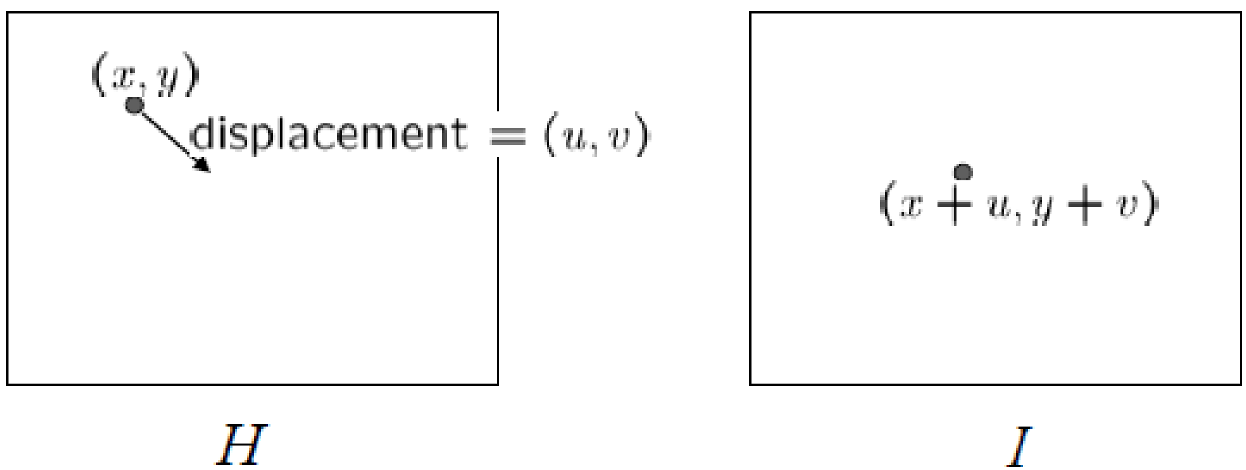
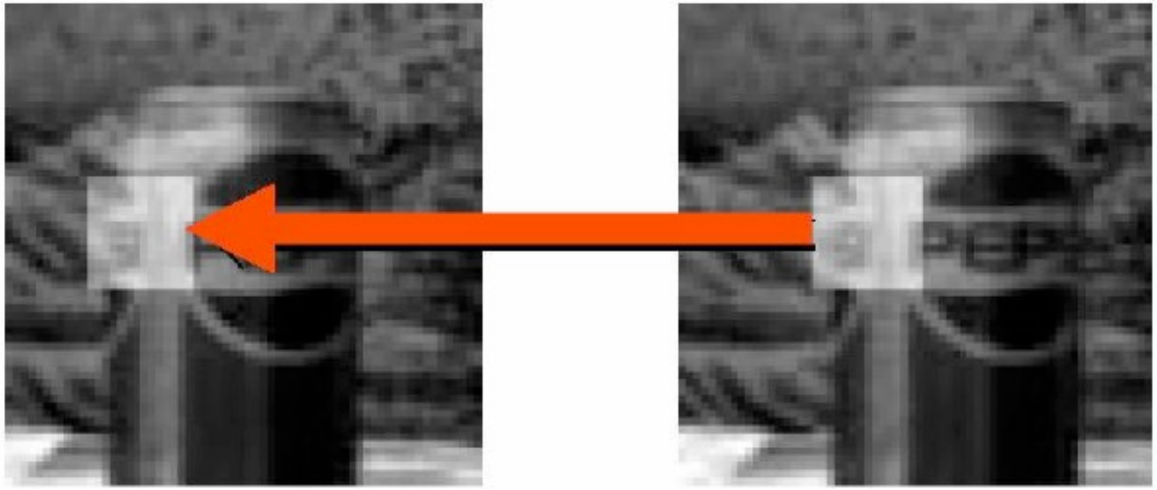
Optical Flow

How to estimate pixel motion from one image to another?



Assumption 1: Brightness is constant.

$$H(x,y) = I(x+u, y+v)$$



Assumption 2: Motion is small.

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

$$\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

(from Taylor series expansion)

Combine

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$\begin{aligned} 0 &= I(x+u, y+v) - H(x,y) \\ &\approx I(x,y) + I_x u + I_y v - H(x,y) \\ &\approx \underbrace{(I(x,y) - H(x,y))}_{I_t} + I_x u + I_y v \\ &\approx I_t + I_x u + I_y v \end{aligned}$$

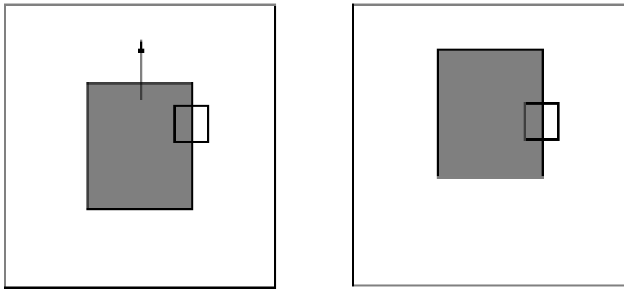
In the limit as u and v goes to zero, the equation becomes exact

$$0 = I_t + I_x u + I_y v \qquad \text{(optical flow equation)}$$

At each pixel, we have one equation, two unknowns.

$$0 = I_t + I_x u + I_y v$$ (optical flow equation)

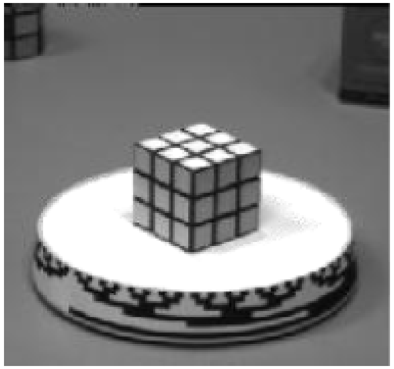
This means that only the flow component in the gradient direction can be determined.



The motion is parallel to the edge, and it cannot be determined.

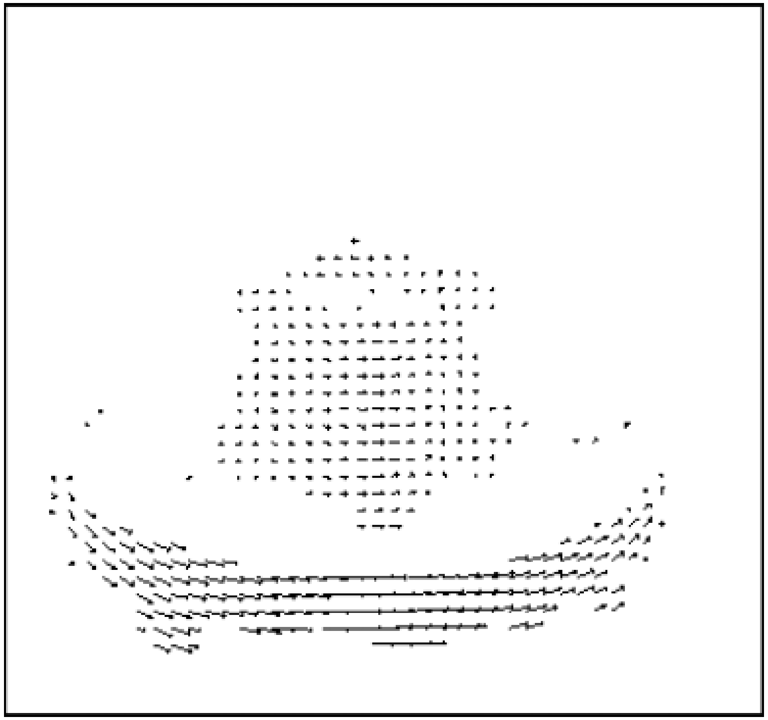
This is called the *aperture problem*.

- Which pixel went where?



Time: t

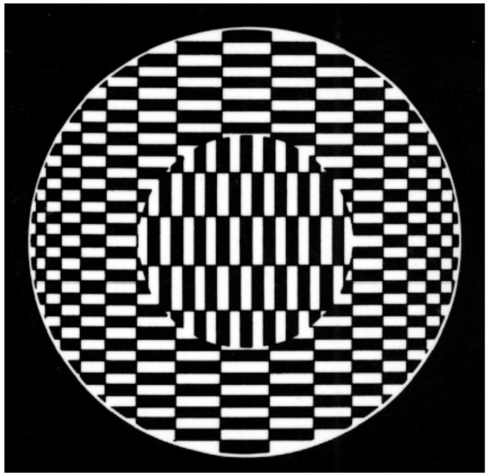
Time: t + dt



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.



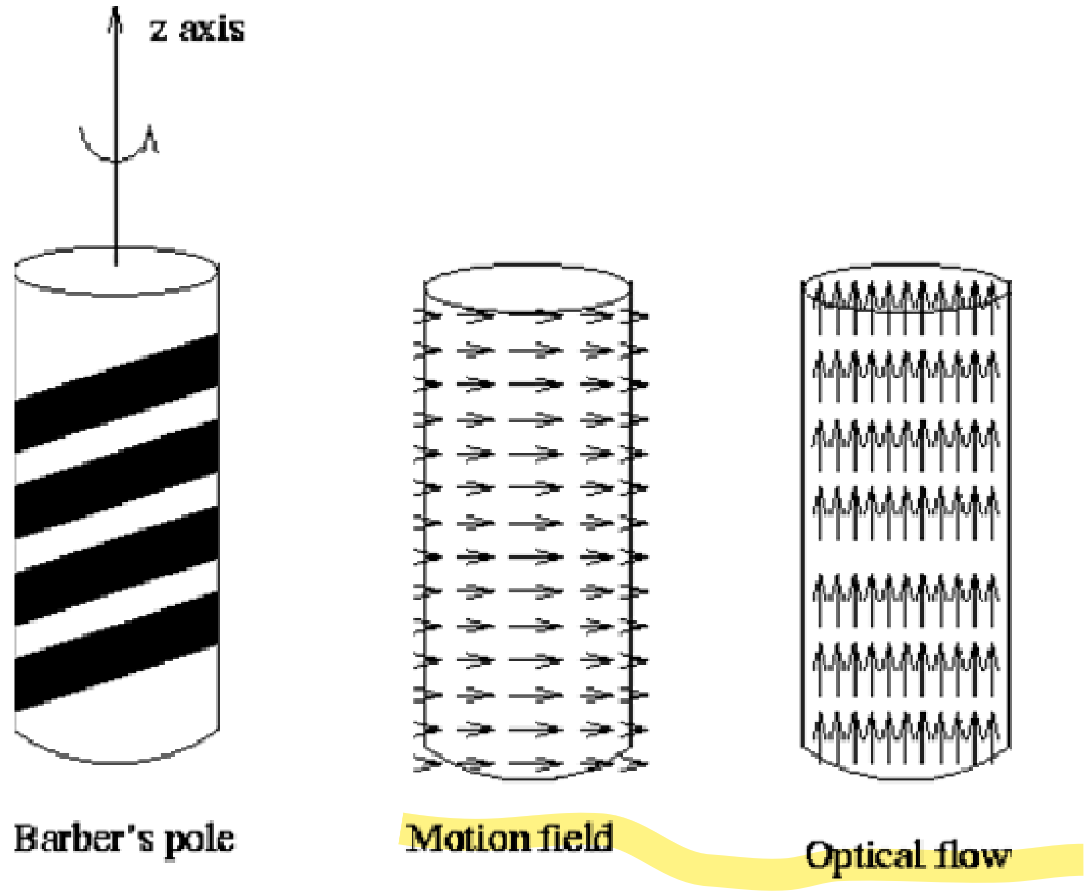
When/where does this break down?

E.g.: In what situations does the displacement of pixel patches not represent physical movement of points in space?

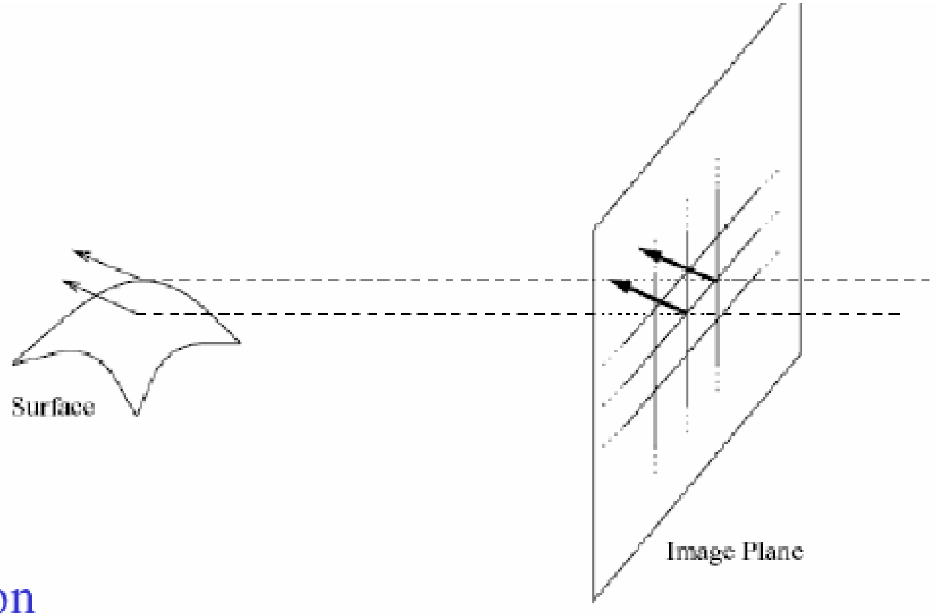
1. Well, TV is based on illusory motion
   – the set is stationary yet things seem to move

2. A uniform rotating sphere
   – nothing seems to move, yet it is rotating

Barber pole illusion



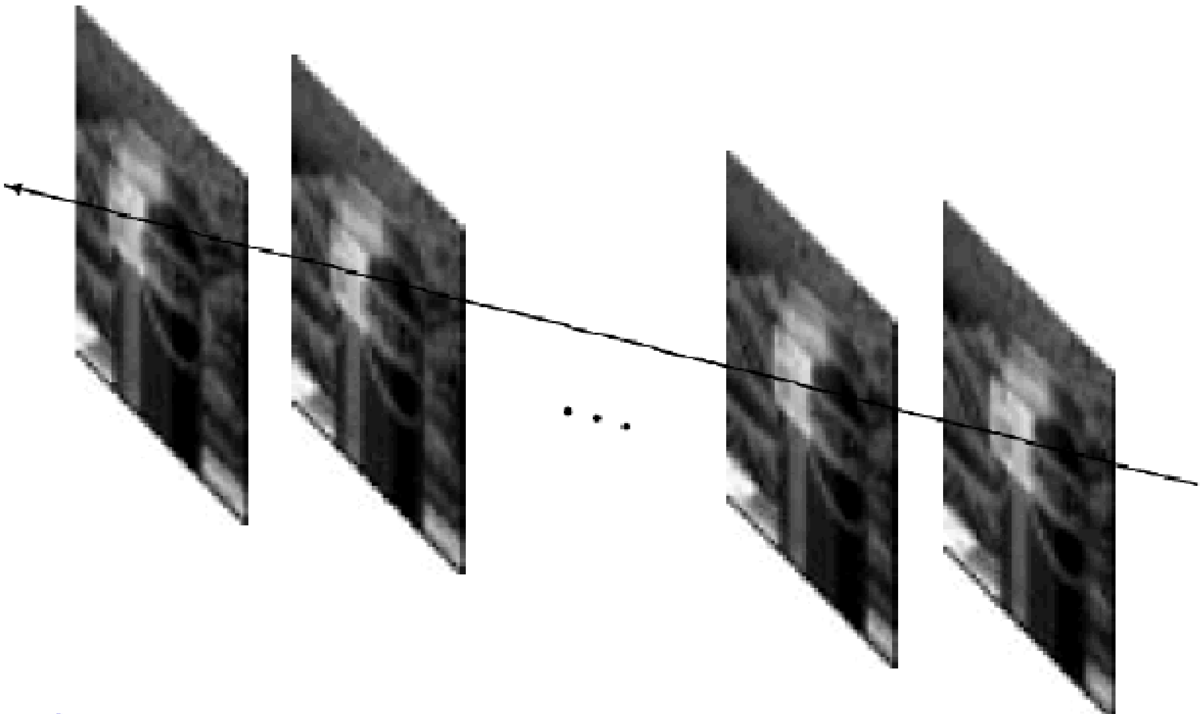
Spatial Coherence



Assumption

* Neighboring points in the scene typically belong to the same surface and hence typically have similar motions.
* Since they also project to nearby points in the image, we expect spatial coherence in image flow.

Temporal Persistence



Assumption:

The image motion of a surface patch changes gradually over time.

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!

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

$$\underbrace{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}}_{\substack{A \\ 25\times 2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\substack{d \\ 2\times 1}} = - \underbrace{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}_{\substack{b \\ 25\times 1}}$$

RGB Image

- 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*3 equations per pixel!

$$0 = I_t(\mathbf{p_i})[0,1,2] + \nabla I(\mathbf{p_i})[0,1,2] \cdot [u \;\; v]$$

$$\underset{\substack{75\text{x}2}}{\underbrace{\begin{bmatrix} I_x(\mathbf{p}_1)[0] & I_y(\mathbf{p}_1)[0] \\ I_x(\mathbf{p}_1)[1] & I_y(\mathbf{p}_1)[1] \\ I_x(\mathbf{p}_1)[2] & I_y(\mathbf{p}_1)[2] \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25})[0] & I_y(\mathbf{p}_{25})[0] \\ I_x(\mathbf{p}_{25})[1] & I_y(\mathbf{p}_{25})[1] \\ I_x(\mathbf{p}_{25})[2] & I_y(\mathbf{p}_{25})[2] \end{bmatrix}}_{A}} \underset{\substack{2\text{x}1}}{\underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d}} = - \underset{\substack{75\text{x}1}}{\underbrace{\begin{bmatrix} I_t(\mathbf{p}_1)[0] \\ I_t(\mathbf{p}_1)[1] \\ I_t(\mathbf{p}_1)[2] \\ \vdots \\ I_t(\mathbf{p}_{25})[0] \\ I_t(\mathbf{p}_{25})[1] \\ I_t(\mathbf{p}_{25})[2] \end{bmatrix}}_{b}}$$

- Prob: we have more equations than unknowns

$$\underset{25\times2}{A}\ \underset{2\times1}{d} = \underset{25\times1}{b} \longrightarrow \text{minimize } \|Ad-b\|^2$$

- Solution: solve least squares problem
  - minimum least squares solution given by solution (in d) of:

$$\underset{2\times2}{(A^T A)}\ \underset{2\times1}{d} = \underset{2\times1}{A^T b}$$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

  - The summations are over all pixels in the K x K window
  - This technique was first proposed by Lukas & Kanade (1981)

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

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

When is This Solvable?

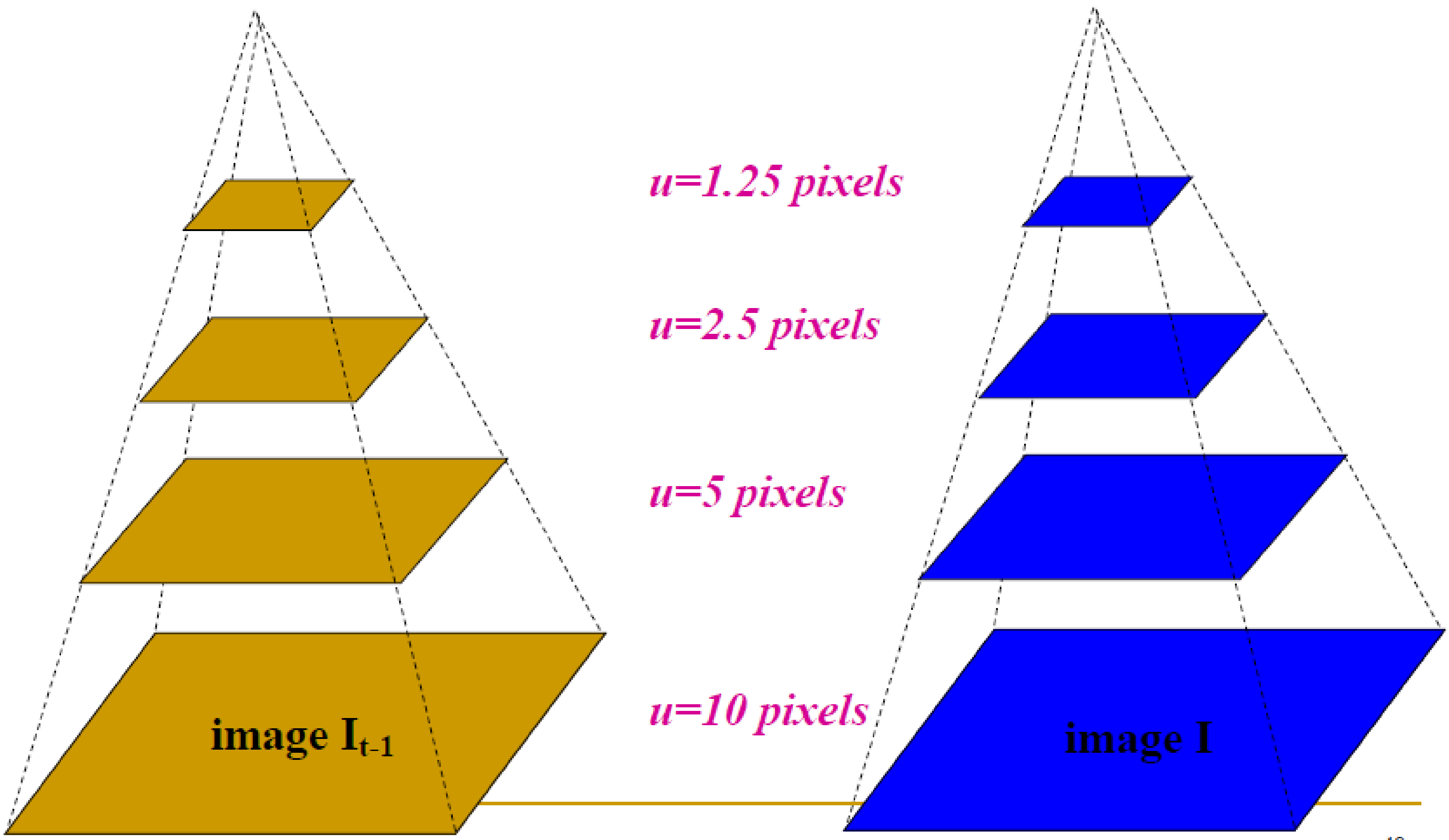
- **$A^T A$** should be invertible
- **$A^T A$** should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of **$A^T A$** should not be too small
- **$A^T A$** should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

- 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?

Iterative Lukas-Kanade Algorithm

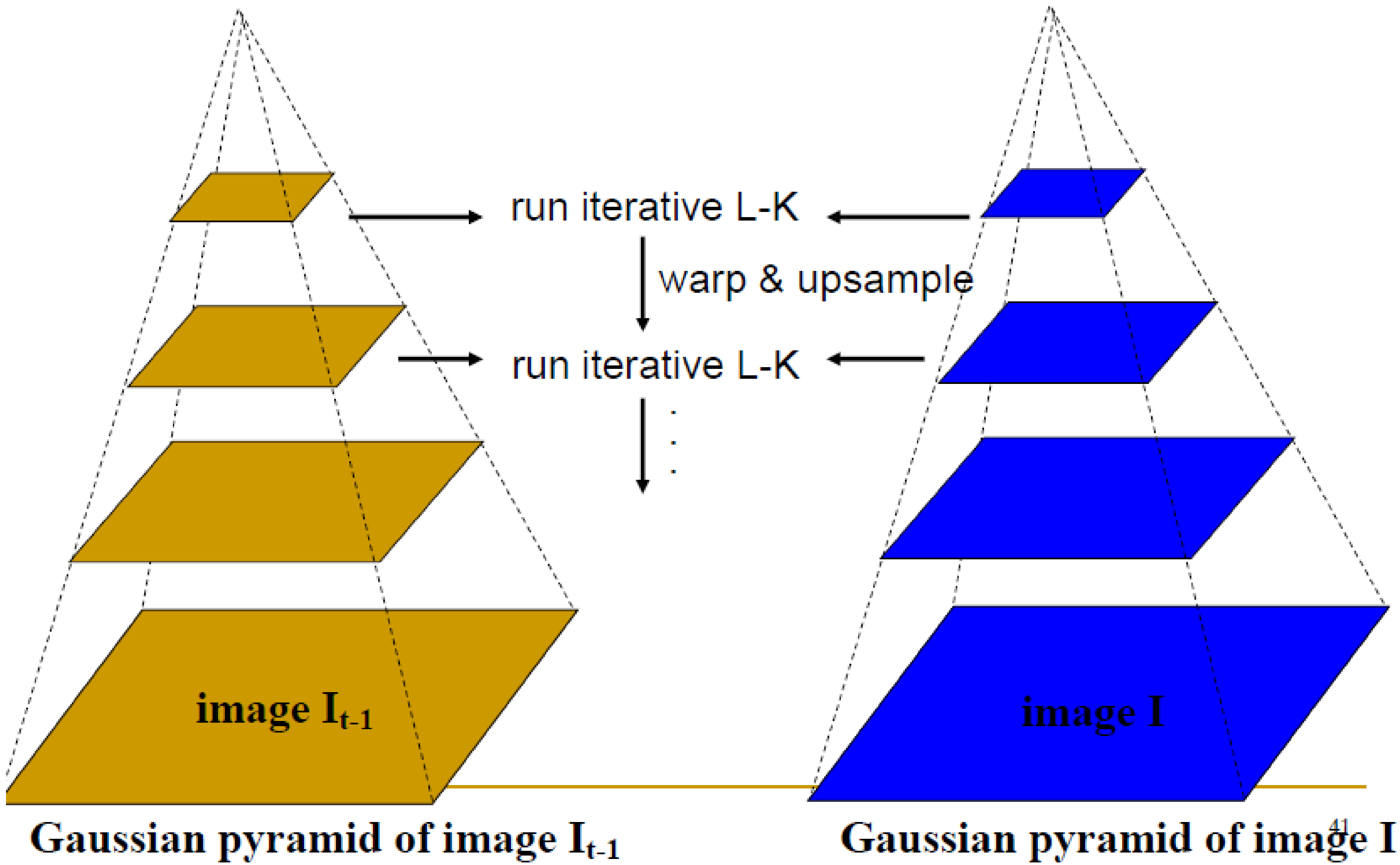
1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp I(t-1) towards I(t) using the estimated flow field
   - *use image warping techniques*
3. Repeat until convergence

Coarse-to-Fine Optical Flow

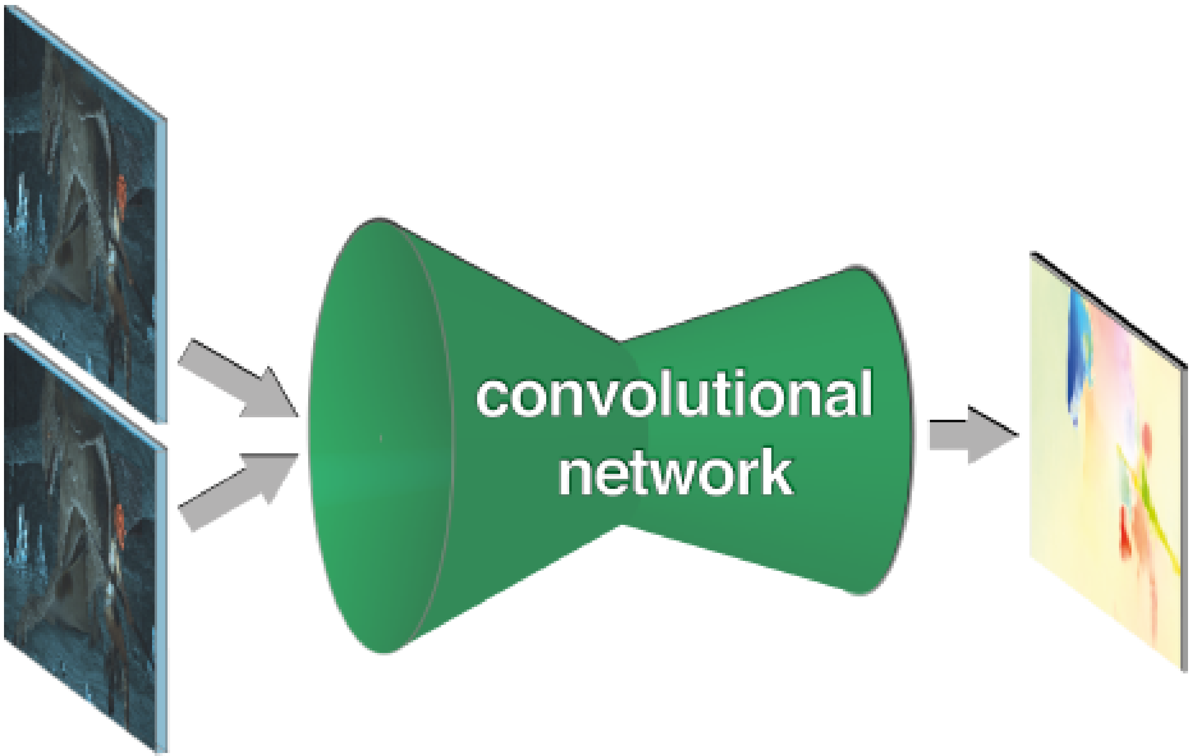


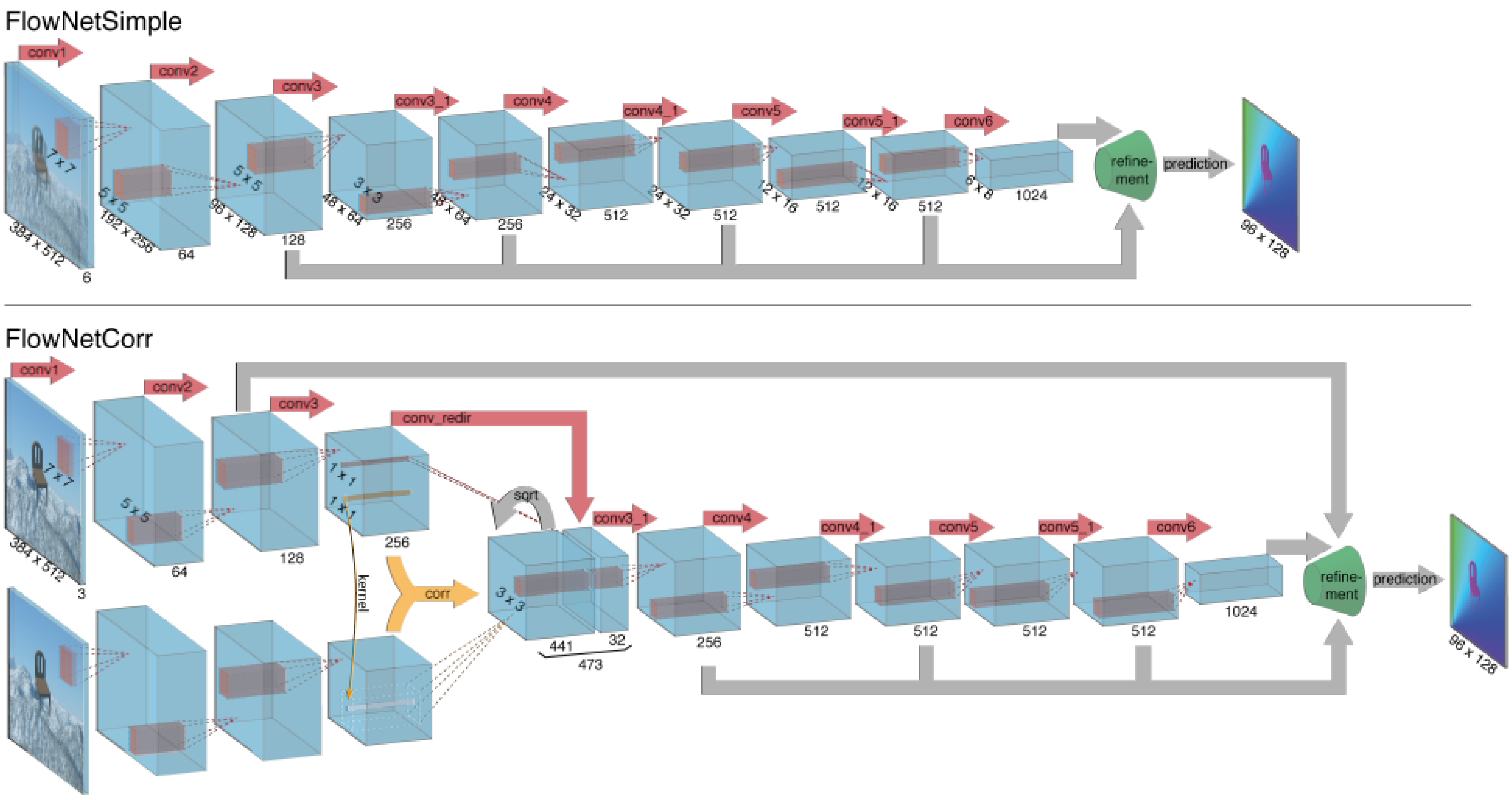
Gaussian pyramid of image I_{t-1} Gaussian pyramid of image I

Coarse-to-Fine Optical Flow

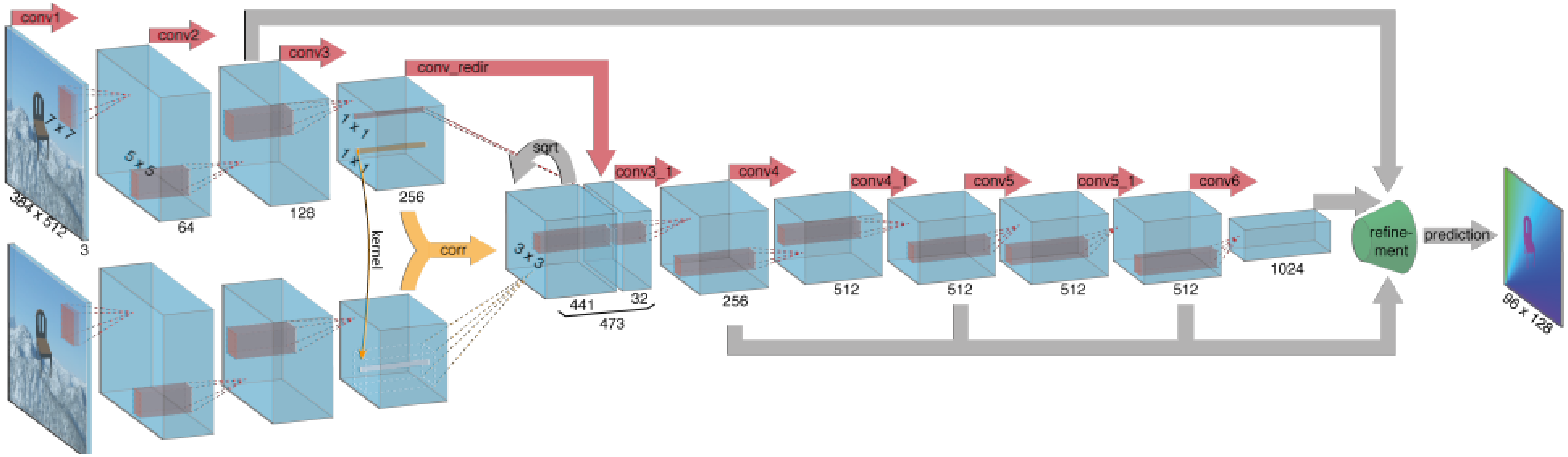


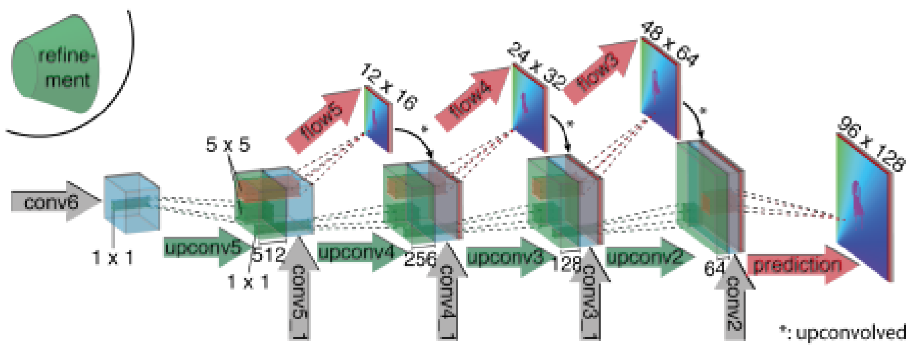
Optical Flow using CNN→ FlowNet




FlowNetSimple
conv1
conv2
conv3
conv3_1
conv4
conv4_1
conv5
conv5_1
conv6
7 x 7
5 x 5
5 x 5
3 x 3
384 x 512
6
192 x 256
64
96 x 128
128
48 x 64
256
48 x 64
256
24 x 32
512
24 x 32
512
12 x 16
512
12 x 16
512
6 x 8
1024
refine-
ment
prediction
96 x 128
FlowNetCorr
conv1
conv2
conv3
conv_redir
7 x 7
5 x 5
1 x 1
1 x 1
384 x 512
3
64
128
256
kernel
corr
sqrt
conv3_1
conv4
conv4_1
conv5
conv5_1
conv6
3 x 3
441
32
473
256
512
512
512
512
1024
refine-
ment
prediction
96 x 128

FlowNetCorr






Ground truth
FlowNetS
FlowNetS+v
EPE: 0.89
EPE: 0.42
EPE: 8.11
EPE: 6.54