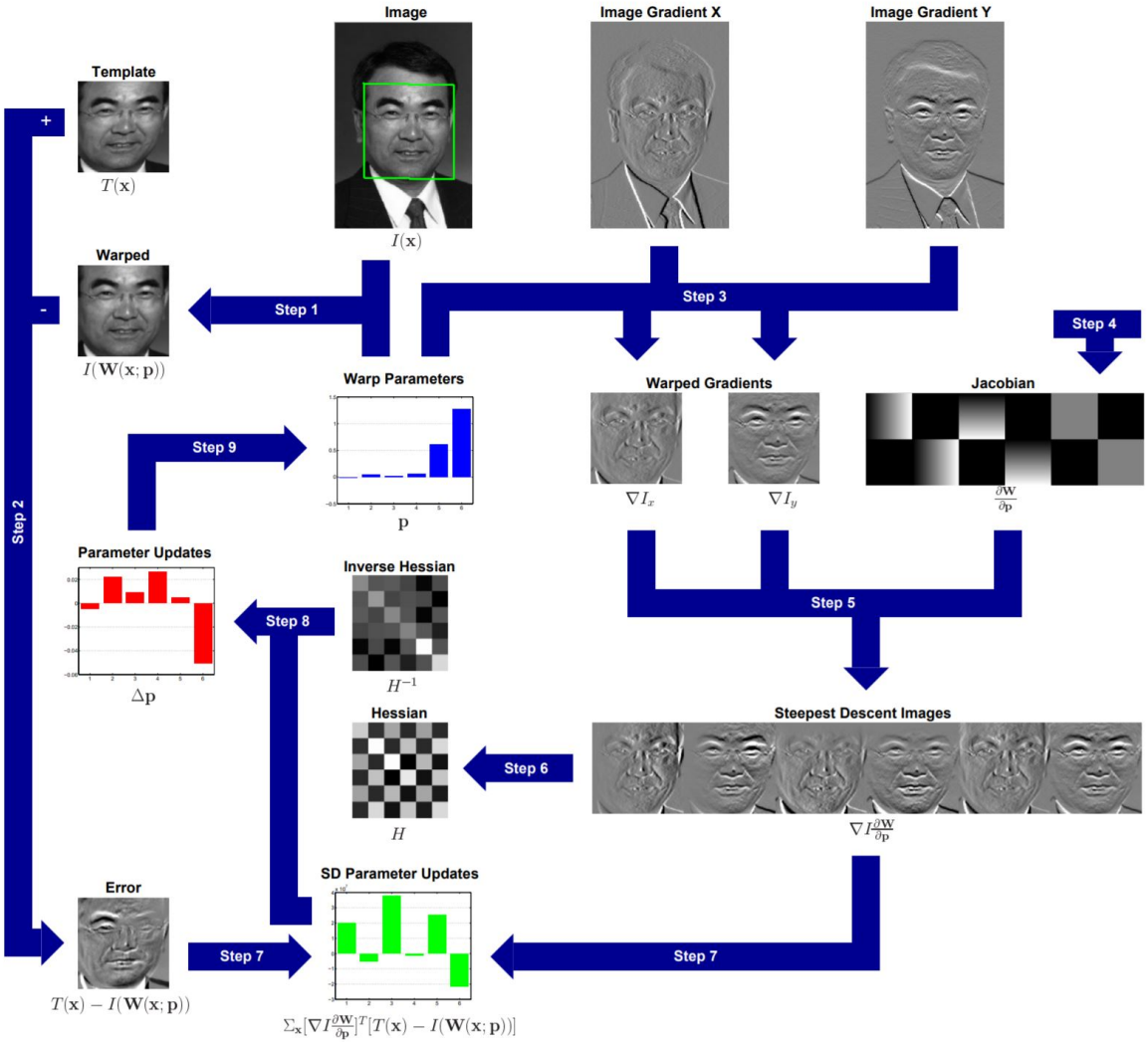
- Affine Warps (May handle more deformable image patch)

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot x & + & p_3 \cdot y & + & p_5 \\ p_2 \cdot x & + & (1+p_4)\cdot y & + & p_6 \end{pmatrix} = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- If we assume current frame at t $\boldsymbol{T(x)}$, called template image

  next frame at $t+1$ $\boldsymbol{I(x)}$, called source image

General Lucas-Knade Image Alignement



General Lucas-Knade Image Alignement

The Inverse Compositional Algorithm

Pre-compute:

Evaluate the gradient ∇T of the template $T(\mathbf{x})$

Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x}; \mathbf{0})$

Compute the steepest descent images $\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$

Compute the Hessian matrix using Equation $H = \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$

Iterate:

Warp I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$

Compute the error image $I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})$

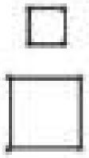
Compute $\sum_{\mathbf{x}} [\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})]$

Compute $\Delta \mathbf{p}$ using Equation $\Delta \mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})]$

Update the warp $\mathbf{W}(\mathbf{x}; \mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta \mathbf{p})^{-1}$

until $\|\Delta \mathbf{p}\| \leq \epsilon$

Similarity Transform

Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, $\mathbf{I}, \mathbf{J}$ (see section 2.7.3).

tform = skimage.transform.estimate_transform('similarity', src, dst)

tformp=np.asmatrix(tform.params)

coordnew=tformp.dot(coord)

- Eliminate Bad Feature

Thanks !