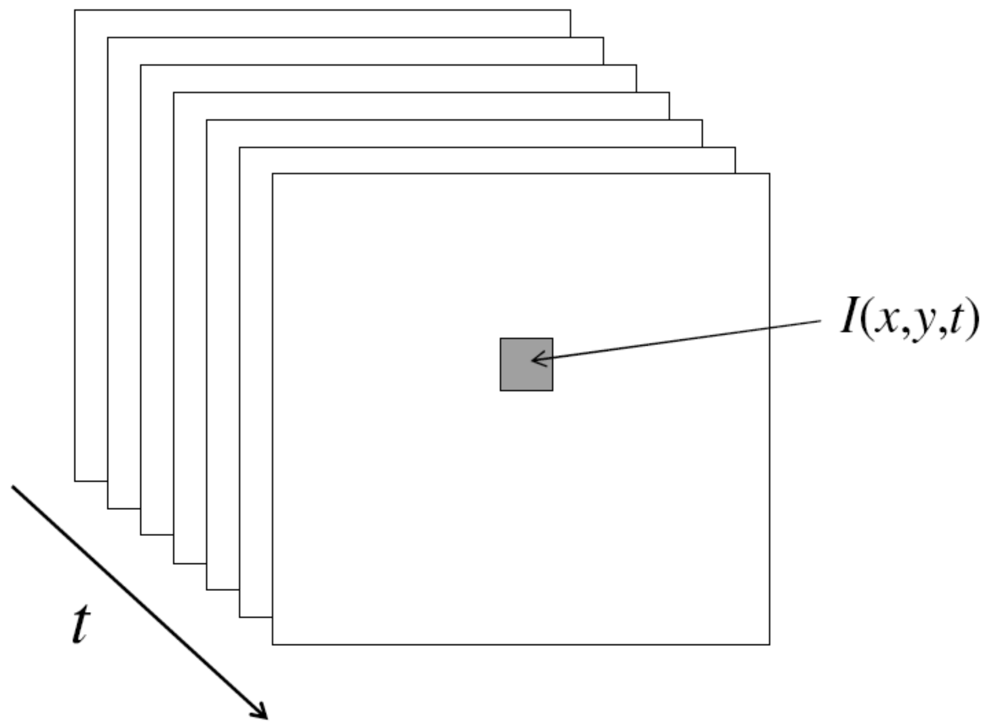
Motion Estimation

Gang Pan
Zhejiang University

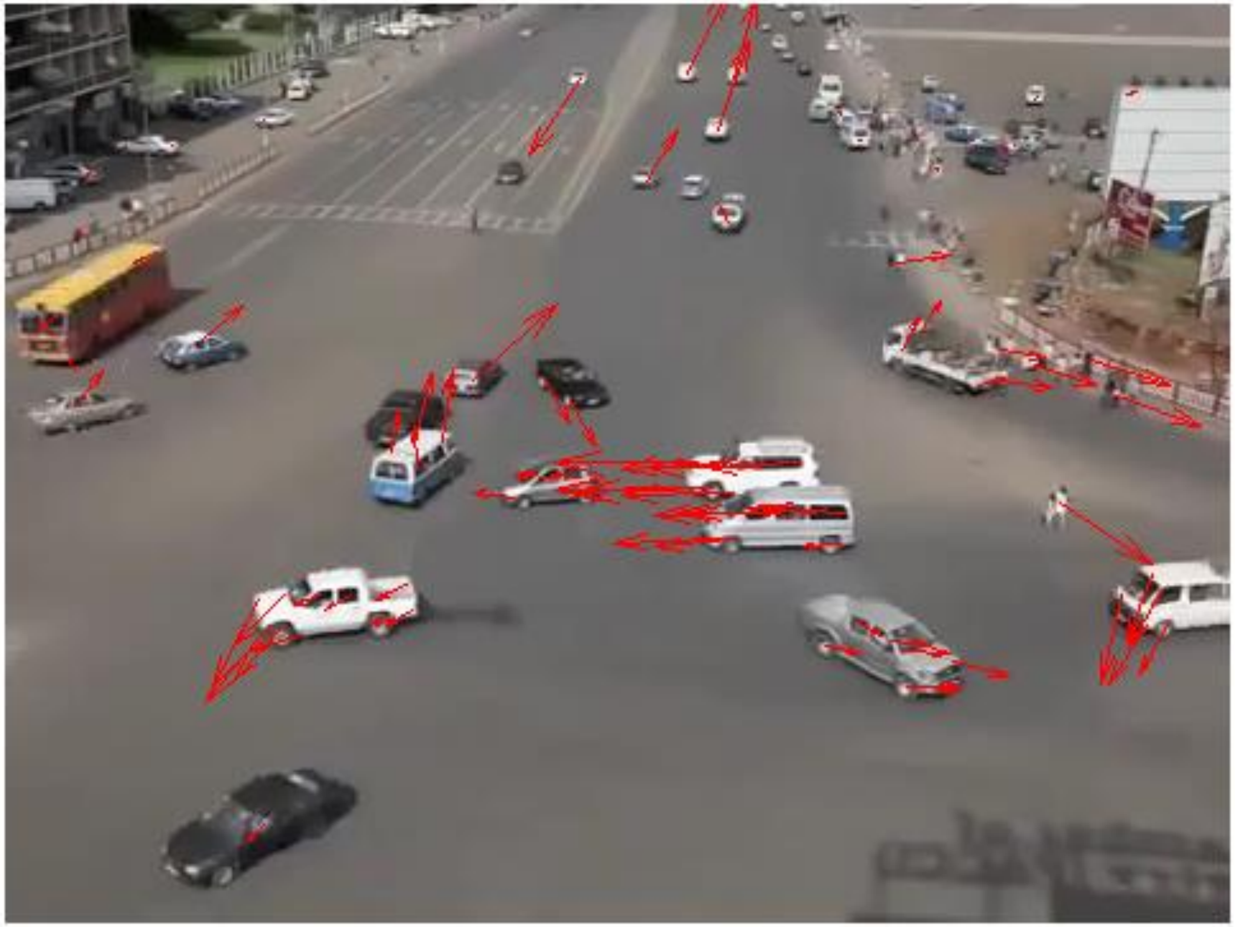
From images to videos

- A video is a sequence of frames captured over time
- Now our image data is a function of space (x, y) and time (t)



Why is motion useful?

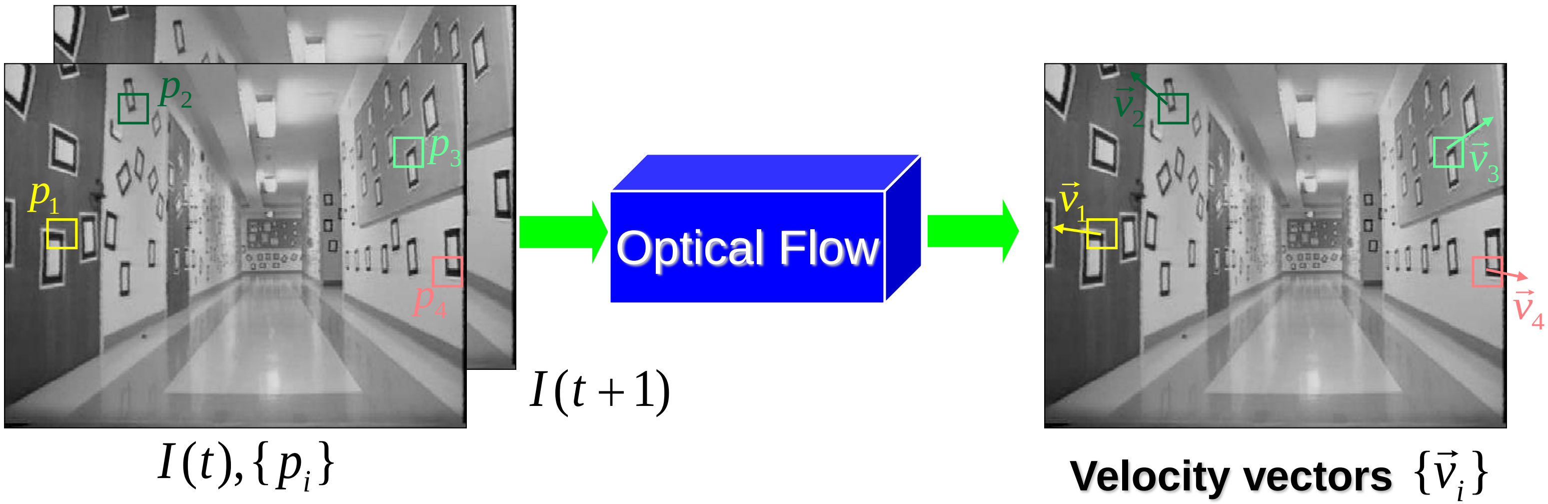
Why is motion useful?



Motion Estimation

- Lots of uses
  - Track object behavior
  - Correct for camera jitter (stabilization)
  - Align images (mosaics)
  - 3D shape reconstruction
  - Special effects

What is Optical Flow?



Optical flow

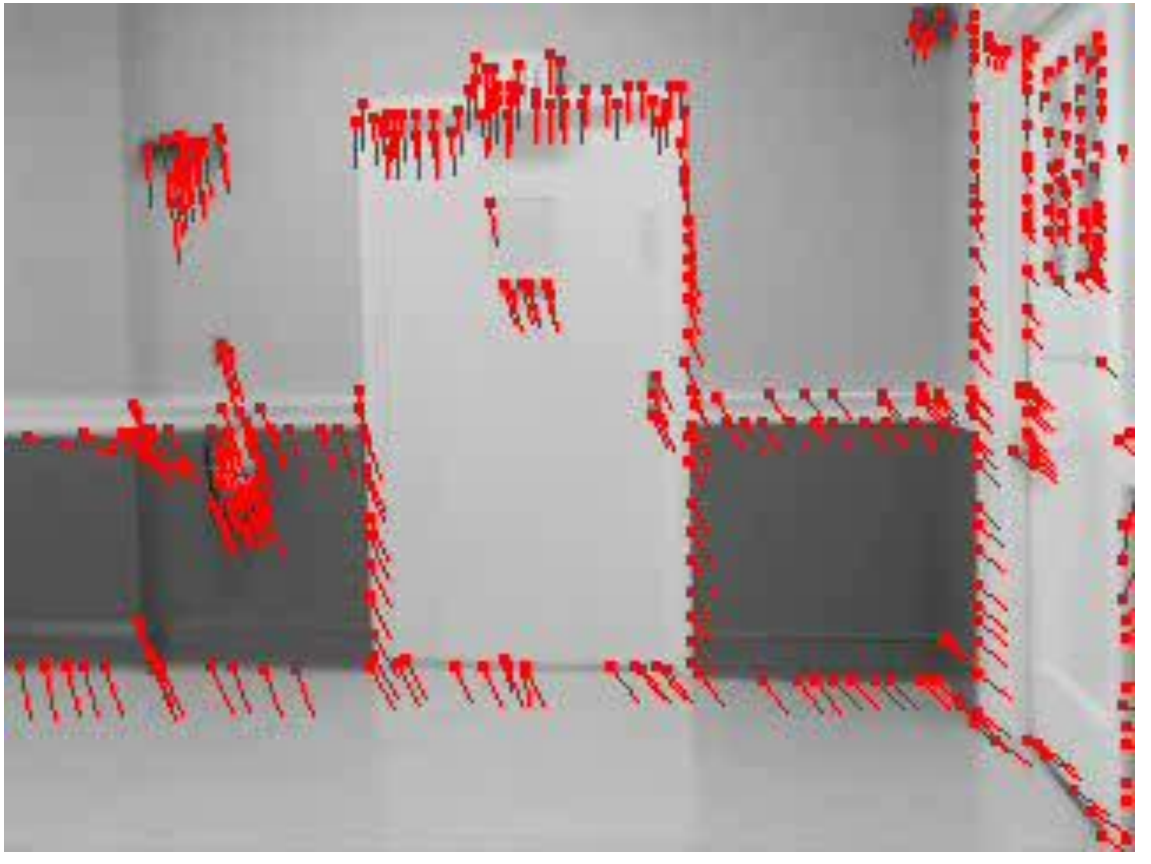
- Definition: optical flow is the *apparent* motion of brightness patterns in the image

- Note: apparent motion can be caused by lighting changes without any actual motion
  - Think of a uniform rotating sphere under fixed lighting vs. a stationary sphere under moving illumination

GOAL: Recover image motion at each pixel from optical flow

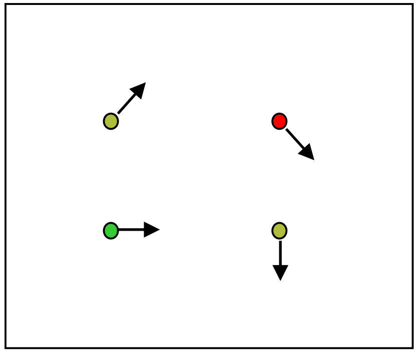
Source: Silvio Savarese

Optical flow

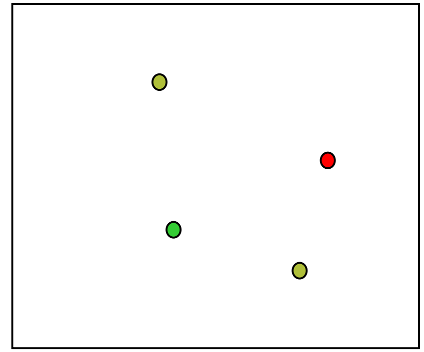


"Joint Tracking of Features and Edges", CVPR2008

Problem definition: optical flow



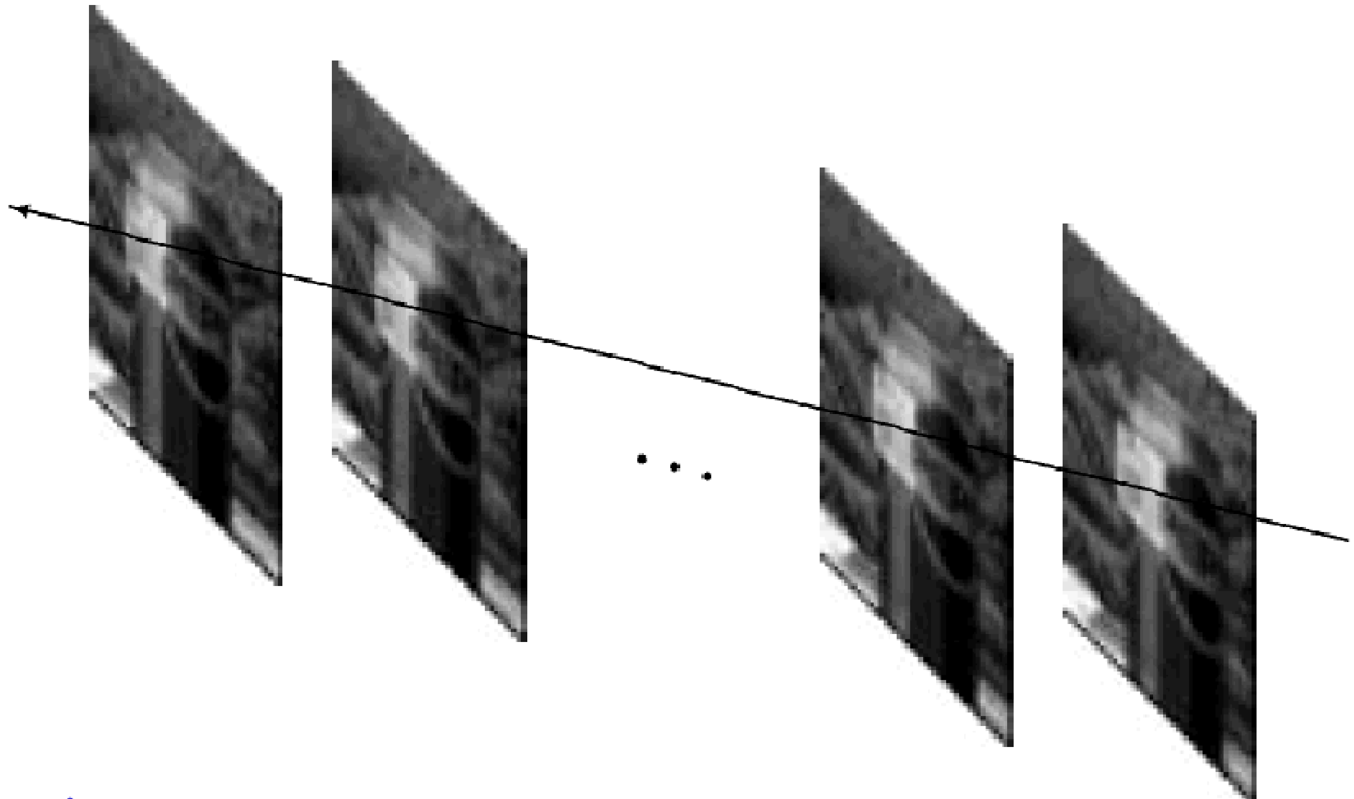
$H(x, y)$



$I(x, y)$

- How to estimate pixel motion from image H to image I?
  - Solve pixel correspondence problem
    - given a pixel in H, look for nearby pixels of the same color in I
- Key assumptions
  - **brightness constancy**
  - **spatial coherence**
  - **small motion**
- This is called the **optical flow** problem

Key Assumptions: small motions

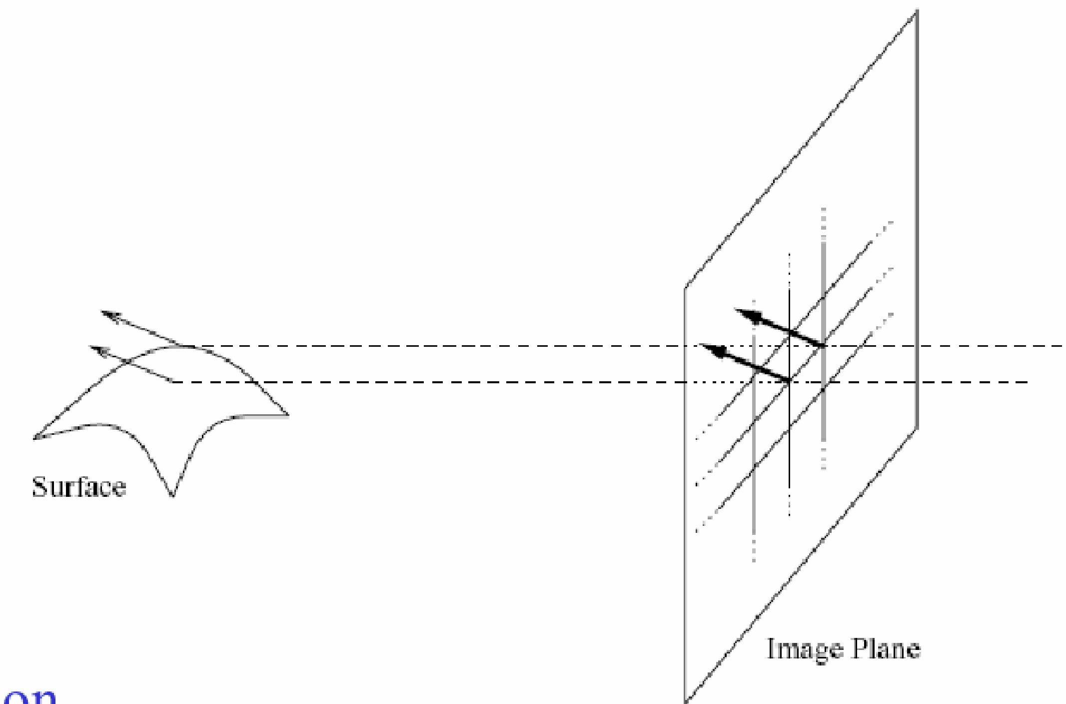


Assumption:

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

* Slide from Michael Black, CS143 2003

Key Assumptions: spatial coherence

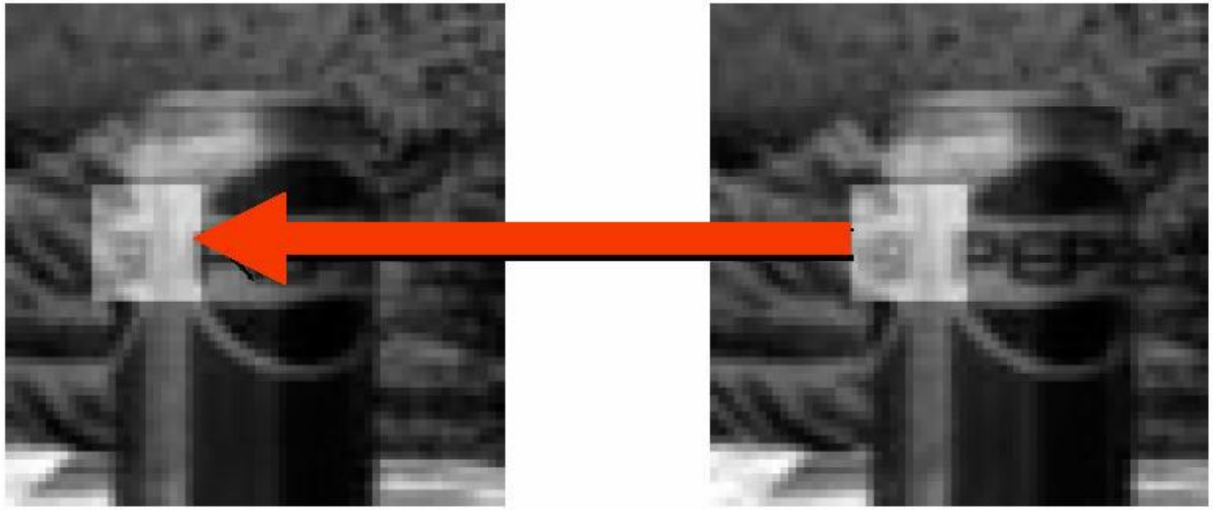


Assumption

* 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.

* Slide from Michael Black, CS143 2003

Key Assumptions: brightness Constancy



Assumption

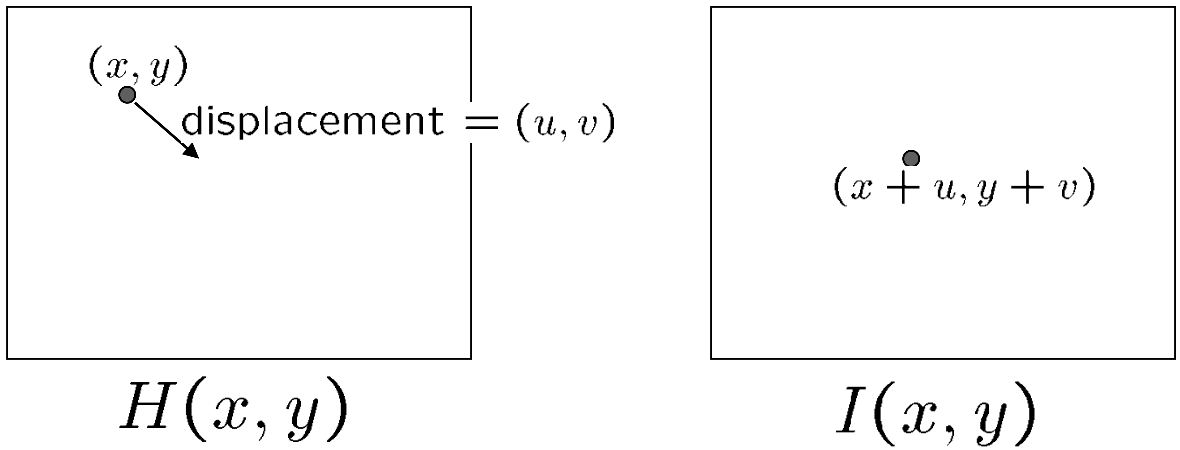
Image measurements (e.g. brightness) in a small region remain the same although their location may change.

$$I(x+u, y+v, t+1) = I(x, y, t)$$

(assumption)

* Slide from Michael Black, CS143 2003

Optical flow constraints (grayscale images)



- Let's look at these constraints more closely
  - brightness constancy: Q: what's the equation?
  - small motion: (u and v are less than 1 pixel)
    - suppose we take the Taylor series expansion of I:

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

Optical flow equation

- Combining these two equations

$$
\begin{aligned}
0 &= I(x+u, y+v) - H(x,y) \\
&\approx I(x,y) + I_x u + I_y v - H(x,y) \quad \text{shorthand: } I_x = \frac{\partial I}{\partial x} \\
&\approx (I(x,y) - H(x,y)) + I_x u + I_y v \\
&\approx I_t + I_x u + I_y v \\
&\approx I_t + \nabla I \cdot [u \;\; v]
\end{aligned}
$$

- In the limit as u and v go to zero, this becomes exact

$$\nabla I \cdot [u \;\; v]^T + I_t = 0$$

The brightness constancy constraint

Can we use this equation to recover image motion (u,v) at each pixel?

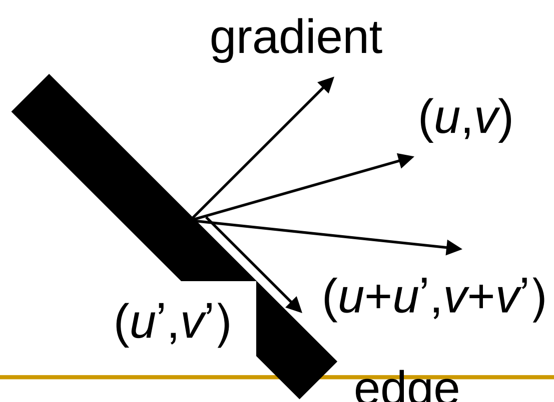
$$\nabla I \cdot [u \ \ v]^T + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns (u,v)

The component of the flow perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (u, v) satisfies the equation,
so does ($u+u'$, $v+v'$) if

$$\nabla I \cdot [u' \ \ v']^T = 0$$



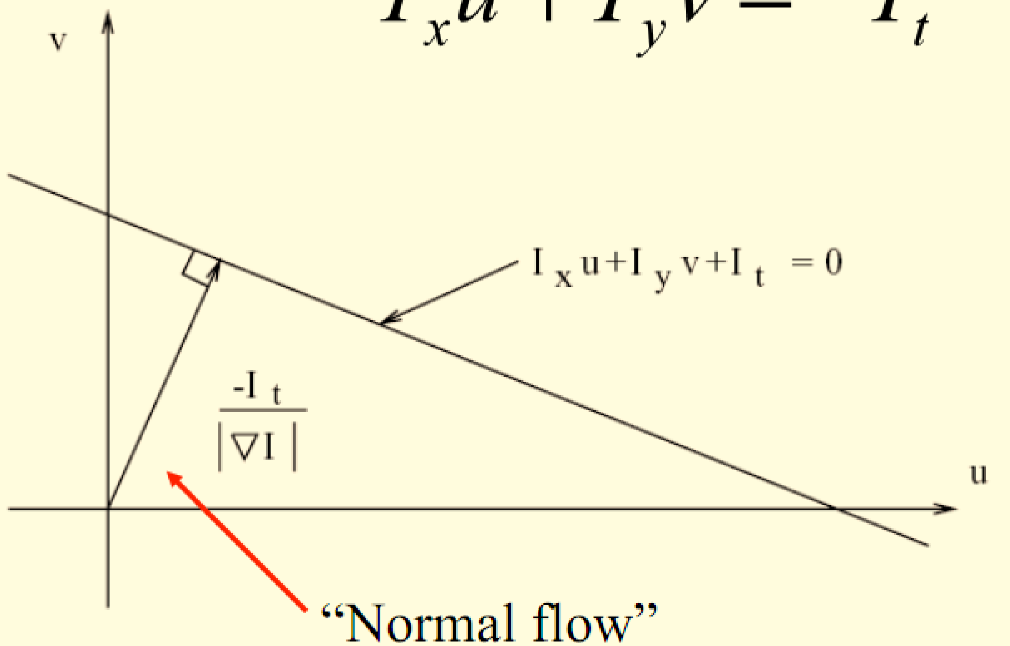
Source: Silvio Savarese

Optical flow equation

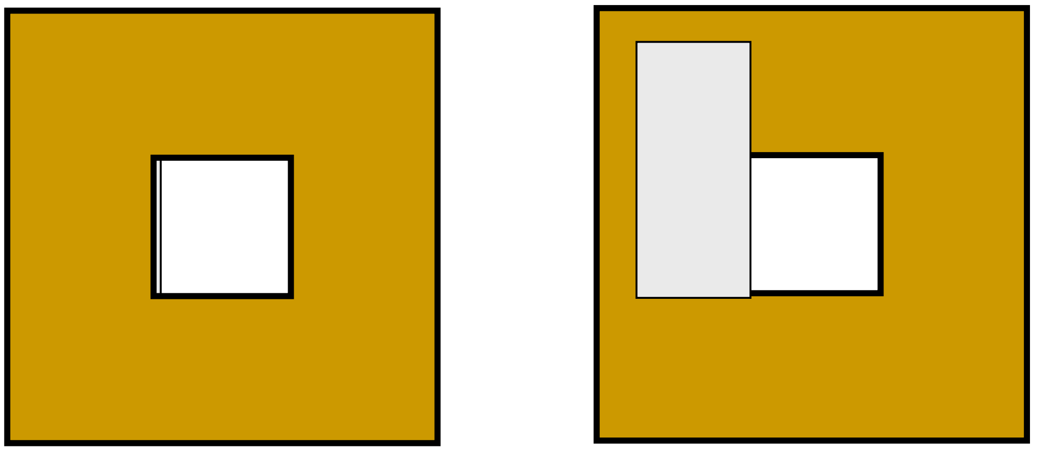
$$0 = I_t + \nabla I \cdot [u \;\; v]$$

At a single image pixel, we get a line:

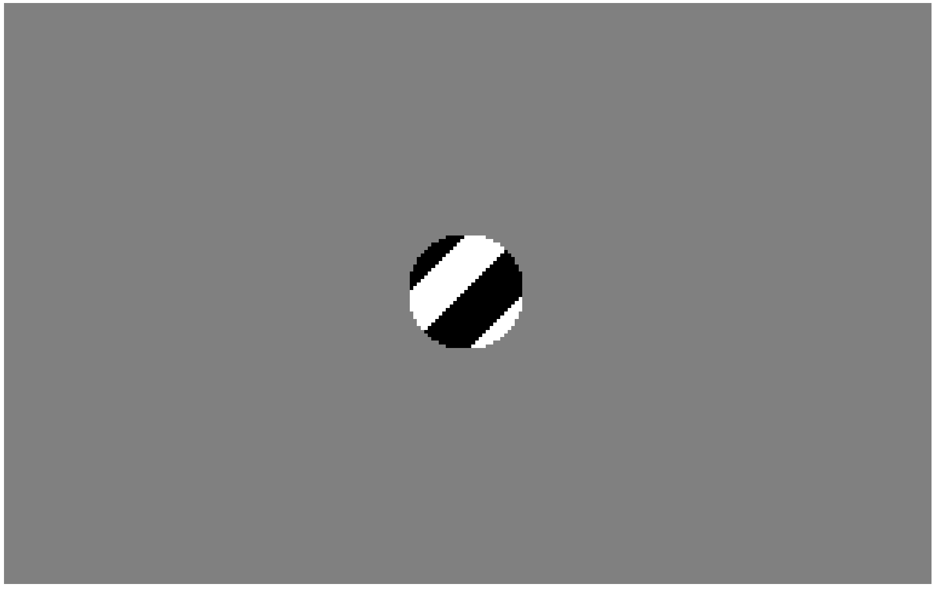
$$I_x u + I_y v = -I_t$$



Aperture problem



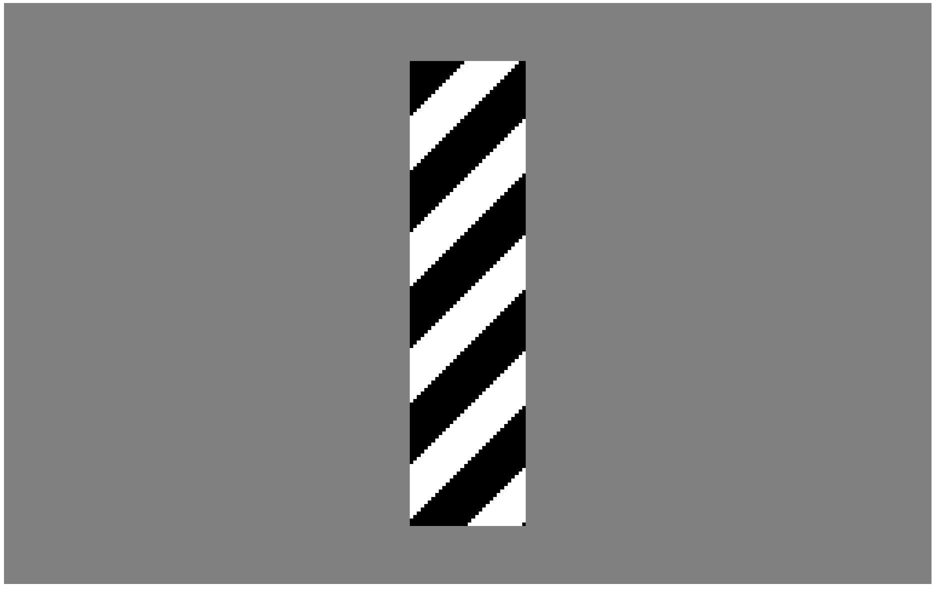
The barber pole illusion



http://en.wikipedia.org/wiki/Barberpole_illusion

Source: Silvio Savarese

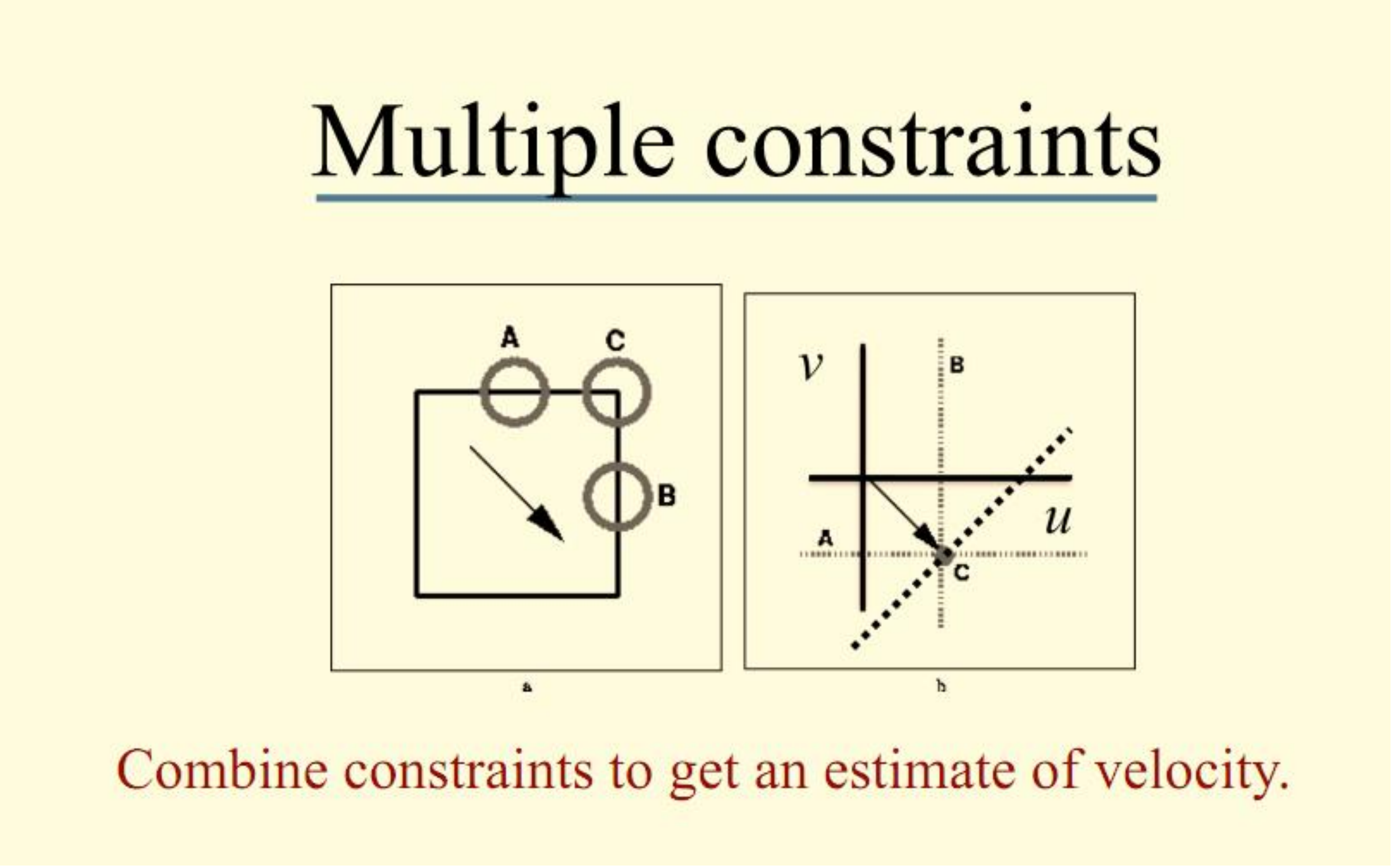
The barber pole illusion



http://en.wikipedia.org/wiki/Barberpole_illusion

Source: Silvio Savarese

Multiple constraints



Optical flow algorithms

- Lucas-Kandade 1981
  - Originally for dense optical flow, now popular for sparse optical flow
- Horn-Schunk 1981
  - For dense optical flow
- Block matching

Lucas-Kanade flow

5x5 window: $\underset{25\times2}{A}\ \underset{2\times1}{d} = \underset{25\times1}{b}$ $\longrightarrow$ minimize $\|Ad - b\|^2$

- Problem: we have more equations than unknowns
  - 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 Lucas & Kanade (1981)

RGB version

- 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!

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

$$\underset{\substack{A \\ 75\times 2}}{\begin{bmatrix} I_x(\mathbf{p_1})[0] & I_y(\mathbf{p_1})[0] \\ I_x(\mathbf{p_1})[1] & I_y(\mathbf{p_1})[1] \\ I_x(\mathbf{p_1})[2] & I_y(\mathbf{p_1})[2] \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}})[0] & I_y(\mathbf{p_{25}})[0] \\ I_x(\mathbf{p_{25}})[1] & I_y(\mathbf{p_{25}})[1] \\ I_x(\mathbf{p_{25}})[2] & I_y(\mathbf{p_{25}})[2] \end{bmatrix}} \underset{\substack{d \\ 2\times 1}}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \underset{\substack{b \\ 75\times 1}}{\begin{bmatrix} I_t(\mathbf{p_1})[0] \\ I_t(\mathbf{p_1})[1] \\ I_t(\mathbf{p_1})[2] \\ \vdots \\ I_t(\mathbf{p_{25}})[0] \\ I_t(\mathbf{p_{25}})[1] \\ I_t(\mathbf{p_{25}})[2] \end{bmatrix}}$$

Conditions for solvability

- 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)

Eigenvectors of A^TA

$$A^TA = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

- Suppose (x,y) is on an edge. What is A^TA?
  - gradients along edge all point the same direction

    $$\left(\sum \nabla I (\nabla I)^T\right) \approx k \nabla I \nabla I^T$$

  - gradients away from edge have small magnitude

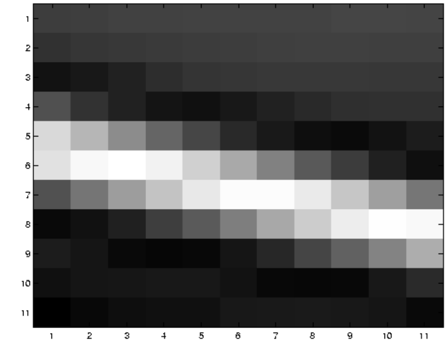
    $$\left(\sum \nabla I (\nabla I)^T\right) \nabla I = k \|\nabla I\|^2 \nabla I$$

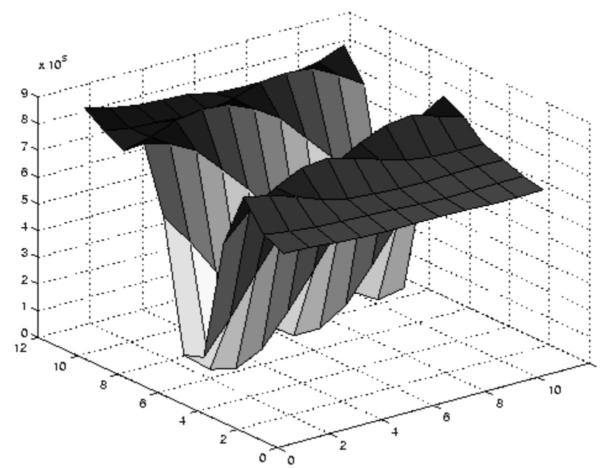
  - ∇I is an eigenvector with eigenvalue $k\|\nabla I\|^2$
  - What's the other eigenvector of A^TA?
    - let N be perpendicular to ∇I

      $$\left(\sum \nabla I (\nabla I)^T\right) N = 0$$

    - N is the second eigenvector with eigenvalue 0
- The eigenvectors of A^TA relate to edge direction and magnitude

Edge

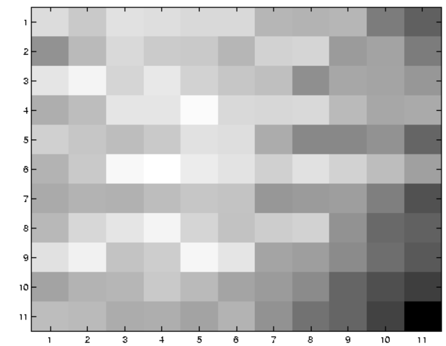


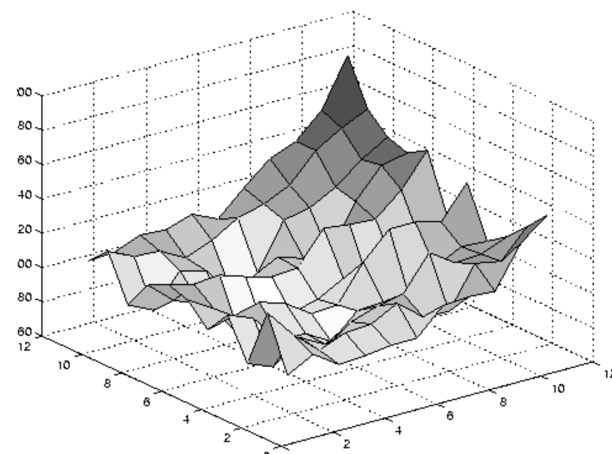


$$\sum \nabla I (\nabla I)^T$$

– large gradients, all the same
– large λ_1, small λ_2

Low texture region



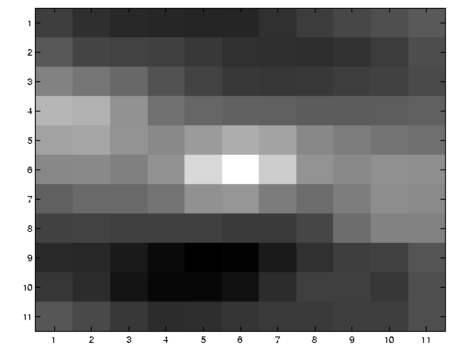


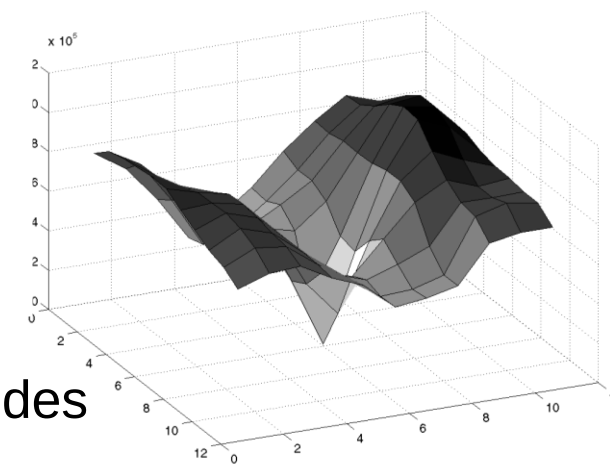
$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small λ_1, small λ_2

High textured region





$$\sum \nabla I (\nabla I)^T$$

– gradients are different, large magnitudes

– large λ_1, large λ_2

Observation

- This is a two-image problem BUT
  - Can measure sensitivity by just looking at one of the images!
  - This tells us which pixels are easy to track, which are hard

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

Improving accuracy

- Recall our small motion assumption

$$0 = I(x+u, y+v) - H(x,y)$$

$$\approx I(x,y) + I_x u + I_y v - H(x,y)$$

- This is not exact
  - To do better, we need to add higher order terms back in:

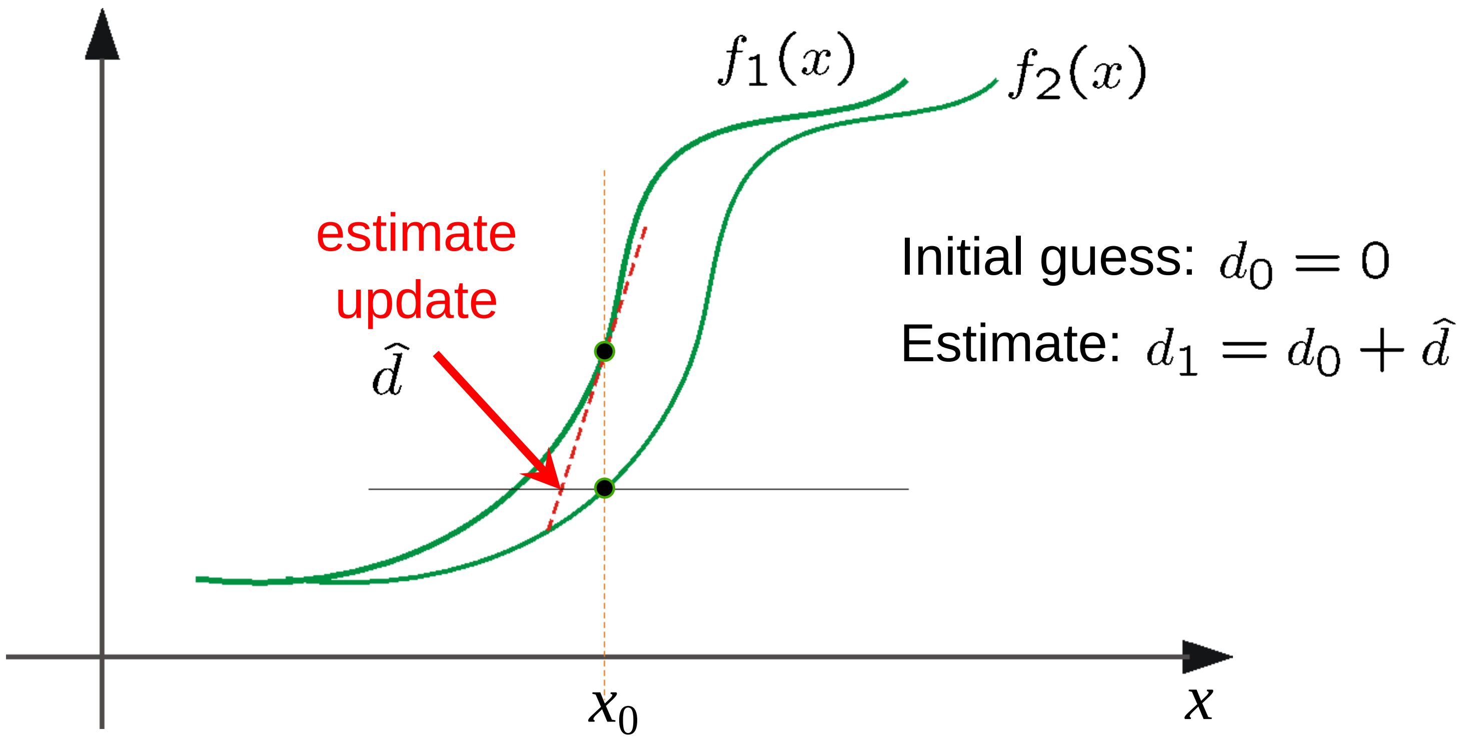
$$= I(x,y) + I_x u + I_y v + \text{higher order terms} - H(x,y)$$

- This is a polynomial root finding problem
  - Can solve using **Newton's method**
    - Also known as **Newton-Raphson** method
  - Approach so far does one iteration of Newton's method
    - Better results are obtained via more iterations

Iterative Refinement

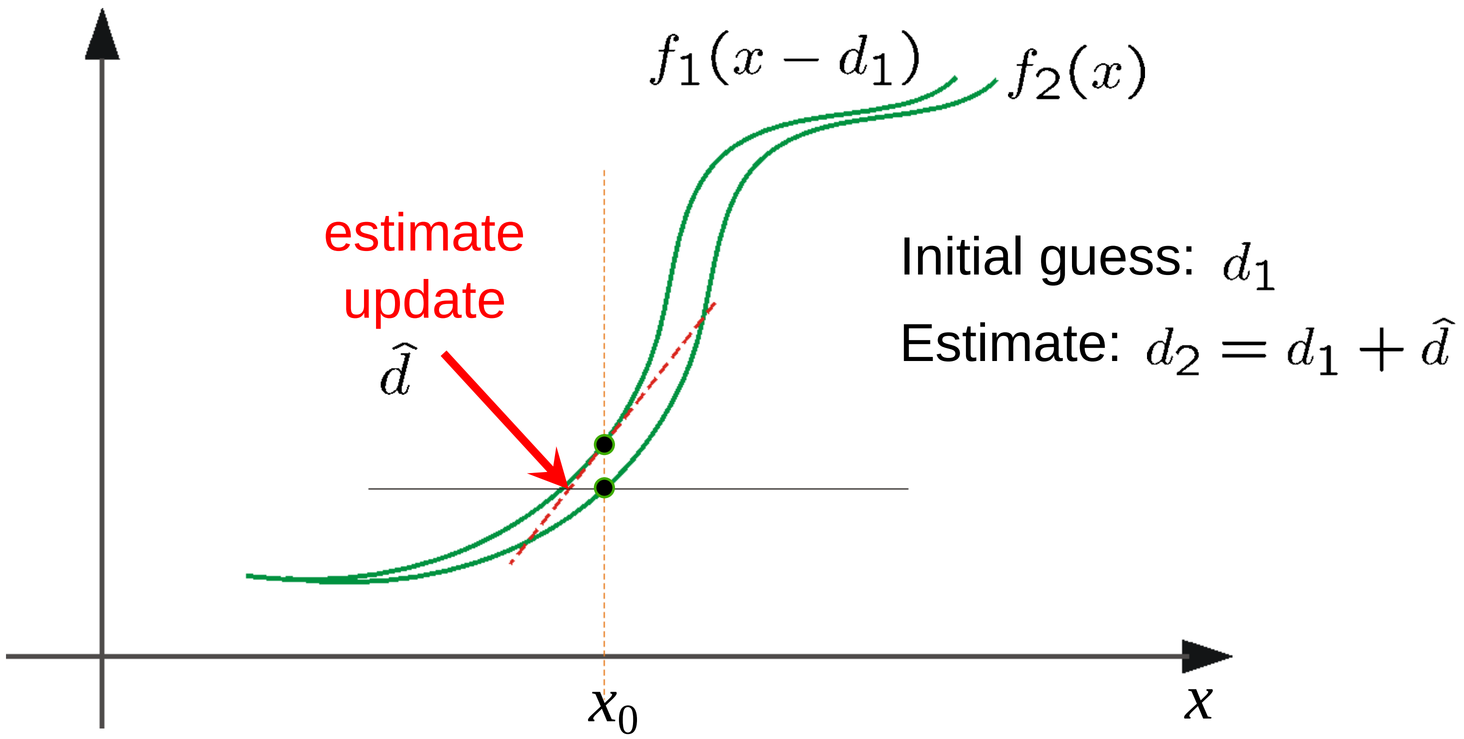
- Iterative Lucas-Kanade Algorithm
  - Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation
  - Warp one image toward the other using the estimated flow field
  - Refine estimate by repeating the process

Optical Flow: Iterative Estimation

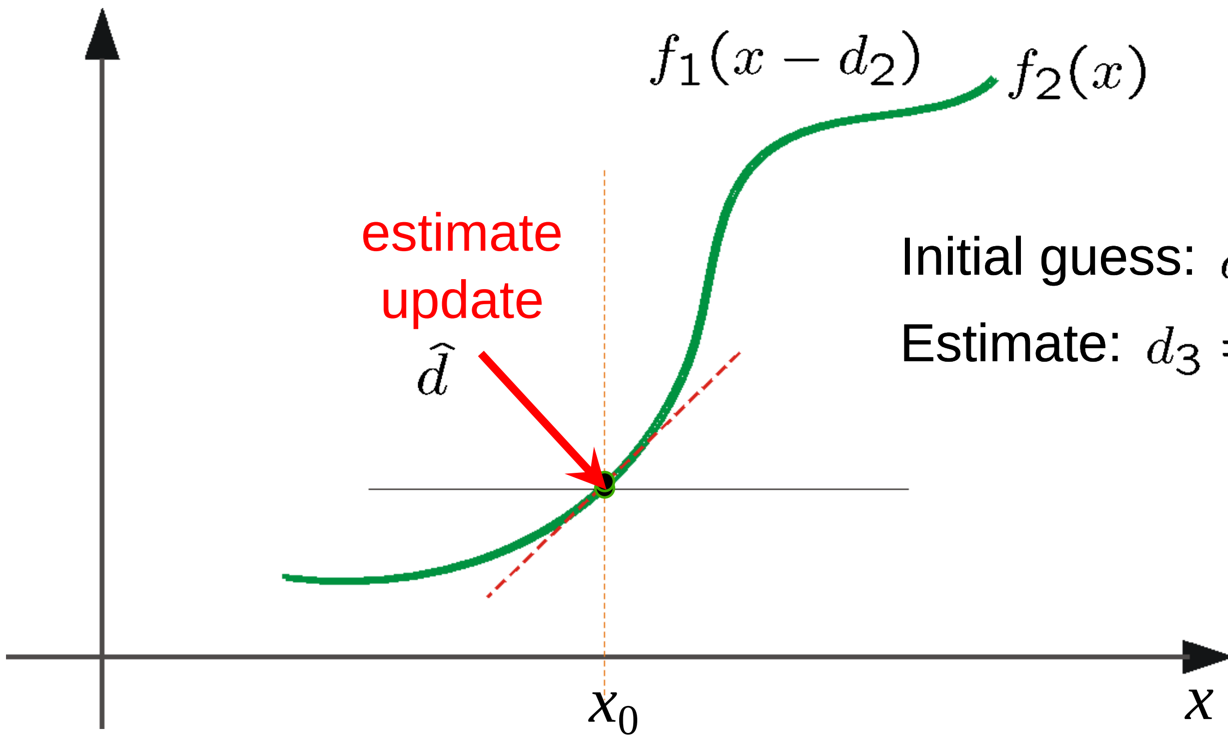


(using d for *displacement* here instead of u)

Optical Flow: Iterative Estimation



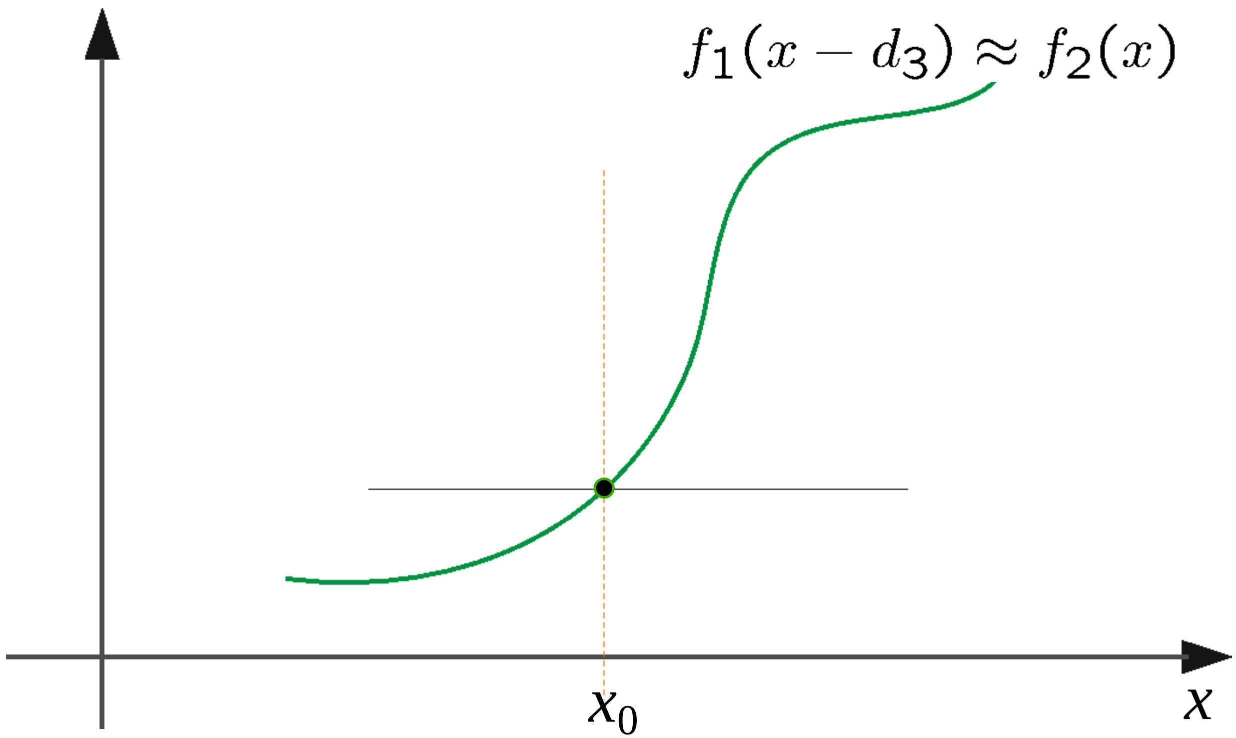
Optical Flow: Iterative Estimation



Initial guess: d_2

Estimate: $d_3 = d_2 + \hat{d}$

Optical Flow: Iterative Estimation



Optical Flow: Iterative Estimation

- Some Implementation Issues:
  - Warping is not easy (ensure that errors in warping are smaller than the estimate refinement)
  - Warp one image, take derivatives of the other so you don’t need to re-compute the gradient after each iteration.
  - Often useful to low-pass filter the images before motion estimation (for better derivative estimation, and linear approximations to image intensity)

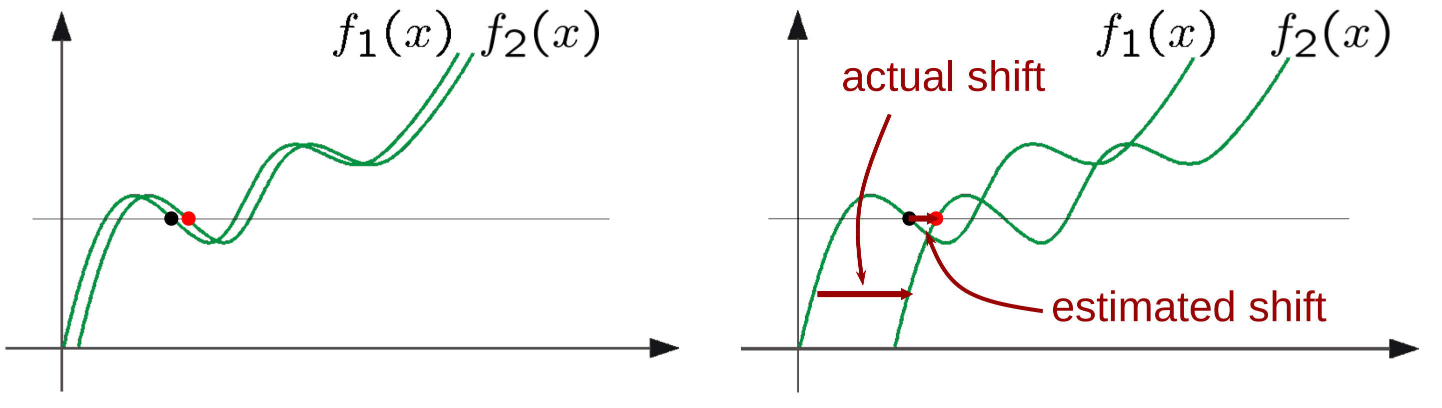
Iterative Refinement

- Iterative Lucas-Kanade Algorithm
  1. Estimate velocity at each pixel by solving Lucas-Kanade equations
  2. Warp H towards I using the estimated flow field
     - *use bilinear interpolation*
     - Repeat until convergence

Optical Flow: Aliasing

Temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity.

I.e., how do we know which ‘correspondence’ is correct?



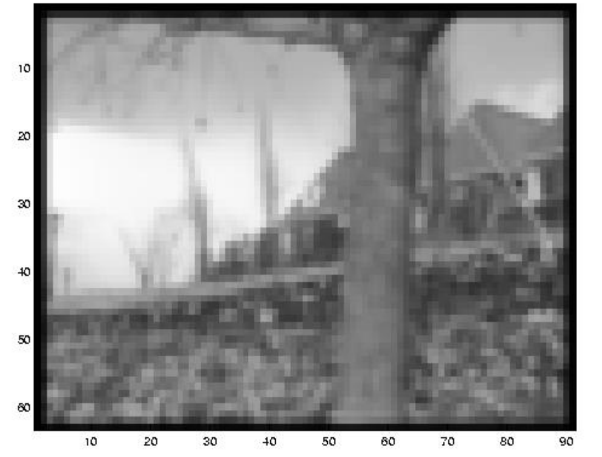
nearest match is correct (no aliasing)

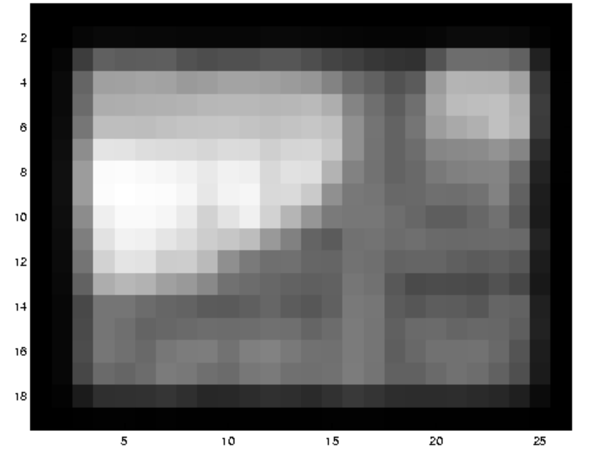
nearest match is incorrect (aliasing)

Revisiting the small motion assumption

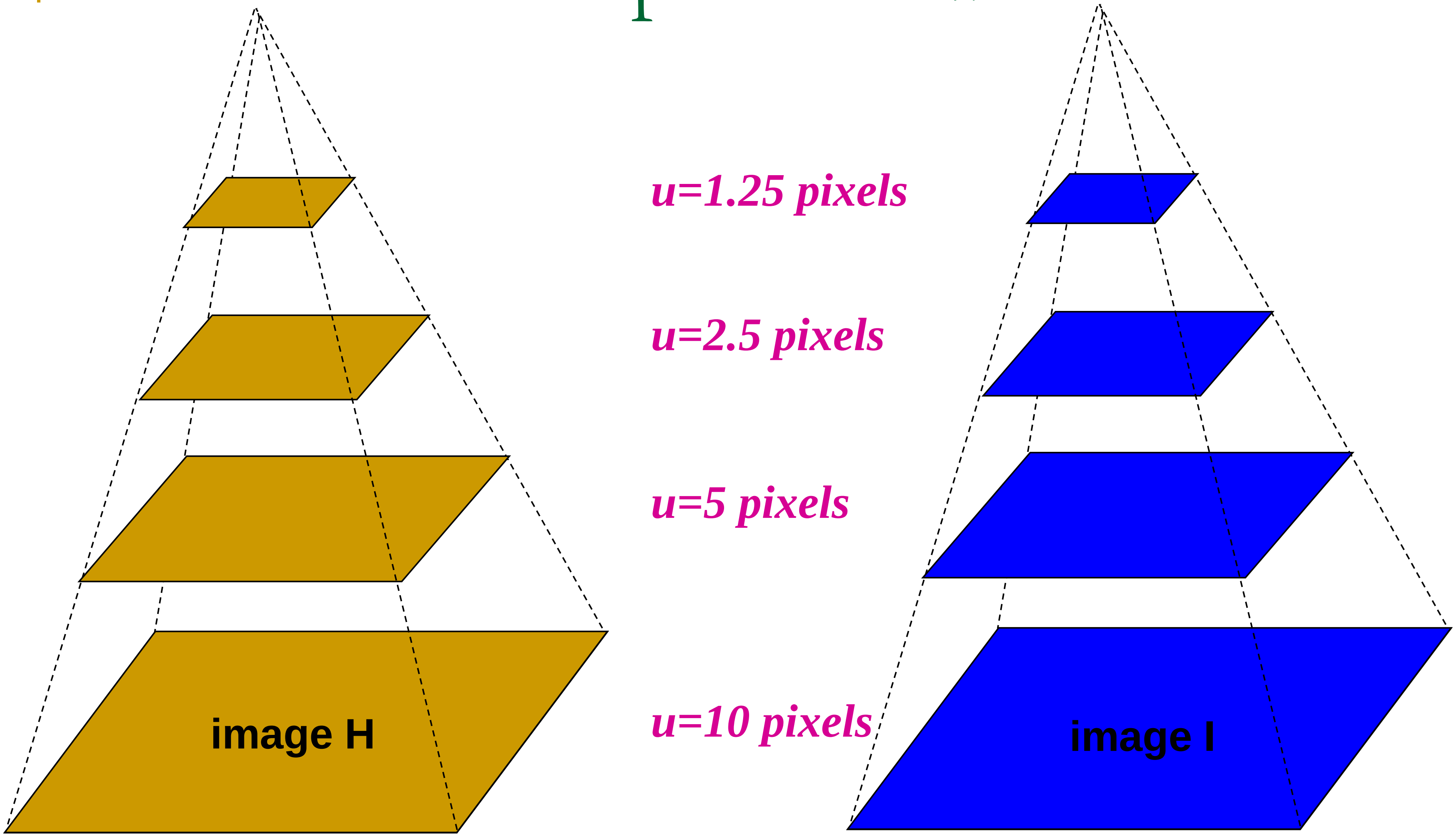
- Is this motion small enough?
  - Probably not—it's much larger than one pixel (2^{nd} order terms dominate)
  - How might we solve this problem?

Reduce the resolution!





Coarse-to-fine optical flow estimation



u=1.25 pixels

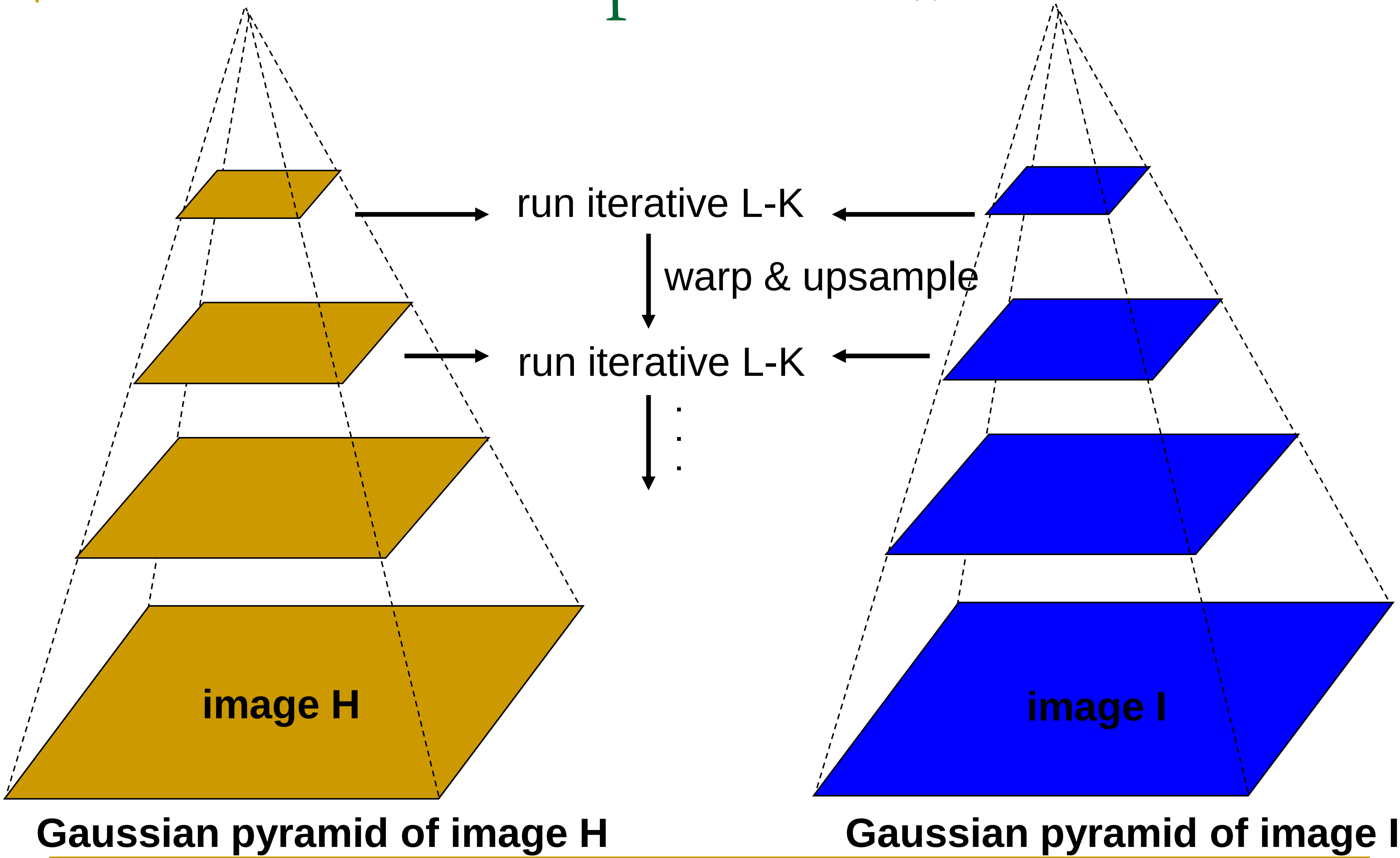
u=2.5 pixels

u=5 pixels

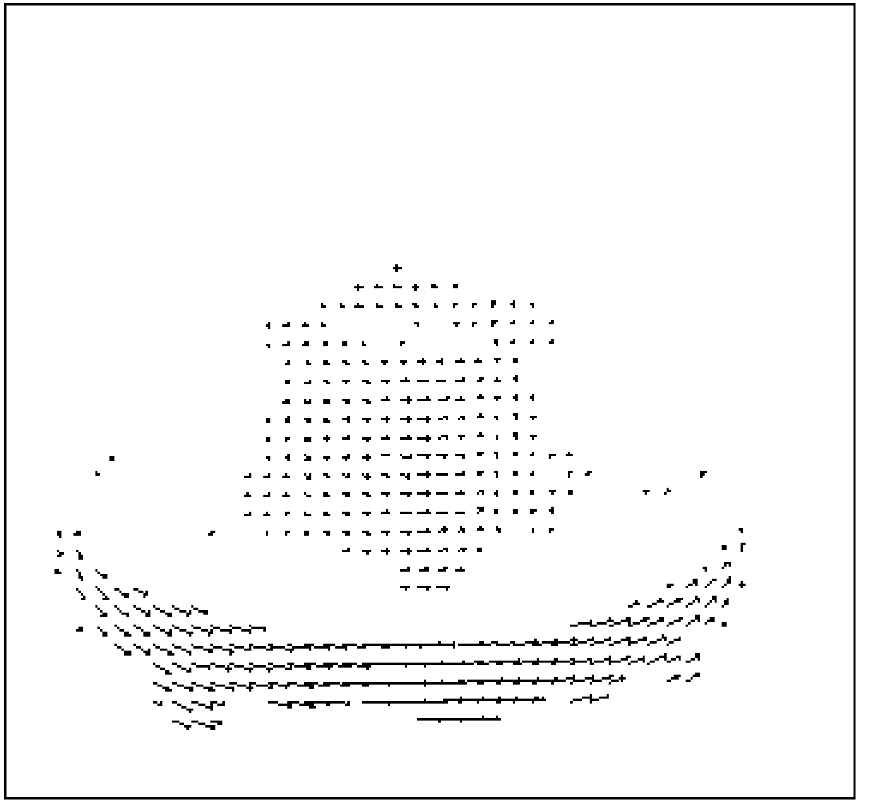
u=10 pixels

Gaussian pyramid of image H **Gaussian pyramid of image I**

Coarse-to-fine optical flow estimation

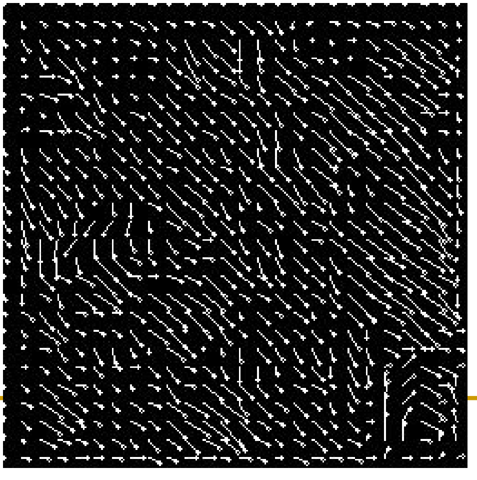


Optical flow result

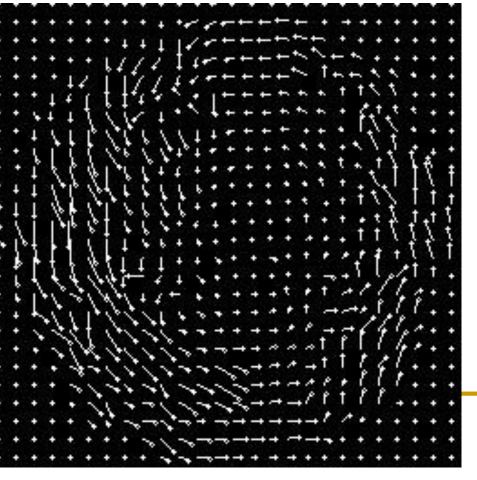


Optical Flow

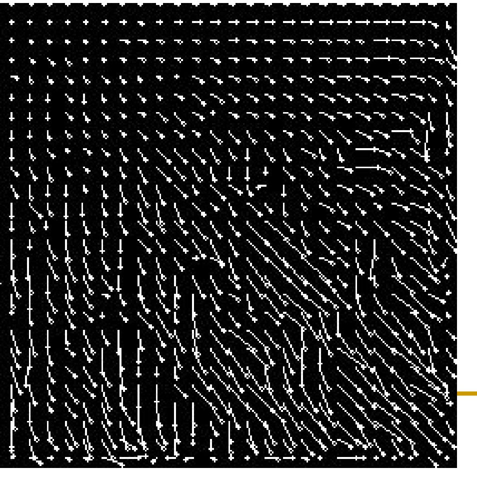
Translation



Rotation



Scaling



Motion in OpenCV

- Optical flow
  - cvCalcOpticalFlowLK
  - cvCalcOpticalFlowPyrLK
  - cvCalcOpticalFlowHS
  - cvCalcOpticalFlowBM
- Tracking
  - cvMeanShift
  - cvCamShift
- Motion templates
- Kalman Filter
- Condensation Algorithm (Particle filter)