CS442/542b: Artificial Intelligence II
Prof. Olga Veksler

Lecture 16: Computer Vision

Motion

Slides are from Steve Seitz (UW), David Jacobs (UMD)

Outline

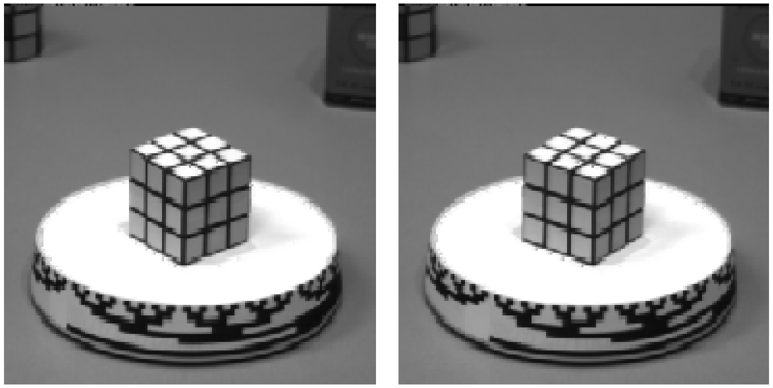
- Motion Estimation
  - Motion Field
  - Optical Flow Field
- Methods for Optical Flow estimation
  1. Discrete Search
  2. Lukas-Kanade Approach to Optical Flow
     - Optical Flow Constraint Equation
     - Aperture Problem
     - Pyramid Approach

Why estimate motion?

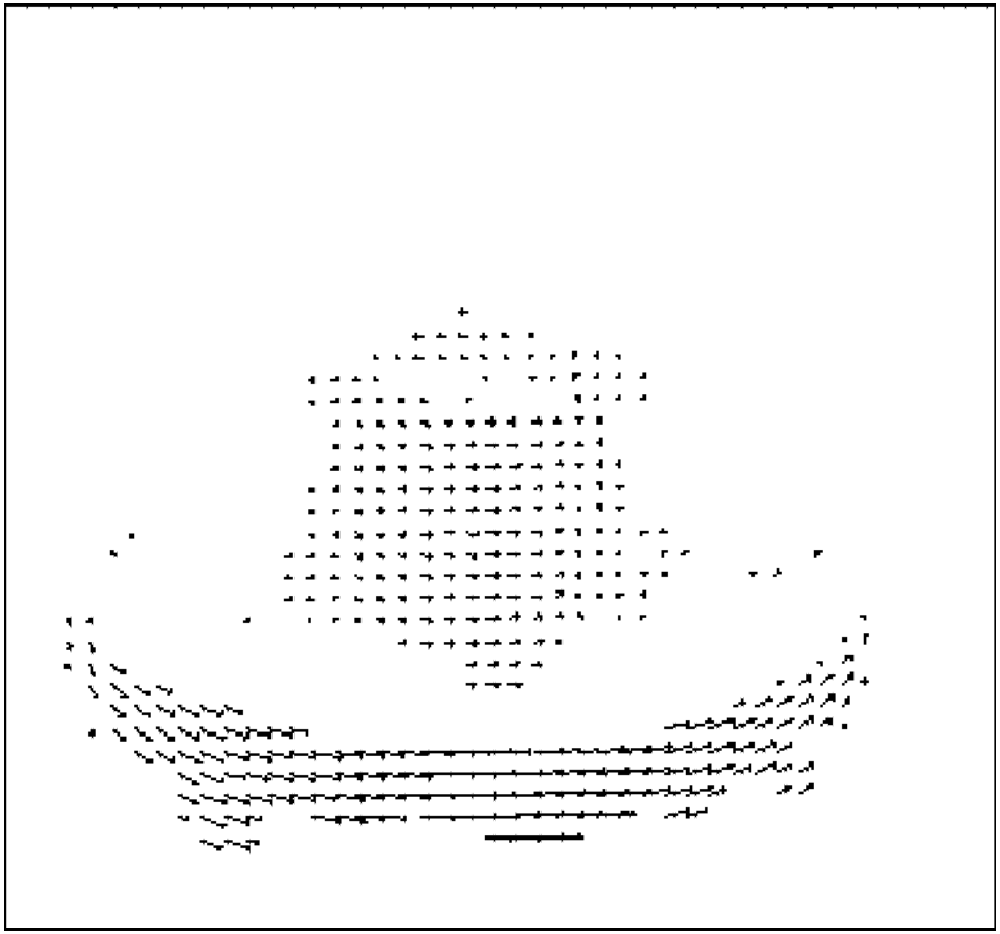
- Lots of uses
  - Track object(s)
  - Correct for camera jitter (stabilization)
  - Align images (mosaics)
  - 3D shape reconstruction
  - Special effects

Optical Flow and Motion Field

- Optical flow is the apparent motion of brightness patterns between 2 (or several) frames in an image sequence
  - Usually represent optical flow by a 2 dimensional vector (***u***,***v***)



Rubik's cube rotating to the right on a turntable



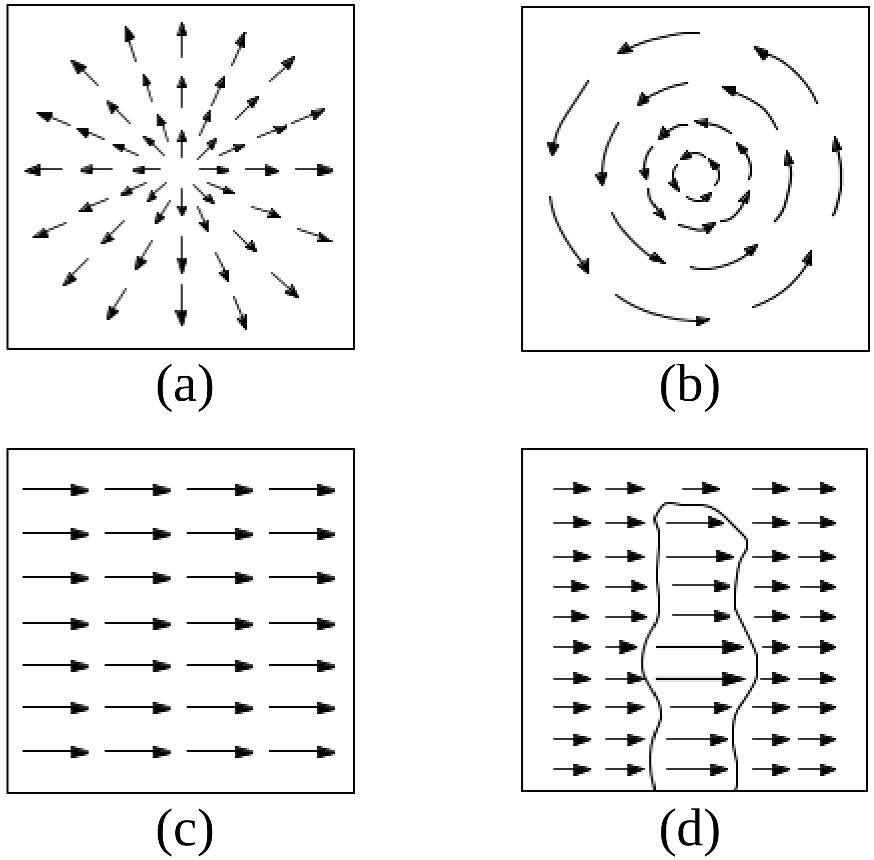
Optical Flow and Motion Field

- Optical flow is the apparent motion of brightness patterns between 2 (or several) frames in an image sequence
- Why does brightness change between frames?
- Assuming that illumination does not change:
  - changes are due to the RELATIVE MOTION between the scene and the camera
  - There are 3 possibilities:
    - Camera still, moving scene
    - Moving camera, still scene
    - Moving camera, moving scene
- Optical Flow is what we ***can*** estimate from image sequences

Motion Field (MF)

- The actual relative motion between 3D scene and the camera is 3 dimensional
  - motion will have horizontal (x), vertical (y), and depth (z) components, in general
- We can project these 3D motions onto the image plane
- What we get is a 2 dimensional motion field
- Motion field is the *projection* of the actual 3D motion in the scene onto the image plane
- Motion Field is what we actually **need** to estimate for applications

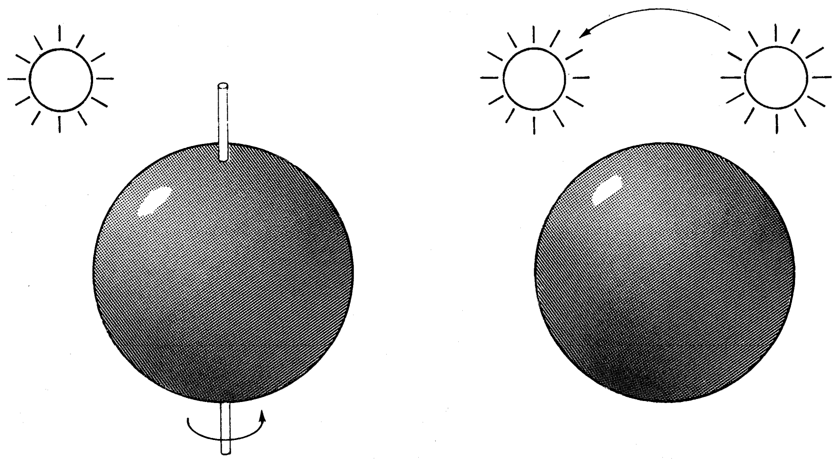
Examples of Motion Fields



(a) Translation perpendicular to a surface. (b) Rotation about axis perpendicular to image plane. (c) Translation parallel to a surface at a constant distance. (d) Translation parallel to an obstacle in front of a more distant background.

Optical Flow vs. Motion Field

- Optical Flow is the apperent motion of brightness patterns
- We equate Optical Flow Field with Motion Field
- Frequently works, but not always



(a) (b)

(a) A smooth sphere is rotating under constant illumination. Thus the optical flow field is zero, but the motion field is not

(b) A fixed sphere is illuminated by a moving source—the shading of the image changes. Thus the motion field is zero, but the optical flow field is not

Optical Flow vs. Motion Field

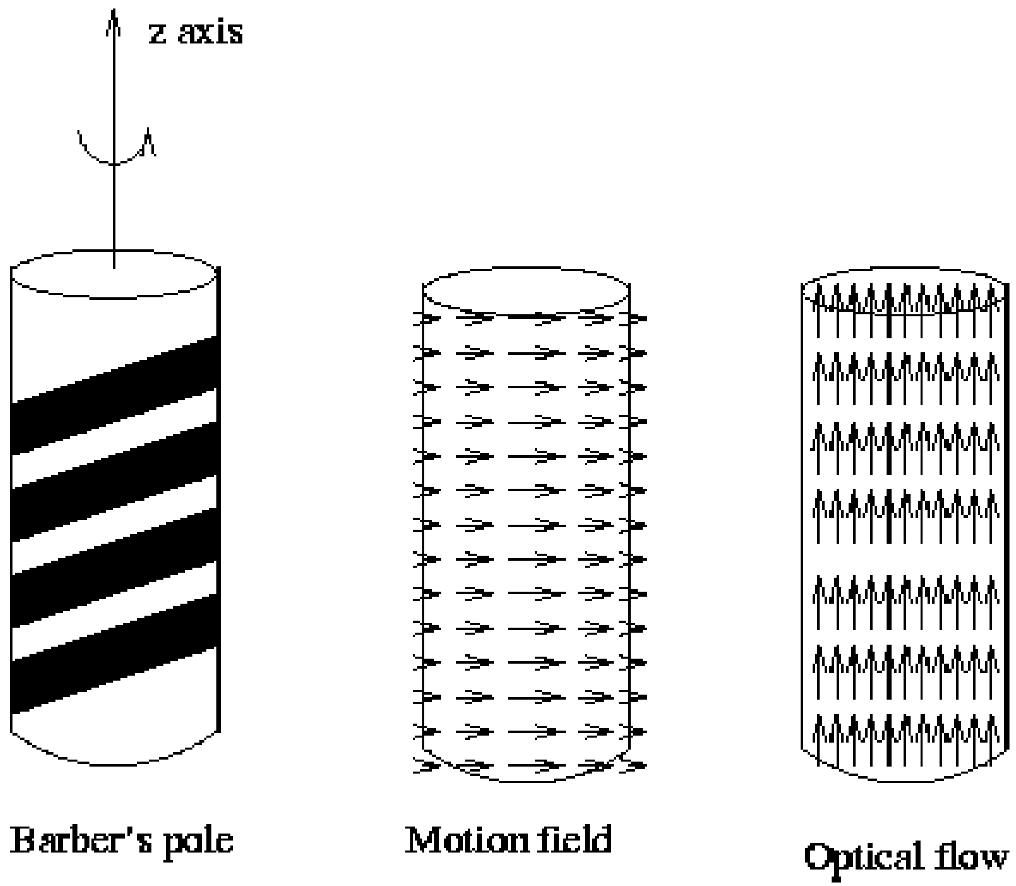
- Famous Illusions
  - Optical flow and motion fields do not coincide

http://www.sandlotscience.com/Distortions/Breathing_Square.htm

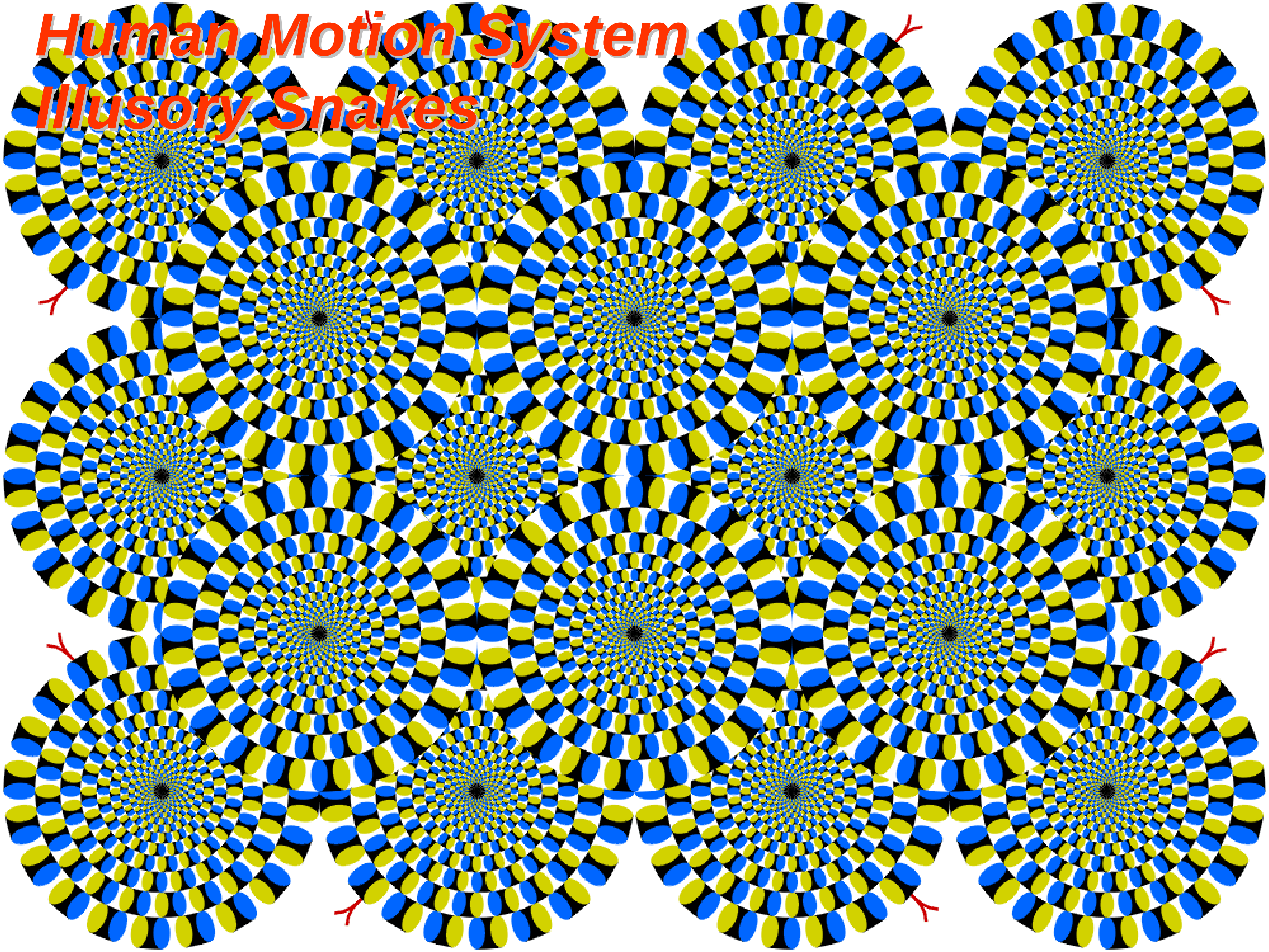
http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm

Optical Flow vs. Motion Field

- Motion field and Optical Flow are very different

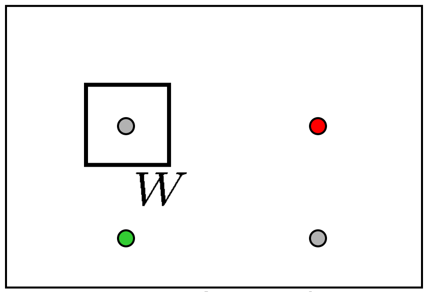


Human Motion System
Illusory Snakes

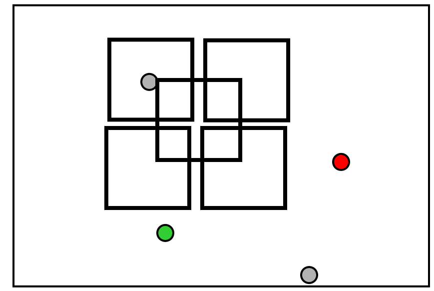


from Gary Bradski and Sebastian Thrun

Discrete Search for Optical Flow



$H(x, y)$



$I(x, y)$

- Given window W in ***H***, find best matching window in ***I***
- Minimize SSD (sum squared difference) or SAD (sum of absolute differences) of pixels in window
- just like window matching for stereo, except the set of locations to search over in the second image is different

$$min_{(u,v)} \left\{ \sum_{(x,y) \in W} |I(x+u, y+v) - H(x,y)|^2 \right\}$$

- search over a specified range of (u,v) values
  - this (u,v) range defines the **search range**
- can use integral image technique for fast search

Computing Optical Flow: Brightness Constancy Equation

- Can we estimate optical flow **without** the search over all possible locations?
  - Yes! If the motion is small…
- Let ***P*** be a moving point in 3D
  - At time ***t***, ***P*** has coordinates (***X***(***t***),***Y***(***t***),***Z***(***t***))
  - Let ***p***=(***x***(***t***),***y***(***t***)) be the coordinates of its image at time ***t***
  - Let $I(x(t),y(t),t)$ be the brightness at ***p*** at time ***t***.
- Brightness Constancy Assumption:
  - As ***P*** moves over time, $I(x(t),y(t),t)$ remains constant

Computing Optical Flow: Brightness Constancy Equation

$$I[x(t), y(t), t] = \text{constant}$$

Taking derivative with respect to time:

$$\frac{d\,I[x(t), y(t), t]}{dt} = 0$$

$$\Downarrow$$

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0$$

Computing Optical Flow: Brightness Constancy Equation

1 equation with 2 unknowns

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t}+\frac{\partial I}{\partial y}\frac{\partial y}{\partial t}+\frac{\partial I}{\partial t}=0$$

Let

$$\nabla I=\begin{bmatrix}\frac{\partial I}{\partial x}\\ \frac{\partial I}{\partial y}\end{bmatrix}$$ (Frame spatial gradient)

$$\begin{bmatrix}u\\ v\end{bmatrix}=\begin{bmatrix}\frac{\partial x}{\partial t}\\ \frac{\partial y}{\partial t}\end{bmatrix}$$ (optical flow)

$$I_t=\frac{\partial I}{\partial t}$$ (derivative across frames)

Computing Optical Flow: Brightness Constancy Equation

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0$$

- Written using dot product notation:

$$\begin{bmatrix} I_x \\ I_y \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t = 0$$

  - Where I have used more compact notation:

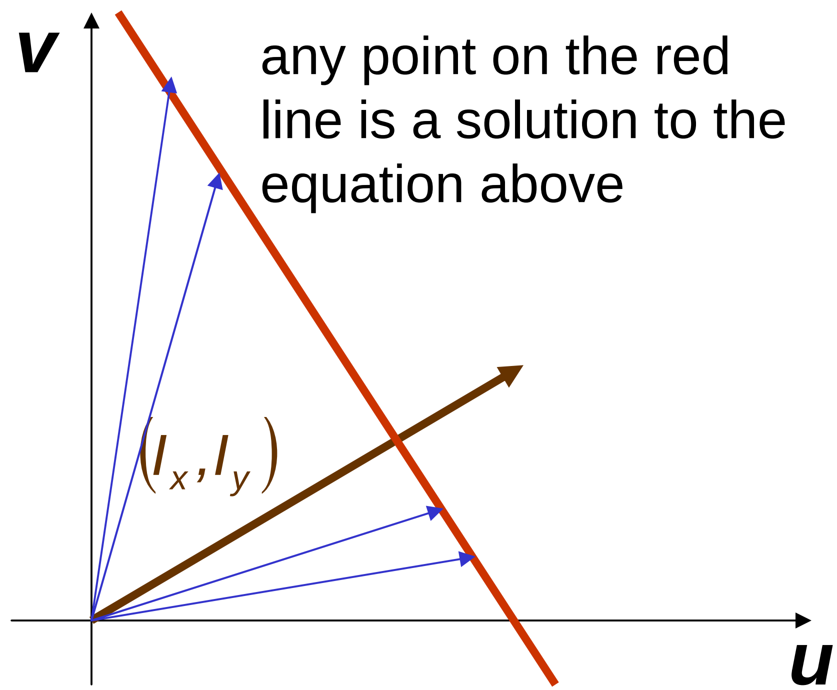
$$\frac{\partial I}{\partial x} = I_x \qquad \frac{\partial I}{\partial y} = I_y$$

Computing Optical Flow: Brightness Constancy Equation

1 equation with 2 unknowns:

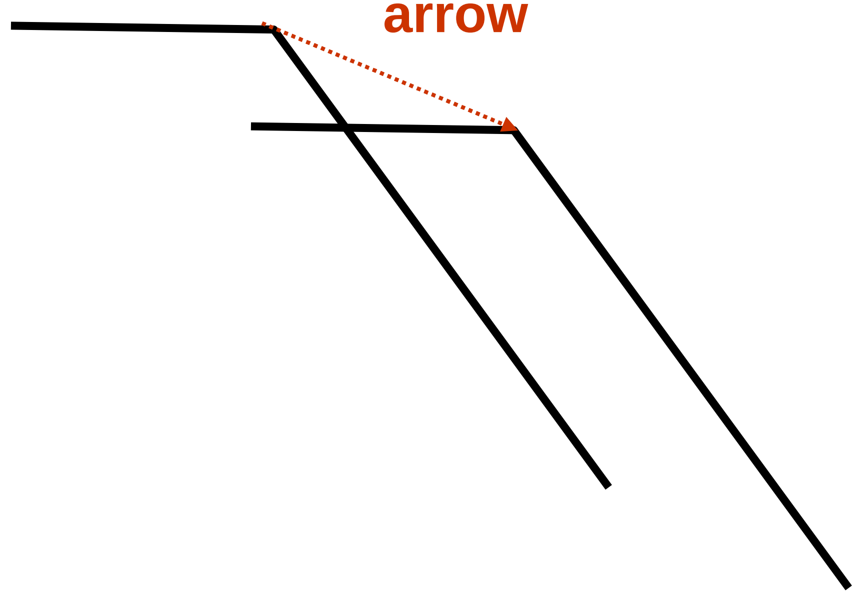
$$\begin{bmatrix} I_x \\ I_y \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t = 0$$

- Intuitively, what does this constraint mean?
  - The component of the flow in the gradient direction is determined
  - Recall that gradient points in the direction perpendicular to the edge
  - The component of the flow parallel to an edge is unknown

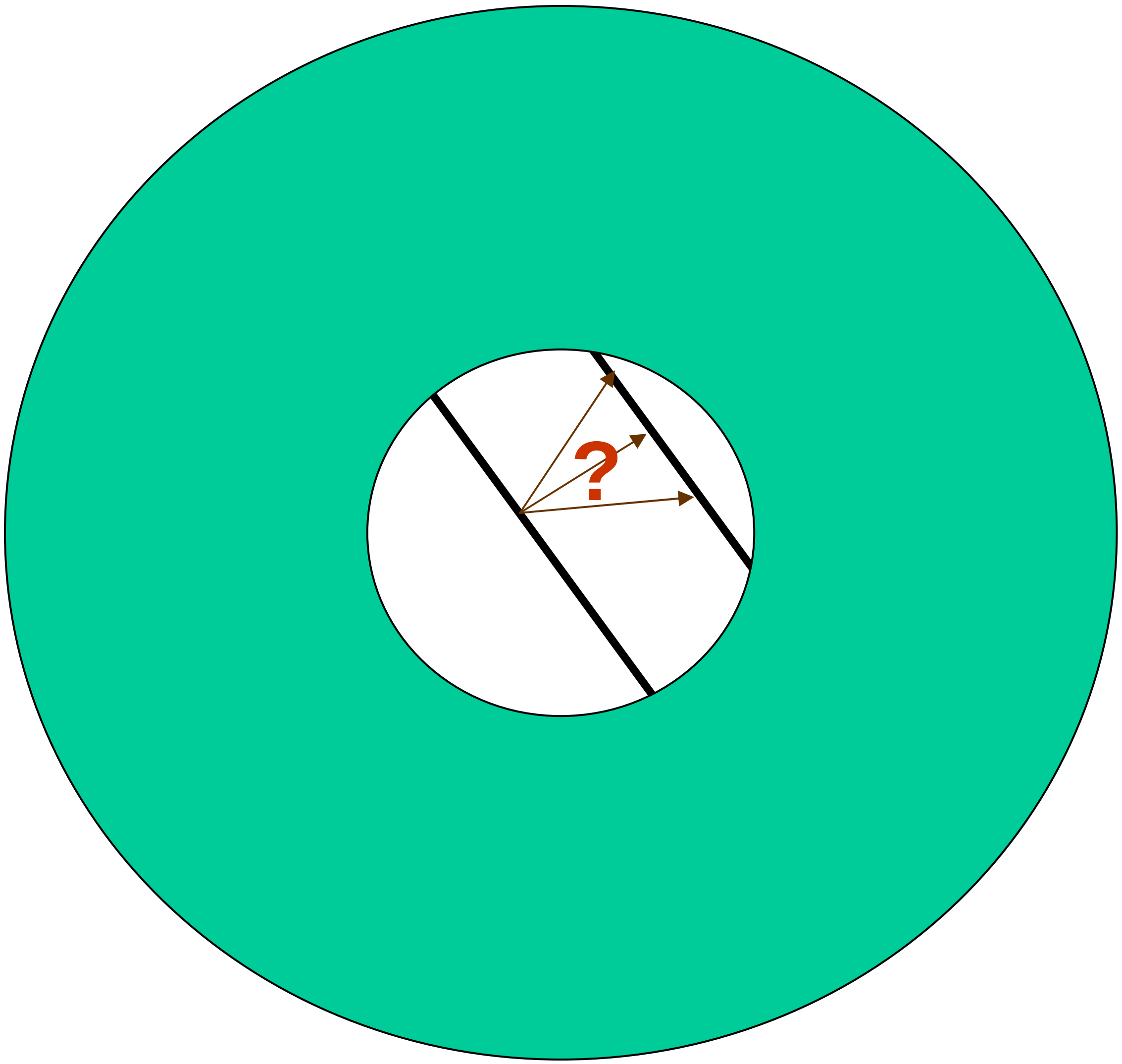


Aperture problem

true motion is in the direction of the red arrow



Aperture problem



Computing Optical Flow: Brightness Constancy Equation

- 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!

$$\boldsymbol{I}_t(\boldsymbol{p}_i) + \nabla \boldsymbol{I}(\boldsymbol{p}_i) \cdot \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} = \boldsymbol{0}$$

$$\underbrace{\begin{bmatrix} \boldsymbol{I}_x(\boldsymbol{p}_1) & \boldsymbol{I}_y(\boldsymbol{p}_1) \\ \boldsymbol{I}_x(\boldsymbol{p}_2) & \boldsymbol{I}_y(\boldsymbol{p}_2) \\ \vdots & \vdots \\ \boldsymbol{I}_x(\boldsymbol{p}_{25}) & \boldsymbol{I}_y(\boldsymbol{p}_{25}) \end{bmatrix}}_{\substack{\text{matrix } \boldsymbol{A} \\ 25\text{x}2}} \underbrace{\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix}}_{\substack{\text{vector } \boldsymbol{d} \\ 2\text{x}1}} = \underbrace{-\begin{bmatrix} \boldsymbol{I}_t(\boldsymbol{p}_1) \\ \boldsymbol{I}_t(\boldsymbol{p}_2) \\ \vdots \\ \boldsymbol{I}_t(\boldsymbol{p}_{25}) \end{bmatrix}}_{\substack{\text{vector } \boldsymbol{b} \\ 25\text{x}1}}$$

Computing Optical Flow: Brightness Constancy Equation

- I_x and I_y are computed just as before (recall lectures on filtering)
  - For example, can use Sobel operator

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad\qquad s_y$$

  - Note that 1/8 factor is now mandatory, unlike in edge detection, since we want the actual gradient value

Computing Optical Flow: Brightness Constancy Equation

- I_t is the derivative between the frames

121	121	122	123	122	123
121	121	122	123	122	123
122	123	124	123	124	123
120	122	122	123	**122**	123
121	121	124	123	124	123
125	120	124	123	124	123

I^5: frame at time = 5

121	121	122	123	20	20
121	121	122	123	22	22
122	123	124	123	24	21
120	122	122	123	**22**	22
121	121	124	123	24	23
125	120	124	123	24	24

I^6: frame at time = 5

- Simplest approximation to $I_t(p) = I^{t+1}(p) - I^t(p)$
- For example for pixel with coordinates (4,3) above

$$I_t(\mathbf{4,3}) = 22 - 122 = -100$$

Lukas-Kanade flow

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

matrix ***A***	vector ***d***	vector ***b***
25x2	2x1	25x1

- Problem: now we have more equations than unknowns
  - Where have we seen this before?
- Can't find the exact solution d, but can solve Least Squares Problem:

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

Lukas-Kanade flow

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

- Solution: solve least squares problem
  - minimum least squares solution given by solution (in ***d***) of:

$$\underset{2\text{x}2}{(A^T A)}\ \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$$

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

  - The summations are over all pixels in the K x K window
  - This technique was first proposed by Lucas & Kanade (1981)
  - Note: solution is at sub-pixel precision, that is you can get answer like u= 0.7 and v = -0.33
    - Contrast this with discrete search: to find answer at sub-pixel precision, you have to search at sub-pixel precision (usually)

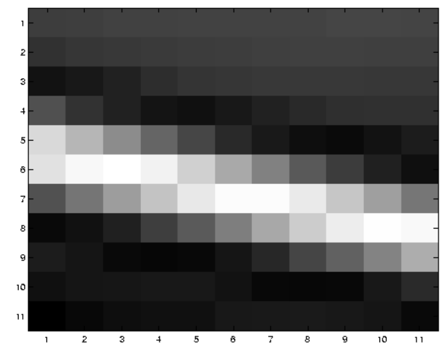
Conditions for solvability

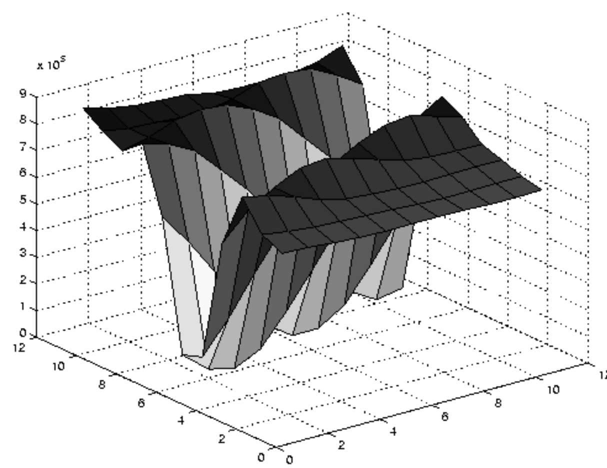
- Optimal (u, v) satisfies Lucas-Kanade equation

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

- When is this solvable?
  - **$A^T A$** should be invertible
  - **$A^T A$** entries should not be too small (noise)
  - **$A^T A$** should be well-conditioned
    - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)
    - The eigenvectors of $A^T A$ relate to edge direction and magnitude

Edge



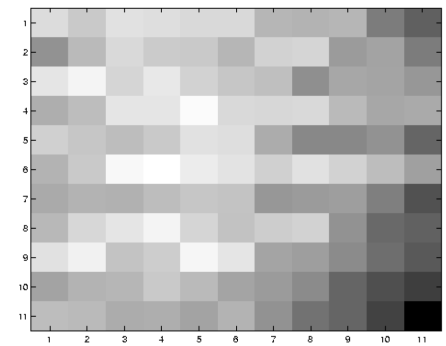


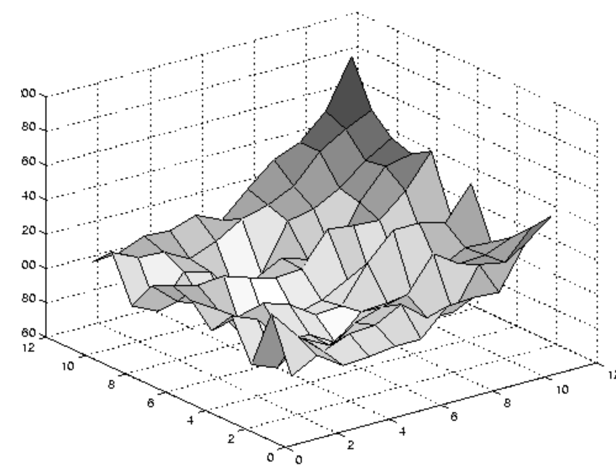
$$\sum \nabla I (\nabla I)^T$$

– gradients very large or very small

– large λ_1, small λ_2

Low texture region



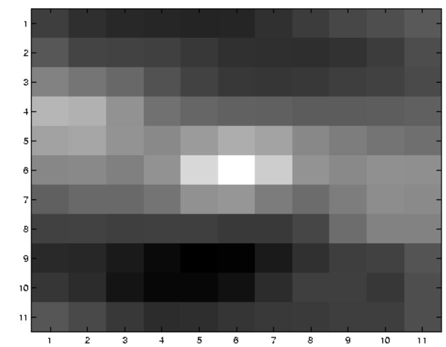


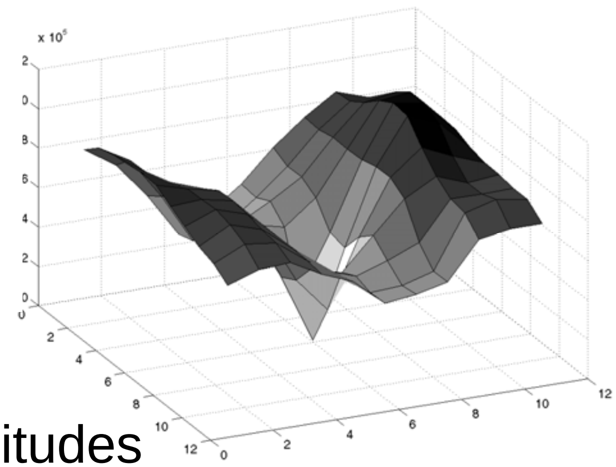
$$\sum \nabla I (\nabla I)^T$$

– 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
    - very useful for feature tracking

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
    - what is the ideal window size?

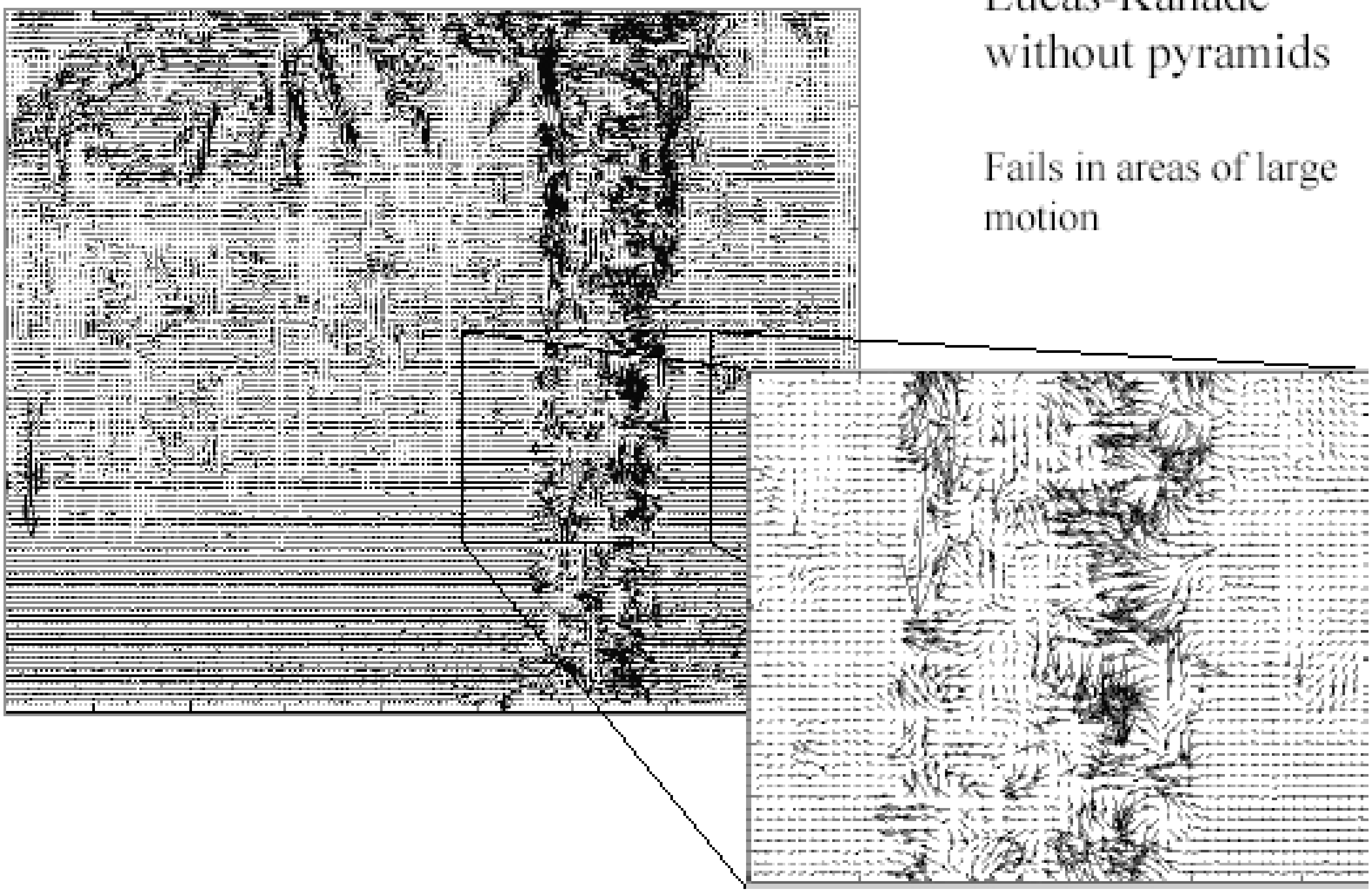
Iterative Refinement

- 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 image warping techniques
  3. Repeat until convergence

Optical Flow Results



Lucas-Kanade without pyramids

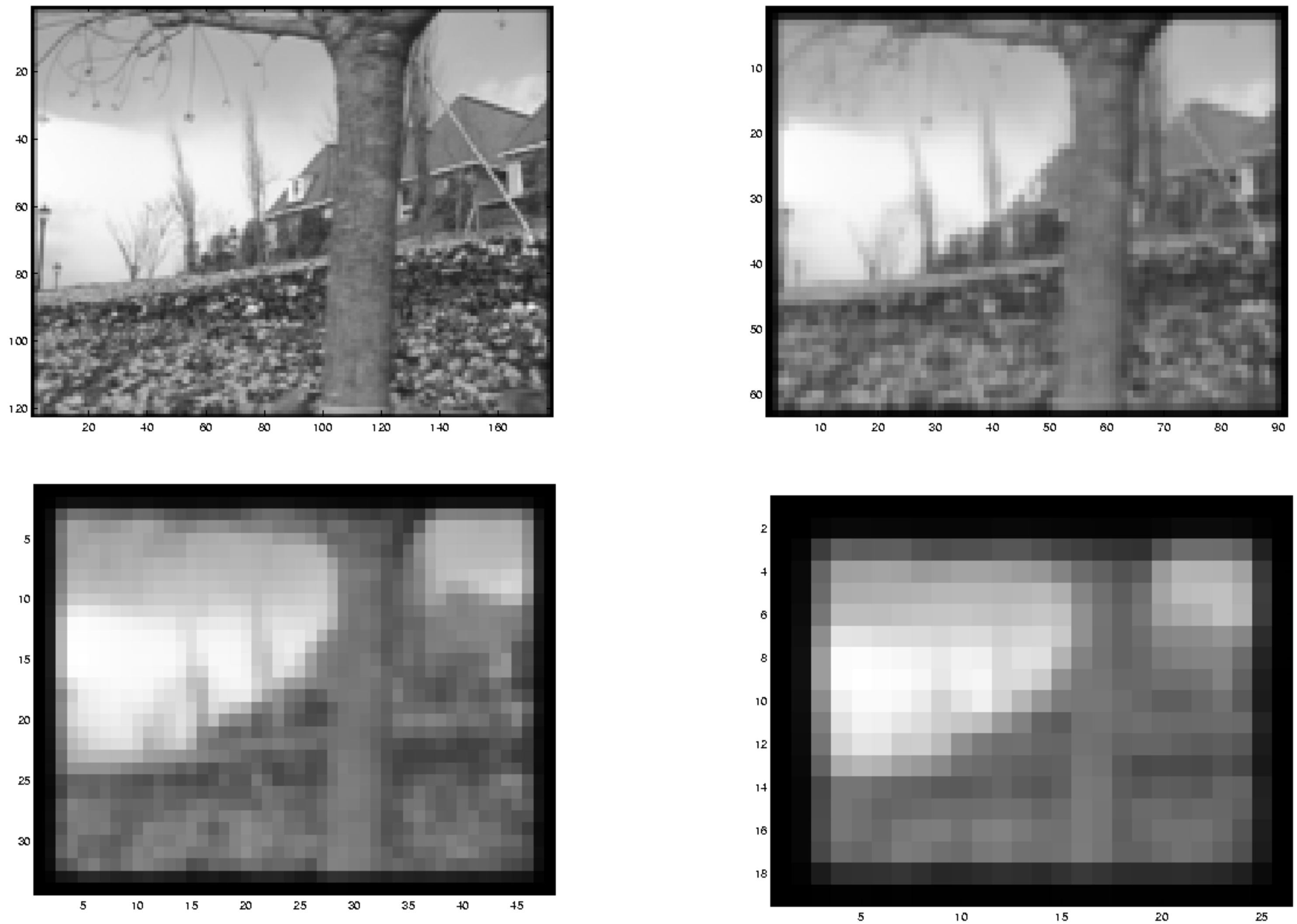
Fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

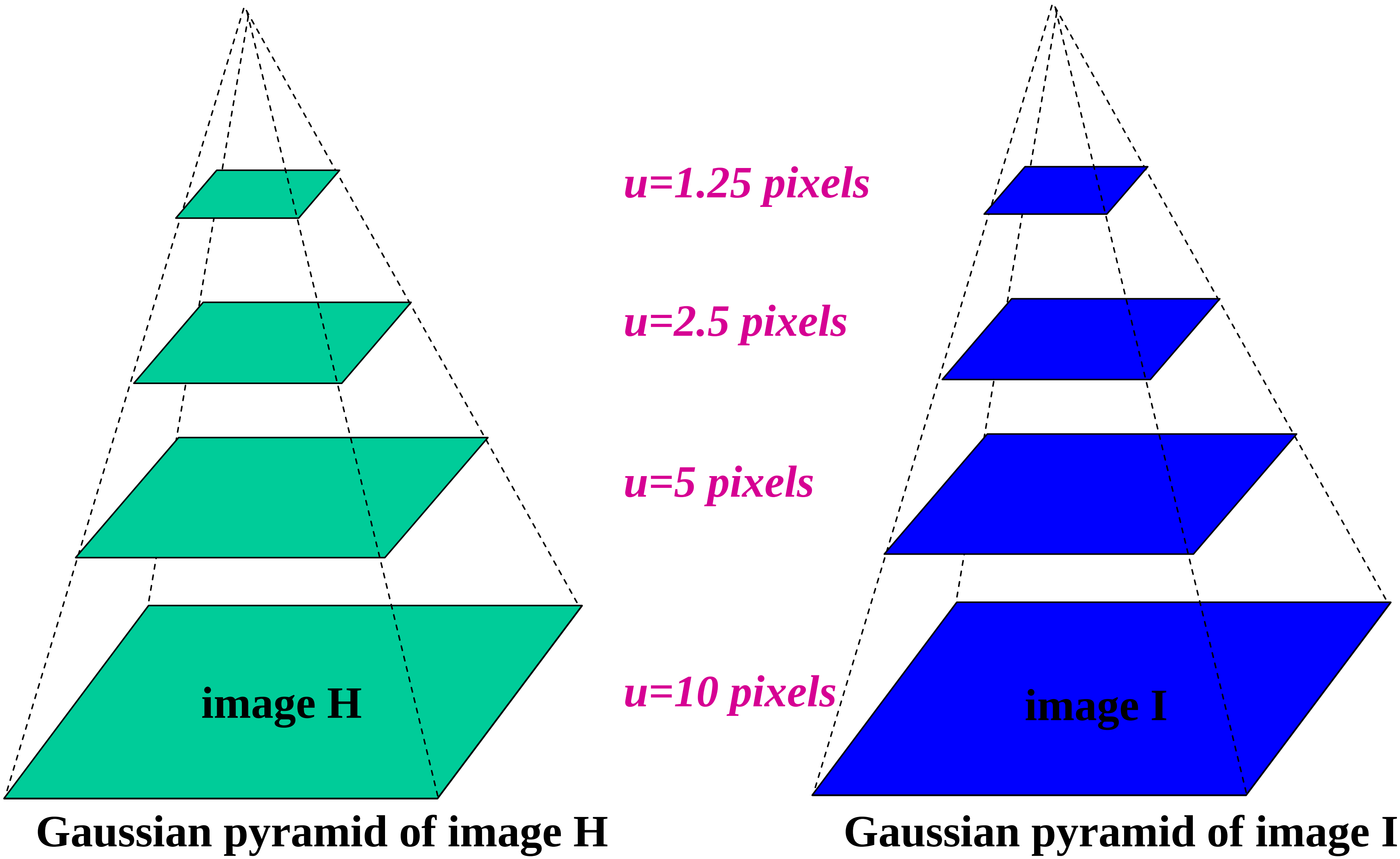
Revisiting the small motion assumption

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

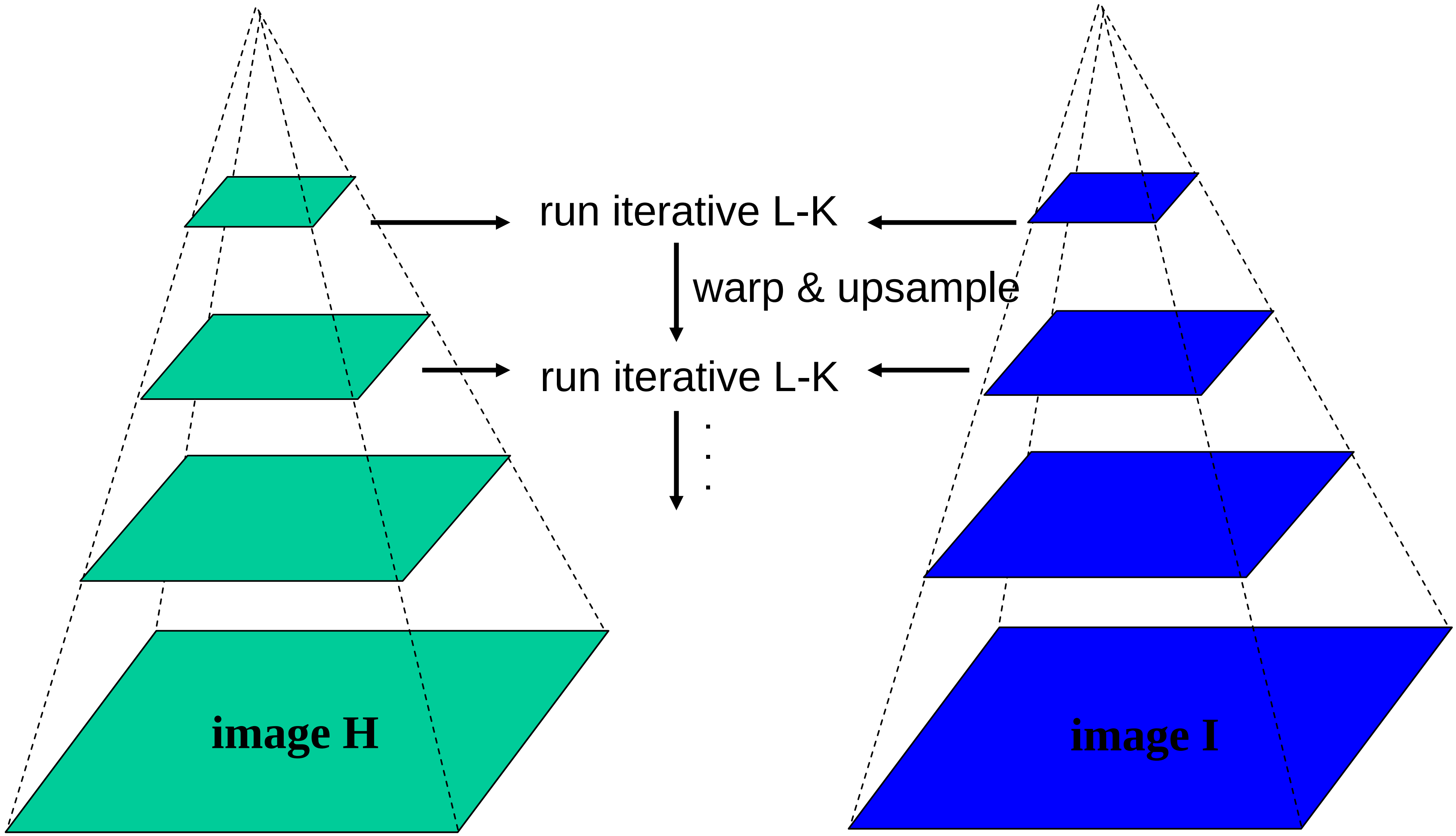
Reduce the resolution!



Coarse-to-fine optical flow estimation

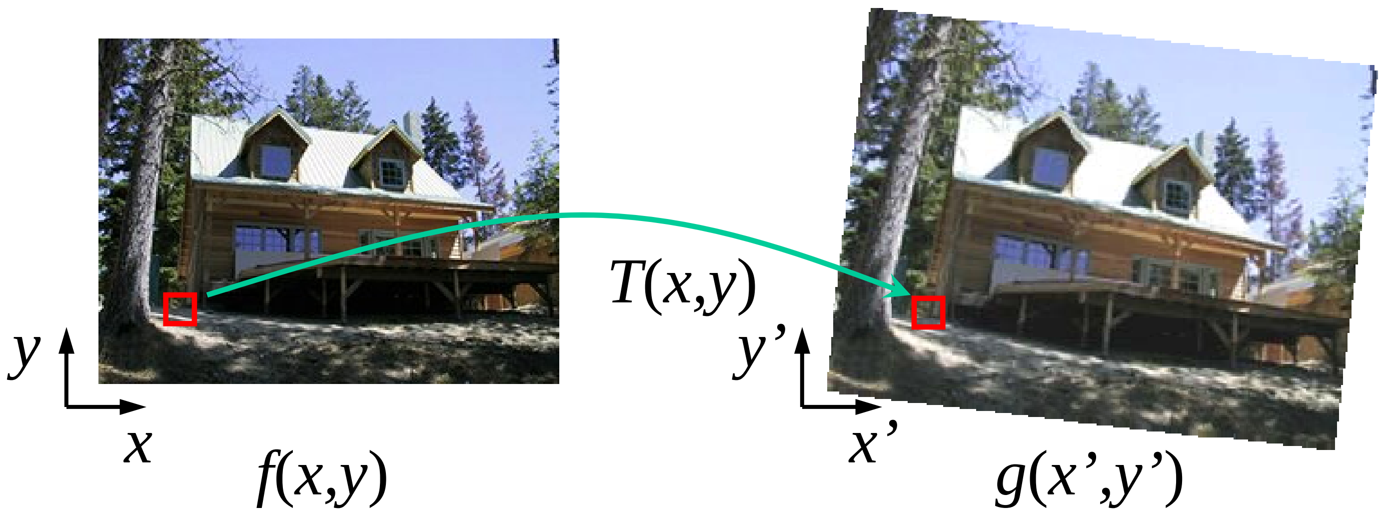


Coarse-to-fine optical flow estimation



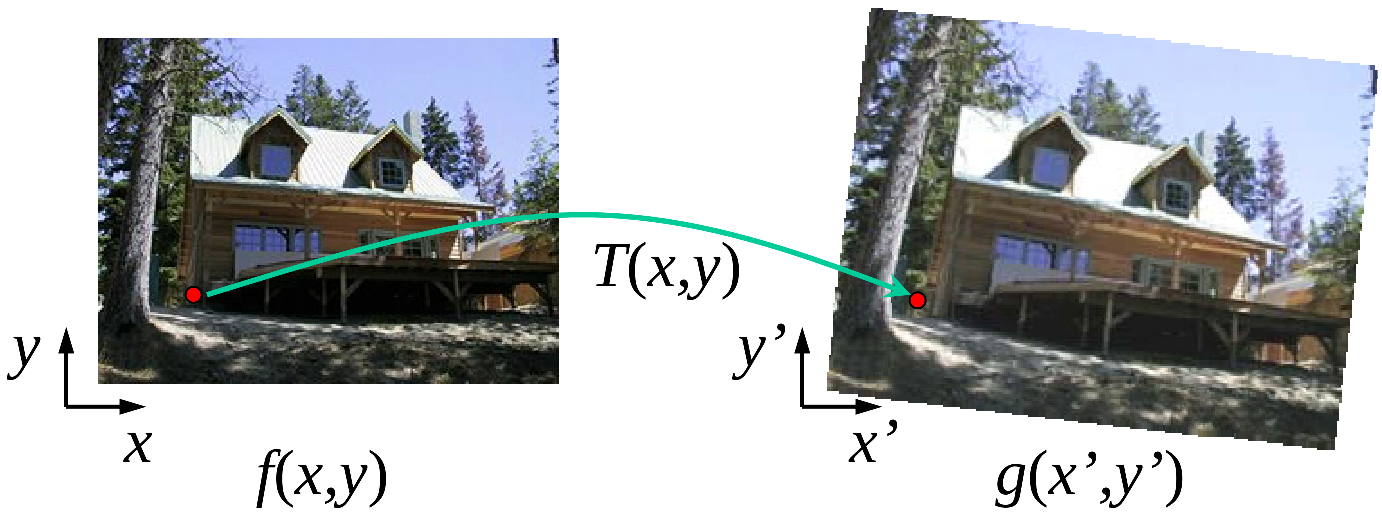
Gaussian pyramid of image H **Gaussian pyramid of image I**

Image warping



- Given a coordinate transform *(x',y')* = *h*(*x,y*) and a source image *f*(*x,y*), how do we compute a transformed image *g(x',y')* = *f*(*T*(*x,y*))?

Forward warping

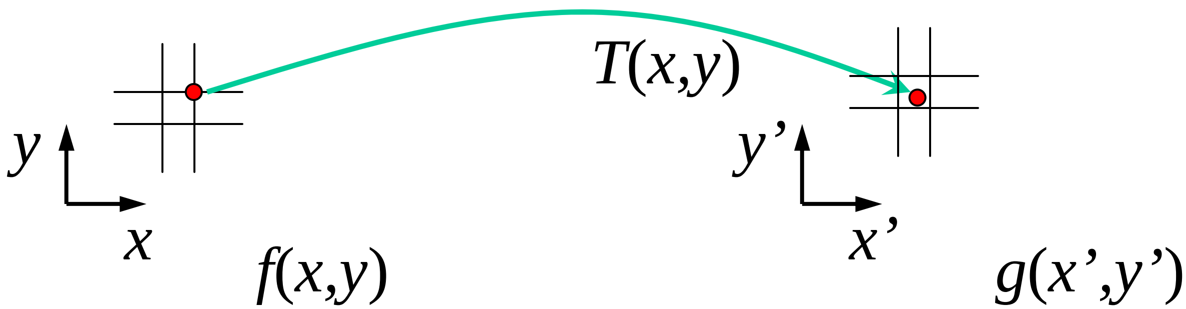


- Send each pixel $f(x,y)$ to its corresponding location

  $(x',y') = T(x,y)$ in the second image

Q: what if pixel lands “between” two pixels?

Forward warping



- Send each pixel $f(x,y)$ to its corresponding location

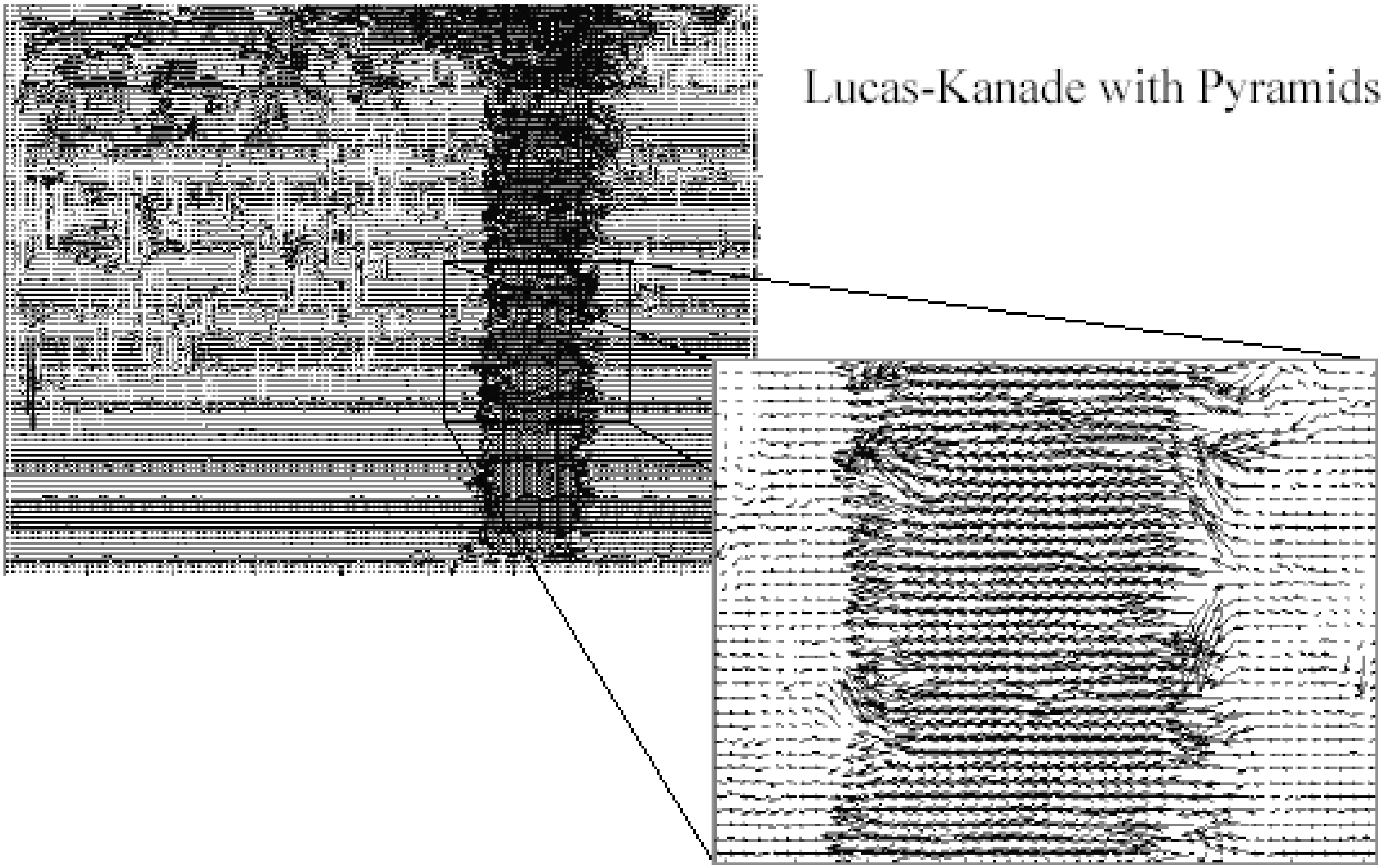
$$(x',y') = T(x,y) \text{ in the second image}$$

Q: what if pixel lands “between” two pixels?

A: distribute color among neighboring pixels (x’,y’)

– Known as “splatting”

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

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 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
  - How to find these features?
    - windows where $\sum \nabla I (\nabla I)^T$ has two large eigenvalues
  - Called the Harris Corner Detector

Feature Detection

Tracking features

- Feature tracking
  - Compute optical flow for that feature for each consecutive H, I
- 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
  - Large motions
    - will pyramid techniques work for feature tracking?

Tracking Over Many Frames

- Feature tracking with m frames
  1. Select features in first frame
  2. Given feature in frame i, compute position in i+1
  3. Select more features if needed
  4. i = i + 1
  5. If i < m, go to step 2

- Issues
  - Discrete search vs. Lucas Kanade?
    - depends on expected magnitude of motion
    - discrete search is more flexible
  - Compare feature in frame i to i+1 or frame 1 to i+1?
    - affects tendency to drift..
  - How big should search window be?
    - too small: lost features. Too large: slow