

# Motion and Optical Flow

EECS 598-08 Fall 2014

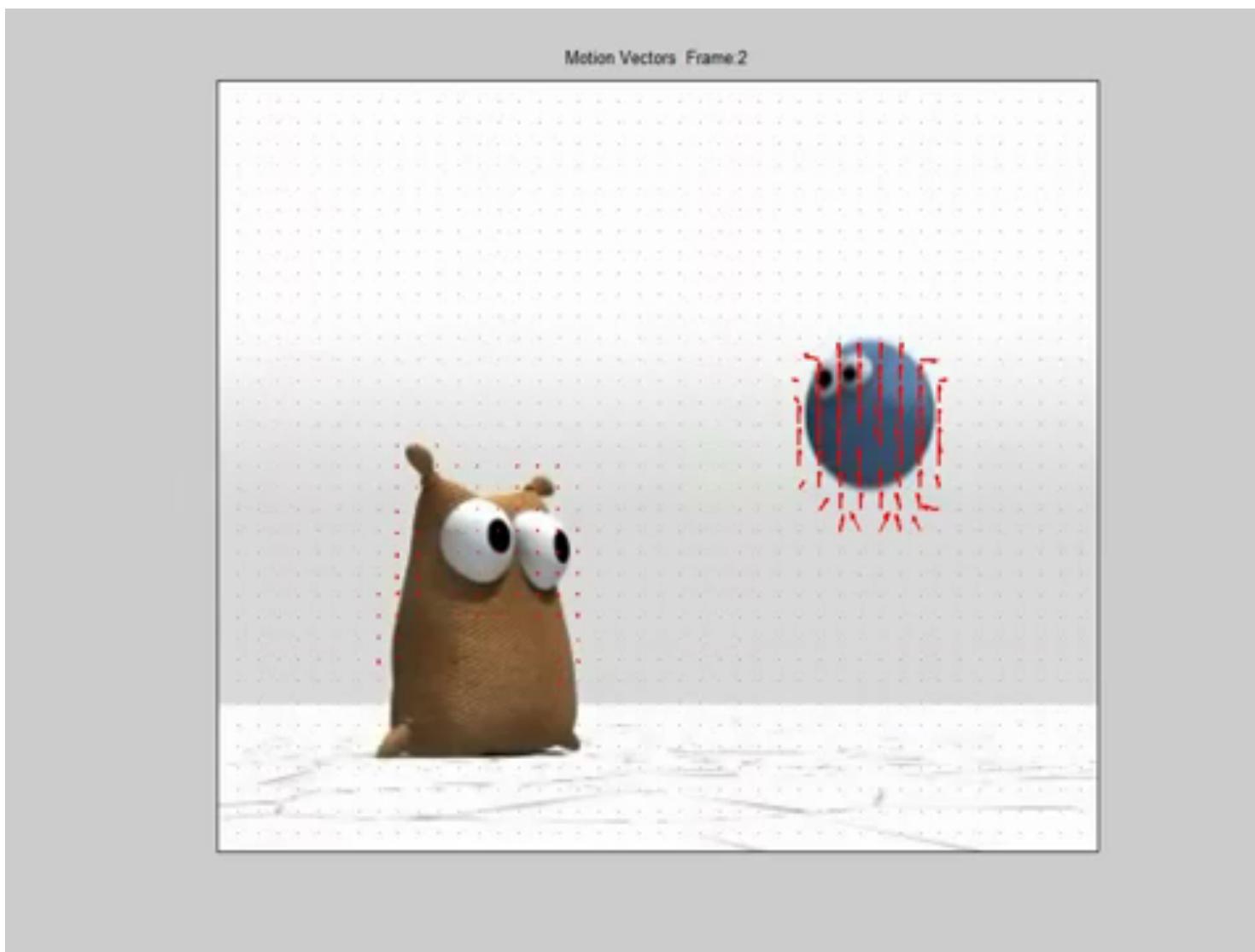
Foundations of Computer Vision

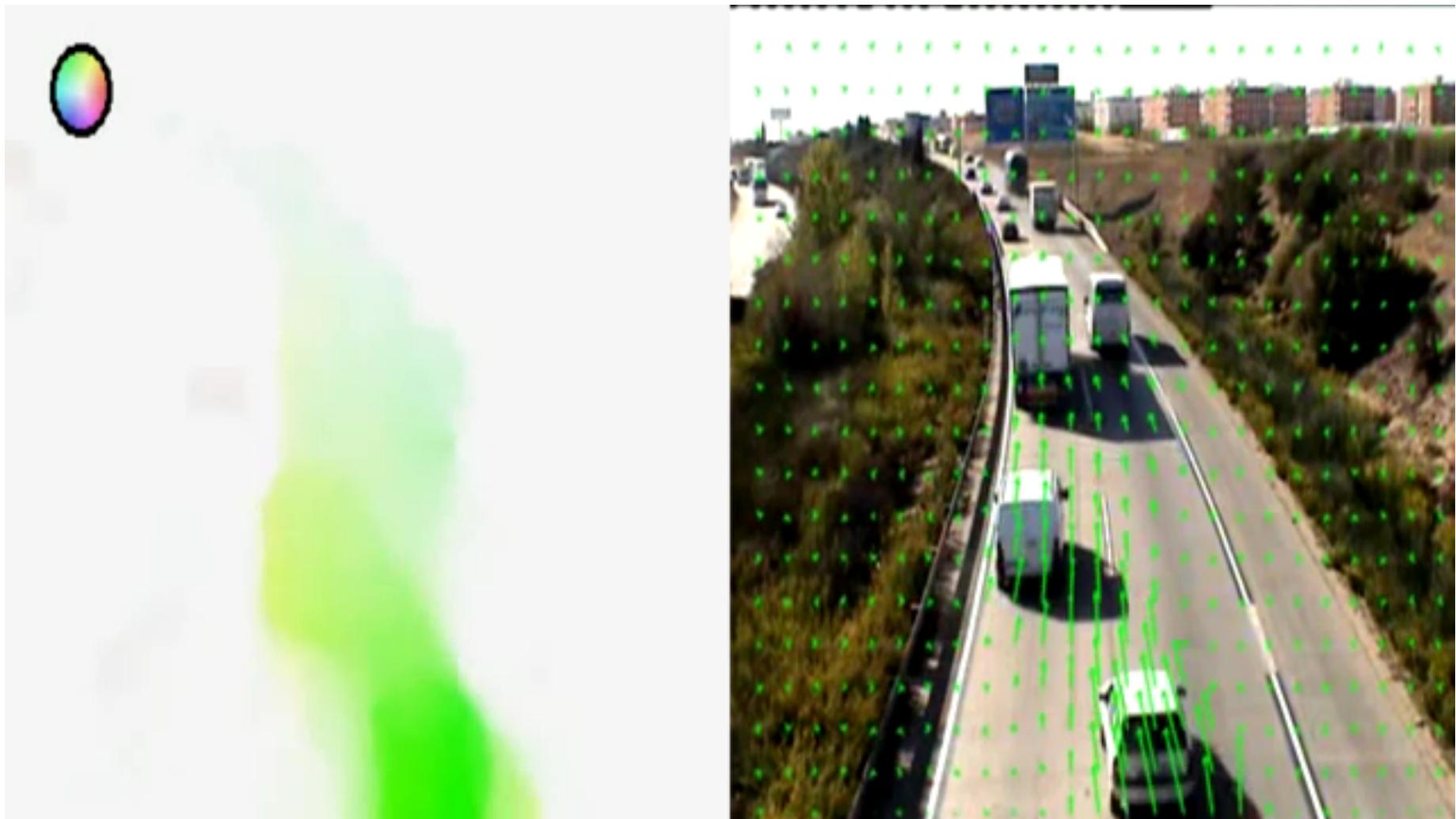
Instructor: Jason Corso (jjcorso)

[web.eecs.umich.edu/~jjcorso/t/598F14](http://web.eecs.umich.edu/~jjcorso/t/598F14)

**Readings:** FP 10.6; SZ 8; TV 8

**Date:** 10/15/14





# Plan

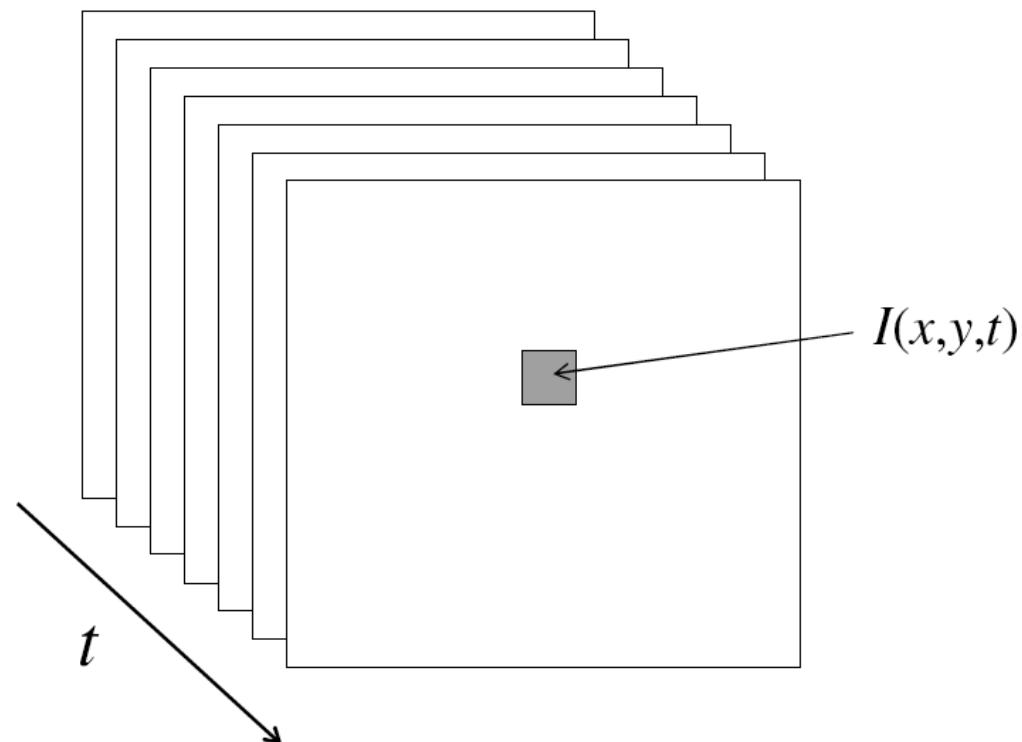
- Motion Field
- Patch-based / Direct Motion Estimation
- (Next: Feature Tracking)
- (Next: Layered Motion Models)



- External Resource:
  - Mubarak Shah's lecture on optical flow
    - <http://www.youtube.com/watch?v=5VyLAH8BhF8>

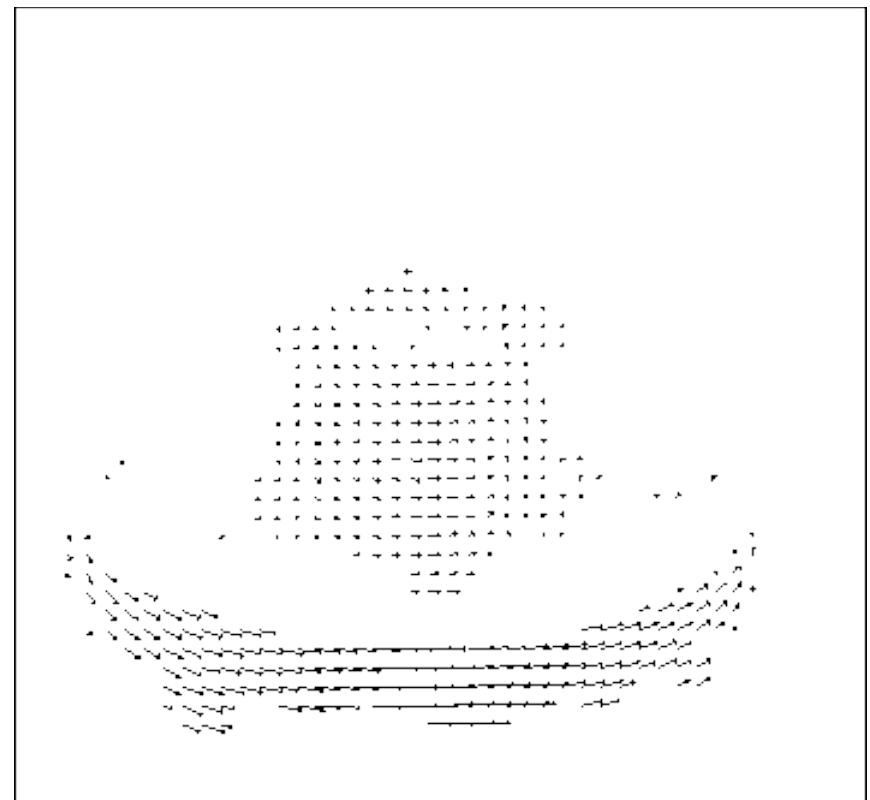
# Video

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



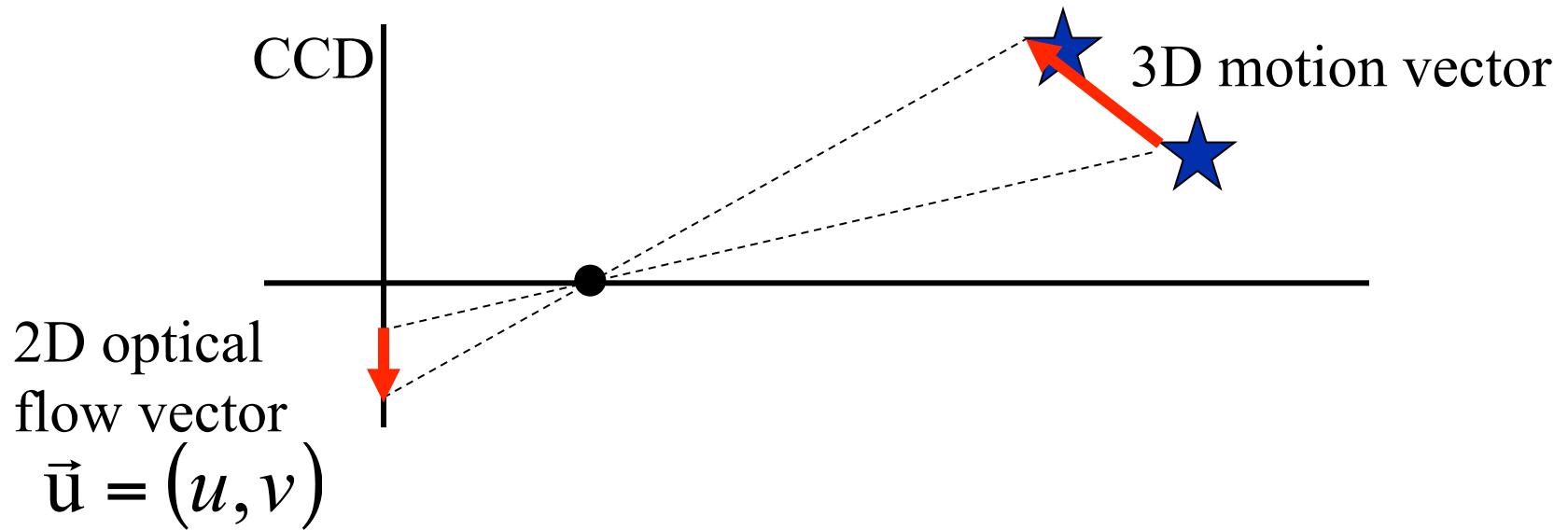
# Motion field

- The motion field is the projection of the 3D scene motion into the image



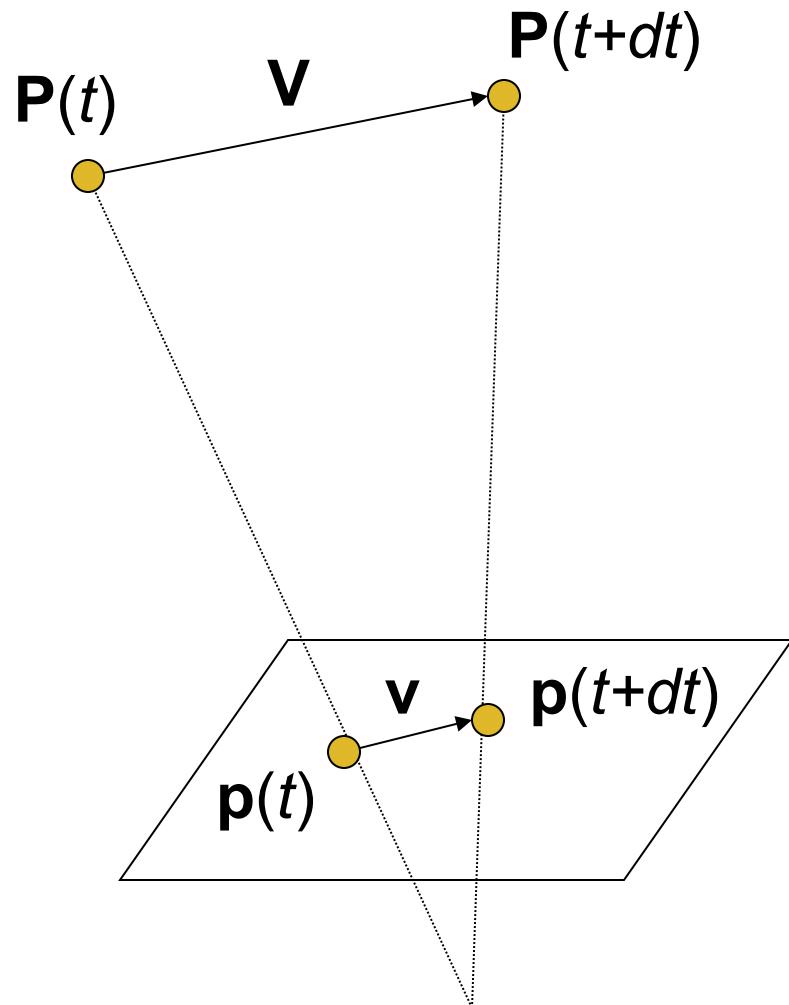
# Motion Field & Optical Flow Field

- Motion Field = Real world 3D motion
- Optical Flow Field = Projection of the motion field onto the 2d image



# Motion field and parallax

- $\mathbf{P}(t)$  is a moving 3D point
- Velocity of scene point:  $\mathbf{V} = \frac{d\mathbf{P}}{dt}$
- $\mathbf{p}(t) = (x(t), y(t))$  is the projection of  $\mathbf{P}$  in the image
- Apparent velocity  $\mathbf{v}$  in the image: given by components  $v_x = dx/dt$  and  $v_y = dy/dt$
- These components are known as the *motion field* of the image



# Motion estimation techniques

- Feature-based methods
  - Extract visual features (corners, textured areas) and track them over multiple frames
  - Sparse motion fields, but more robust tracking
  - Suitable when image motion is large (10s of pixels)
- Direct methods
  - Directly recover image motion at each pixel from spatio-temporal image brightness variations
  - Dense motion fields, but sensitive to appearance variations
  - Suitable for video and when image motion is small

# Optical flow

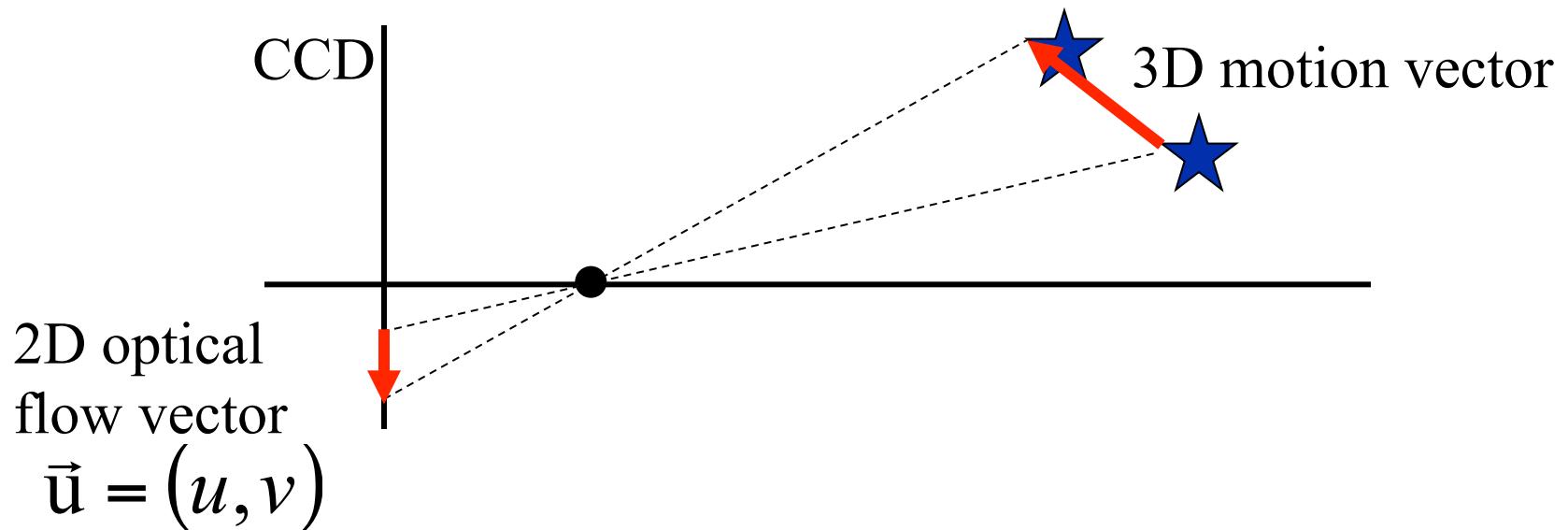
- Definition: optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion



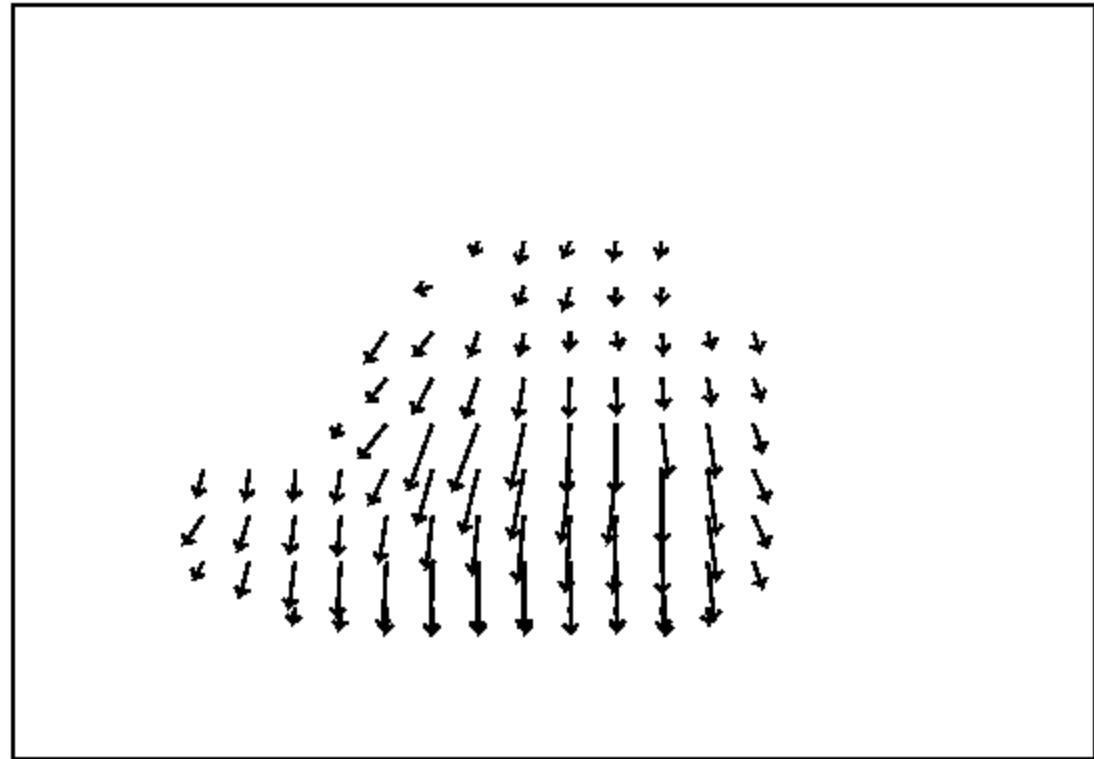
Where did each pixel in image 1 go to in image 2

# Motion Field & Optical Flow Field

- Motion Field = Real world 3D motion
- Optical Flow Field = Projection of the motion field onto the 2d image



# Optical Flow

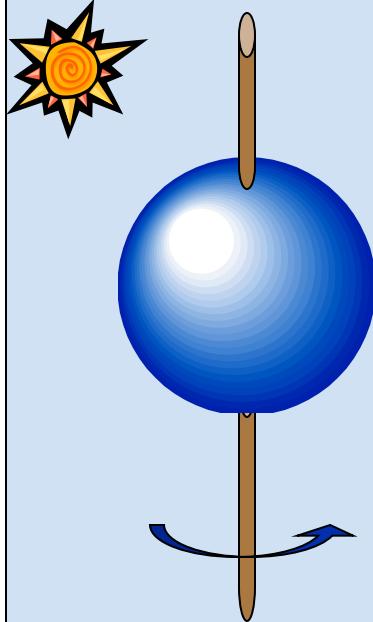


Pierre Kornprobst's Demo

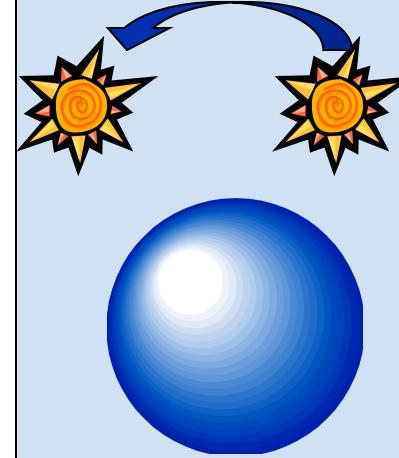
# When does it break?



The screen is  
stationary yet  
displays  
motion



Homogeneous  
objects generate  
zero optical flow.

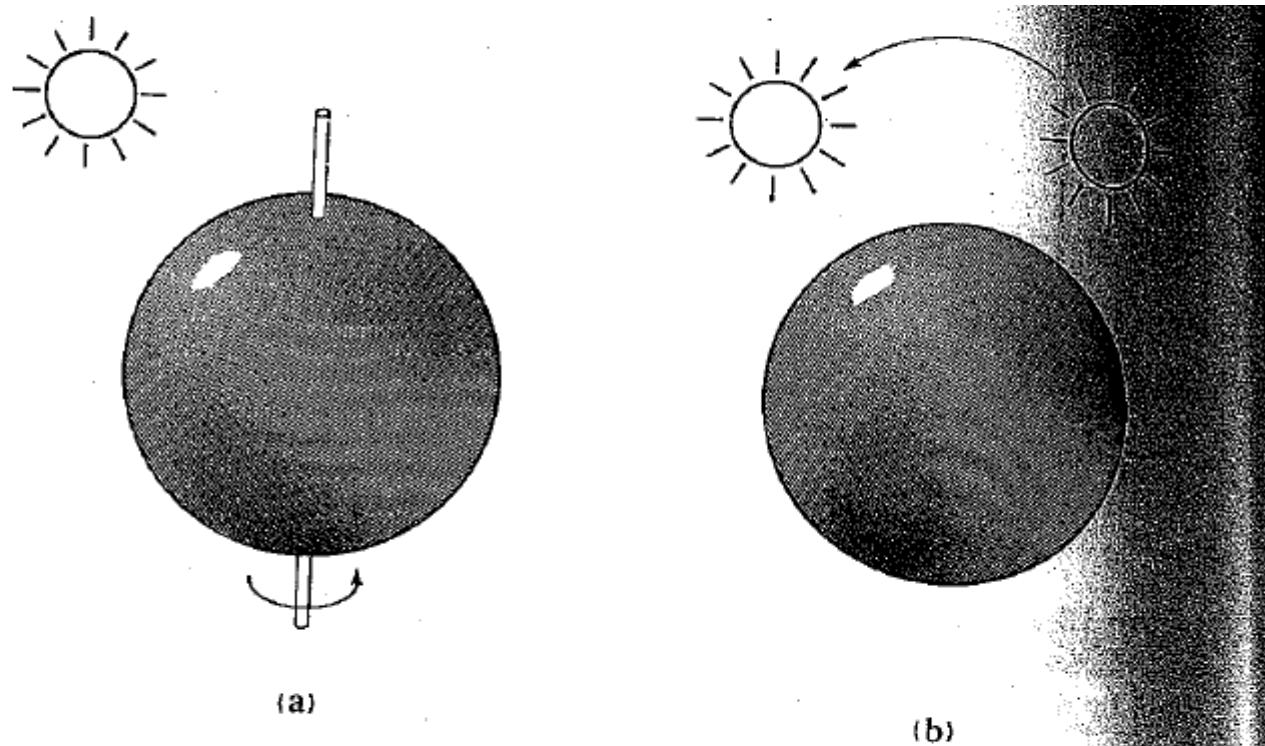


Fixed sphere.  
Changing light  
source.



Non-rigid texture  
motion

# Apparent motion $\approx$ motion field



**Figure 12-2.** The optical flow is not always equal to the motion field. In (a) a smooth sphere is rotating under constant illumination—the image does not change, yet the motion field is nonzero. In (b) a fixed sphere is illuminated by a moving source—the shading in the image changes, yet the motion field is zero.

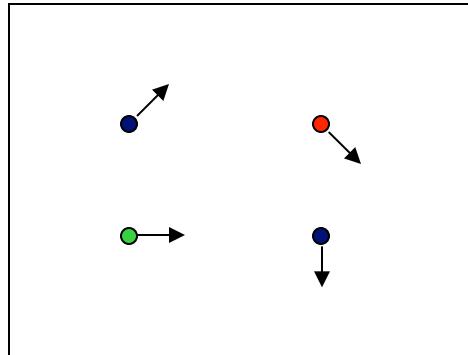
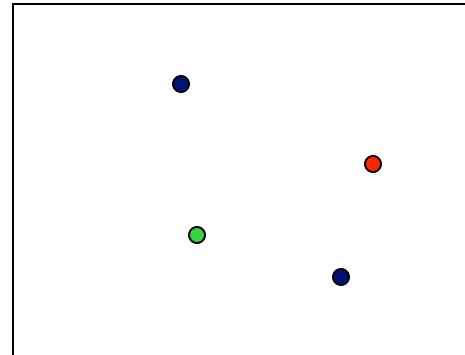
Figure from Horn book

# The Optical Flow Field

Still, in many cases it does work....

- Goal:  
Find for each pixel a velocity vector  $\vec{u} = (u, v)$   
which says:
  - How quickly is the pixel moving across the image
  - In which direction it is moving

# Estimating optical flow

 $I(x,y,t-1)$  $I(x,y,t)$ 

- Given two subsequent frames, estimate the apparent motion field between them.
- Key assumptions
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

# Brightness constancy

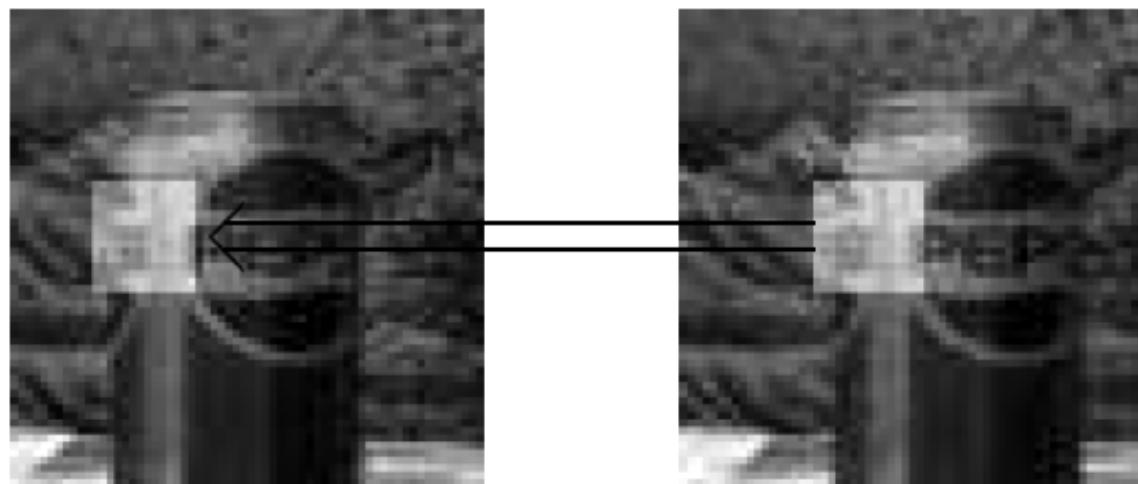
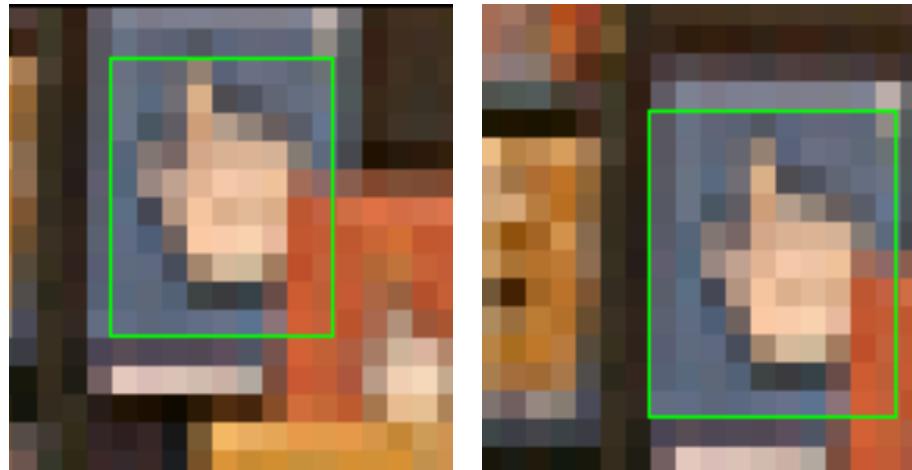
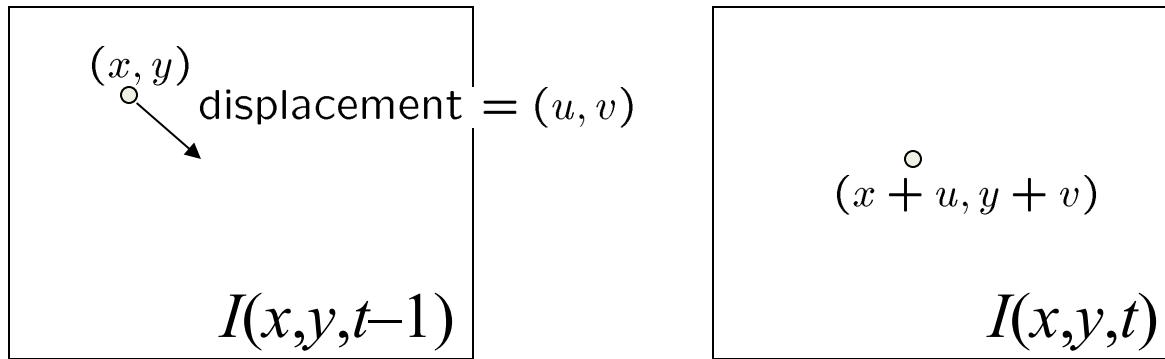


Figure 1.5: Data conservation assumption. The highlighted region in the right image looks roughly the same as the region in the left image, despite the fact that it has moved.

Figure by Michael Black

# The brightness constancy constraint



- Brightness Constancy Equation:

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

Can be written as:

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

$$I(x, y, t - 1) \approx I(x, y, t) + I_x \cdot u(x, y) + I_y \cdot v(x, y)$$

So,

$$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

# The brightness constancy constraint

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

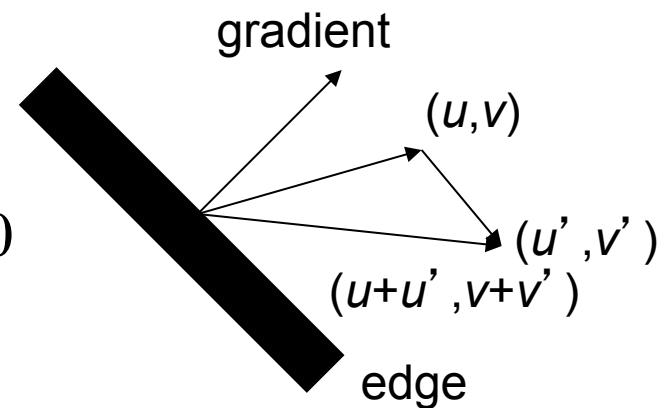
- How many equations and unknowns per pixel?
  - One equation, two unknowns
- Intuitively, what does this constraint mean?
$$\nabla I \cdot (u, v) + I_t = 0$$
- The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is unknown

# The brightness constancy constraint

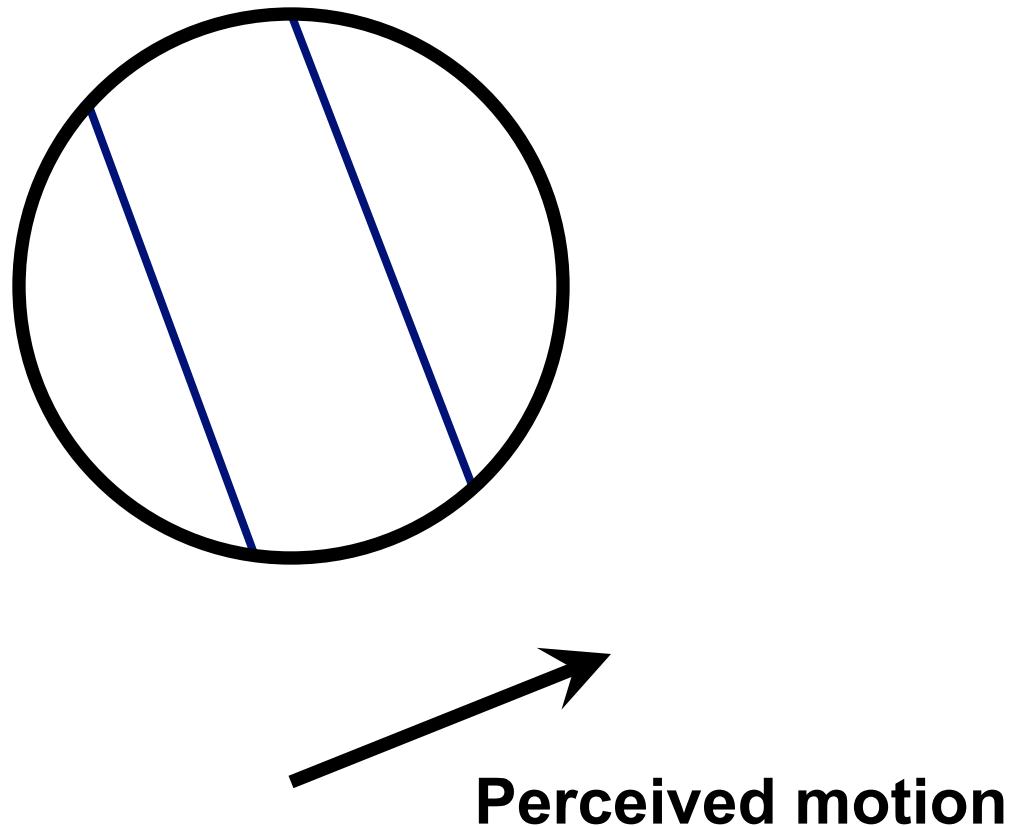
$$I_x \cdot u + I_y \cdot v + I_t = 0$$

- How many equations and unknowns per pixel?
    - One equation, two unknowns
  - Intuitively, what does this constraint mean?
- $$\nabla I \cdot (u, v) + I_t = 0$$
- The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is unknown

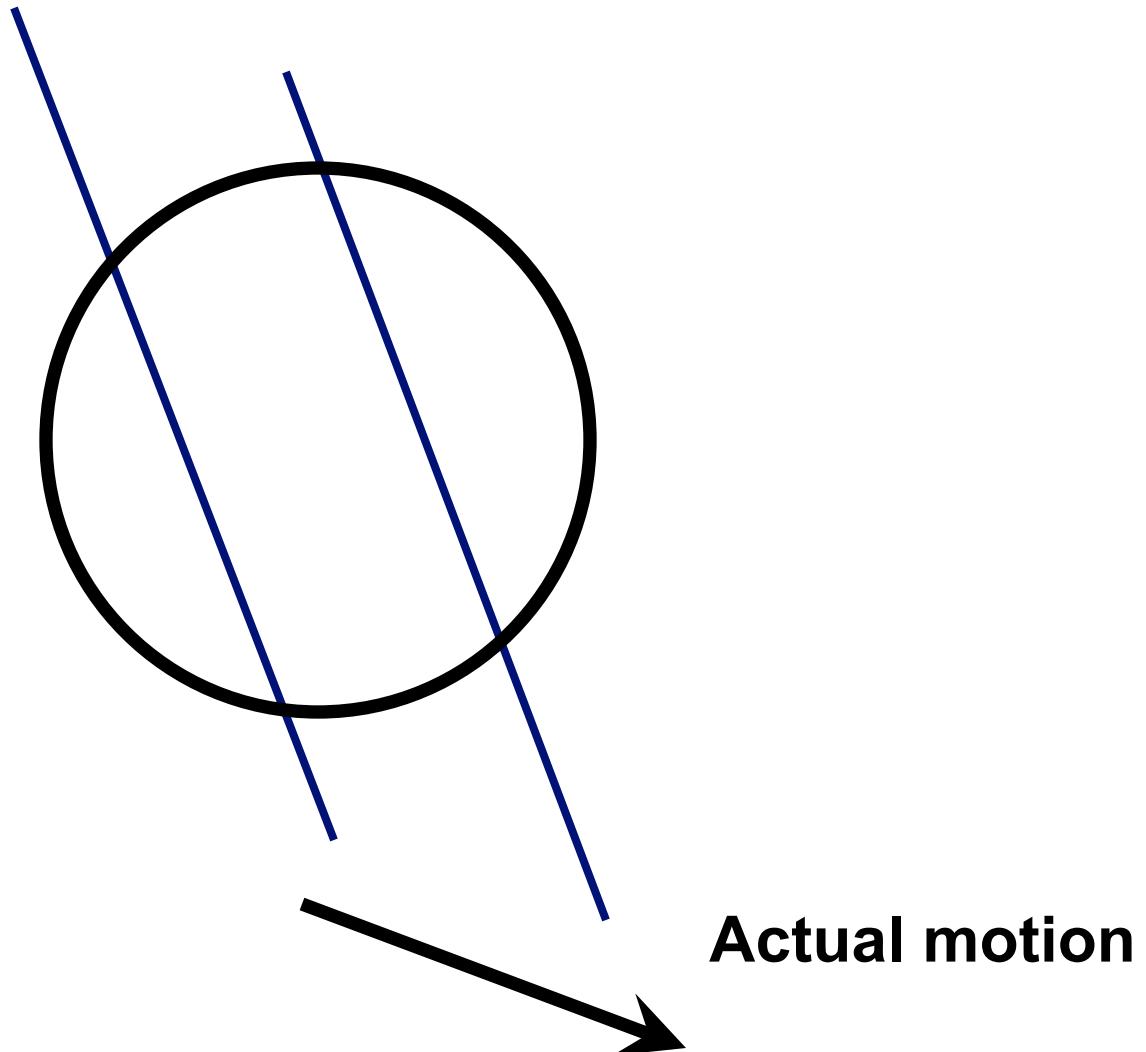
If  $(u, v)$  satisfies the equation,  
so does  $(u+u', v+v')$  if  $\nabla I \cdot (u', v') = 0$



# The aperture problem



# The aperture problem



# The barber pole illusion



[http://en.wikipedia.org/wiki/Barberpole\\_illusion](http://en.wikipedia.org/wiki/Barberpole_illusion)

# The barber pole illusion



[http://en.wikipedia.org/wiki/Barberpole\\_illusion](http://en.wikipedia.org/wiki/Barberpole_illusion)

# The barber pole illusion



[http://en.wikipedia.org/wiki/Barberpole\\_illusion](http://en.wikipedia.org/wiki/Barberpole_illusion)

## Lucas-Kanade: Solving the aperture problem (grayscale image)

- How to get more equations for a pixel?
- **Spatial coherence constraint:** pretend the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

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

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\begin{array}{ccc} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{array}$$

# Lucas-Kanade: Solving the aperture problem

Prob: we have more equations than unknowns

$$\begin{matrix} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{matrix} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

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

$$\begin{matrix} (A^T A) & d = A^T b \\ 2 \times 2 & 2 \times 1 & 2 \times 1 \end{matrix}$$

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

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

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

# Conditions for solvability

Look Familiar?

$$\left[ \begin{array}{cc} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{array} \right] \left[ \begin{array}{c} u \\ v \end{array} \right] = - \left[ \begin{array}{c} \sum I_x I_t \\ \sum I_y I_t \end{array} \right]$$

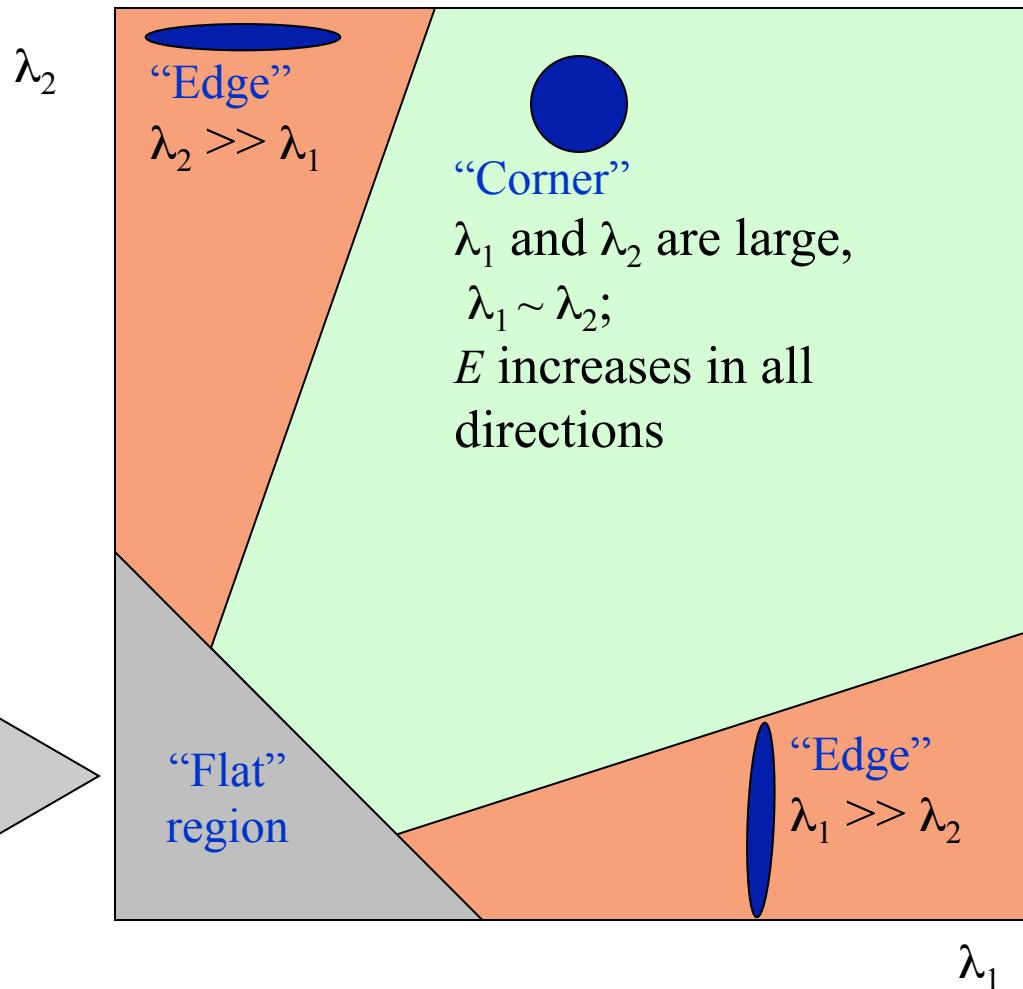
$$A^T A \quad A^T b$$

When is this solvable?

- $\mathbf{A}^T \mathbf{A}$  should be invertible
- $\mathbf{A}^T \mathbf{A}$  should not be too small
  - eigenvalues  $\lambda_1$  and  $\lambda_2$  of  $\mathbf{A}^T \mathbf{A}$  should not be too small
- $\mathbf{A}^T \mathbf{A}$  should be well-conditioned
  - $\lambda_1 / \lambda_2$  should not be too large ( $\lambda_1$  = larger eigenvalue)

# Conditions for solvability

Classification of image points using eigenvalues of  $M$ :



$\lambda_1$  and  $\lambda_2$  are small;  
 $E$  is almost constant  
in all directions



# Edge



- gradients very large or very small
- large  $\lambda_1$ , small  $\lambda_2$

# Low-texture region



- gradients have small magnitude
- small  $\lambda_1$ , small  $\lambda_2$

# High-texture region



- gradients are different, large magnitudes
- large  $\lambda_1$ , large  $\lambda_2$

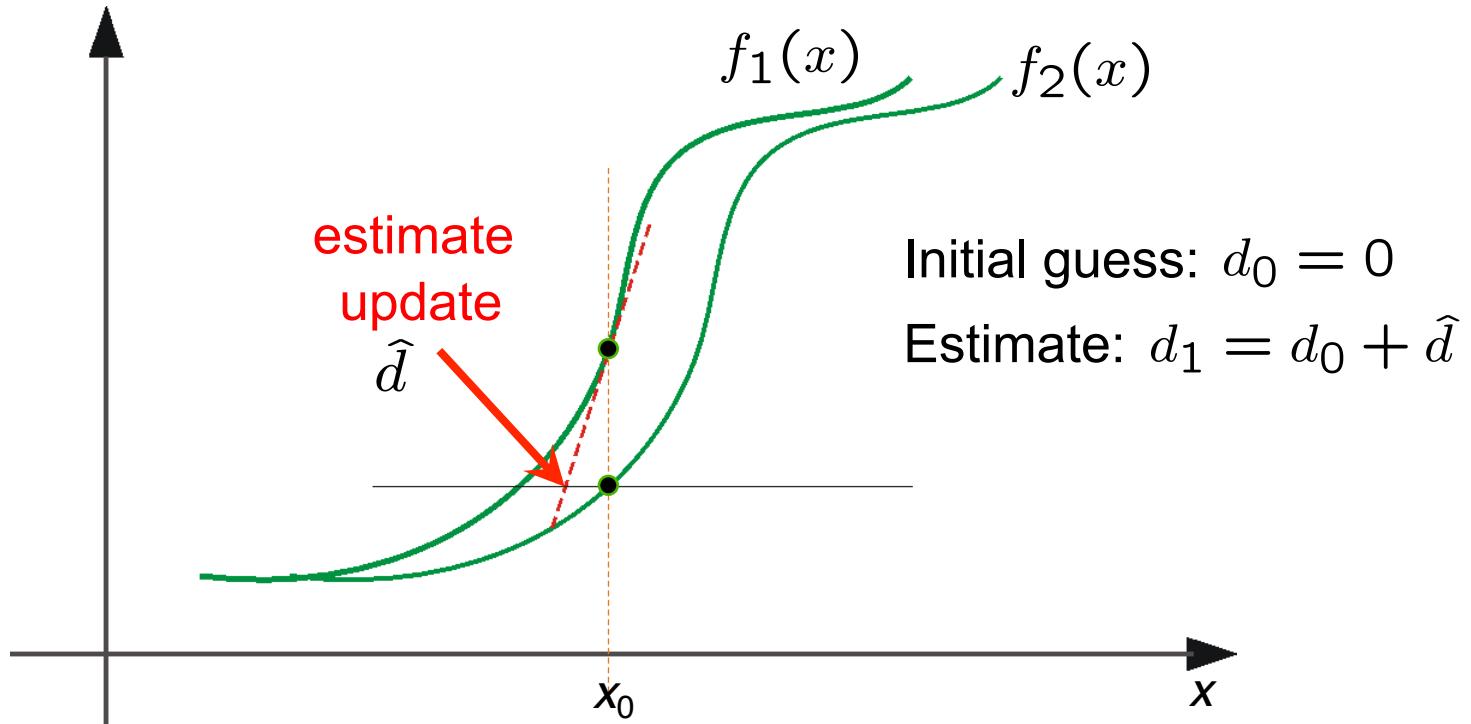
# Can we measure optical flow reliability?

- Can we measure “quality” of optical flow in regions from just a single image?
- High Quality / Good features to track:
  - Harris corners (guarantee small error sensitivity)
- Poor Quality / Bad features to track:
  - Image points when either  $\lambda_1$  or  $\lambda_2$  (or both) is small (i.e., edges or uniform textured regions)

# Iterative Refinement

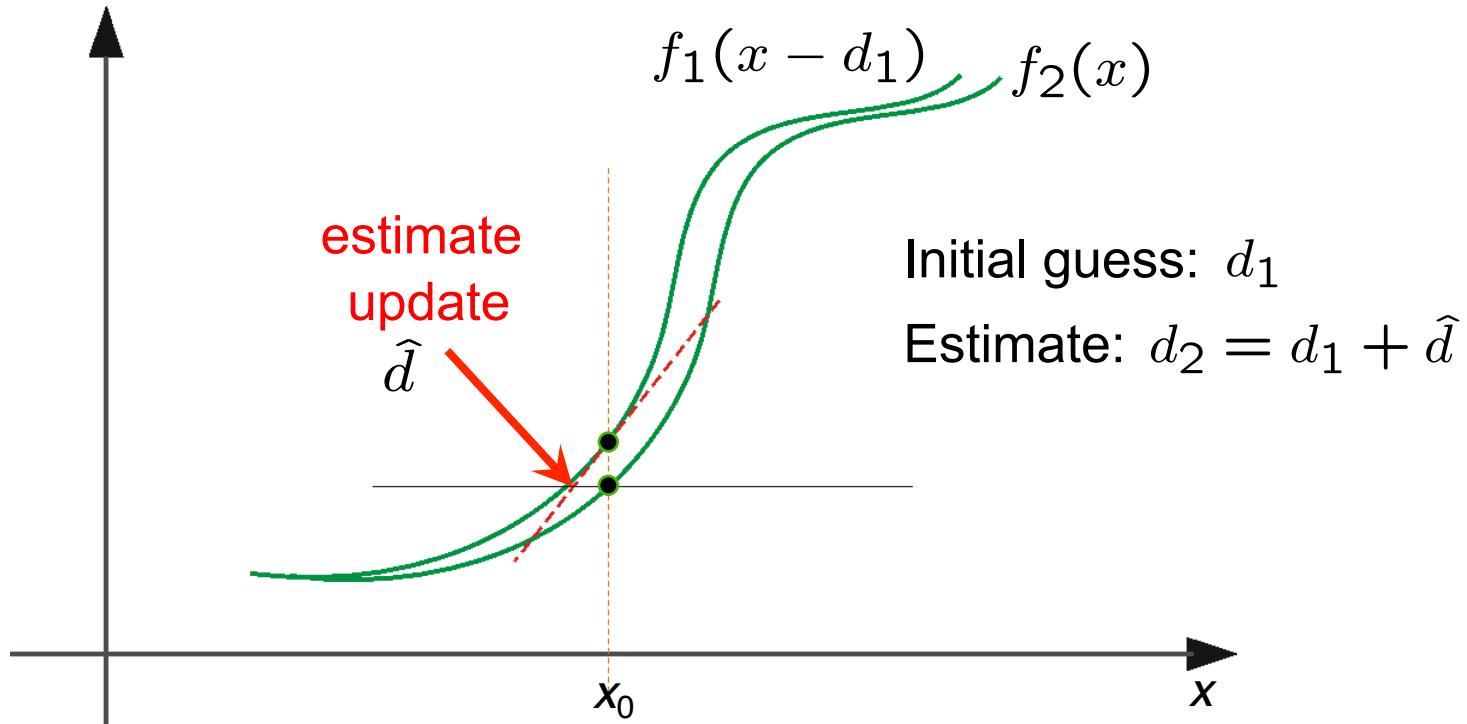
- Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation
- Warp one image toward the other using the estimated flow field  
*(easier said than done)*
- Refine estimate by repeating the process

# Optical Flow: Iterative Estimation

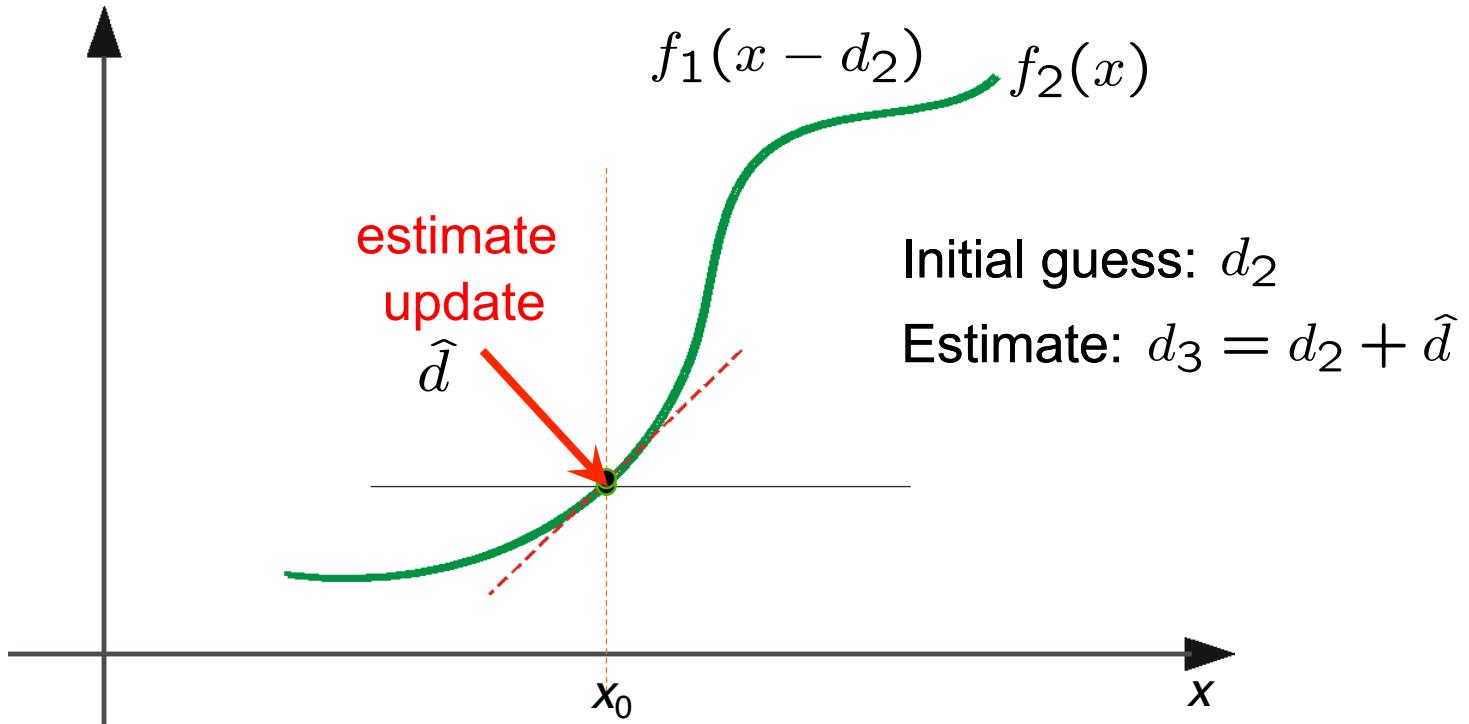


(using  $d$  for *displacement* here instead of  $u$ )

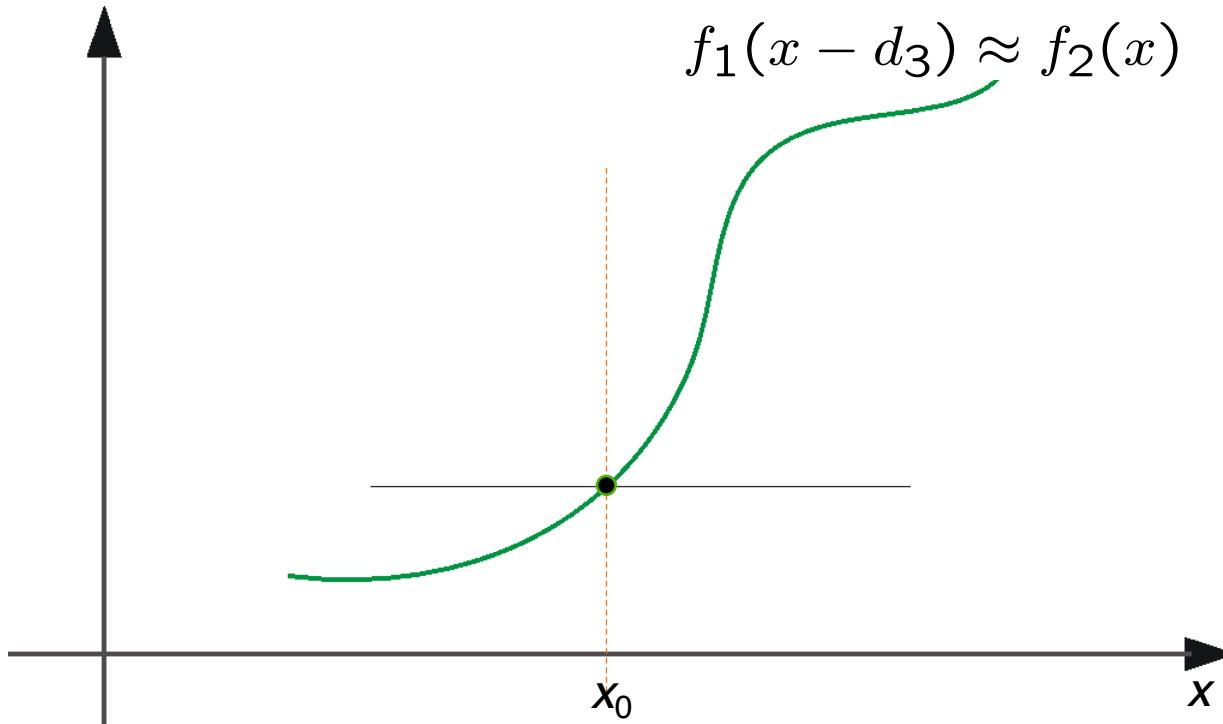
# Optical Flow: Iterative Estimation



# Optical Flow: Iterative Estimation



# Optical Flow: Iterative Estimation



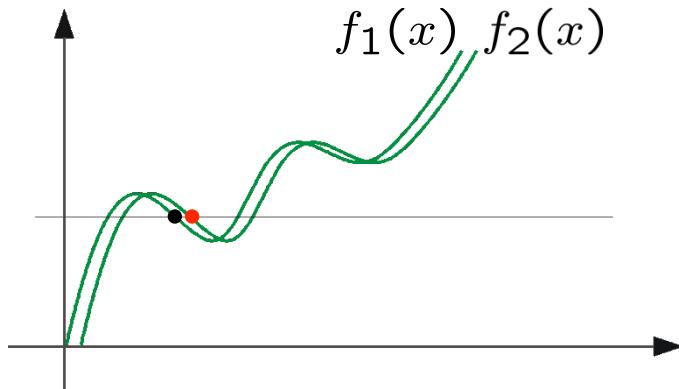
# Optical Flow: Iterative Estimation

- 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)

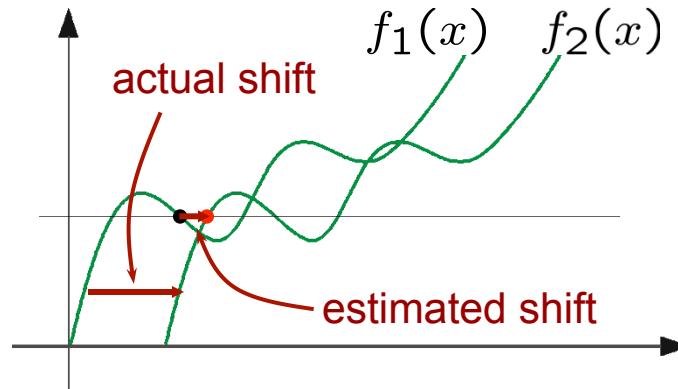
# Optical Flow: Aliasing

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)*

To overcome aliasing: coarse-to-fine estimation.

# Limits of the gradient method

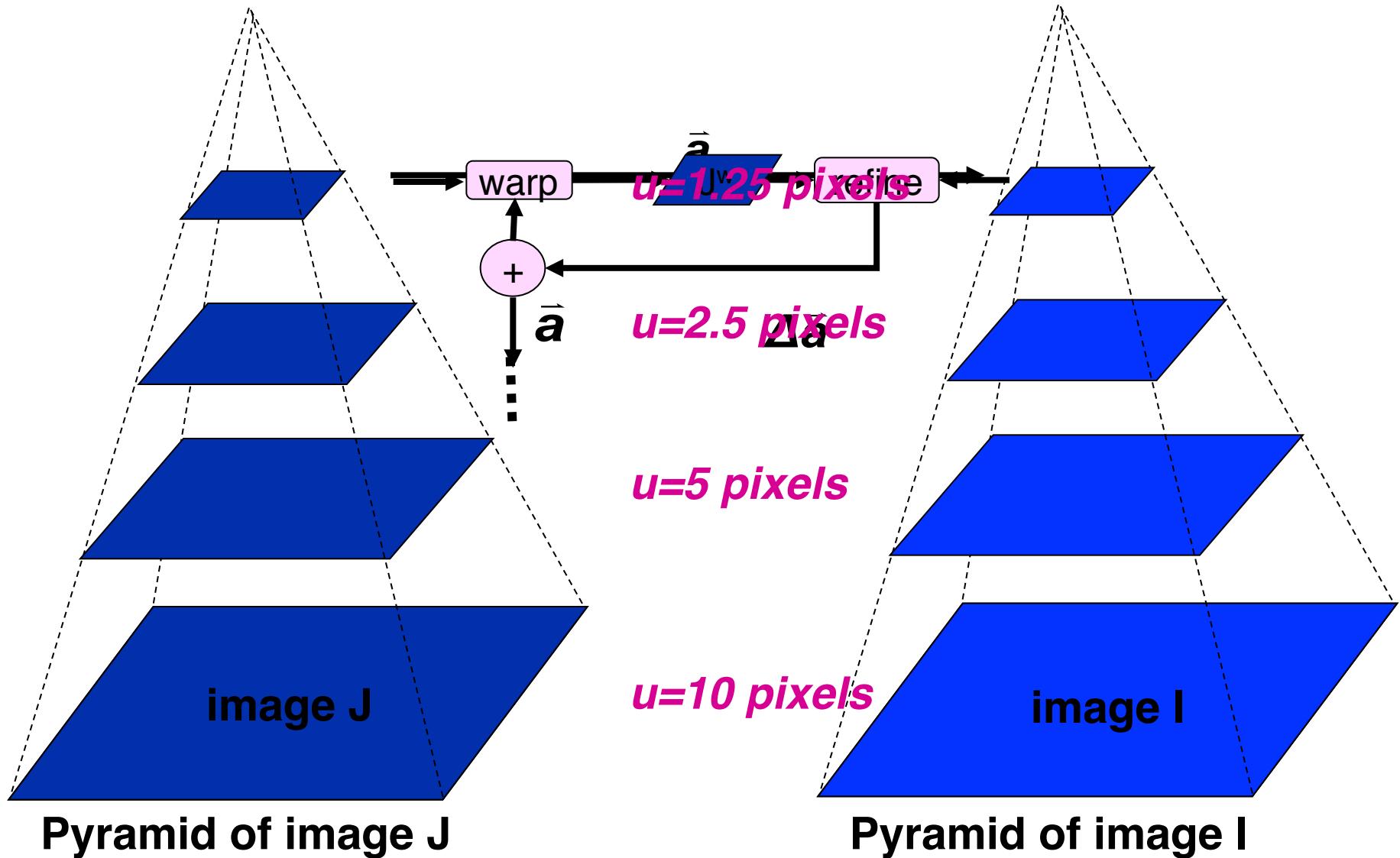
Fails when intensity structure in window is poor

Fails when the displacement is large (typical operating range is motion of 1 pixel)

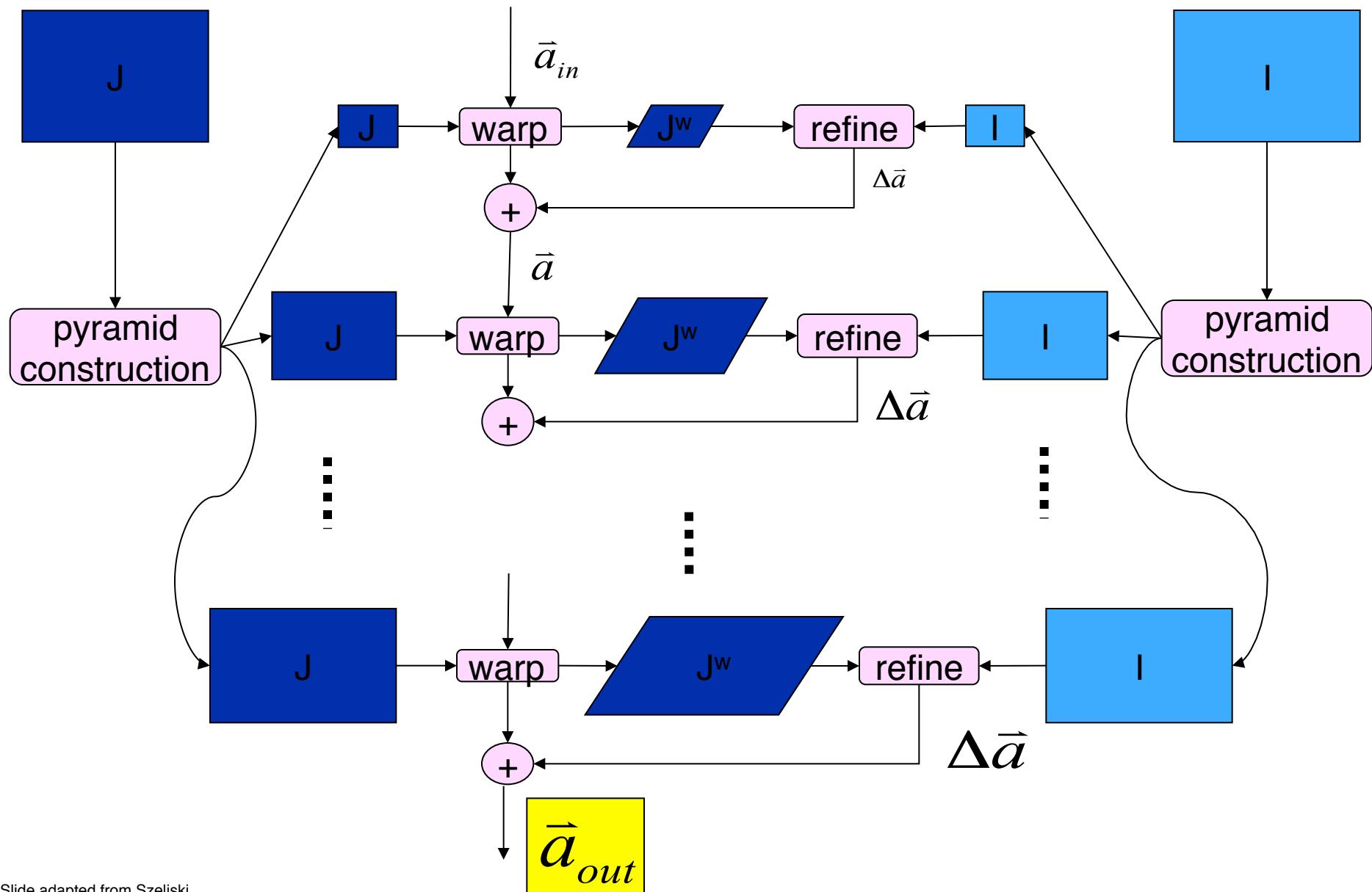
*Linearization of brightness is suitable only for small displacements*

- Also, brightness is not strictly constant in images  
*actually less problematic than it appears, since we can pre-filter images to make them look similar*

# Coarse-to-Fine Estimation

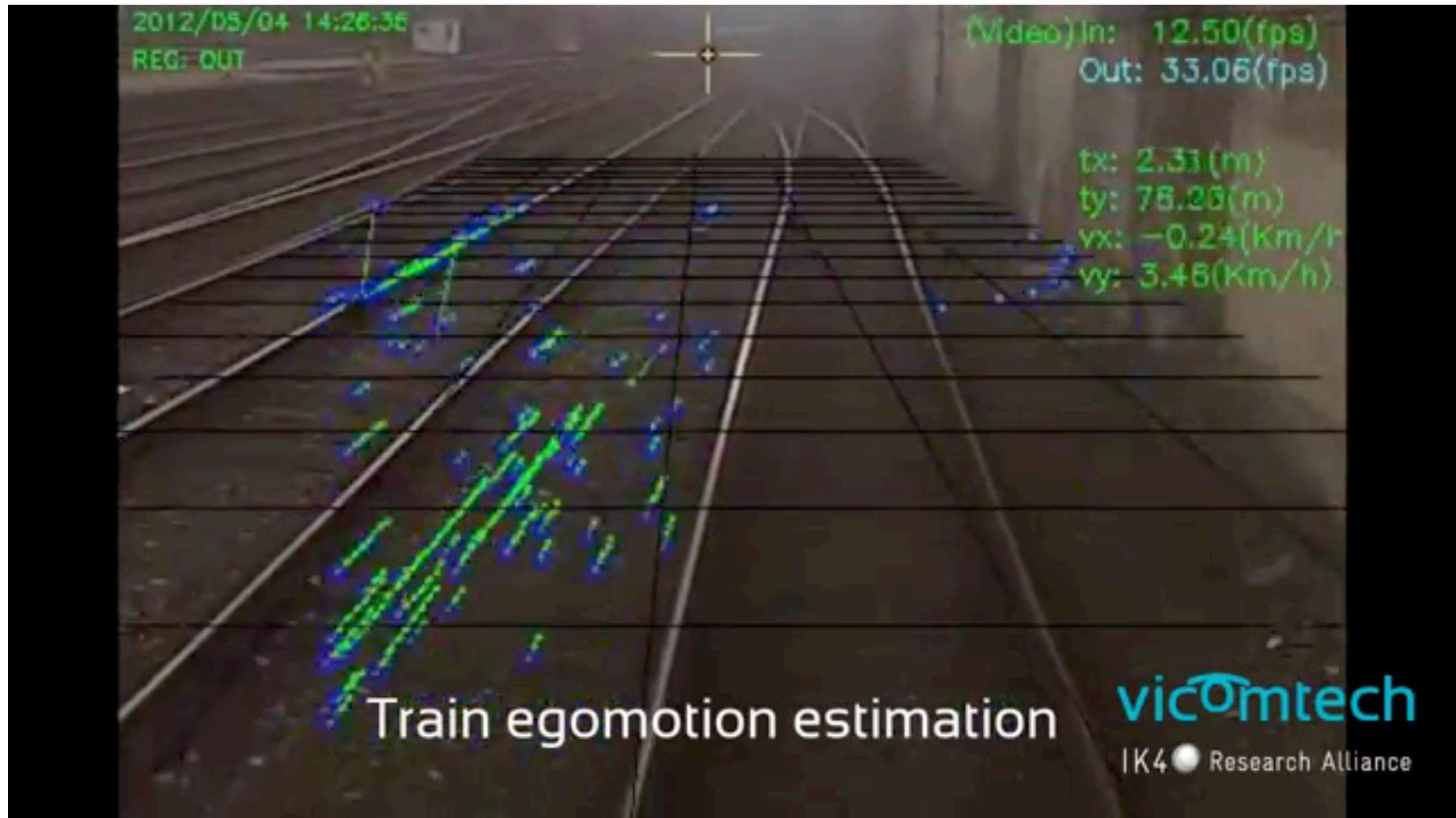


# Coarse-to-Fine Estimation



# Applications of Optical Flow

# Egomotion Estimation on the Railway



# Applications to Segmentation

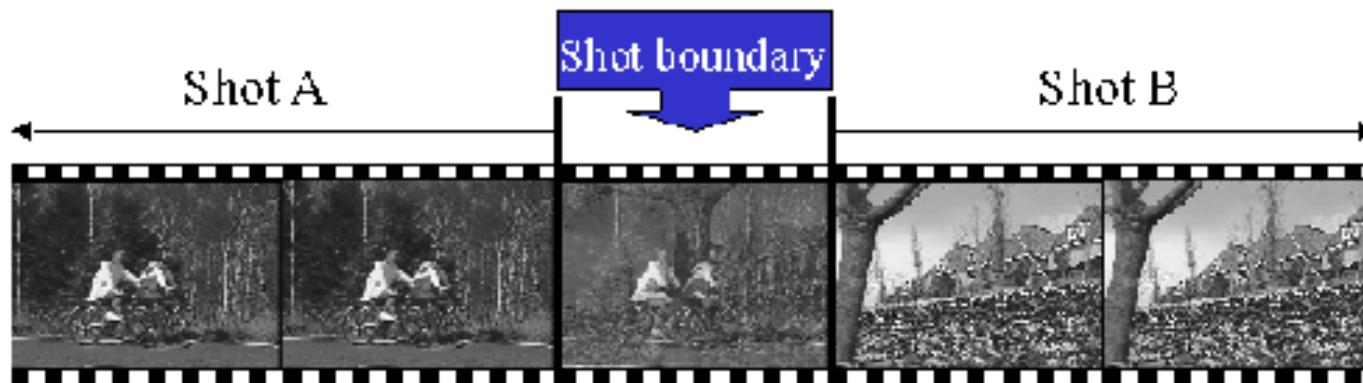
- Background subtraction
  - A static camera is observing a scene
  - Goal: separate the static *background* from the moving *foreground*

*How to come up  
with background  
frame estimate  
without access  
to “empty”  
scene?*



# Applications to Segmentation

- Background subtraction
- Shot boundary detection
  - Commercial video is usually composed of *shots* or sequences showing the same objects or scene
  - Goal: segment video into shots for summarization and browsing (each shot can be represented by a single keyframe in a user interface)
  - Difference from background subtraction: the camera is not necessarily stationary

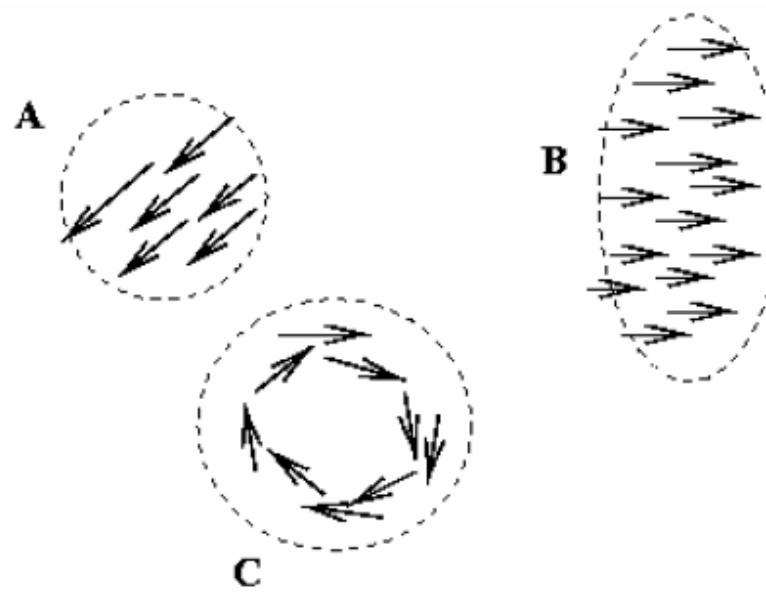


# Applications to Segmentation

- Background subtraction
- Shot boundary detection
  - For each frame
    - Compute the distance between the current frame and the previous one
      - Pixel-by-pixel differences
      - Differences of color histograms
      - Block comparison
    - If the distance is greater than some threshold, classify the frame as a shot boundary

# Applications To Segmentation

- Background subtraction
- Shot boundary detection
- Motion segmentation
  - Segment the video into multiple *coherently* moving objects



# Motion and perceptual organization

- Sometimes, motion is the only cue



Not grouped



Proximity



Similarity



Similarity



Common Fate



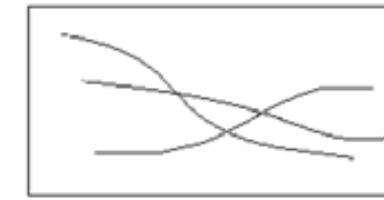
Common Region



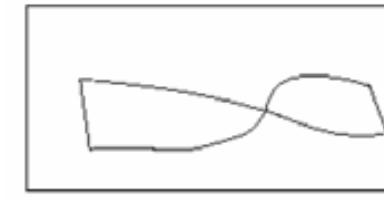
Parallelism



Symmetry



Continuity



Closure

# Motion and perceptual organization

- Sometimes, motion is foremost cue



# Motion and perceptual organization

- Even “impoverished” motion data can evoke a strong percept

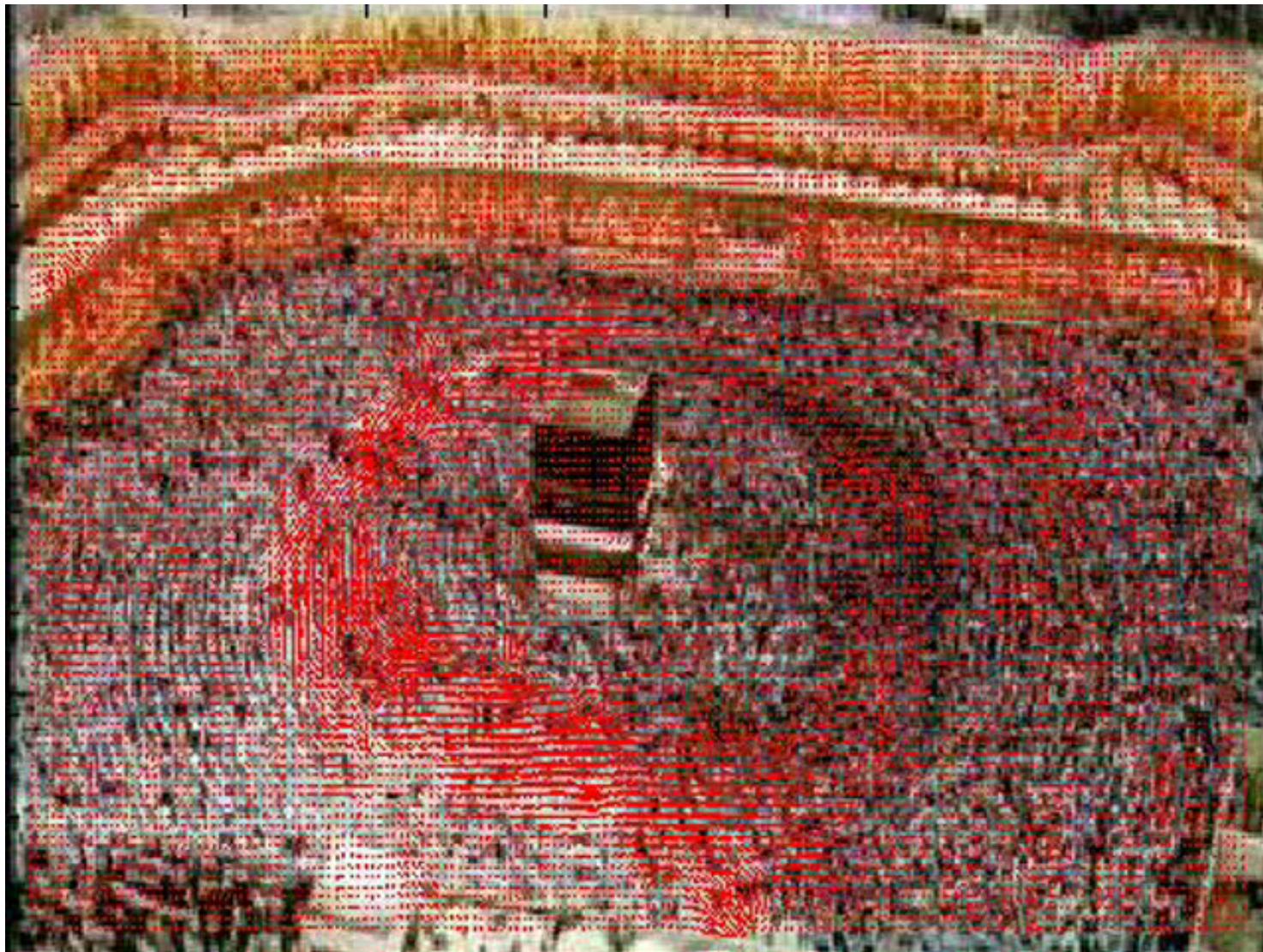


Sources: Maas 1971 with Johansson; downloaded from Youtube.

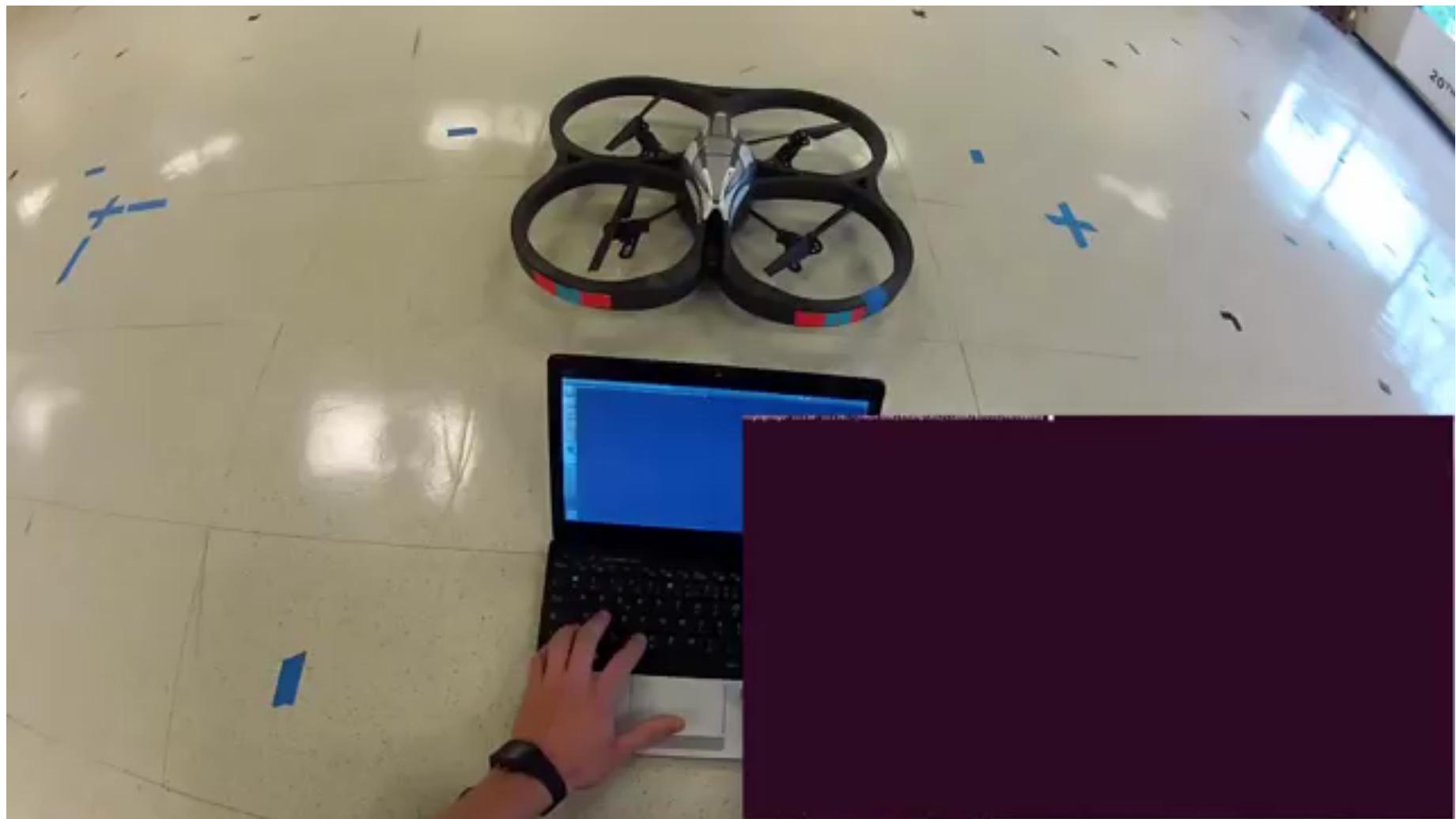
# Uses of motion

- Estimating 3D structure
- Segmenting objects based on motion cues
- Learning dynamical models
- Recognizing events and activities
- Improving video quality (motion stabilization)

# Crowd Analysis



# Aerial Vehicle Target Tracking



# A Camera Mouse

- Video interface: use feature tracking as mouse replacement



- User clicks on the feature to be tracked
  - Take the 15x15 pixel square of the feature
  - In the next image do a search to find the 15x15 region with the highest correlation
  - Move the mouse pointer accordingly
  - Repeat in the background every 1/30th of a second

# A Camera Mouse

- Specialized software for communication, games



James Gips and Margrit Betke  
<http://www.bc.edu/schools/csom/eagleeyes/>

# Optical Flow for Games!

Simple Optical Flow Application

# Motion Paint: an example use of optical flow

Use optical flow to track brush strokes, in order to animate them to follow underlying scene motion.



<http://www.fxguide.com/article333.html>

# Motion Paint: an example use of optical flow



# Next Lecture: Tracking

- Readings: FP 10.6; SZ 8; TV 8
  - Global, Parametric Motion Models.