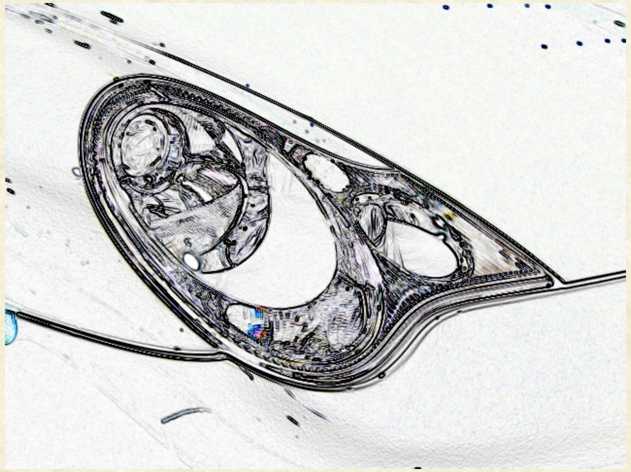
CS7.505: Computer Vision

Spring 2022: Optical Flow and Tracking

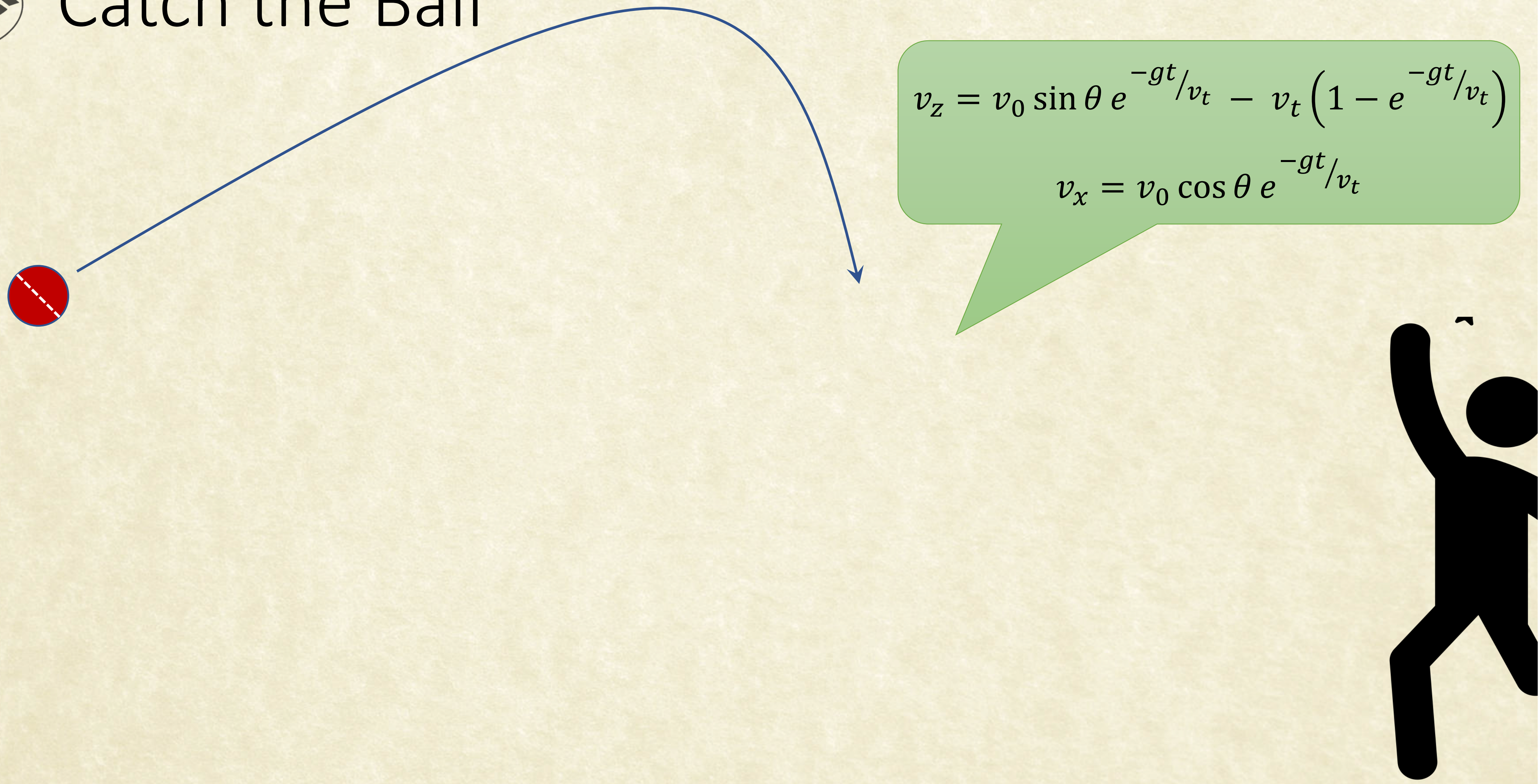


Anoop M. Namboodiri

Biometrics and Secure ID Lab, CVIT,

IIIT Hyderabad

Catch the Ball

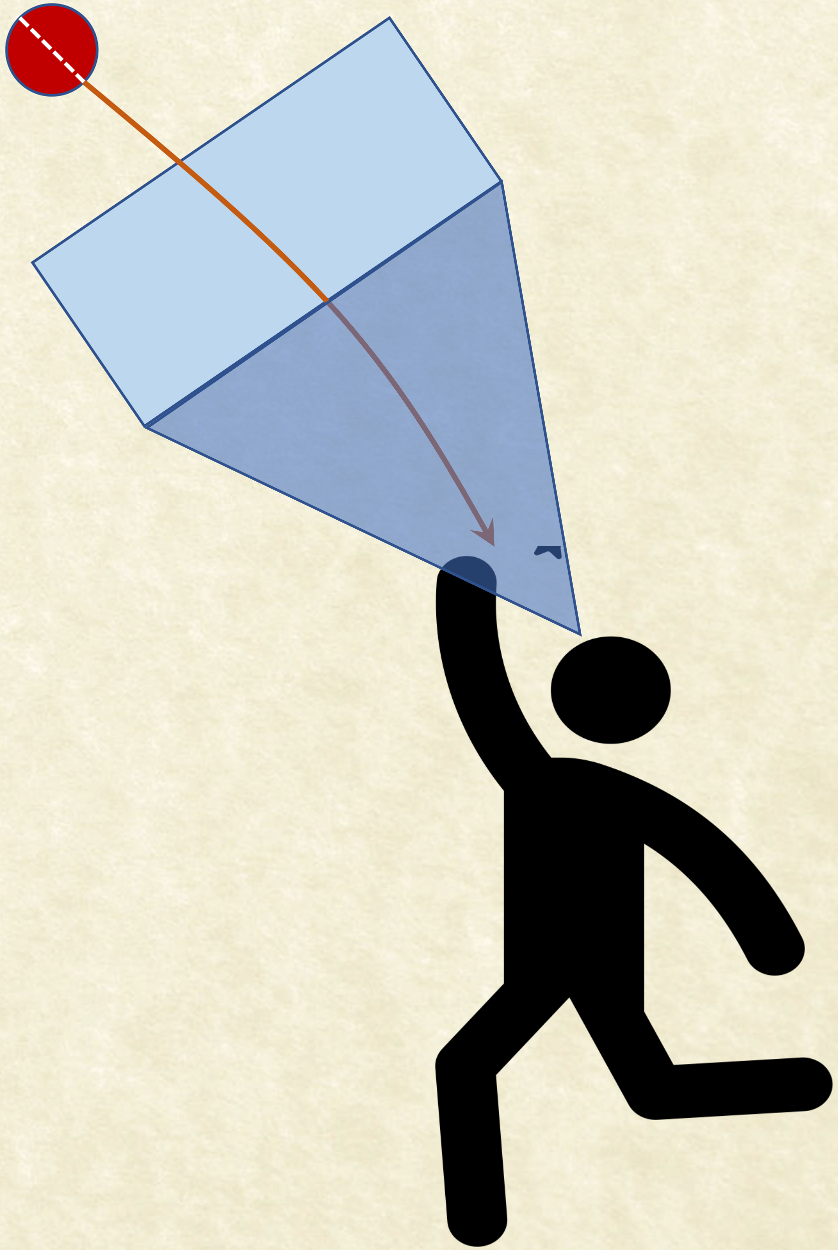


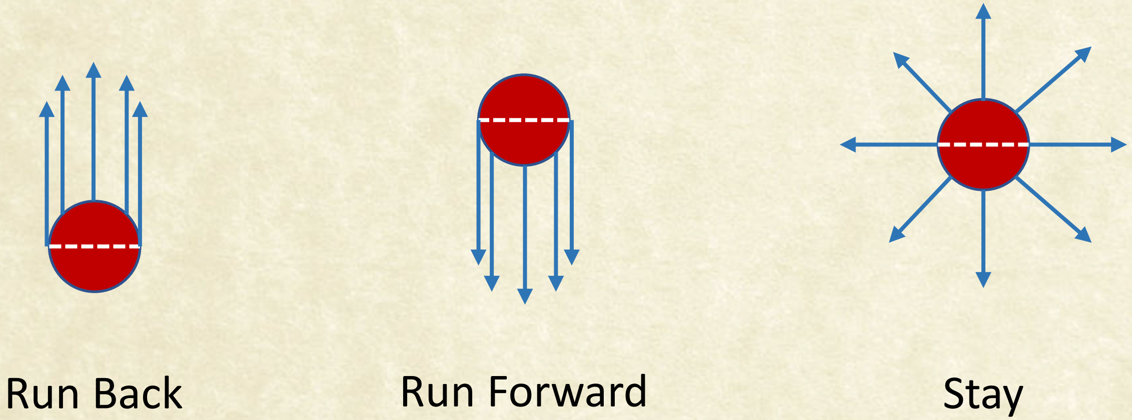
$$v_z = v_0 \sin\theta \, e^{-gt/v_t} - v_t\left(1 - e^{-gt/v_t}\right)$$

$$v_x = v_0 \cos\theta \, e^{-gt/v_t}$$

Catch the Ball

- We look for simple movement patterns
- Optical flow of the image we see

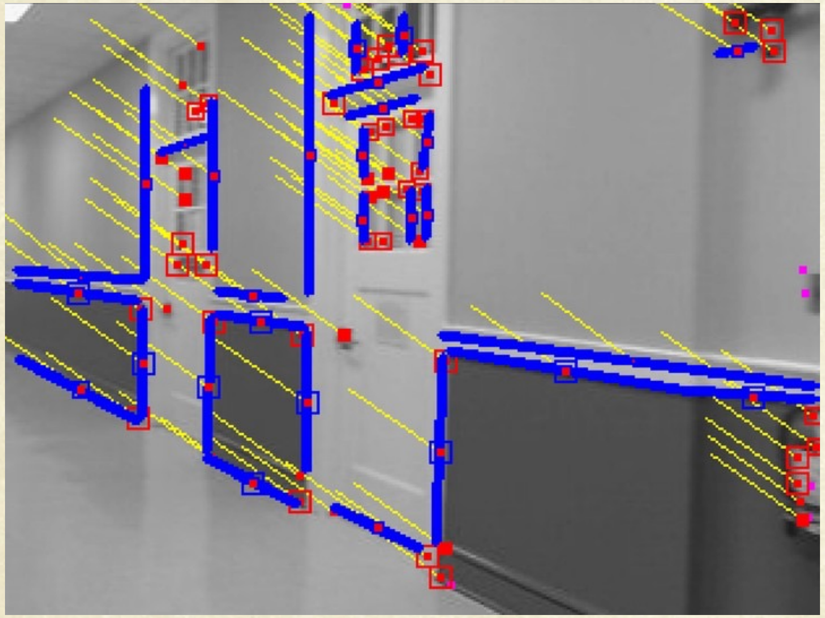




Feature Tracking vs. Optical Flow

Feature Tracking: Extract visual features (corners, textured areas) and “track” them over multiple frames (**sparse corr**.)

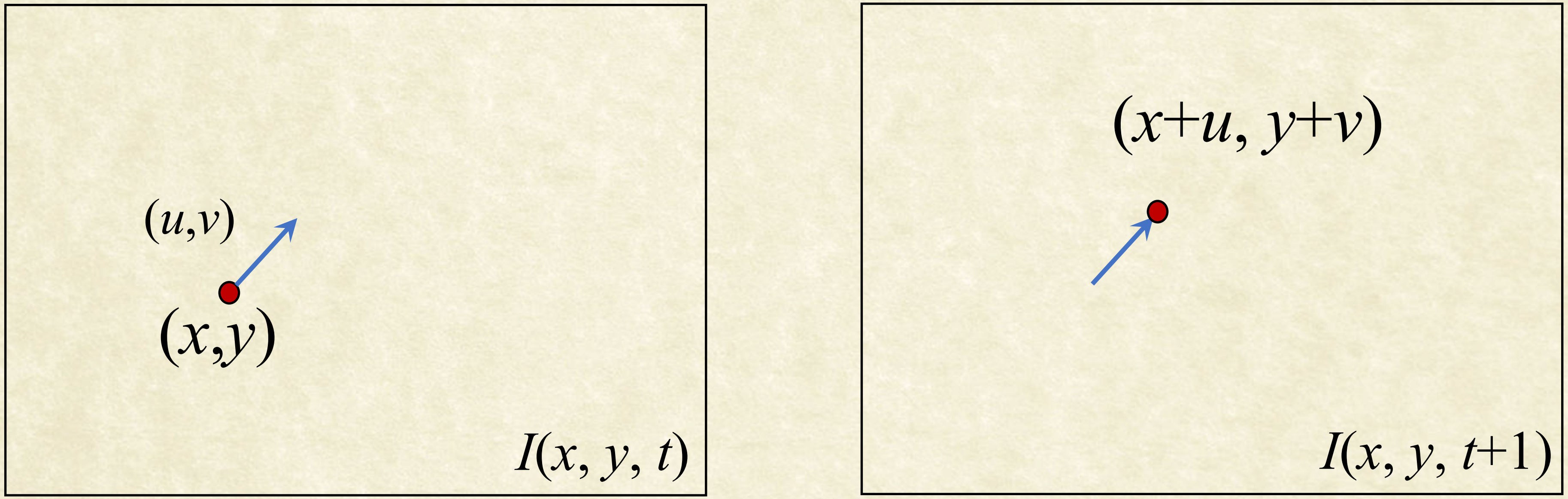
Optical Flow: Recover image motion at each pixel from spatio-temporal image brightness variations (**dense corr**.)



- Relationship to stereo matching, SFM
- Dense correspondence is challenging, often ill-posed

Optical Flow

- Brightness Constancy Assumption



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

A Linear Motion Model

Take Taylor expansion of $I(x+u, y+v, t+1)$ at (x,y,t) to linearize the right side:

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

Image derivative along x

Difference over frames

$$I(x+u, y+v, t+1) \approx I(x,y,t) + \boxed{I_x} \cdot u + I_y \cdot v + \boxed{I_t}$$

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

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla \mathrm{I} \cdot [\mathrm{u} \;\; \mathrm{v}]^{\mathrm{T}} + \mathrm{I_t} = 0$

Ambiguity of Motion

- Can we use this equation to recover image motion (u,v) at each pixel?
- How many equations and unknowns per pixel?

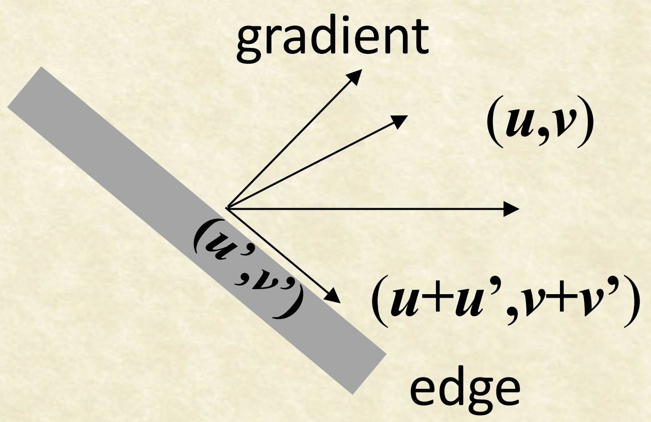
$$\nabla \mathrm{I} \cdot \left[\mathrm{u} \;\; \mathrm{v}\right]^{\mathrm{T}} + \mathrm{I}_{\mathrm{t}} = 0$$

- One equation (this is a scalar equation!), two unknowns (u,v)

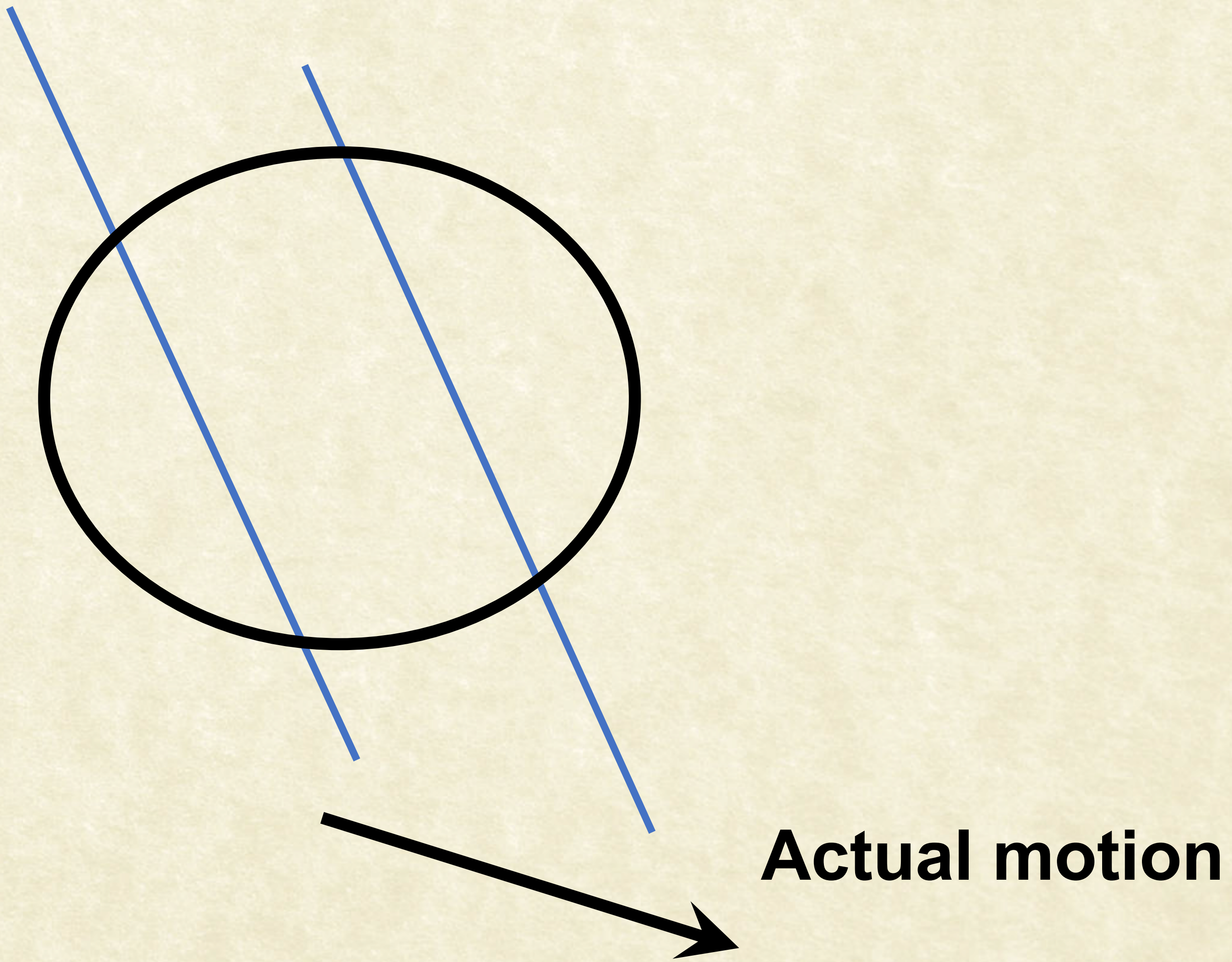
The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

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

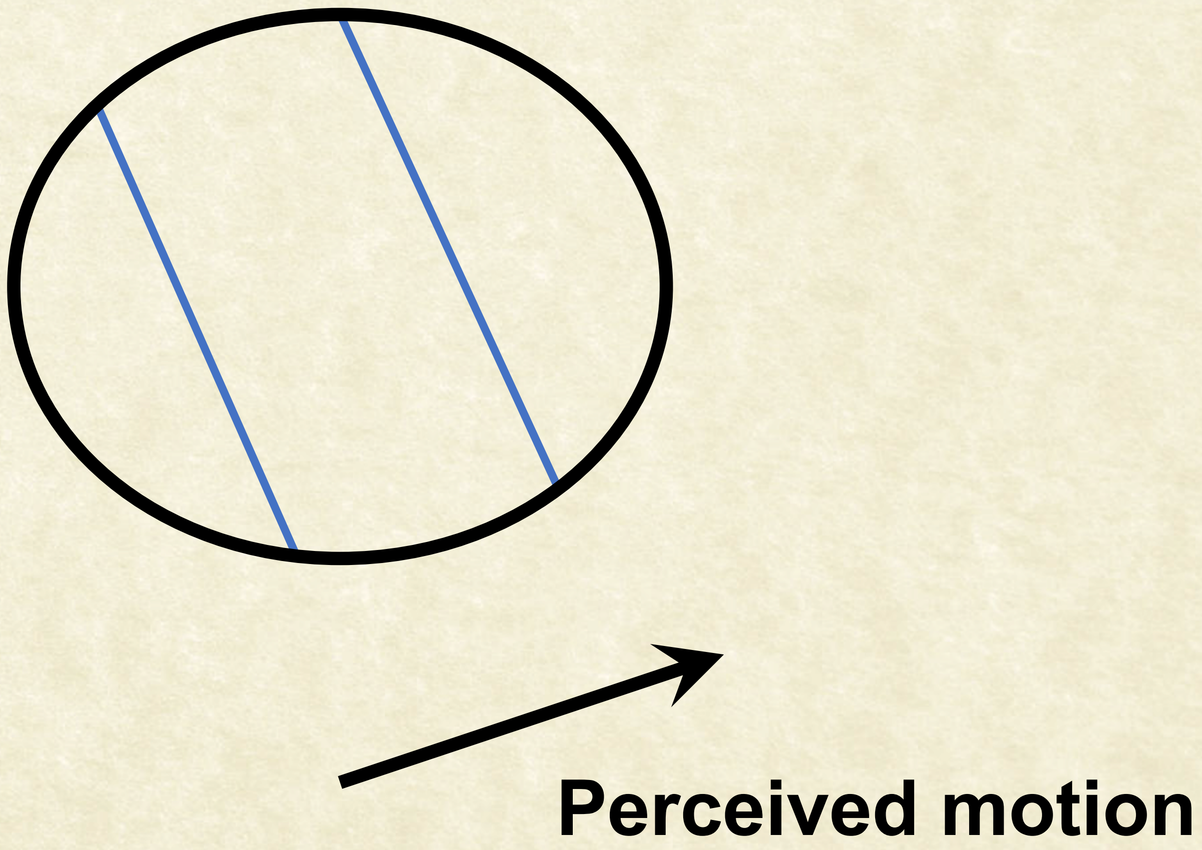
$$\nabla \mathrm{I} \cdot \left[\mathrm{u'} \;\; \mathrm{v'}\right]^{\mathrm{T}} = 0$$



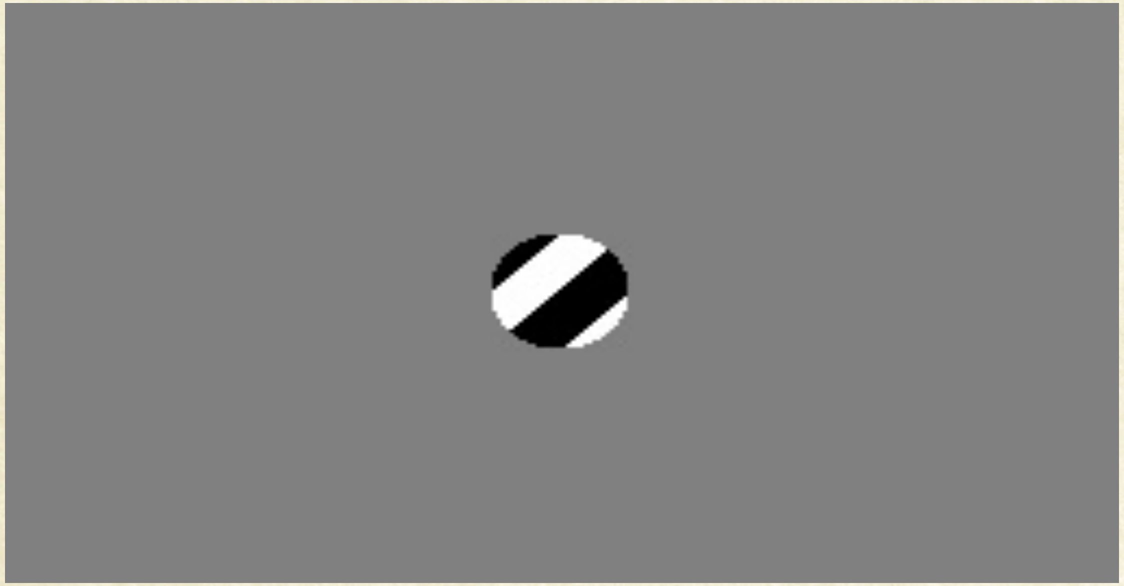
The Aperture Problem



The Aperture Problem



Motion Ambiguity



- Motion perpendicular to gradient direction is not discernible.
- Not just for vision !!

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

Solving the Ambiguity…

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

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

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Solving the Ambiguity…

- Least squares problem:

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

$$\underset{25\times2}{A} \; \underset{2\times1}{d} = \underset{25\times1}{b}$$

Matching Patches Across Images

- Overconstrained linear system

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

$$\underset{25\times2}{A} \; \underset{2\times1}{d} = \underset{25\times1}{b}$$

Least squares solution for *d* given by $(A^T A)\ d = A^T b$

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

The summations are over all pixels in the K x K window

Conditions for Solvability

Optimal (u, v) satisfies Lucas-Kanade equation

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

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

When is this solvable? i.e., what are good points to track?

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

Does this remind you of anything?

Criteria for Harris corner detector

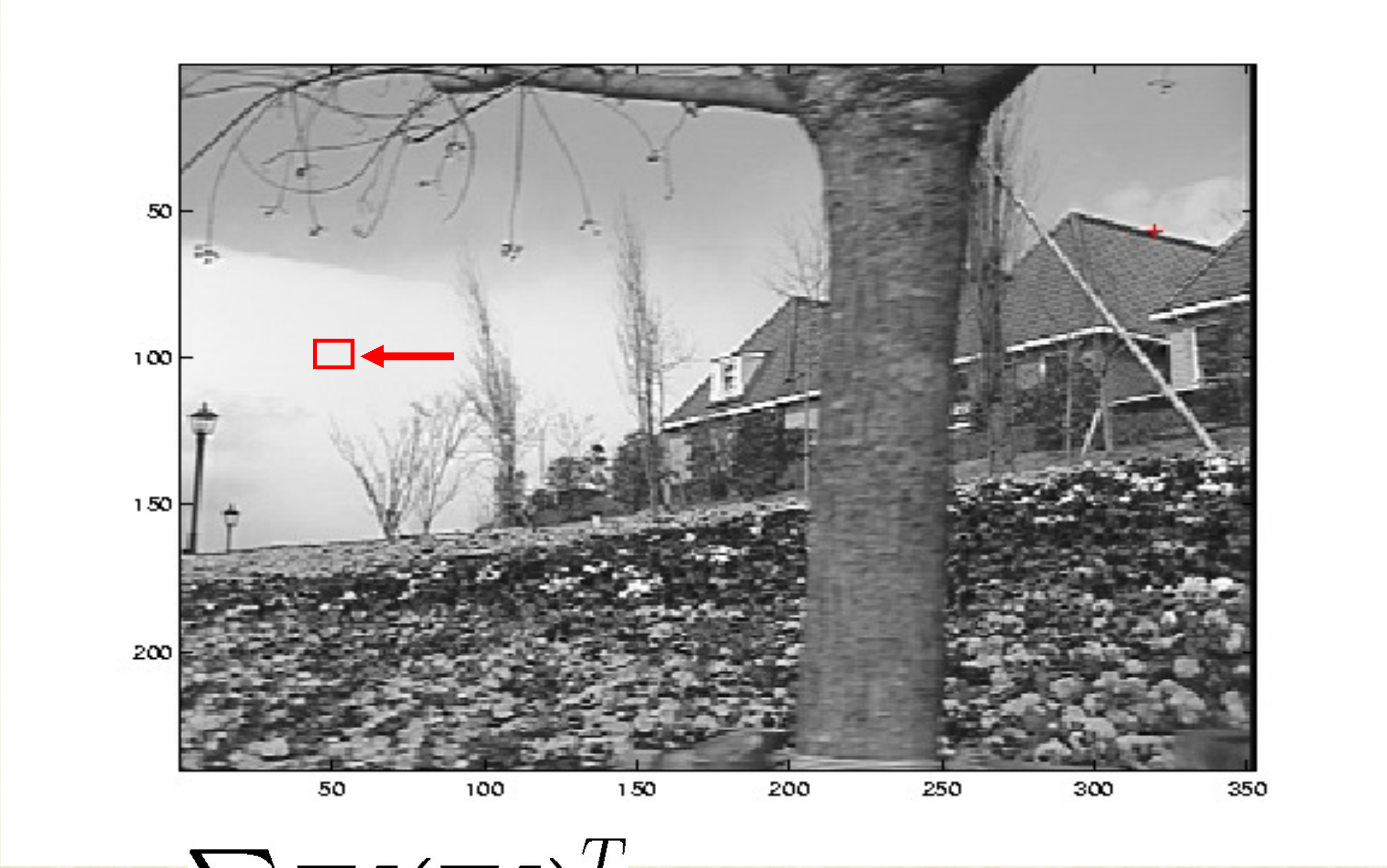
Trackable Features & Harris Corners

- $M = ATA$ is the second moment matrix! (Harris Corner detector)

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

- Eigenvectors and eigenvalues of $A^T A$ 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

Low-Texture Region

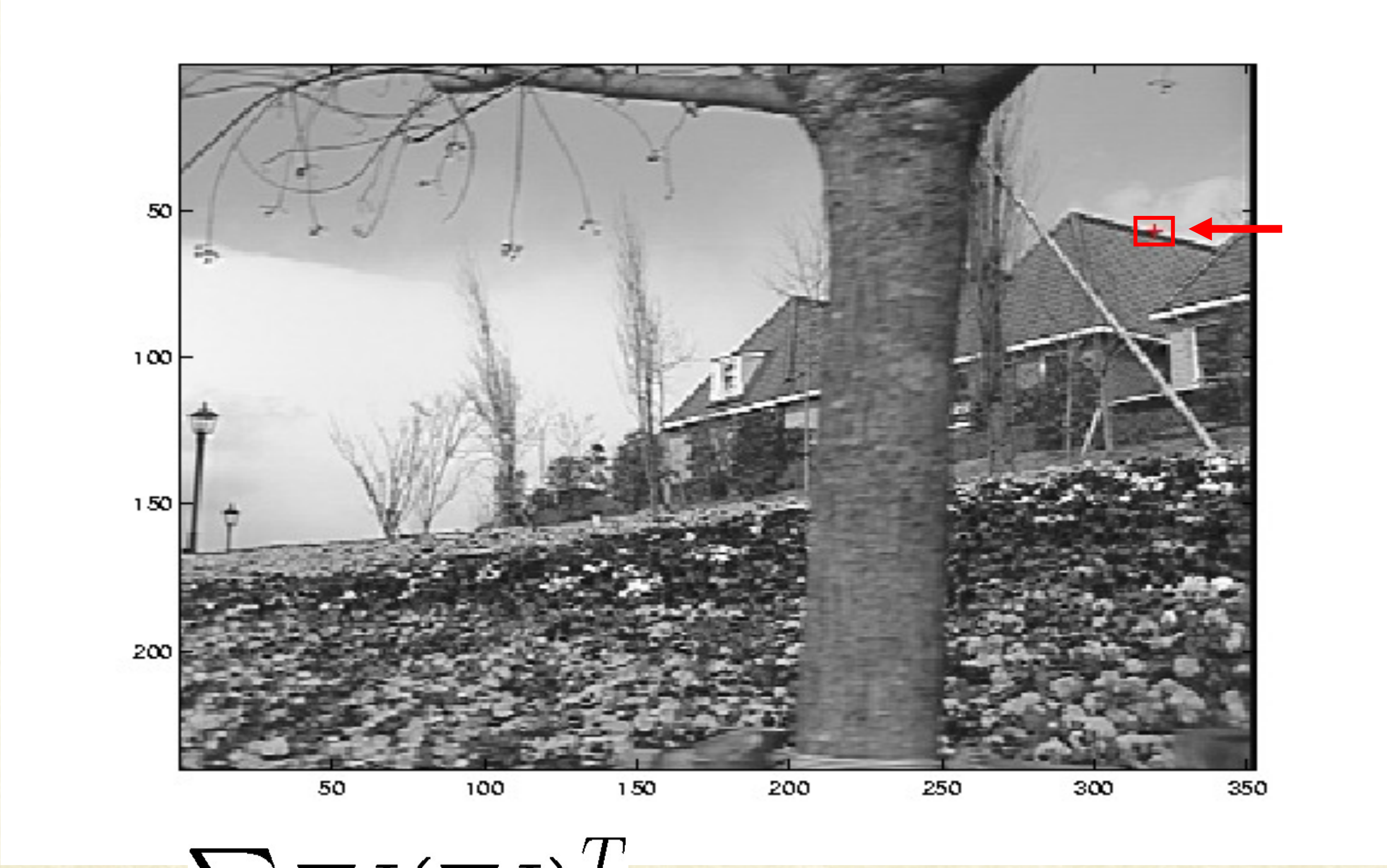


$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small λ_1, small λ_2

Edge

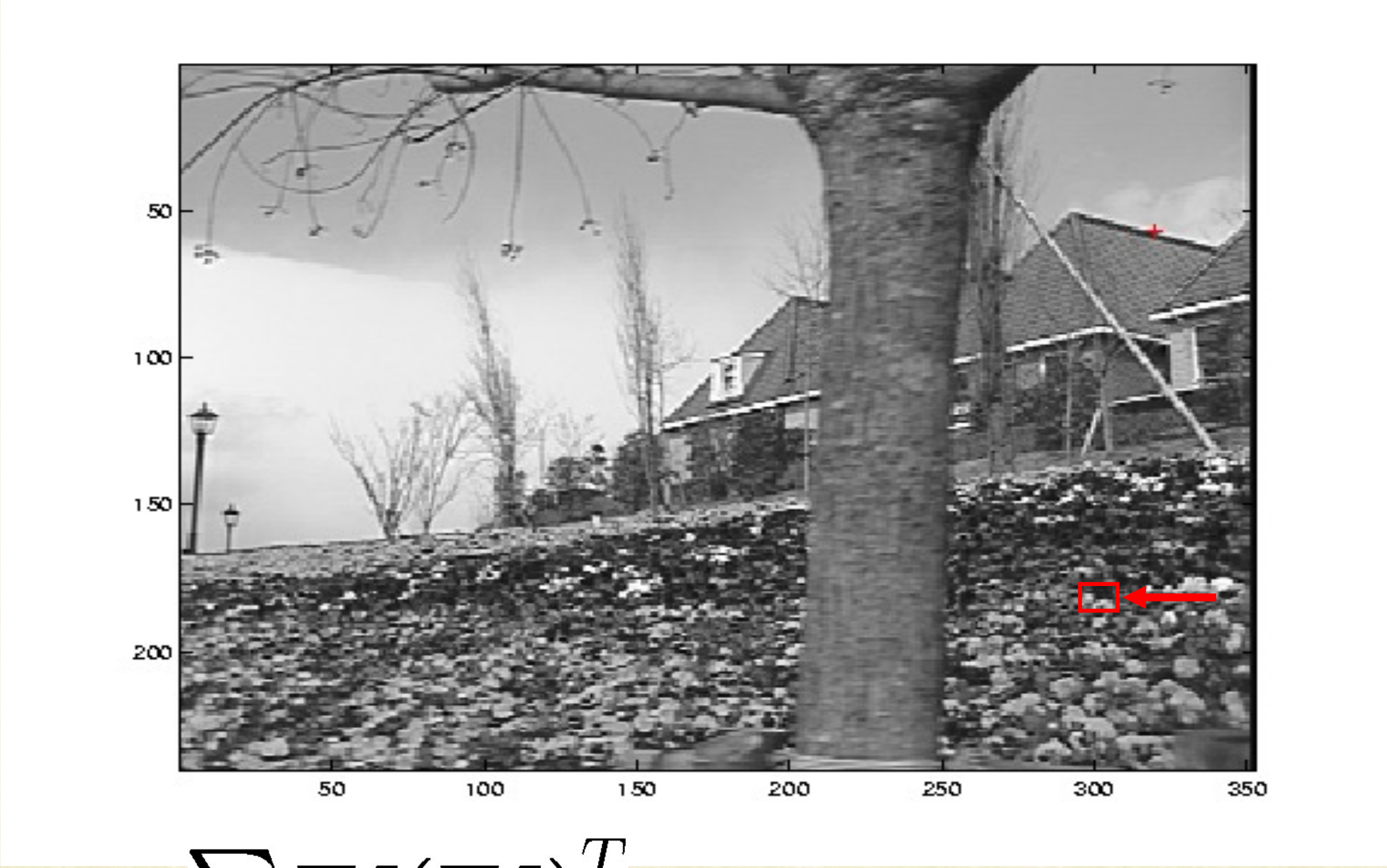


$$\sum \nabla I (\nabla I)^T$$

– gradients very large or very small

– large λ_1, small λ_2

High-Texture Region

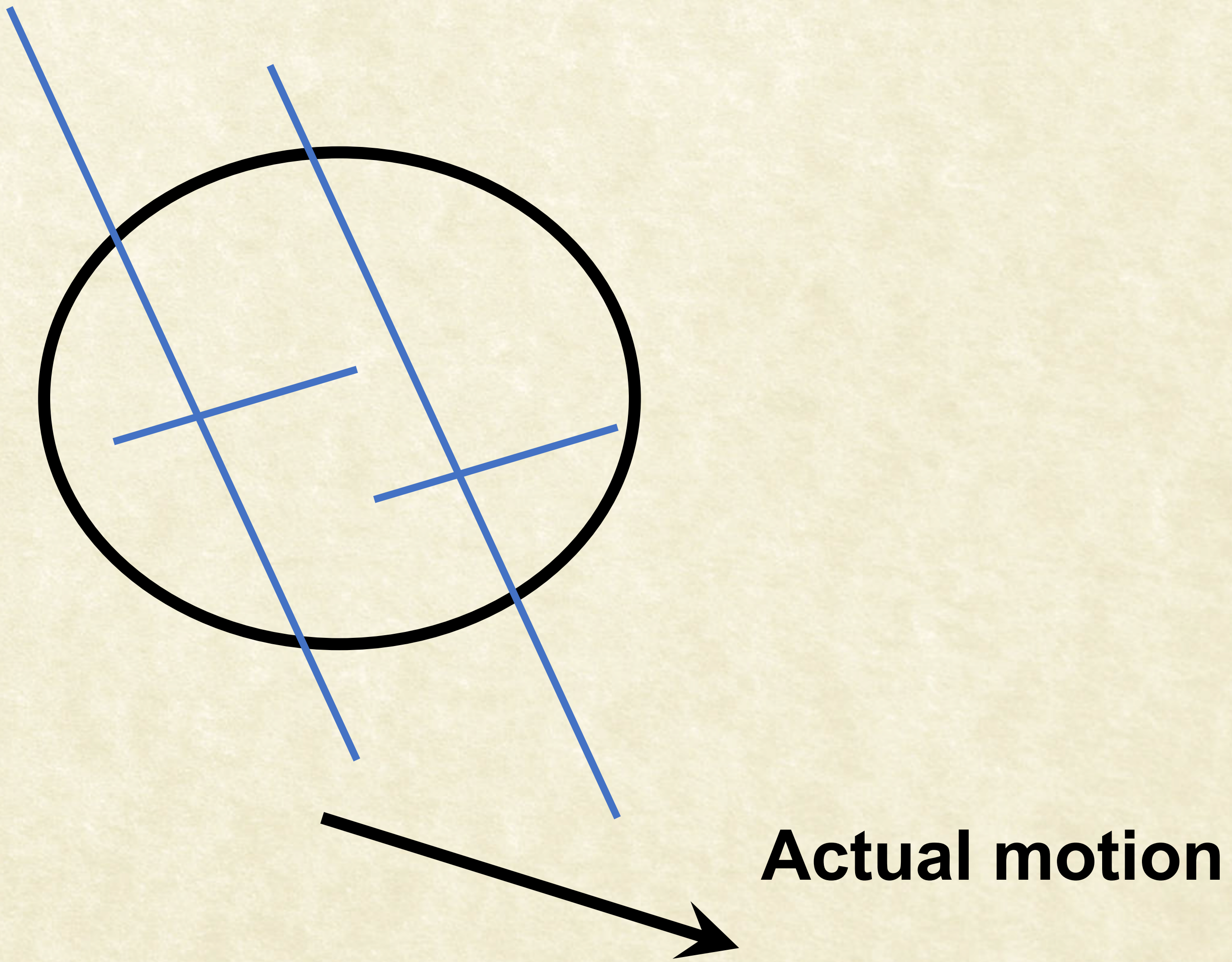


$$\sum \nabla I (\nabla I)^T$$

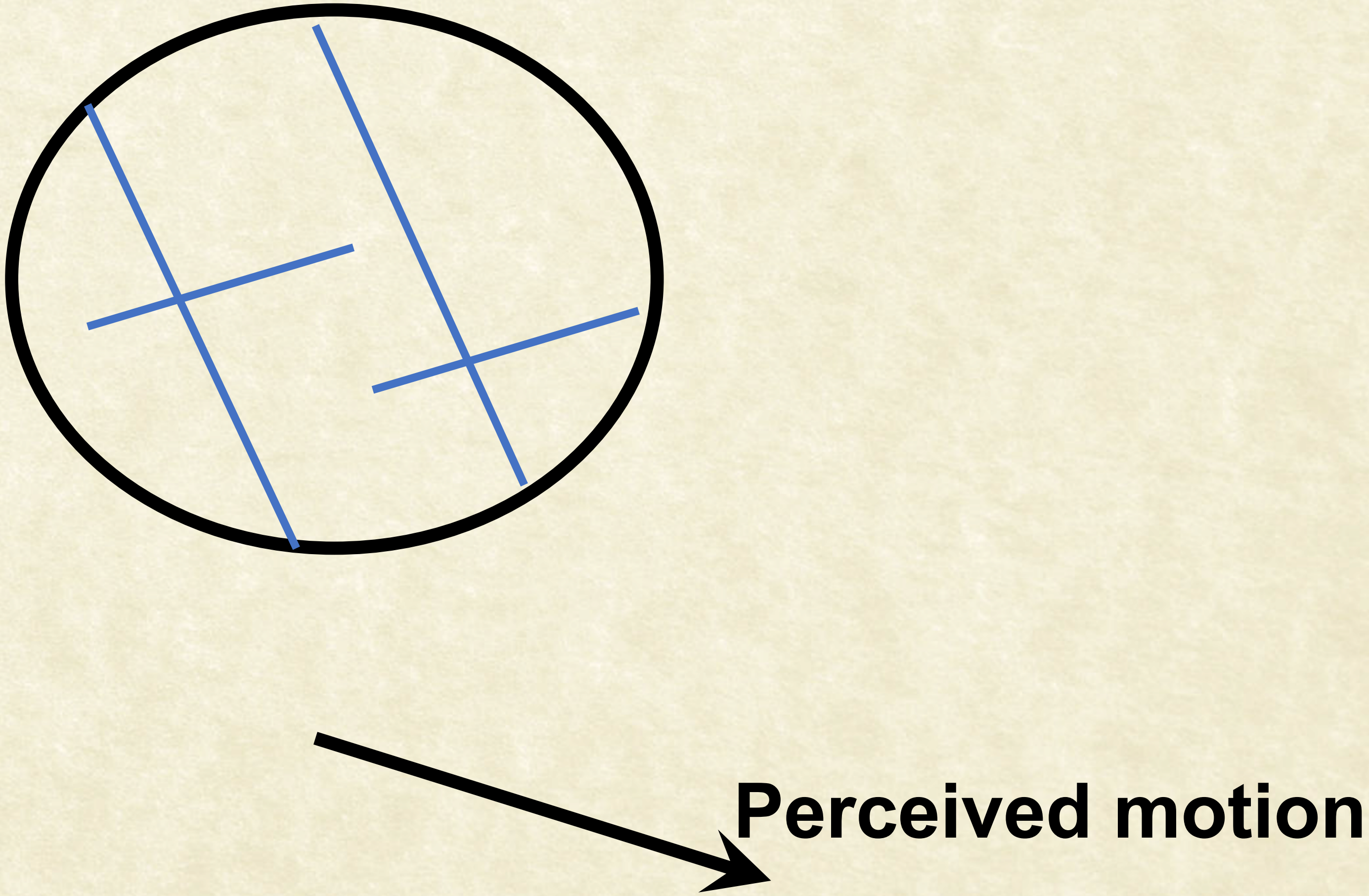
– gradients are different, large magnitudes

– large λ_1, large λ_2

The aperture problem resolved



The aperture problem resolved



Dealing with Larger Movements: Iterative Refinement

1. Initialize (x',y') = (x,y)
2. Compute (u,v) by

Original (x,y) position

$$I_t = I(x', y', t+1) - I(x, y, t)$$

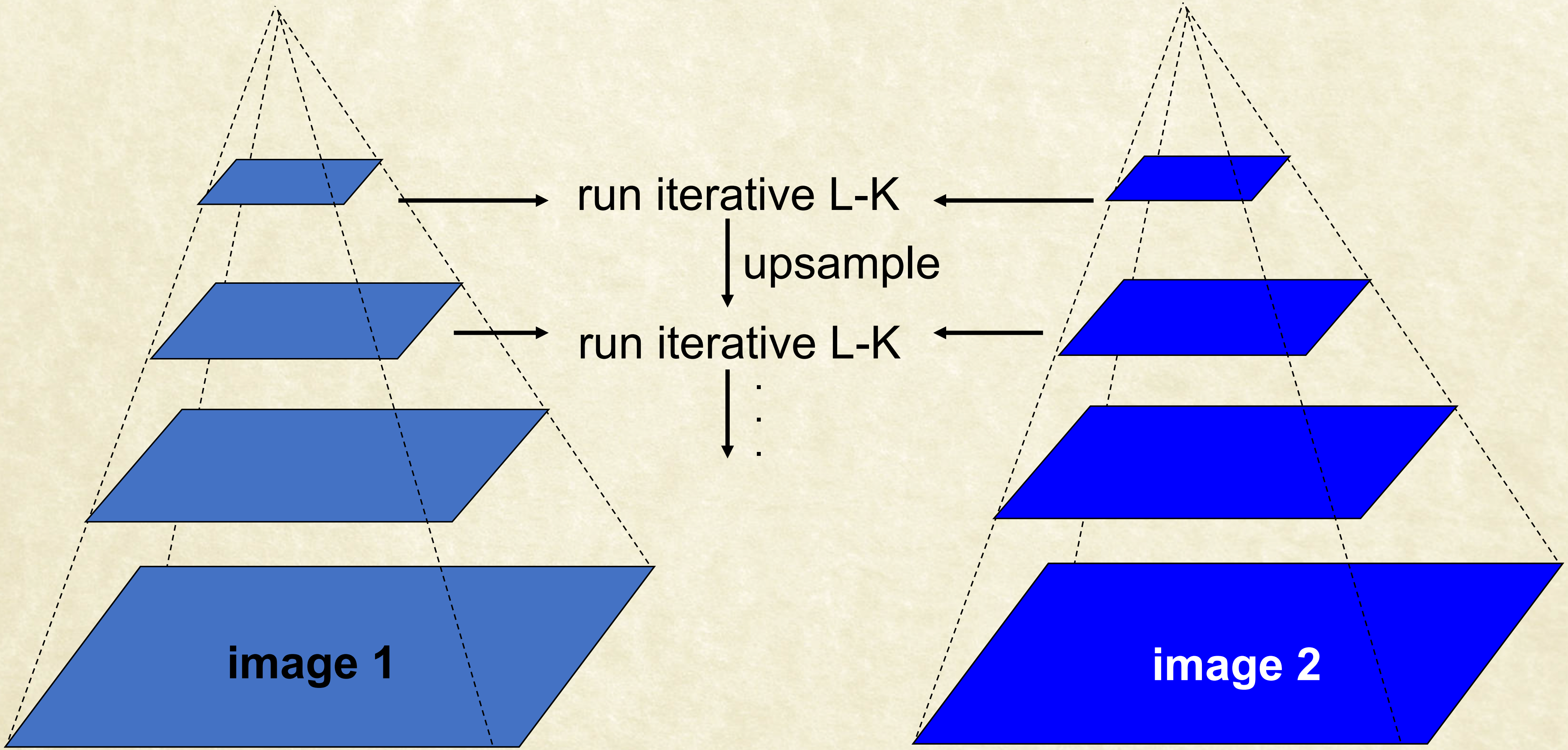
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

2nd moment matrix for feature patch in first image

displacement

3. Shift window by (u, v): $x' = x' + u;\ y' = y' + v;$
4. Recalculate I_t
5. Repeat steps 2-4 until small change
   - Use interpolation for subpixel values

Dealing with Larger Movements: Coarse-to-Fine Registration



Gaussian pyramid of image 1 (t)

Gaussian pyramid of image 2 (t+1)

Shi-Tomasi Feature Tracker

- Find good features using eigenvalues of 2nd-moment matrix (e.g., Harris detector or threshold on the smallest eigenvalue)
  - Key idea: "good" features to track are the ones whose motion can be estimated reliably

- Track from frame to frame with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement

- Check consistency of tracks by *affine* registration to the first observed instance of the feature
  - Affine model is more accurate for larger displacements
  - Comparing to the first frame helps to minimize drift

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Tracking Example

Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Summary of KLT Tracking

- Find a good point to track (harris corner)

- Use intensity second moment matrix and difference across frames to find displacement

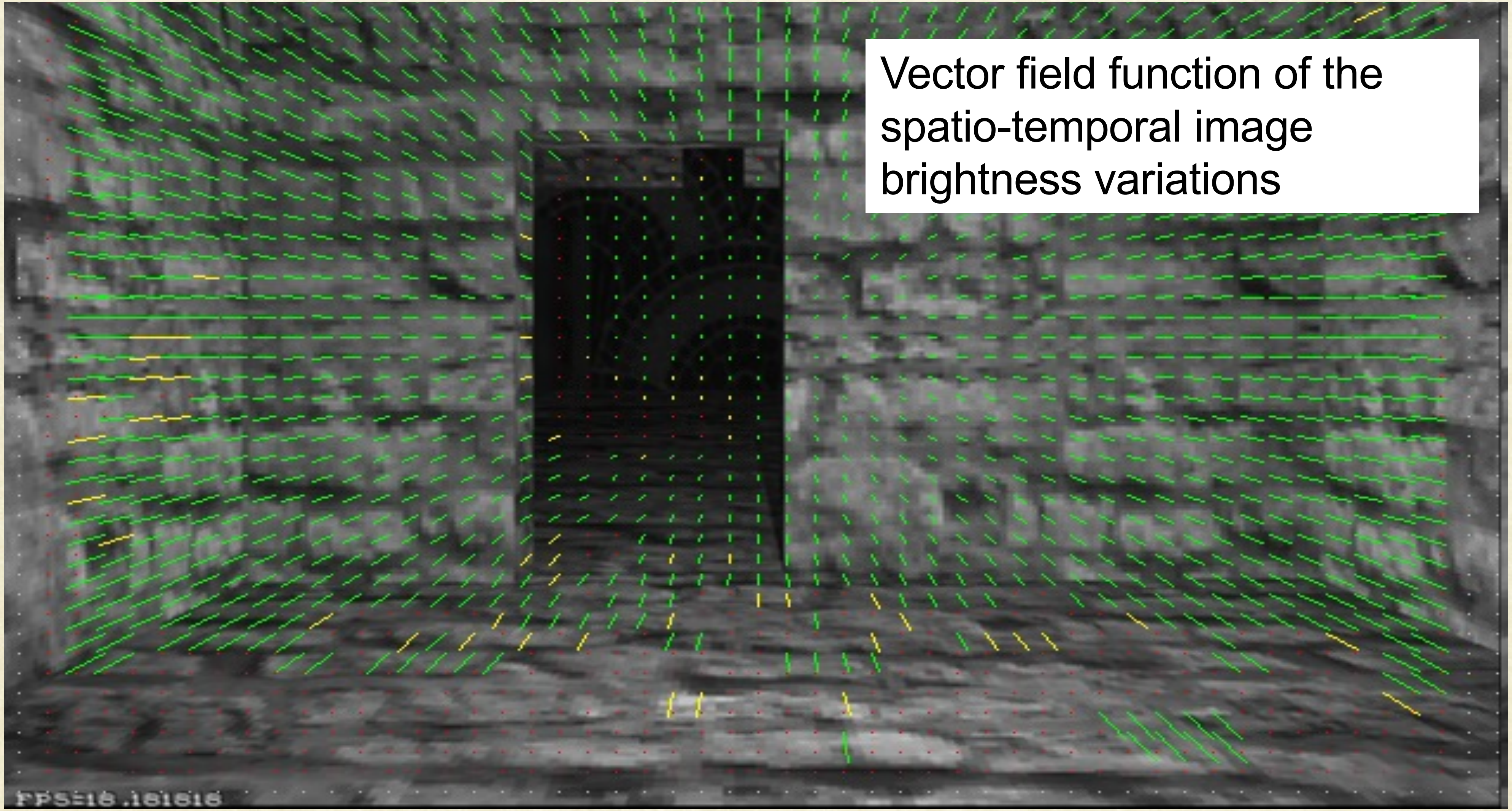
- Iterate and use coarse-to-fine search to deal with larger movements

- When creating long tracks, check appearance of registered patch against appearance of initial patch to find points that have drifted

Implementation Issues

- Window size
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it's slower)
  - 15x15 to 31x31 seems typical
- Weighting the window
  - Common to apply weights so that center matters more (e.g., with Gaussian)

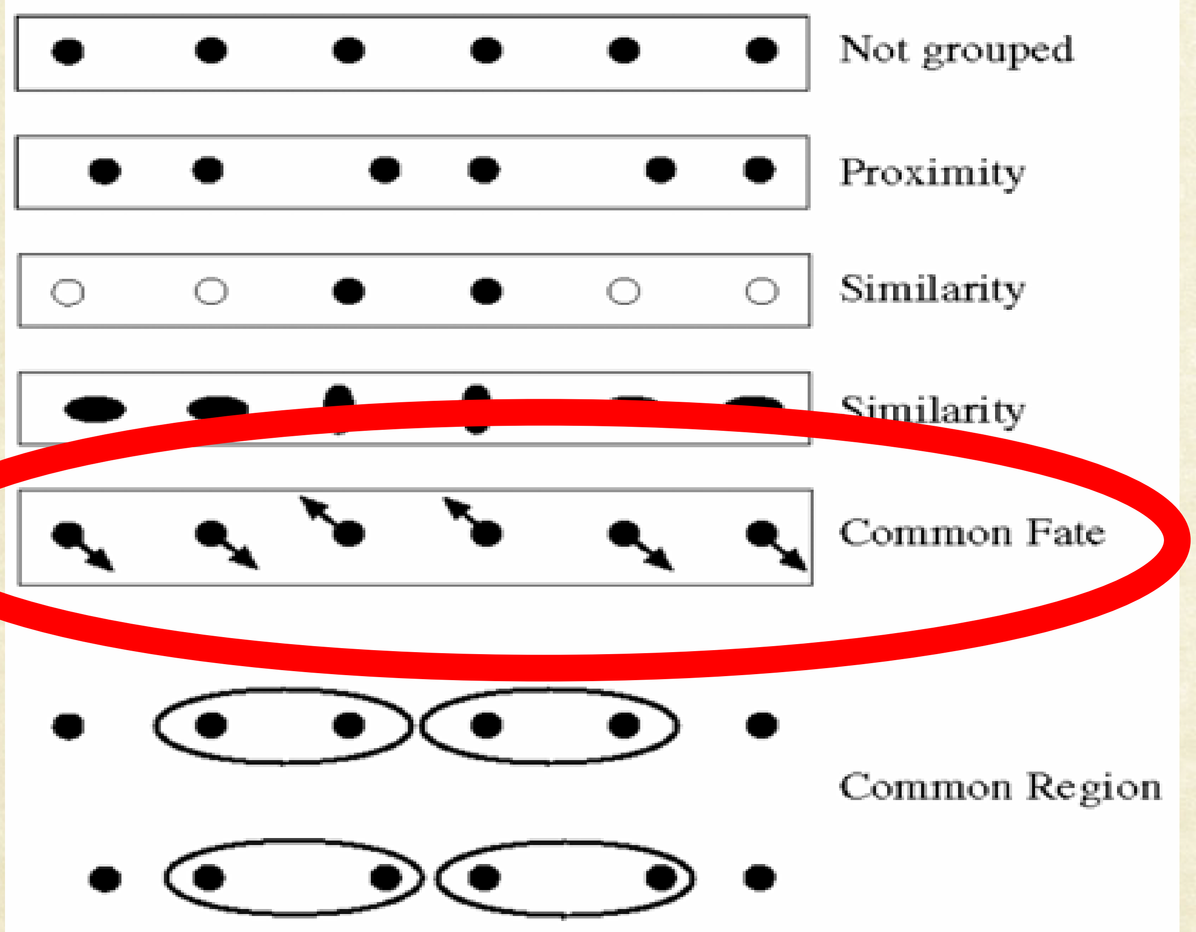
Optical Flow

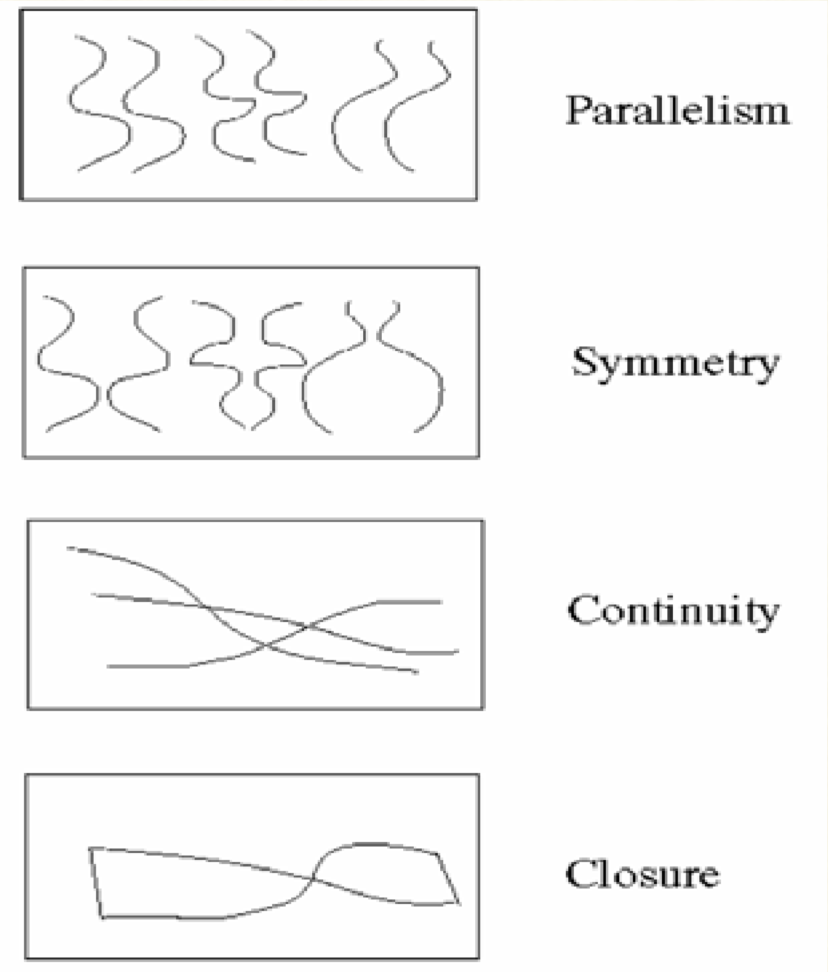


Picture courtesy of Selim Temizer - Learning and Intelligent Systems (LIS) Group, MIT

Motion and Perceptual Organization

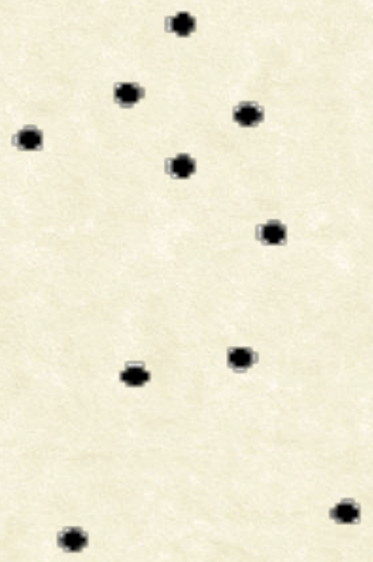
- Sometimes, motion is the only cue





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

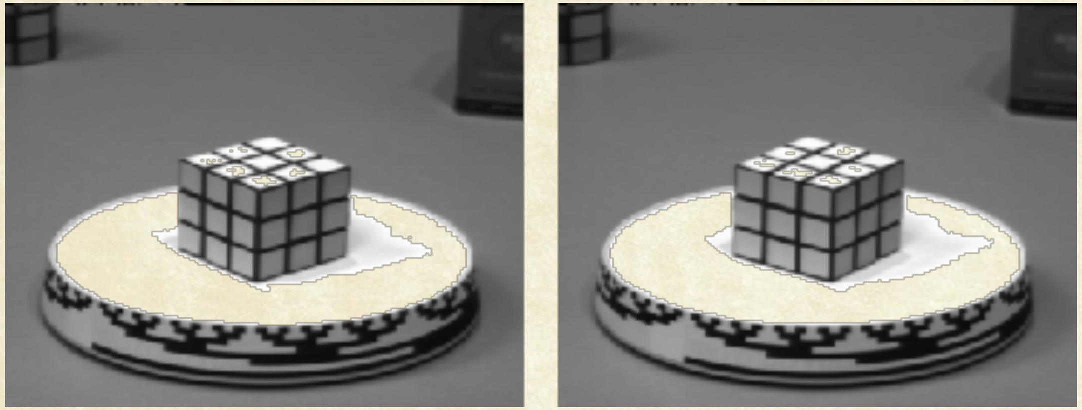
G. Johansson, "Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

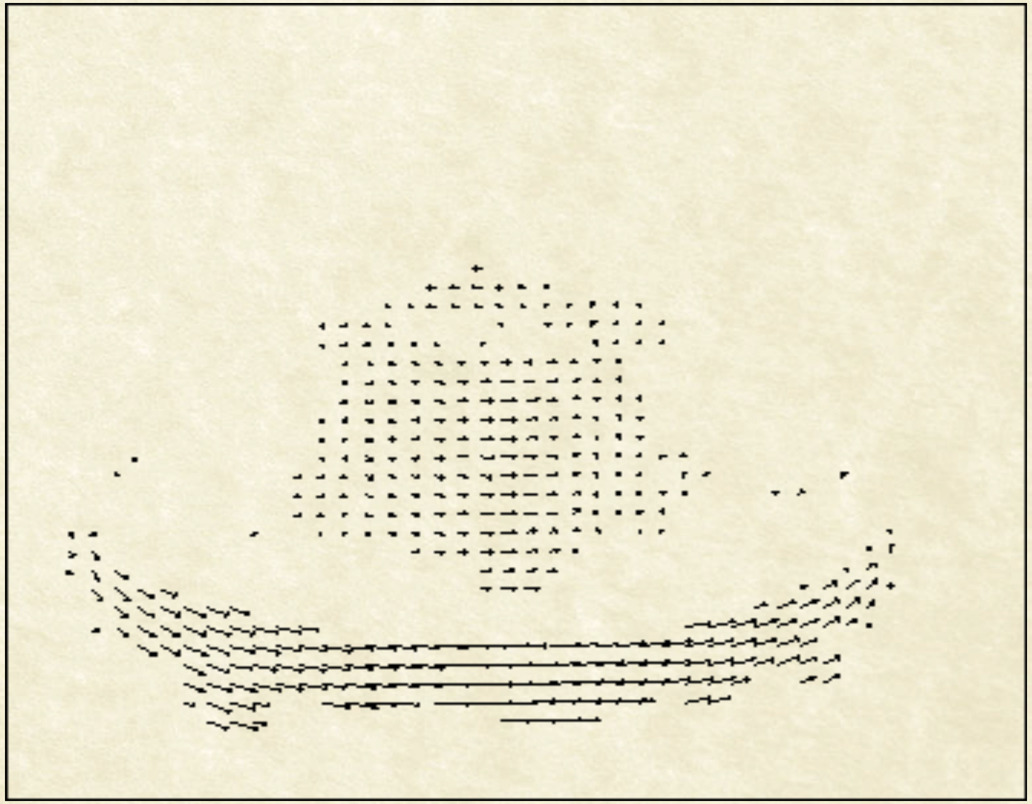
Uses of Motion Estimation

- Estimating 3D structure
- Segmenting objects based on motion cues
- Learning and tracking dynamical models
- Recognizing events and activities
- Improving video quality (motion stabilization)
- Video Compression (MPEG-4)

Motion Field

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





What would the motion field of a non-rotating ball moving towards the camera look like?

Optical Flow vs Motion Field

- 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
  - Think of a uniform rotating sphere under fixed lighting vs. a stationary sphere under moving illumination

Lucas-Kanade Optical Flow

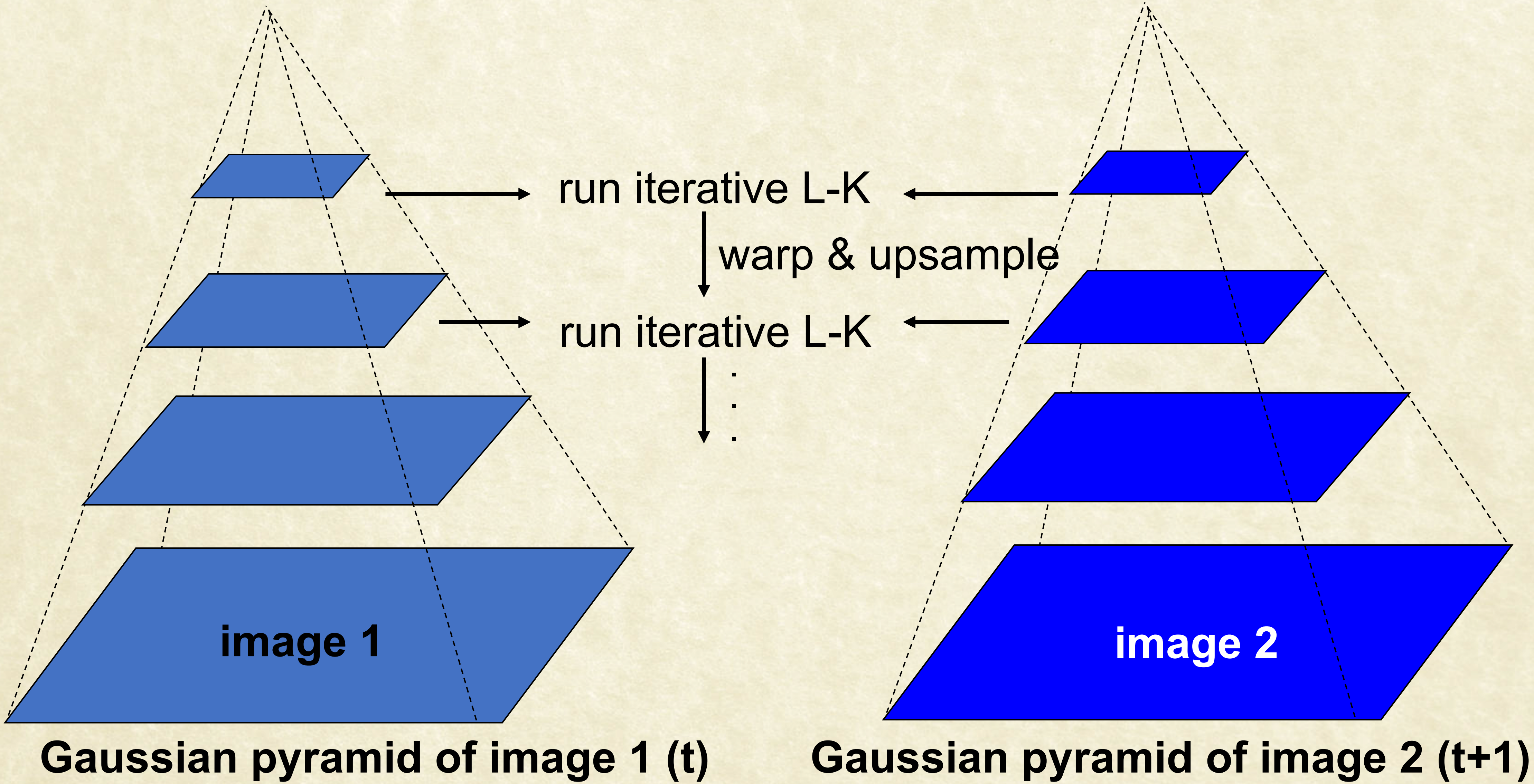
- Same as Lucas-Kanade feature tracking, but for each pixel
  - As we saw, works better for textured pixels
- Operations can be done one frame at a time, rather than pixel by pixel
  - Efficient

Iterative Refinement

- Iterative Lukas-Kanade Algorithm
  1. Estimate displacement at each pixel by solving Lucas-Kanade equations
  2. Warp I(t) towards I(t+1) using the estimated flow field
     - - Basically, just interpolation
  3. Repeat until convergence

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

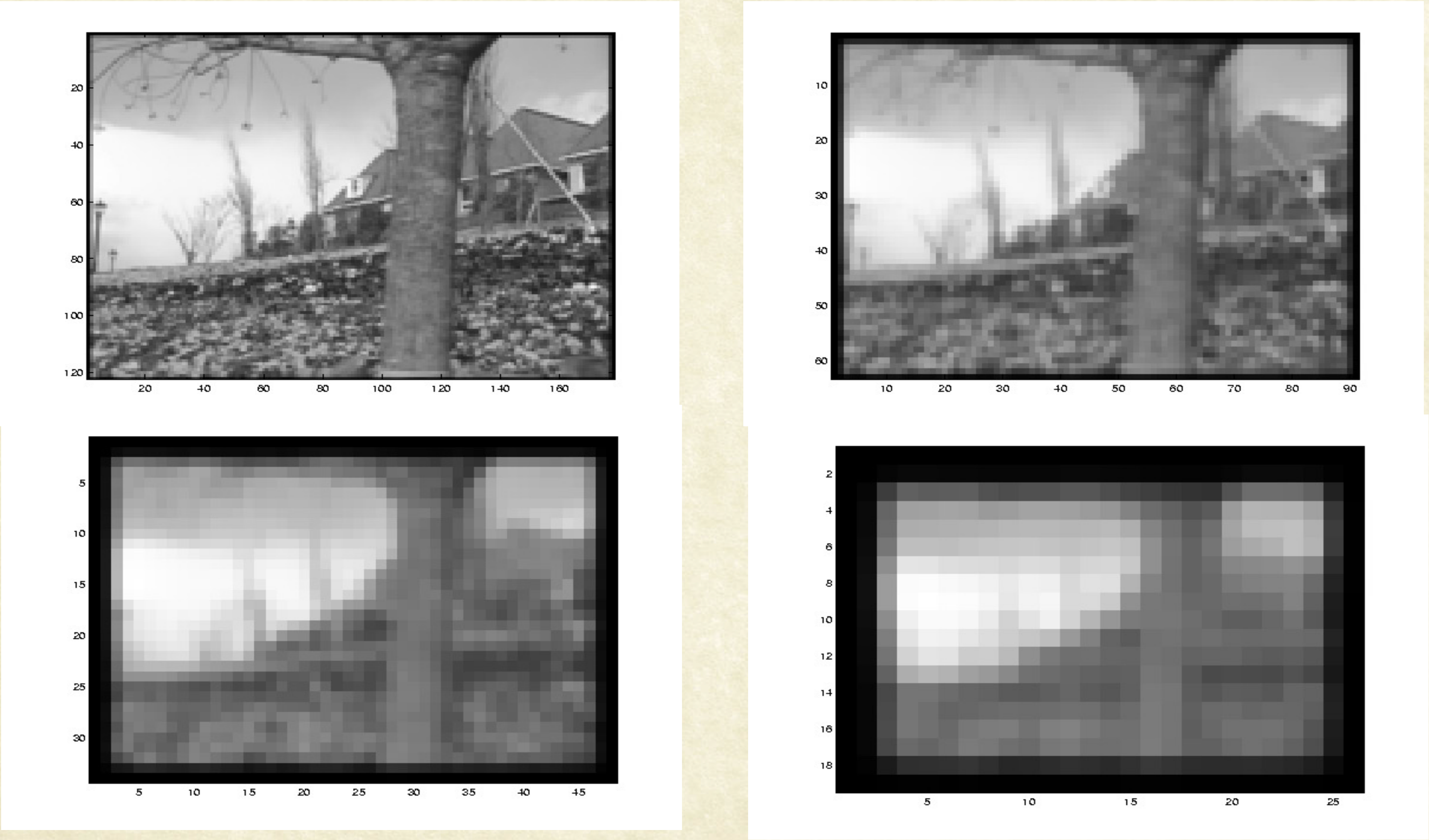
Coarse-to-fine Optical Flow Estimation



Example

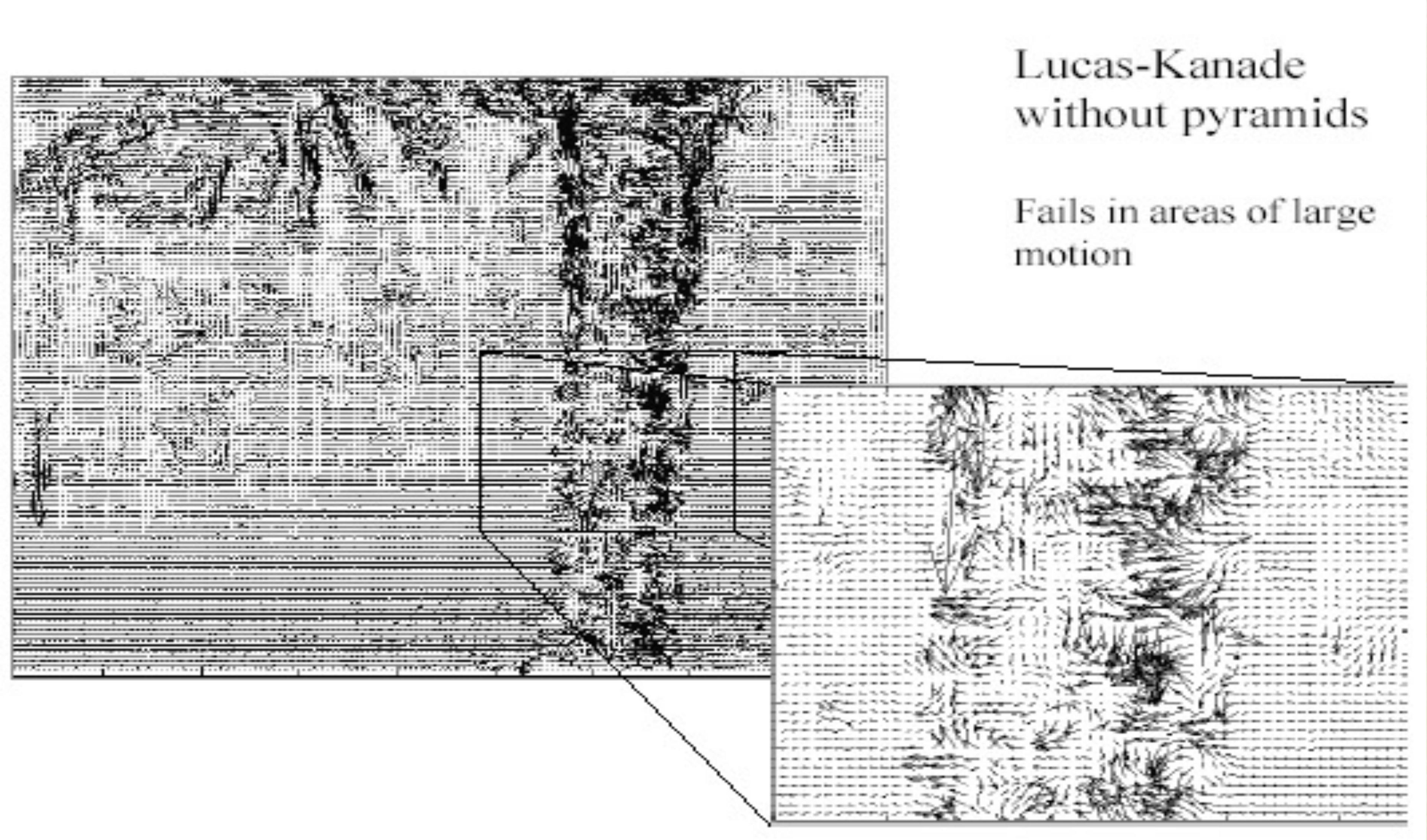
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Multi-resolution registration



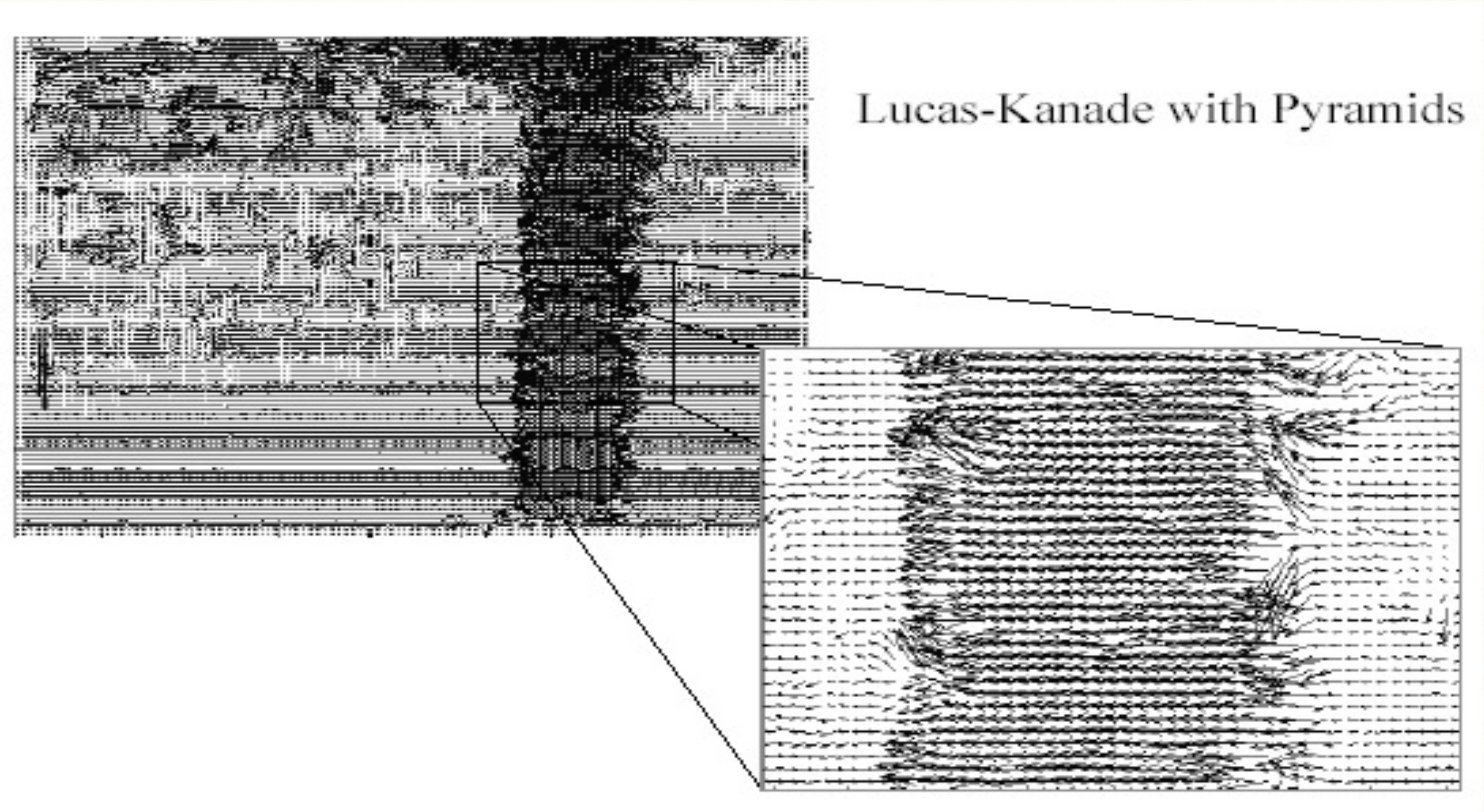
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Errors in Lucas-Kanade

- The motion is large
  - Possible Fix: Keypoint matching
- A point does not move like its neighbors
  - Possible Fix: Region-based matching
- Brightness constancy does not hold
  - Possible Fix: Gradient constancy

Improvements in Optical Flow

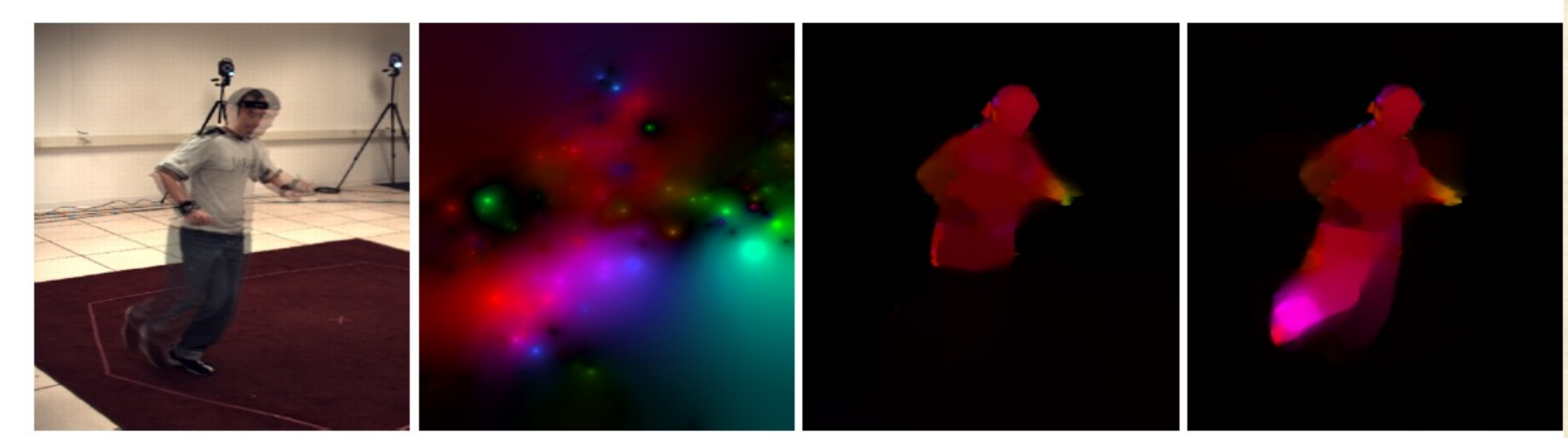
Start with something similar to Lucas-Kanade

+ gradient constancy

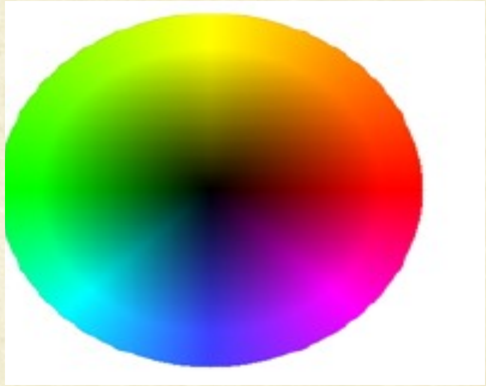
+ energy minimization with smoothing term

+ region matching

+ keypoint matching (long-range)



Region-based +Pixel-based +Keypoint-based



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

Summary

- Major contributions from Kanade Lucas, Tomasi
  - Tracking feature points
  - Optical flow
- Key ideas
  - By assuming brightness constancy, truncated Taylor expansion leads to simple and fast patch matching across frames
  - Coarse-to-fine registration
  - Use Gradient Consistency, Region Matching and Key-point Matching with better optimization techniques to overcome issues

Questions?