Lecture 15

Motion

Why estimate motion?

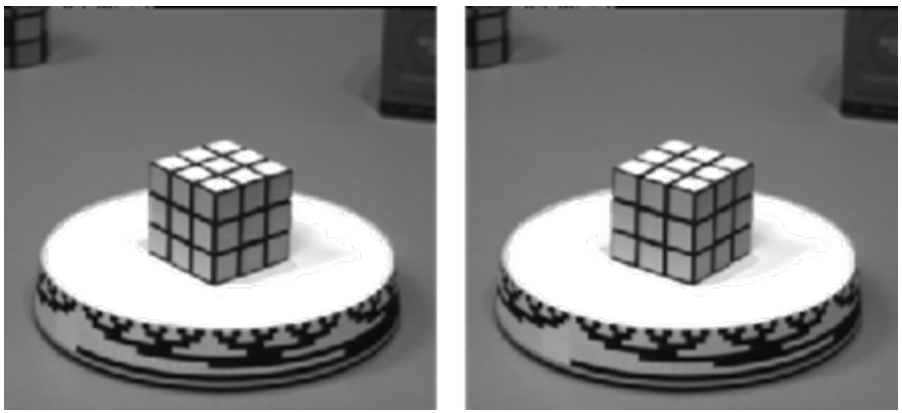
We live in a 4-D world (x,y,z,t)!

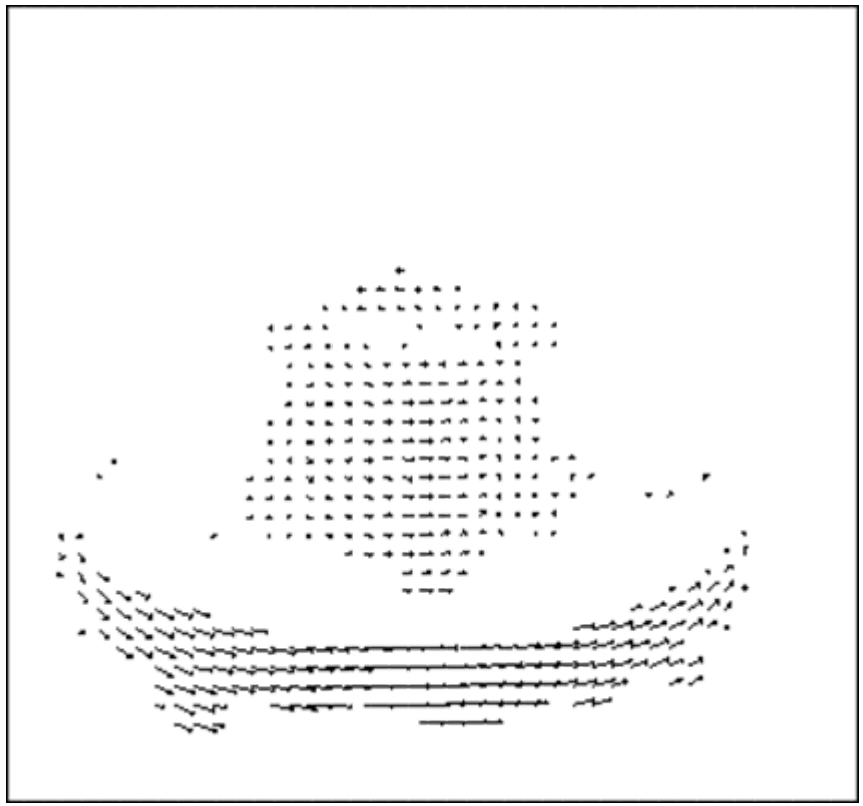
Wide applications

- Motion detection and object tracking (surveillance etc.)
- Correct for camera jitter (stabilization)
- Align images (panoramic mosaics)
- 3D shape reconstruction (shape from motion)
- Video compression (MPEG)
- Robotics (navigation etc.)
- Entertainment: Special Effects, Sportscasting, Video Games

Slides adapted from Steve Seitz, Linda Shapiro, and others

Fundamental Problem: Optical flow

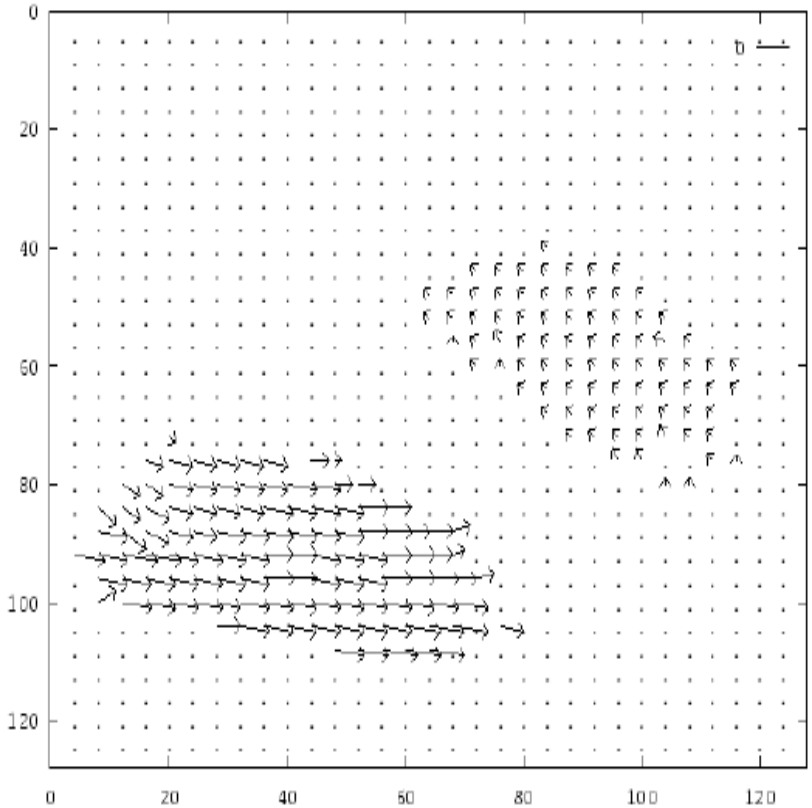




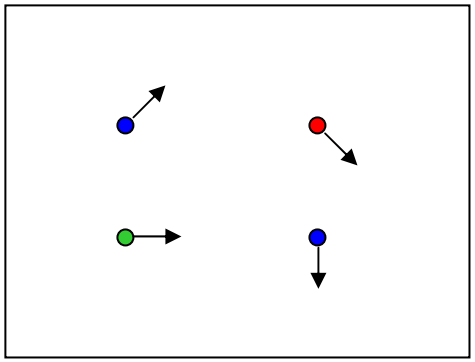
Estimate motion vectors at every pixel from image sequence

Another Example

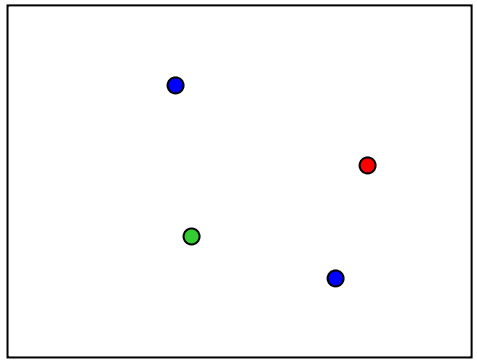
Hamburg Taxi Sequence



Problem definition: optical flow



$H(x, y)$



$I(x, y)$

How to estimate pixel motion from image H to image I?

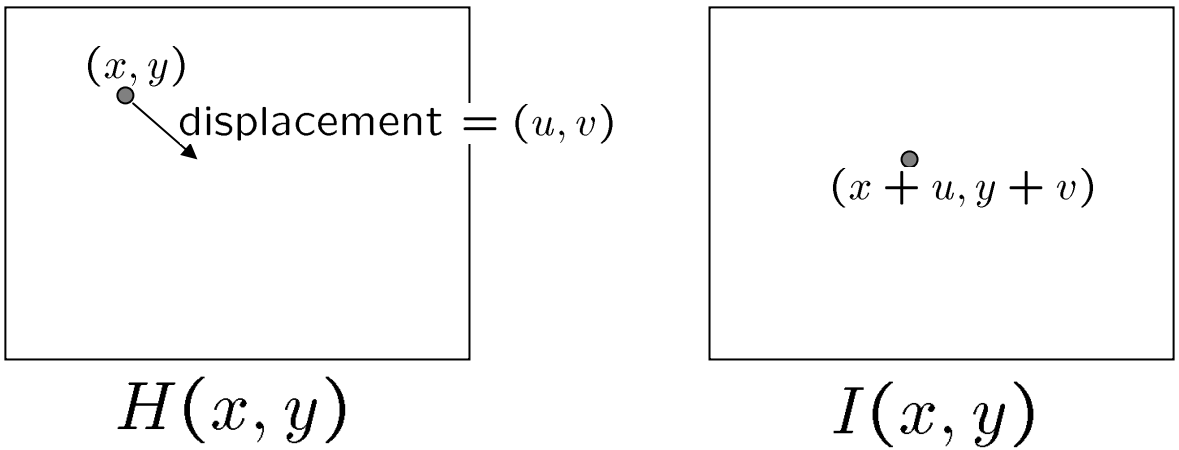
- Solve pixel correspondence problem
  - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions

- **color constancy:** a point in H looks the same in I
  - For grayscale images, this is **brightness constancy**
- **small motion**: points do not move very far

This is called the **optical flow** problem

Optical flow constraints (grayscale images)



Let's look at these constraints more closely

- brightness constancy:

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

- 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

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

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

x-component of gradient vector

Gradient $\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$

What is I_t? The time derivative of the image at (x,y)

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

$$0 = I_t + \nabla I \cdot [\frac{\partial x}{\partial t} \;\; \frac{\partial y}{\partial t}]$$

Optical flow equation

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

Q: how many unknowns and equations per pixel?

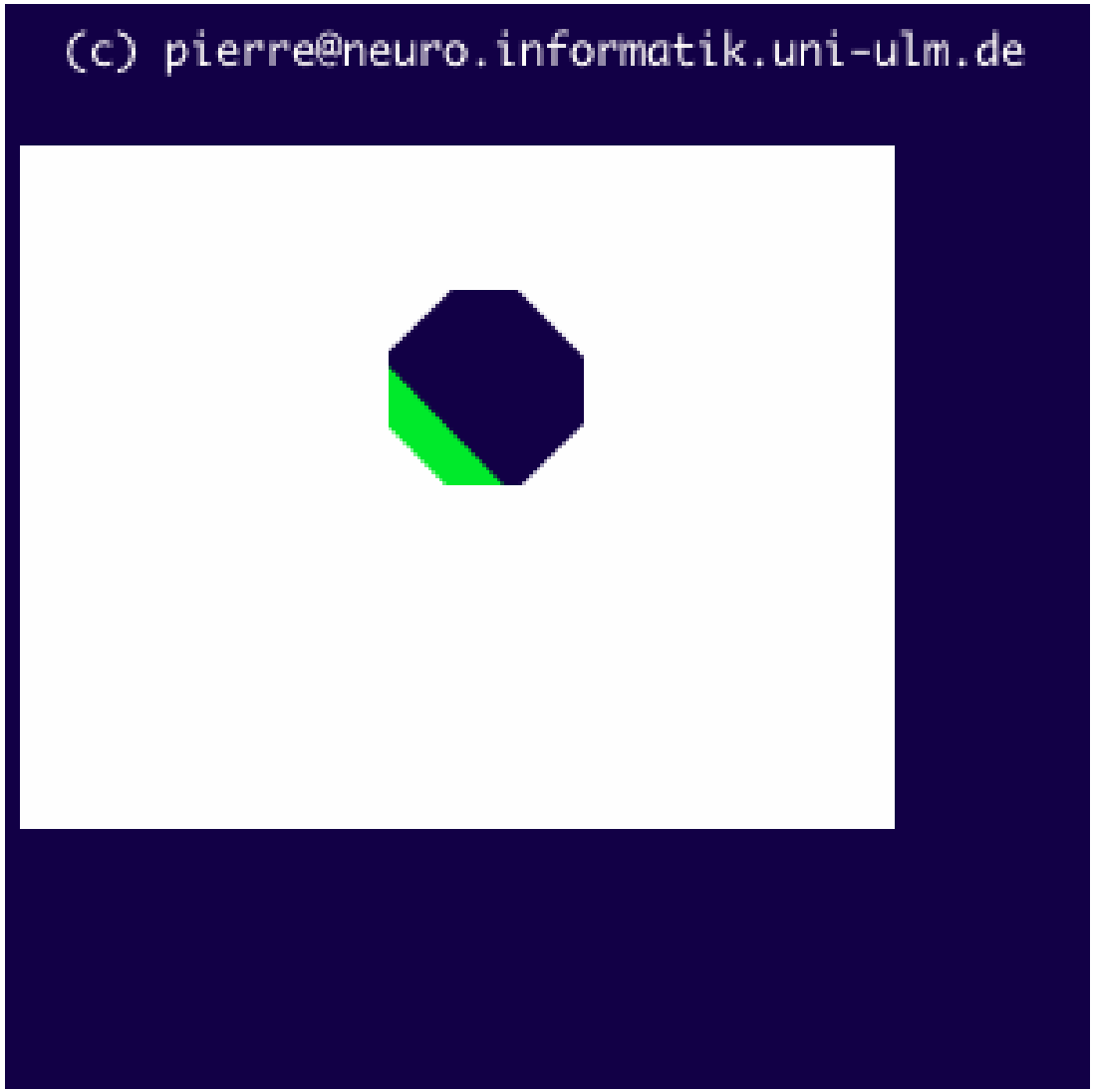
1 equation, but 2 unknowns (*u* and *v*)

Intuitively, what does this constraint mean?

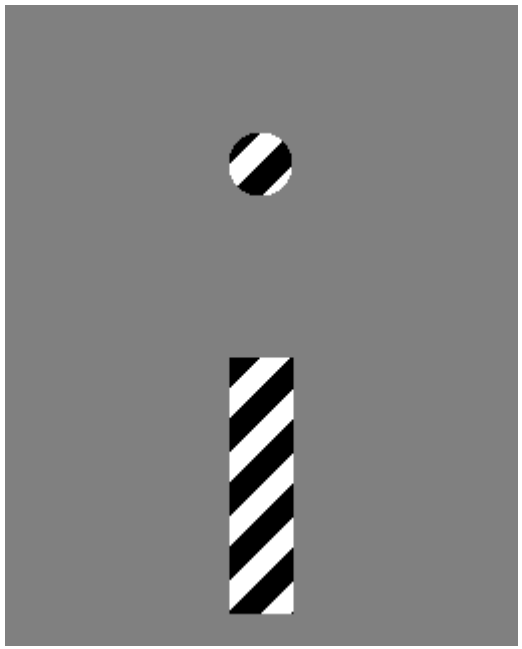
- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown

This leads to the Aperture Problem…

Aperture problem



Barber Pole Illusion



Solving the aperture problem

How to get more equations for a pixel?

- Basic idea: impose additional constraints

Example: Assume motion field is smooth locally

Lucas & Kanade: assume locally constant motion

- pretend the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel!

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

Many other methods exist. Here's an overview:

- Barron, J.L., Fleet, D.J., and Beauchemin, S, Performance of optical flow techniques, *International Journal of Computer Vision*, 12(1):43-77, 1994.

Solving the aperture problem

Lucas & Kanade: assume locally constant motion

- assume the pixel's neighbors have the same (u,v)
  - 5x5 window: gives us 25 equations per pixel!

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

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

$$A \qquad d \qquad b$$

Aside: What if we have RGB color images?

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

$$\begin{bmatrix} I_x(\mathbf{p_1})[0] & I_y(\mathbf{p_1})[0] \\ I_x(\mathbf{p_1})[1] & I_y(\mathbf{p_1})[1] \\ I_x(\mathbf{p_1})[2] & I_y(\mathbf{p_1})[2] \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}})[0] & I_y(\mathbf{p_{25}})[0] \\ I_x(\mathbf{p_{25}})[1] & I_y(\mathbf{p_{25}})[1] \\ I_x(\mathbf{p_{25}})[2] & I_y(\mathbf{p_{25}})[2] \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1})[0] \\ I_t(\mathbf{p_1})[1] \\ I_t(\mathbf{p_1})[2] \\ \vdots \\ I_t(\mathbf{p_{25}})[0] \\ I_t(\mathbf{p_{25}})[1] \\ I_t(\mathbf{p_{25}})[2] \end{bmatrix}$$

A 75x2 $\quad d$ 2x1 $\quad b$ 75x1

Back to Lucas-Kanade Method

Prob: we have more equations than unknowns

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

Solution: solve least squares problem

- minimum least squares solution given by:

$$d = (A^T A)^{-1} A^T b$$

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}$$

Does this look familiar?

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

Conditions for solvability

- Optimal d = (u, v) given by:

$$d = (A^T A)^{-1} A^T b$$

When is This Feasible?

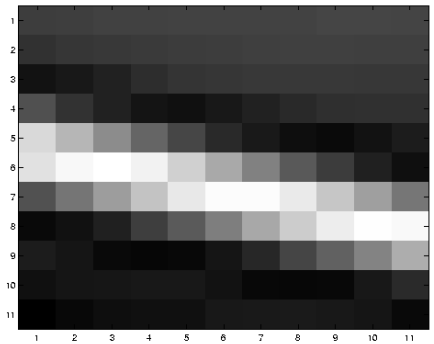
- $\mathbf{A^TA}$ should be invertible
- $\mathbf{A^TA}$ should not be too small (noise)
  - **eigenvalues** λ_1 and λ_2 of $\mathbf{A^TA}$ should **not be too small**
- $\mathbf{A^TA}$ should be well-conditioned
  - Ratio λ_1 / λ_2 should **not be too large** (λ_1 = larger eigenvalue)

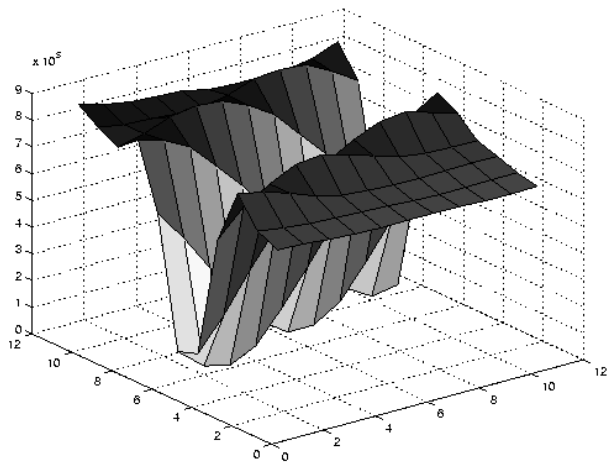
Déjà vu?
This is related to our old friend the Harris operator…

Example motion sequence

This is the famous “flower garden sequence” in computer vision

Edges may cause problems

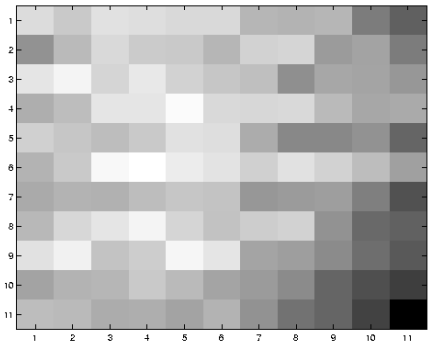


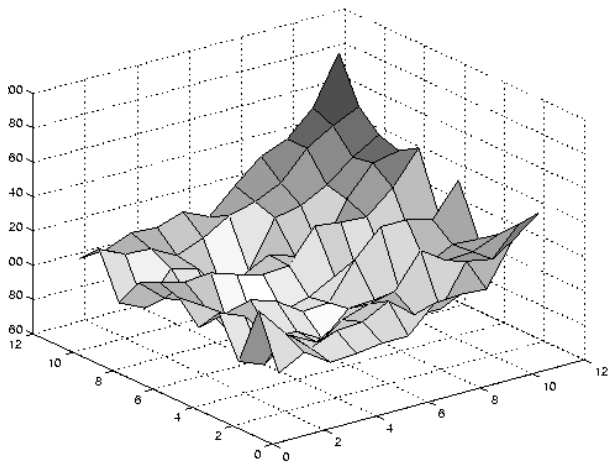


$$A^T A$$

- large gradients in one direction
- large λ_1, small λ_2
- $A^T A$ may not be well-conditioned

Low texture regions may perform badly

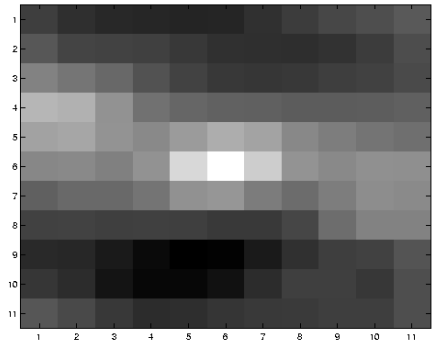


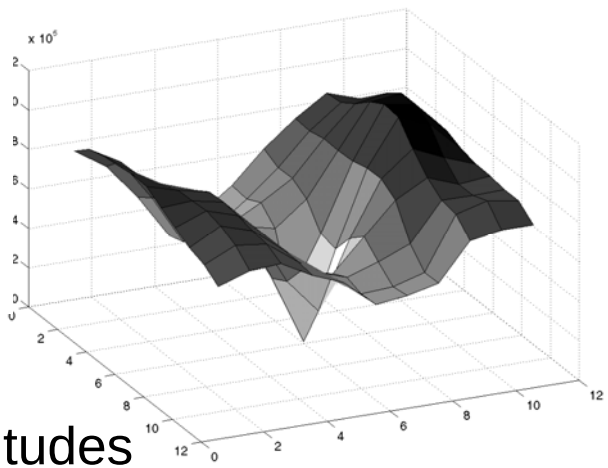


$$A^T A$$

- gradients have small magnitude
- small λ_1, small λ_2
- Solution numerically unstable

Highly textured regions work best





$$A^T A$$

- gradients are different, large magnitudes
- large λ_1, large λ_2
- $A^T A$ invertible

Errors in the Lucas-Kanade Method

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 are our assumptions are violated

- Brightness constancy is **not** satisfied
- A point does **not** move like its neighbors
- The motion is **not** small

Improving accuracy: Beyond small motion

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:

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

This is a polynomial root finding problem

- Can solve using, e.g., **Newton's method**
  - Also known as **Newton-Raphson** method
  - http://en.wikipedia.org/wiki/Newton's_method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Iterative Refinement of (u,v)

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 image warping techniques*
3. Repeat until convergence

Beyond small motion: Take 2

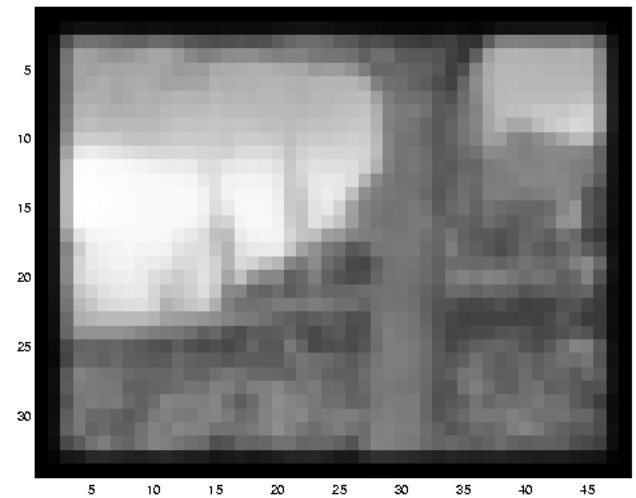
Is the motion between frames small enough?

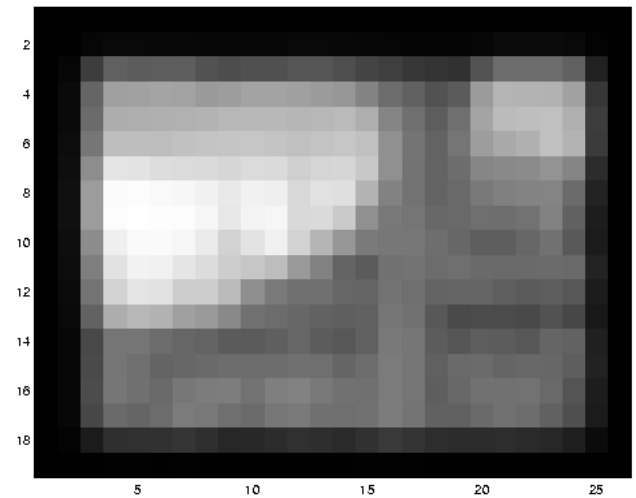
- Probably not—it's much larger than one pixel (2nd order terms dominate)
- How might we solve this problem without higher-order terms?

Reduce the resolution!

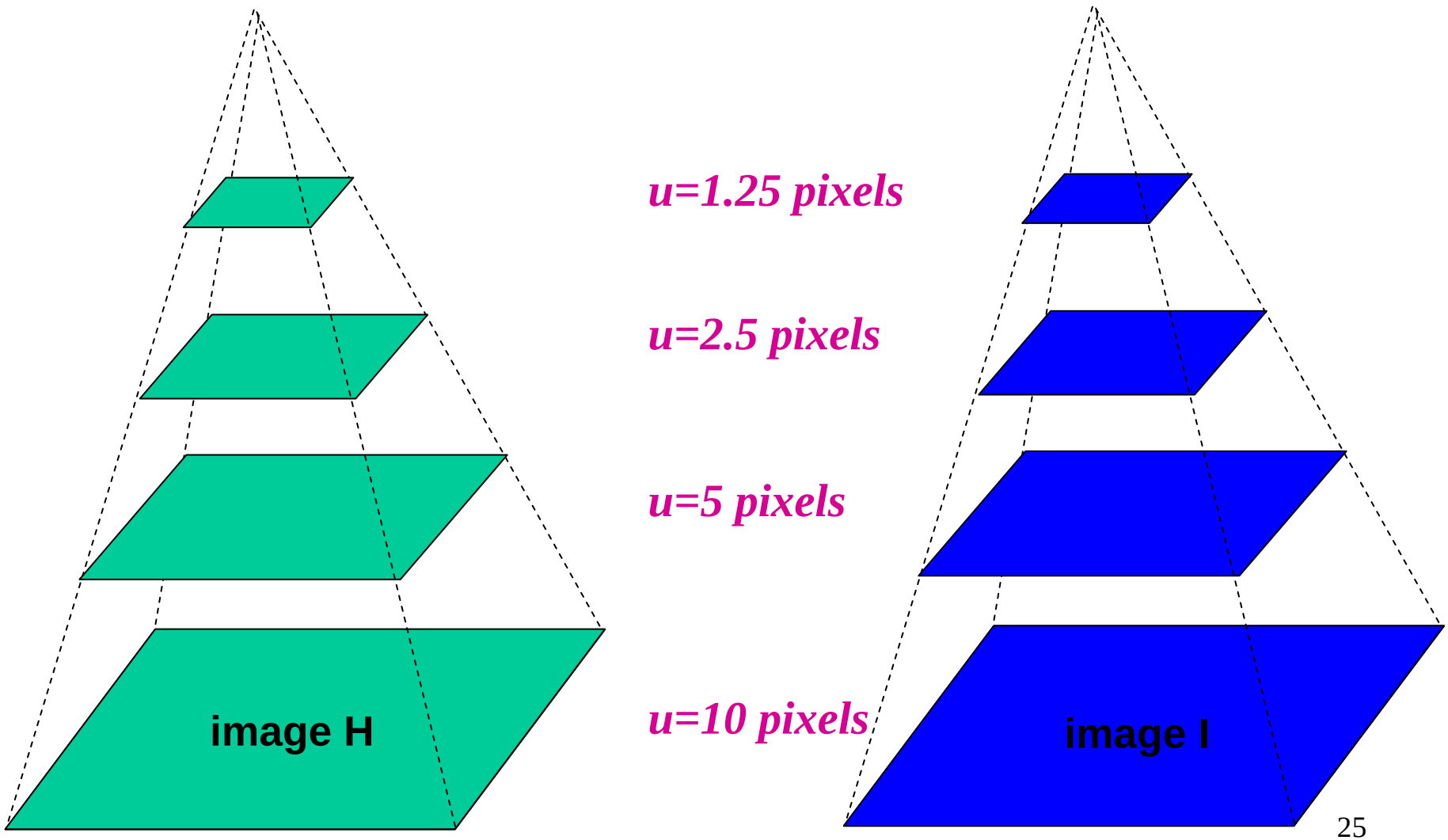
Large motion

Small motion





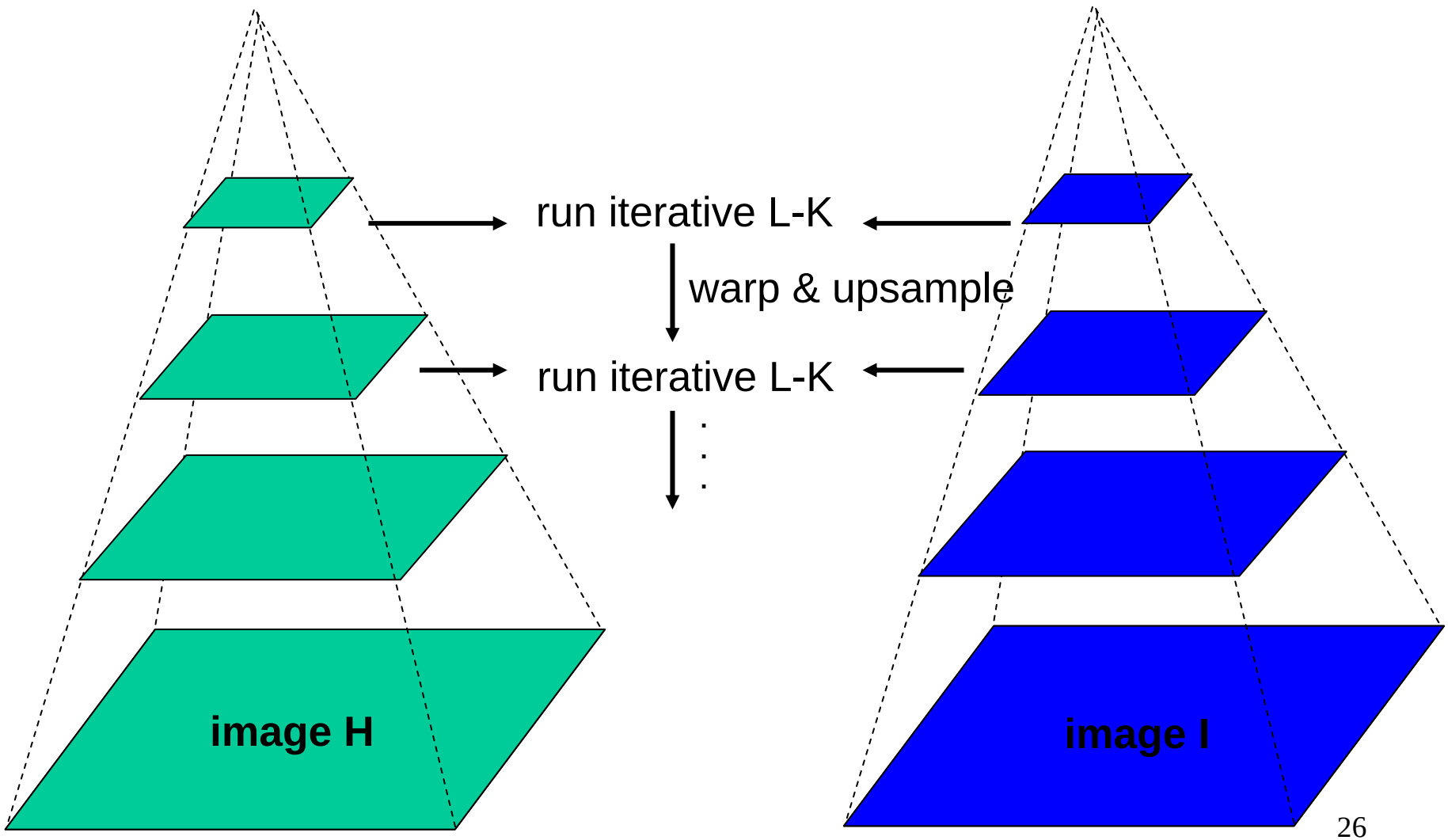
Coarse-to-fine optical flow estimation



Gaussian pyramid of image H

Gaussian pyramid of image I

Coarse-to-fine optical flow estimation



Gaussian pyramid of image H

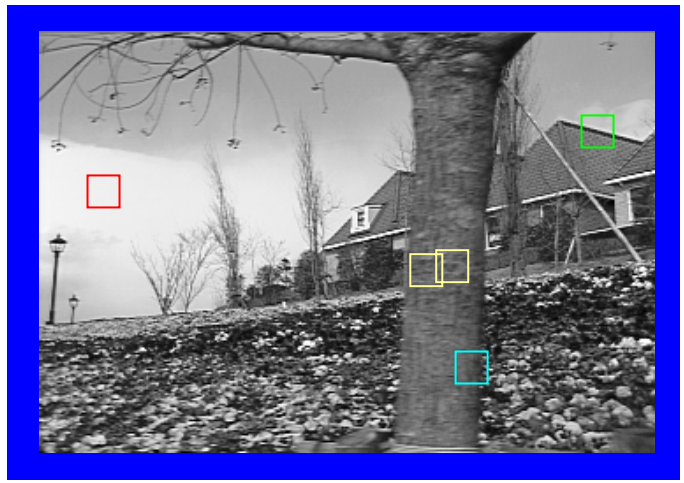
Gaussian pyramid of image I

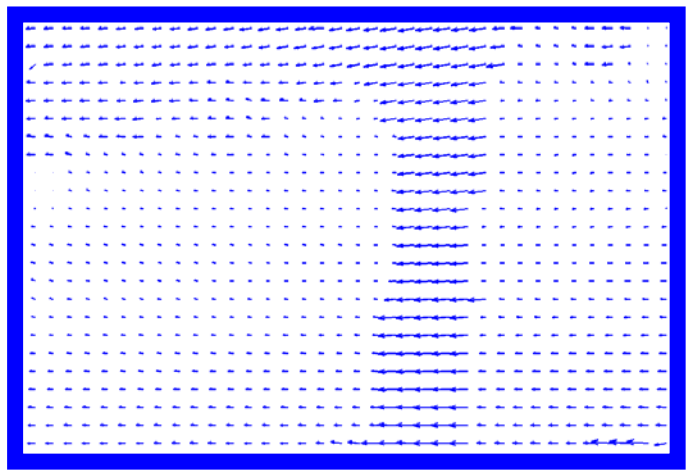
A Few Details

- Top Level
  - Apply L-K to get flow field from 1st frame to 2nd frame.
  - Apply this flow to warp 1st frame toward 2nd frame.
  - Rerun L-K on new warped image to get flow field from it to 2nd frame.
  - Repeat till convergence.
- Next Level
  - Upsample flow field to the next level as first guess of flow at that level.
  - Apply this flow field to warp 1st frame toward 2nd frame.
  - Rerun L-K and warping till convergence as above.
- Etc.

The Flower Garden Video

What should the optical flow be?



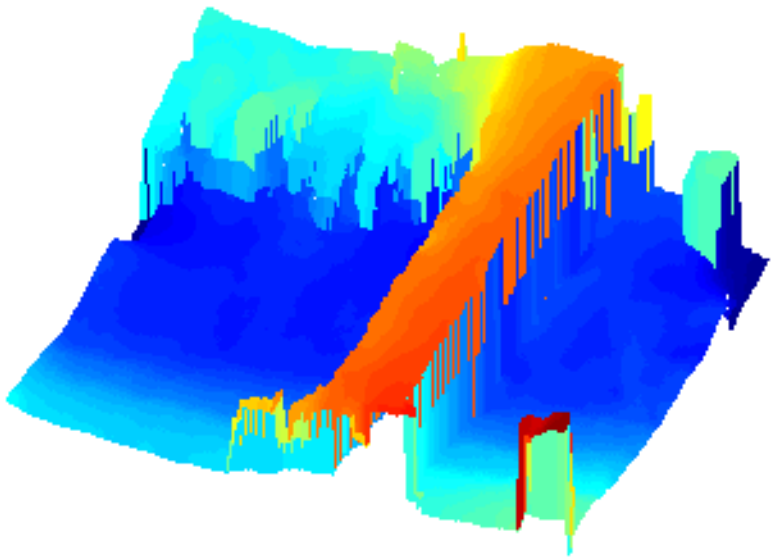


When objects move at equal speed, those more remote seem to move more slowly.

Euclid, 300 BC

Other Applications: Structure From Motion

Estimated motion (horizontal)



Depth map

Other Applications: MPEG encoding

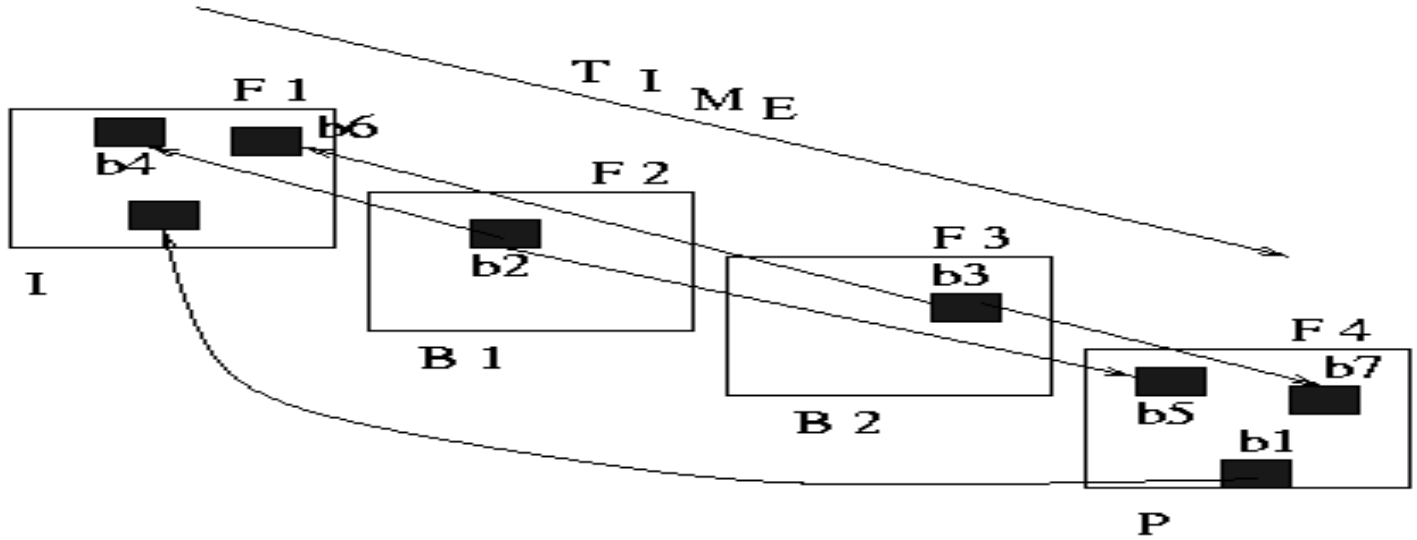
Some frames are encoded in terms of others.

I*ndependent frame* encoded as a still image using JPEG

P*redicted frame* encoded via **flow vectors** relative to the independent frame and difference image.

B*etween frame* encoded using flow vectors and independent and predicted frame.

MPEG compression method

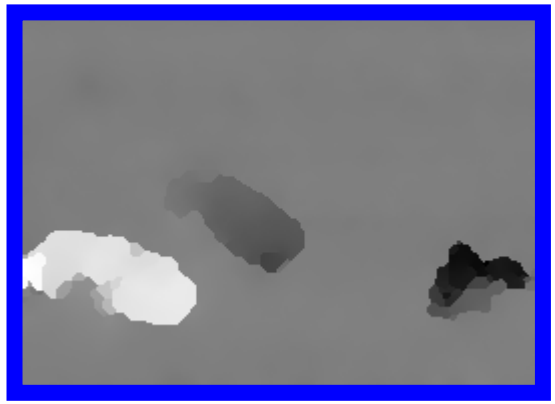


F1 is independent. F4 is predicted. F2 and F3 are between.

Each 16x16 block of P is matched to its closest match in I and represented by a motion vector and a block difference image.

Frames B1 and B2 between I and P are represented using two motion vectors per block referring to blocks in F1 and F4.

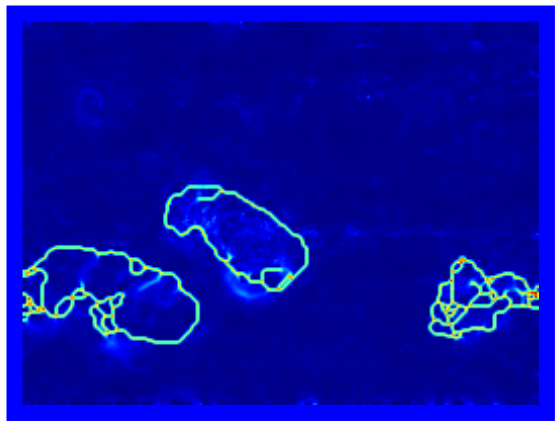
Other Applications: Scene Dynamics Understanding



Brighter pixels => larger speeds.

Estimated horizontal motion

- Surveillance
- Event analysis
- Video compression



Motion boundaries are smooth.

Motion smoothness

Target Detection and Tracking

A tiny airplane --- only observable by its distinct motion

Tracking results

Motion tracking

Suppose we have more than two images

- How to track a point through all of the images?
  - In principle, we could estimate motion between each pair of consecutive frames
  - Given point in first frame, follow arrows in consecutive frames to trace out it's path
  - Problem: DRIFT
    - » small errors will tend to grow and grow over time—the point will drift way off course

Feature Tracking

- Choose only the points ("features") that are easily tracked
- You already know about feature detectors and descriptors
- Pick your favorite (e.g., Harris, SIFT) and use for tracking

Tracking features

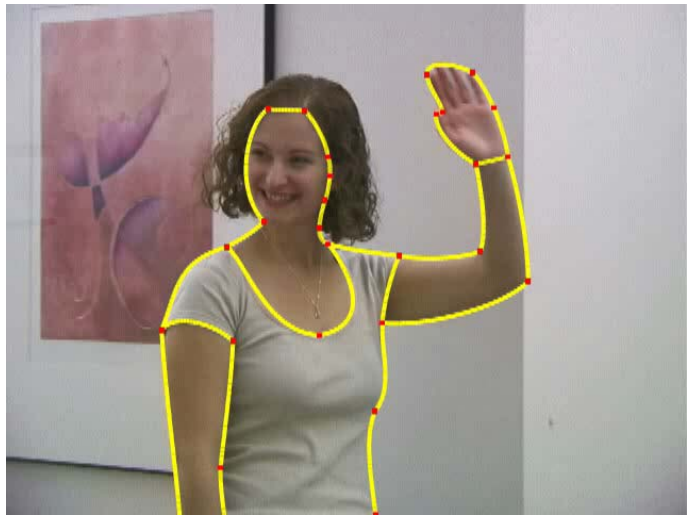
Feature tracking

- Find feature correspondence between consecutive H, I
- Chain these together to find long-range correspondences

When will this go wrong?

- Occlusions—feature may disappear
  - need mechanism for deleting, adding new features
- Changes in shape, orientation
  - allow the feature to deform
- Changes in color

Example Application: Rotoscoping (demo)



Keyframe-Based Tracking for Rotoscoping and Animation
Agarwala et al., SIGGRAPH'04

Next Time: Stereo and 3D Vision

Things to do:

- Work on Project 4
- Read Sec. 12.3 – 12.6