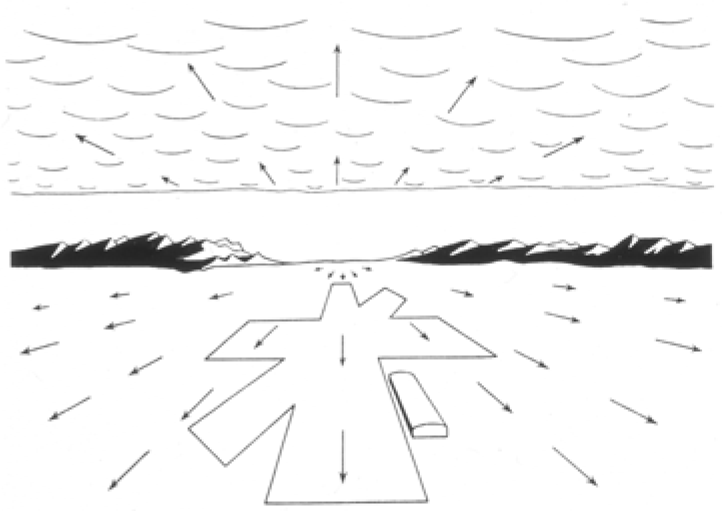
CS 4495 Computer Vision
Motion and Optic Flow

Aaron Bobick
School of Interactive Computing



Administrivia

- PS4 is out, due Sunday Oct 27th.
  - All relevant lectures posted
- Details about Problem Set:
  - You may ***not*** use built in Harris corner functions.
  - We provide instructions to download a SIFT library. (VLFeat) It is callable from Matlab.
    - Library provides both SIFT computation and matching.
  - If using C or Python, you can use the relevant functions in OpenCV
  - There is a “supplement” document that explains these two systems.
  - Scale is not explored.

Visual motion

Many slides adapted from S. Seitz, R. Szeliski, M. Pollefeys, K. Grauman and others…

Visual motion

Many slides adapted from S. Seitz, R. Szeliski, M. Pollefeys, K. Grauman and others…

Visual motion

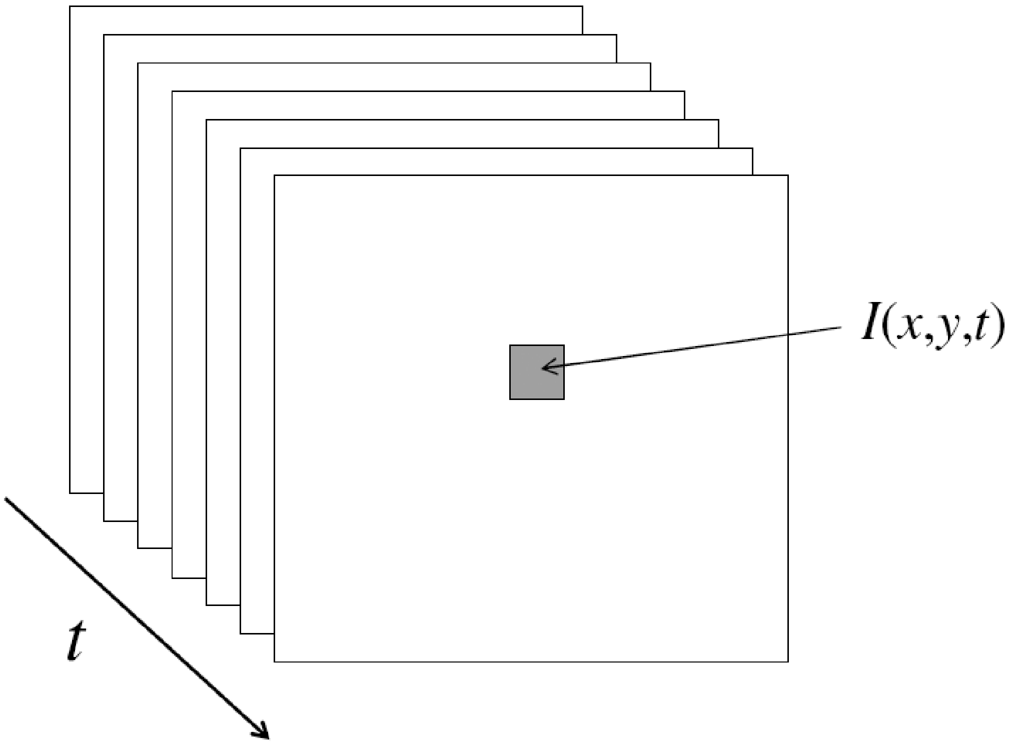
Many slides adapted from S. Seitz, R. Szeliski, M. Pollefeys, K. Grauman and others…

Visual motion

Many slides adapted from S. Seitz, R. Szeliski, M. Pollefeys, K. Grauman and others…

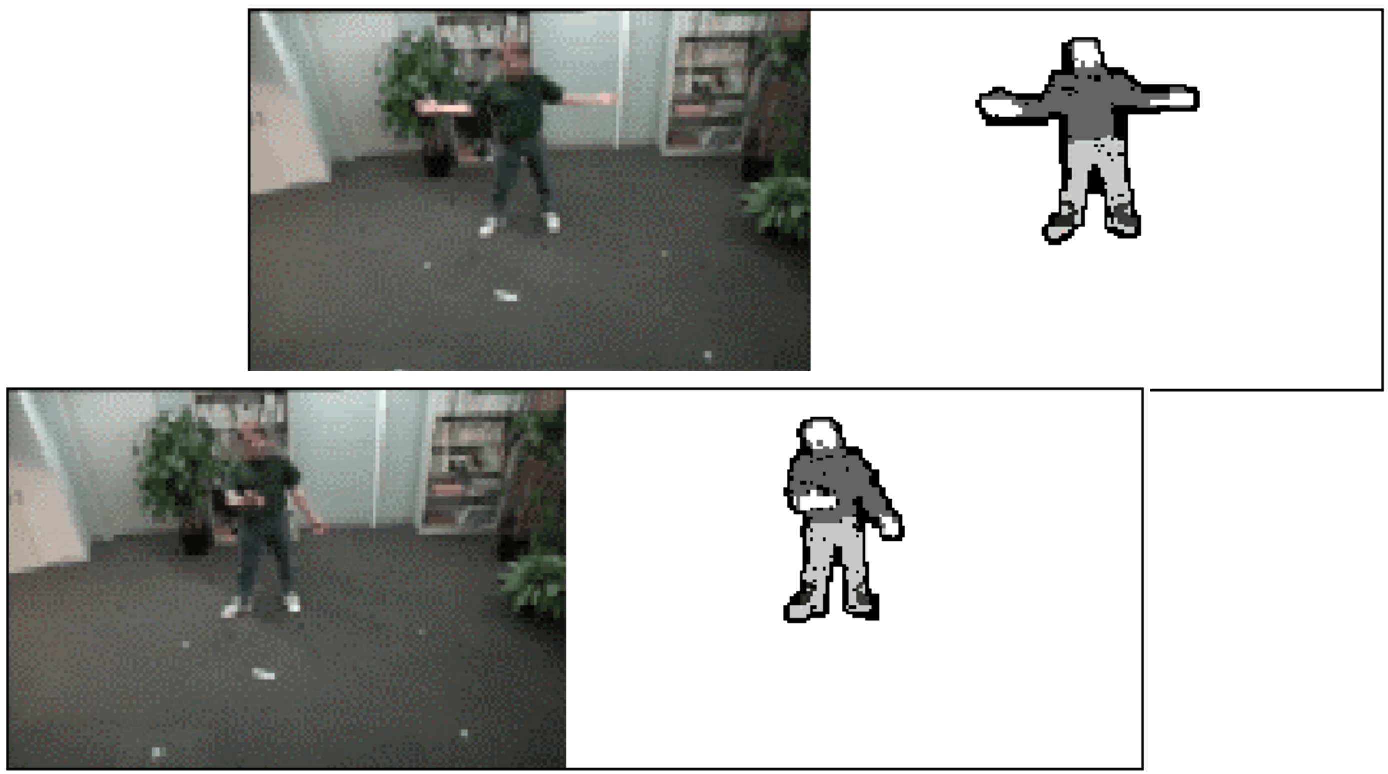
Video

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



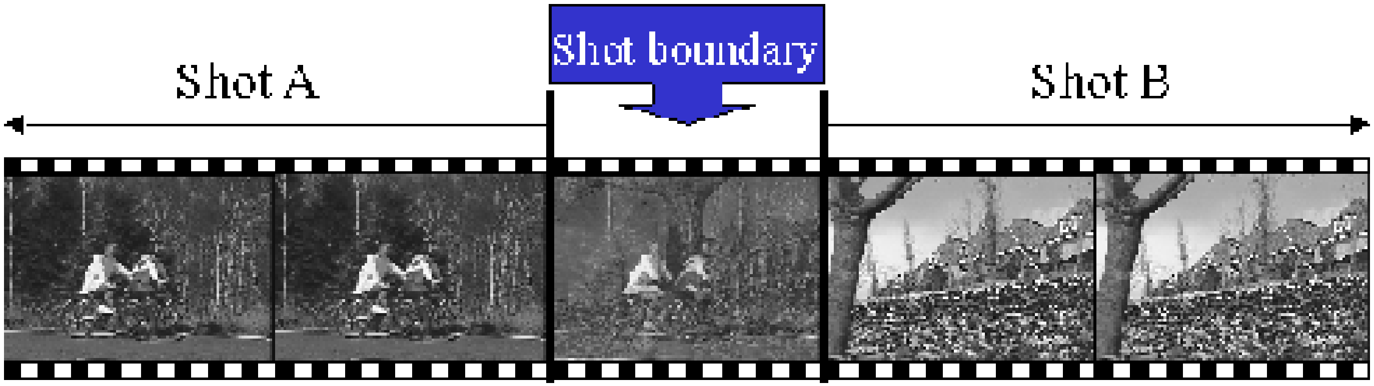
Motion Applications: Segmentation of video

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



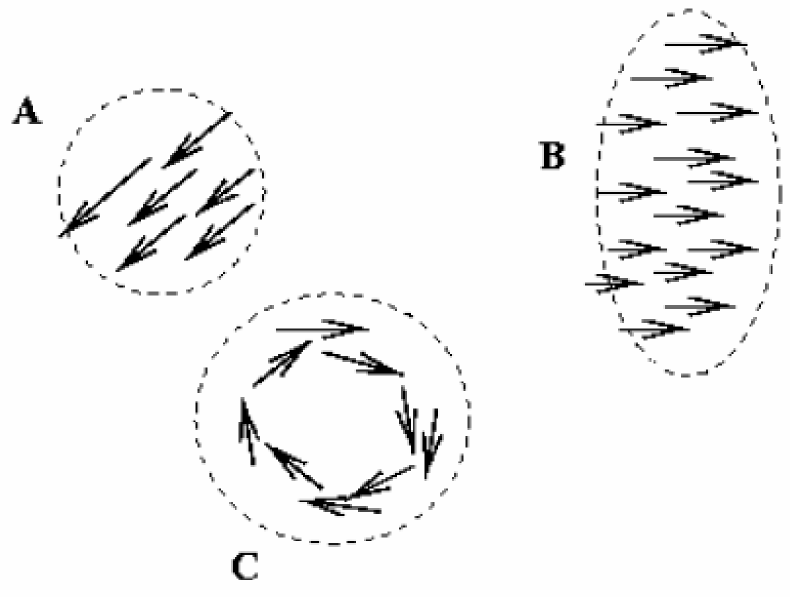
Motion Applications: Segmentation of video

- 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



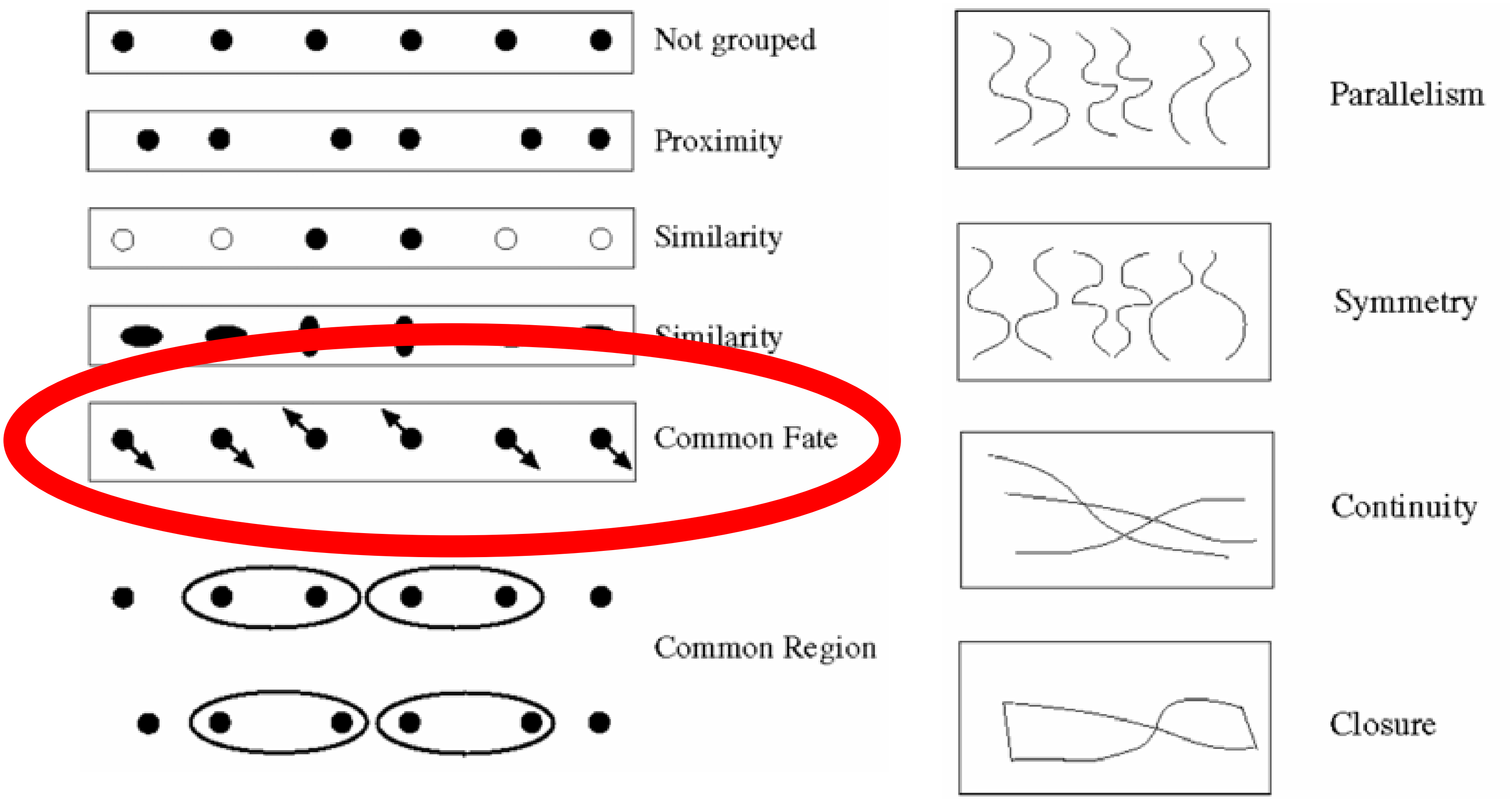
Motion Applications: Segmentation of video

- 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

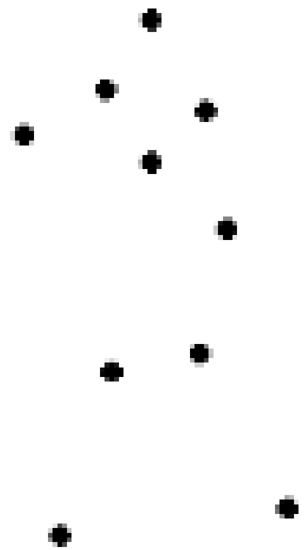


Motion and perceptual organization

- Sometimes, motion is the only cue
  http://www.youtube.com/watch?v=aEoxO_RdGhE

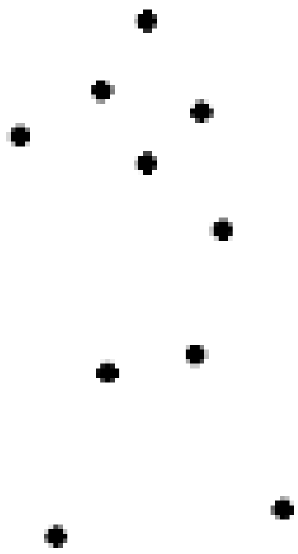
Motion and perceptual organization

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

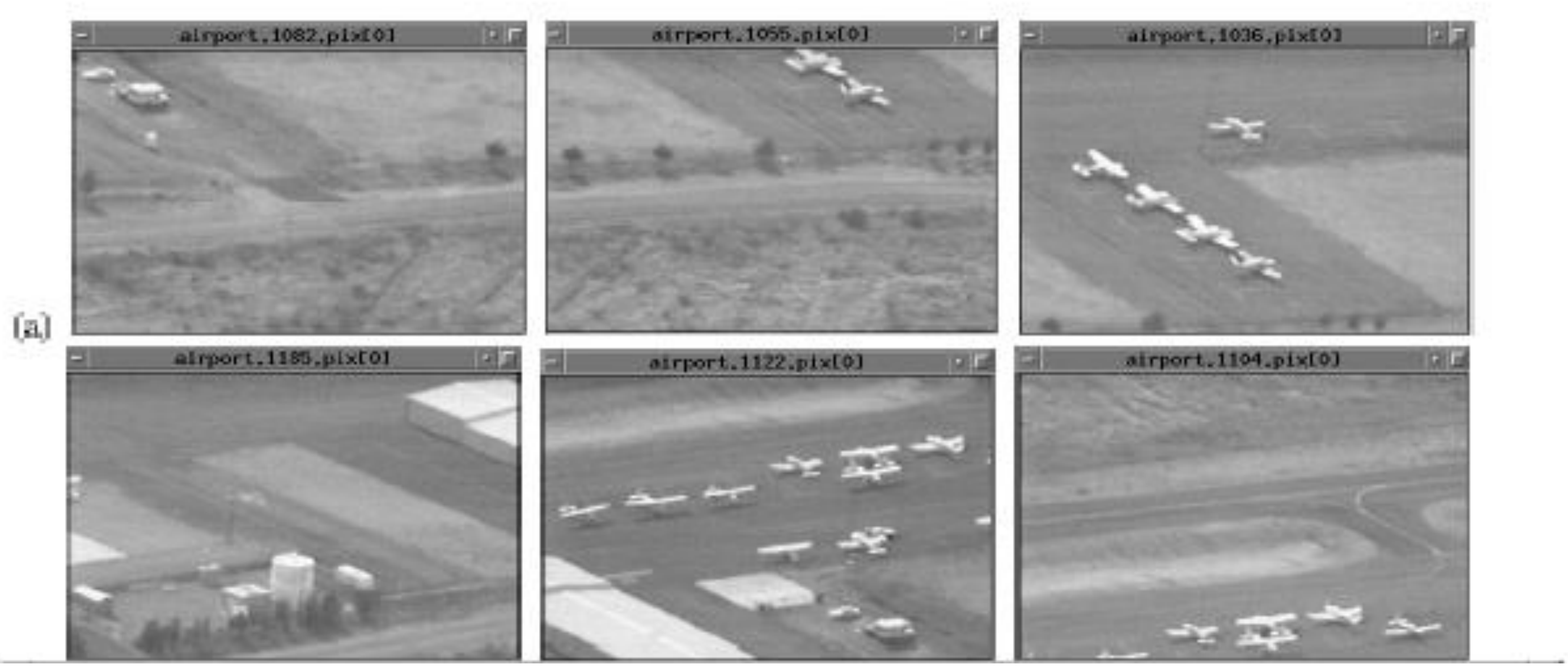


Motion and perceptual organization

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



Mosaicing



(Michal Irani, Weizmann)

Mosaicing

g. 1. Static background mosaic of an airport video clip.
(a) A few representative frames from the minute-long video clip. The video shows an airport being imaged from the air with a moving camera. The scene itself is static (i.e., no moving objects). (b) The static background mosaic image which provides an extended view of the entire scene imaged by the camera in the one-minute video clip.

(Michal Irani, Weizmann)

More applications of motion

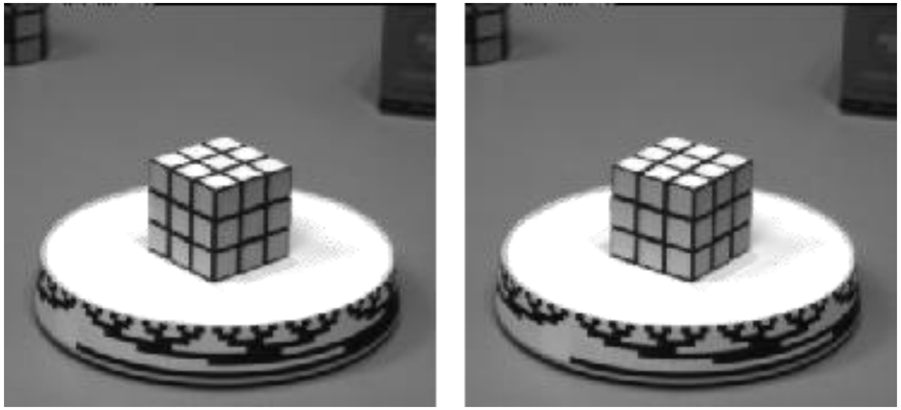
- Segmentation of objects in space or time
- Estimating 3D structure
- Learning dynamical models – how things move
- Recognizing events and activities
- Improving video quality (motion stabilization)

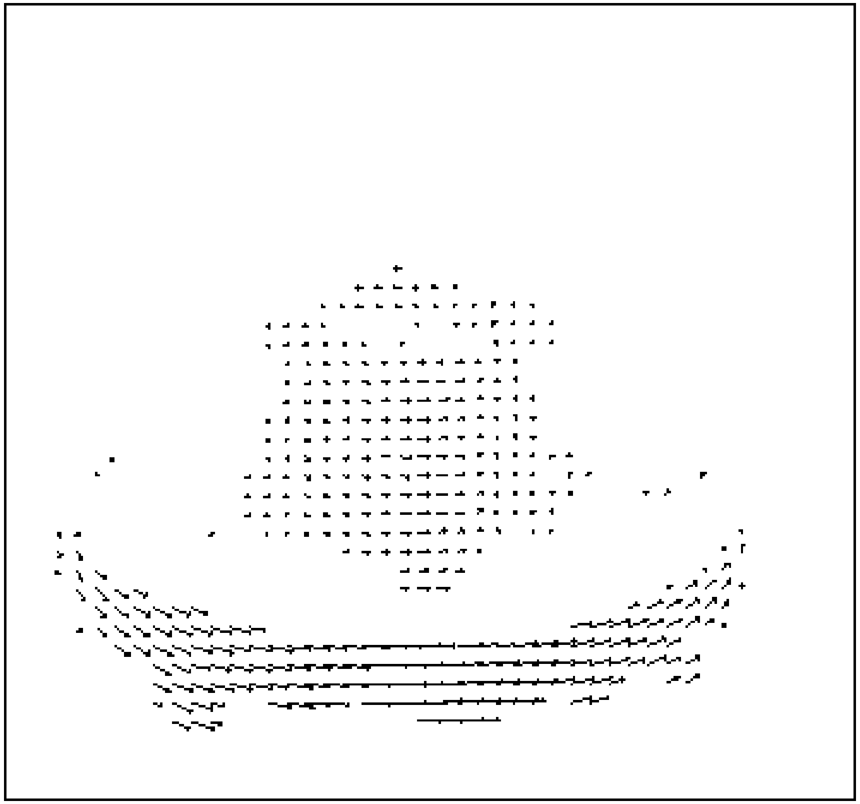
Motion estimation techniques

- Direct, dense 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

- 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)

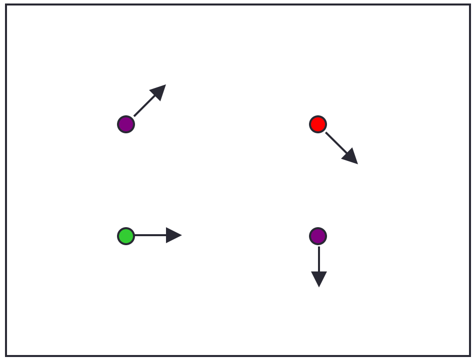
Motion estimation: Optical flow



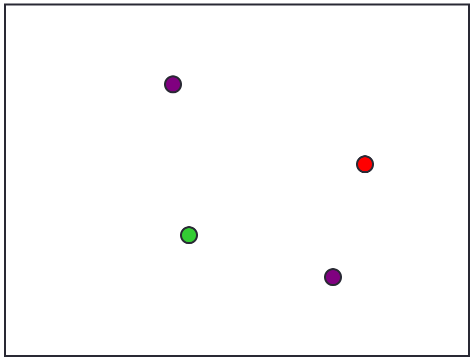


Will start by estimating motion of each pixel separately
Then will consider motion of entire image

Problem definition: optical flow



$I(x, y, t)$



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

How to estimate pixel motion from image $I(x,y,t)$ to $I(x,y,t+1)$

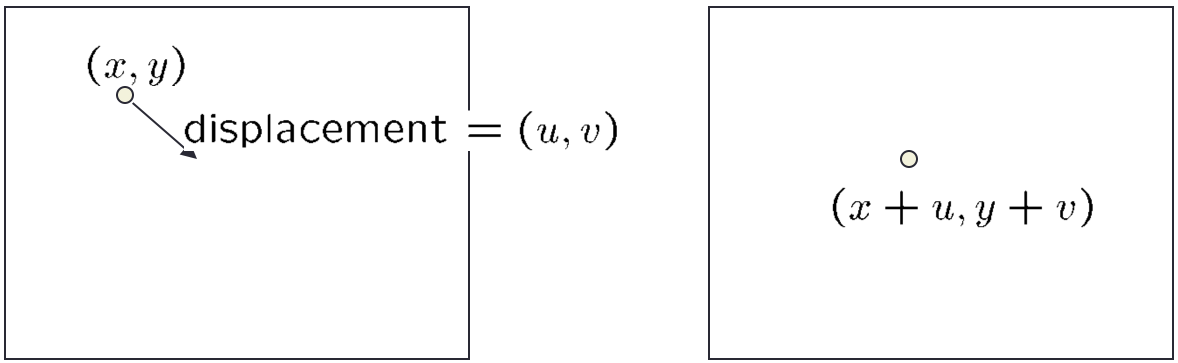
- Solve pixel correspondence problem
  - given a pixel in I(x,y,t), look for nearby pixels of the same color in $I(x,y,t+1)$

Key assumptions

- color constancy: a point in $I(x,y,$ looks the same in I(x,y,t+1)
  - 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)



$I(x, y, t)$ $\qquad$ $I(x, y, t+1)$

- Let's look at these constraints more closely
  - brightness constancy constraint (equation)

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

  - small motion: (u and v are less than 1 pixel, or smooth)
    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

$$0 = I(x+u, y+v, t+1) - I(x, y, t)$$

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

(Short hand: $I_x = \frac{\partial I}{\partial x}$ for t **or** $t+1$)

Optical flow equation

- Combining these two equations

$$
\begin{aligned}
0 &= I(x+u, y+v, t+1) - I(x, y, t) \\
&\approx I(x, y, t+1) + I_x u + I_y v - I(x, y, t) \\
&\approx [I(x, y, t+1) - I(x, y, t)] + I_x u + I_y v \\
&\approx I_t + I_x u + I_y v \\
&\approx I_t + \nabla I \cdot \langle u, v \rangle
\end{aligned}
$$

(Short hand: $I_x = \frac{\partial I}{\partial x}$
for t **or** $t+1$)

Optical flow equation

- Combining these two equations

$$
\begin{aligned}
0 &= I(x+u, y+v, t+1) - I(x, y, t) \\
&\approx I(x, y, t+1) + I_x u + I_y v - I(x, y, t) \\
&\approx \boxed{[I(x, y, t+1) - I(x, y, t)]} + I_x u + I_y v \\
&\approx \boxed{I_t} + I_x u + I_y v \\
&\approx I_t + \nabla I \cdot \langle u, v \rangle
\end{aligned}
$$

(Short hand: $I_x = \frac{\partial I}{\partial x}$ for t **or** $t+1$)

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

$$0 = I_t + \nabla I \cdot \langle u, v \rangle$$

Brightness constancy constraint equation

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

Optical flow equation

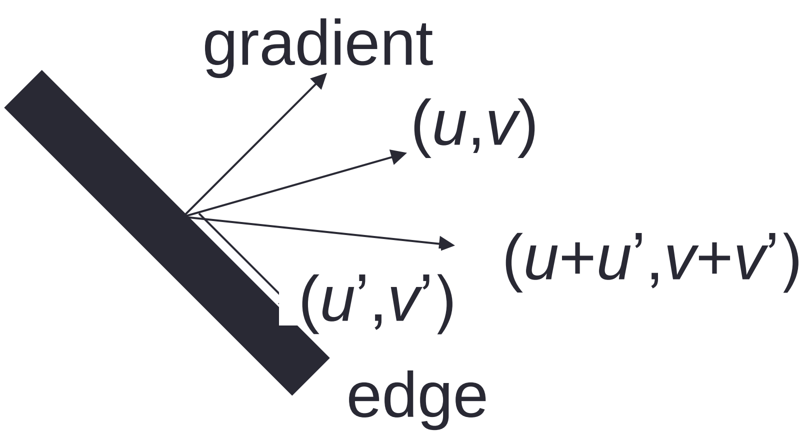
$$0 = I_t + \nabla I \cdot \langle u, v \rangle \quad \text{or} \quad I_x u + I_y v + I_t = 0$$

- Q: how many unknowns and equations per pixel?

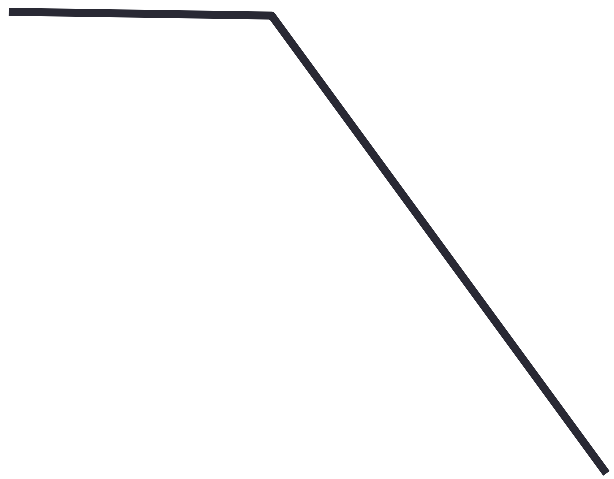
2 unknowns, one equation

Intuitively, what does this constraint mean?

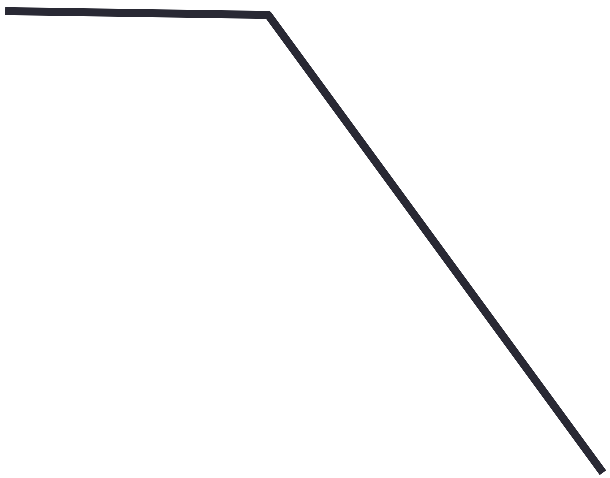
- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown



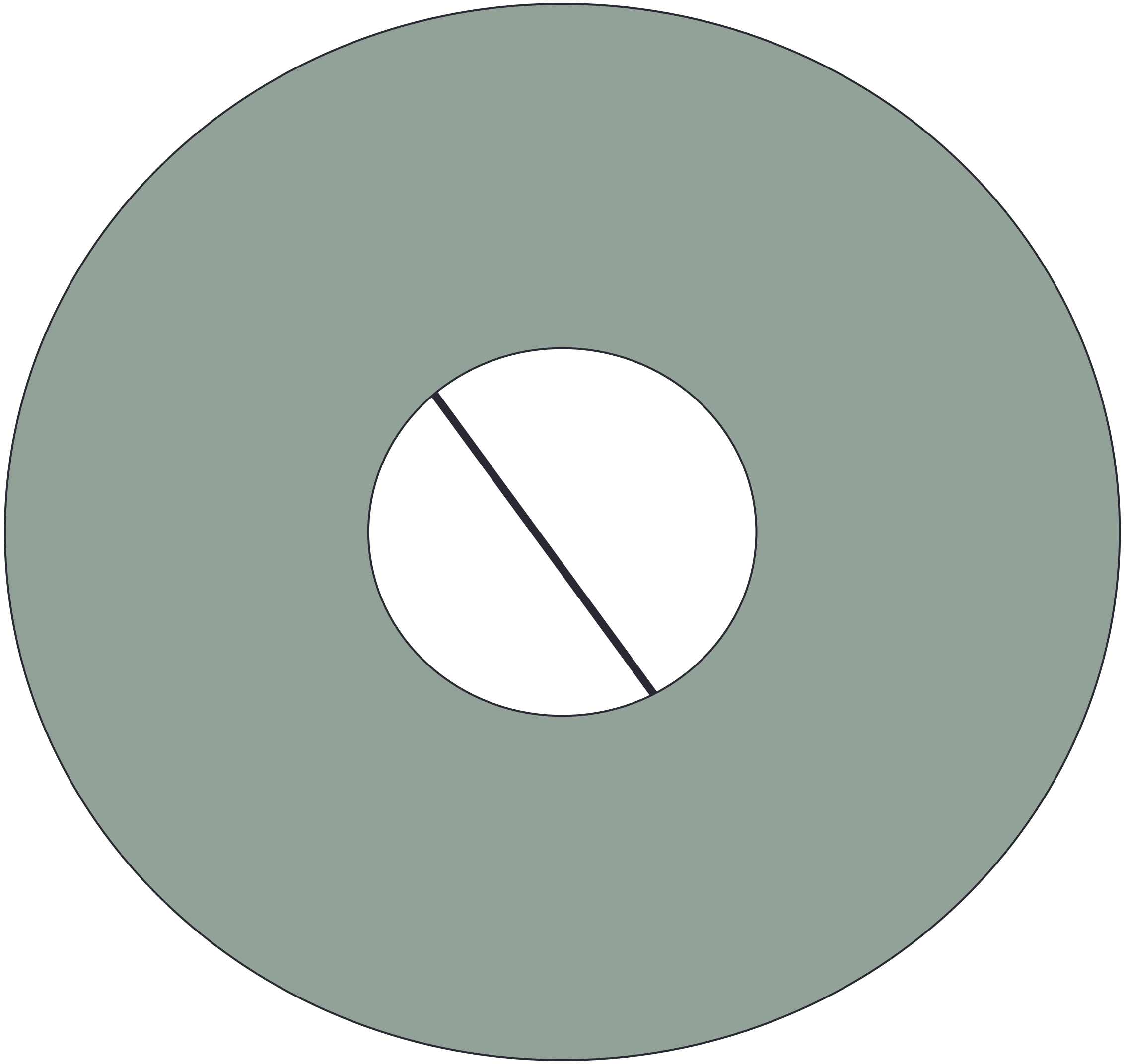
Aperture problem



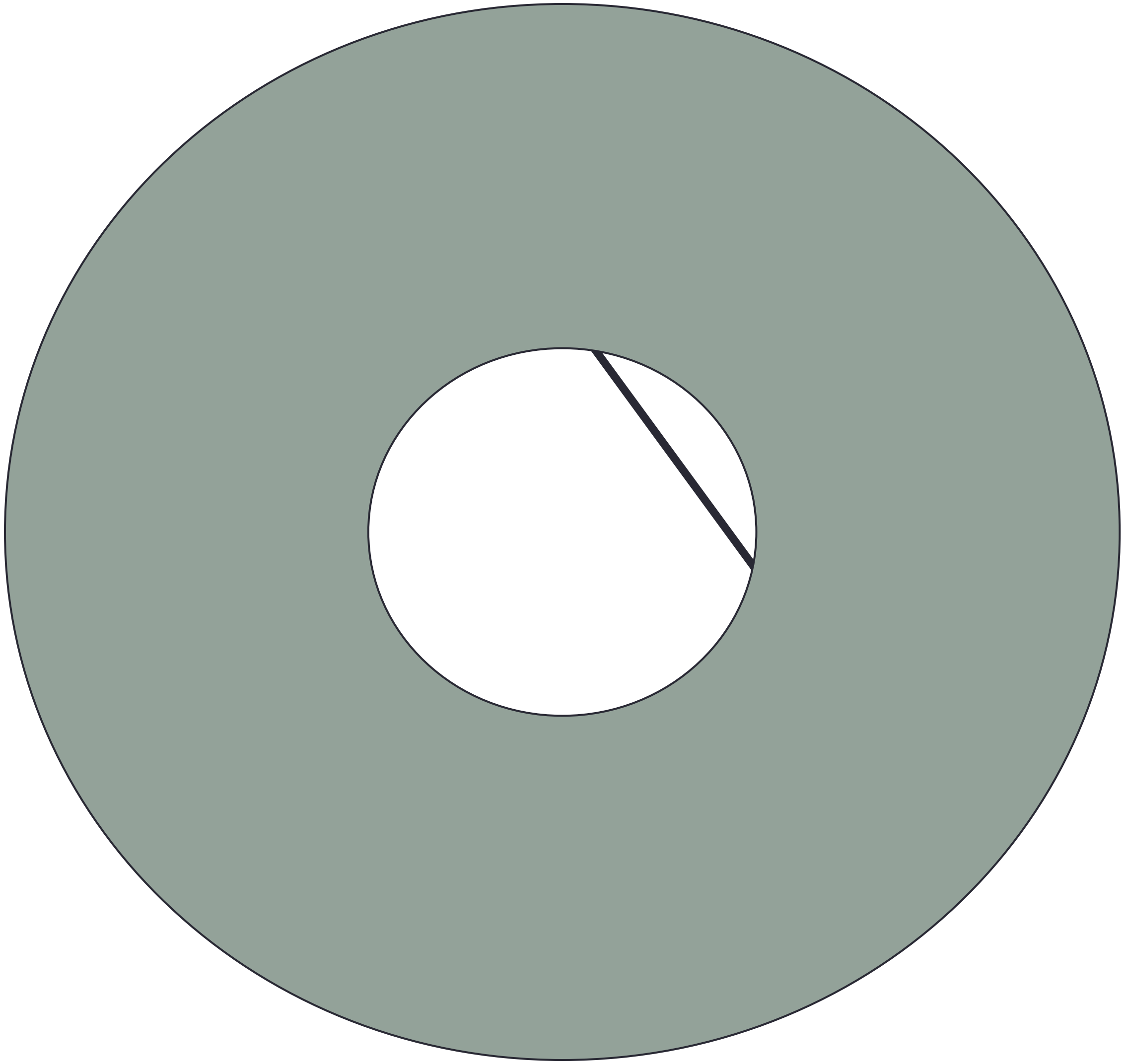
Aperture problem



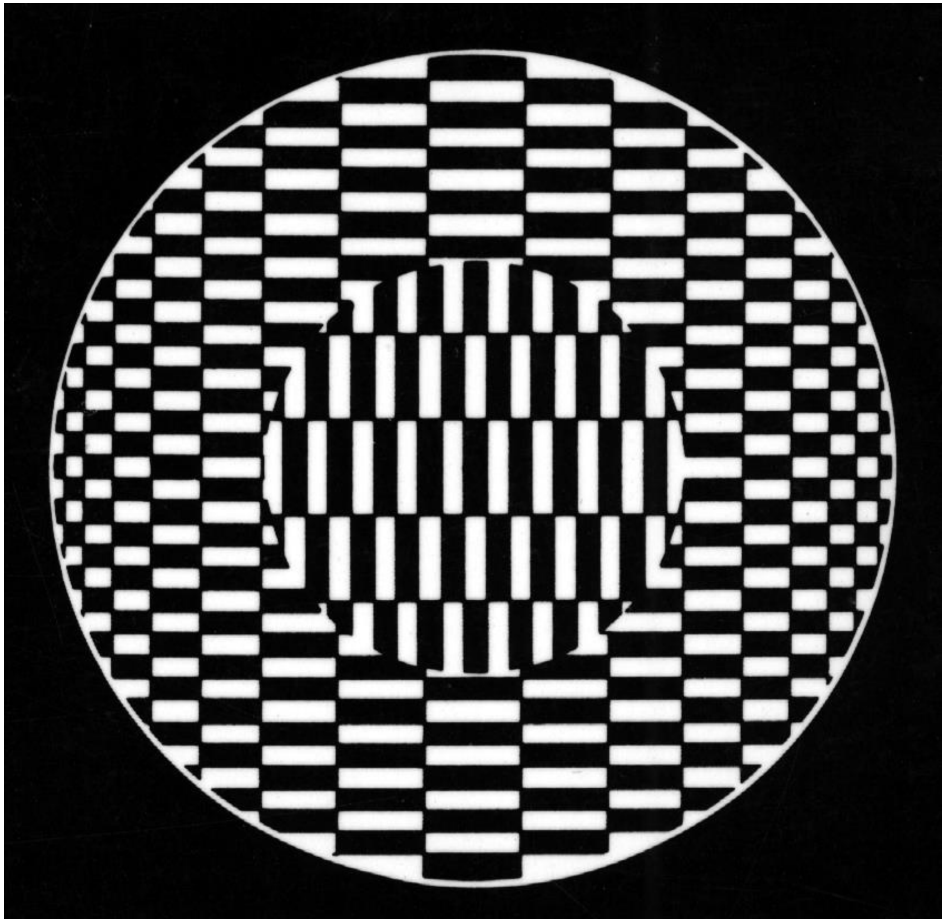
Aperture problem



Aperture problem



Apparently an aperture problem



Optical flow equation

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

- Q: how many unknowns and equations per pixel?

2 unknowns, one equation

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

http://en.wikipedia.org/wiki/Barber's_pole

Some folks say: "This explains the Barber Pole illusion"
http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm
http://www.liv.ac.uk/~marcob/Trieste/barberpole.html

Not quite… where do the vectors point?

Smooth Optical Flow (Horn and Schunk - long ago)

- **Formulate Error in Optical Flow Constraint:**

$$e_c = \iint_{image} (I_x u + I_y v + I_t)^2 \, dx \, dy$$

- **We need additional constraints!**

- **Smoothness Constraint (as in shape from shading and stereo):**

Usually motion field varies smoothly in the image.
So, penalize departure from smoothness:

$$e_s = \iint_{image} (u_x^2 + u_y^2) + (v_x^2 + v_y^2) \; dx \; dy$$

- **Find (u,v) at each image point that MINIMIZES:**

$$e = e_s + \lambda e_c$$

weighting factor

Solving the aperture 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!

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

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

RGB 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!

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

$$\underbrace{\begin{bmatrix} I_x(\mathbf{p_1})[0] & I_y(\mathbf{p_1})[0] \\ I_x(\mathbf{p_1})[1] & I_y(\mathbf{p_1})[1] \\ I_x(\mathbf{p_1})[2] & I_y(\mathbf{p_1})[2] \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}})[0] & I_y(\mathbf{p_{25}})[0] \\ I_x(\mathbf{p_{25}})[1] & I_y(\mathbf{p_{25}})[1] \\ I_x(\mathbf{p_{25}})[2] & I_y(\mathbf{p_{25}})[2] \end{bmatrix}}_{\substack{A \\ 75\times2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\substack{d \\ 2\times1}} = - \underbrace{\begin{bmatrix} I_t(\mathbf{p_1})[0] \\ I_t(\mathbf{p_1})[1] \\ I_t(\mathbf{p_1})[2] \\ \vdots \\ I_t(\mathbf{p_{25}})[0] \\ I_t(\mathbf{p_{25}})[1] \\ I_t(\mathbf{p_{25}})[2] \end{bmatrix}}_{\substack{b \\ 75\times1}}$$

Note that RGB alone at pixel is not enough to disambiguate because R, G & B are correlated. Just provides better gradient

Lukas-Kanade flow

- 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

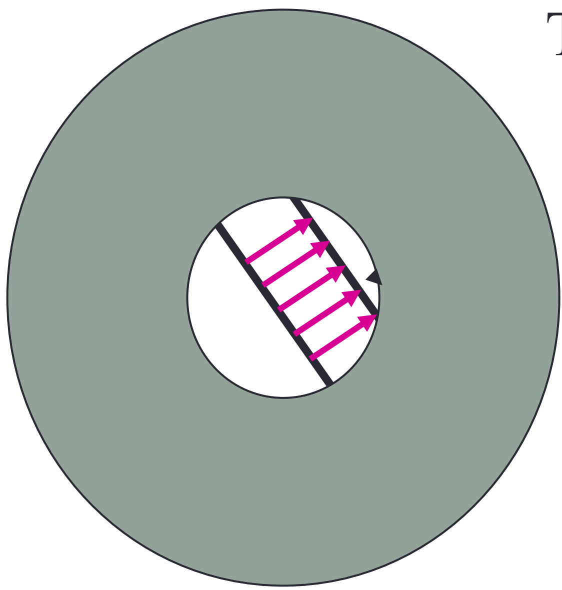
- minimum least squares solution given by solution (in d) of:

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

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

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

Aperture Problem and Normal Flow

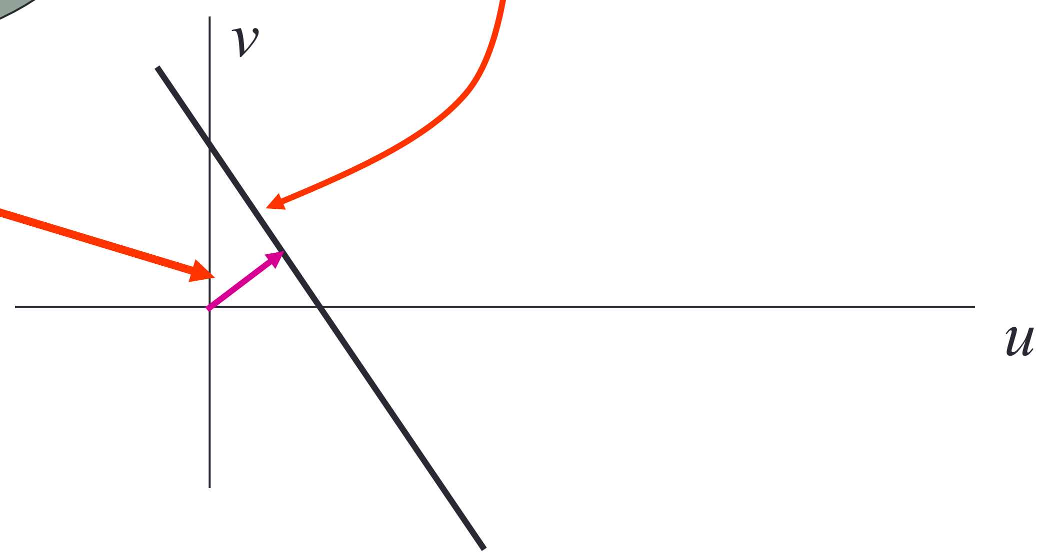


The gradient constraint:

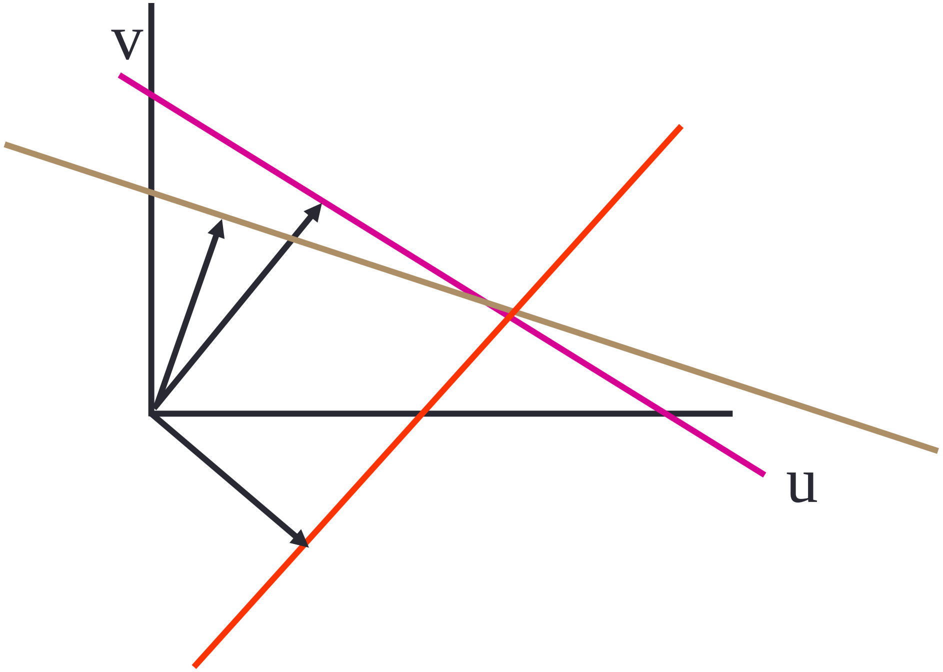
$$\boxed{\begin{aligned} I_x u + I_y v + I_t &= 0 \\ \nabla I \bullet \vec{U} &= 0 \end{aligned}}$$

Defines a line in the *(u,v)* space

Normal Flow:



Combining Local Constraints



$$\nabla I^1 \bullet U = -I_t^1$$

$$\nabla I^2 \bullet U = -I_t^2$$

$$\nabla I^3 \bullet U = -I_t^3$$

etc.

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

A^TA is solvable when there is no aperture problem

– Does this remind you of something???

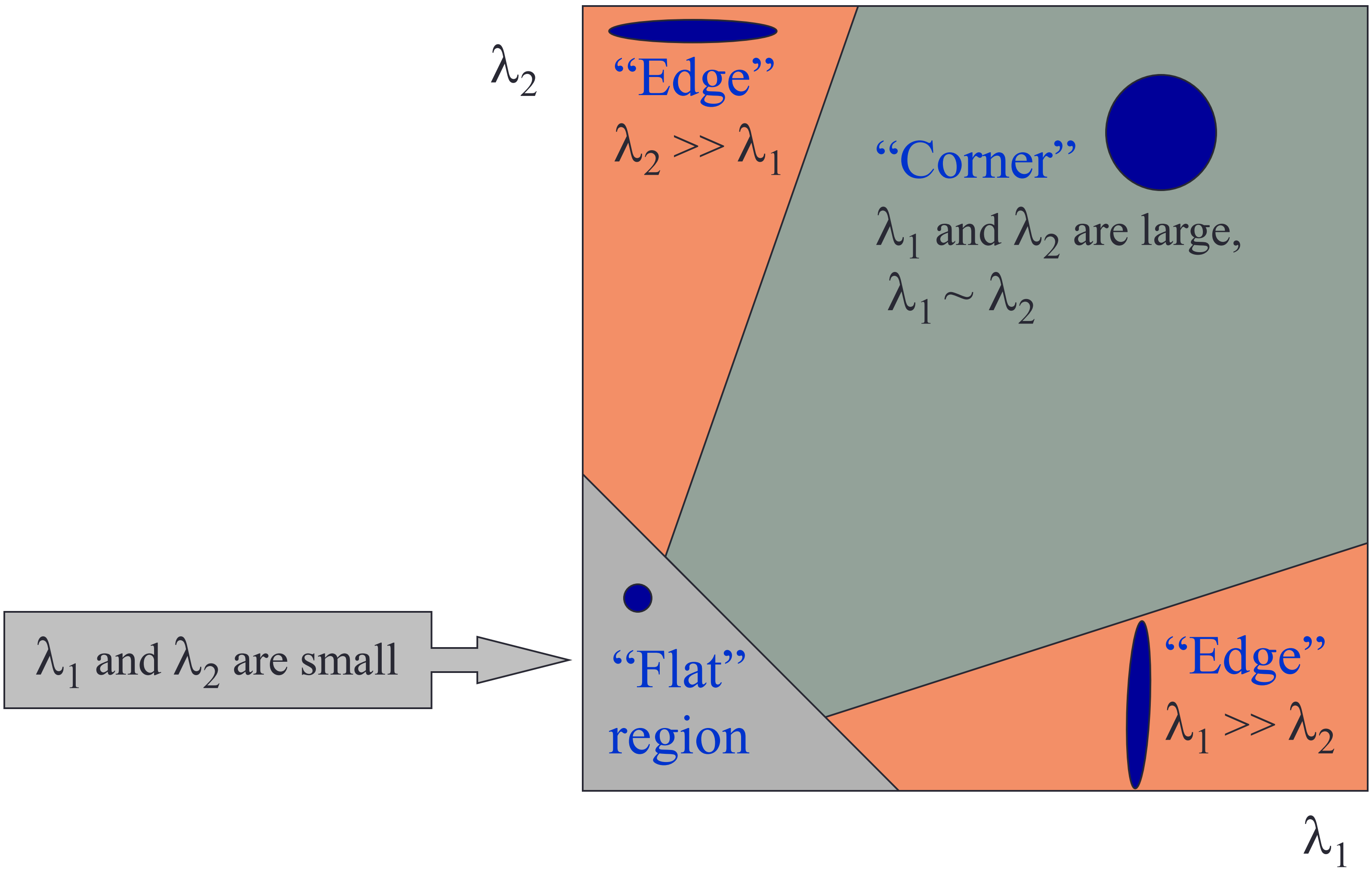
Eigenvectors of A^TA

$$A^T A = \left[\begin{array}{cc} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{array} \right] = \sum \left[\begin{array}{c} I_x \\ I_y \end{array} \right] [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

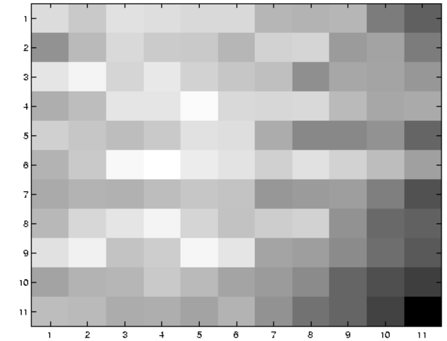
- Recall the Harris corner detector: $M = A^TA$ is the *second moment matrix*
- The eigenvectors and eigenvalues of M relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it

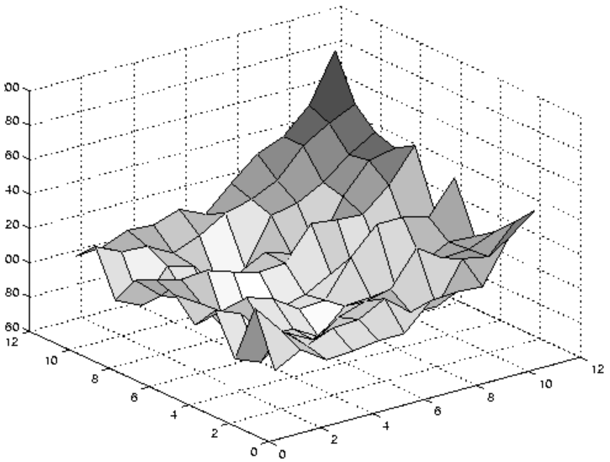
Interpreting the eigenvalues

Classification of image points using eigenvalues of the second moment matrix:



Low texture region



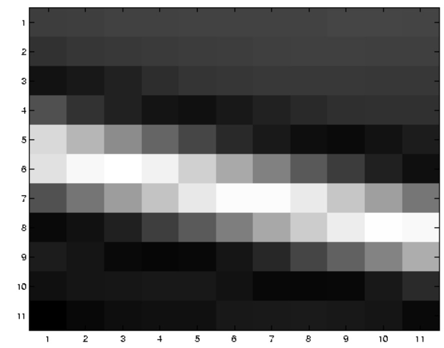


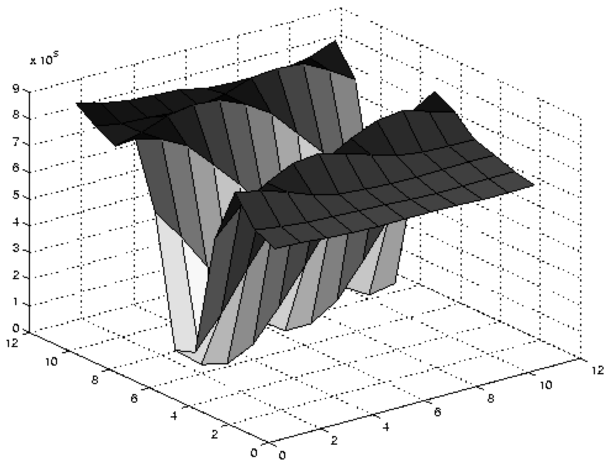
$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small λ_1, small λ_2

Edge

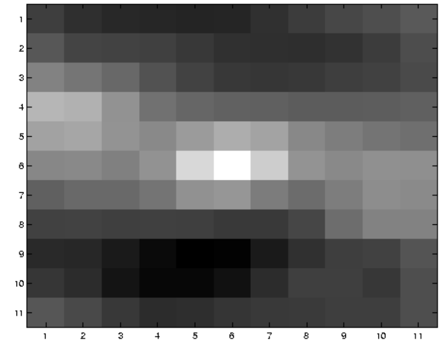


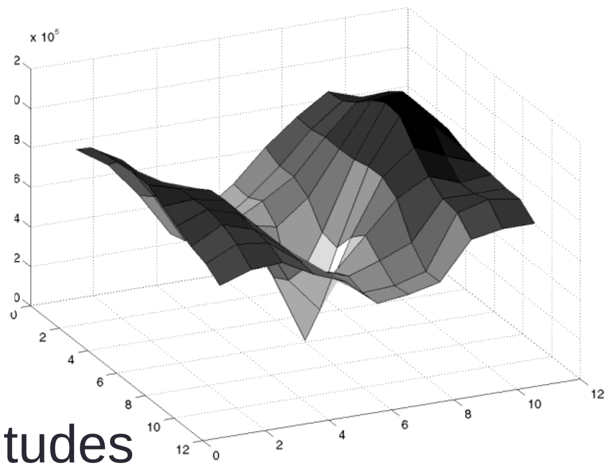


$$\sum \nabla I (\nabla I)^T$$

– large gradients, all the same
– large 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

Errors in Lucas-Kanade

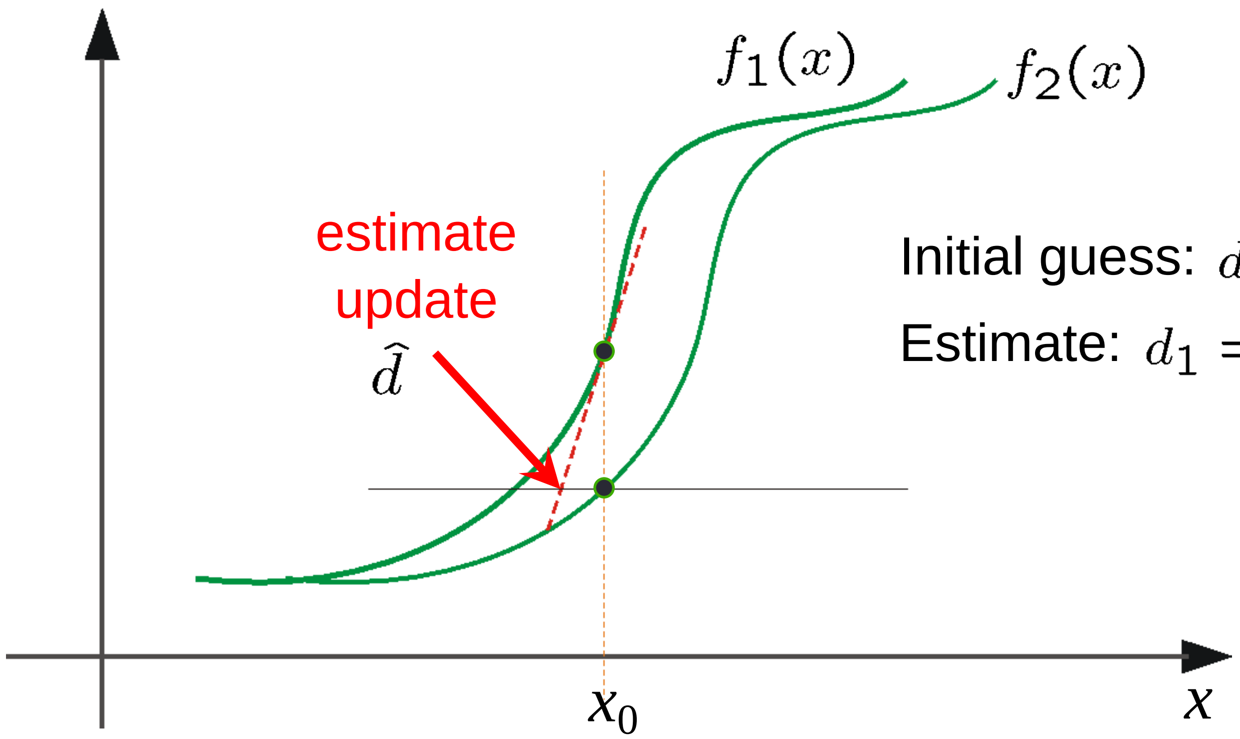
- The motion is large (larger than a pixel)
  - Not-linear: Iterative refinement
  - Local minima: coarse-to-fine estimation
- A point does not move like its neighbors
  - Motion segmentation
- Brightness constancy does not hold
  - Do exhaustive neighborhood search with normalized correlation - tracking features – maybe SIFT – more later….

Not tangent: Iterative Refinement

Iterative Lukas-Kanade Algorithm

1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp I_t towards I_{t+1} using the estimated flow field
   - *- use image warping techniques*
3. Repeat until convergence

Optical Flow: Iterative Estimation

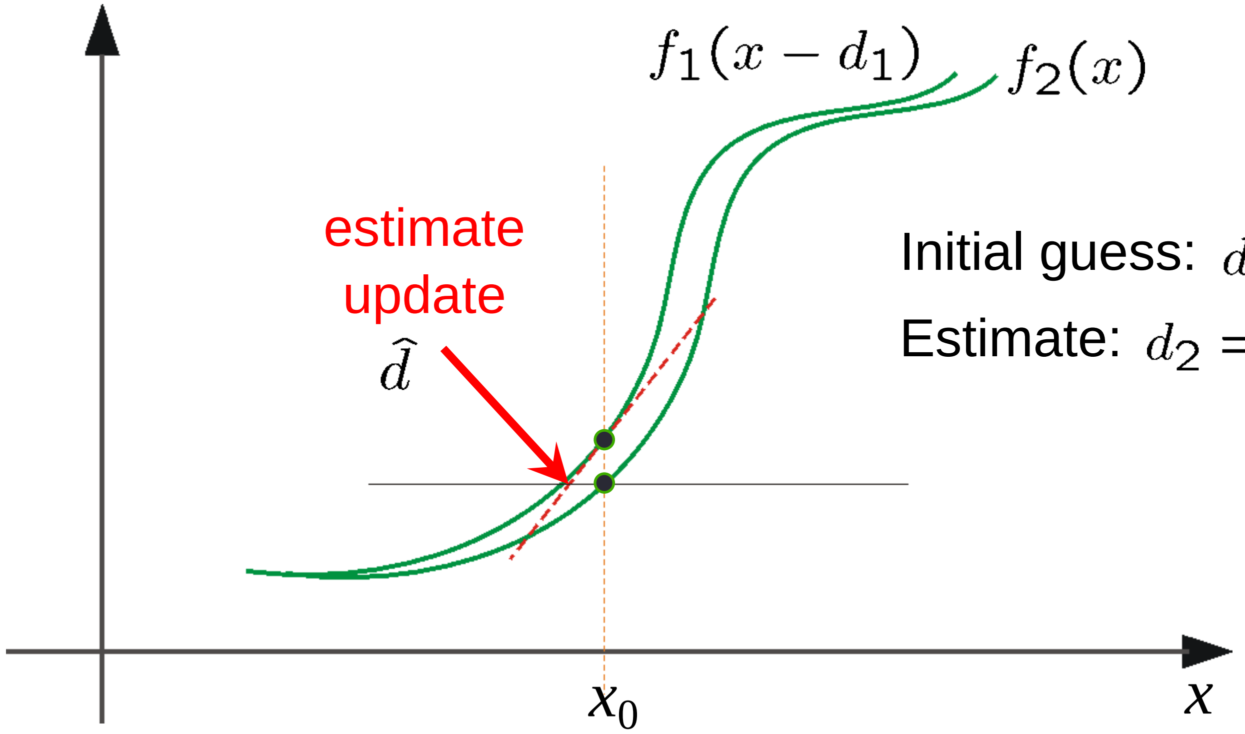


Initial guess: $d_0 = 0$

Estimate: $d_1 = d_0 + \hat{d}$

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

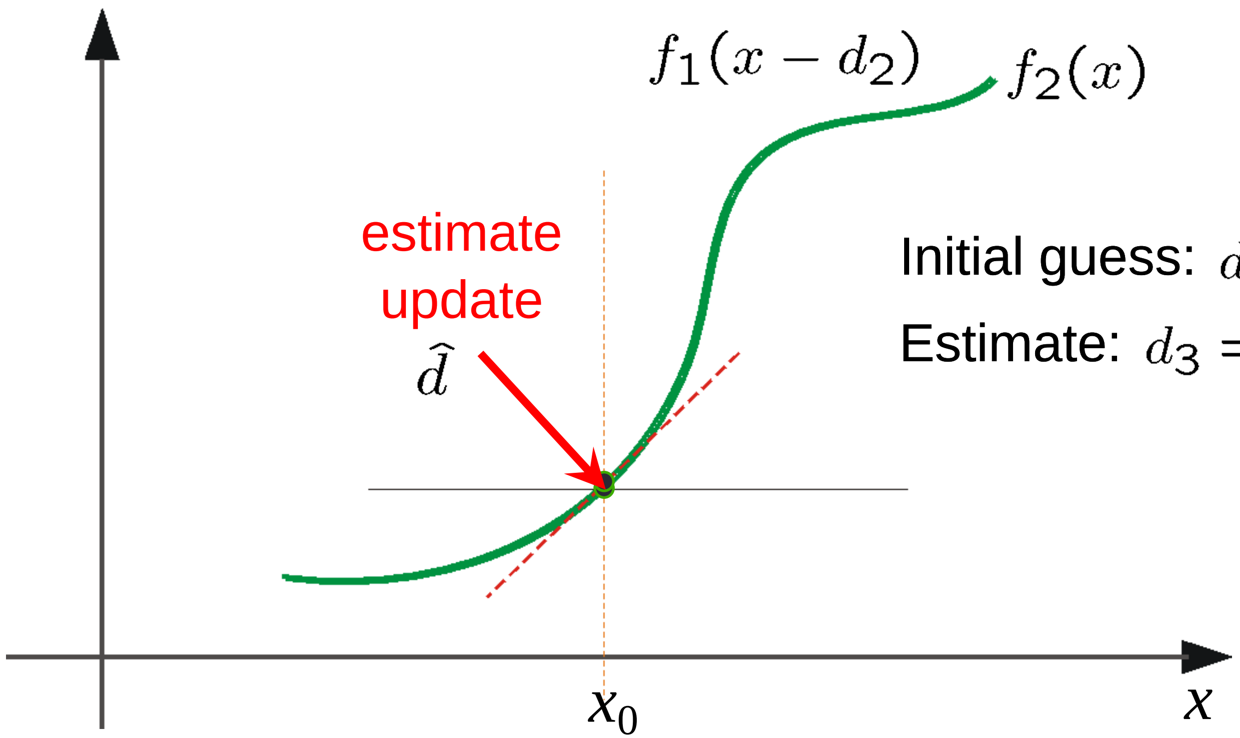
Optical Flow: Iterative Estimation



Initial guess: d_1

Estimate: $d_2 = d_1 + \hat{d}$

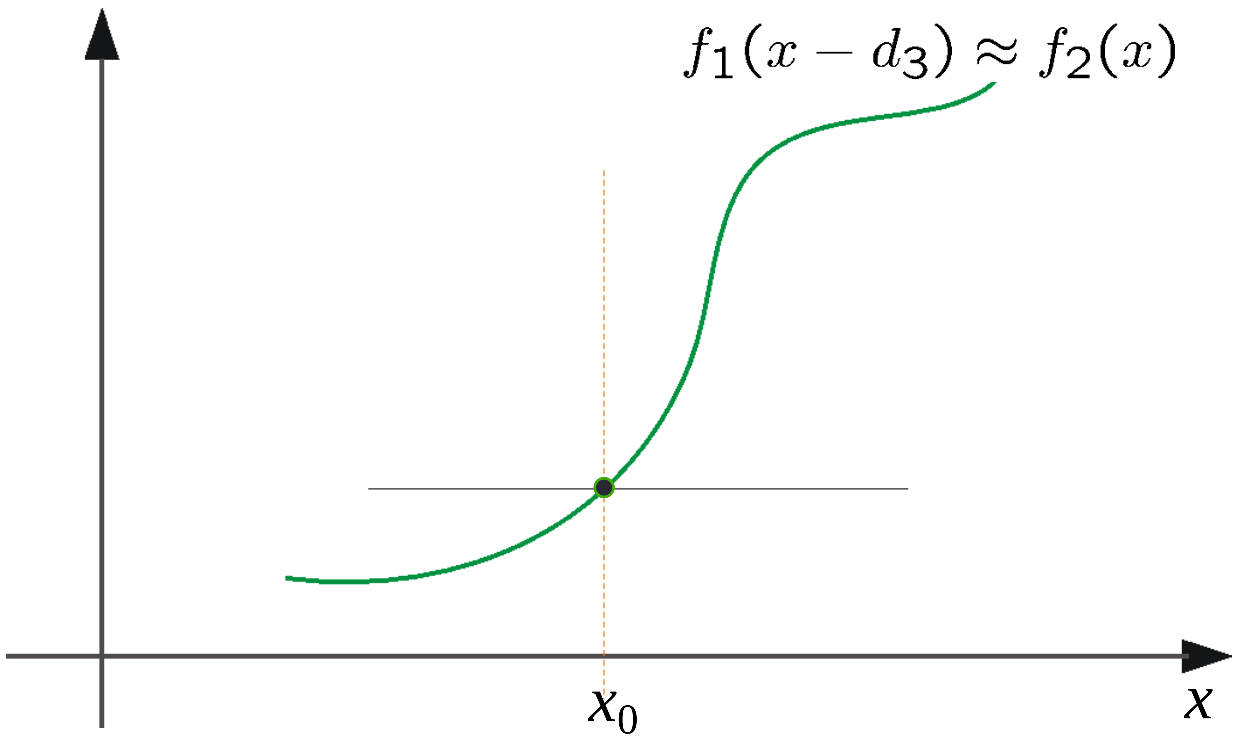
Optical Flow: Iterative Estimation



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) – but it is in MATLAB!
  - Often useful to low-pass filter the images before motion estimation (for better derivative estimation, and linear approximations to image intensity)

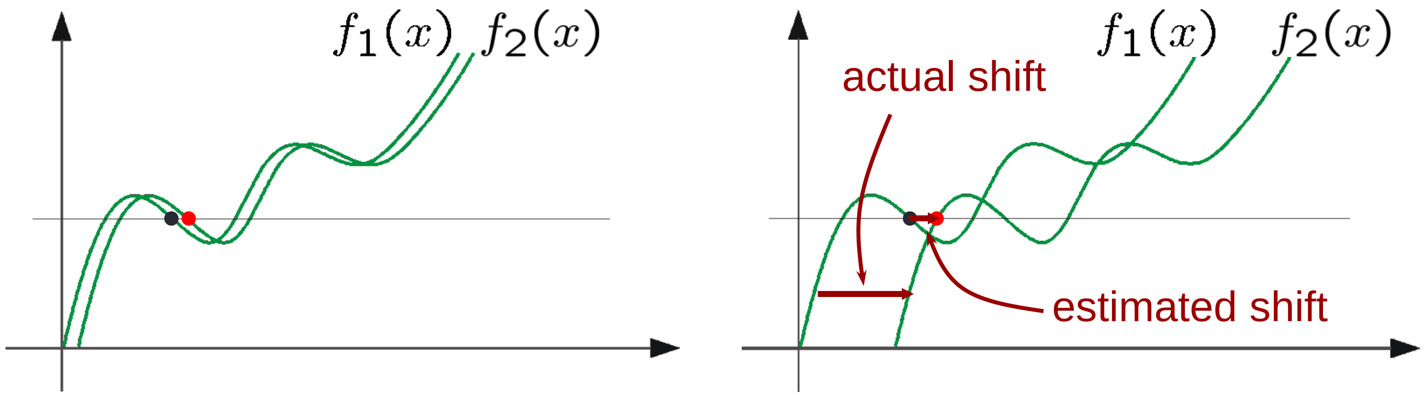
Revisiting the small motion assumption

- Is this motion small enough?
  - Probably not—it's much larger than one pixel
  - How might we solve this problem?

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?

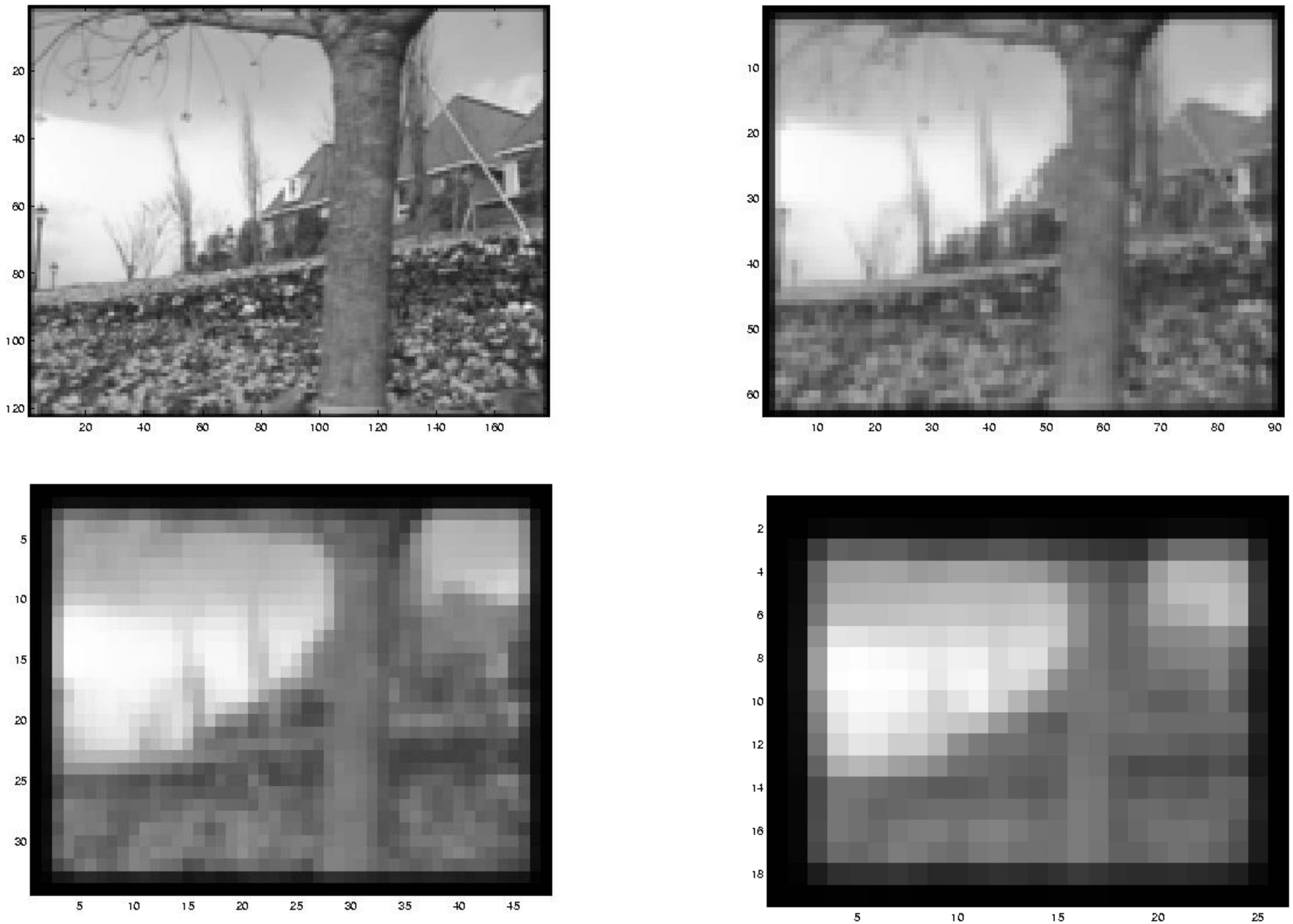


nearest match is correct (no aliasing)

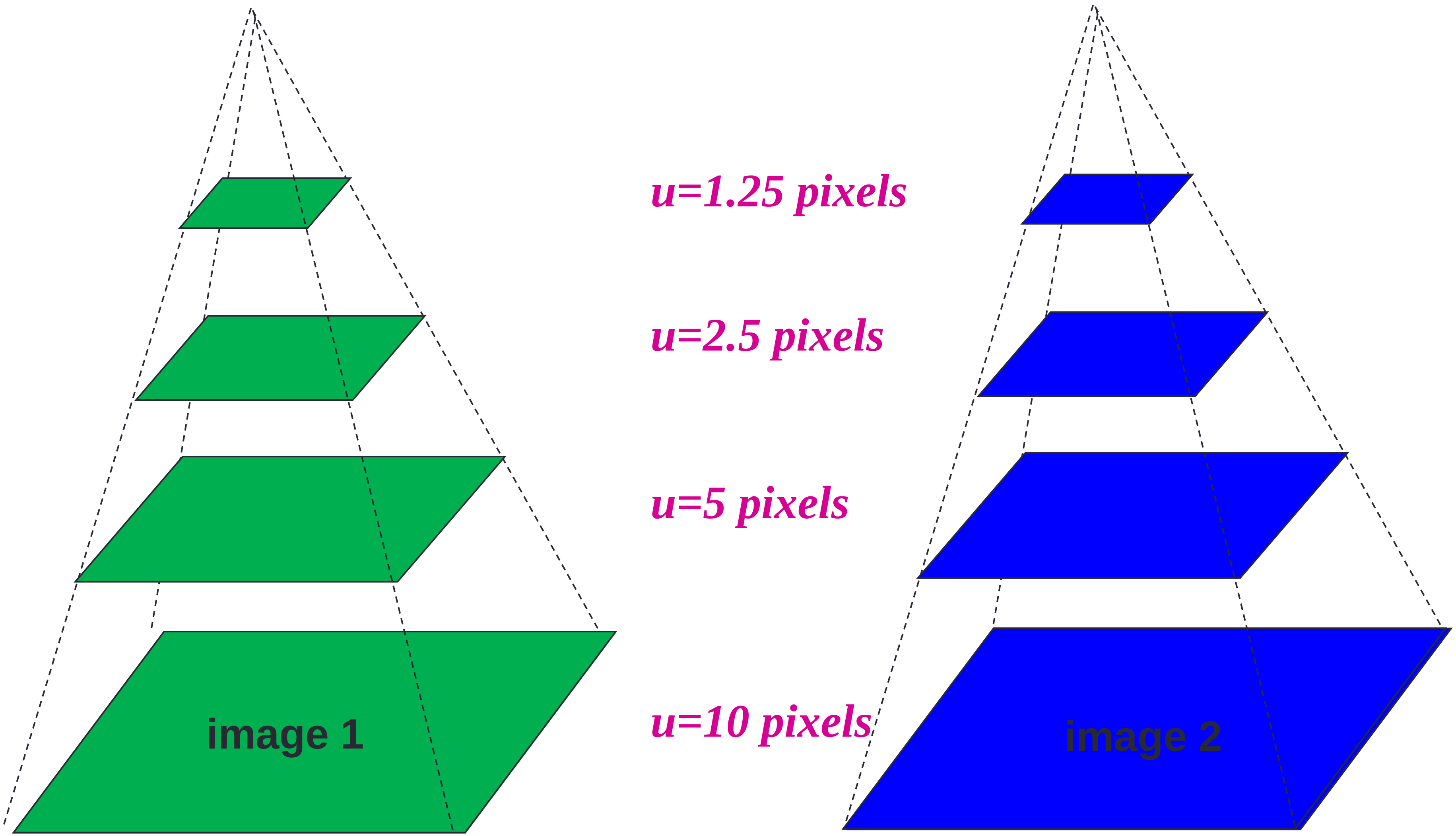
nearest match is incorrect (aliasing)

To overcome aliasing: coarse-to-fine estimation.

Reduce the resolution!



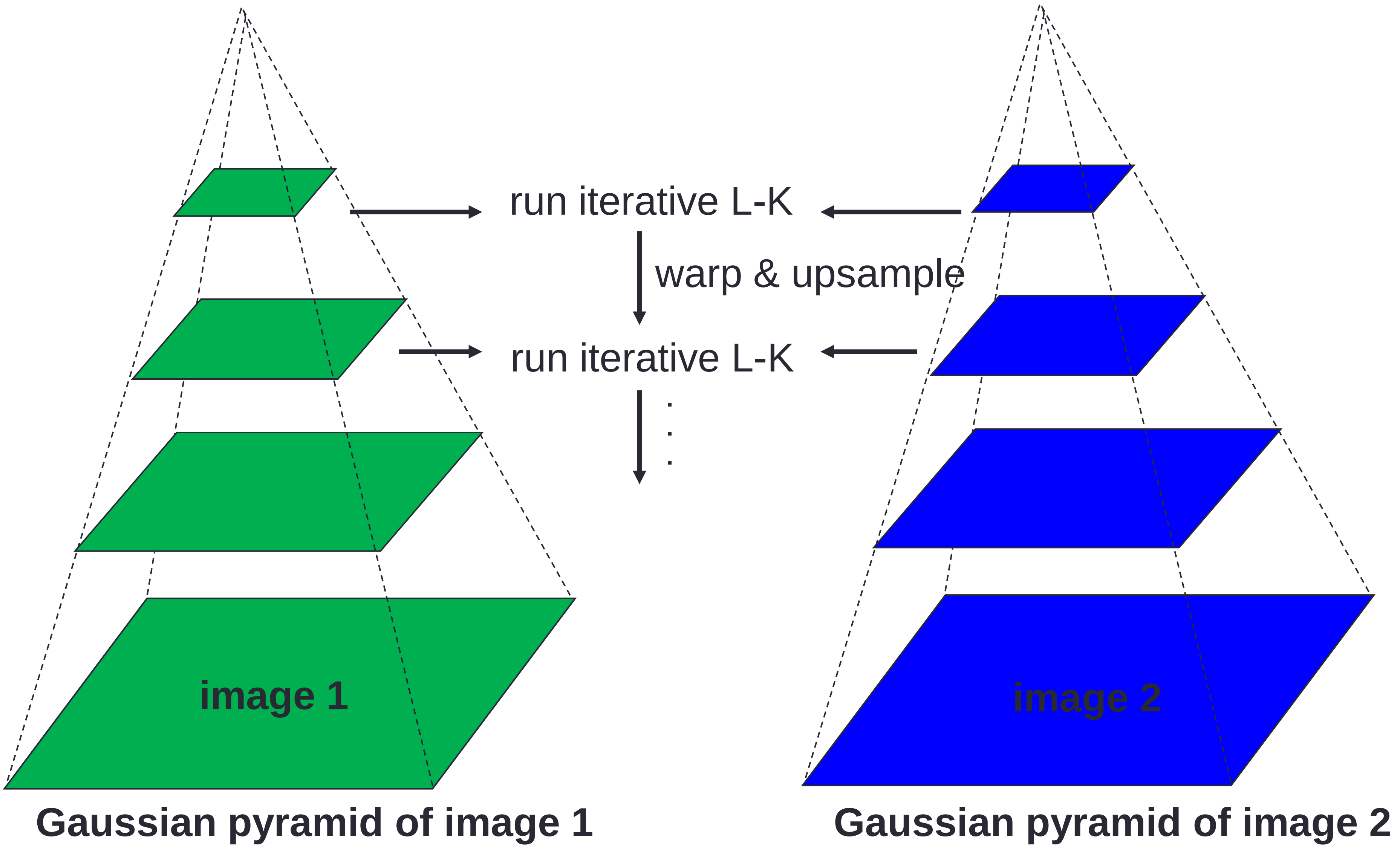
Coarse-to-fine optical flow estimation



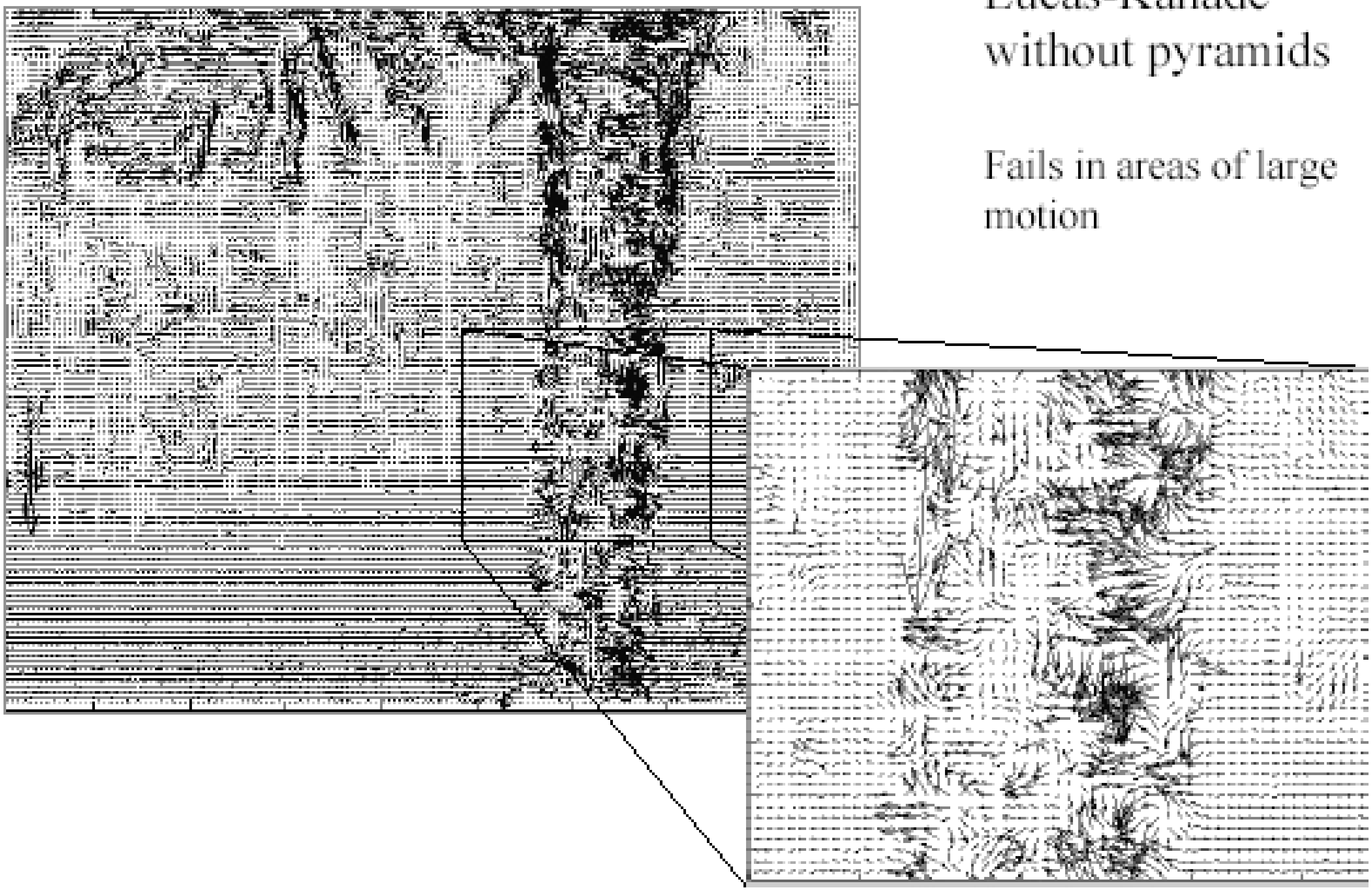
Gaussian pyramid of image 1

Gaussian pyramid of image 2

Coarse-to-fine optical flow estimation



Optical Flow Results

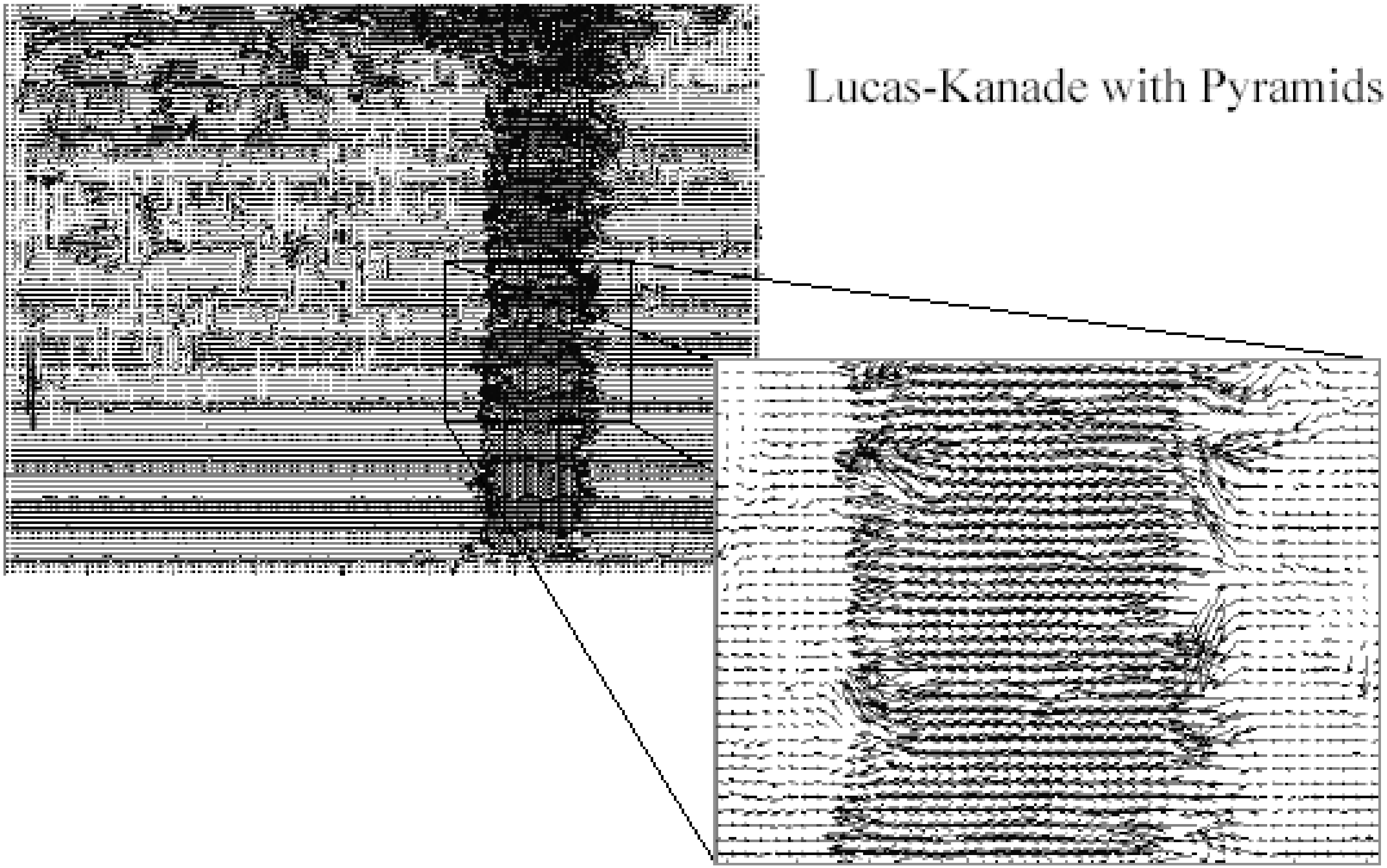


Lucas-Kanade without pyramids

Fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



Lucas-Kanade with Pyramids

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

State-of-the-art optical flow

Start with something similar to Lucas-Kanade

+ gradient constancy

+ energy minimization with smoothing term

+ region matching

+ keypoint matching (long-range)

Region-based +Pixel-based +Keypoint-based

Large displacement optical flow, Brox et al., CVPR 2009