CSE 152: Computer Vision

Hao Su

Lecture 1: Filters and Features

Image filtering

- Image filtering:
  - Compute function of local neighborhood at each position

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

Image filtering

- Image filtering:
  - Compute function of local neighborhood at each position

h=output f=filter I=image

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

2d coords=k,l 2d coords=m,n

[ ] [ ] [ ]

Example: box filter

$$f[\cdot,\cdot]$$

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

Slide credit: David Lowe (UBC)

Image filtering

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

$I[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

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

Image filtering

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

$I[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10							

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

Credit: S. Seitz

Image filtering

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

$I[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10	20						

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

Credit: S. Seitz

Image filtering

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

$I[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10	20	30					

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

Credit: S. Seitz

Image filtering

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

$I[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10	20	30	30				

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$I[\cdot,\cdot]$

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	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

$h[\cdot,\cdot]$

	0	10	20	30	30				
				?					

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

Credit: S. Seitz

Image filtering

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

$I[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10	20	30	30				
						?			
				50					

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

Credit: S. Seitz

Image filtering

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

$I[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	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	

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

Credit: S. Seitz

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

$$f[\cdot,\cdot]$$

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

Slide credit: David Lowe (UBC)

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

- Why does it sum to one?

$$f[\cdot,\cdot]$$

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

Slide credit: David Lowe (UBC)

James Hays

Smoothing with box filter

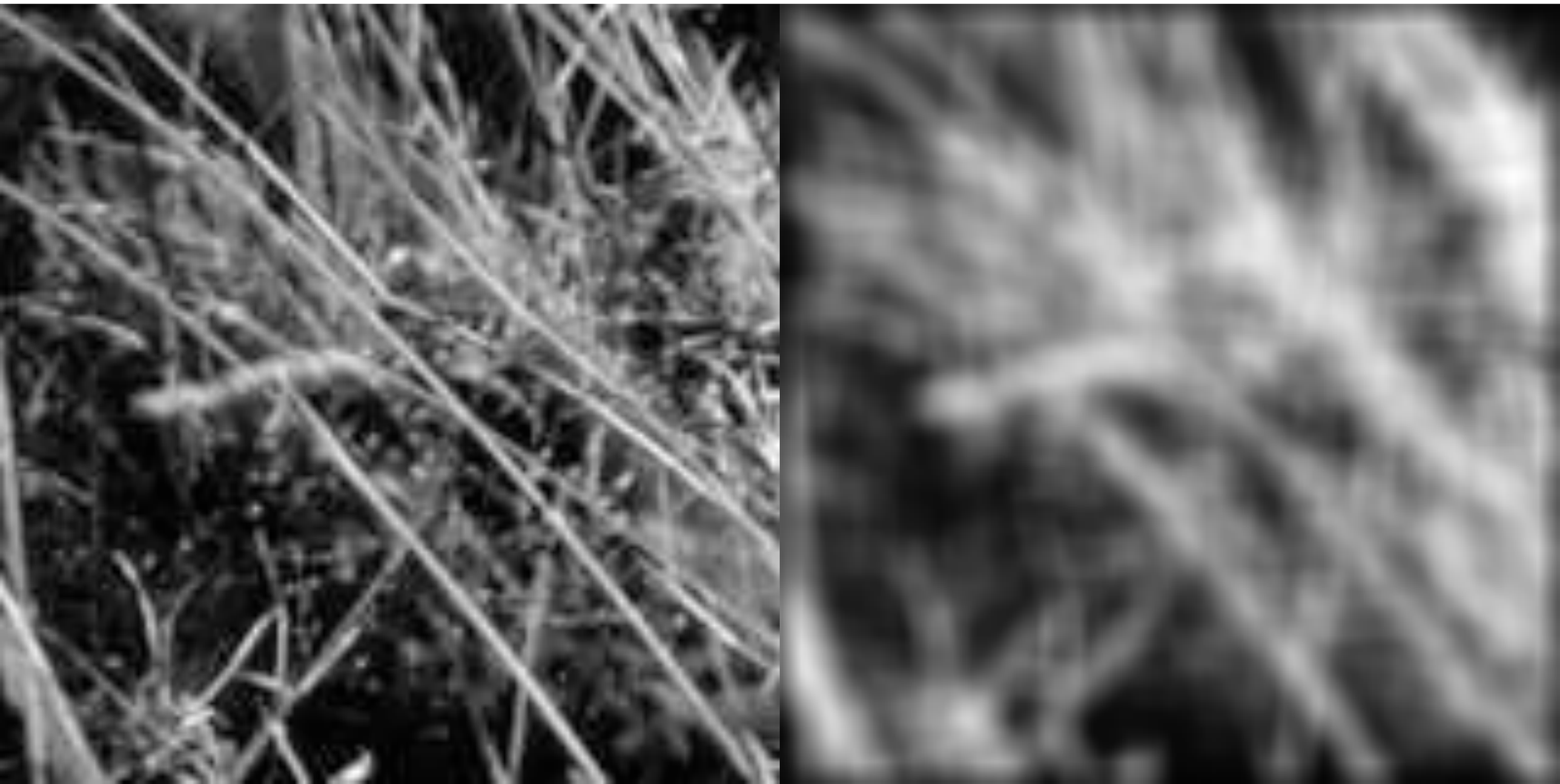


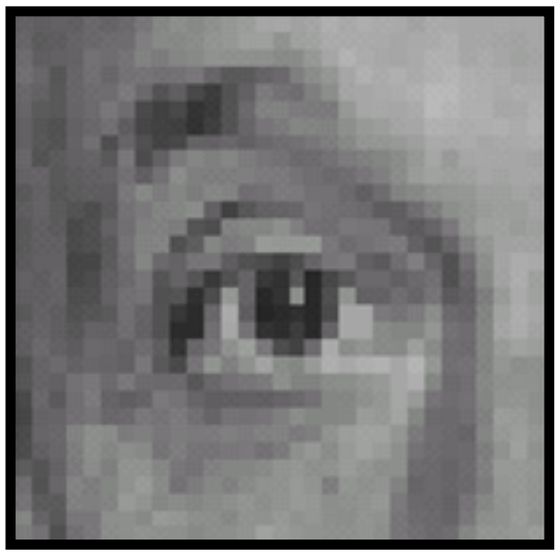
Image filtering

- Image filtering:
  - Compute function of local neighborhood at each position

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

- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

1. Practice with linear filters



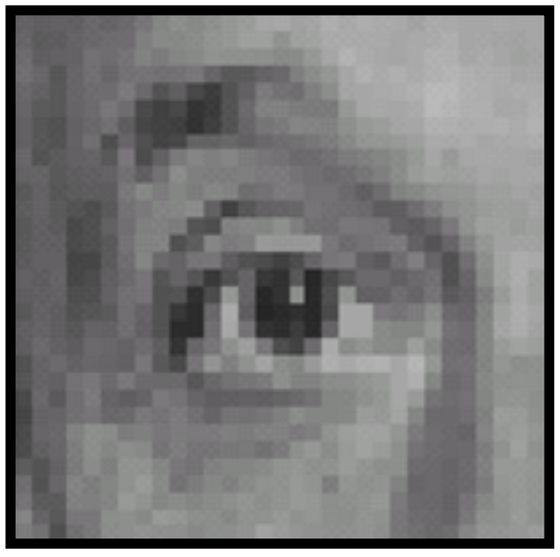
Original

0	0	0
0	1	0
0	0	0

?

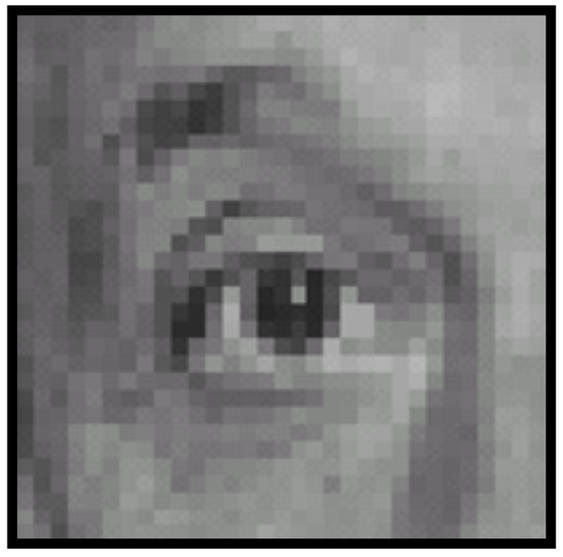
Source: D. Lowe

1. Practice with linear filters



Original

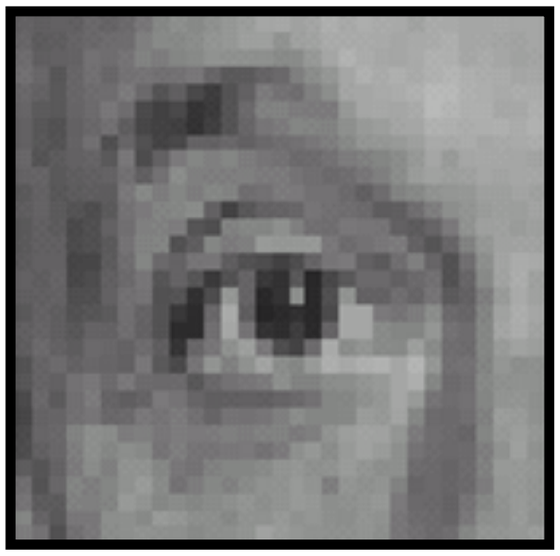
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

2. Practice with linear filters



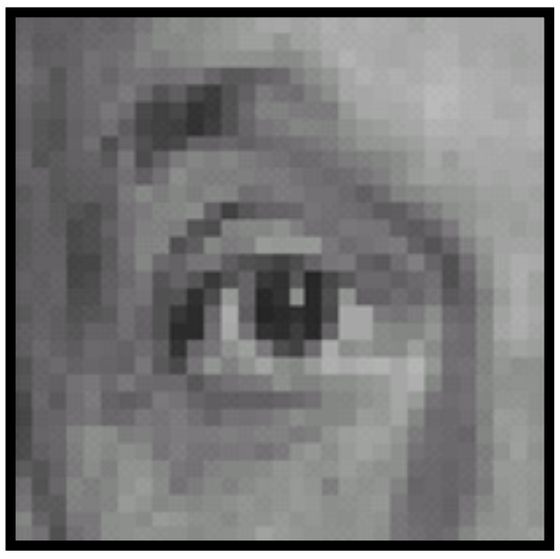
Original

0	0	0
0	0	1
0	0	0

?

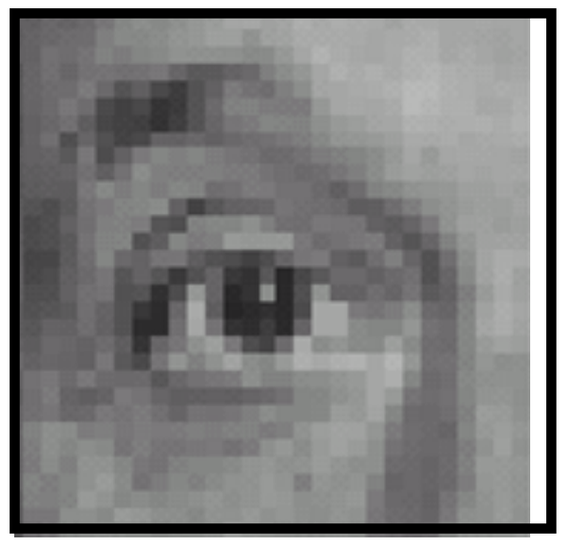
Source: D. Lowe

2. Practice with linear filters



Original

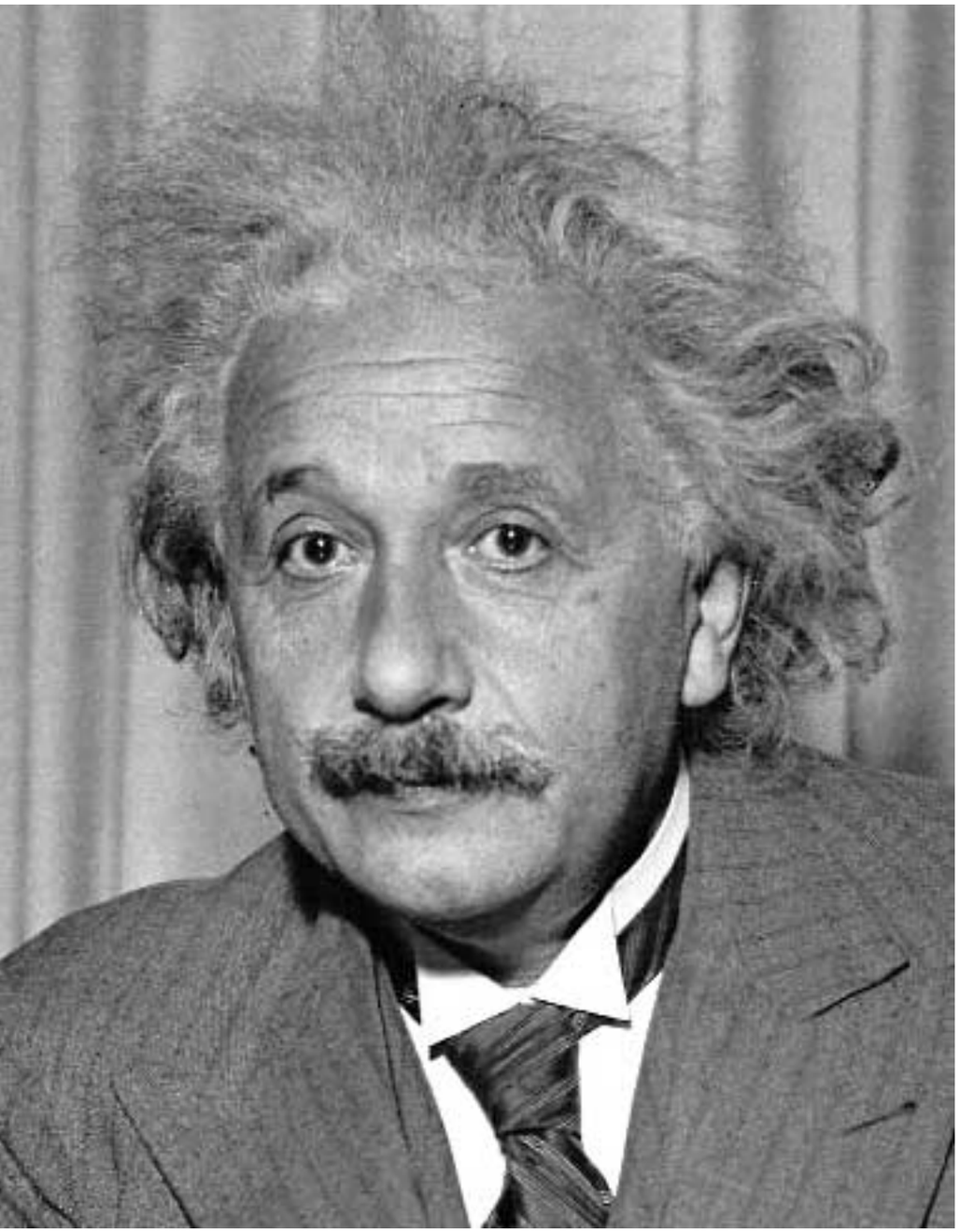
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

3. Practice with linear filters



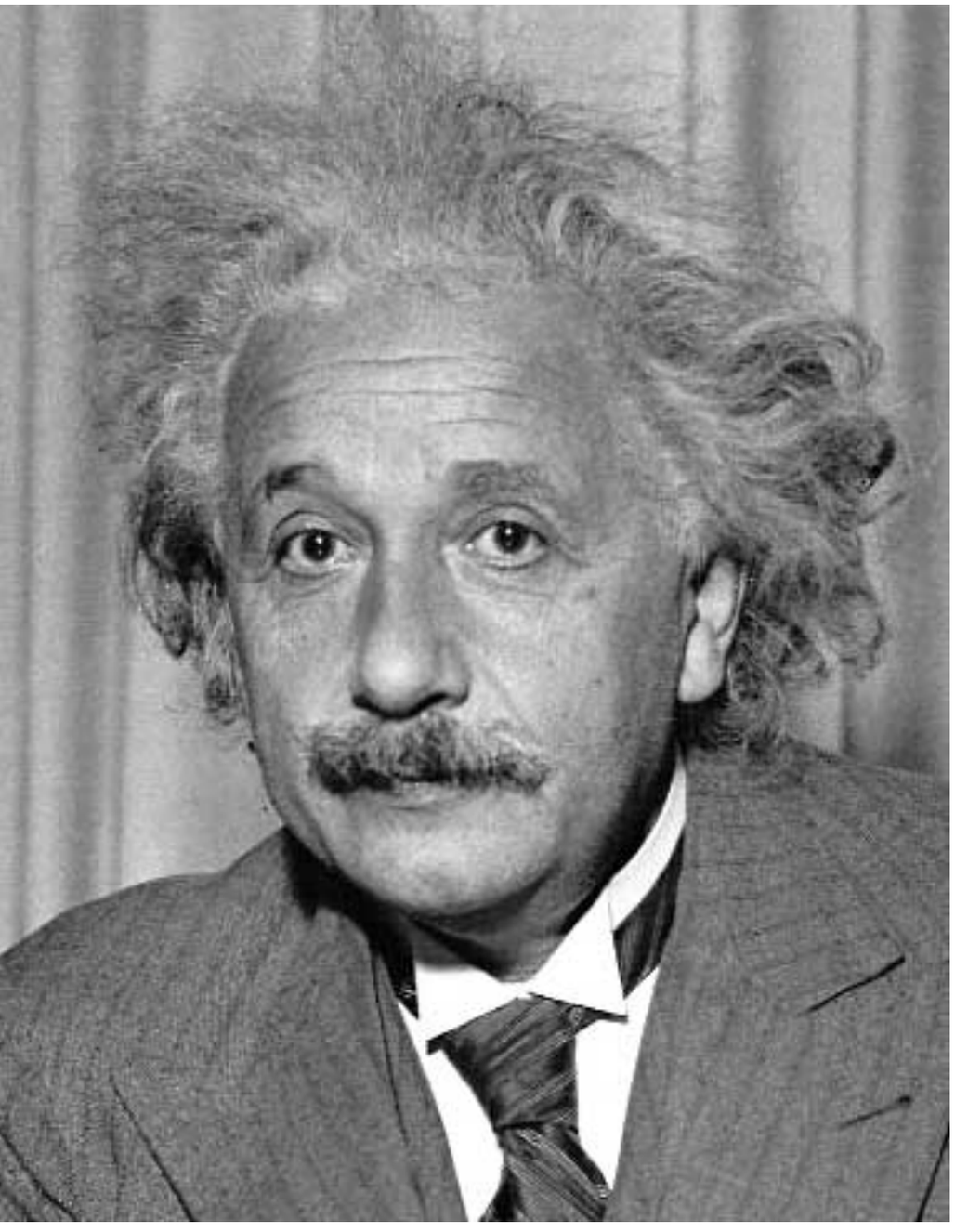
1	0	-1
2	0	-2
1	0	-1

Sobel

Vertical Edge
(absolute value)

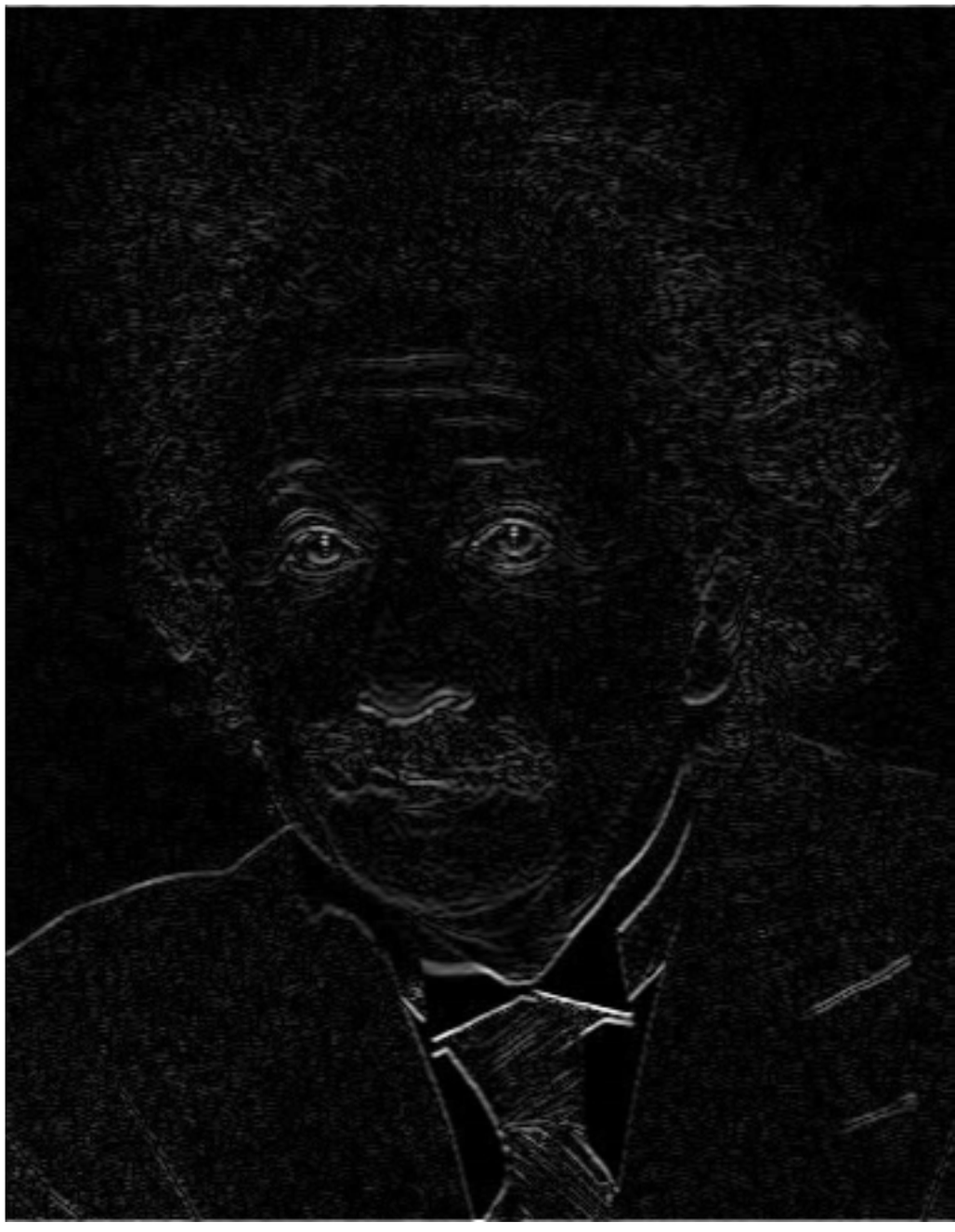
David Lowe

3. Practice with linear filters



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

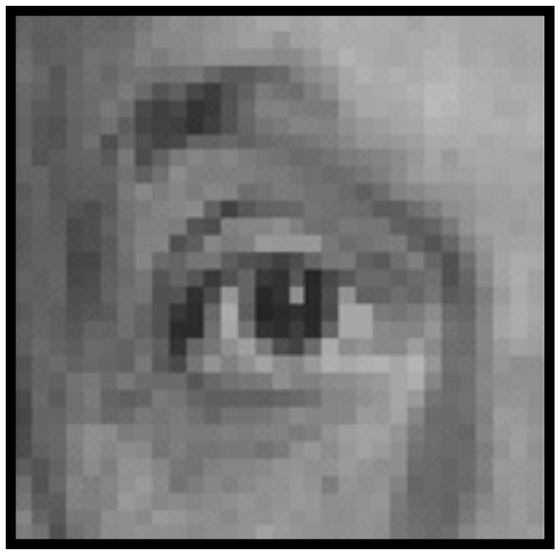
Sobel



Horizontal Edge
(absolute value)

David Lowe

4. Practice with linear filters



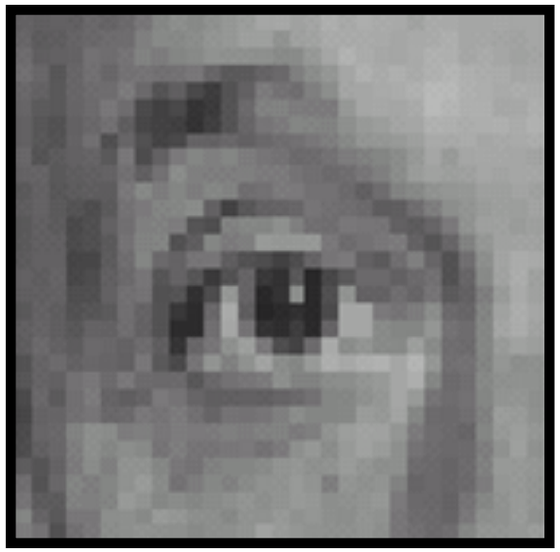
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

(Note that filter sums to 1)

Source: D. Lowe

4. Practice with linear filters



Original

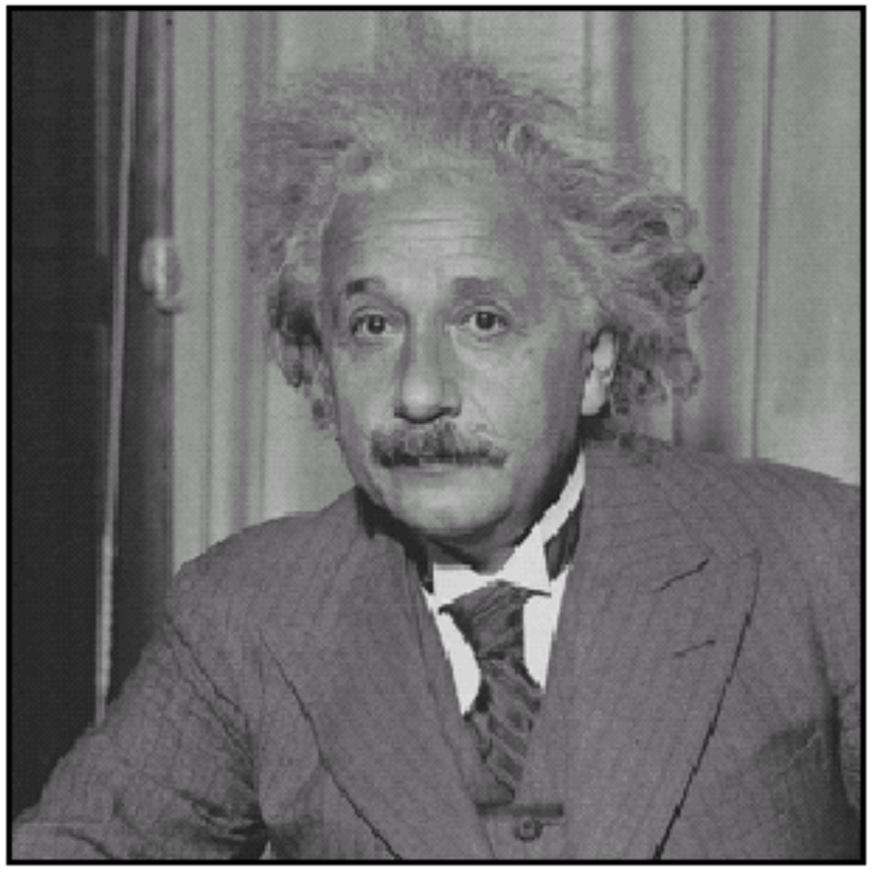
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

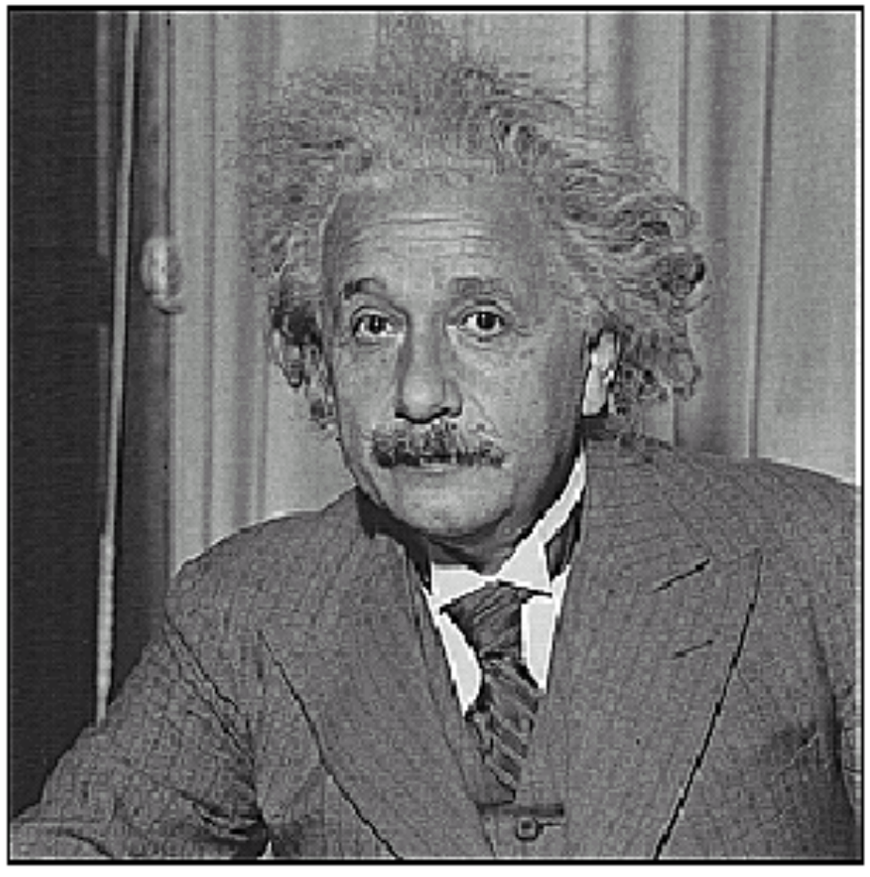
- Accentuates differences with local average

Source: D. Lowe

4. Practice with linear filters



before



after

Source: D. Lowe

Correlation and Convolution

- 2d correlation

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

Correlation and Convolution

- 2d correlation

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

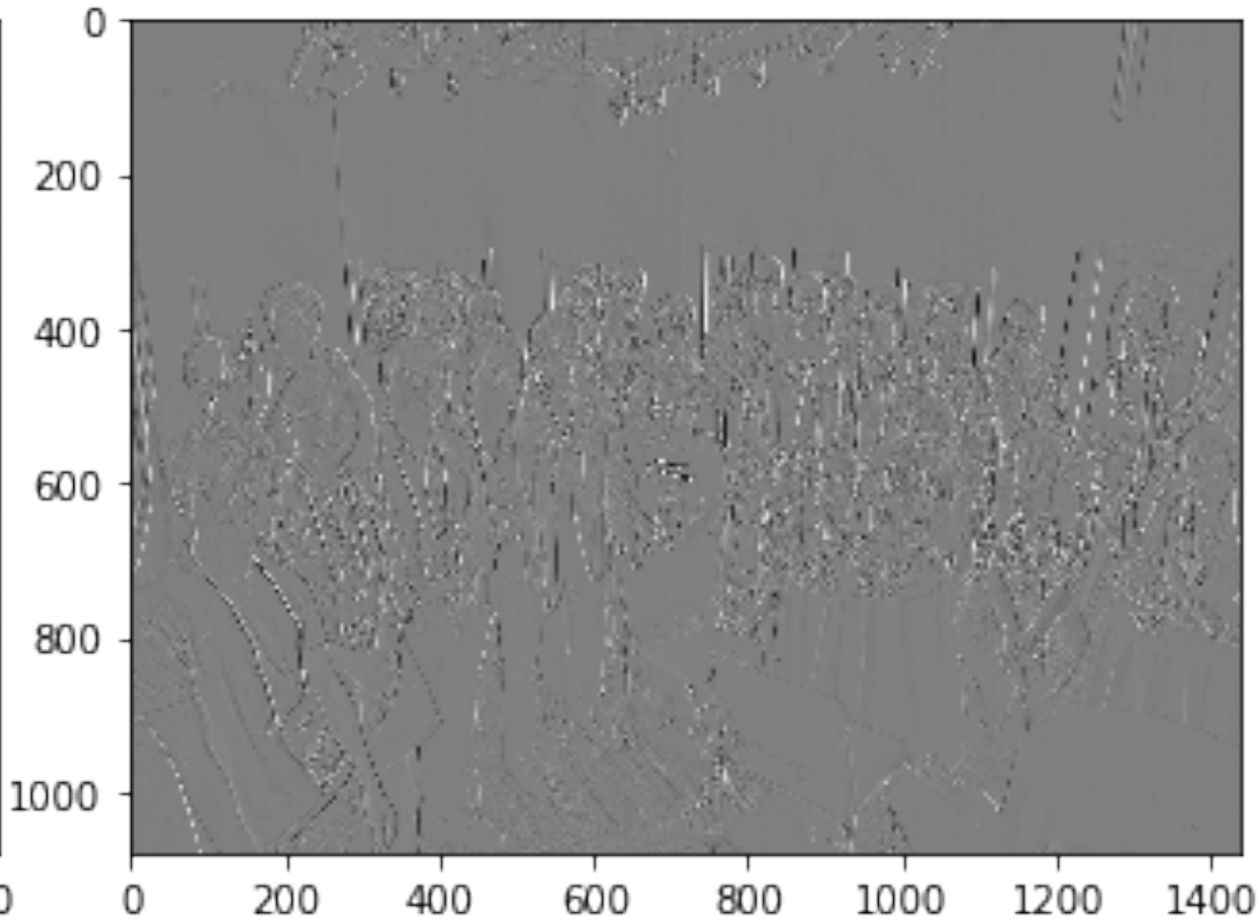
Example code

```
import numpy as np
import scipy.signal as ss
import matplotlib.pyplot as plt

input = plt.imread('leo.jpeg') # input is H*W*3
input = input.mean(axis=-1) # input now becomes H*W
kernel = np.array([[-1, 0, 1]])
kernel = kernel[::-1, ::-1] # flip the kernel to make it convolution
out = ss.convolve2d(input, kernel, mode="same")

plt.figure(); plt.imshow(input, cmap="gray")
plt.figure(); plt.imshow(out, cmap="gray")
```


0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400


0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400

Convolution properties

Commutative: a * b = b * a

- Conceptually no difference between filter and signal
- But particular filtering implementations might break this equality, e.g., image edges

Associative: a * (b * c) = (a * b) * c

- Often apply several filters one after another: (((a * b_1) * b_2) * b_3)
- This is equivalent to applying one filter: a * (b_1 * b_2 * b_3)

Source: S. Lazebnik

Convolution properties

Commutative: a * b = b * a

- Conceptually no difference between filter and signal
- But particular filtering implementations might break this equality, e.g., image edges

Associative: a * (b * c) = (a * b) * c

- Often apply several filters one after another: (((a * b_1) * b_2) * b_3)
- This is equivalent to applying one filter: a * (b_1 * b_2 * b_3)

Source: S. Lazebnik

Convolution properties

- Commutative: a * b = b * a
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality, e.g., image edges

- Associative: a * (b * c) = (a * b) * c
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

- Distributes over addition: a * (b + c) = (a * b) + (a * c)

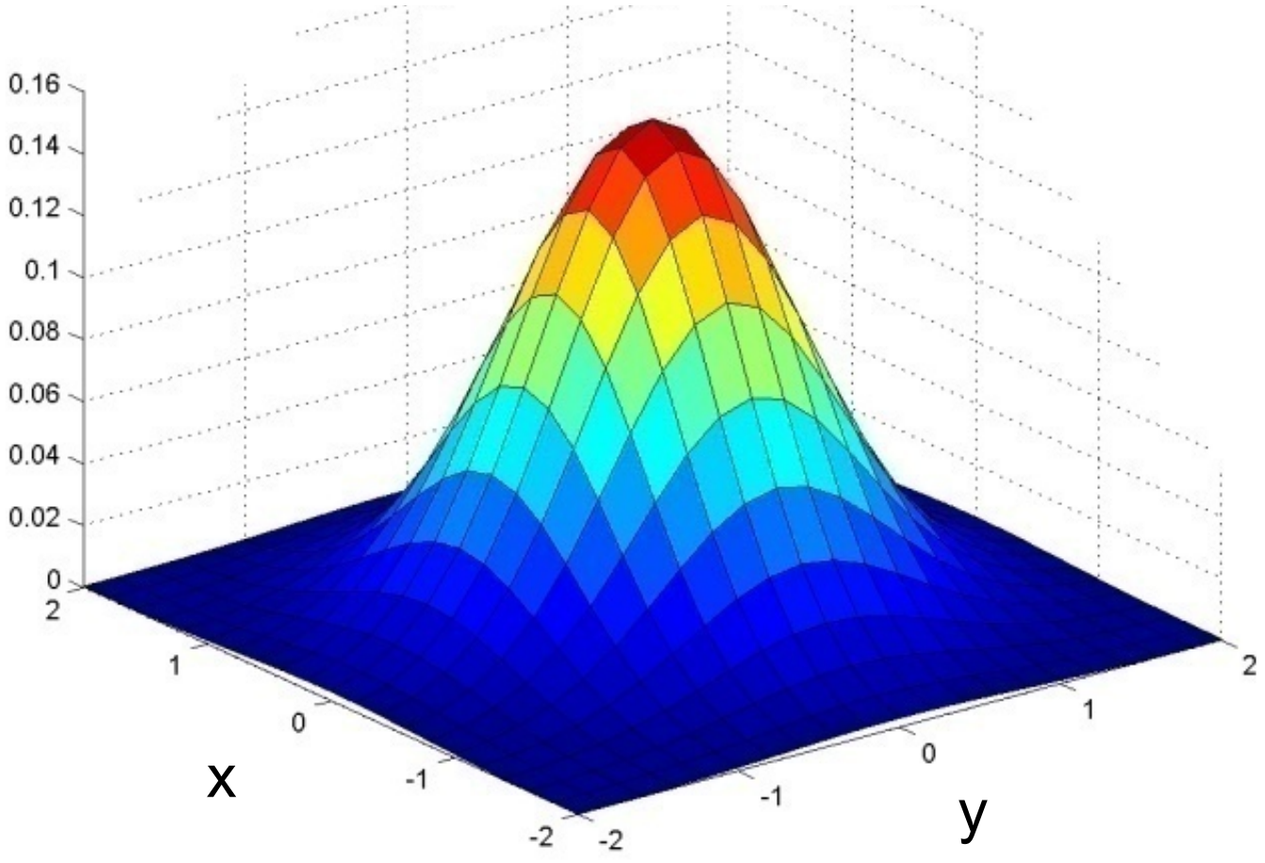
- Scalars factor out: ka * b = a * kb = k (a * b)

- Identity: unit impulse e = [0, 0, 1, 0, 0], a * e = a

Source: S. Lazebnik

Important filter: Gaussian

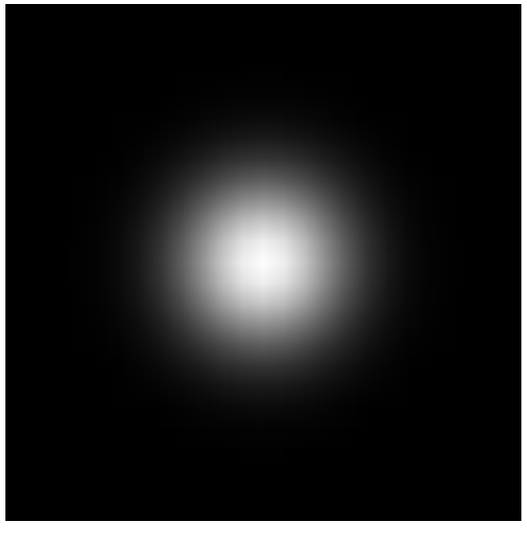
- Weight contributions of neighboring pixels by nearness



x

y				
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

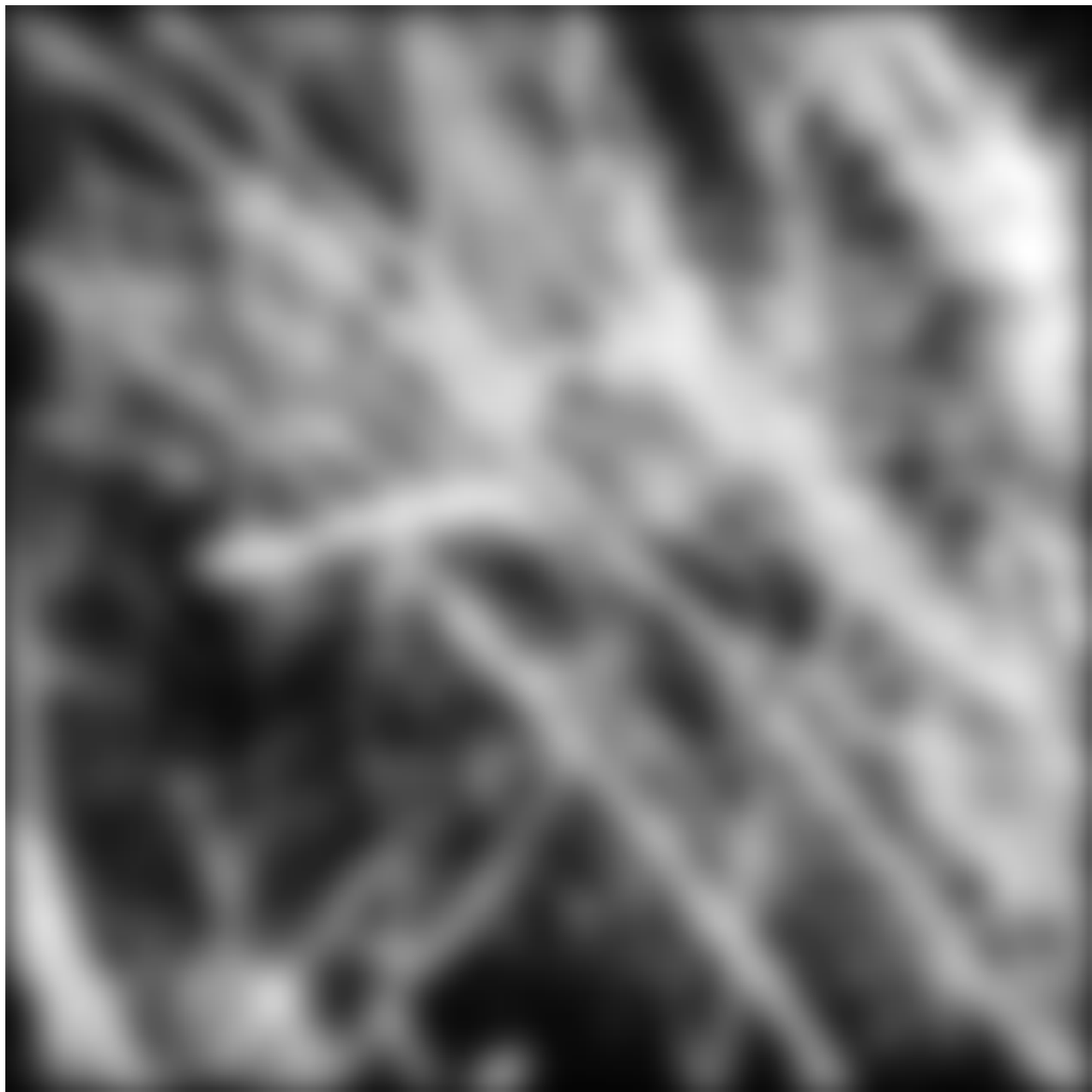
5 x 5, $\sigma = 1$



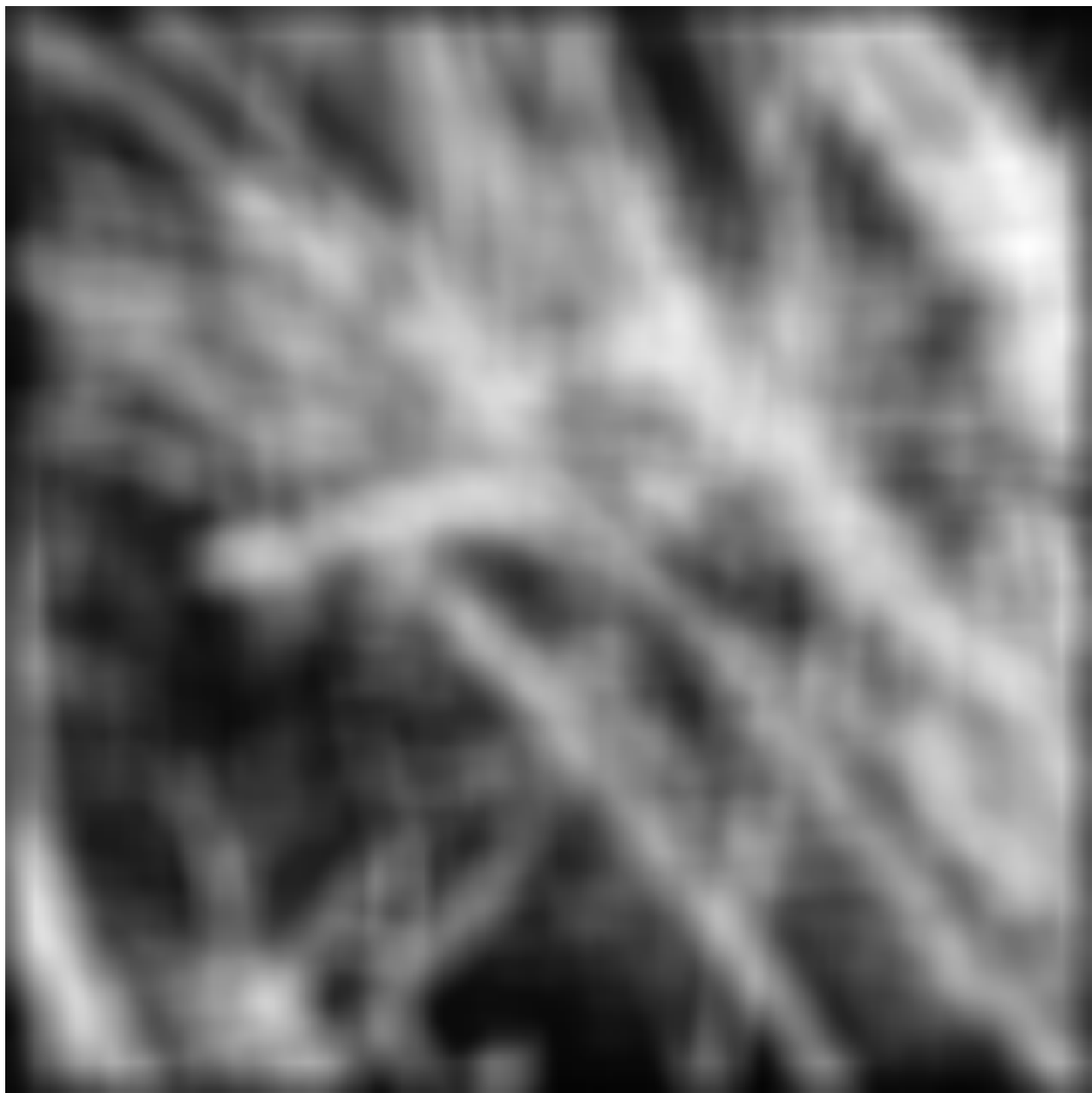
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

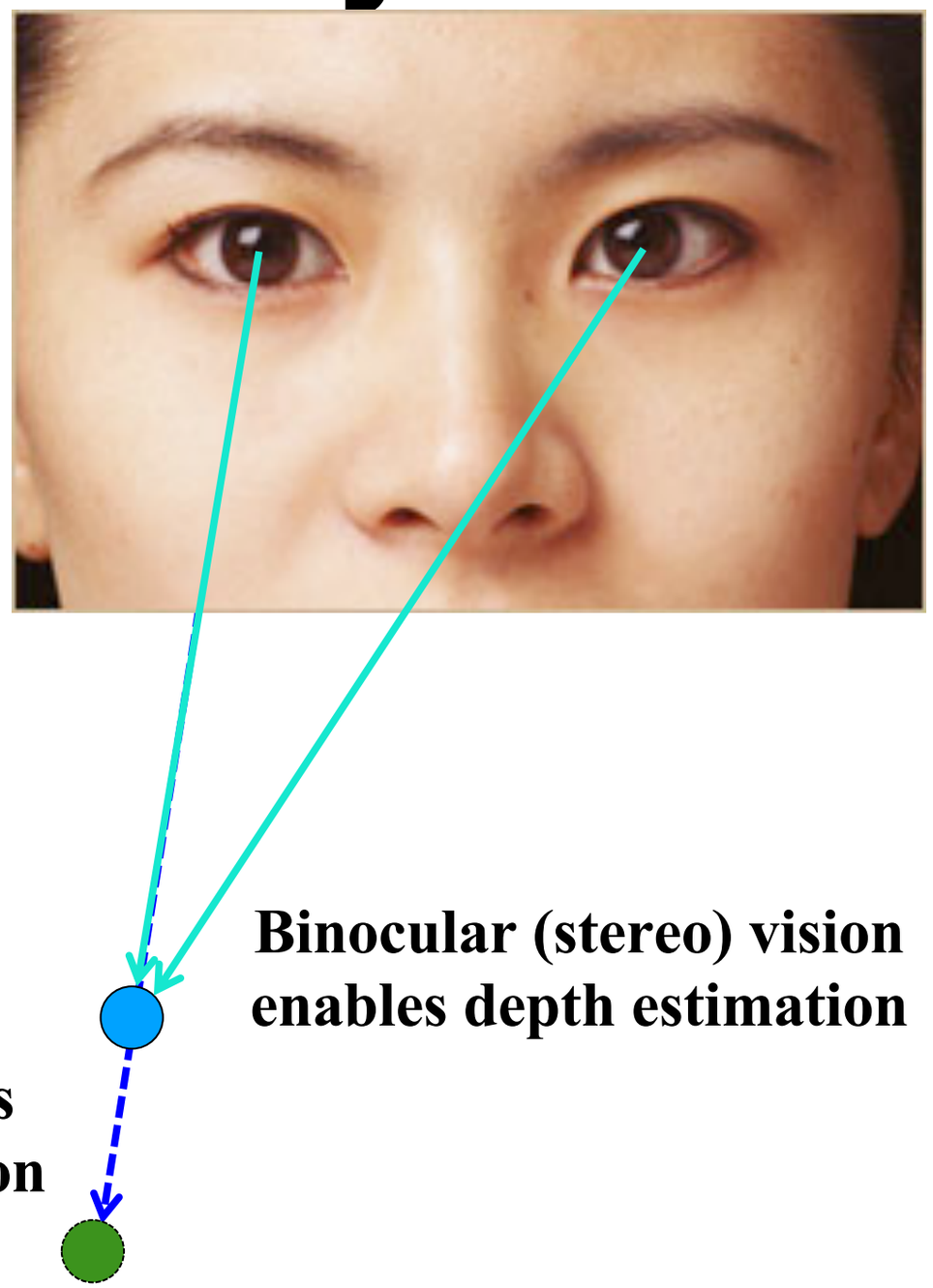
Smoothing with Gaussian filter



Smoothing with box filter



Why Do We Have Two Eyes?



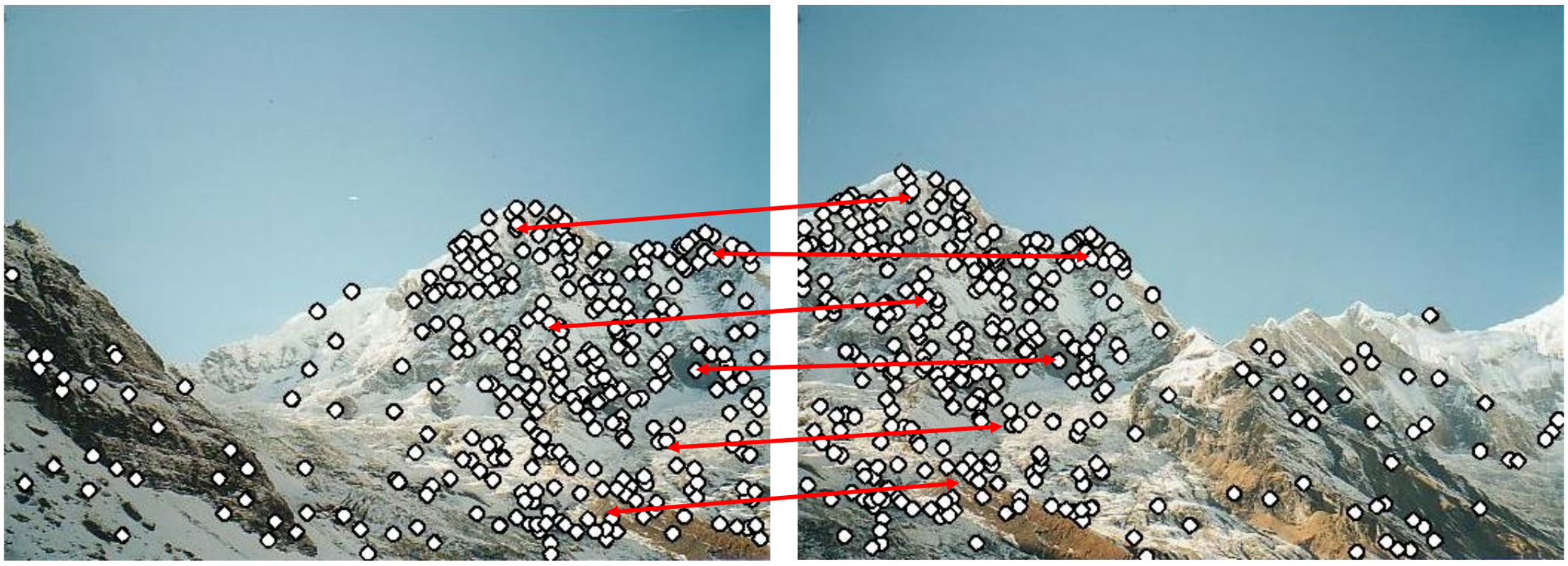
Correspondence estimation

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Slide courtesy: Rick Szeliski

Correspondence estimation

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract features
Step 2: match features

Slide courtesy: Rick Szeliski

Correspondence estimation

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features
Step 2: match features
Step 3: align images

Slide courtesy: Rick Szeliski

Image matching

by [Diva Sian]

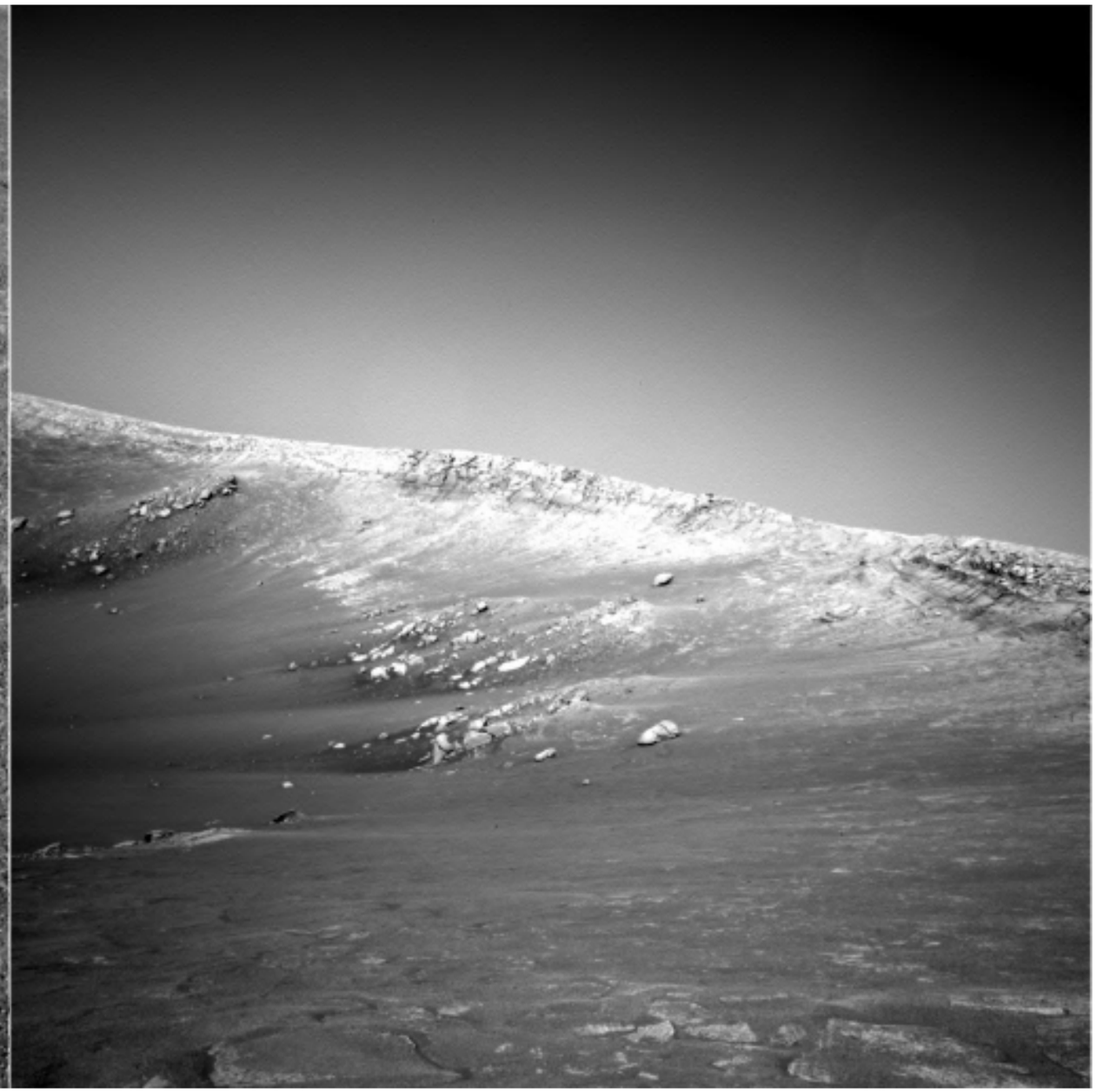
by [swashford]

Harder case

by [Diva Sian]

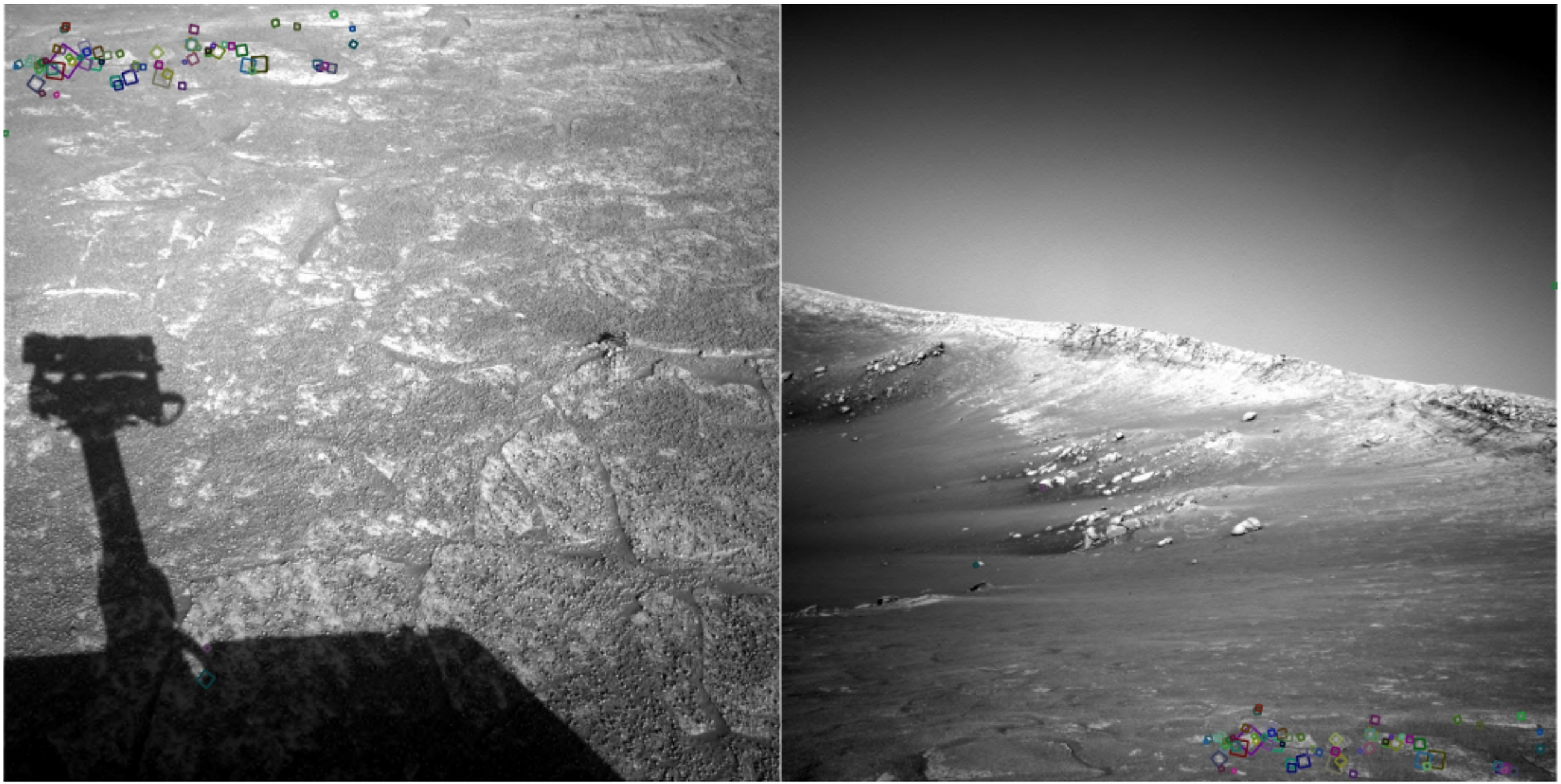
by [scgbt]

Harder still?



NASA Mars Rover images

Answer below (look for tiny colored squares…)

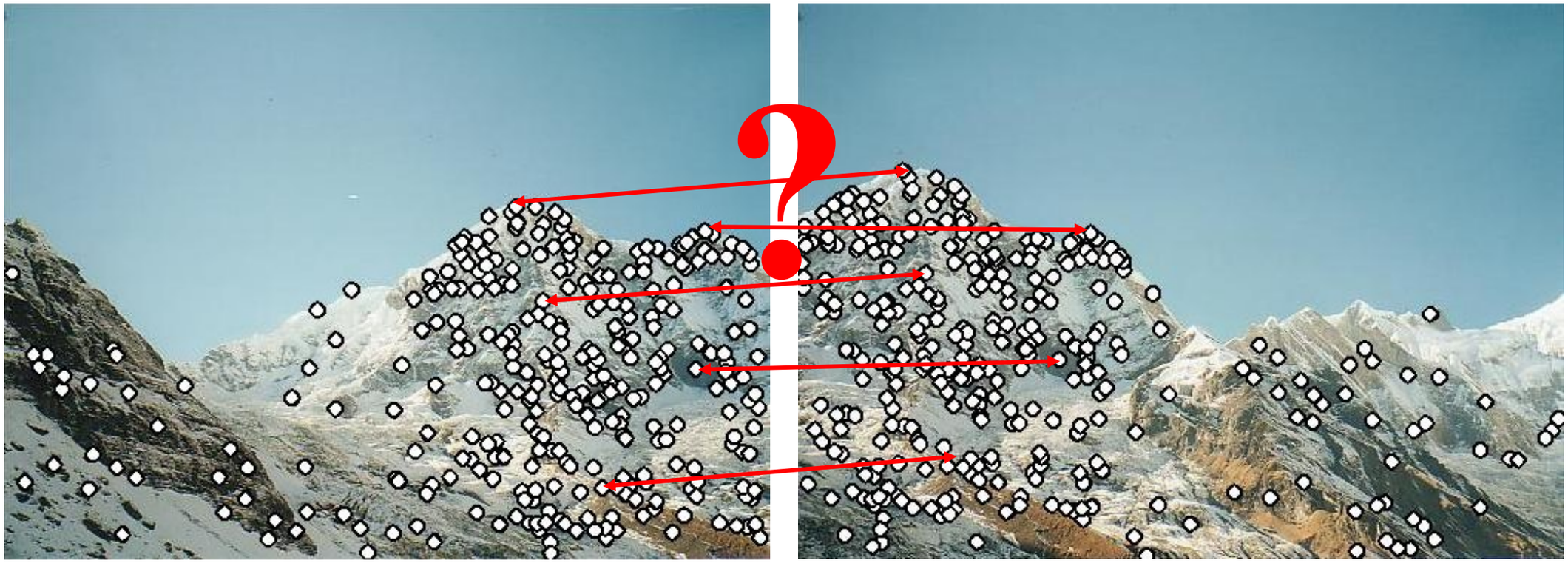


NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Feature descriptors

We know how to detect good points

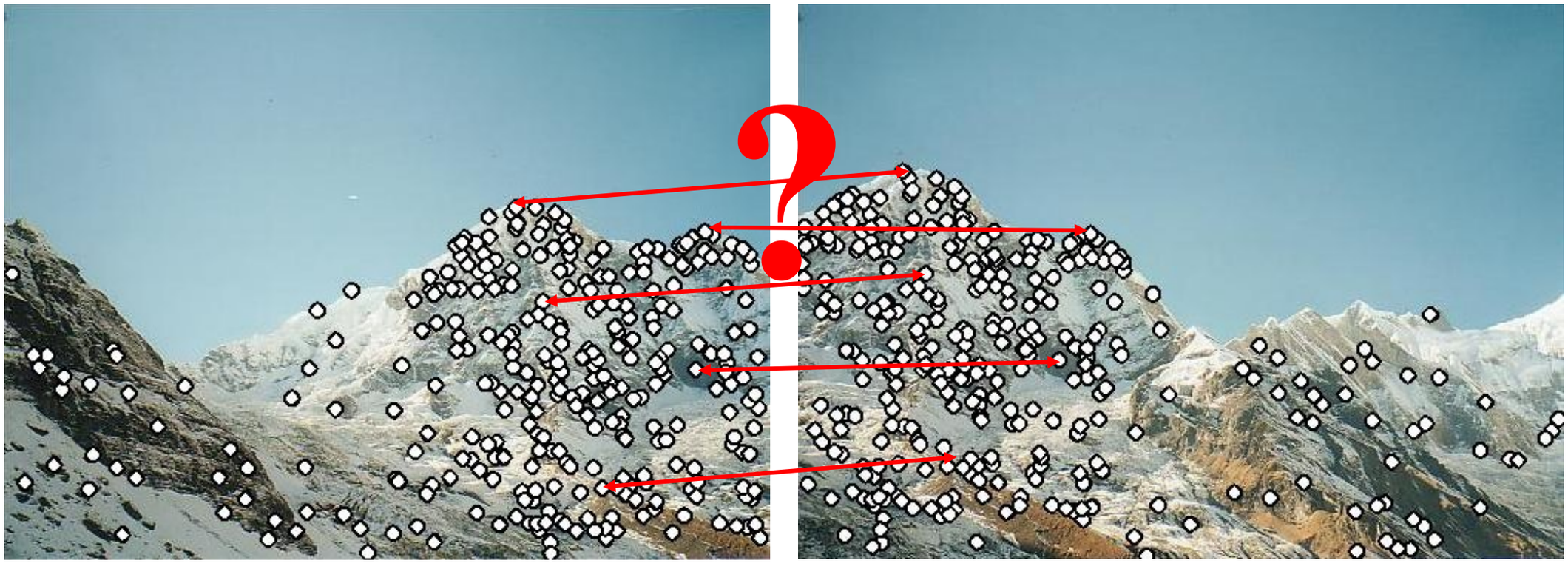
Next question: **How to match them?**



Slide courtesy:
Rick Szeliski

Feature descriptors

We know how to detect good points
Next question: **How to match them?**

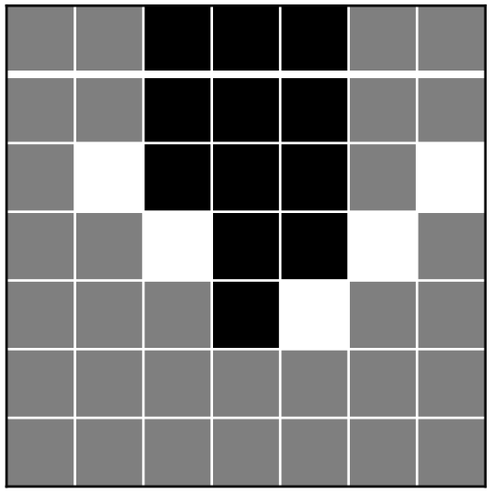


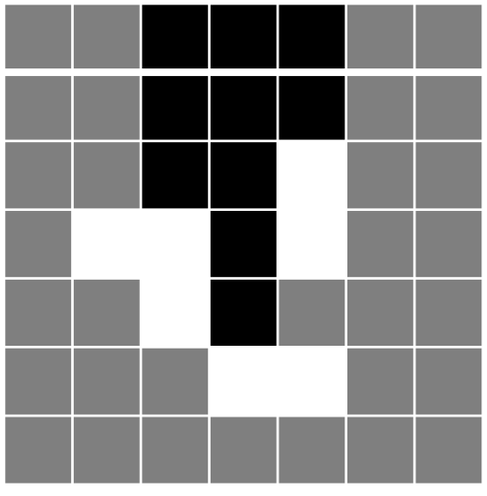
Lots of possibilities (this is a popular research area)

- Simple option: match square windows around the point
- State of the art approach: SIFT
  - David Lowe, UBC http://www.cs.ubc.ca/~lowe/keypoints/

Slide courtesy: Rick Szeliski

Simple matching methods





- SSD (Sum of Squared Differences)

$$\sum_{x,y} |W_1(x,y) - W_2(x,y)|^2$$

- NCC (Normalized Cross Correlation)

$$\frac{\sum_{x,y}(W_1(x,y) - \overline{W_1})(W_2(x,y) - \overline{W_2})}{\sigma_{W_1}\sigma_{W_2}}$$

where $\overline{W_i} = \frac{1}{n}\sum_{x,y} W_i \quad \sigma_{W_i} = \sqrt{\frac{1}{n}\sum_{x,y}(W_i - \overline{W_i})^2}$

- What advantages might NCC have over SSD?

Invariance

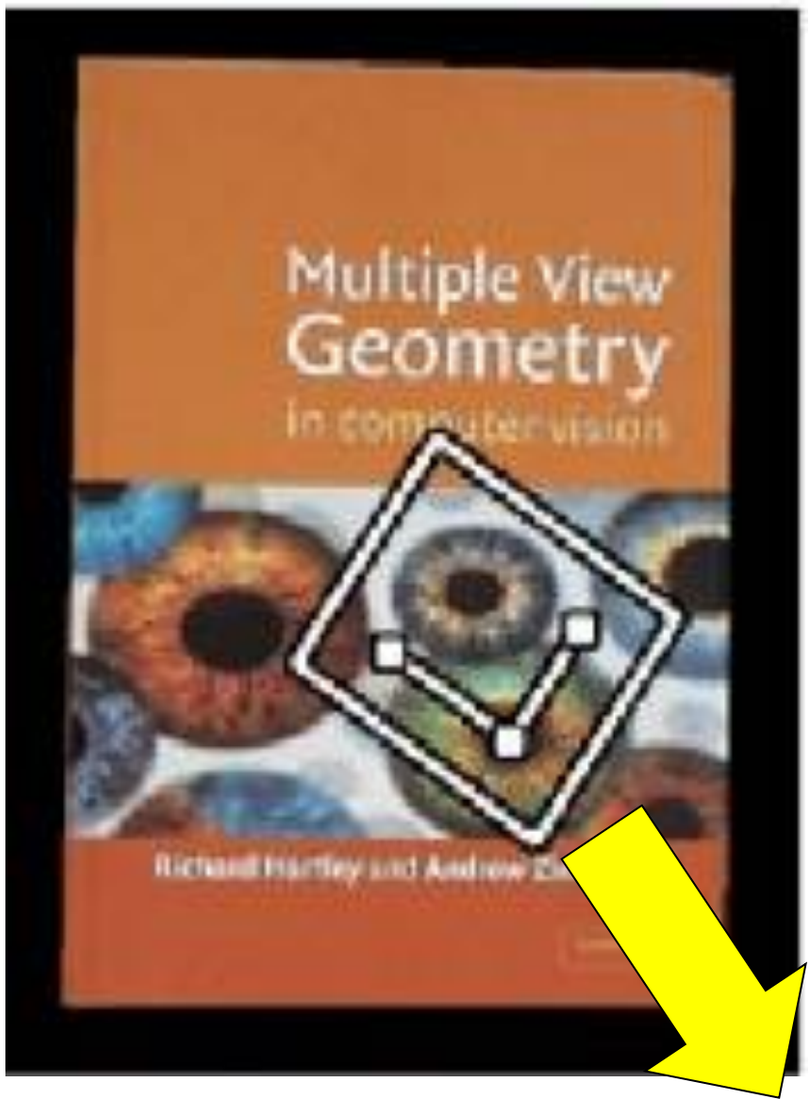
Suppose we are comparing two images I_1 and I_2

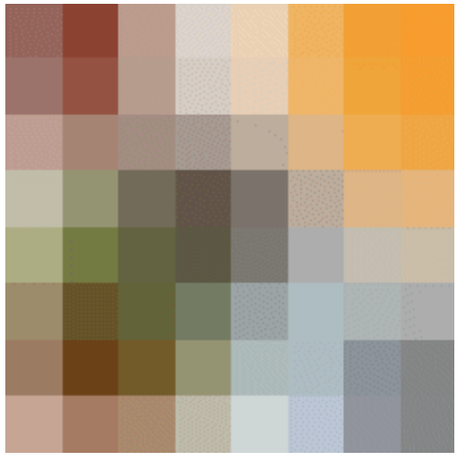
- I_2 may be a transformed version of I_1
- What kinds of transformations are we likely to encounter in practice?

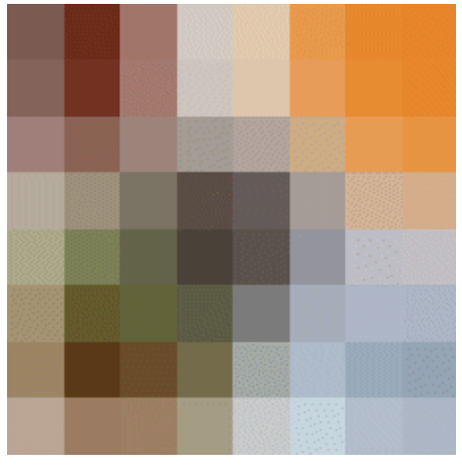
We'd like to find the same features regardless of the transformation

- This is called transformational ***invariance***
- Most feature methods are designed to be invariant to
  - Translation, 2D rotation, scale
- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 30 degrees)
  - Limited illumination or contrast changes

Feature Matching



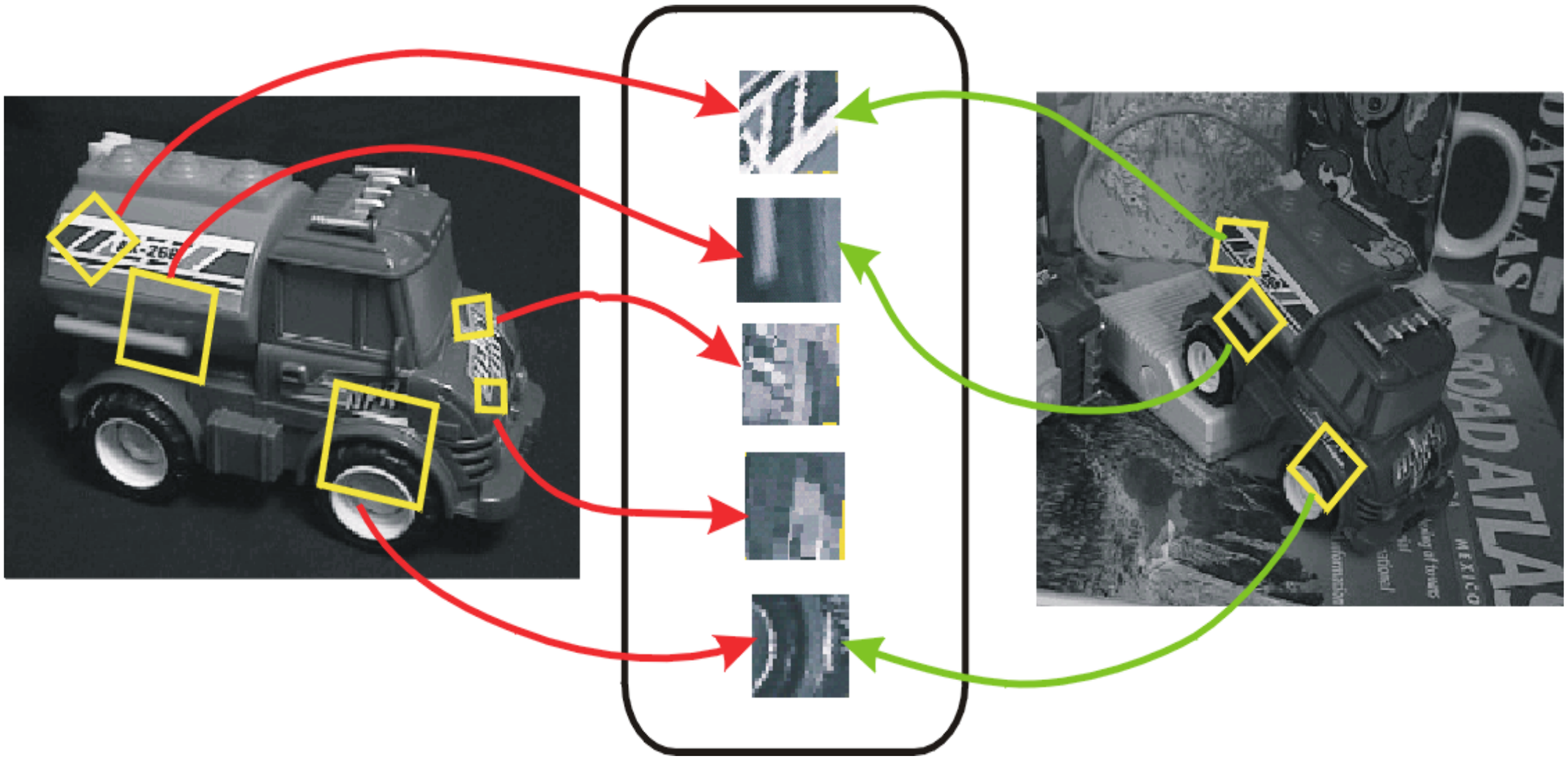




Desirable property: invariance

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale



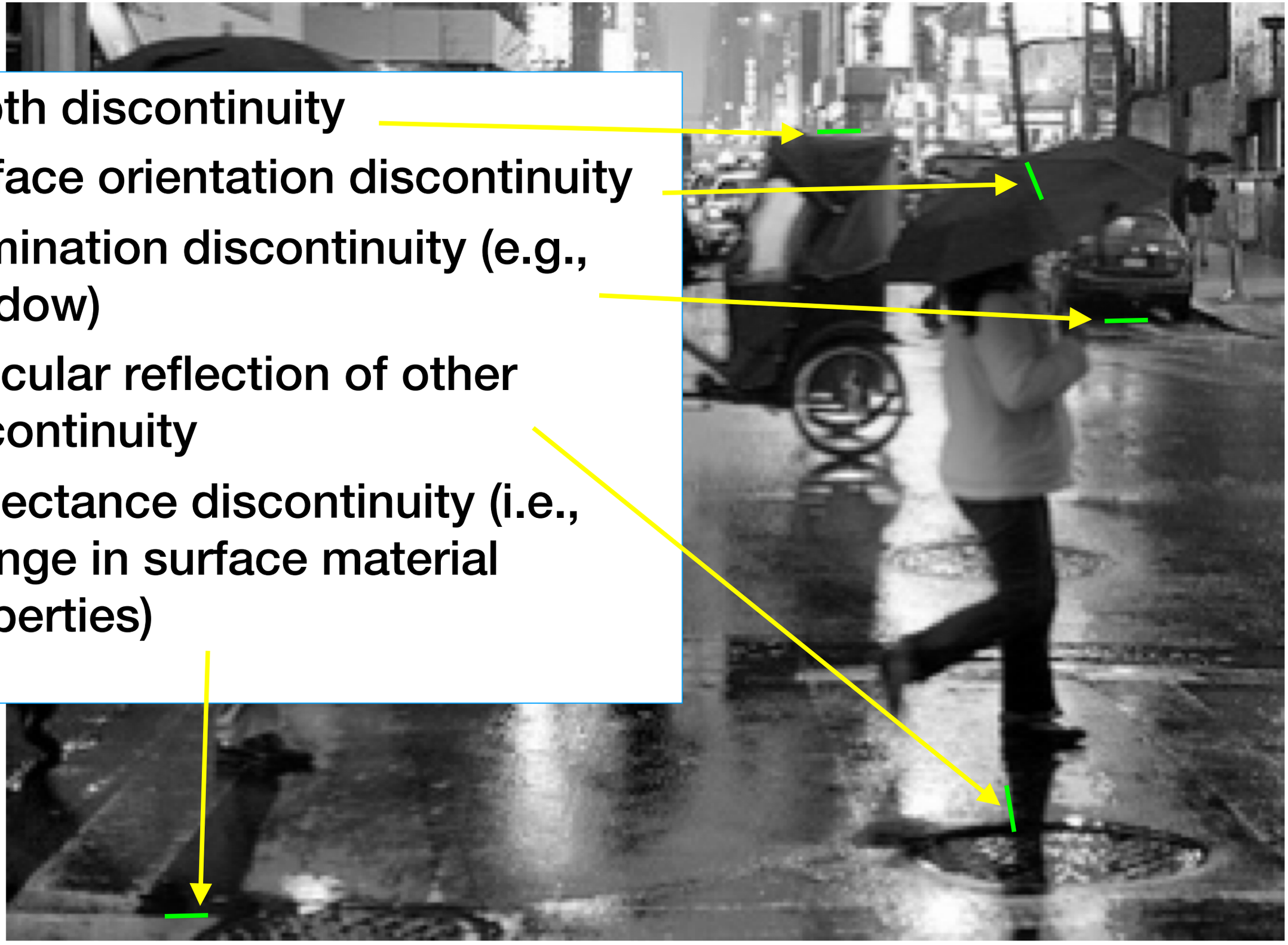
Feature Descriptors

Edges in Natural Images

Source: Photografr.com

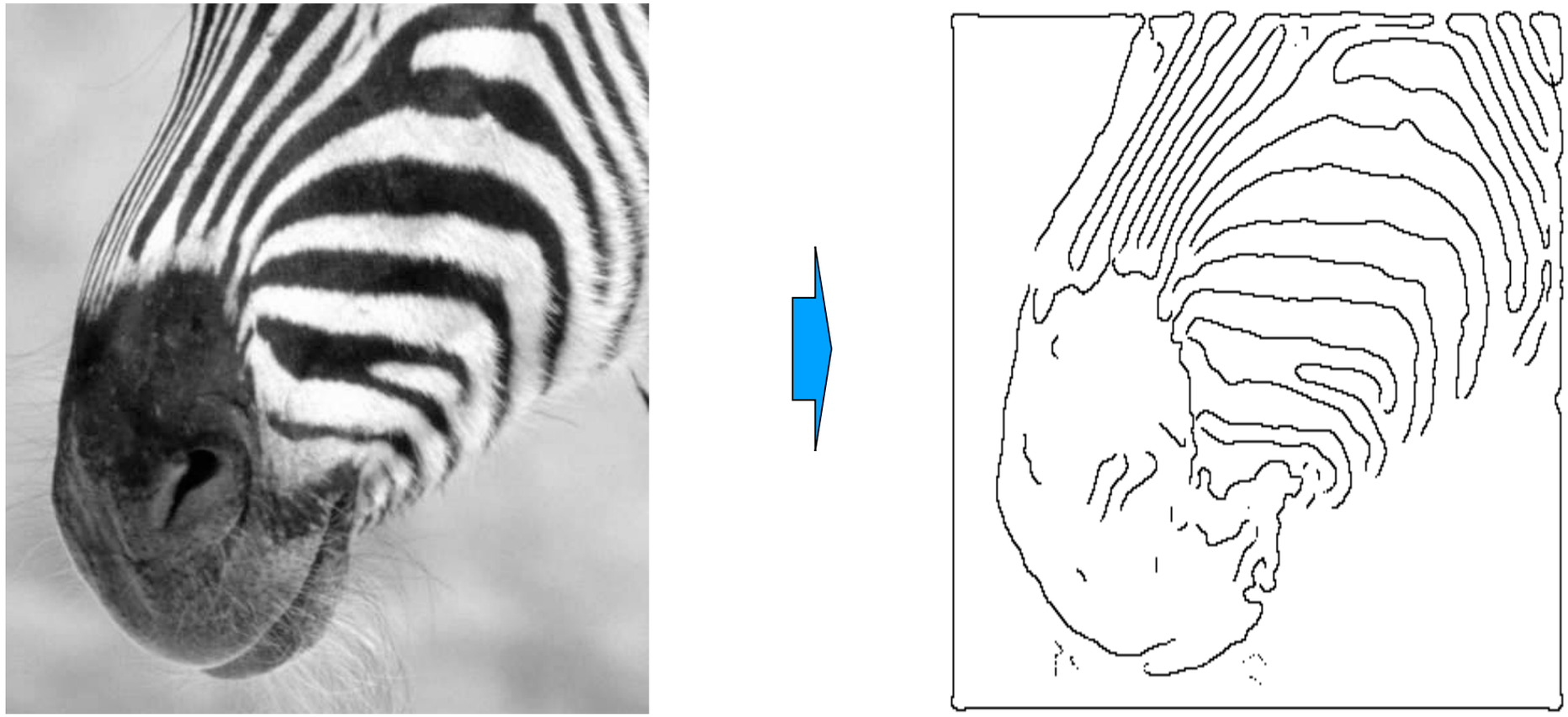
What Causes an Edge?

- Depth discontinuity
- Surface orientation discontinuity
- Illumination discontinuity (e.g., shadow)
- Specular reflection of other discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)



Source: Photografr.com

How Can We Find Edges?



- Find regions where magnitude of gradient is large.

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

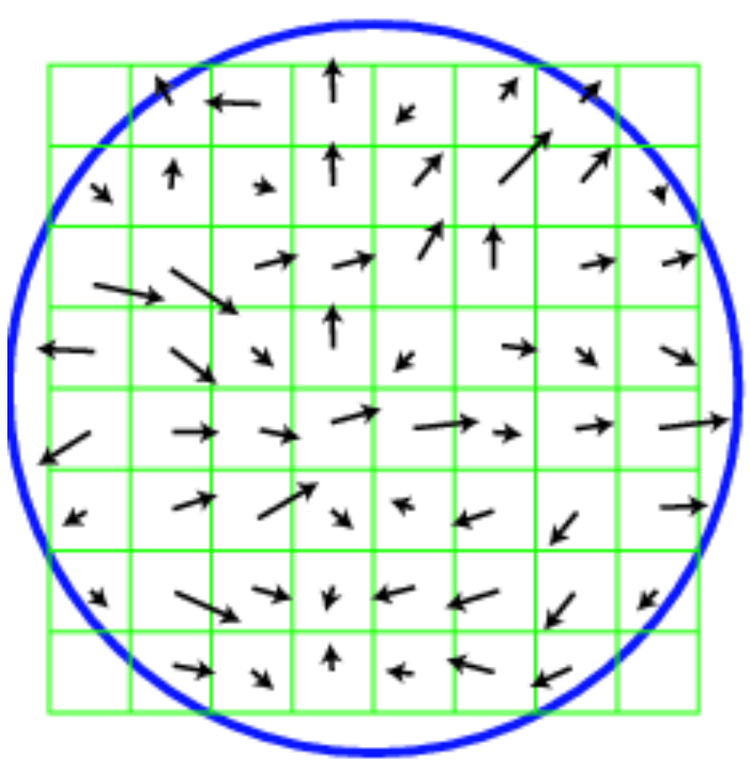
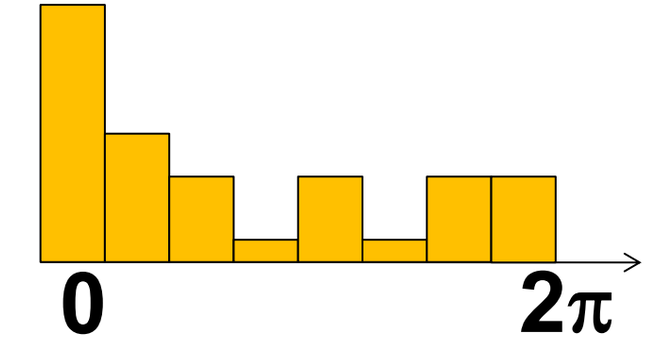
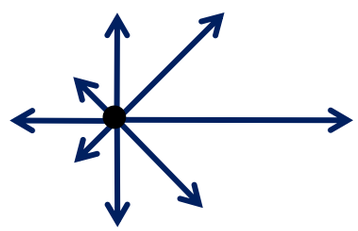


Image gradients



angle histogram

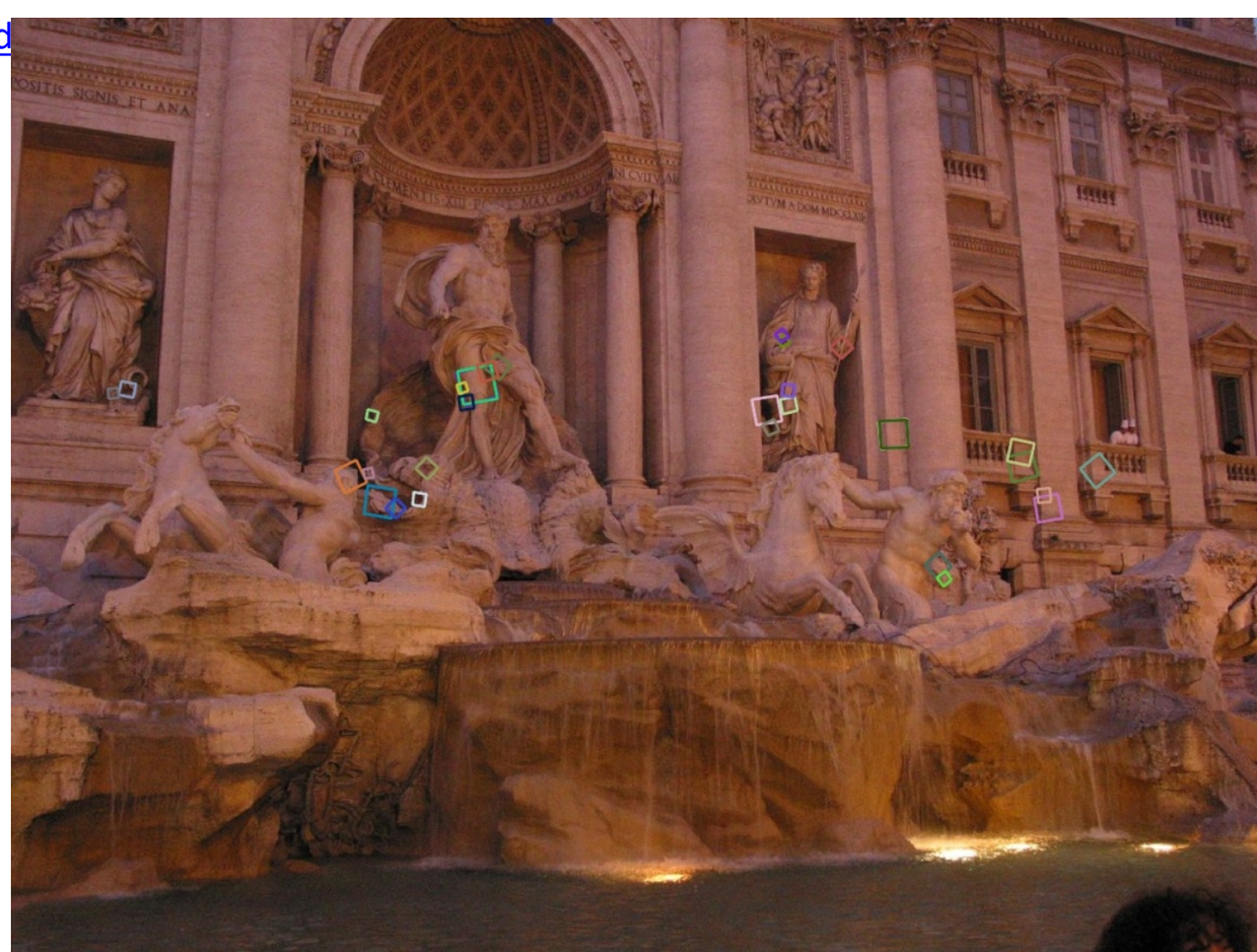


Adapted from slide by David Lowe

Properties of SIFT

Extraordinarily robust matching technique

- Can handle changes in viewpoint
  - Up to about 30 degrees out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available



Feature matching

Given a feature in I_1, how to find the best match in I_2?

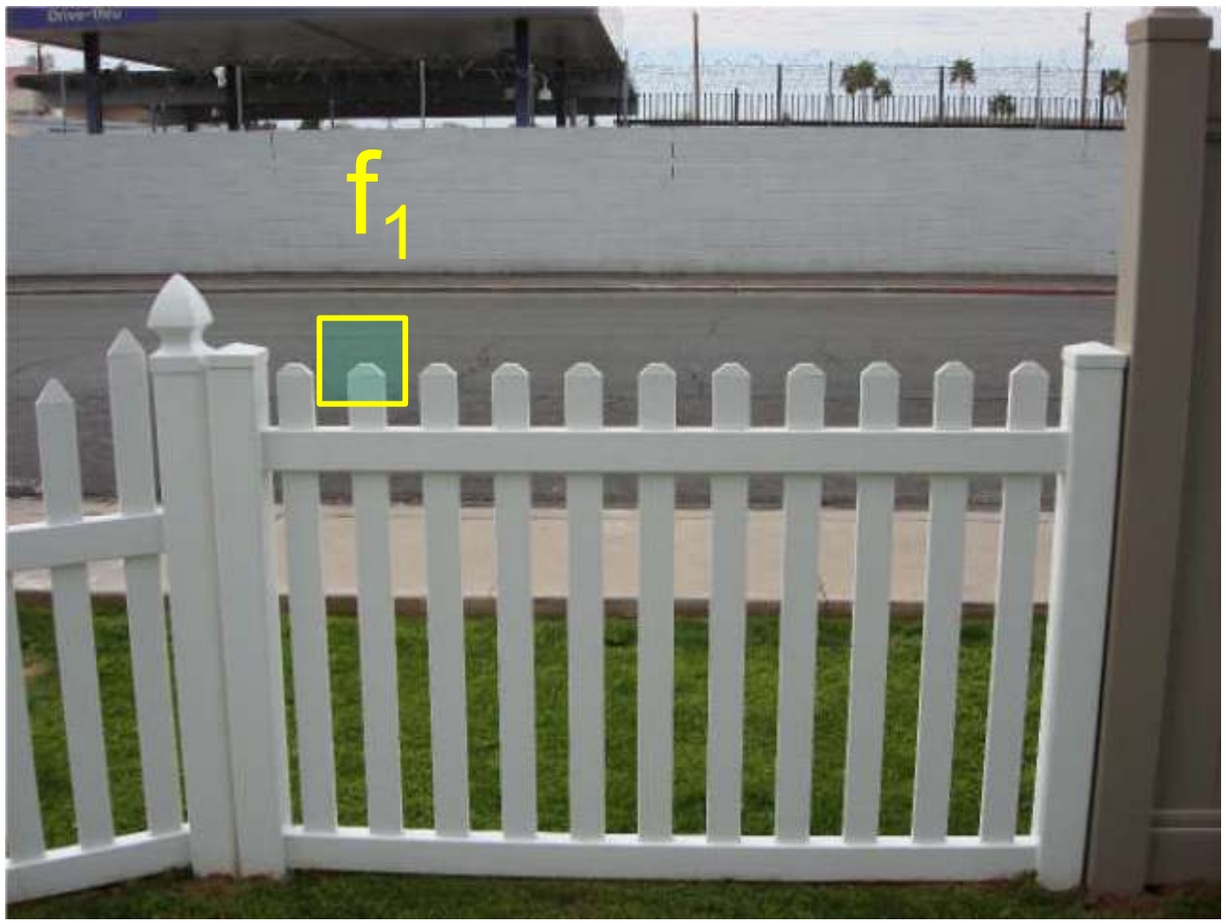
1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

Slides courtesy: Rick Szeliski

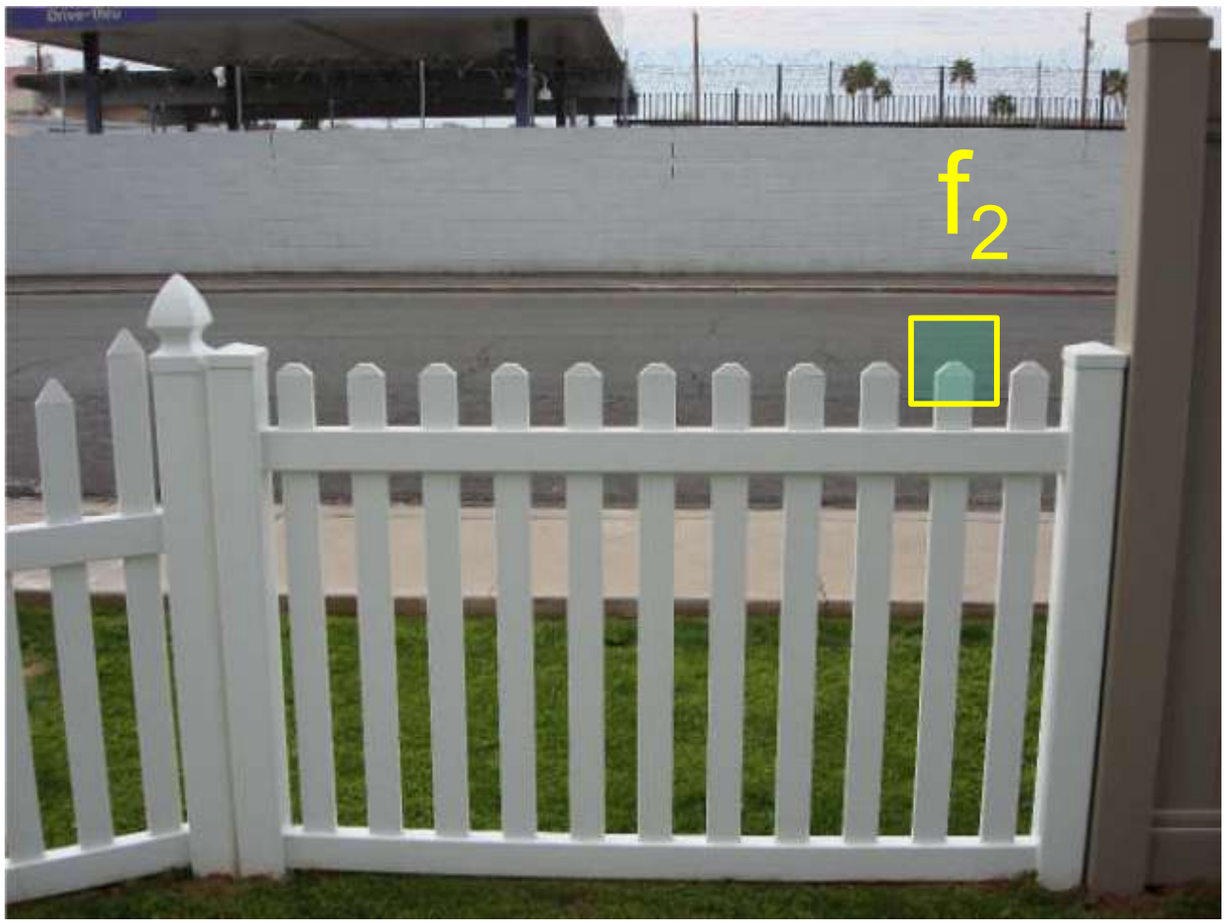
Feature distance

How to define the difference between two features f_1, f_2?

- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches



I_1

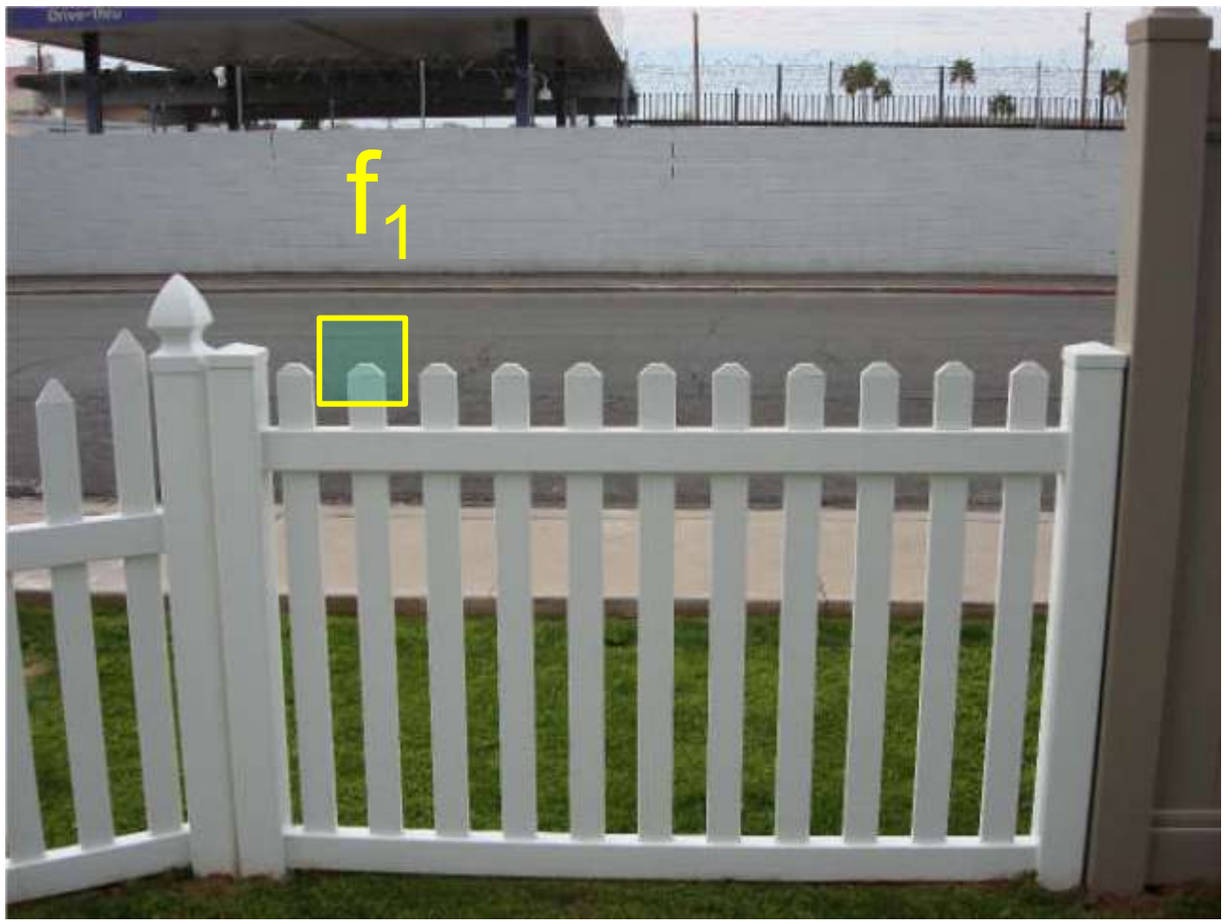


I_2

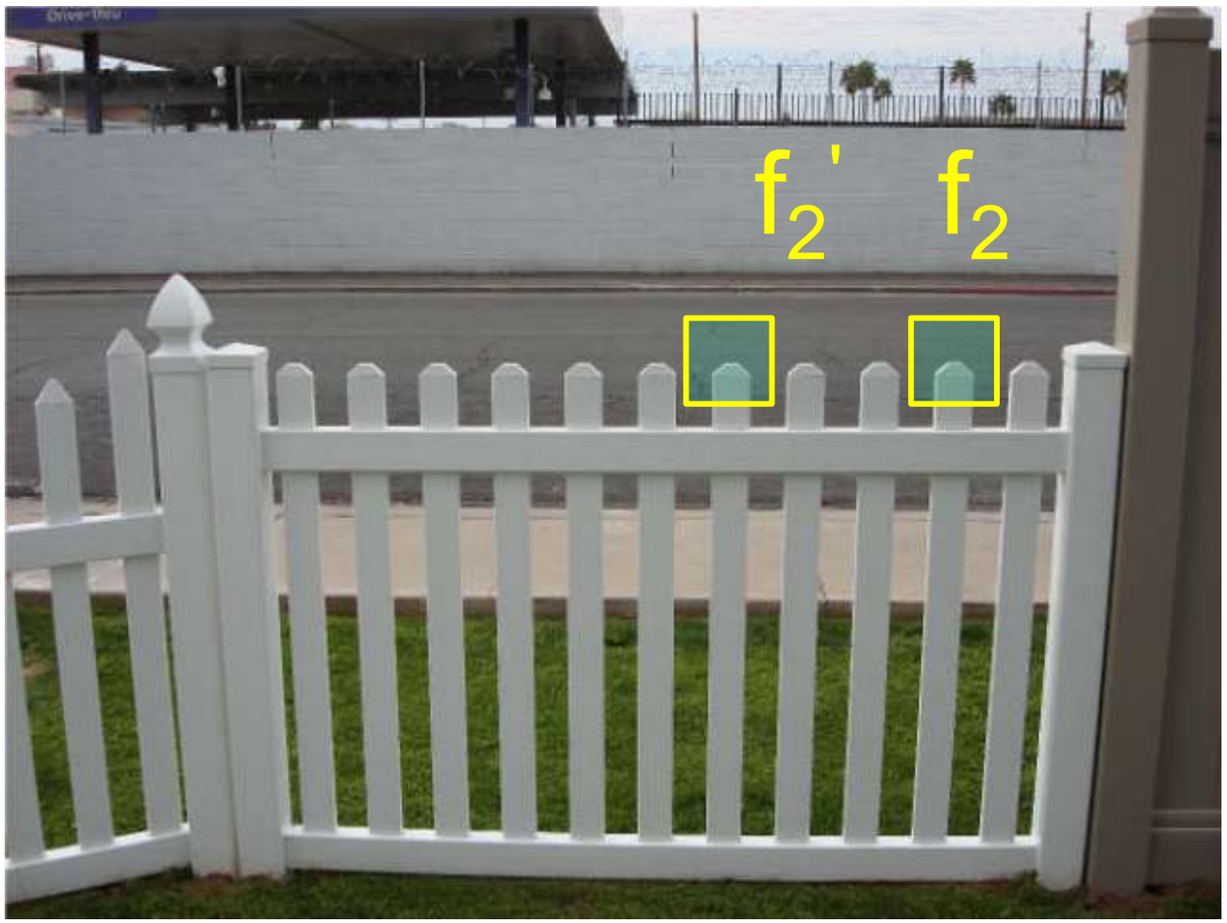
Feature distance

How to define the difference between two features f_1, f_2?

- Better approach: ratio distance = SSD(f_1, f_2) / SSD(f_1, f_2’)
  - f_2 is best SSD match to f_1 in I_2
  - f_2’ is 2nd best SSD match to f_1 in I_2
  - gives small values for ambiguous matches



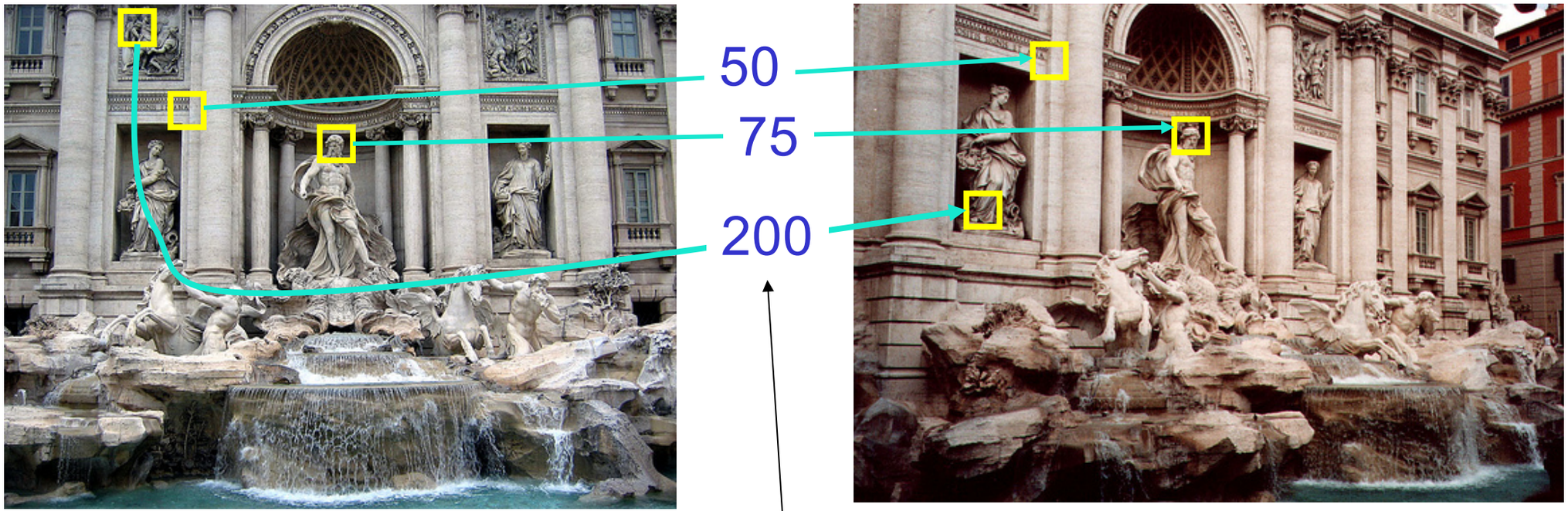
I_1



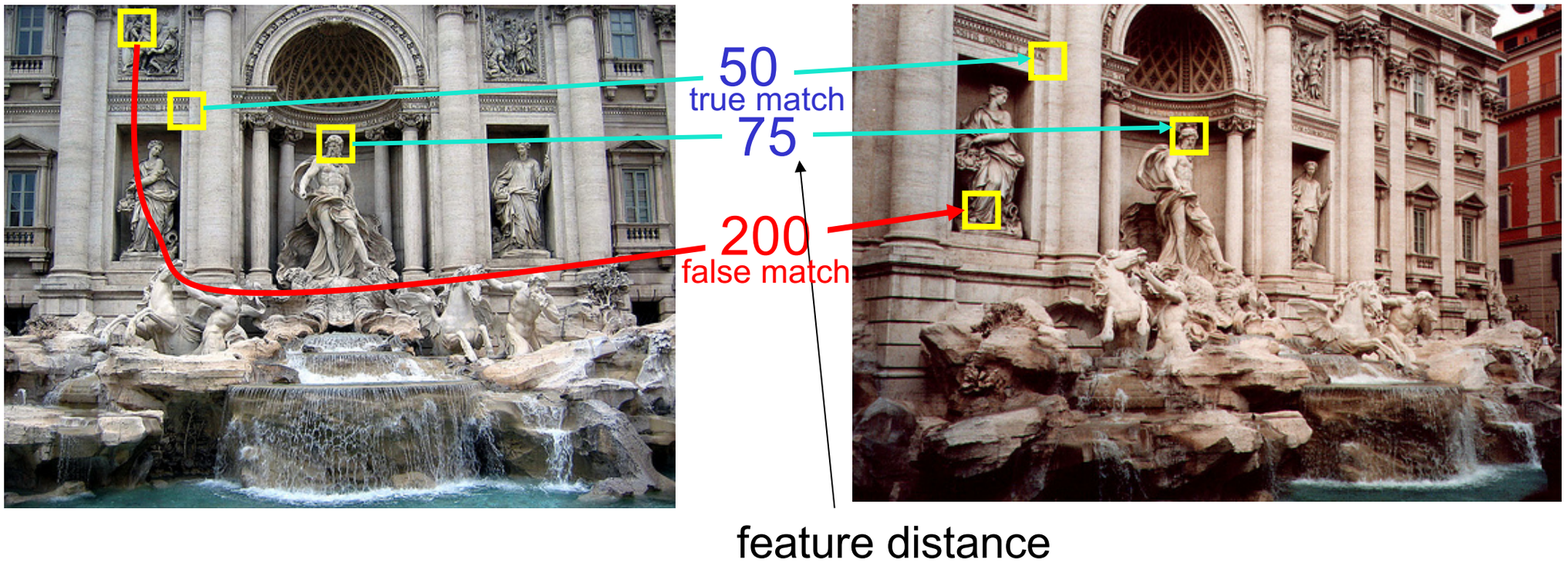
I_2

Evaluating the results

How can we measure the performance of a feature matcher?



True or false positives



The distance threshold affects performance

- True positives = number of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?

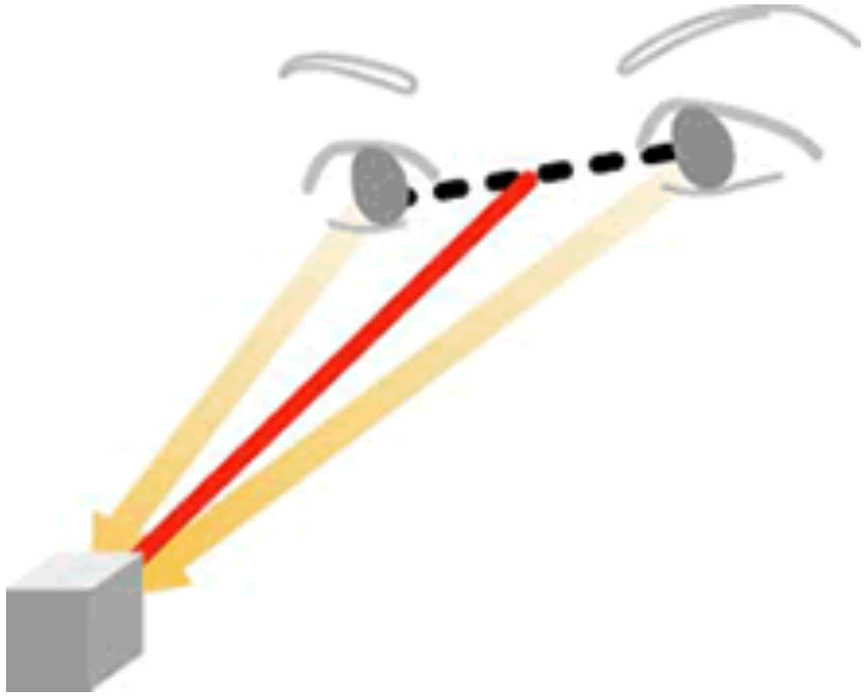
- False positives = number of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

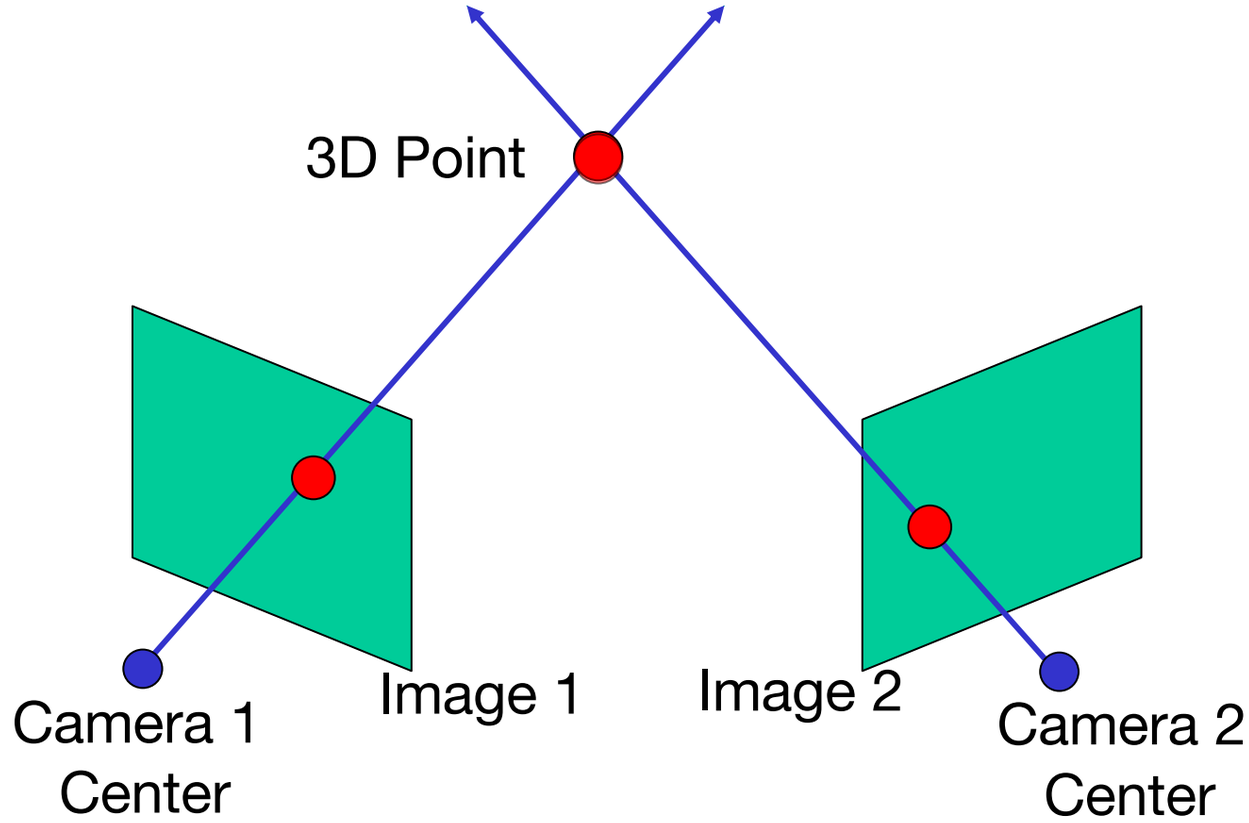
Sony Aibo

SIFT usage:

- Recognize charging station
- Communicate with visual cards
- Teach object recognition

Correspondence is a vital 3D cue





We haven't yet described how such projections are performed