CS9840
Machine Learning in Computer Vision
Olga Veksler

Lecture 3

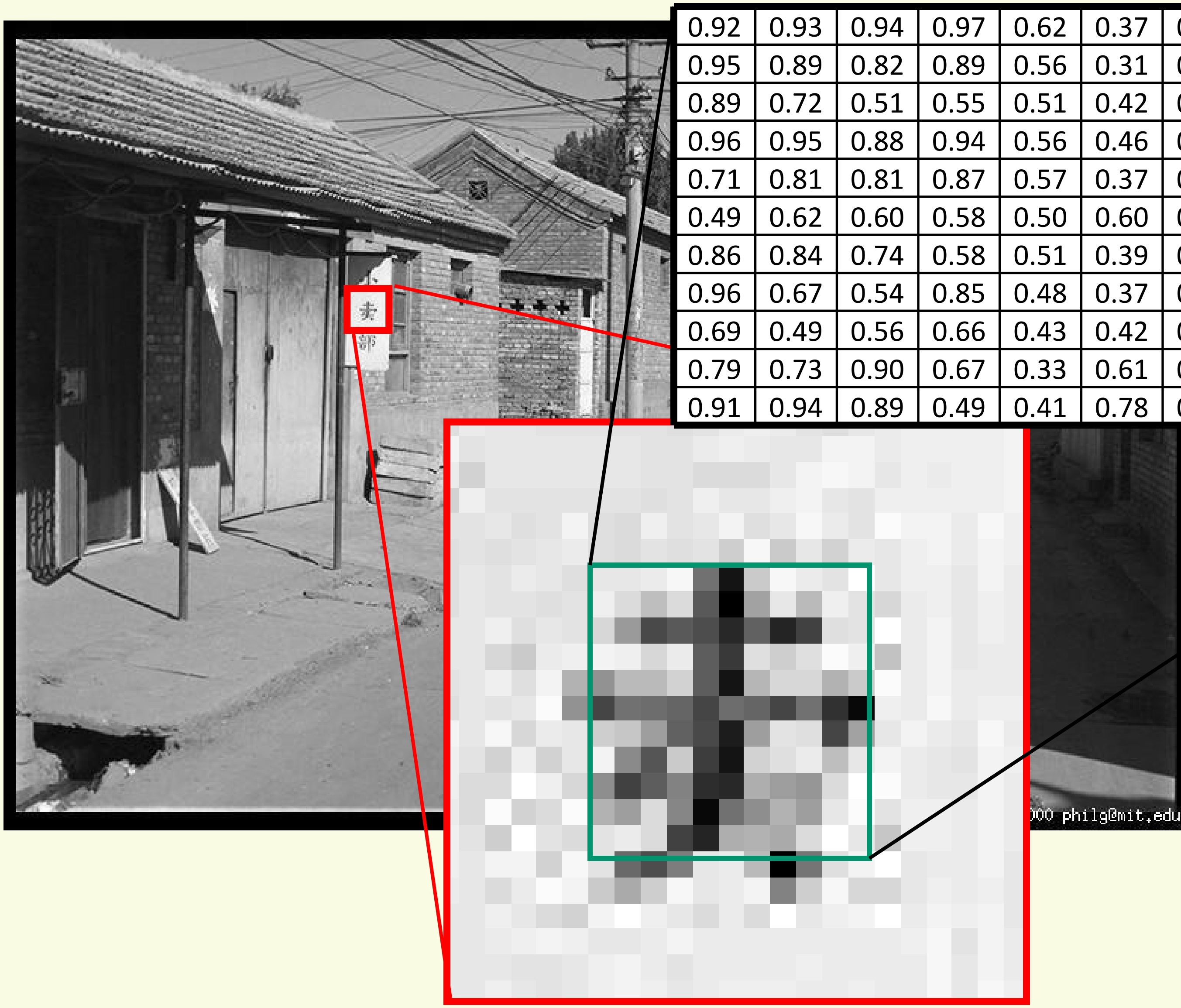
Computer Vision Concepts

Some Slides are from Cornelia, Fermüller, Mubarak Shah, Gary Bradski, Sebastian Thrun, Derek Hoiem

Outline

- Some Concepts in Image Processing/Vision
  - Filtering
  - Edge Detection
  - Image Features
  - Measures for Template matching
    - Correlation
    - SSD
    - Normalized Cross Correlation
  - Motion and Optical Flow Field

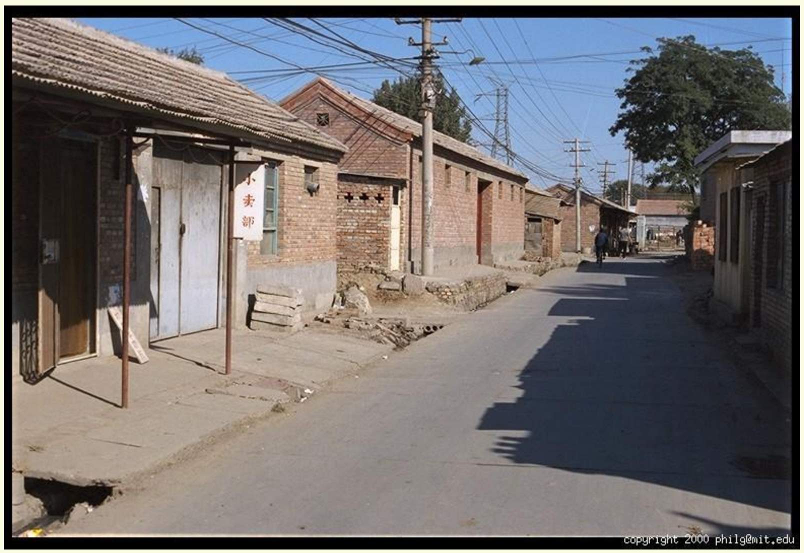
The Raster Image (Pixel Matrix)

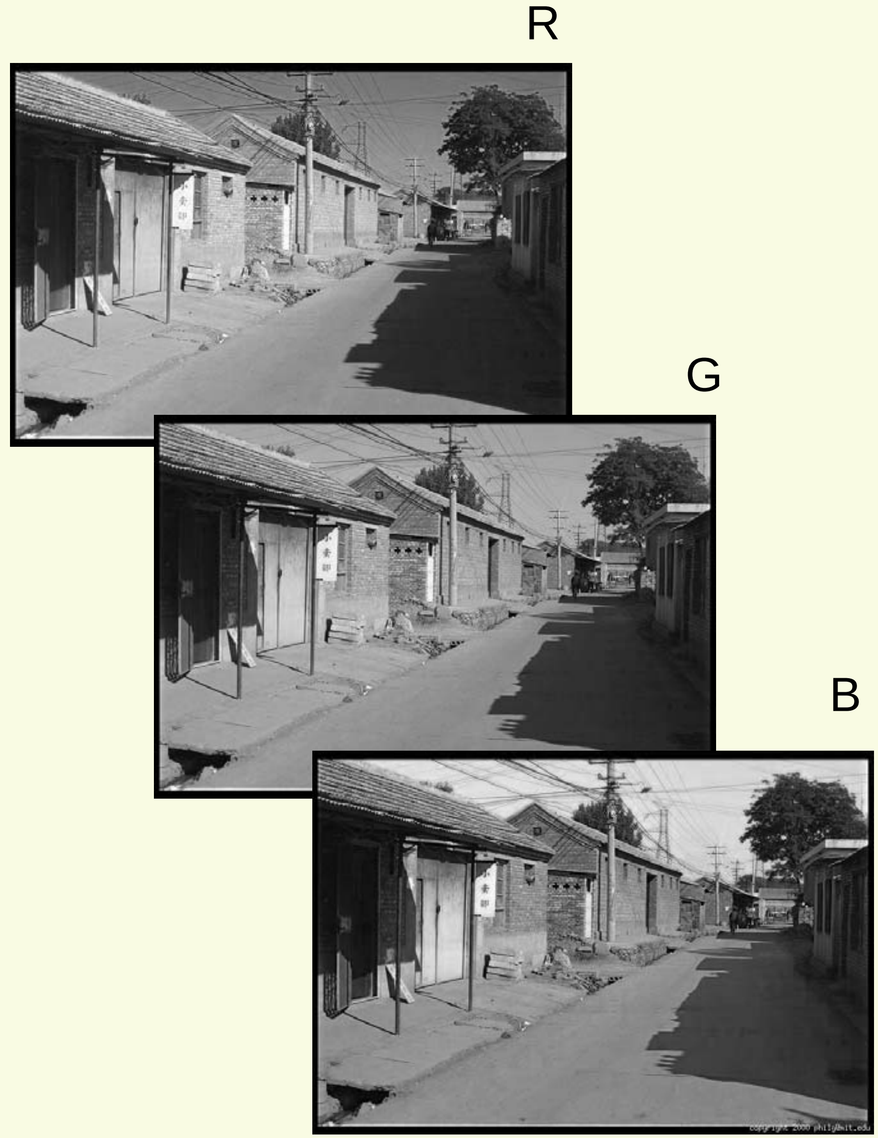


0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

Slide Credit: D. Hoeim

Color Image





Slide Credit: D. Hoeim

Basic Image Processing: Filtering

- Example: Box Filter

$$g[\cdot\ ,\cdot\ ]$$

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

Slide credit: David Lowe (UBC)

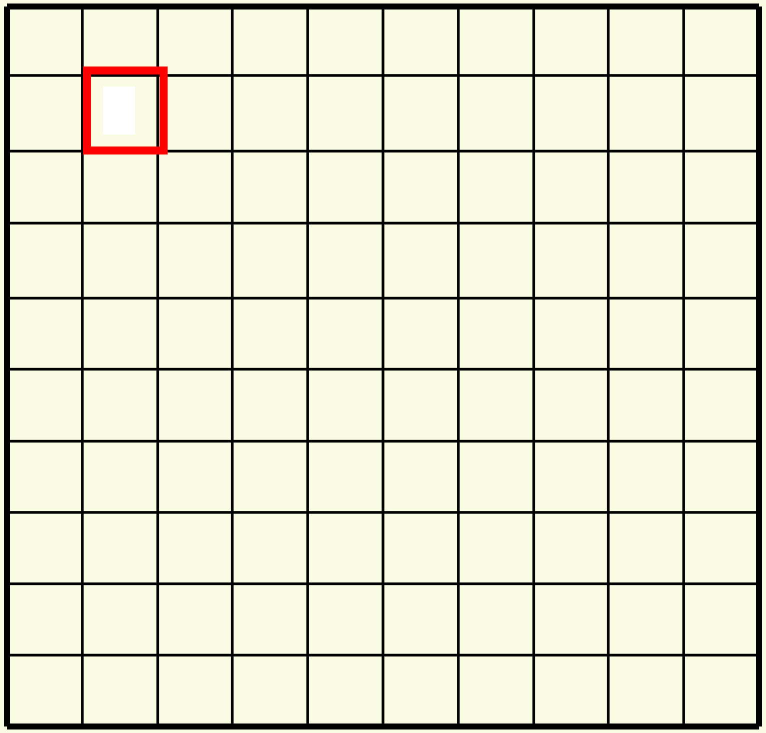
Image Filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$



$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20						

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$$g[\cdot,\cdot]\ \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$$g[\cdot\,,\cdot\,] \quad \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\; f[m+k, n+l]$$

Credit: S. Seitz

Convolution

- Convolution is the operation of applying a filter or a “kernel” to each pixel of an image

image

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

kernel

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

- Result of convolution has the same dimension as the image
- For example:

$$O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23}$$

- Convolution is frequently denoted by *, for example I*K

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

$$g[\cdot\,,\cdot\,]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

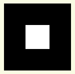
Slide credit: David Lowe (UBC)

Image filtering

- Image filtering: compute function of local neighborhood at each position
- Linear filtering: function is a weighted sum/difference of pixel values
- What does it do?
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract features from images
    - edges, distinctive points, texture, etc.
  - Detect patterns
    - Template matching

Slide Credit: D. Hoeim

Smoothing with box filter

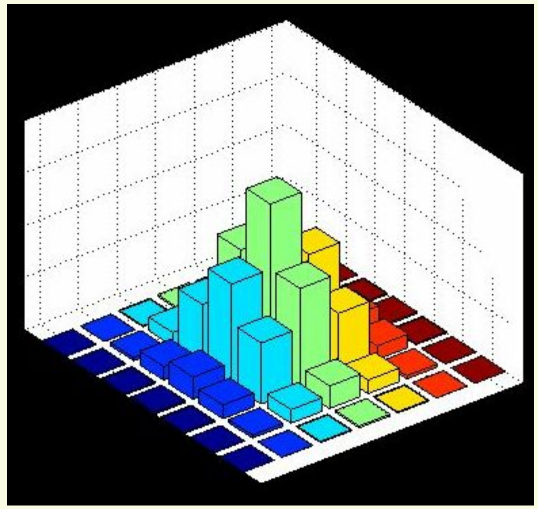


Slide Credit: D. Hoeim

Gaussian Filtering

- Gaussian smoothing (blurring) removes small detail and noise from an image

*



=

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Gaussian Smoothing vs. Averaging

smoothing by box filter

$\frac{1}{25}$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

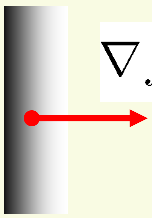
Gaussian Smoothing

$\frac{1}{273}$

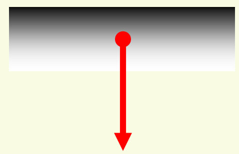
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Image Gradient

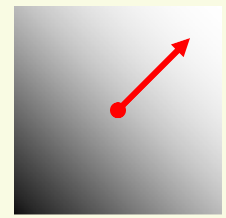
- Image gradient: points in the direction of the most rapid increase in intensity of image ***f***



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- Sobel filter for gradient:

$\frac{1}{8}$

-1	0	1
-2	0	2
-1	0	1

$\frac{\partial f}{\partial x}$

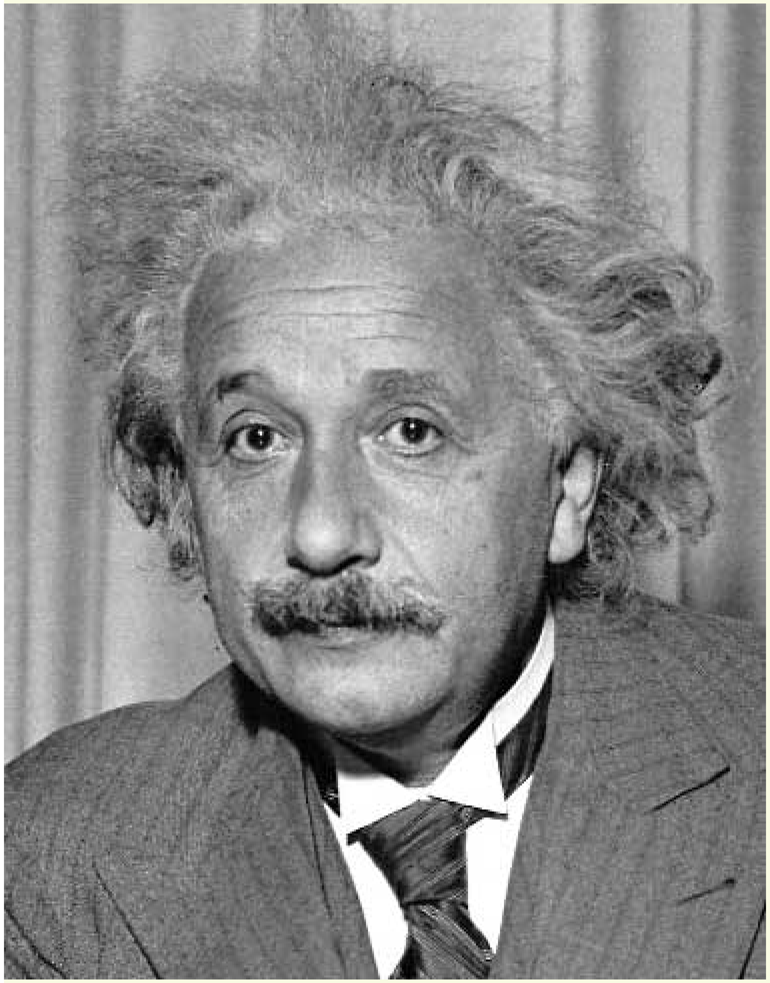
$\frac{1}{8}$

1	2	1
0	0	0
-1	-2	-1

$\frac{\partial f}{\partial y}$

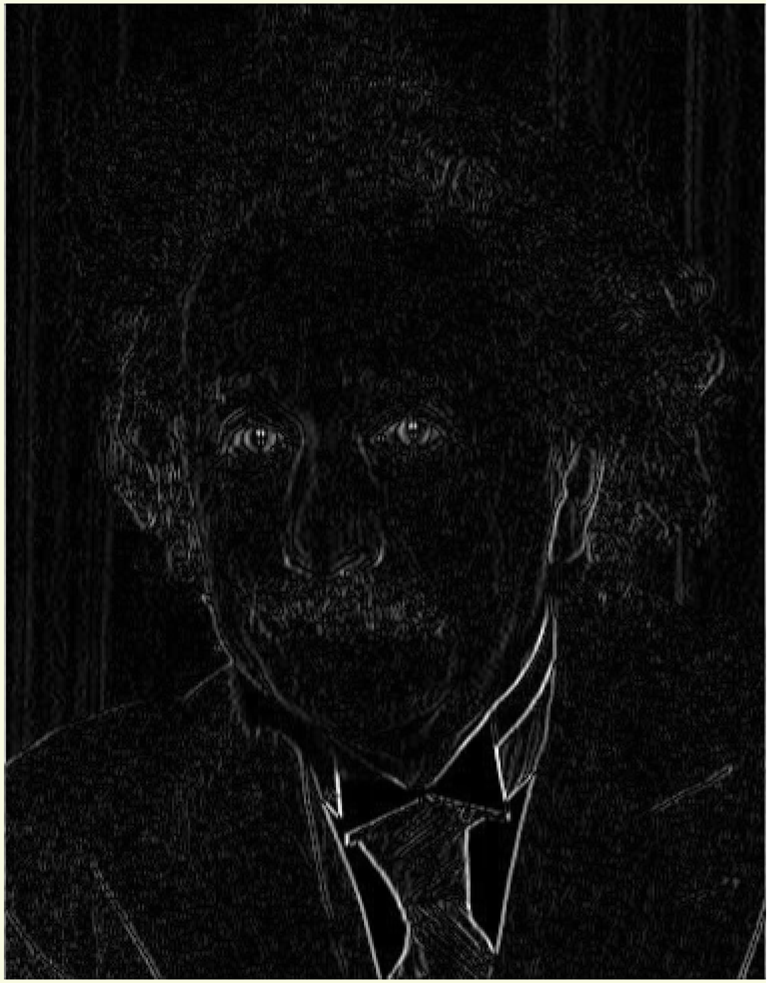
- Gradient is useful for edge detection

Sobel Filter for Vertical Gradient Component



1	0	-1
2	0	-2
1	0	-1

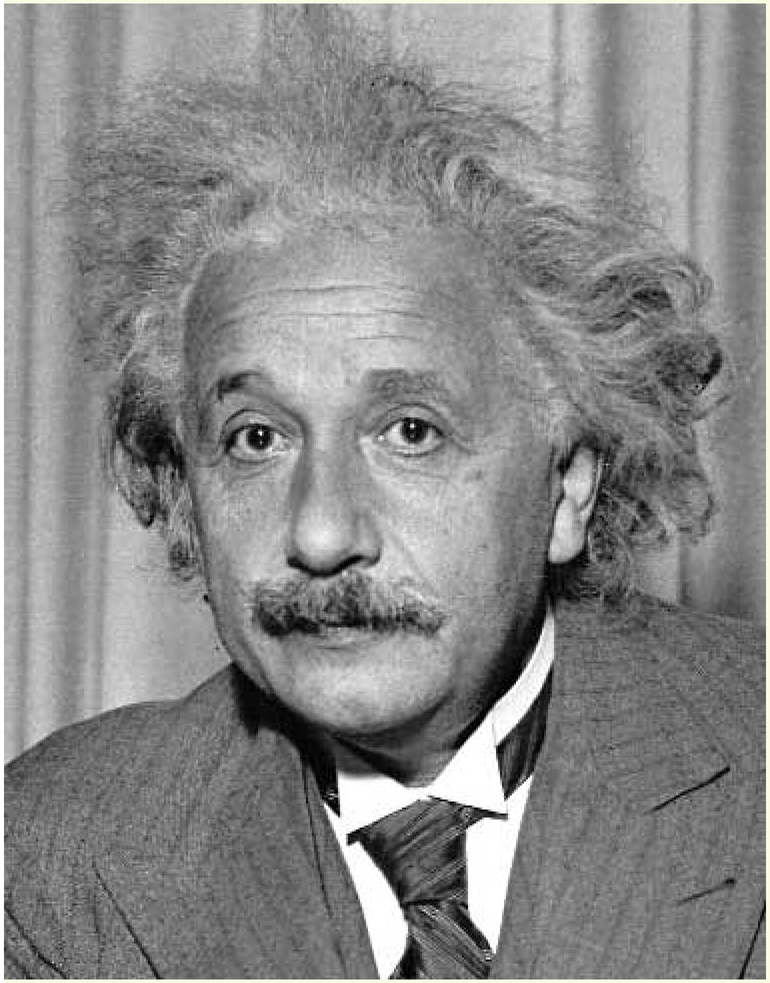
Sobel



Vertical Edge
(absolute value)

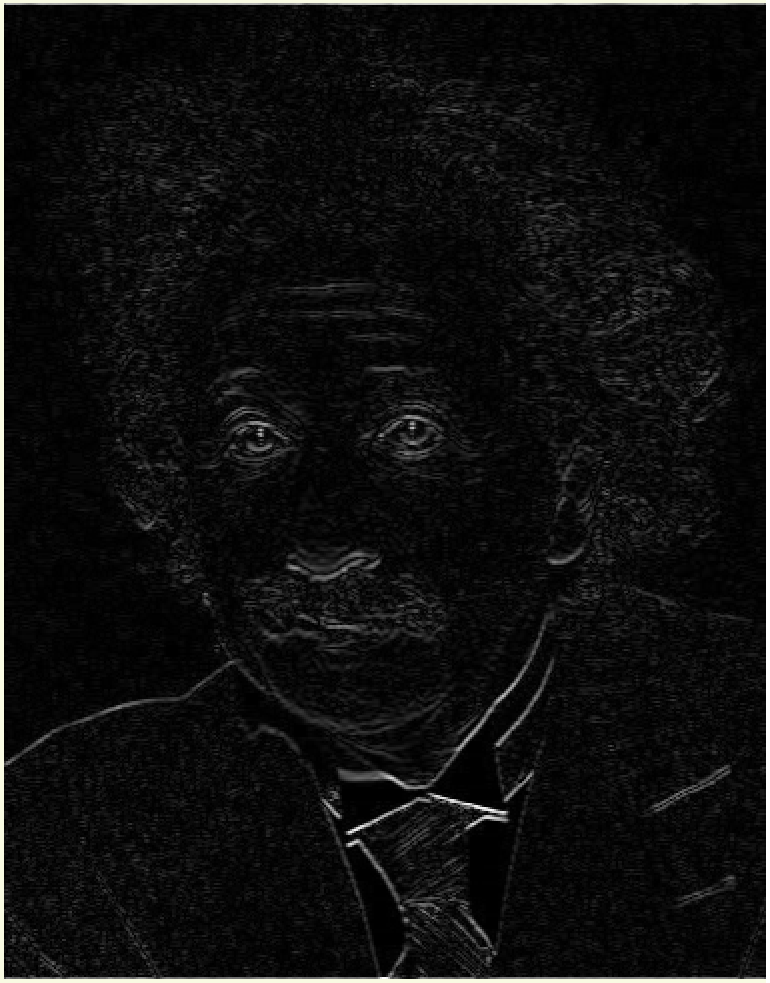
Slide Credit: D. Hoeim

Sobel Filter for Horizontal Gradient Component



1	2	1
0	0	0
-1	-2	-1

Sobel



Horizontal Edge
(absolute value)

Slide Credit: D. Hoeim

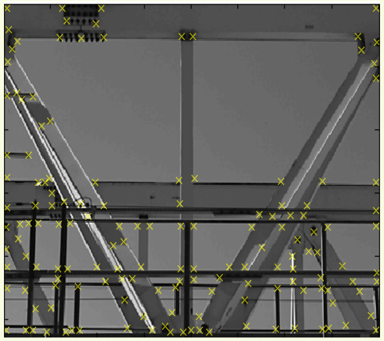
Edge Detection

canny edge detector

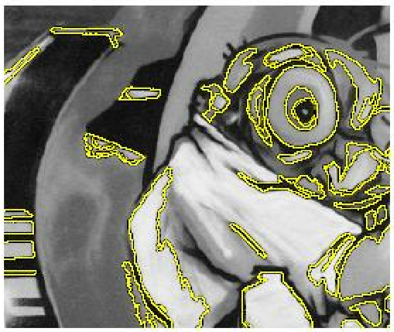
- Smooth image
  - gets rid of noise and small detail
- Compute Image gradient (with Sobel filter, etc)
- Pixels with large gradient magnitude are marked as edges
- Can also apply non-maximum suppression to “thin” the edges and other post-processing

Image Features

- Edge features capture places where something interesting is happening
  - large change in image intensity
- Edges is just one type of image features or “interest points”
- Various type of corner features, etc. are popular in vision
- Other features:



corners

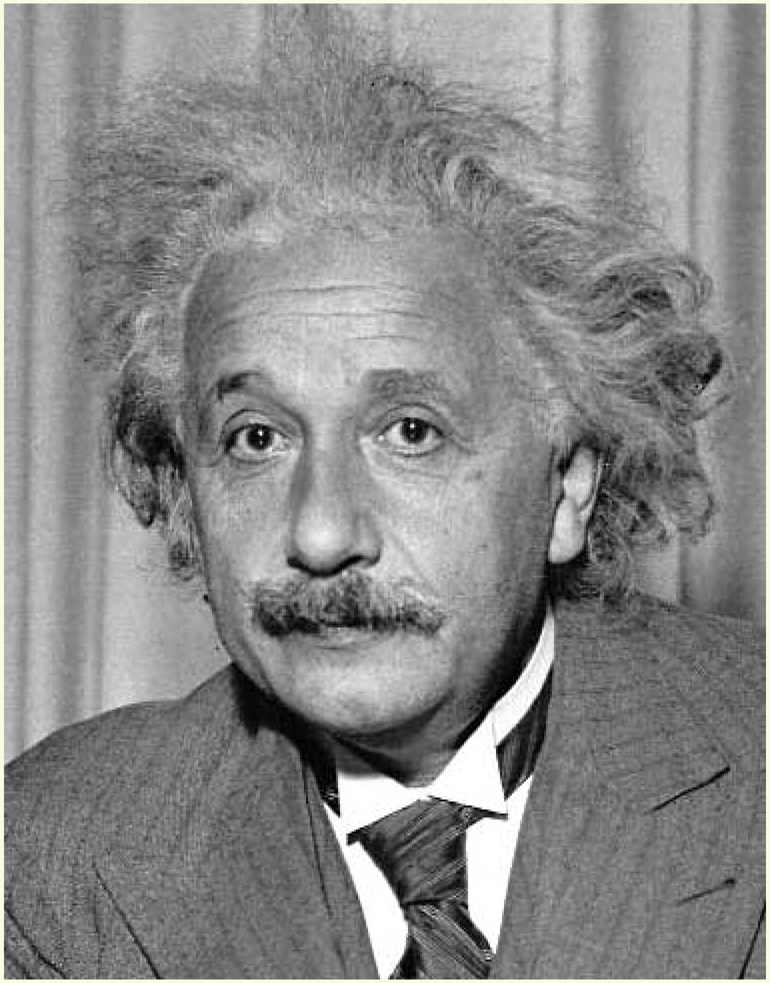


stable regions

SIFT

Template matching

- Goal: find  in image
- Main challenge: What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation



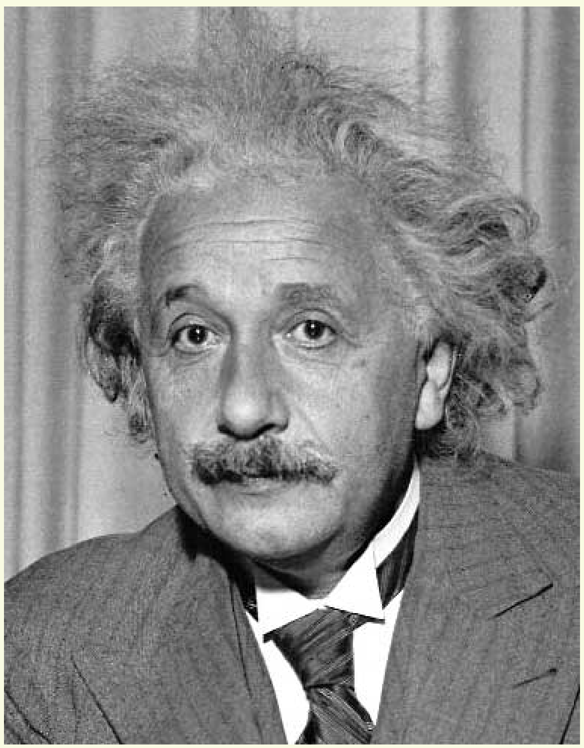
Slide Credit: D. Hoeim

Method 0: Correlation

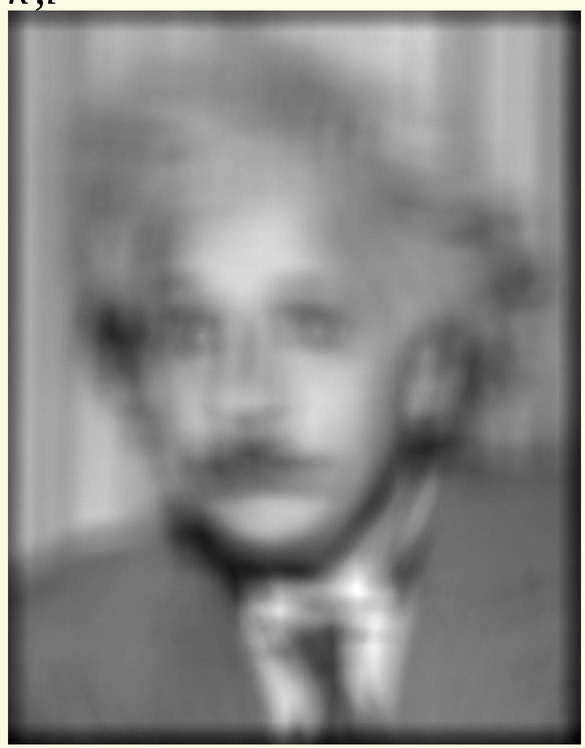
- Goal: find  in image
- Filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k, n+l]$$

f = image
g = filter



Input



Filtered Image

What went wrong?

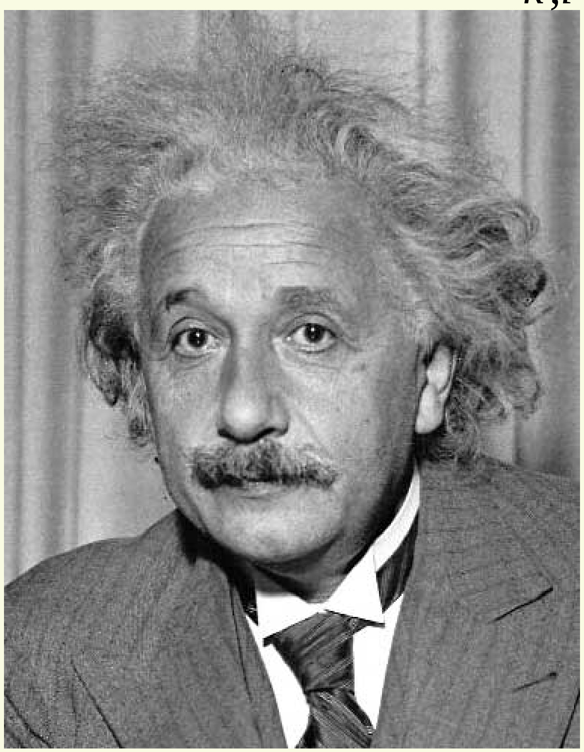
Slide Credit: D. Hoeim

Method 1: zero-mean Correlation

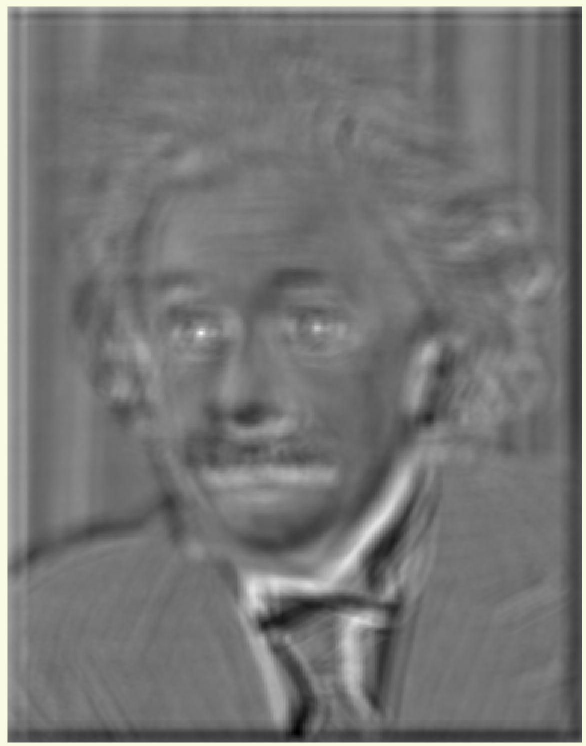
- Goal: find  in image
- Filter the image with zero-mean eye

$$h[m,n]=\sum_{k,l}(g[k,l]-\bar{g})\,(f[m+k,n+l])$$

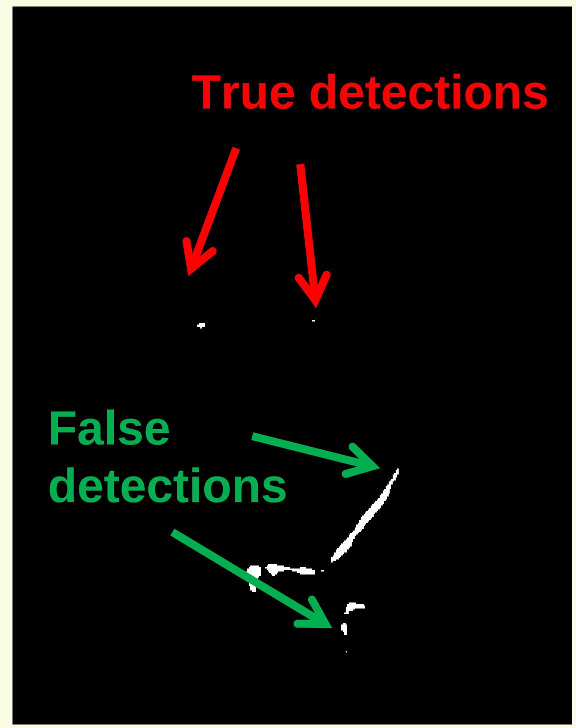
mean of template g



Input



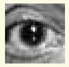
Filtered Image (scaled)



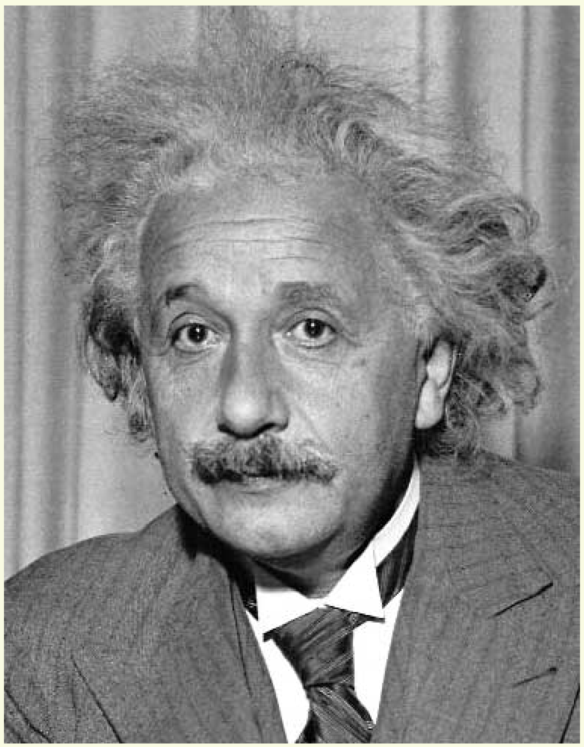
Thresholded Image

Slide Credit: D. Hoeim

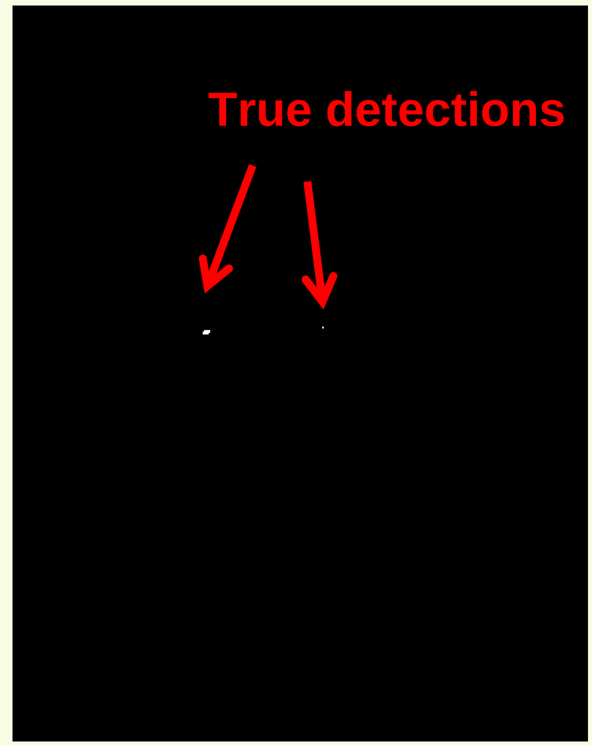
Method 3: Sum of Squared Differences

- Goal: find  in image

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$


Input


1- sqrt(SSD)

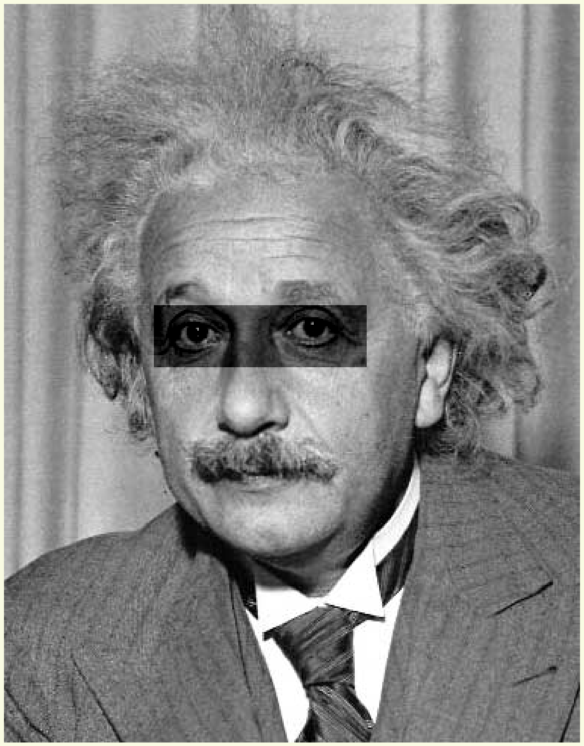


Thresholded Image

Slide Credit: D. Hoeim

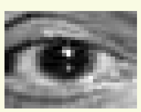
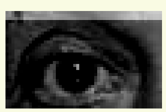
Problem with SSD

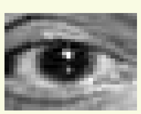
- SSD is sensitive to changes in brightness



Input

1- sqrt(SSD)

( - )^2 = large

( - )^2 = medium

Slide Credit: D. Hoeim

Method 3: Normalized Cross-Correlation

- Goal: find  in image

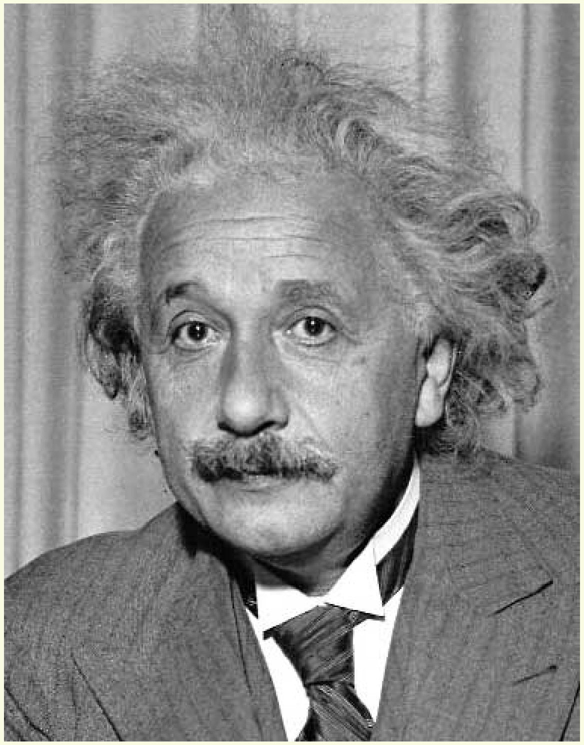
mean template

mean image patch

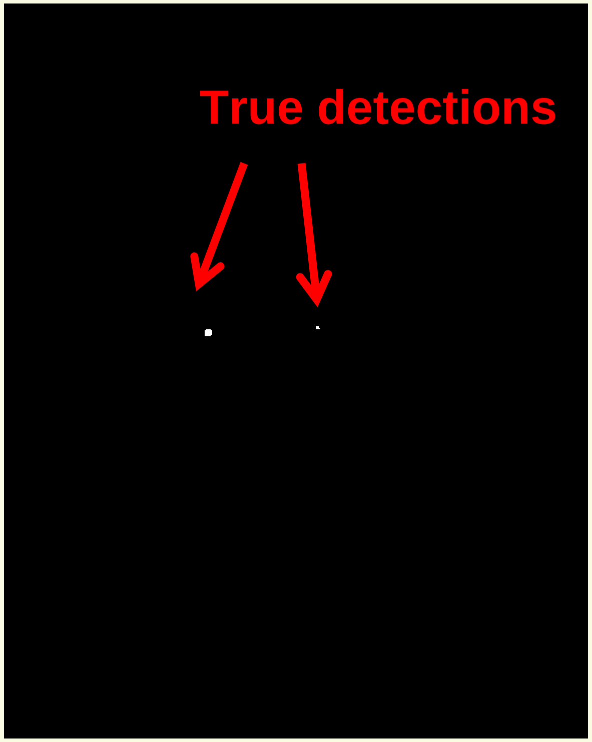
$$h[m,n] = \frac{\sum_{k,l} (g[k,l] - \bar{g})(f[m+k,n+l] - \bar{f}_{m,n})}{\left(\sum_{k,l} (g[k,l] - \bar{g})^2 \sum_{k,l} (f[m+k,n+l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$

Slide Credit: D. Hoeim

Method 3: Normalized Cross-Correlation



Input



Thresholded Image

Normalized X-Correlation

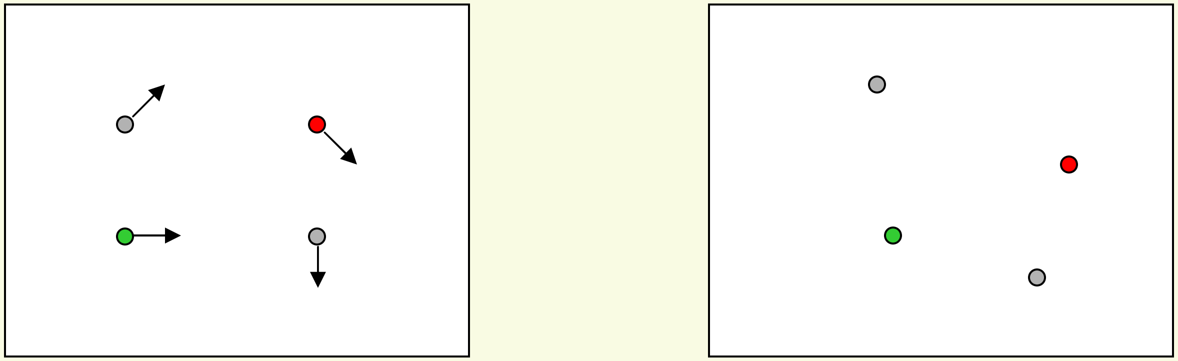
Slide Credit: D. Hoeim

Comparison

- Zero-mean filter: fastest but not a great matcher
- SSD: next fastest, sensitive to overall intensity
- Normalized cross-correlation: slowest, but invariant to local average intensity and contrast

Slide Credit: D. Hoeim

Optical flow

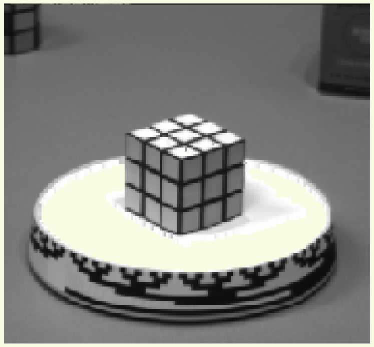


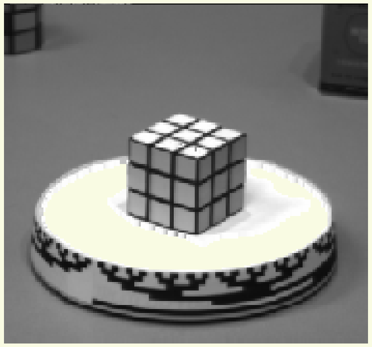
first image I_1 **second image I_2**

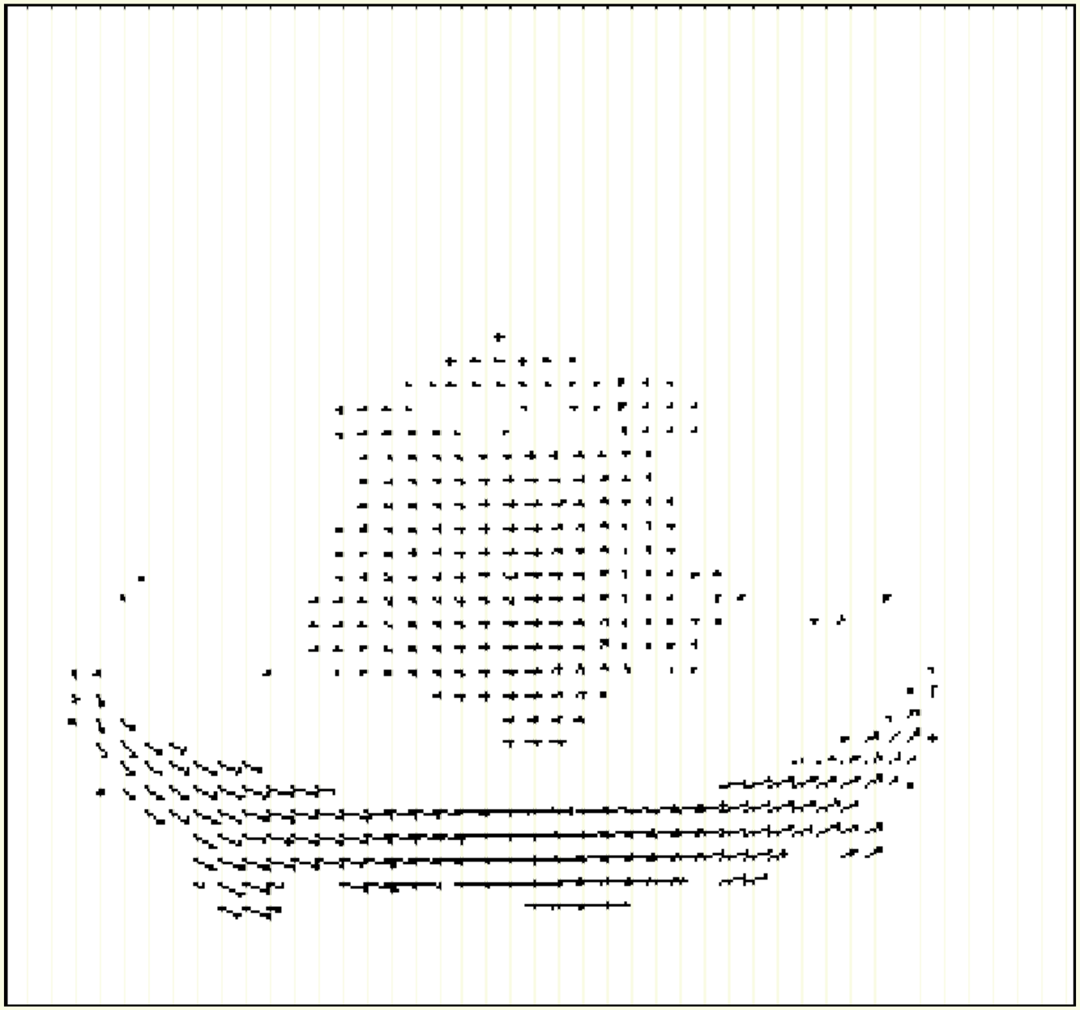
- How to estimate pixel motion from image $\boldsymbol{I_1}$ to image $\boldsymbol{I_2}$?
  - Solve pixel correspondence problem
    - given a pixel in $\boldsymbol{I_1}$, find pixels with similar color in $\boldsymbol{I_2}$

- Key assumptions
  - **color constancy:** a point in $\boldsymbol{I_1}$ looks the same in $\boldsymbol{I_2}$
    - For grayscale images, this is **brightness constancy**
  - **small motion**: points do not move very far
- This is called the **optical flow** problem

Optical Flow Field







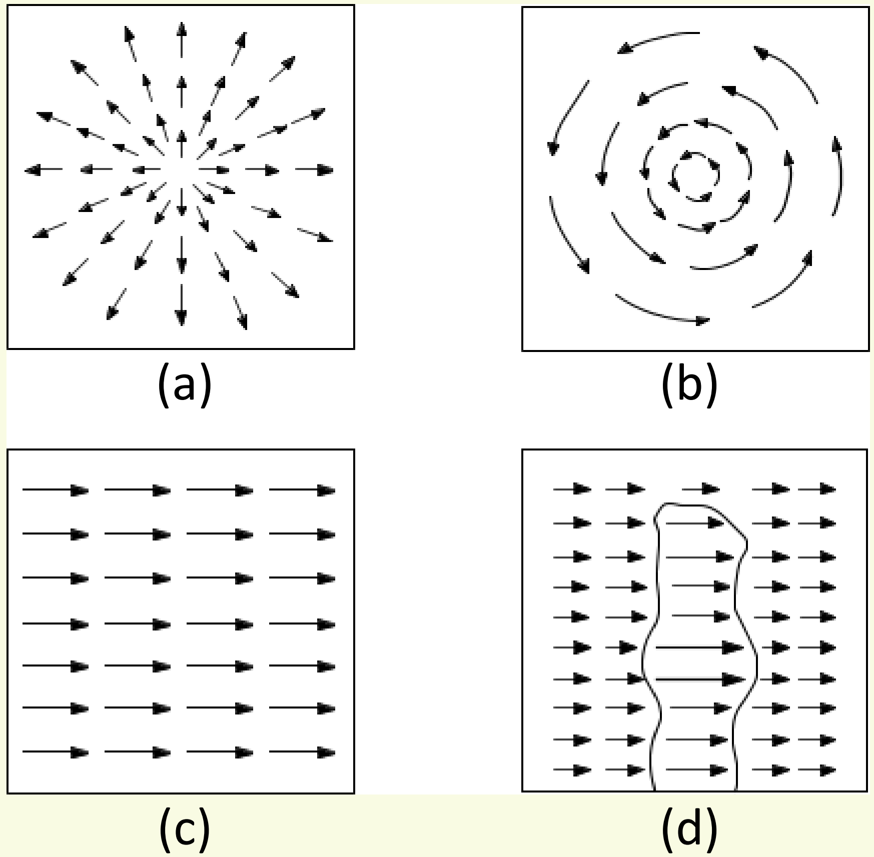
Optical Flow and Motion Field

- Optical flow field 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

Motion Field (MF)

- The MF assigns a velocity vector to each pixel in the image
- These velocities are INDUCED by the RELATIVE MOTION between the camera and the 3D scene
- The MF is the *projection* of the 3D velocities on the image plane

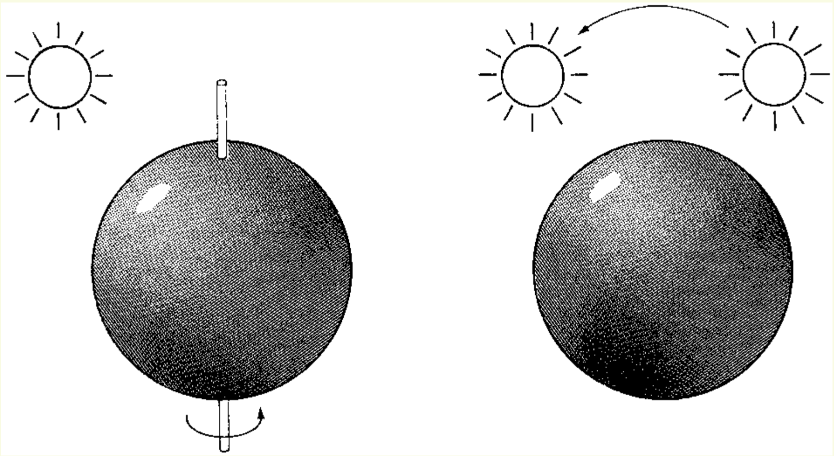
Examples of Motion Fields



(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

- Recall that Optical Flow is the apparent motion of brightness patterns
- We equate Optical Flow Field with Motion Field
- Frequently works, but now always:



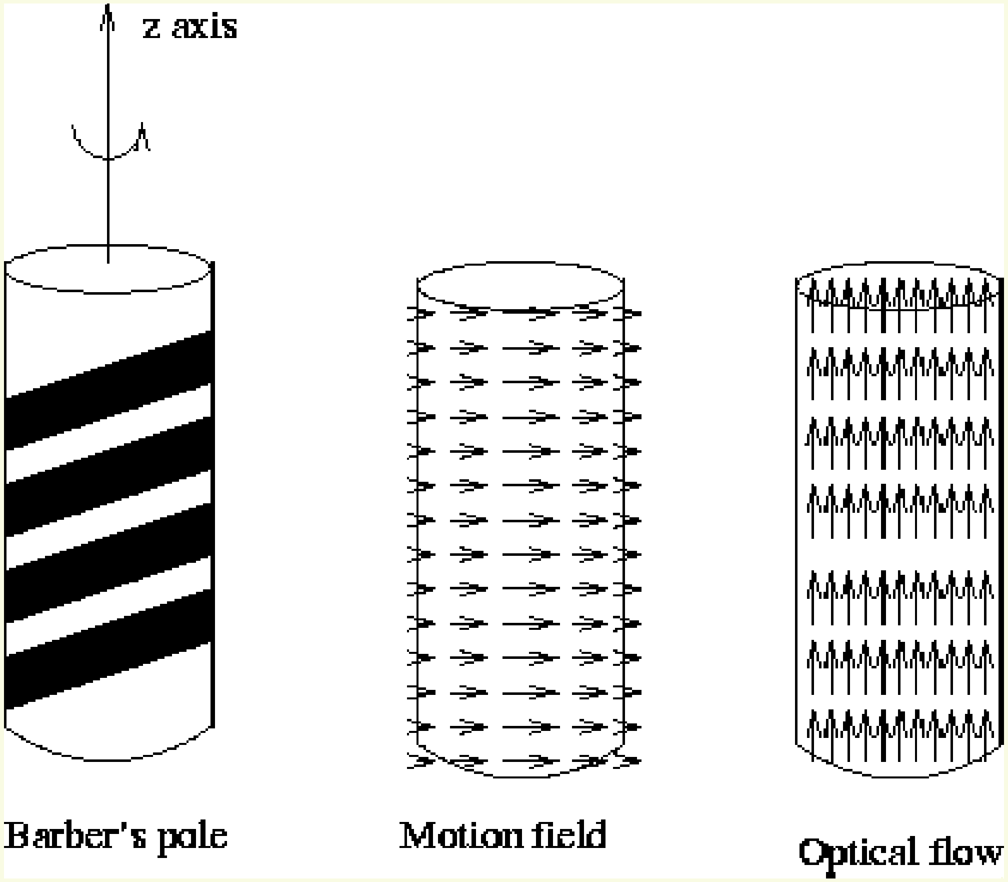
(a) (b)

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

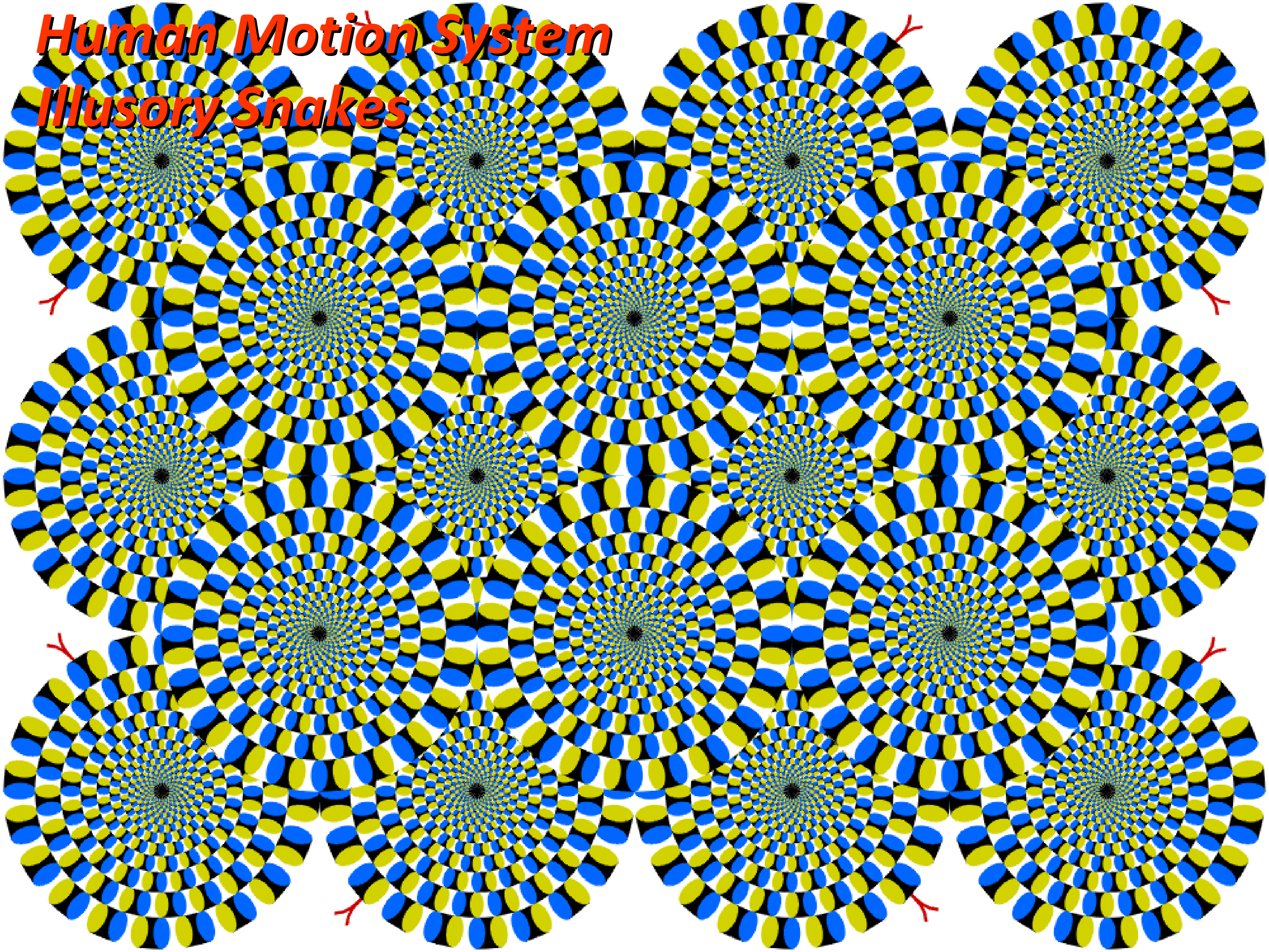
(b) A fixed sphere is illuminated by a moving source—the shading of the image changes. Thus the motion field is zero, but the optical flow field is not

Optical Flow vs. Motion Field

- Often (but not always) optical flow corresponds to the true motion of the scene



Human Motion System
Illusory Snakes



from Gary Bradski and Sebastian Thrun

Computing Optical Flow: Brightness Constancy Equation

- 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 $E(x(t),y(t),t)$ be the brightness at p at time t.
- Brightness Constancy Assumption:
  - As P moves over time, $E(x(t),y(t),t)$ remains constant

Computing Optical Flow: Brightness Constancy Equation

$$E(x(t), y(t), t) = Constant$$

Taking derivative wrt time:

$$\frac{dE(x(t), y(t), t)}{dt} = 0$$

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0$$

Computing Optical Flow: Brightness Constancy Equation

1 equation with 2 unknowns

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0$$

Let

$$\nabla E = \begin{bmatrix} \frac{\partial E}{\partial x} \\ \frac{\partial E}{\partial y} \end{bmatrix}$$

(Frame spatial gradient)

$$v = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix}$$

(optical flow)

and

$$E_t = \frac{\partial E}{\partial t}$$

(derivative across frames)

Computing Optical Flow: Brightness Constancy Equation

- How to get more equations for a pixel?
- Idea: impose additional constraints
  - assume that the flow field is smooth locally
  - i.e. pretend the pixel's neighbors have the same (u,v)
    - If we use a 5x5 window, that gives us 25 equations per pixel!

$$\boldsymbol{E_t(p_i) + \nabla E(p_i) \cdot [u \quad v] = 0}$$

$$\begin{bmatrix} E_x(p_1) & E_y(p_1) \\ E_x(p_2) & E_y(p_2) \\ \vdots & \vdots \\ E_x(p_{25}) & E_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} E_t(p_1) \\ E_t(p_2) \\ \vdots \\ E_t(p_{25}) \end{bmatrix}$$

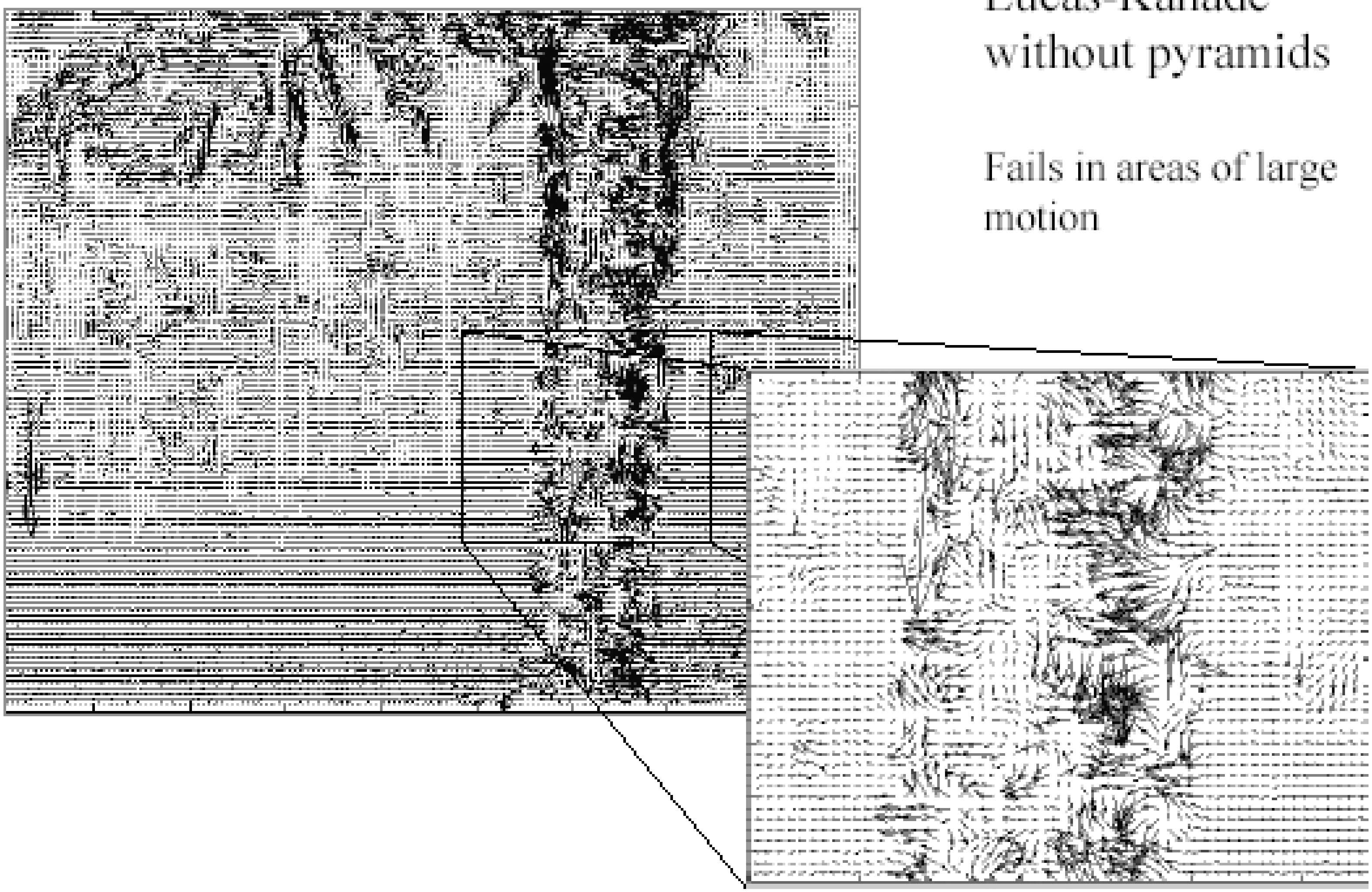
matrix ***E*** 25x2 — vector ***d*** 2x1 — vector ***b*** 25x1

Video Sequence

*

* Picture from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



Lucas-Kanade without pyramids

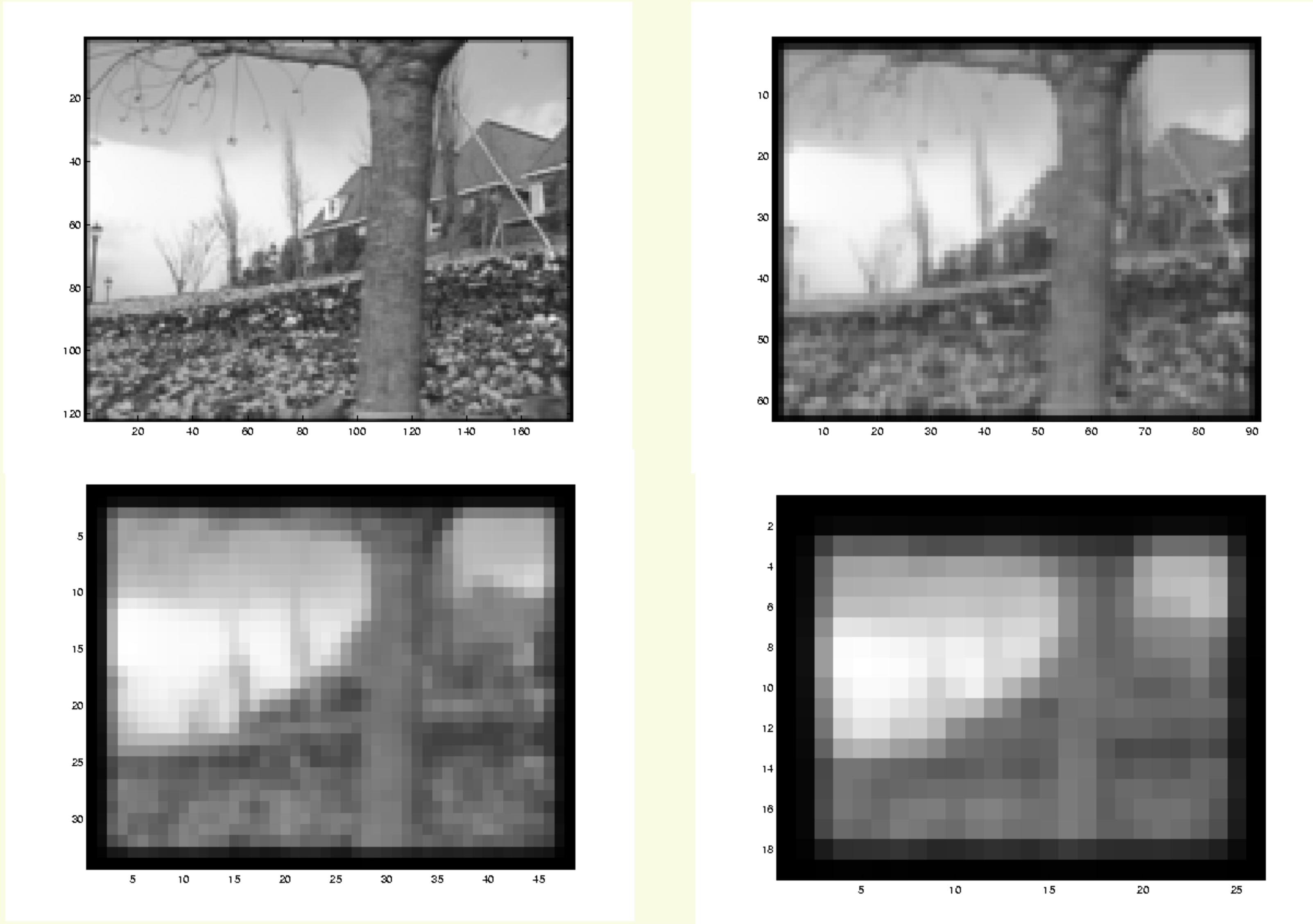
Fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

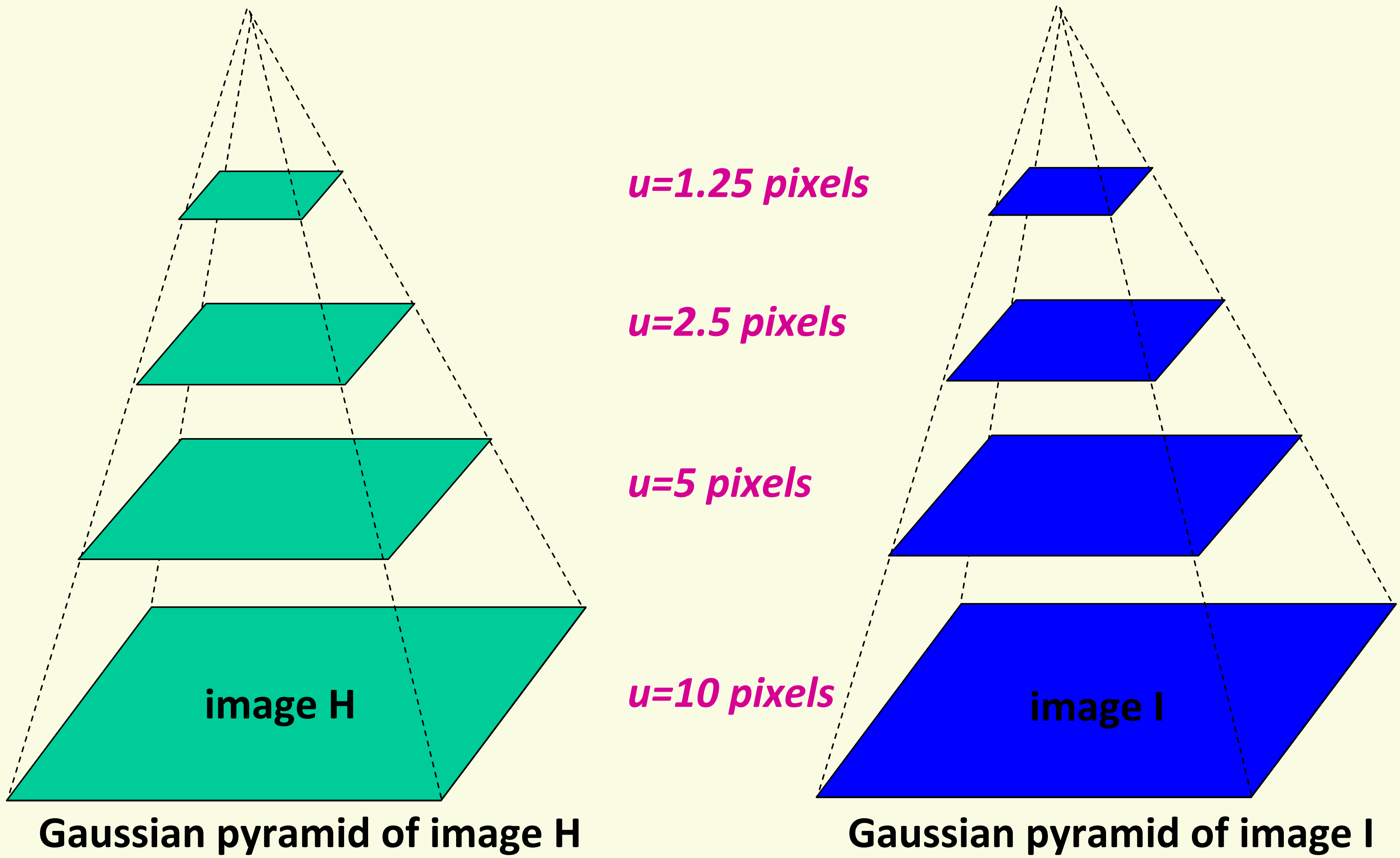
Revisiting the small motion assumption

- 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

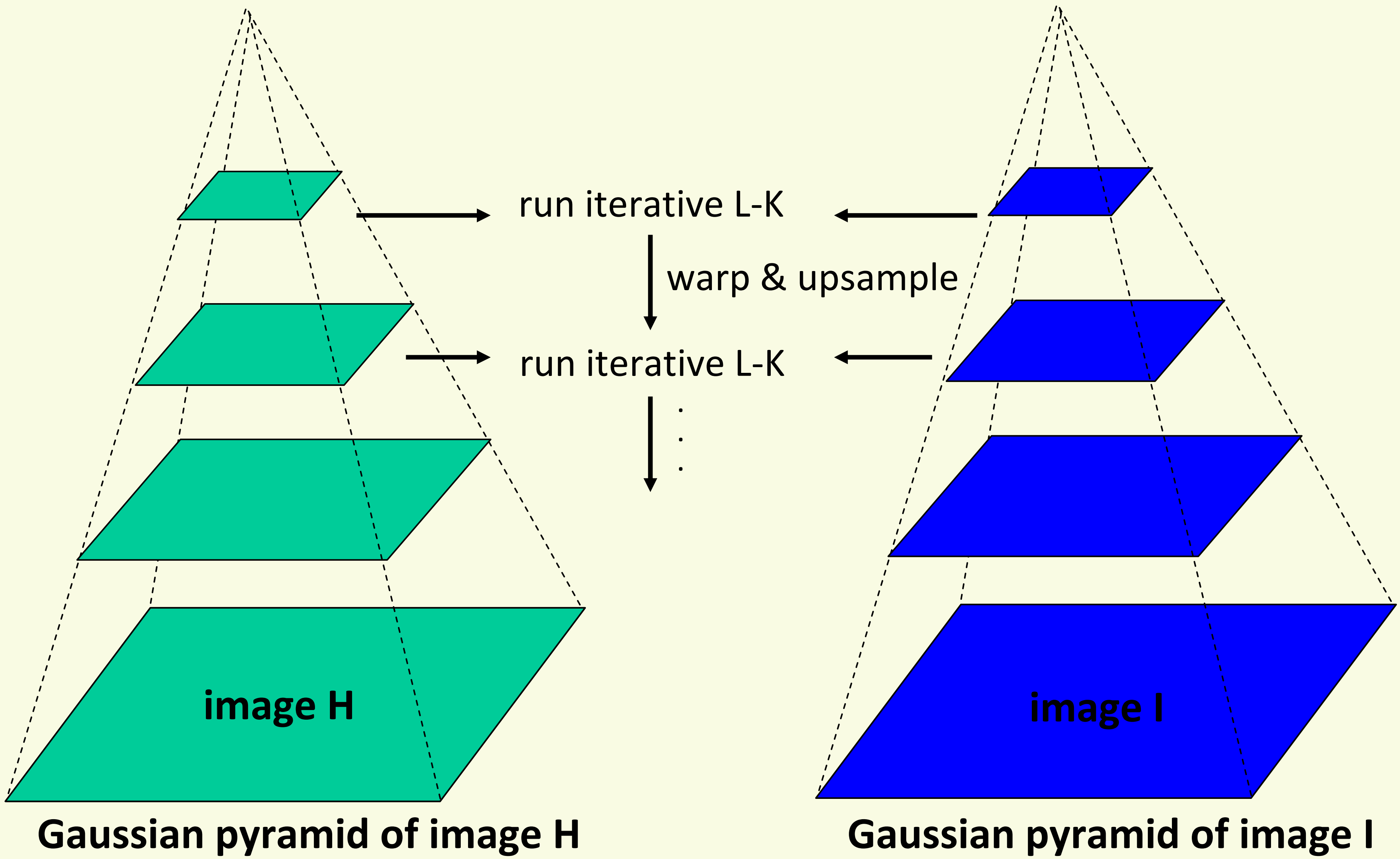


Iterative Refinement

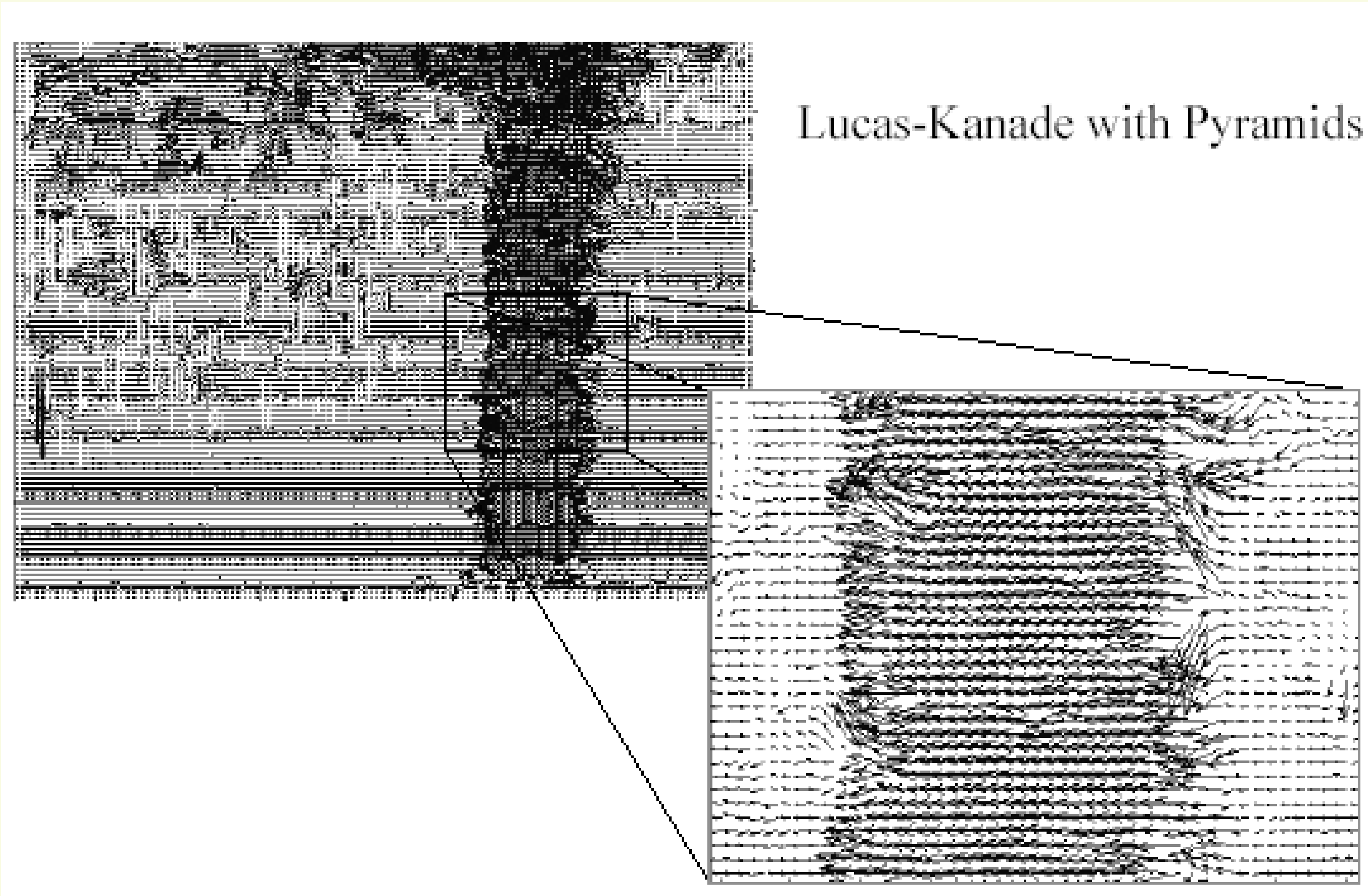
- Iterative Lukas-Kanade Algorithm
  1. Estimate velocity at each pixel by solving Lucas-Kanade equations
  2. Warp H towards I using the estimated flow field

     - use image warping techniques
  3. Repeat until convergence

Coarse-to-fine optical flow estimation



Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Modern OF Algorithms

- A lot of development in the past 10 years
- See Middlebury Optical Flow Evaluation
  - http://vision.middlebury.edu/flow/
  - Dataset with ground truth