



# EE3662 DSP Lab

## Unit II – Visual Signal

Instructor: 孫民、黃朝宗

TA: 吳俊毅、邱彥廷

National Tsing Hua University  
Department of Electrical Engineering



# Syllabus

- 1<sup>st</sup> Week: 3-hr lecture + take-home lab (Lab 9)
- 2<sup>nd</sup> Week: 3-hr lab on hybrid image (Lab 10)
- 3<sup>rd</sup> Week: 3-hr lab on corner detection (Lab 11)
- 4<sup>th</sup> Week: 3-hr lab on seam carving (Lab 12)

\* MATLAB is used for all labs



# Lecture Outline

## Lab Map

- **Image and Video Basics** Pixel array manipulation (Lab 9)
- **Image Filtering** Image filtering (Lab 10)
- **Interest Point** Image feature (Lab 11)
- **Seam Carving** Image Manipulation (Lab 12)



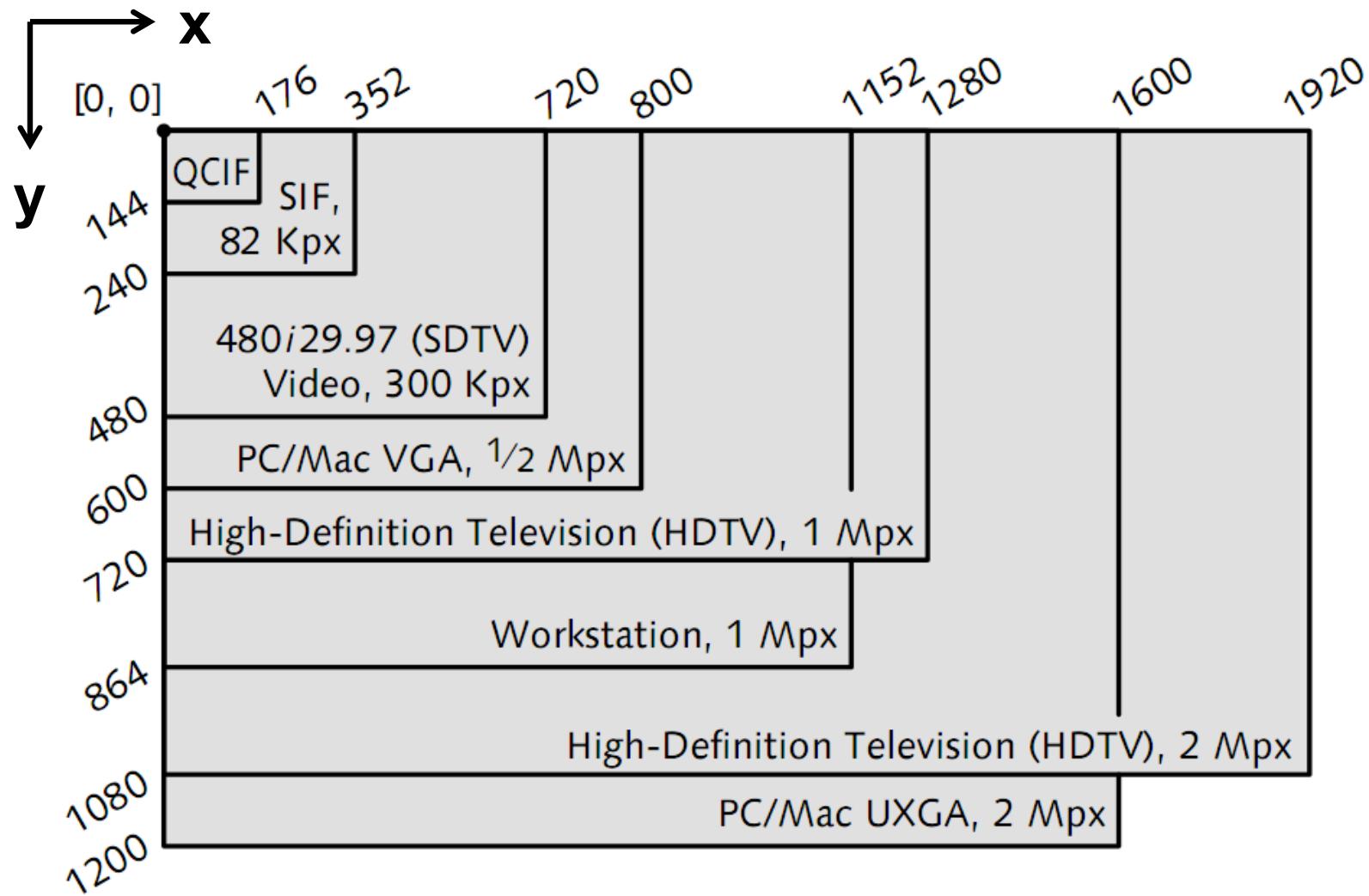
# Basics Representation

- Image
  - Pixel Array (spatial resolution)
  - Pixel Value (luminance and color)
- Video
  - Frame Rate (temporal resolution)



# Pixel Array

- Picture Element, Pel or Pixel, in (x,y)



Ref: *Digital Video and HDTV*



# Pixel Value

Grayscale (灰階) image



```
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
,255,191,128,128,64,0,0,0,0,0,0,0,  
5,255,255,255,255,255,191,64,0,0,0,  
55,255,255,255,255,255,255,255,255,  
,0,128,191,255,255,255,191,0,0,0,0,  
,255,255,128,0,0,0,0,0,0,0,0,0,0,0,  
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

0,1,2,...255 uint8 values



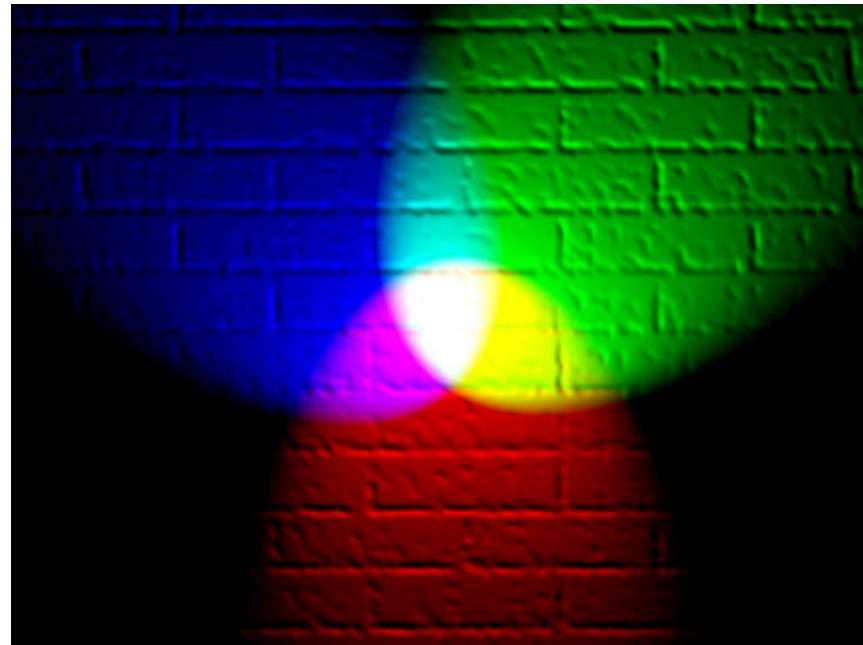
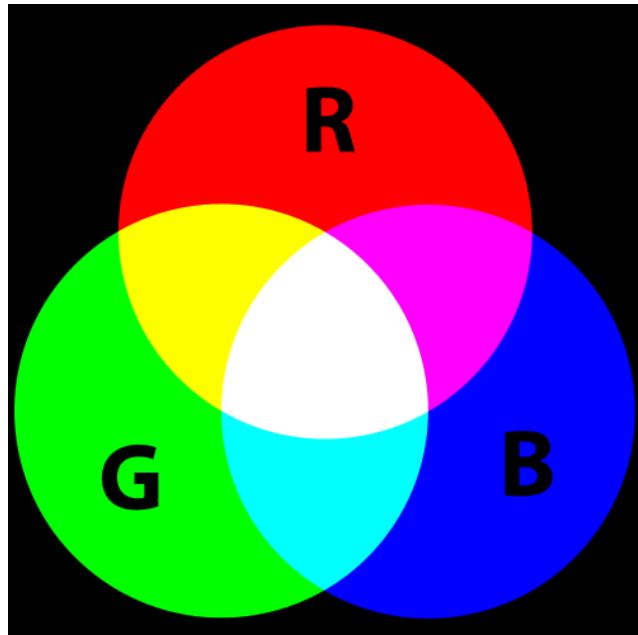
# RGB color space

Color image





# Additive Color



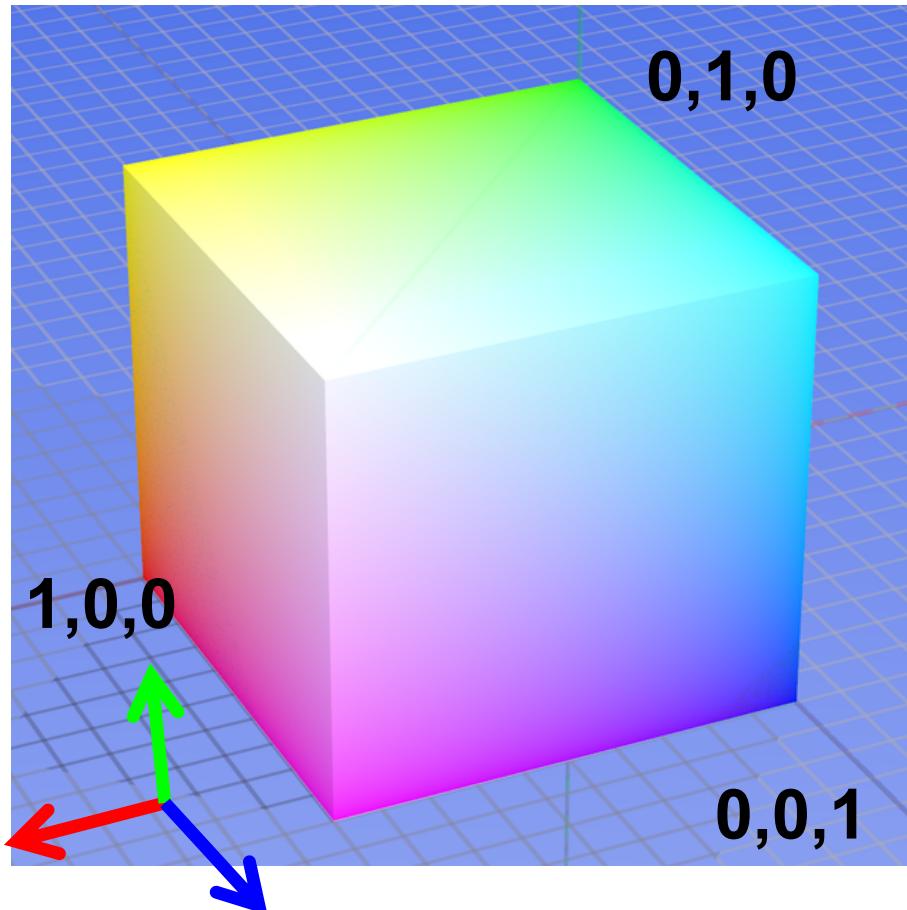
**Additive color** created by mixing light of colors, usually the three primary colors: Red, Green, and Blue (RGB). (used in display devices)



Ref: Wikipedia

# Color spaces: RGB

Default color space



## Some drawbacks

- Strongly correlated channels
- Non-perceptual



**R**  
(G=0,B=0)



**G**  
(R=0,B=0)



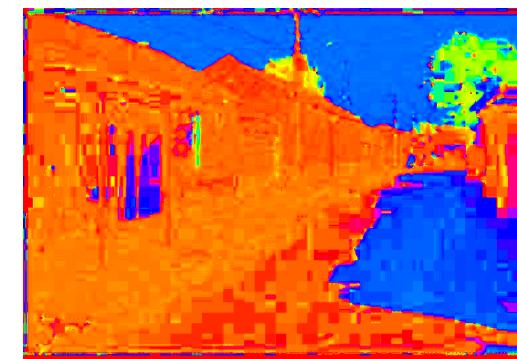
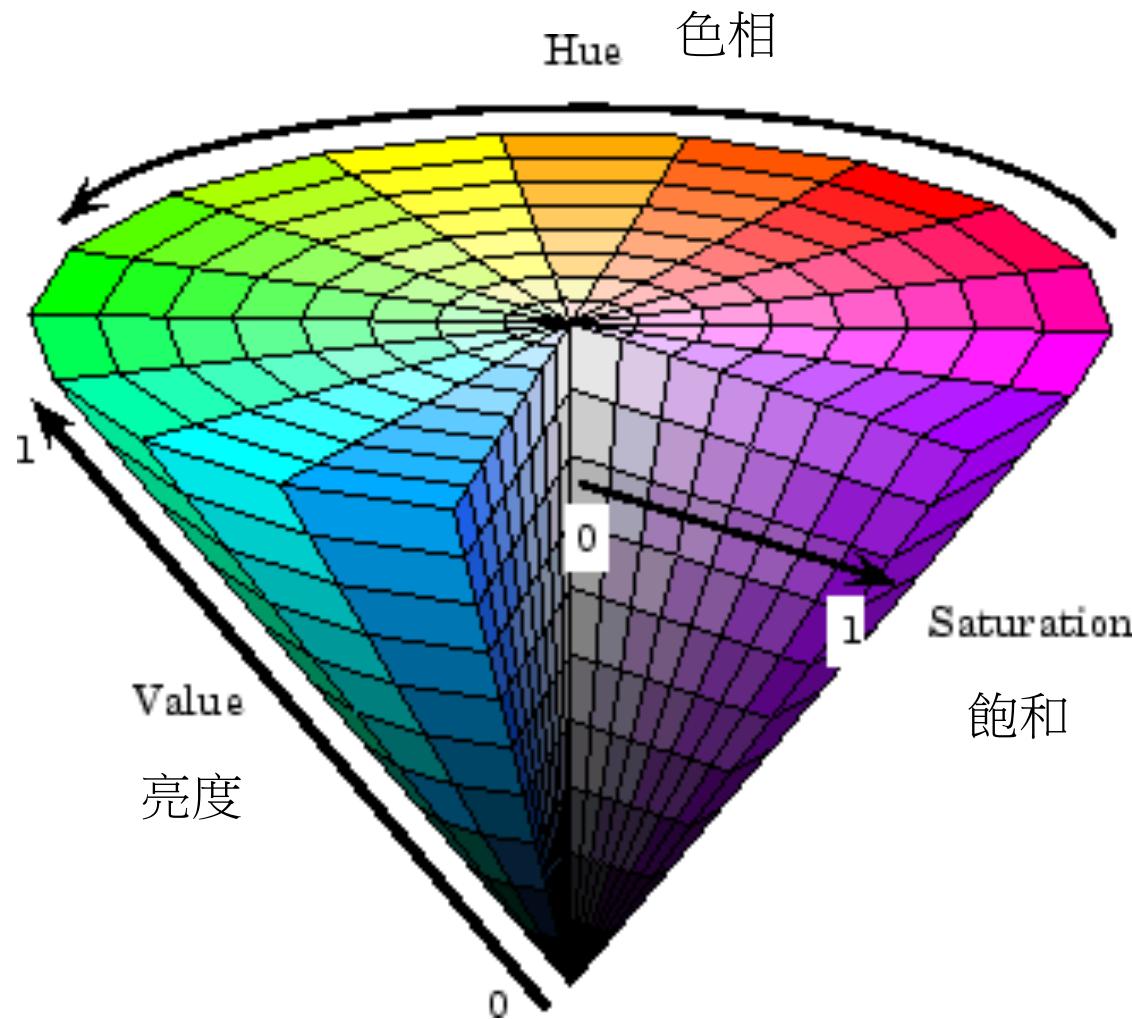
**B**  
(R=0,G=0)

Image from: [http://en.wikipedia.org/wiki/File:RGB\\_color\\_solid\\_cube.png](http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png)

# Color spaces: HSV



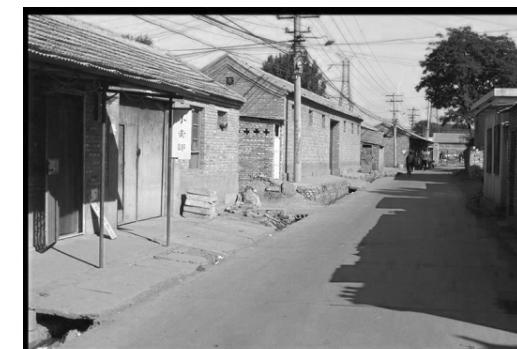
Intuitive color space



**H**  
( $S=1, V=1$ )



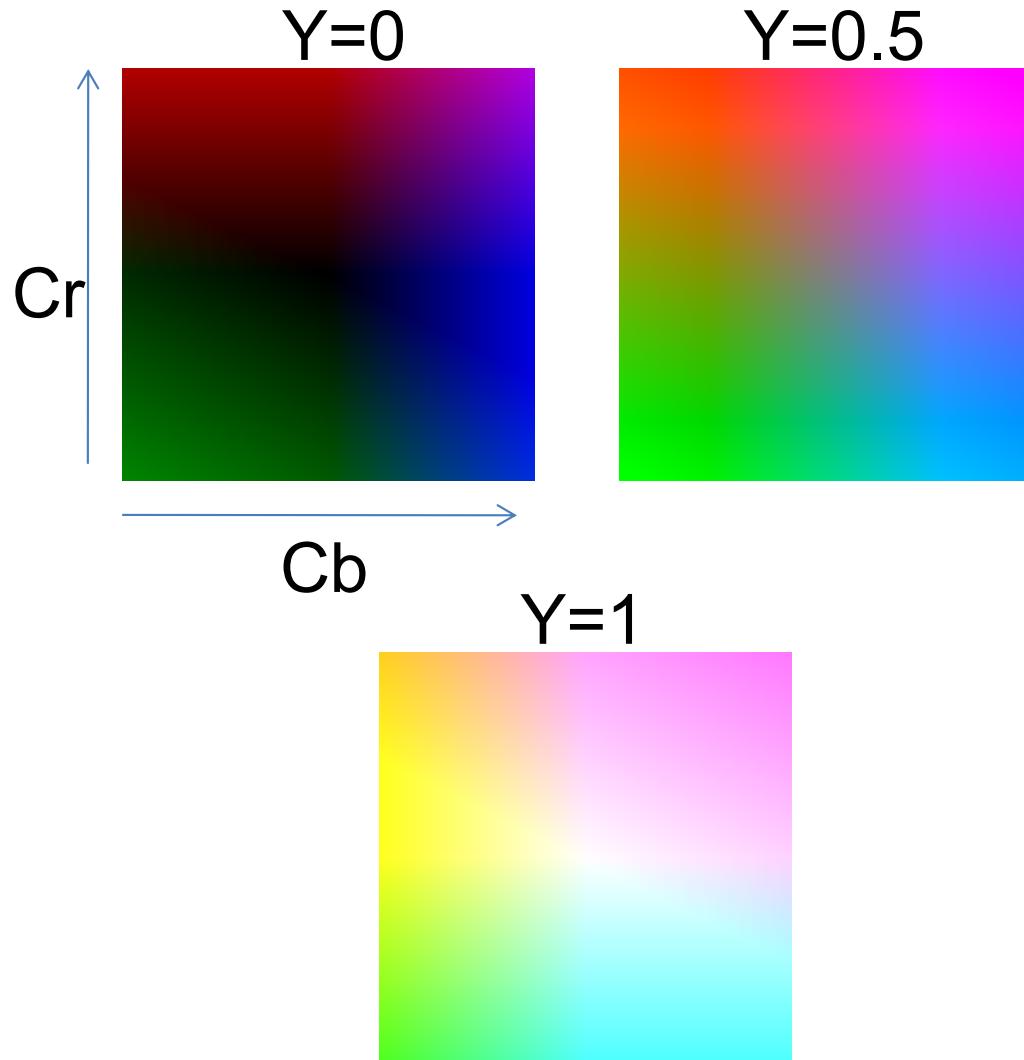
**S**  
( $H=1, V=1$ )



**V**  
( $H=1, S=0$ )

# Color spaces: YCbCr

Fast to compute, good for compression, used by TV



**Y**  
( $Cb=0.5, Cr=0.5$ )



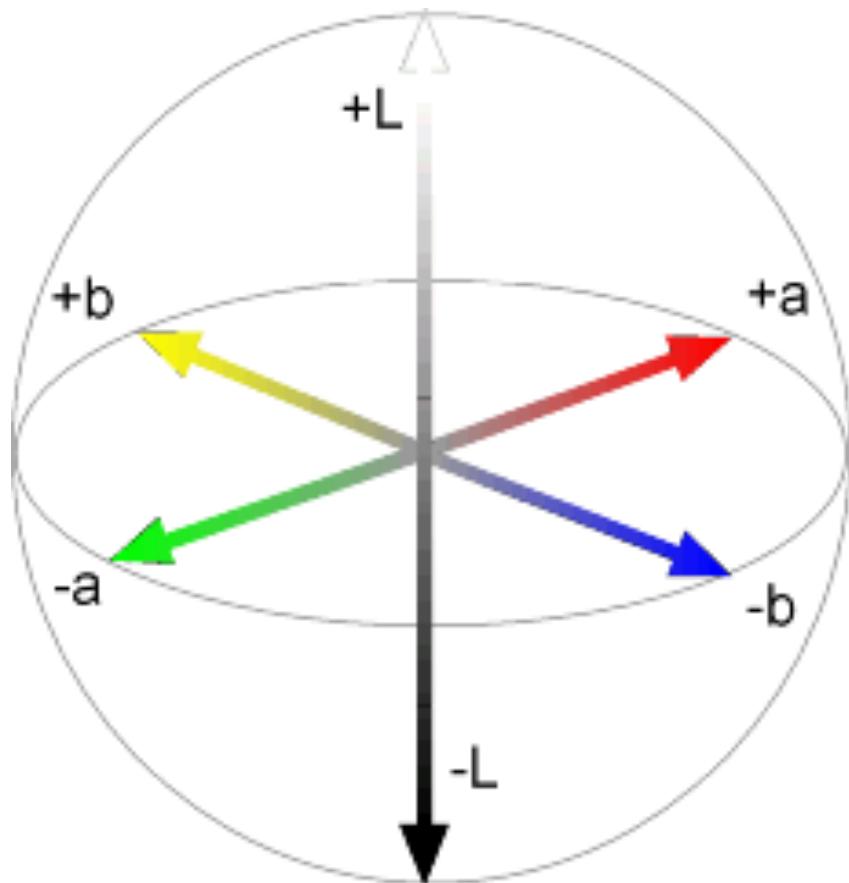
**Cb**  
( $Y=0.5, Cr=0.5$ )



**Cr**  
( $Y=0.5, Cb=0.5$ )

# Color spaces: L\*a\*b\*

“Perceptually uniform”\* color space



**L**  
( $a=0, b=0$ )



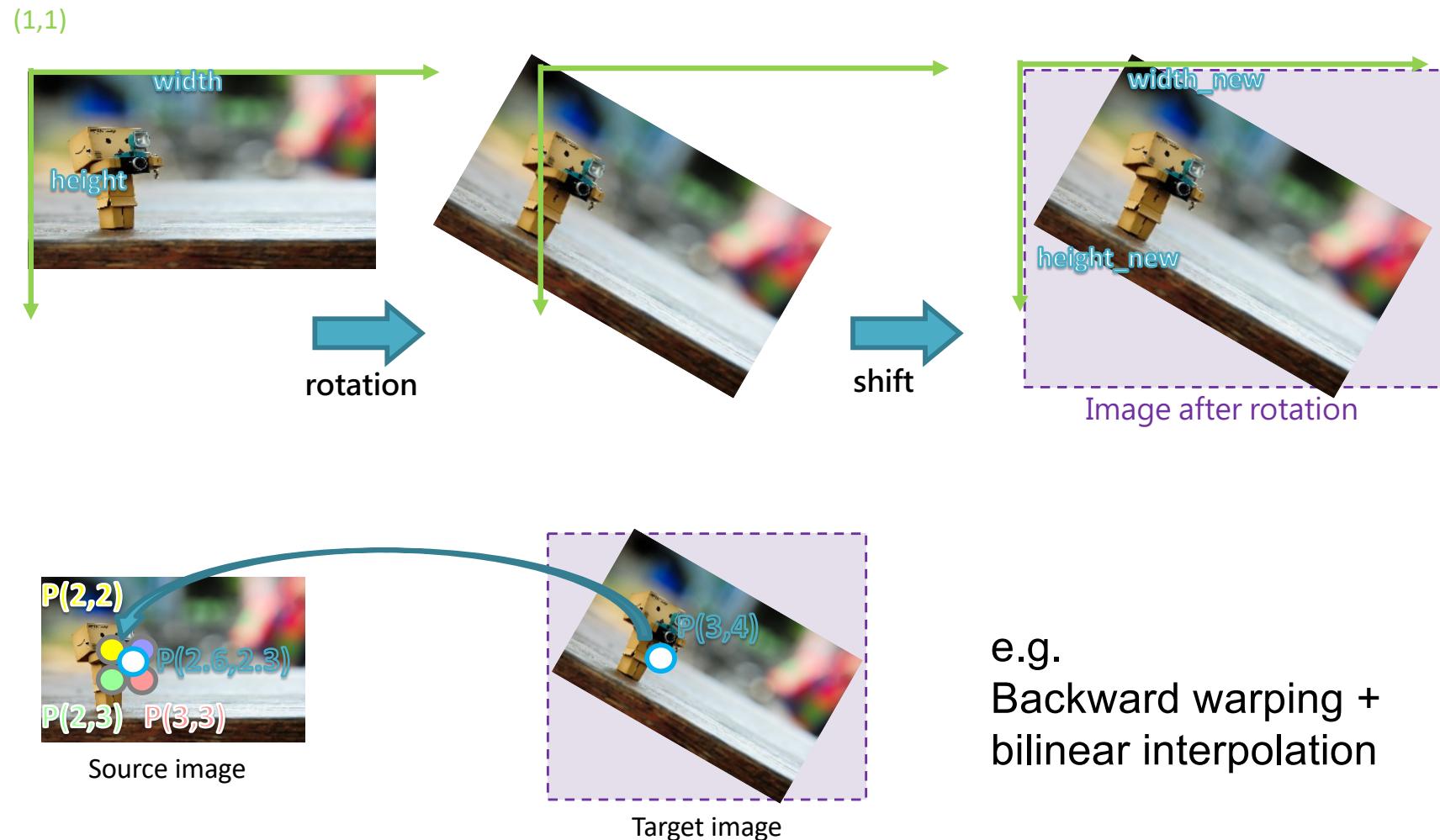
**a**  
( $L=65, b=0$ )



**b**  
( $L=65, a=0$ )

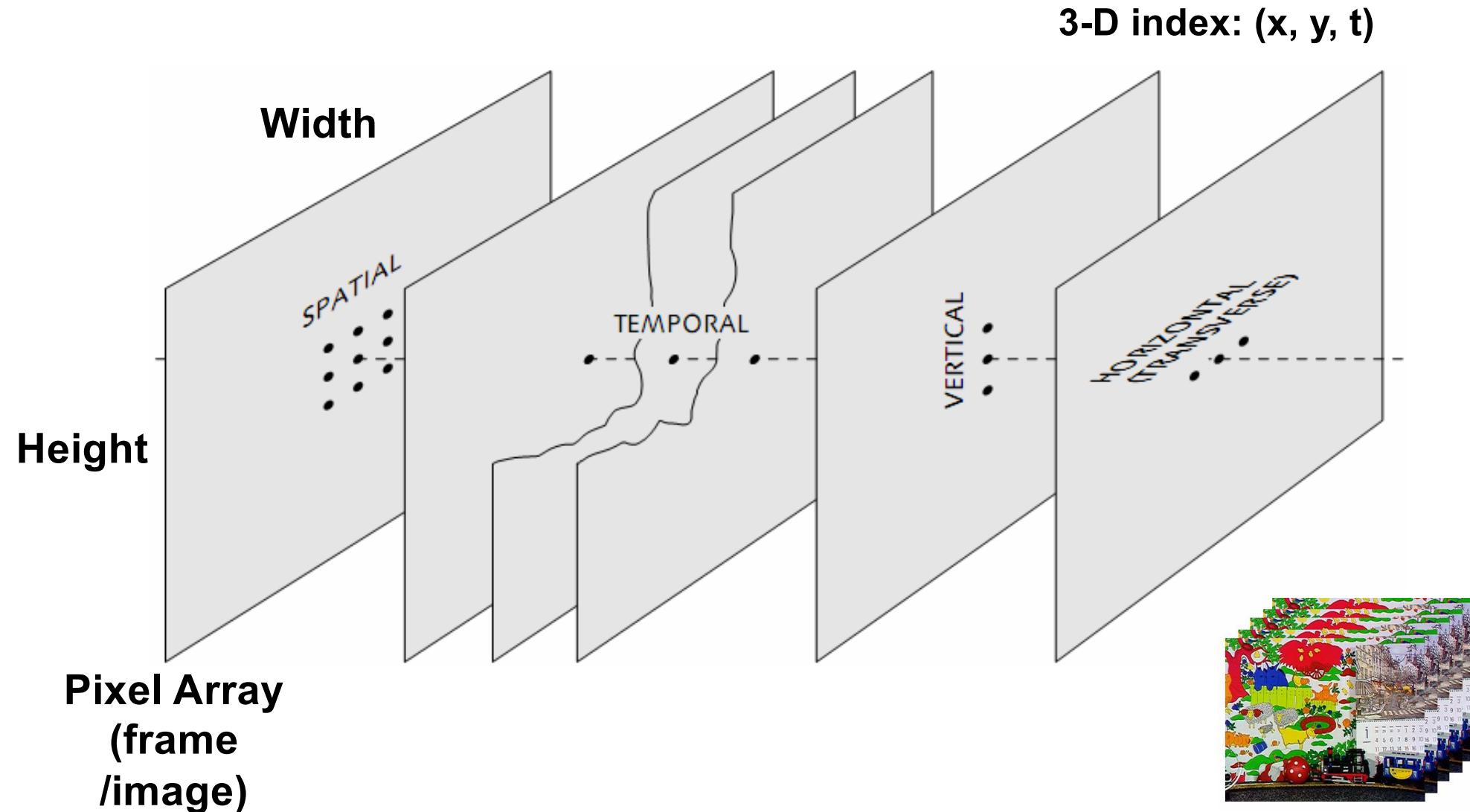


# Example: Rotation (Lab 9)





# Video = Sequence of Images



Ref: *Digital Video and HDTV*



# Refresh Rate vs. Frame Rate

| <i>Viewing environment</i> | <i>Ambient illumination</i> | <i>Refresh (flash) rate, Hz</i>  | <i>Frame rate, Hz</i>   |
|----------------------------|-----------------------------|----------------------------------|-------------------------|
| Cinema                     | Dark                        | 48                               | 24                      |
| Television                 | Dim                         | 50                               | 25                      |
|                            | Dim                         | ≈60                              | ≈30                     |
| Office                     | Bright                      | various, e.g.,<br>66, 72, 76, 85 | same as<br>refresh rate |

At least high enough to avoid flicker. ←

Ref: *Digital Video and HDTV*

Higher for fast-moving objects.

e.g. for 120Hz LCD TV:

Encoded Video: 30Hz (frame per second, fps)

HDMI 1080p : 60Hz (progressive)

Refresh : 120Hz

# Resolution Explosion



**85" 8Kx4K (sharp)**



**84" 4Kx2K (LG)**

monitor

(Eizo, ViewSonic,  
IBM, Sharp, ...)

**20"~36" 4Kx2K**



**3.5" 320p  
(Apple)**



**3.5" 640p  
(Apple)**



**4" 640p  
(Apple)**



**5" 1080p  
(HTC)**

**2009**

**2010**

**2011**

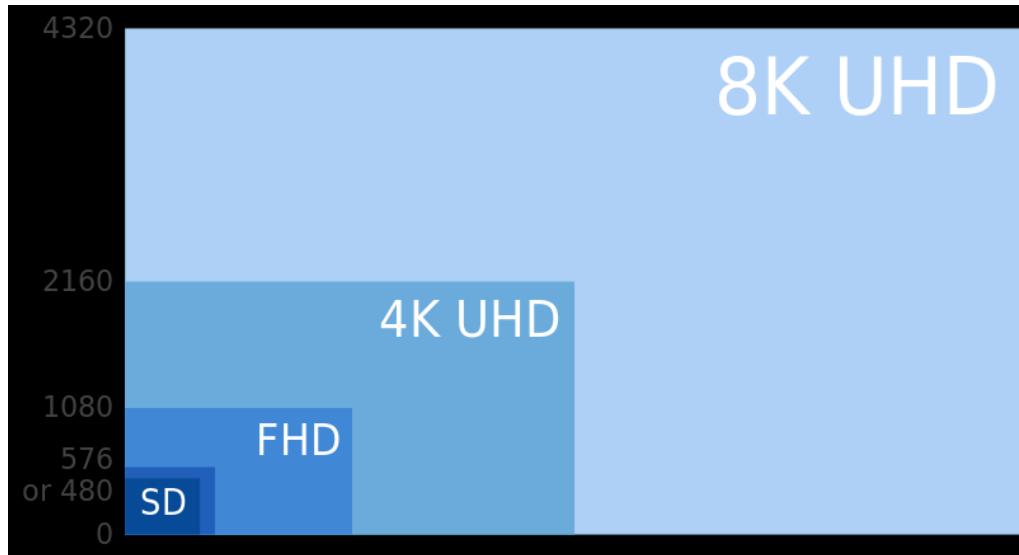
**2012**



**1080p  
iPhone6  
2014**



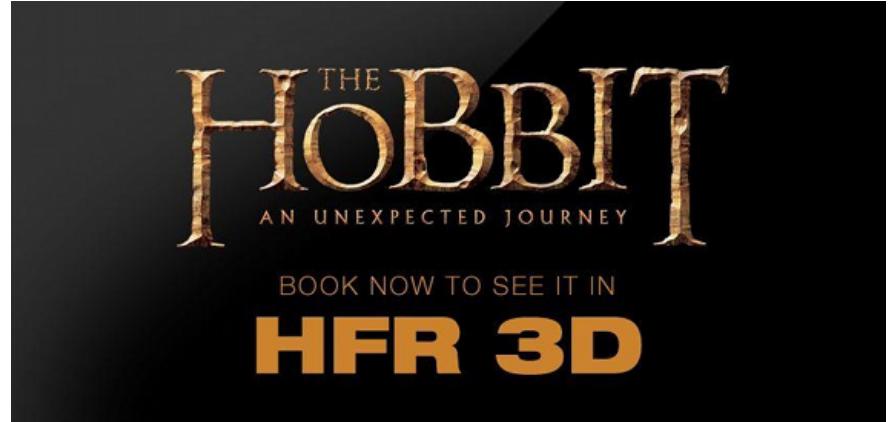
# Frame Rate Going Up



## Ultra High Definition TV

4Kx2K/8Kx4K

Up to 120 fps by 2020



## Cinema

HFR (High Frame Rate)  
48fps



LCD/LED TV  
**240Hz** refresh rate



# Audio/Image/Video in Practice

- Audio
  - 1-D signal
  - Intensity
  - Long-tap filter
  - Throughput
    - ~ 44K samples/s
- Image
  - 2-D signal
  - RGB triple
  - Short-tap filter
    - Locally stationary
  - Large size
    - e.g. 24M pixels
- Video
  - 2-D + time
  - YCbCr triple
  - Motion model
  - Resource demanding
    - 8K4K 120fps  
=> 3.8Gpixel/s
    - > 60% of internet traffic



# Lecture Outline

- Image and Video Basics
- Recap on Image filtering
- Interest Point
- Seam Carving



# Image filtering

- Image filtering: compute function of local neighborhood at each position
- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc. **[Lab]**
  - Detect patterns
    - Template matching

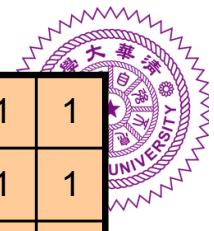
# Example: box filter

$g[\cdot, \cdot]$

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Slide credit: David Lowe (UBC)

# Image filtering



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

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$f[\cdot, \cdot]$

0

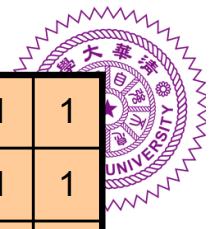
|   |   |    |    |    |    |    |    |   |   |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 |
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0  | 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 |  |  |  |  |  |  |
|  |  |  |   |  |  |  |  |  |  |
|  |  |  |   |  |  |  |  |  |  |
|  |  |  |   |  |  |  |  |  |  |
|  |  |  |   |  |  |  |  |  |  |
|  |  |  |   |  |  |  |  |  |  |
|  |  |  |   |  |  |  |  |  |  |
|  |  |  |   |  |  |  |  |  |  |

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

# Image filtering



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

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$f[\cdot, \cdot]$

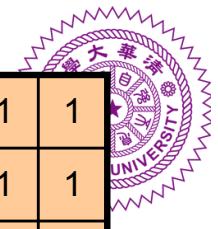
|   |   |    |    |    |    |    |    |   |   |   |
|---|---|----|----|----|----|----|----|---|---|---|
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 | 0 |
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 0  | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 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 | 0 | 0 |

$h[\cdot, \cdot]$

|   |    |  |  |  |  |  |  |  |  |  |
|---|----|--|--|--|--|--|--|--|--|--|
|   |    |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |
| 0 | 10 |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |
|   |    |  |  |  |  |  |  |  |  |  |

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

# Image filtering



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

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$f[\cdot, \cdot]$

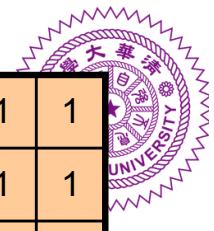
|   |   |    |    |    |    |    |    |    |   |
|---|---|----|----|----|----|----|----|----|---|
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 |
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 0 |
| 0 | 0 | 0  | 90 | 0  | 90 | 90 | 90 | 90 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 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 |  |  |  |  |  |  |
|  |   |    |    |  |  |  |  |  |  |
|  |   |    |    |  |  |  |  |  |  |
|  |   |    |    |  |  |  |  |  |  |
|  |   |    |    |  |  |  |  |  |  |
|  |   |    |    |  |  |  |  |  |  |
|  |   |    |    |  |  |  |  |  |  |
|  |   |    |    |  |  |  |  |  |  |

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

# Image filtering



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

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

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

|   |   |    |    |    |    |    |    |    |   |
|---|---|----|----|----|----|----|----|----|---|
| 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 | 0  | 90 | 90 | 90 | 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 |

$$h[\cdot, \cdot]$$

|  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

A 10x10 grid representing the output image. The value at position (m, n) is highlighted with a red box and labeled '30'.

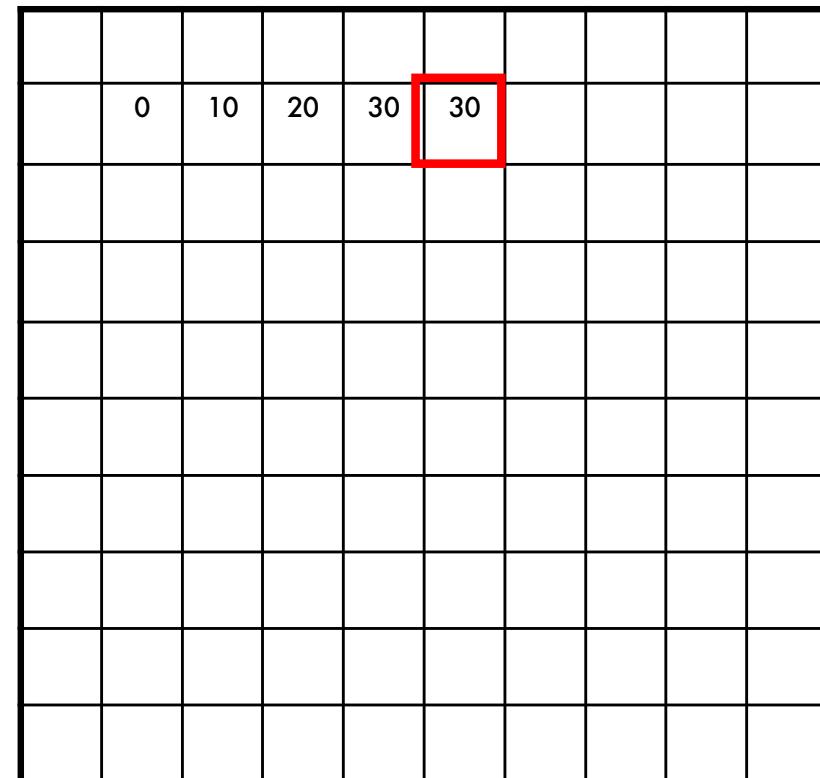
$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

# Image filtering

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

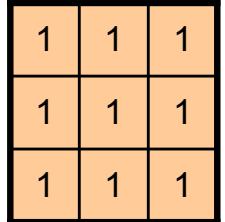
$$f[\cdot, \cdot]$$

$h[.,.]$



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

# Image filtering

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


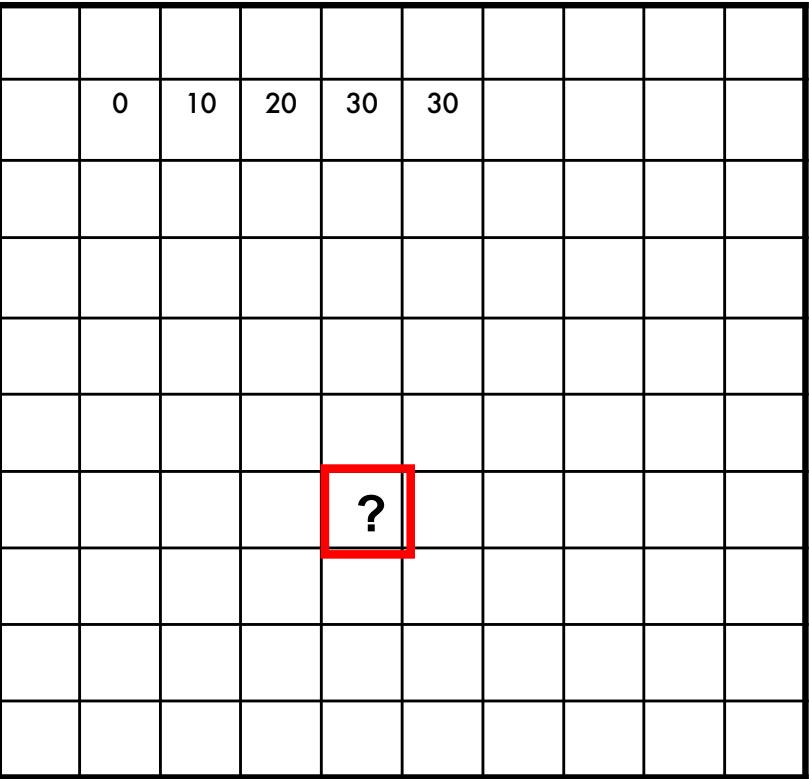
A 3x3 matrix with all entries equal to 1/9. The matrix is enclosed in a black border.

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

|   |   |    |    |    |    |    |    |   |   |
|---|---|----|----|----|----|----|----|---|---|
| 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 |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |



A 10x10 grid of empty cells. A red box highlights the cell at the intersection of the 5th column and the 4th row, which contains a question mark '?'.

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

# Image filtering

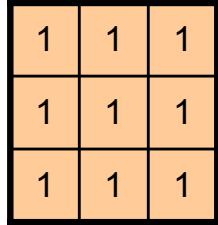
 $f[\cdot, \cdot]$ 

|   |   |    |    |    |    |    |    |    |   |   |
|---|---|----|----|----|----|----|----|----|---|---|
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 |
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0  | 90 | 0  | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 90 | 0 | 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  | 0 | 0 |

 $h[\cdot, \cdot]$ 

|  |   |    |    |    |    |  |  |  |  |  |
|--|---|----|----|----|----|--|--|--|--|--|
|  |   |    |    |    |    |  |  |  |  |  |
|  | 0 | 10 | 20 | 30 | 30 |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |
|  |   |    |    |    |    |  |  |  |  |  |

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

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


A 3x3 matrix with all entries equal to 1/9. The matrix is defined by the equation  $g[\cdot, \cdot] \frac{1}{9}$ .

# Image filtering

$$g[\cdot, \cdot] \quad \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

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

|   |   |    |    |    |    |    |    |   |   |   |
|---|---|----|----|----|----|----|----|---|---|---|
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 | 0 |
| 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 0  | 90 | 90 | 90 | 0 | 0 | 0 |
| 0 | 0 | 0  | 90 | 90 | 90 | 90 | 90 | 0 | 0 | 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 | 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  |  |  |
|  |    |    |    |    |    |    |    |    |  |  |

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

# Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

$g[\cdot, \cdot]$

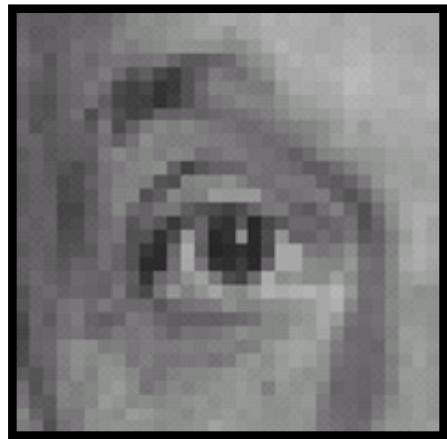
$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

# Smoothing with box filter





# Practice with linear filters



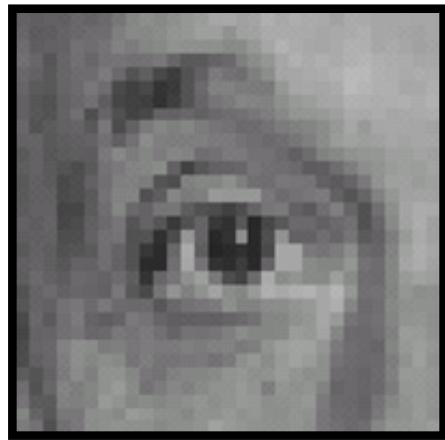
Original

|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

?

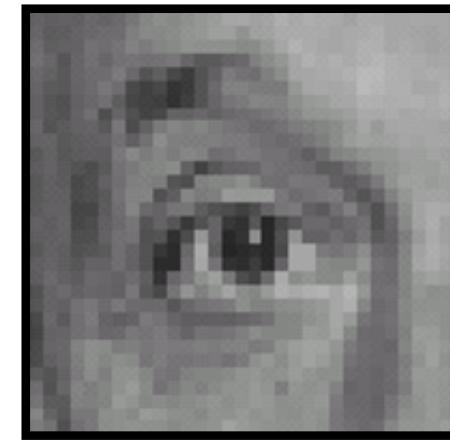


# Practice with linear filters



Original

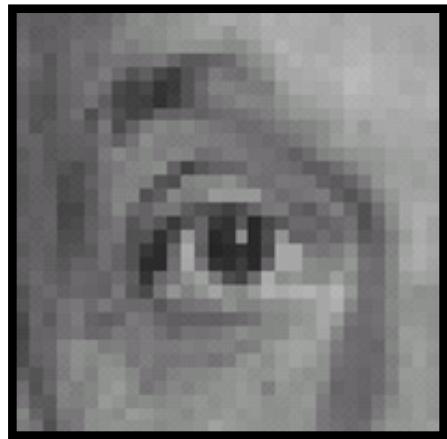
|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |



Filtered  
(no change)



# Practice with linear filters



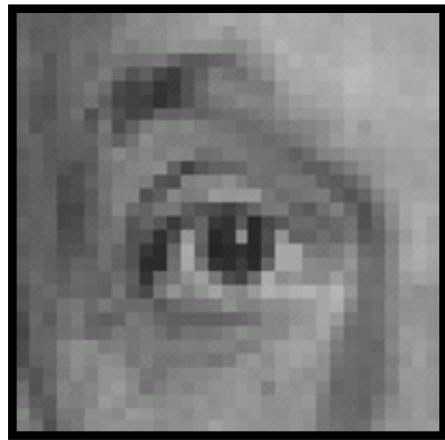
Original

|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

?

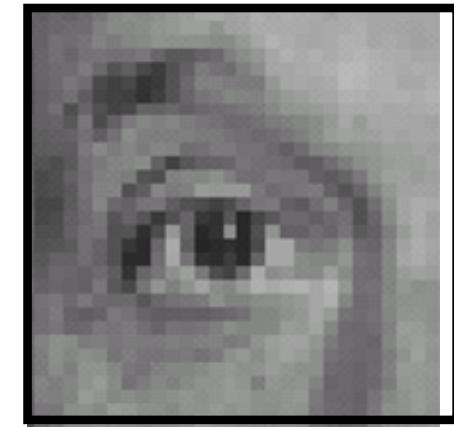


# Practice with linear filters



Original

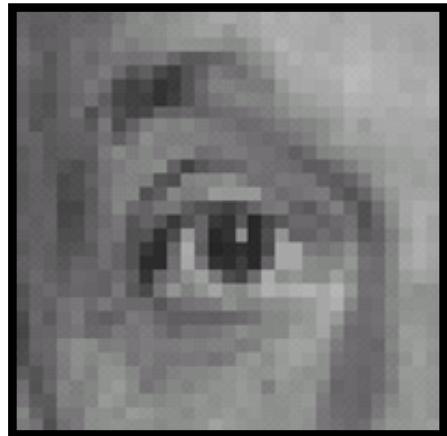
|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |



Shifted left  
By 1 pixel



# Practice with linear filters



Original

|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 2 | 0 |
| 0 | 0 | 0 |

-

$\frac{1}{9}$

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

?

(Note that filter sums to 1)



# Practice with linear filters

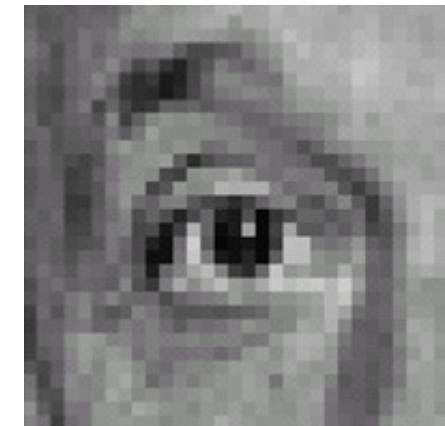


Original

|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 2 | 0 |
| 0 | 0 | 0 |

-

|               |   |   |   |
|---------------|---|---|---|
| $\frac{1}{9}$ | 1 | 1 | 1 |
| 1             | 1 | 1 | 1 |
| 1             | 1 | 1 | 1 |

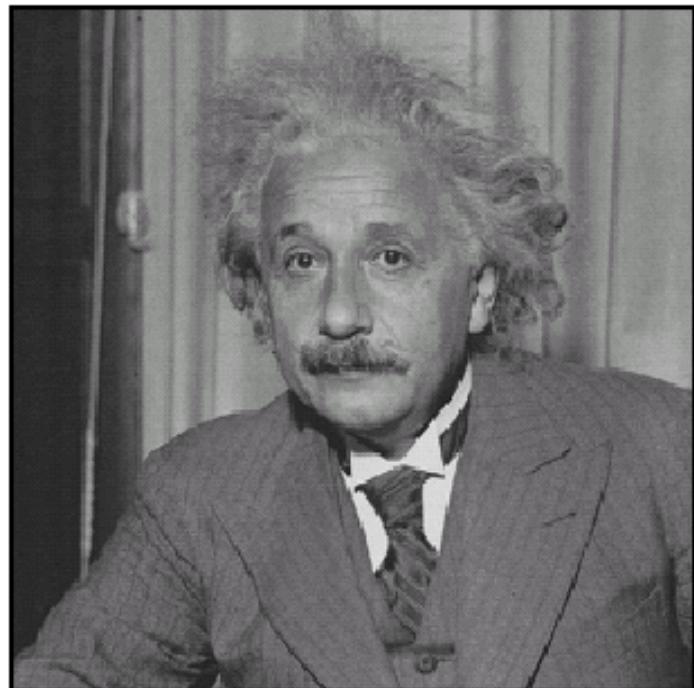


## Sharpening filter

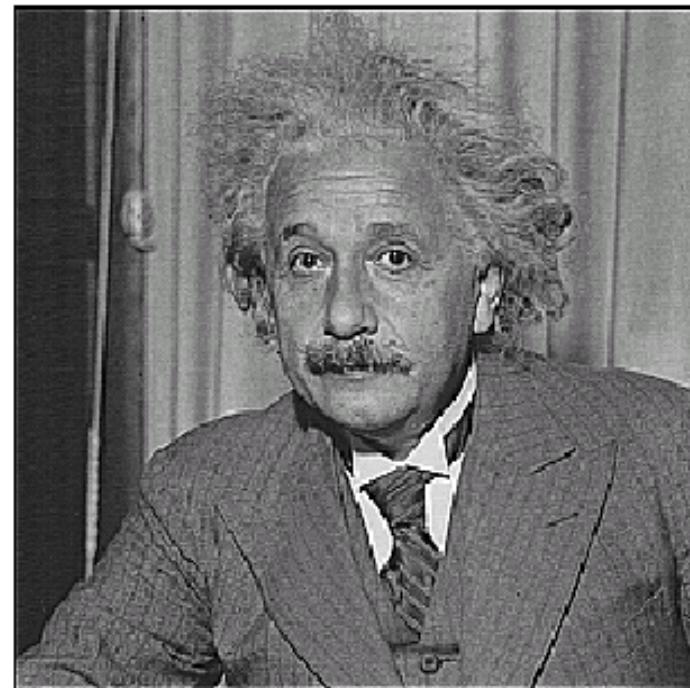
- Accentuates (突顯) differences with local average



# Sharpening



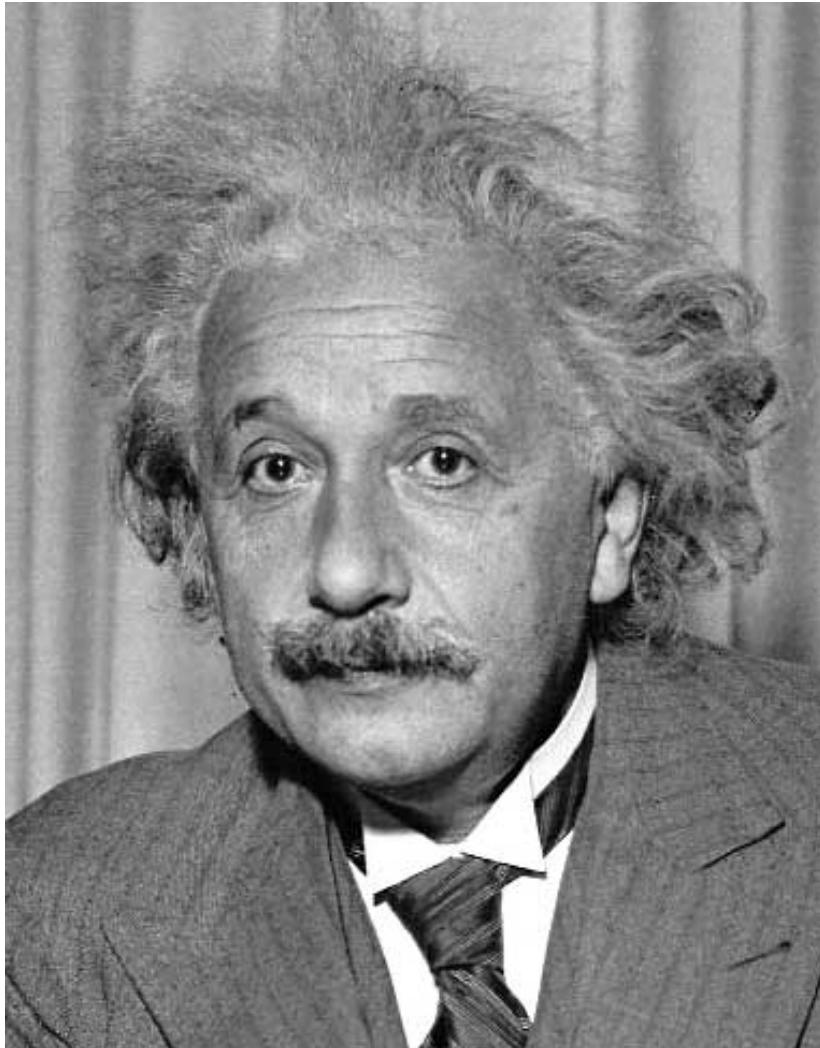
**before**



**after**

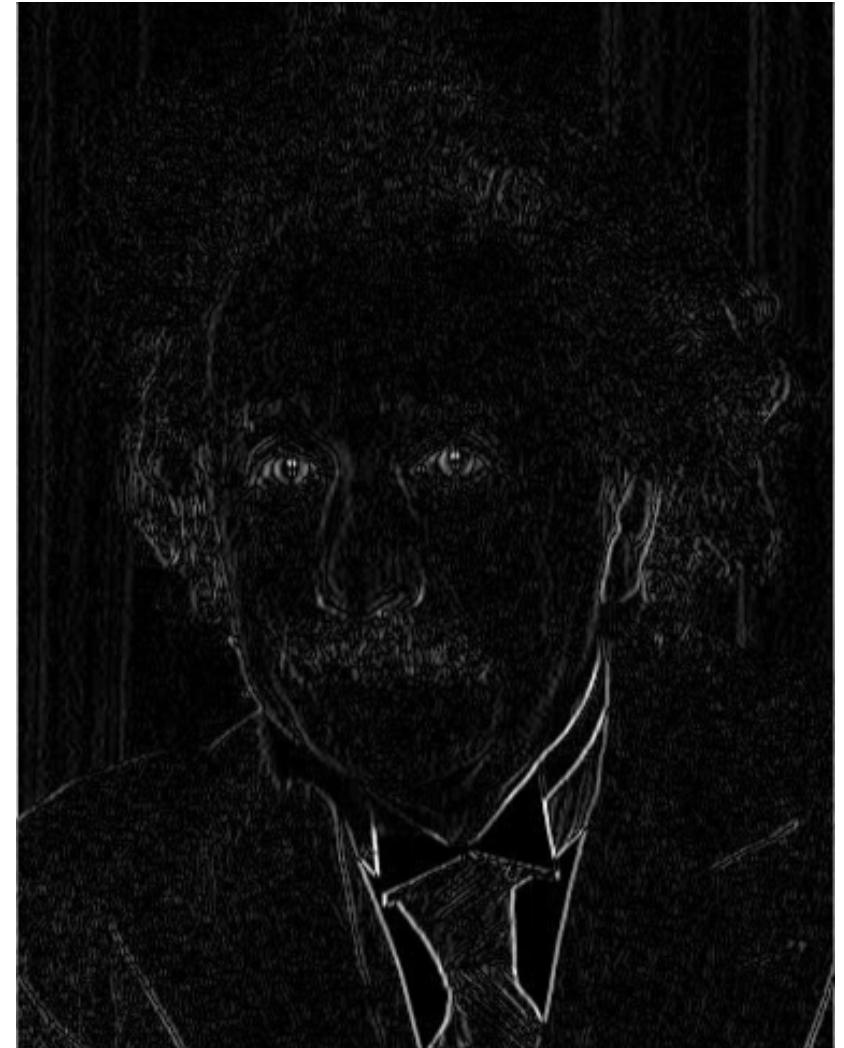


# Other filters



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

Sobel

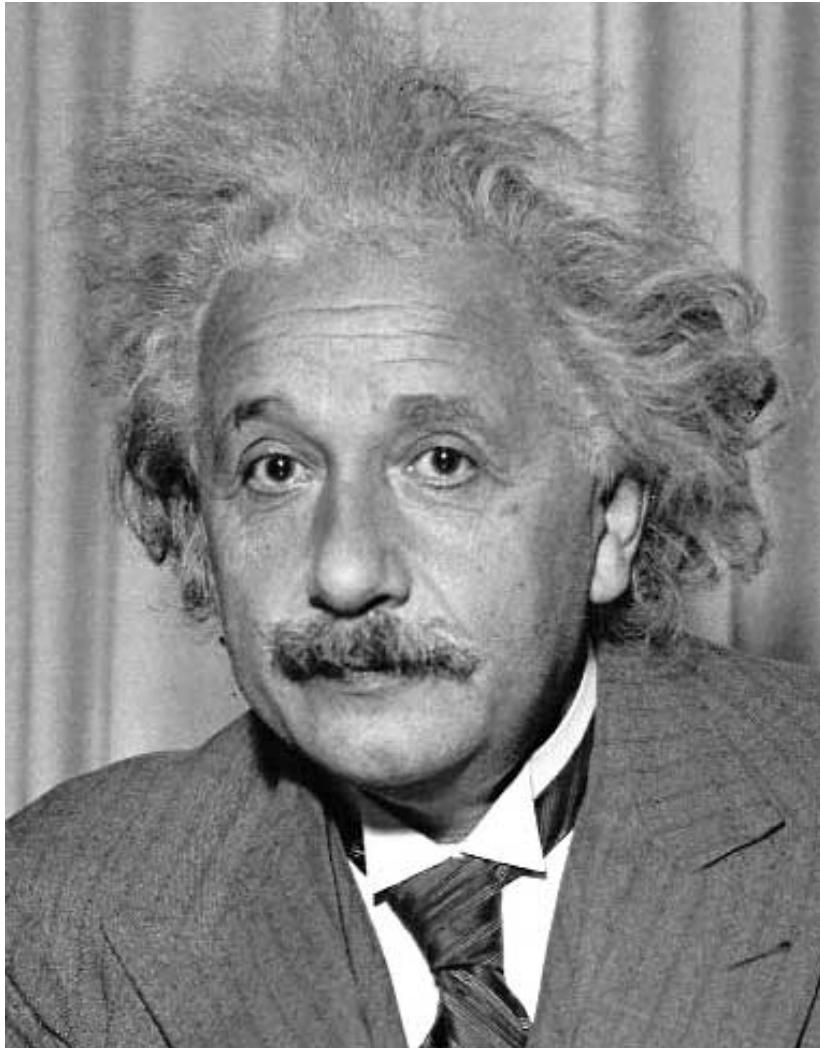


Vertical Edge  
(absolute value)

[http://en.wikipedia.org/wiki/Sobel\\_operator](http://en.wikipedia.org/wiki/Sobel_operator)

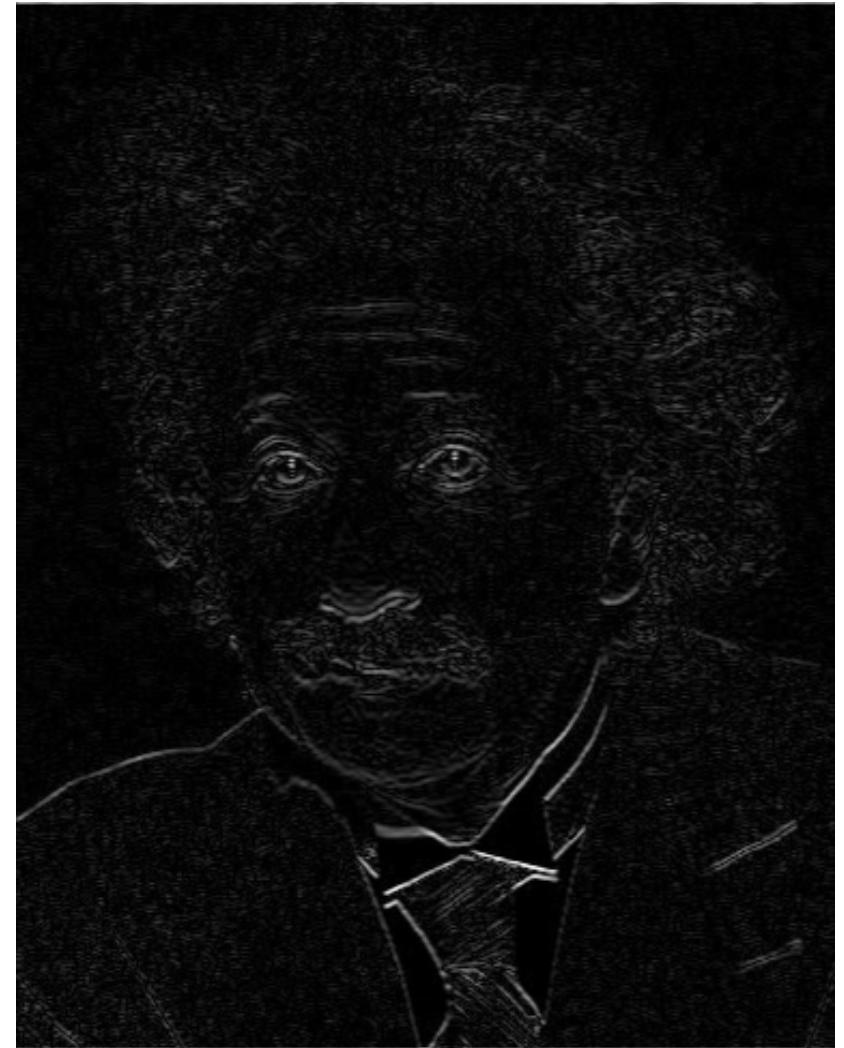


# Other filters



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

Sobel



Horizontal Edge  
(absolute value)



# Filtering vs. Convolution

- filtering

- `h=filter2(g,f);` or

- `h=imfilter(f,g);`

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

g=filter                    f=image

- 2d convolution

- `h=conv2(g,f);`

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

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



# Key properties of linear filters

## Linearity:

$$\text{Conv2}(g_1 + g_2, f) = \text{conv2}(g_1, f) + \text{conv2}(g_2, f)$$

**Shift invariance:** same behavior regardless of pixel location

$$\text{Conv2}(\text{shift}(g), f) = \text{shift}(\text{conv2}(g, f))$$

Any linear, shift-invariant operator can be represented as a **convolution**



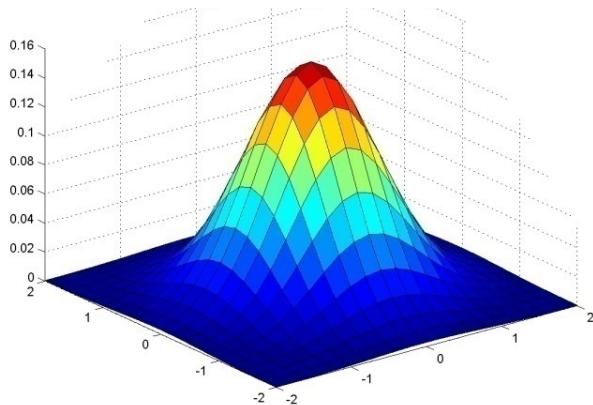
# More properties

- Commutative:  $a * b = b * a$ 
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality
- 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  $e = [0, 0, 1, 0, 0]$ ,  
 $a * e = a$



# Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness

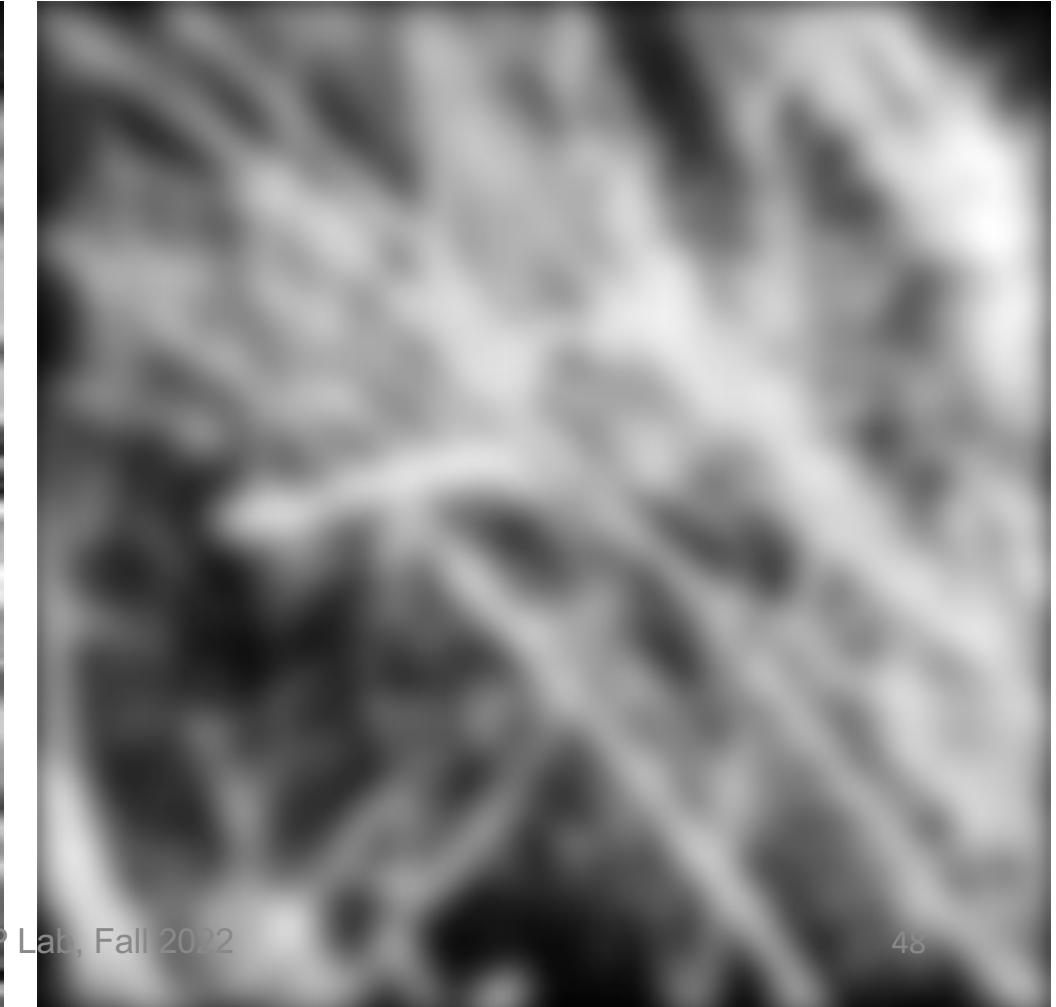
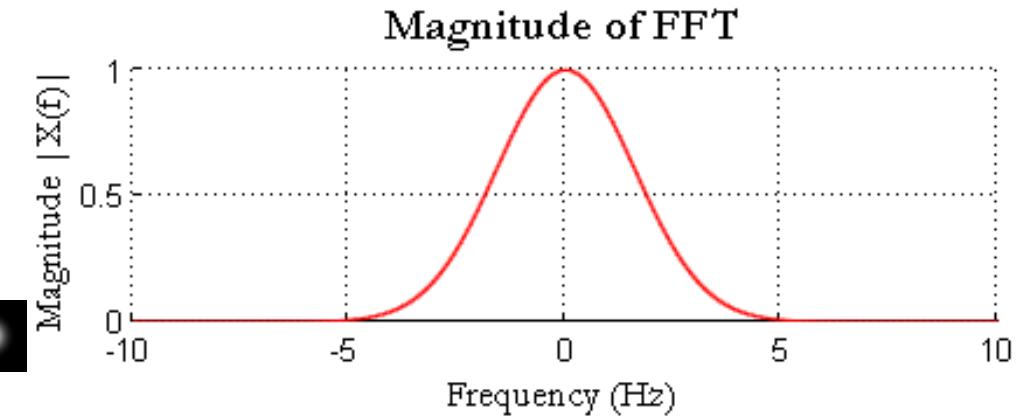
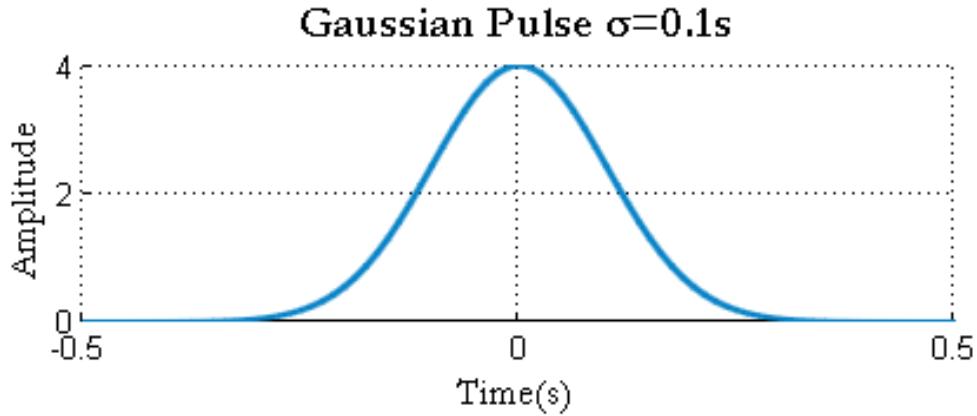


|       |       |       |       |       |
|-------|-------|-------|-------|-------|
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.022 | 0.097 | 0.159 | 0.097 | 0.022 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |

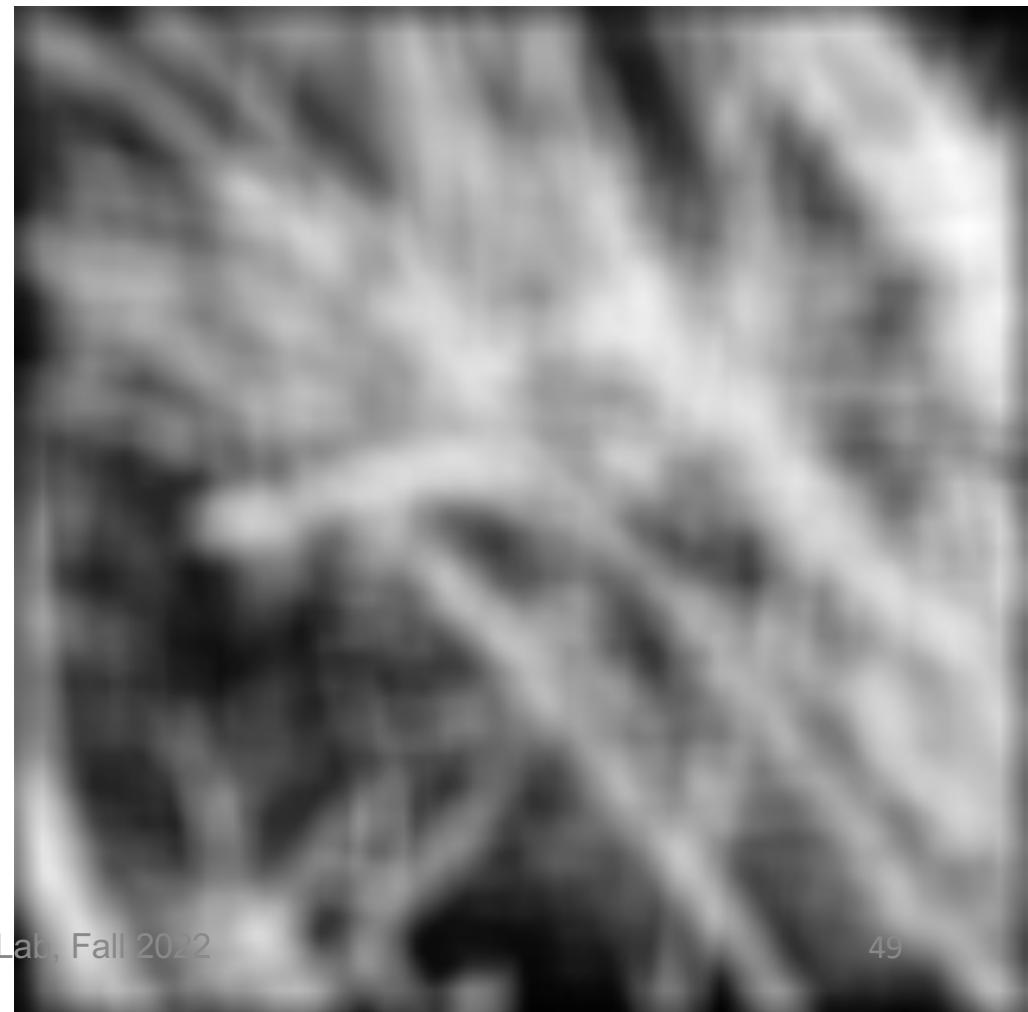
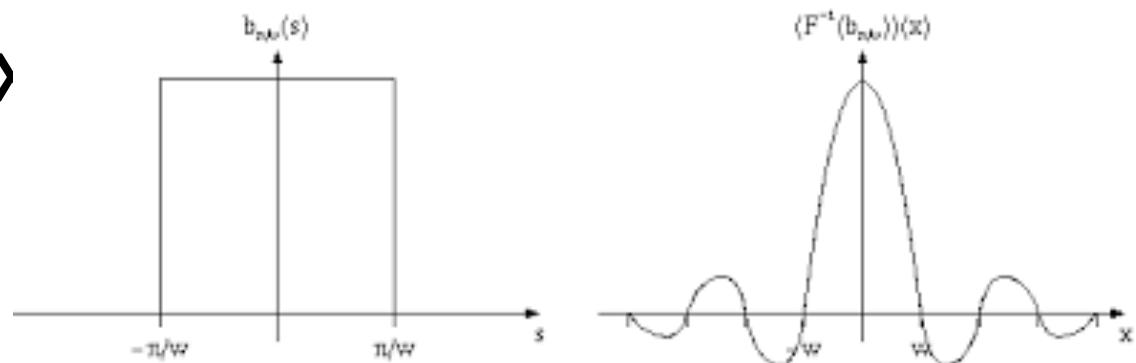
$5 \times 5, \sigma = 1$

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

# Smoothing with Gaussian filter



# Smoothing with bo





# Gaussian filters

- Remove “high-frequency” components from the image (**low-pass filter**)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width  $\sigma$  is same as convolving once with kernel of width  $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians



# Separability of the Gaussian filter

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 + y^2}{2\sigma^2}} \\ &= \left( \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left( \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

The 2D Gaussian can be expressed as the product of two functions, one a function of  $x$  and the other a function of  $y$

In this case, the two functions are the (identical) 1D Gaussian



# Separability example

2D convolution  
(center location only)

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} * \begin{matrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{matrix}$$

The filter factors  
into a product of 1D  
filters:

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} = \begin{matrix} 1 \\ 2 \\ 1 \end{matrix} \times \begin{matrix} 1 & 2 & 1 \end{matrix}$$

Perform convolution  
along rows:

$$\begin{matrix} 1 & 2 & 1 \end{matrix} * \begin{matrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{matrix} = \begin{matrix} 11 \\ 18 \\ 18 \end{matrix}$$

Followed by convolution  
along the remaining column:



# Separability

- Why is separability useful in practice?
  - Full filter: Filtering an  $M$ -by- $N$  image with a  $P$ -by- $Q$  filter kernel requires roughly  $MNPQ$  multiplies and adds.
  - Separable filter: The first step requires about  $MNP$  multiplies and adds. The second requires about  $MNQ$  multiplies and adds, for a total of  $MN(P + Q)$ .

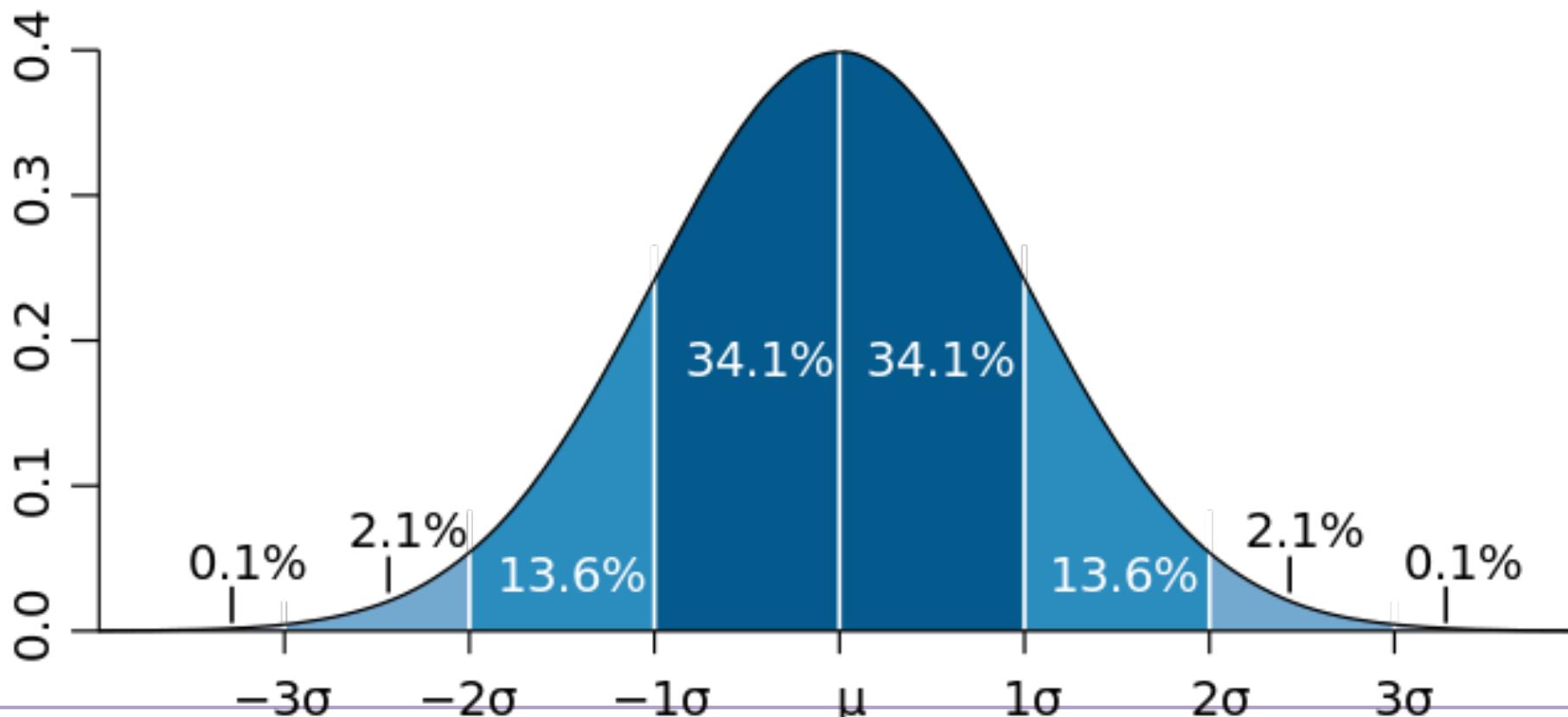
<http://blogs.mathworks.com/steve/2006/10/04/separable-convolution/>



# Practical matters

## How big should the filter be?

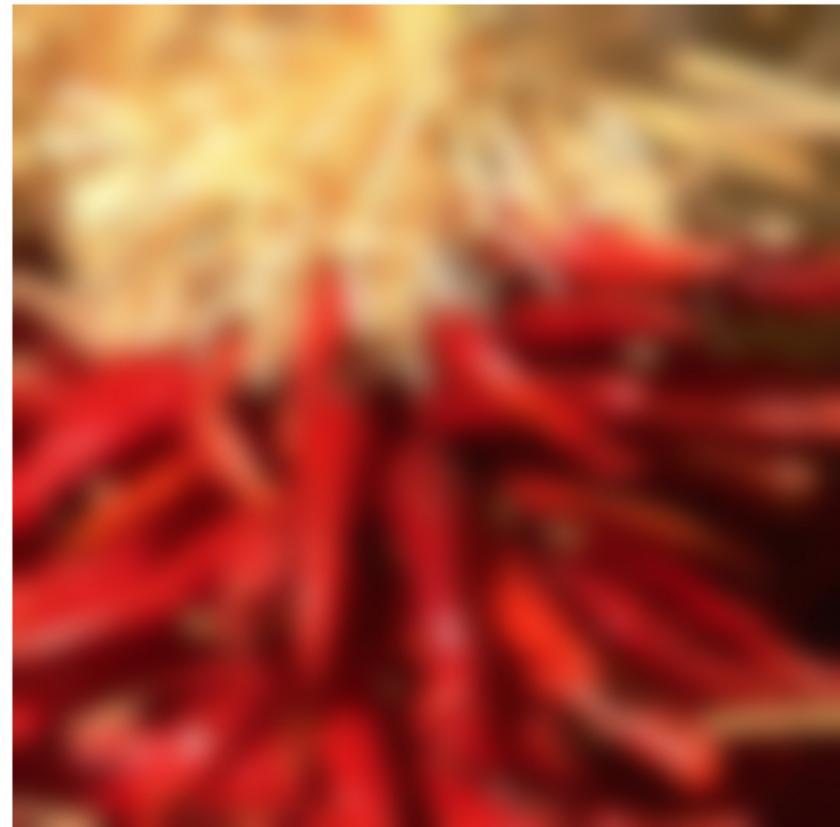
- Values at edges should be near zero
- Rule of thumb for Gaussian: set filter half-width to about  $3\sigma$





# Practical matters

- What about near the image edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge





# Practical matters

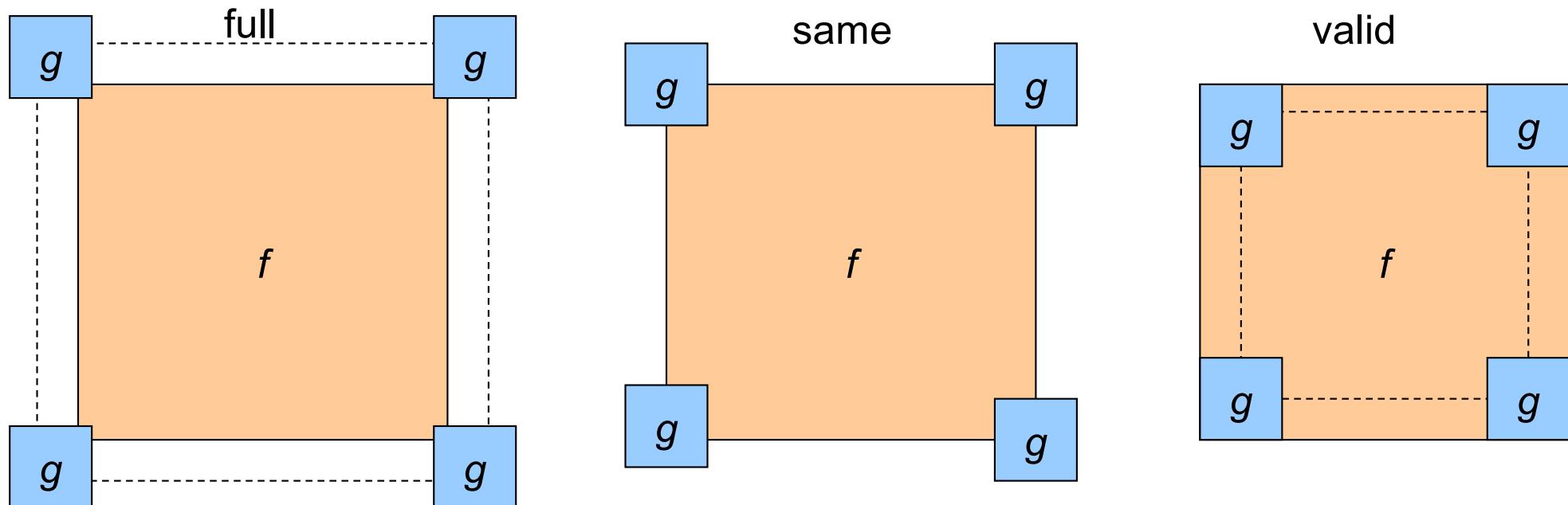
## – methods (MATLAB):

- clip filter (black): `imfilter(f, g, 0)`
- wrap around: `imfilter(f, g, 'circular')`
- copy edge: `imfilter(f, g, 'replicate')`
- reflect across edge: `imfilter(f, g, 'symmetric')`



# Practical matters

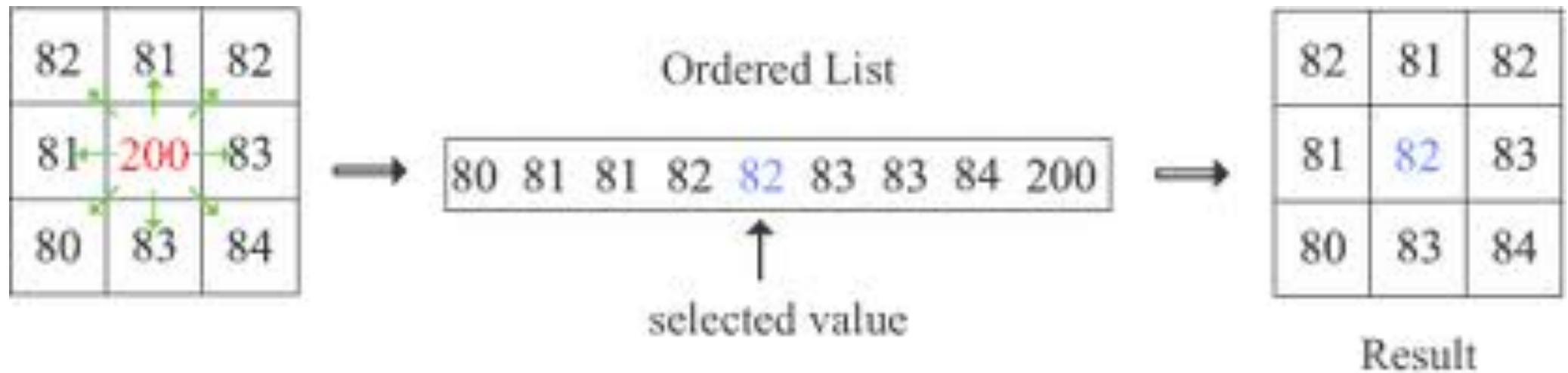
- What is the size of the output?
- MATLAB: `filter2(g, f, shape)`
  - `shape = 'full'` : output size is sum of sizes of  $f$  and  $g$
  - `shape = 'same'` : output size is same as  $f$
  - `shape = 'valid'` : output size is difference of sizes of  $f$  and  $g$





# Median filters

- A **Median Filter** operates over a window by selecting the median intensity in the window.

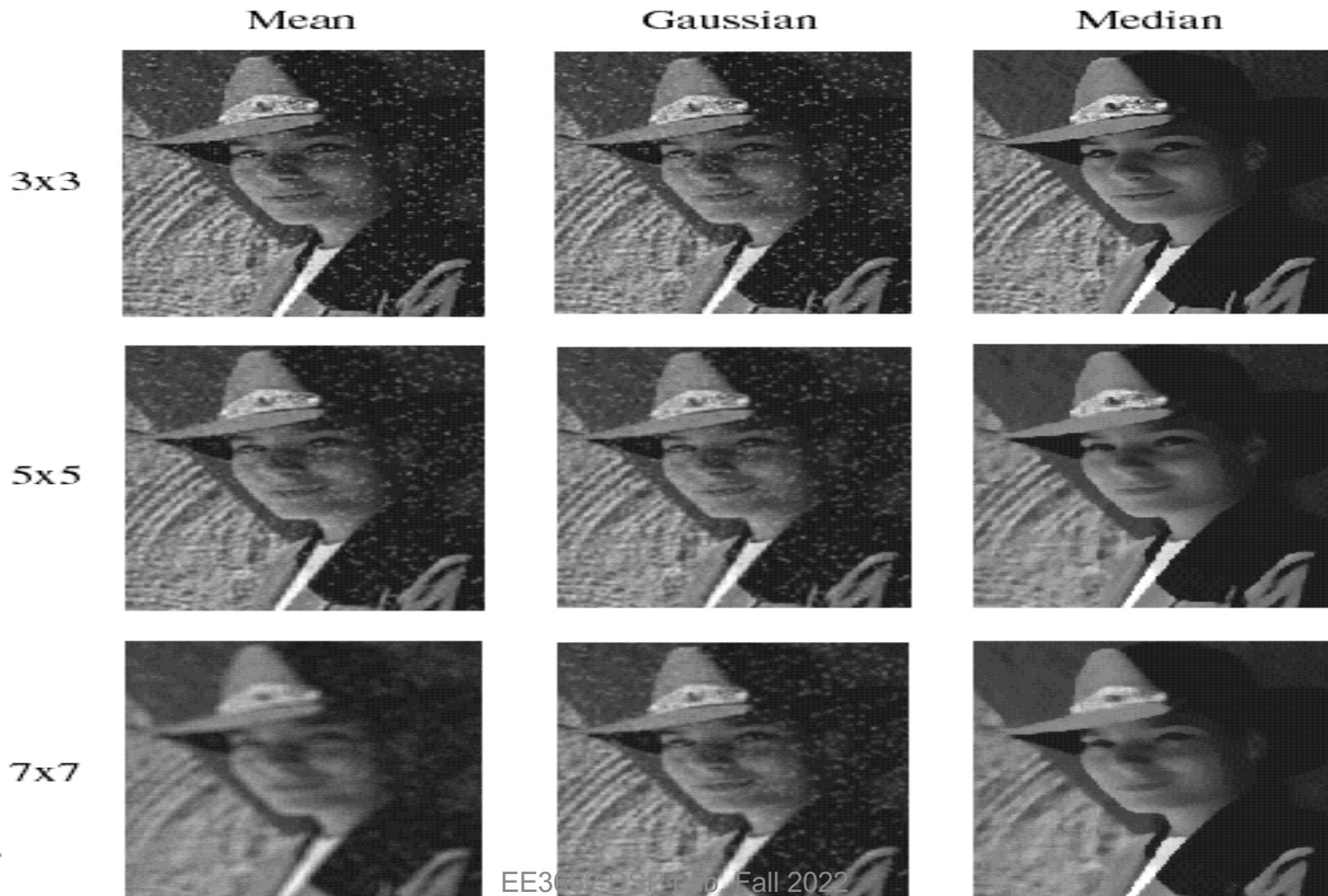


$B = \text{ordfilt2}(A, \text{order}, \text{domain})$   
In matlab



# Median filters

- What advantage does a median filter have over a mean filter? (salt and pepper noise)





# Median filters

- Is a median filter a kind of convolution?

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

$$h[m, n] = \text{median}_{kl}(f[m + k, n + l])$$

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 1 | 1 | 1 |

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 5 | 1 | 1 |
| 1 | 1 | 1 |

|   |   |   |
|---|---|---|
| 1 | 5 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Median:

1

1

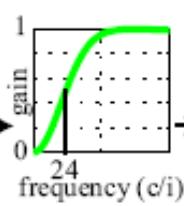
1



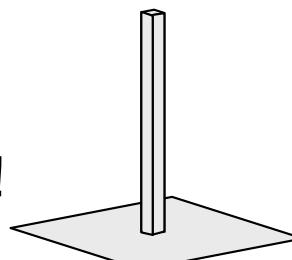
# Lab 6: Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns,  
[“Hybrid Images,”](#) SIGGRAPH 2006

Gaussian Filter!

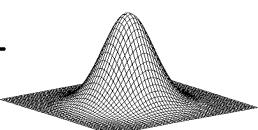


Laplacian Filter!



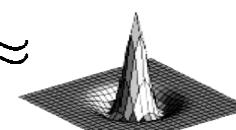
unit impulse

-



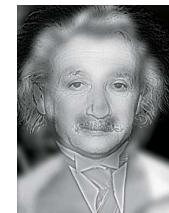
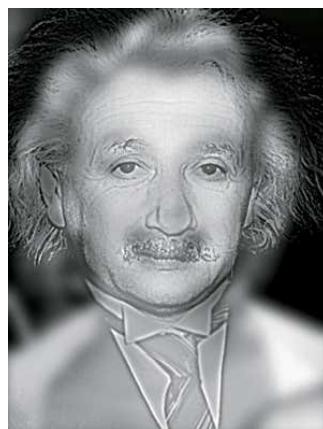
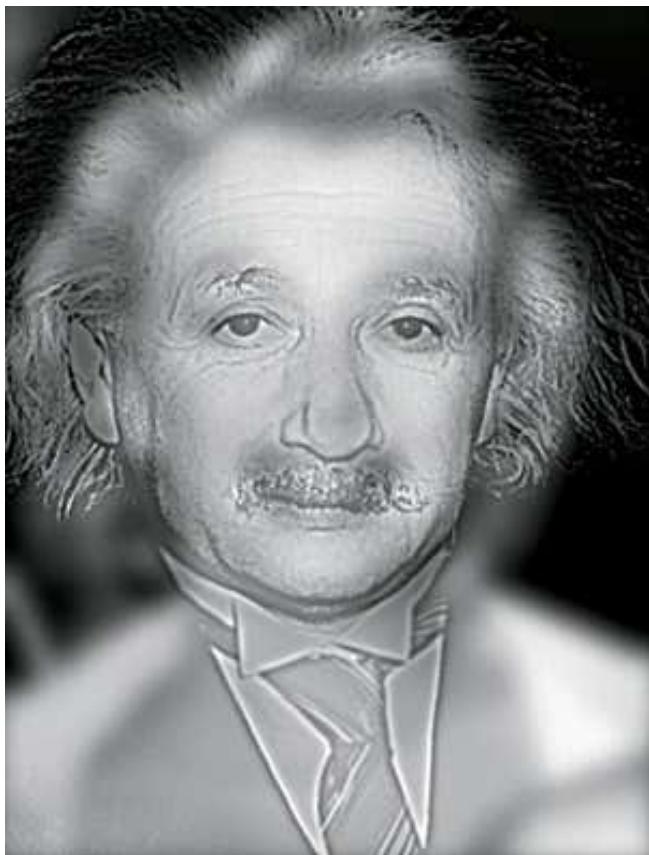
Gaussian

≈



Laplacian of Gaussian

**Down-sampling**





# Lecture Outline

- Image and Video Basics
- Recap on Image Filtering
- Interest Point
  - Corners
- Seam Carving

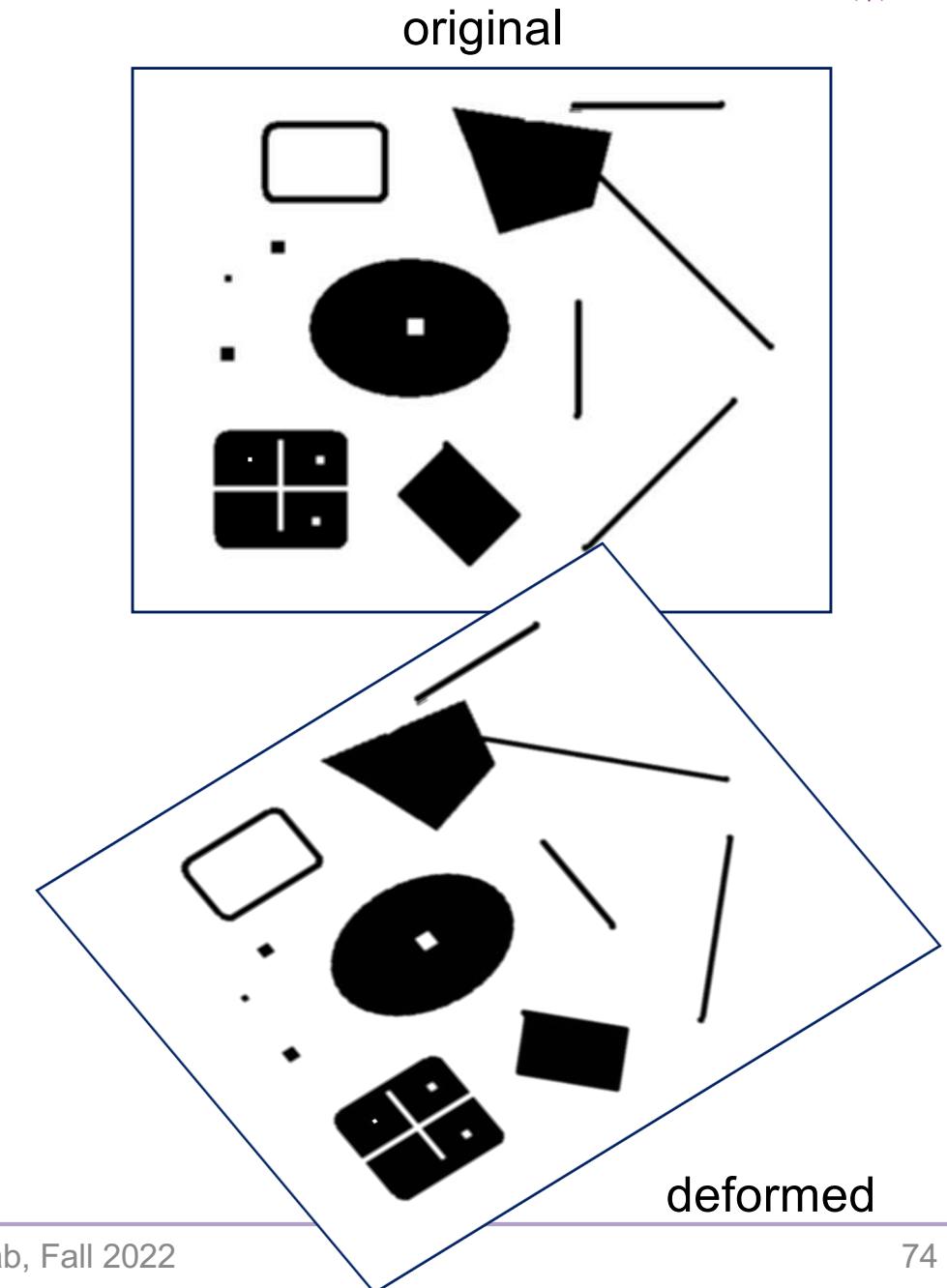


# Interest points

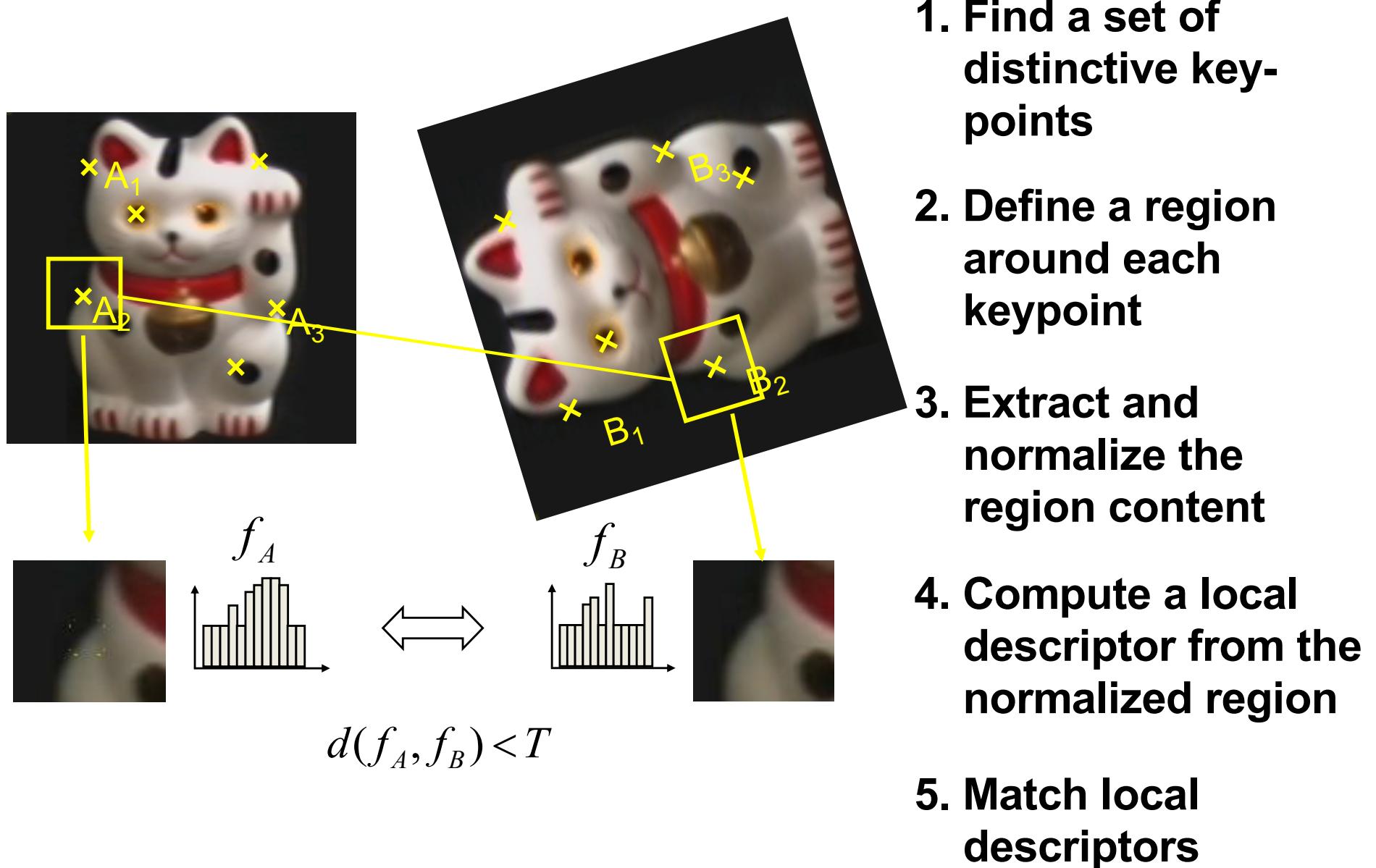
- Note: “interest points” = “keypoints”, also sometimes called “features”
- Many applications
  - tracking: which points are good to track?
  - recognition: find patches likely to tell us something about object category
  - 3D reconstruction: find correspondences across different views

# Interest points

- Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.
  - Which points would you choose?



# Overview of Keypoint Matching





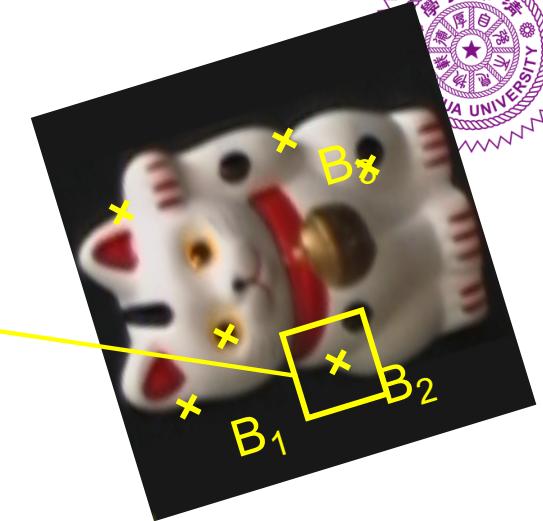
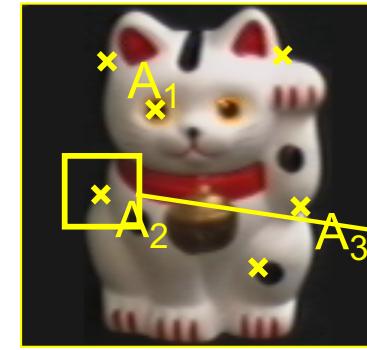
# Goals for Keypoints



Detect points that are *repeatable* and *distinctive*

*Repeatable*: the same feature can be found in several images despite geometric and photometric transformations

# Key trade-offs



## Detection of interest points



More Repeatable

Robust detection  
Precise localization

More Points

Robust to occlusion  
Works with less texture

## Description of patches



More Distinctive

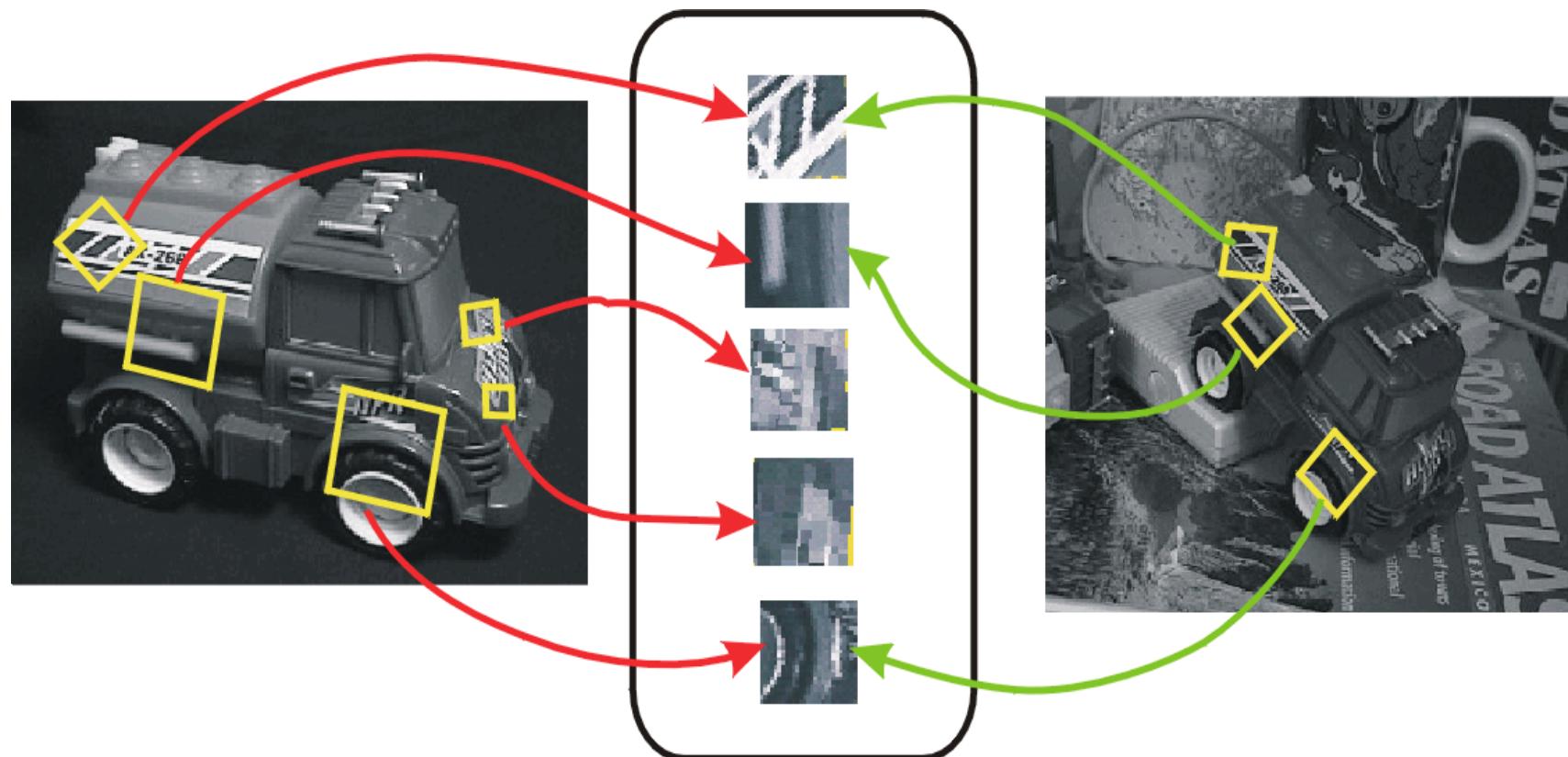
Minimize wrong matches

More Flexible

Robust to expected variations  
Maximize correct matches

# Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



**Feature Descriptors**



# Feature extraction: Corners



Slides from Rick Szeliski, Svetlana Lazebnik, and Kristin Grauman



# Many Existing Detectors Available

Hessian & Harris

[Beaudet '78], [Harris '88]

Laplacian, DoG

[Lindeberg '98], [Lowe 1999]

Harris-/Hessian-Laplace

[Mikolajczyk & Schmid '01]

Harris-/Hessian-Affine

[Mikolajczyk & Schmid '04]

EBR and IBR

[Tuytelaars & Van Gool '04]

MSER

[Matas '02]

Salient Regions

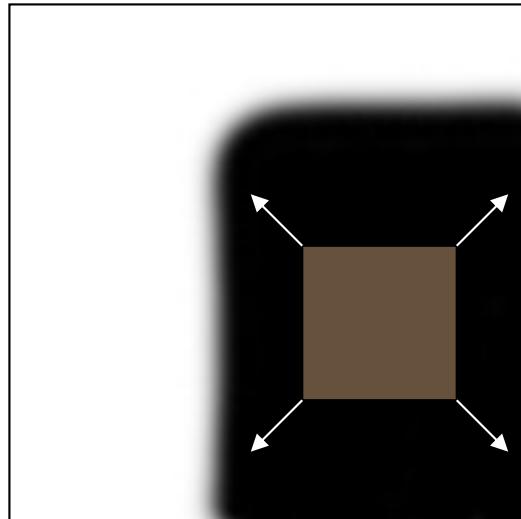
[Kadir & Brady '01]

Others...

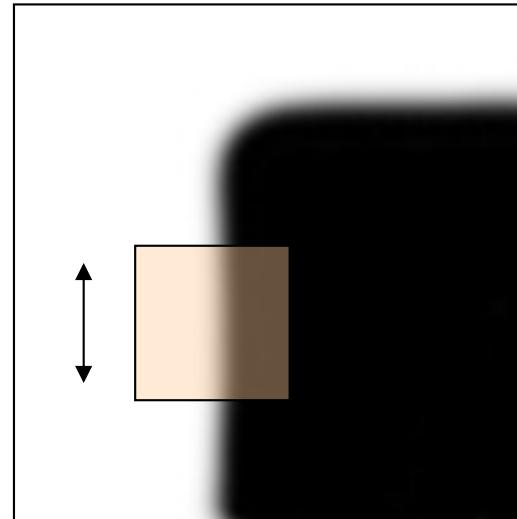


# Corner Detection (Lab 7): Basic Idea

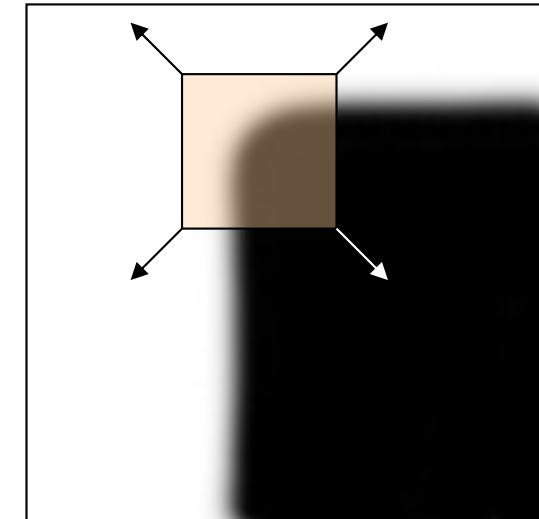
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



**“flat” region:**  
no change in  
all directions



**“edge”:**  
no change  
along the edge  
direction



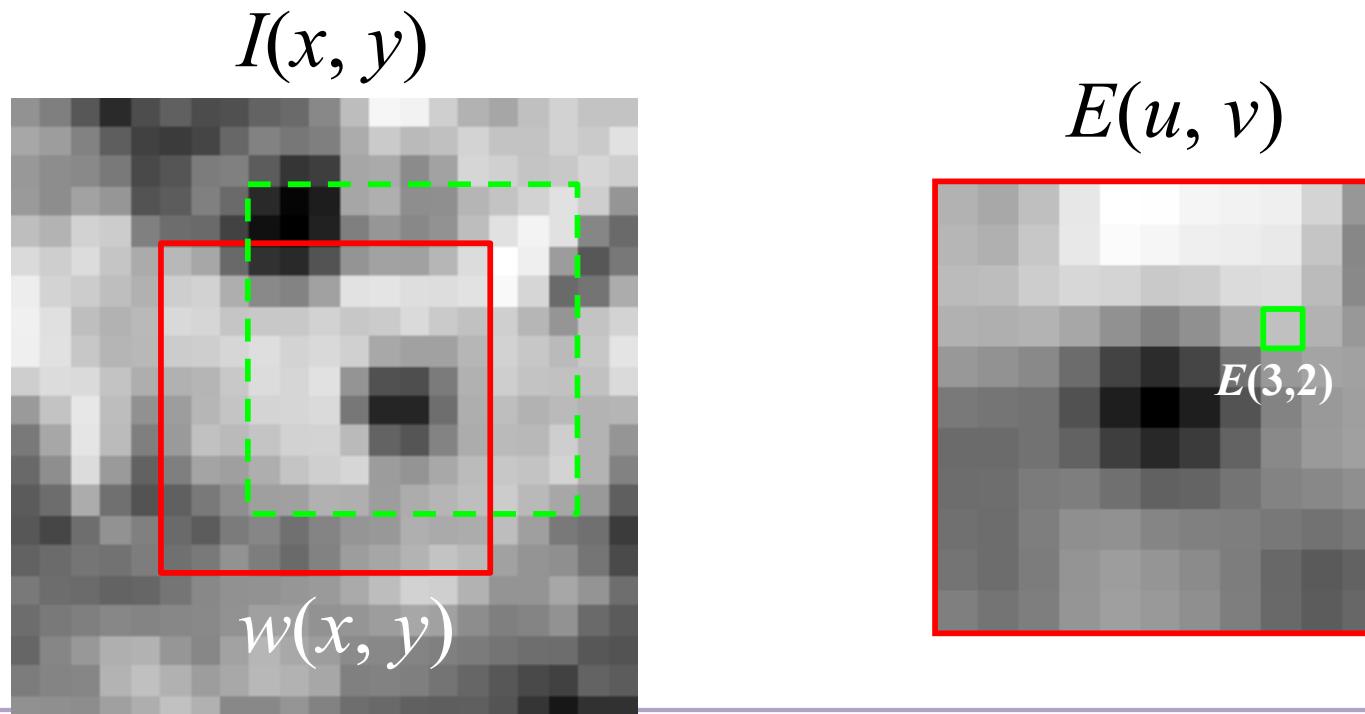
**“corner”:**  
significant  
change in all  
directions



# Corner Detection: Mathematics

Change in appearance of window  $w(x,y)$   
for the shift  $[u,v]$ :

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

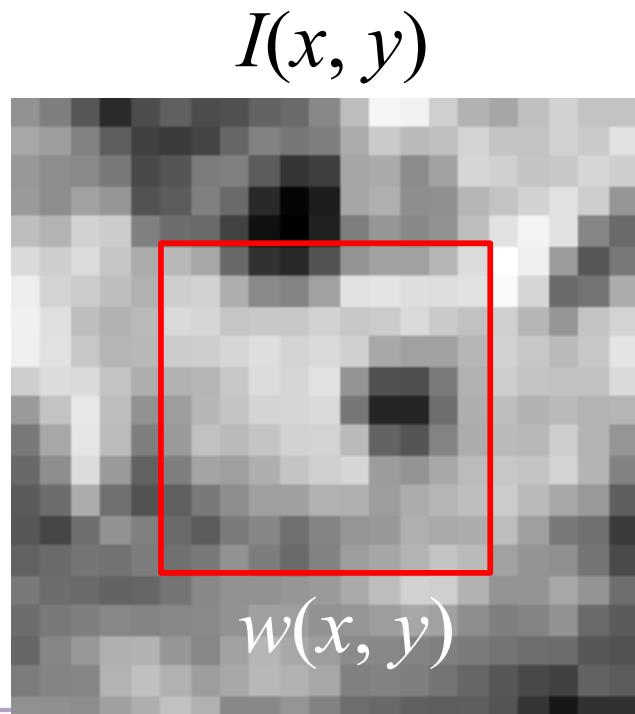




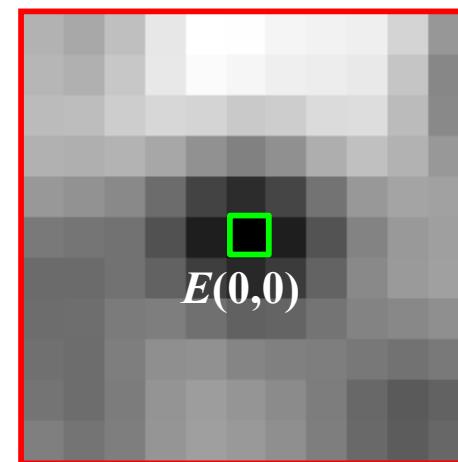
# Corner Detection: Mathematics

Change in appearance of window  $w(x,y)$   
for the shift  $[u,v]$ :

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$



$$E(u, v)$$





# Corner Detection: Mathematics

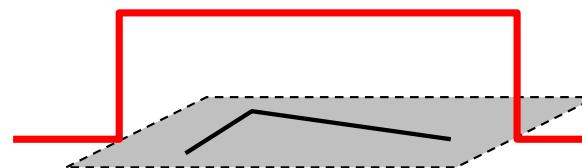
Change in appearance of window  $w(x,y)$   
for the shift  $[u,v]$ :

$$E(u,v) = \sum_{x,y} w(x,y) [I(x+u, y+v) - I(x, y)]^2$$

Diagram illustrating the components of the corner detection formula:

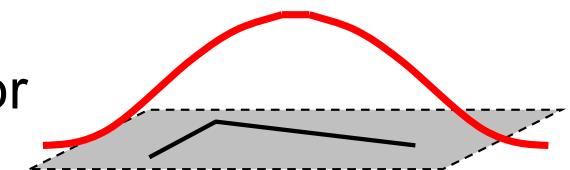
- Window function: A green rounded rectangle pointing to the term  $w(x,y)$ .
- Shifted intensity: A green rounded rectangle pointing to the term  $I(x+u, y+v)$ .
- Intensity: A green rounded rectangle pointing to the term  $I(x, y)$ .

Window function  $w(x,y) =$



1 in window, 0 outside

or



Gaussian



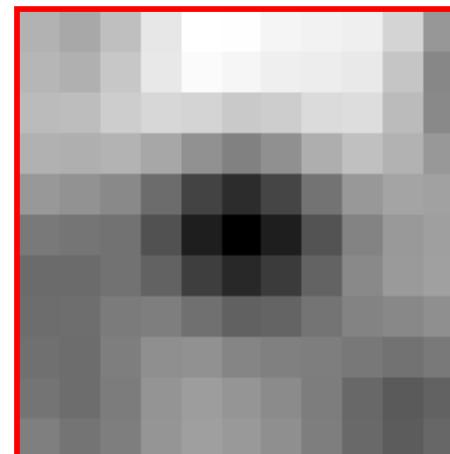
# Corner Detection: Mathematics

Change in appearance of window  $w(x,y)$   
for the shift  $[u,v]$ :

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

We want to find out how this function behaves for small shifts

$$E(u, v)$$





# Corner Detection: Mathematics

Change in appearance of window  $w(x,y)$   
for the shift  $[u,v]$ :

$$E(u,v) = \sum_{x,y} w(x,y) [I(x+u, y+v) - I(x, y)]^2$$

We want to find out how this function behaves for small shifts

Local quadratic approximation of  $E(u,v)$  in the neighborhood of  $(0,0)$  is given by the *second-order Taylor expansion*:

$$E(u,v) \approx E(0,0) + [u \ v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} [u \ v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

# Corner Detection: Mathematics



$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Second-order Taylor expansion of  $E(u, v)$  about  $(0, 0)$ :

$$E(u, v) \approx E(0, 0) + [u \ v] \begin{bmatrix} E_u(0, 0) \\ E_v(0, 0) \end{bmatrix} + \frac{1}{2} [u \ v] \begin{bmatrix} E_{uu}(0, 0) & E_{uv}(0, 0) \\ E_{uv}(0, 0) & E_{vv}(0, 0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E_u(u, v) = \sum_{x,y} 2w(x, y) [I(x+u, y+v) - I(x, y)] I_x(x+u, y+v)$$

$$\begin{aligned} E_{uu}(u, v) = & \sum_{x,y} 2w(x, y) I_x(x+u, y+v) I_x(x+u, y+v) \\ & + \sum_{x,y} 2w(x, y) [I(x+u, y+v) - I(x, y)] I_{xx}(x+u, y+v) \end{aligned}$$

$$\begin{aligned} E_{uv}(u, v) = & \sum_{x,y} 2w(x, y) I_y(x+u, y+v) I_x(x+u, y+v) \\ & + \sum_{x,y} 2w(x, y) [I(x+u, y+v) - I(x, y)] I_{xy}(x+u, y+v) \end{aligned}$$

# Corner Detection: Mathematics



$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Second-order Taylor expansion of  $E(u, v)$  about  $(0, 0)$ :

$$E(u, v) \approx E(0, 0) + [u \ v] \begin{bmatrix} E_u(0, 0) \\ E_v(0, 0) \end{bmatrix} + \frac{1}{2} [u \ v] \begin{bmatrix} E_{uu}(0, 0) & E_{uv}(0, 0) \\ E_{uv}(0, 0) & E_{vv}(0, 0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0, 0) = 0$$

$$E_u(0, 0) = 0$$

$$E_v(0, 0) = 0$$

$$E_{uu}(0, 0) = \sum_{x,y} 2w(x, y) I_x(x, y) I_x(x, y)$$

$$E_{vv}(0, 0) = \sum_{x,y} 2w(x, y) I_y(x, y) I_y(x, y)$$

$$E_{uv}(0, 0) = \sum_{x,y} 2w(x, y) I_x(x, y) I_y(x, y)$$



# Corner Detection: Mathematics

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Second-order Taylor expansion of  $E(u, v)$  about  $(0, 0)$ :

$$E(u, v) \approx [u \ v] \begin{bmatrix} \sum_{x,y} w(x, y) I_x^2(x, y) & \sum_{x,y} w(x, y) I_x(x, y) I_y(x, y) \\ \sum_{x,y} w(x, y) I_x(x, y) I_y(x, y) & \sum_{x,y} w(x, y) I_y^2(x, y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x, y) I_x(x, y) I_x(x, y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x, y) I_y(x, y) I_y(x, y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x, y) I_x(x, y) I_y(x, y)$$

# Corner Detection: Mathematics



The quadratic approximation simplifies to

$$E(u, v) \approx [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

where  $M$  is a *second moment matrix* computed from image derivatives:

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

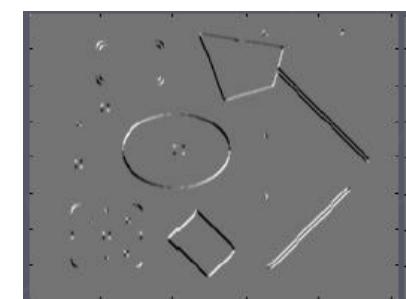
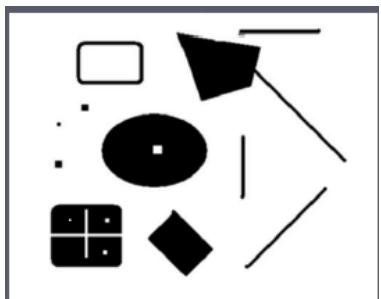
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$



# Corners as distinctive interest points

$$M = \sum w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).



Notation:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x}$$

$$I_y \Leftrightarrow \frac{\partial I}{\partial y}$$

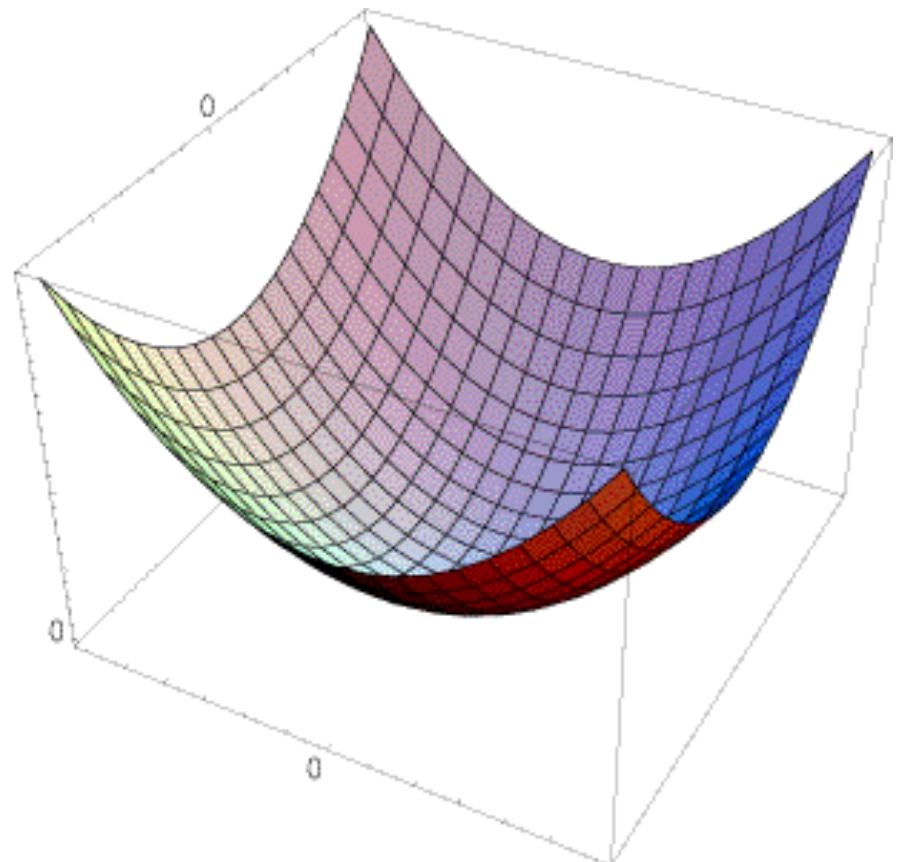
$$I_x I_y \Leftrightarrow \frac{\partial I}{\partial x} \frac{\partial I}{\partial y}$$

# Interpreting the second moment matrix

The surface  $E(u,v)$  is locally approximated by a quadratic form. Let's try to understand its shape.

$$E(u,v) \approx [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

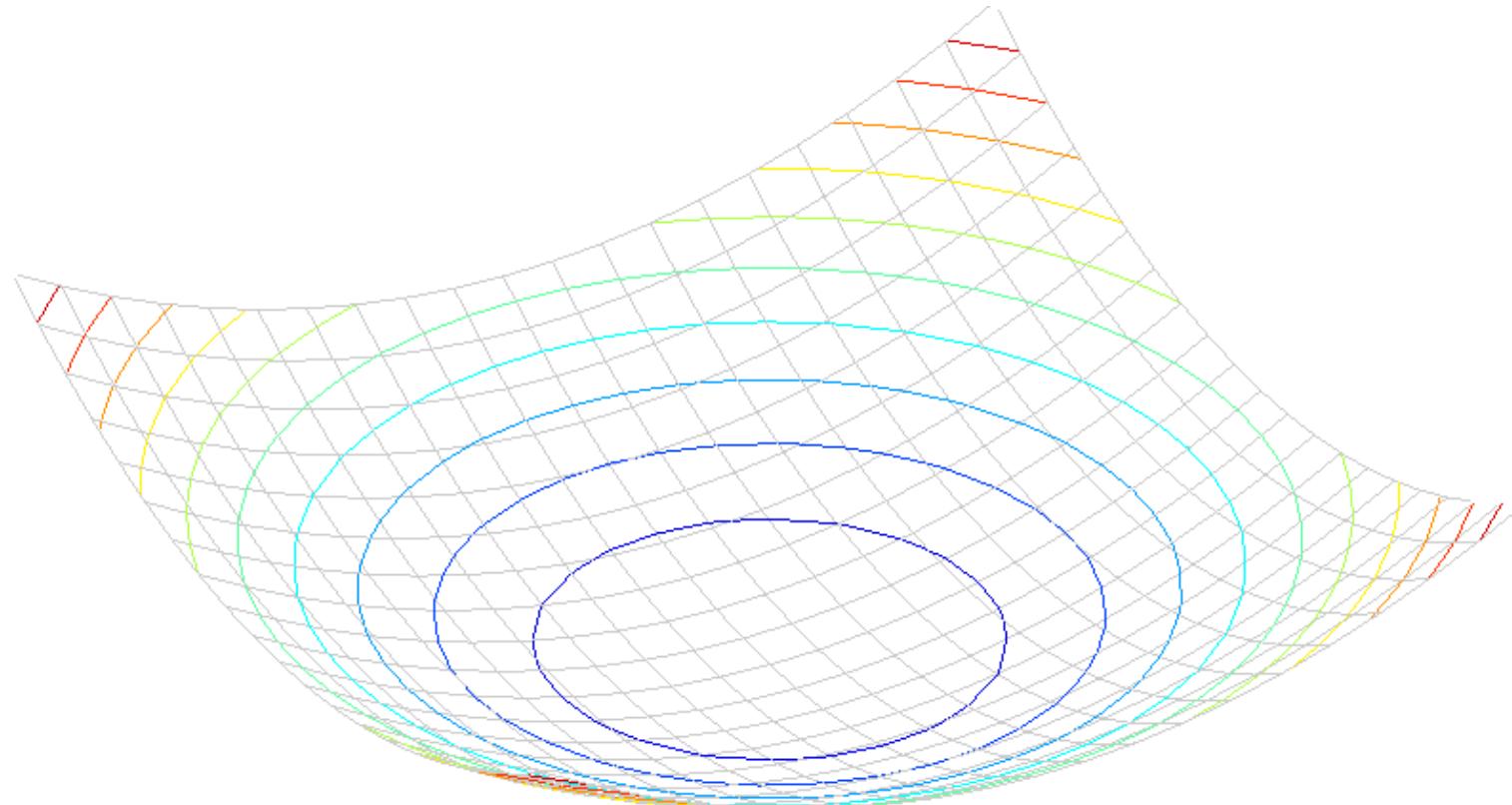




# Interpreting the second moment matrix

Consider a horizontal “slice” of  $E(u, v)$ :  $[u \ v] M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.





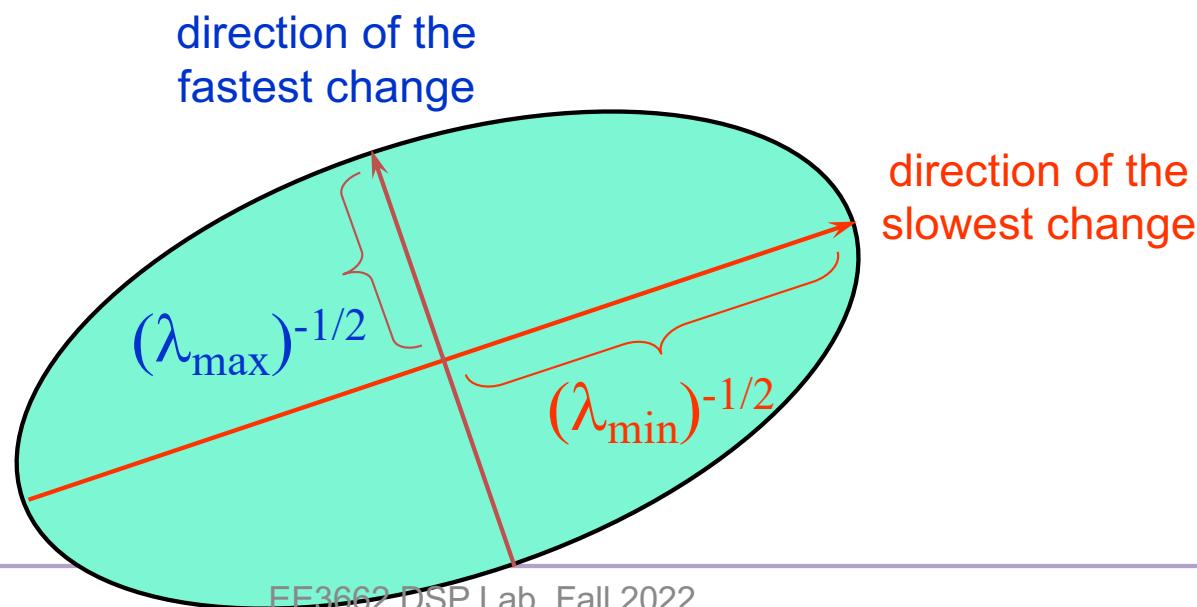
# Interpreting the second moment matrix

Consider a horizontal “slice” of  $E(u, v)$ :  $[u \ v] M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.

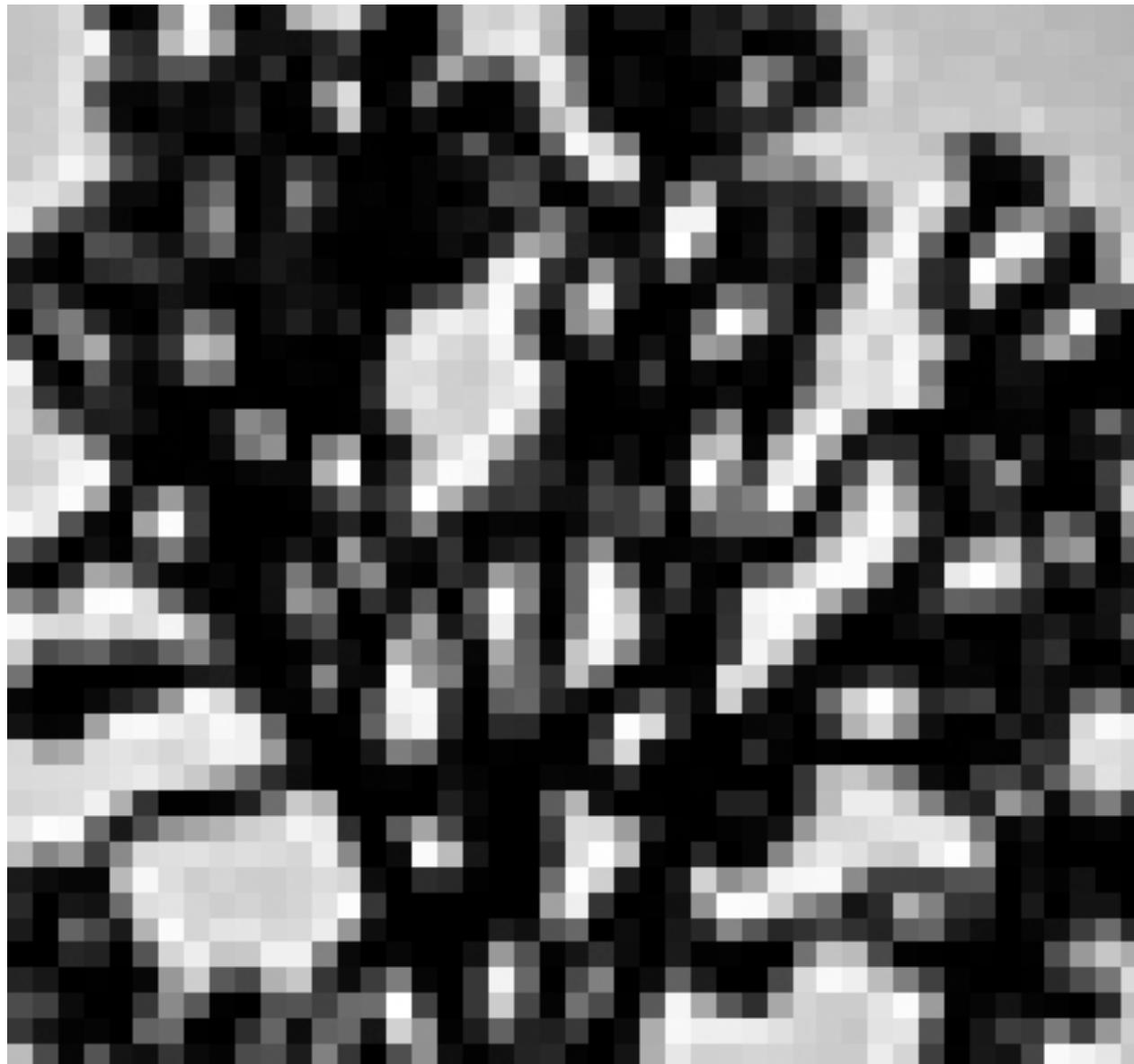
Diagonalization of  $M$ :  $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by  $R$



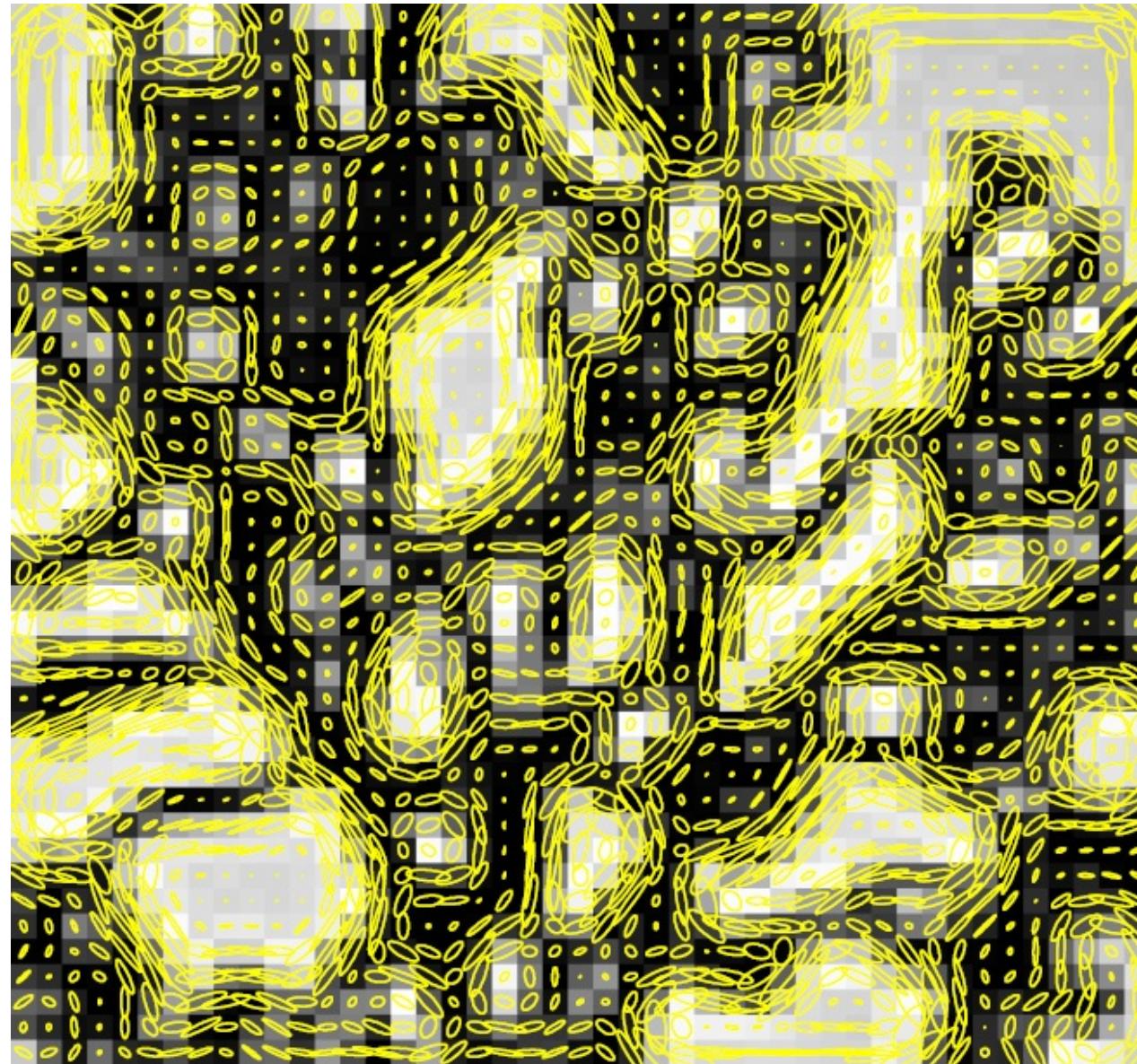


# Visualization of second moment matrices





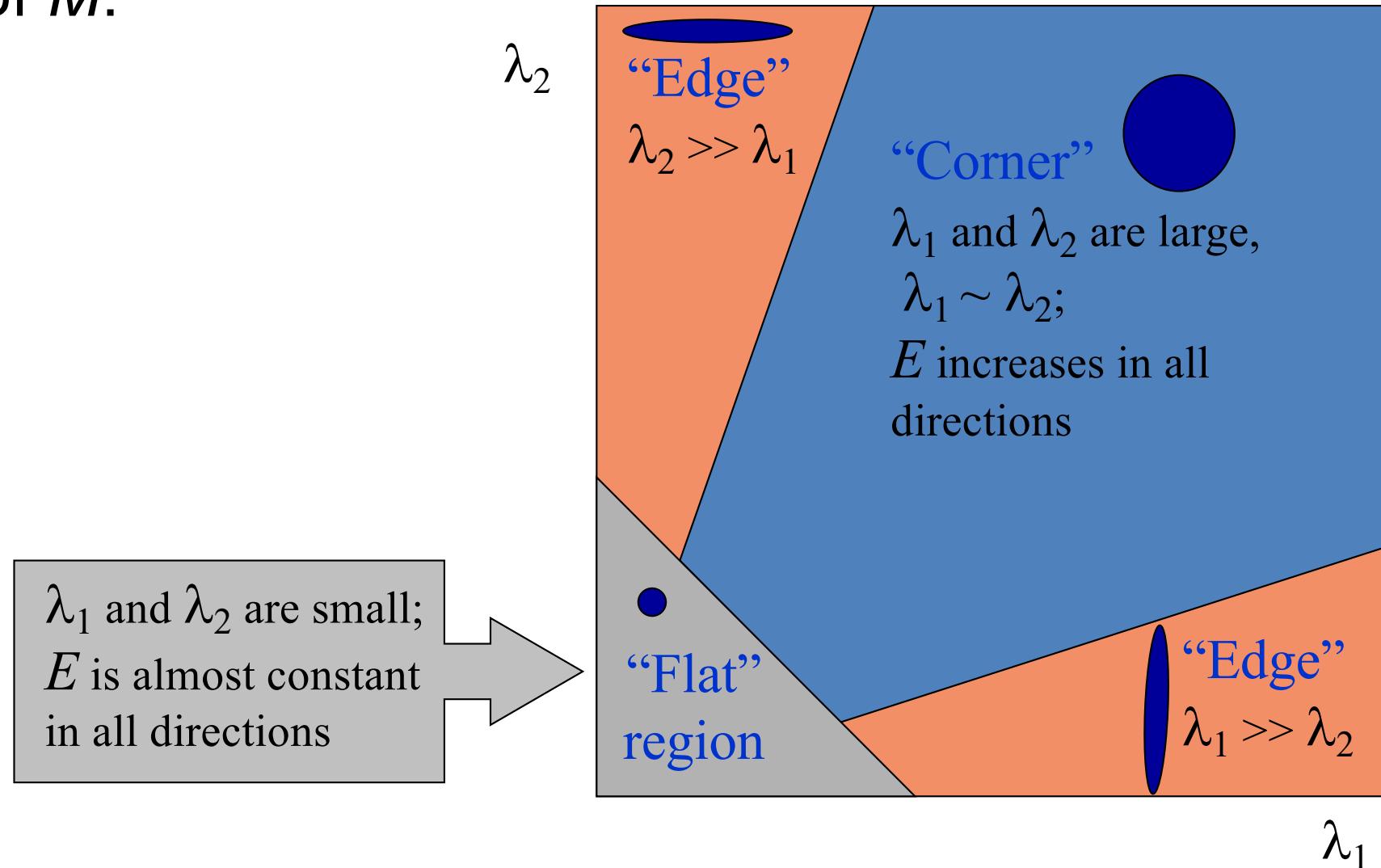
# Visualization of second moment matrices





# Interpreting the eigenvalues

Classification of image points using eigenvalues of  $M$ :

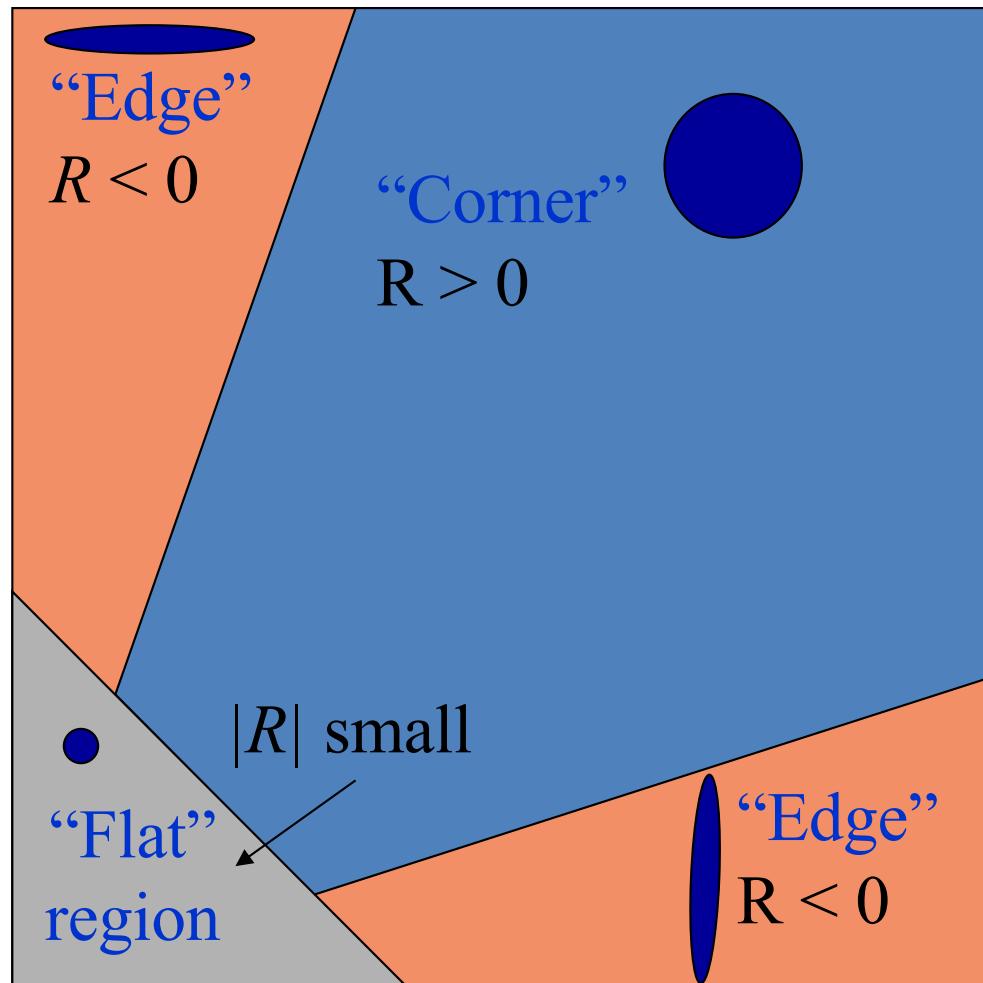




# Corner response function

$$R = \det(M) - \alpha \operatorname{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

$\alpha$ : constant (0.04 to 0.06)





# Harris corner detector

- 1) Compute  $M$  matrix for each image window to get their *cornerness* scores.
- 2) Find points whose surrounding window gave large corner response ( $R >$  threshold)
- 3) Take the points of local maxima, i.e., perform non-maximum suppression

C.Harris and M.Stephens. "[A Combined Corner and Edge Detector.](#)"  
*Proceedings of the 4th Alvey Vision Conference:* pages 147—151, 1988.

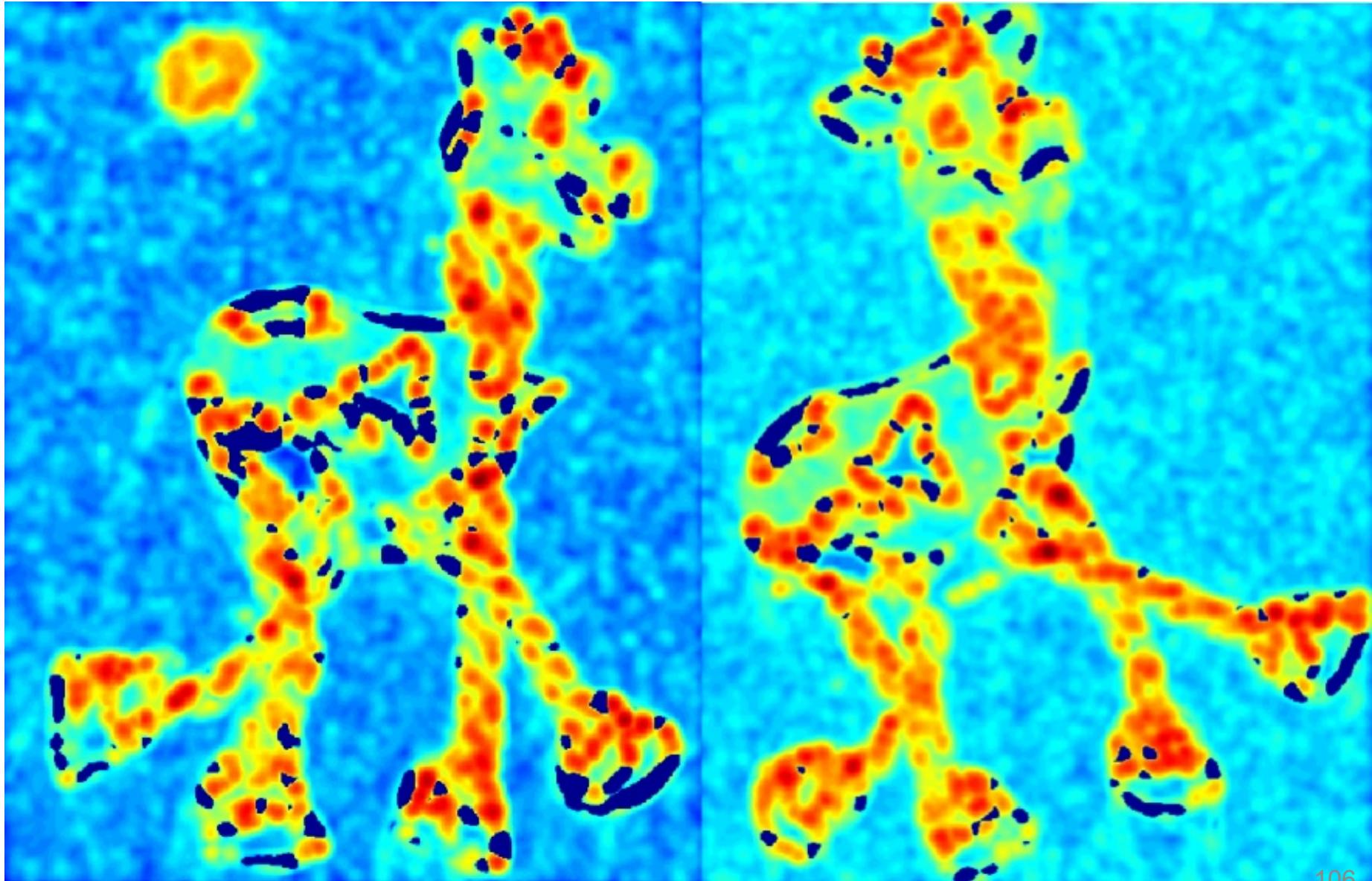
# Harris Detector: Steps



105

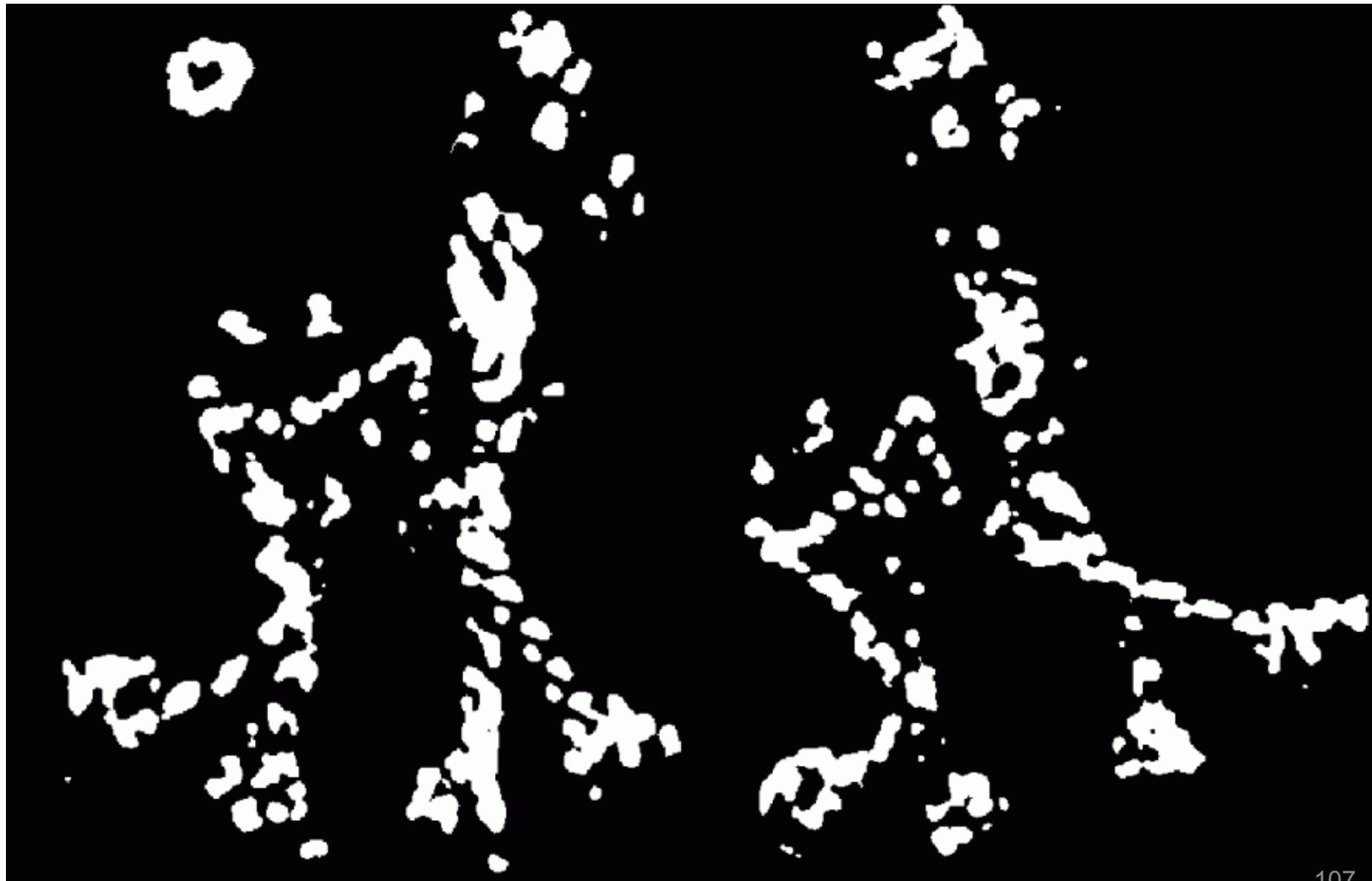
# Harris Detector: Steps

Compute corner response  $R$



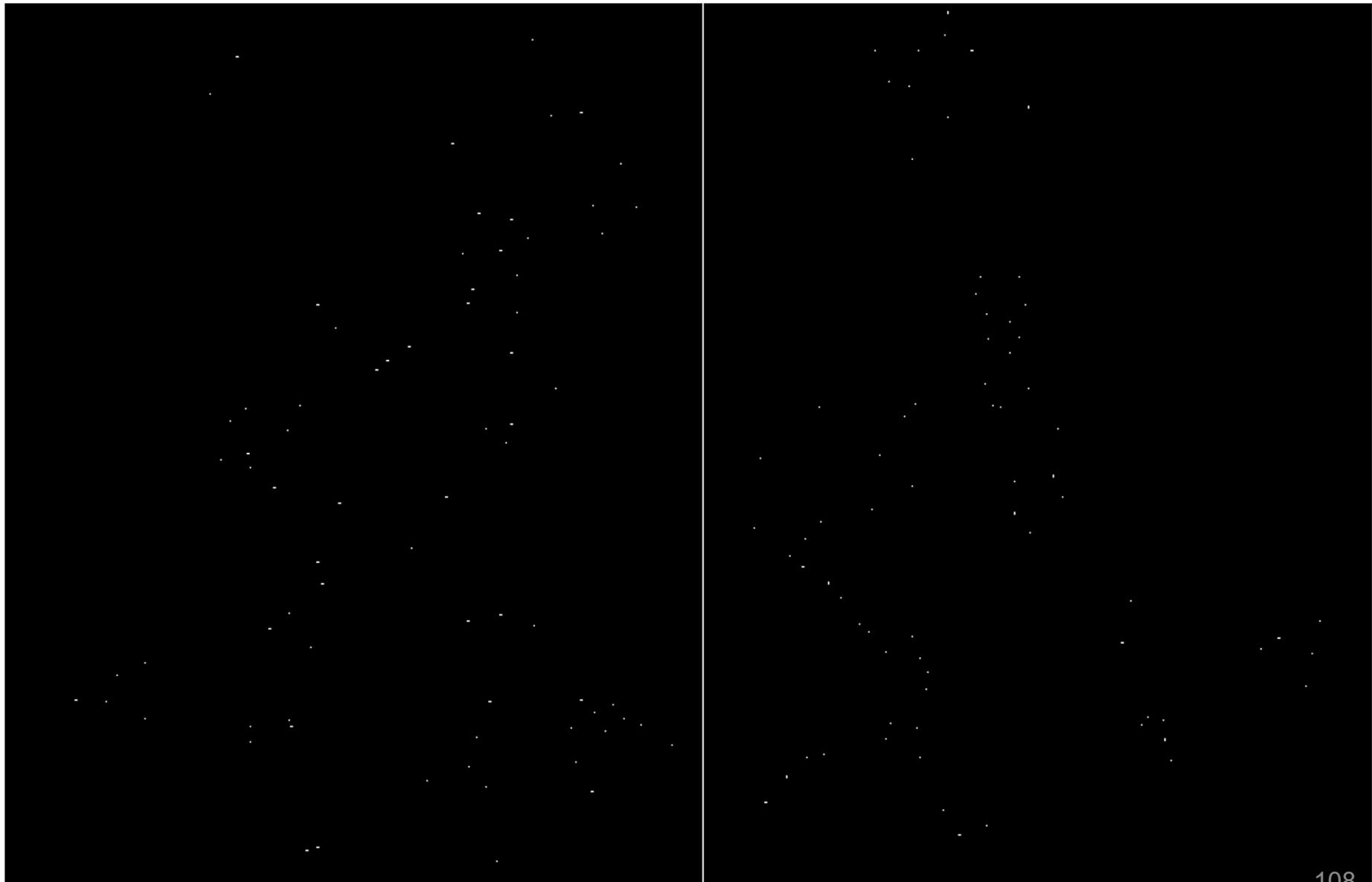
# Harris Detector: Steps

Find points with large corner response:  $R > \text{threshold}$



# Harris Detector: Steps

Take only the points of local maxima of  $R$



108

# Harris Detector: Steps



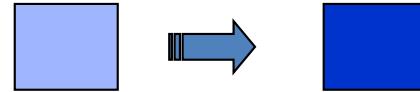


# Invariance and covariance

- We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance:** image is transformed and corner locations do not change
  - **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

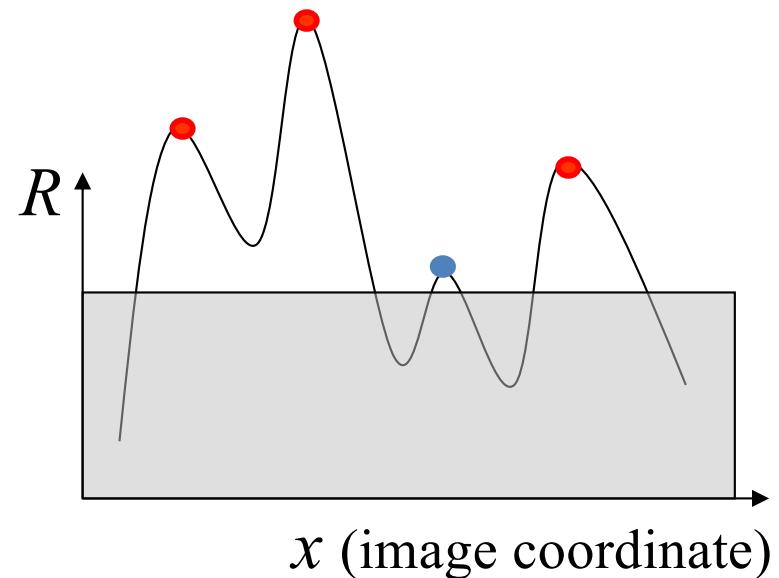
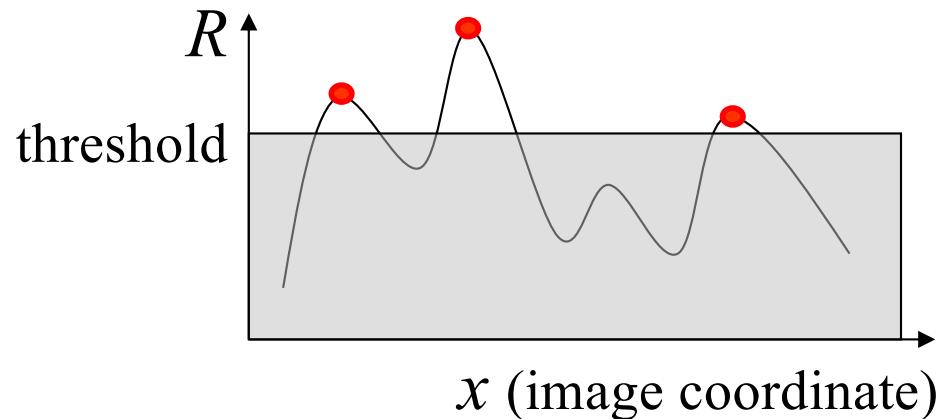


# Affine intensity change



$$I \rightarrow a I + b$$

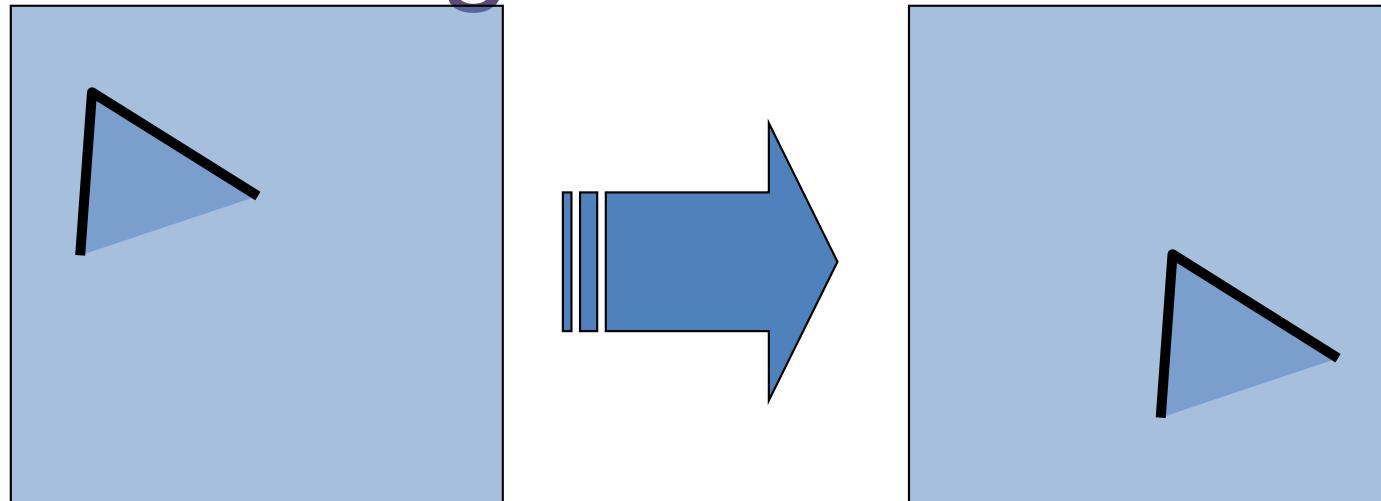
- Only derivatives are used => invariance to intensity shift  $I \rightarrow I + b$
- Intensity scaling:  $I \rightarrow a I$



*Partially invariant to affine intensity change*



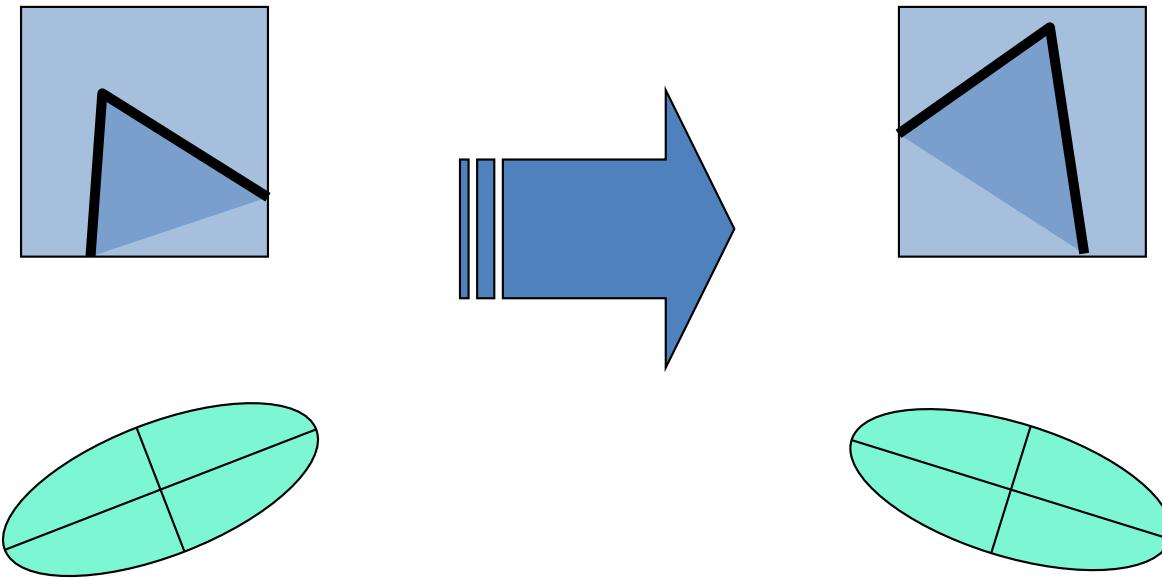
# Image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

# Image rotation

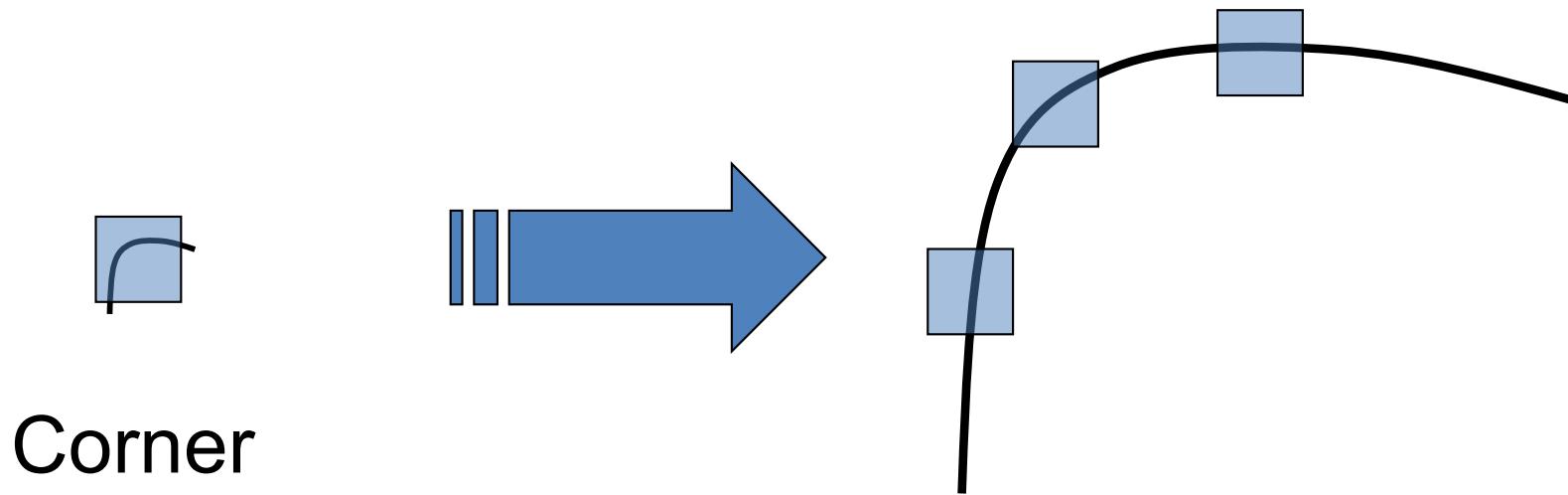


Second moment ellipse rotates but its shape  
(i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation



# Scaling



All points will  
be classified  
as **edges**

Corner location is not covariant to scaling!



# Lab 7: Image Filtering and Corner Detection



(Left: original image. Right: image overlap with corners.)



# Lecture Outline

- Image and Video Basics
- Recap on Image Filtering
- Interest Point
- Seam Carving



# Seam Carving for Content-Aware Image Resizing



Seam carving



Conventional scaling

S. Avidan and A. Shamir. "[Seam Carving for Content-Aware Image Resizing](#)," *ACM Trans. Graph.*, 2007.



# Energy function



$$e_1(\mathbf{I}) = \left| \frac{\partial}{\partial x} \mathbf{I} \right| + \left| \frac{\partial}{\partial y} \mathbf{I} \right|$$

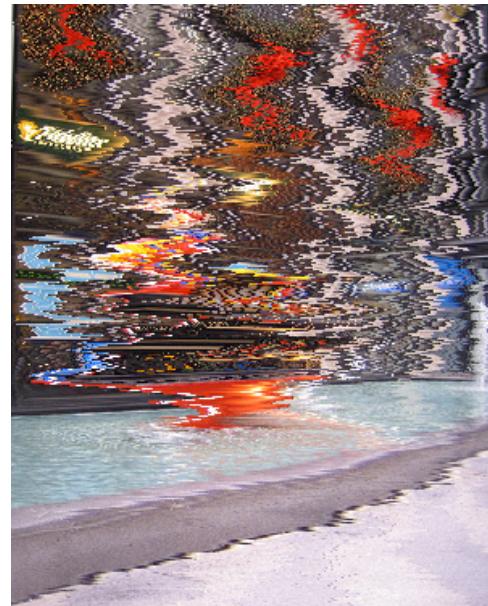
Energy for pixel I



# Energy-based pixel removal



Crop



Remove the lowest-energy pixels per row



Remove the lowest-energy columns



Remove the lowest-energy seams



# Seam carving

- Define seams

$$\mathbf{s}^x = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \text{ s.t. } \forall i, |x(i) - x(i-1)| \leq 1$$

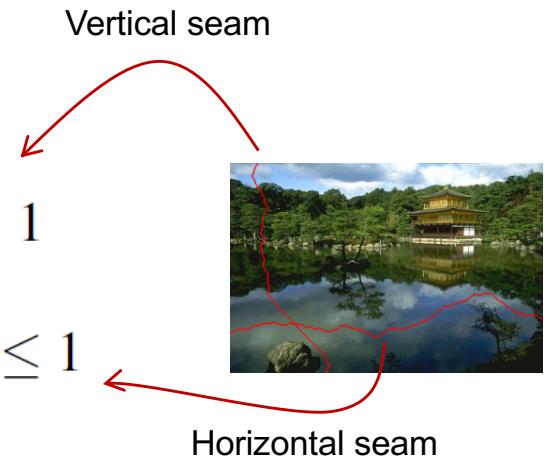
$$\mathbf{s}^y = \{s_j^y\}_{j=1}^m = \{(j, y(j))\}_{j=1}^m, \text{ s.t. } \forall j, |y(j) - y(j-1)| \leq 1$$

- Find optimal seam

$$s^* = \min_{\mathbf{s}} E(\mathbf{s}) = \min_{\mathbf{s}} \sum_{i=1}^n e(\mathbf{I}(s_i))$$

- through dynamic programming

$$M(i, j) = e(i, j) + \min(M(i-1, j-1), M(i-1, j), M(i-1, j+1))$$



# Extension

- Seam insertion



- Object removal

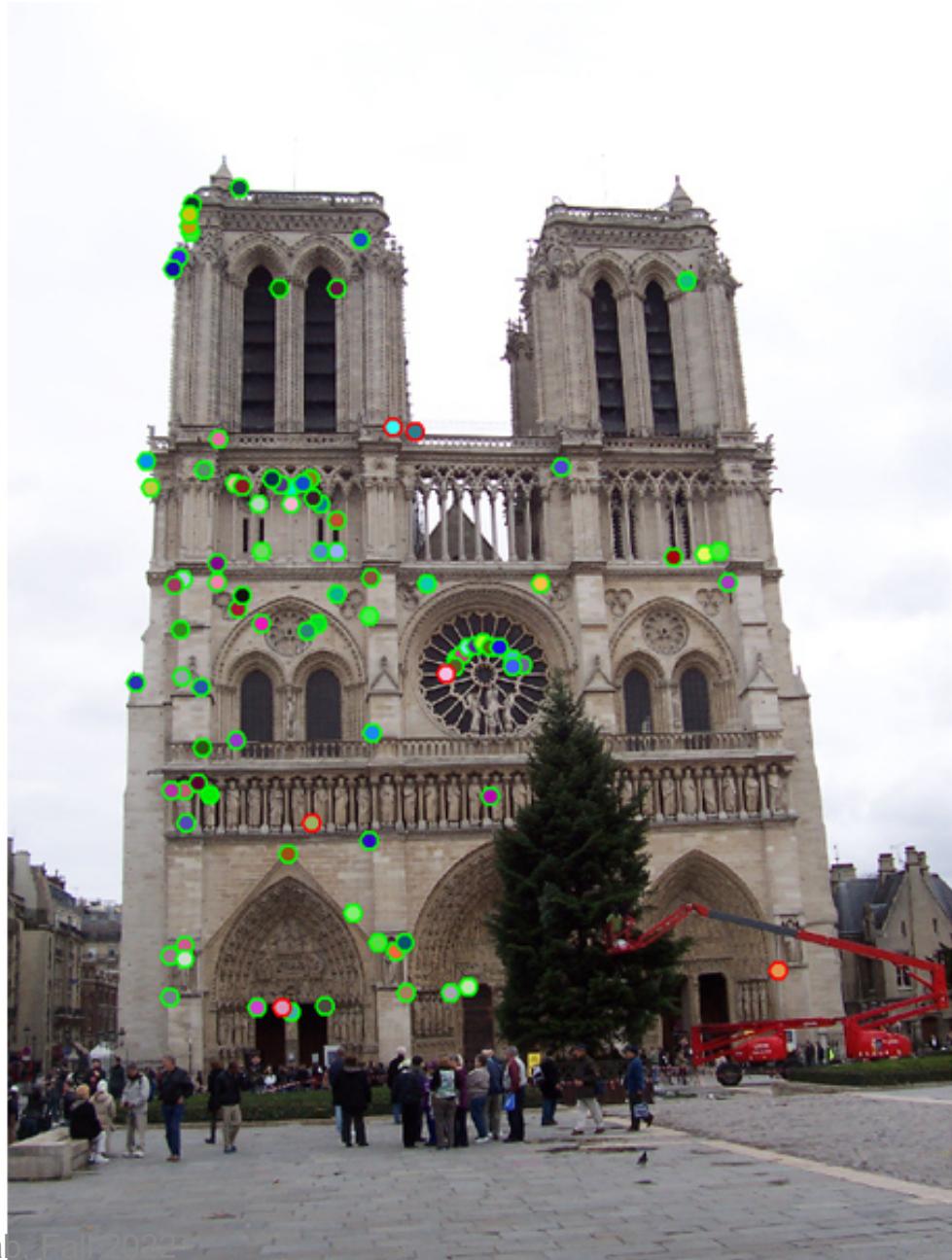
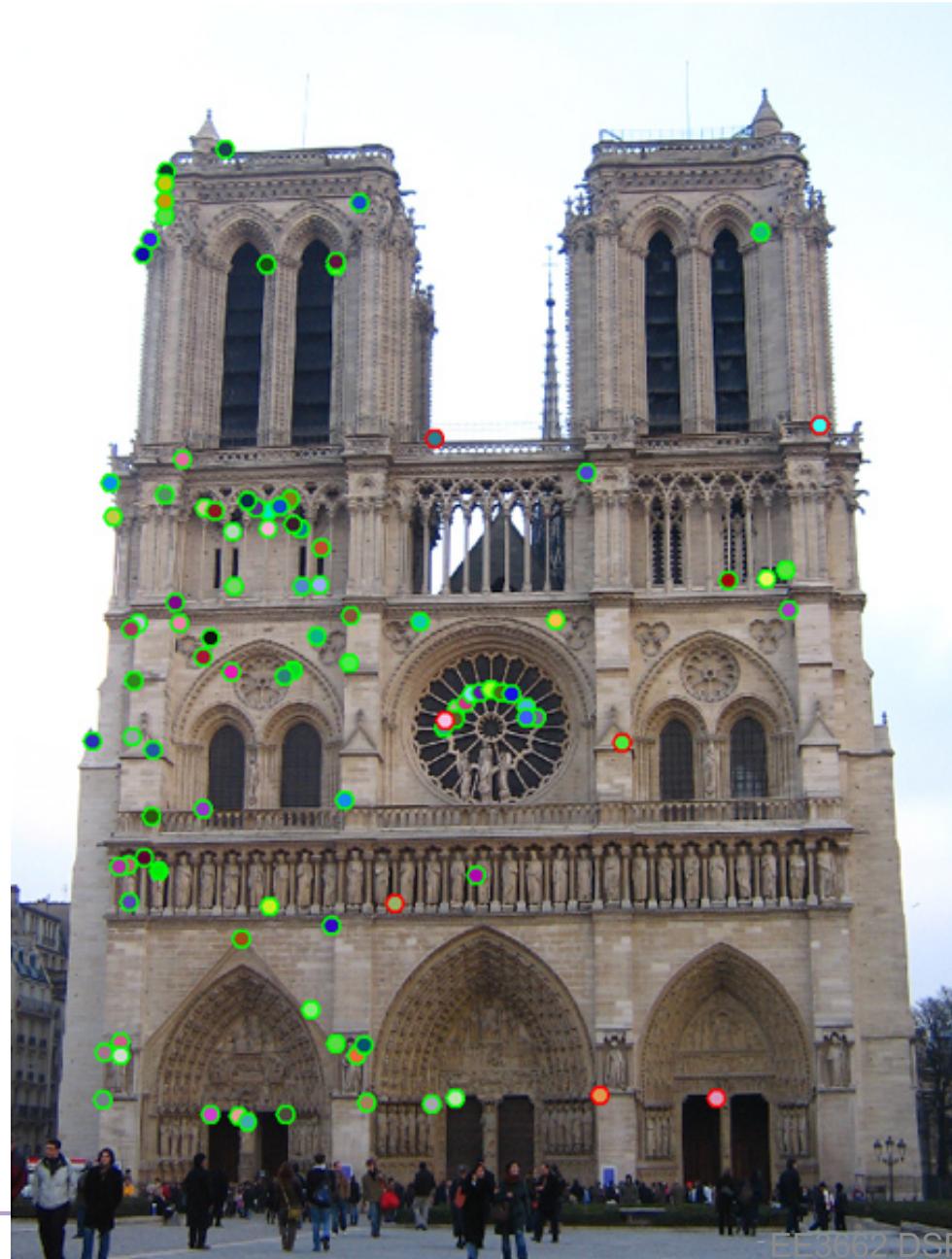




# Appendix

- Image and Video Basics
- Recap on Image Filtering
- Interest Point
- Seam Carving
- Image Matching/Alignment
  - Transformation
  - RANSAC

# How do we fit the best alignment?



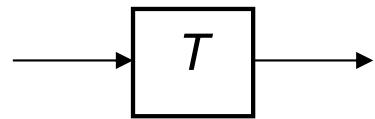


# Alignment

- Alignment: find parameters of model that maps one set of points to another
- Typically want to solve for a global transformation that accounts for **\*most\*** true correspondences
- Difficulties
  - Noise (typically 1-3 pixels)
  - Outliers (often 50%)
  - Many-to-one matches or multiple objects



# Parametric (global) warping



$$p = (x, y)$$

$$p' = (x', y')$$

Transformation  $T$  is a coordinate-changing machine:

$$p' = T(p)$$

What does it mean that  $T$  is global?

- Is the same for any point  $p$
- can be described by just a few numbers (parameters)

For linear transformations, we can represent  $T$  as a matrix

$$p' = \mathbf{T}p$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$



# Common transformations



original

## Transformed



translation



rotation



aspect



affine

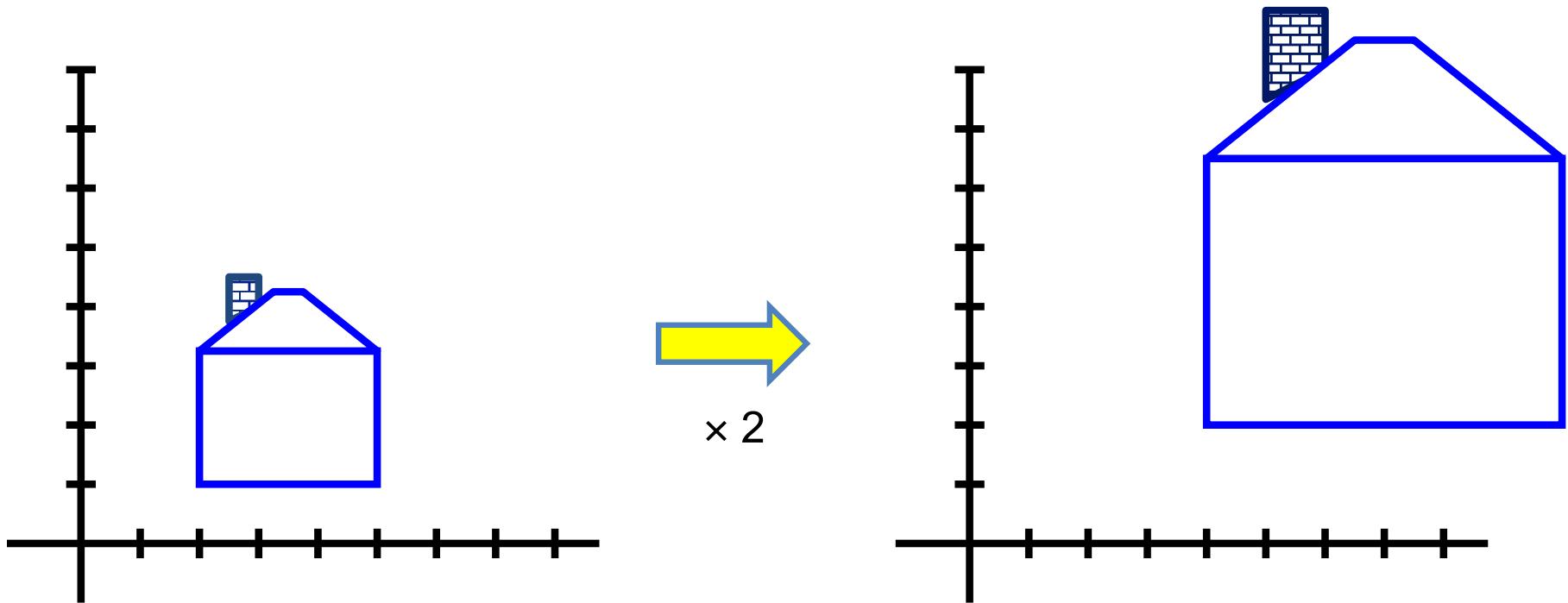


perspective



# Scaling

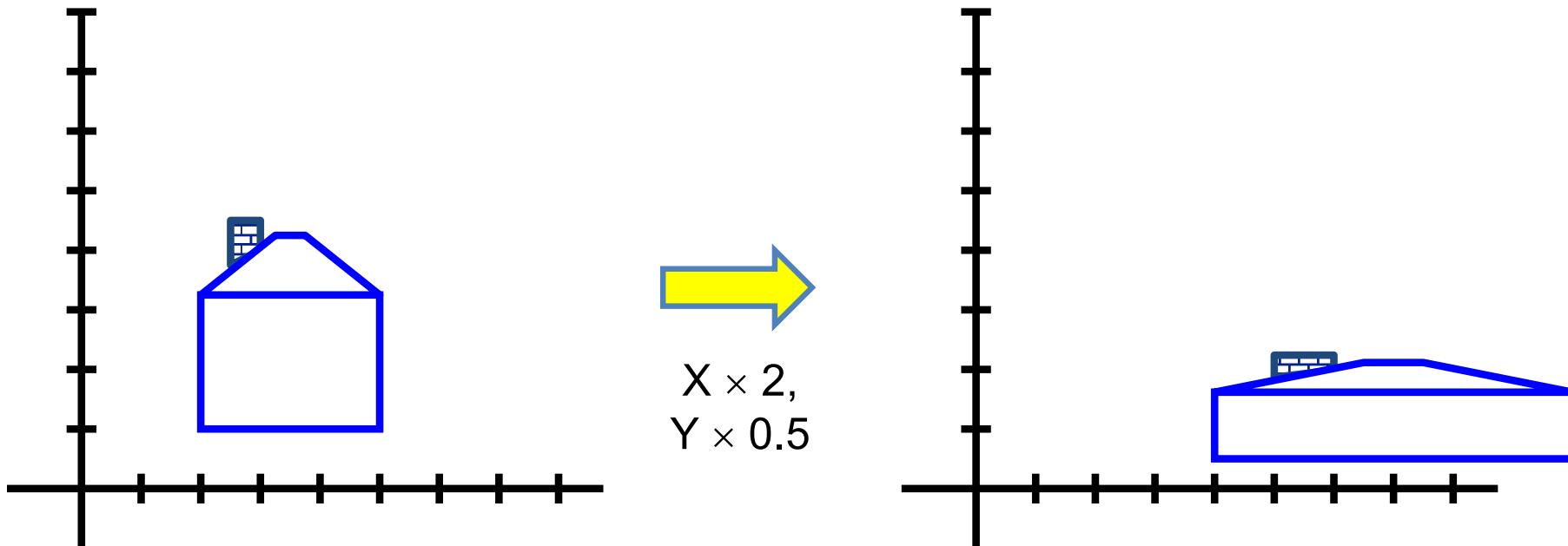
- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:





# Scaling

- *Non-uniform scaling*: different scalars per component:





# Scaling

- Scaling operation:  $x' = ax$

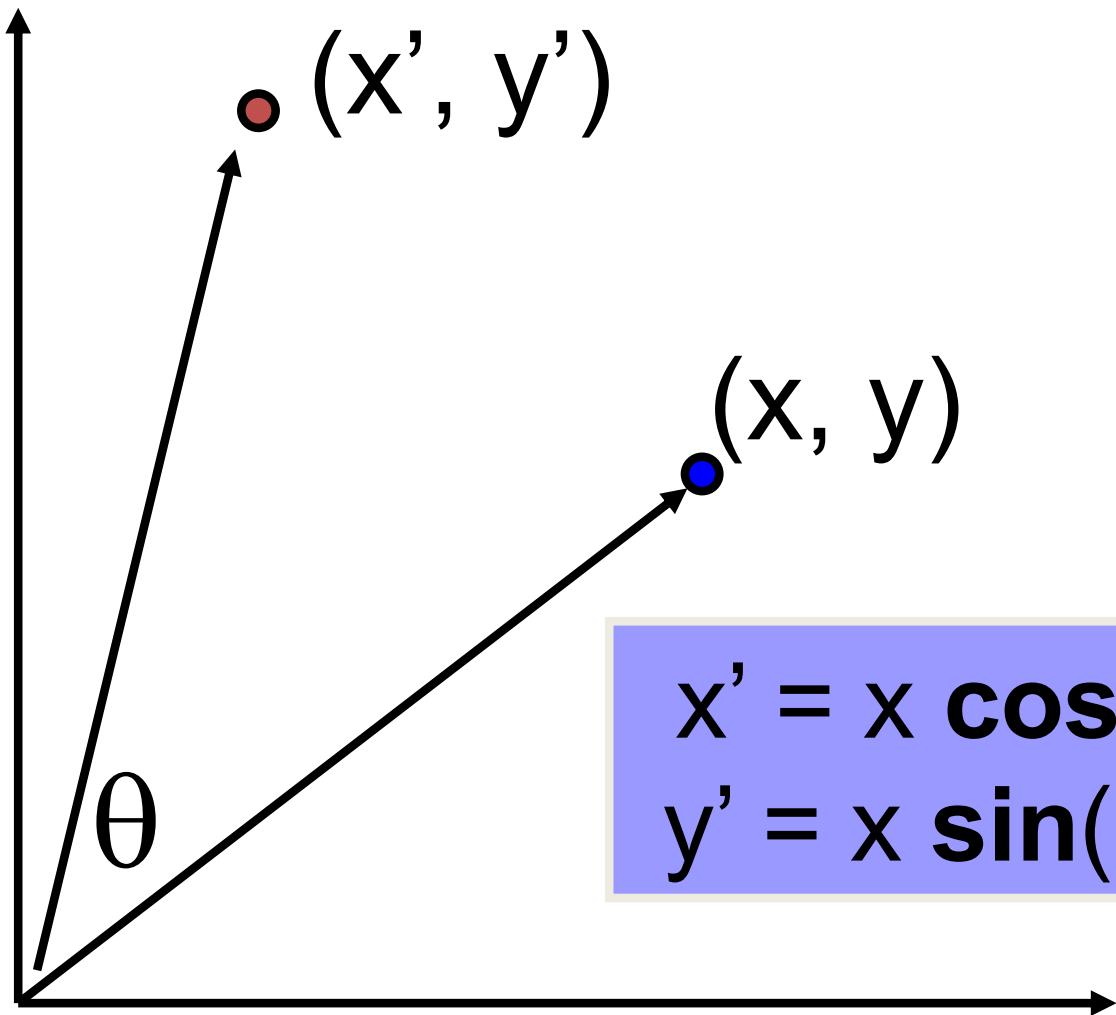
$$y' = by$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$



# 2-D Rotation



# 2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though  $\sin(\theta)$  and  $\cos(\theta)$  are nonlinear functions of  $\theta$ ,

- $x'$  is a linear combination of  $x$  and  $y$
- $y'$  is a linear combination of  $x$  and  $y$

What is the inverse transformation?

- Rotation by  $-\theta$
- For rotation matrices     $\mathbf{R}^{-1} = \mathbf{R}^T$



# Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shear



# Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



# Projective Transformations

Projective transformations are combos of

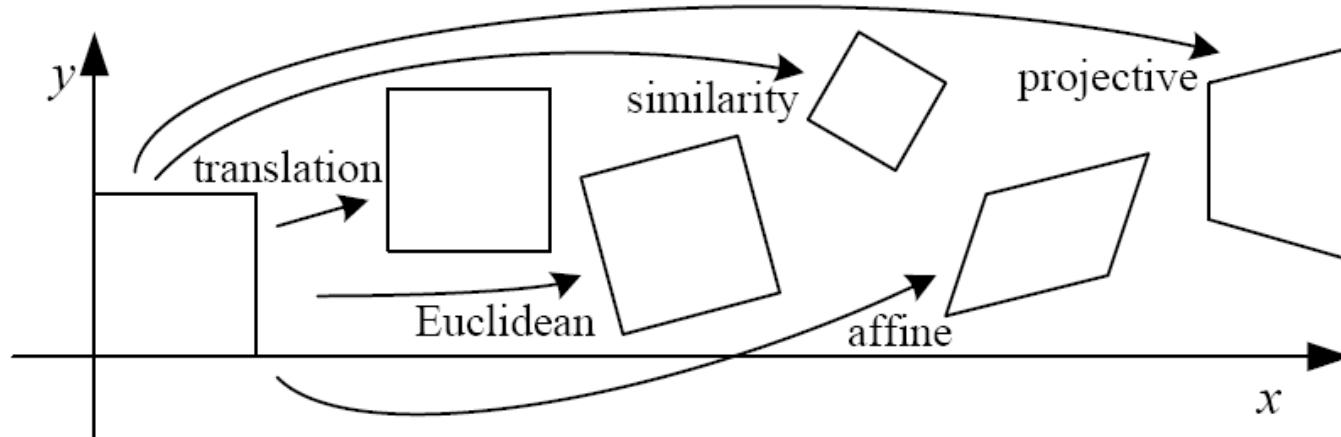
- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

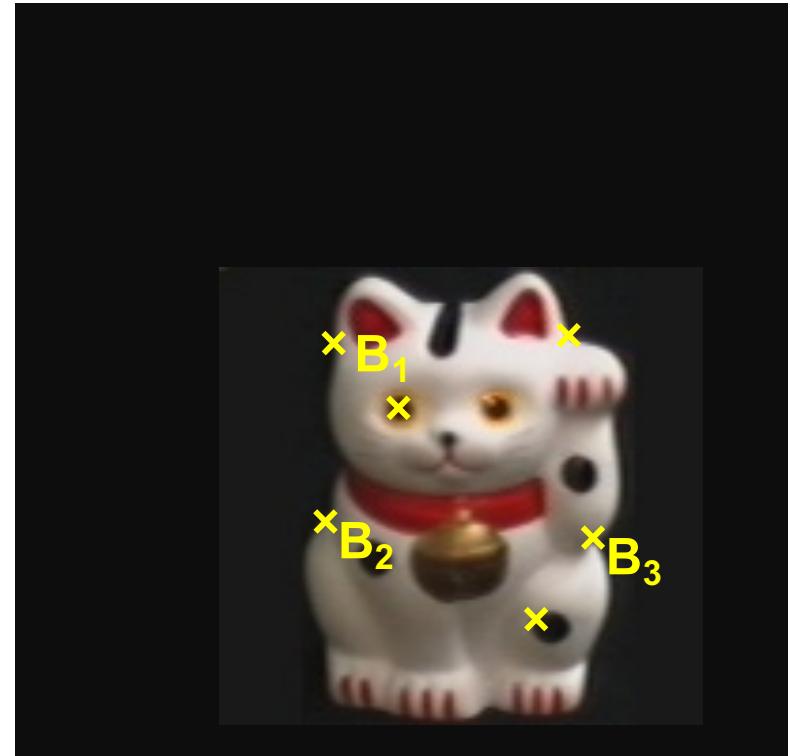
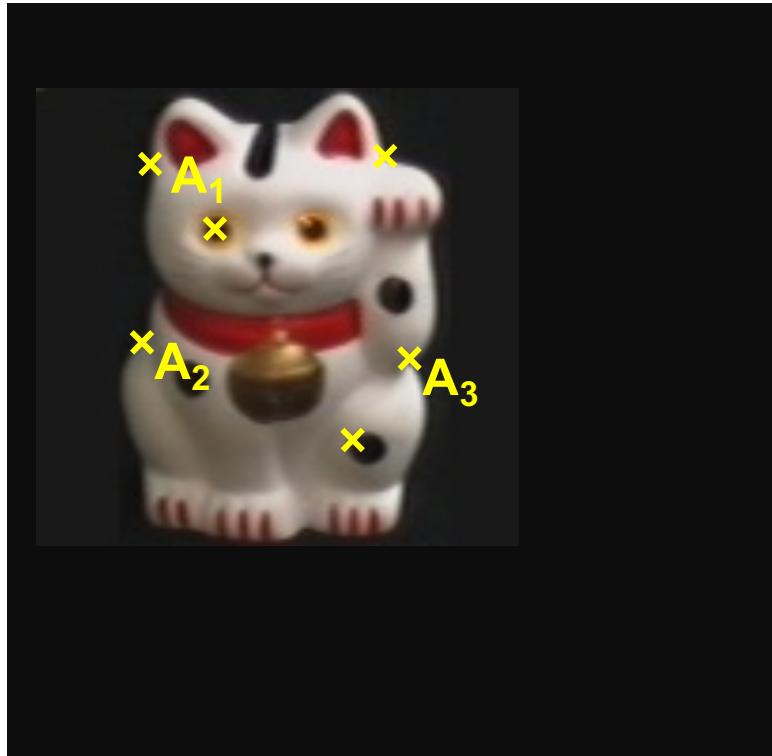
# 2D image transformations (reference table)



| Name              | Matrix   | # D.O.F. | Preserves:        | Icon |
|-------------------|--|----------|-------------------|------|
| translation       | $\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2\times 3}$  | 2        | orientation + ... |      |
| rigid (Euclidean) | $\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$  | 3        | lengths + ...     |      |
| similarity        | $\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$ | 4        | angles + ...      |      |
| affine            | $\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2\times 3}$               | 6        | parallelism + ... |      |
| projective        | $\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3\times 3}$       | 8        | straight lines    |      |



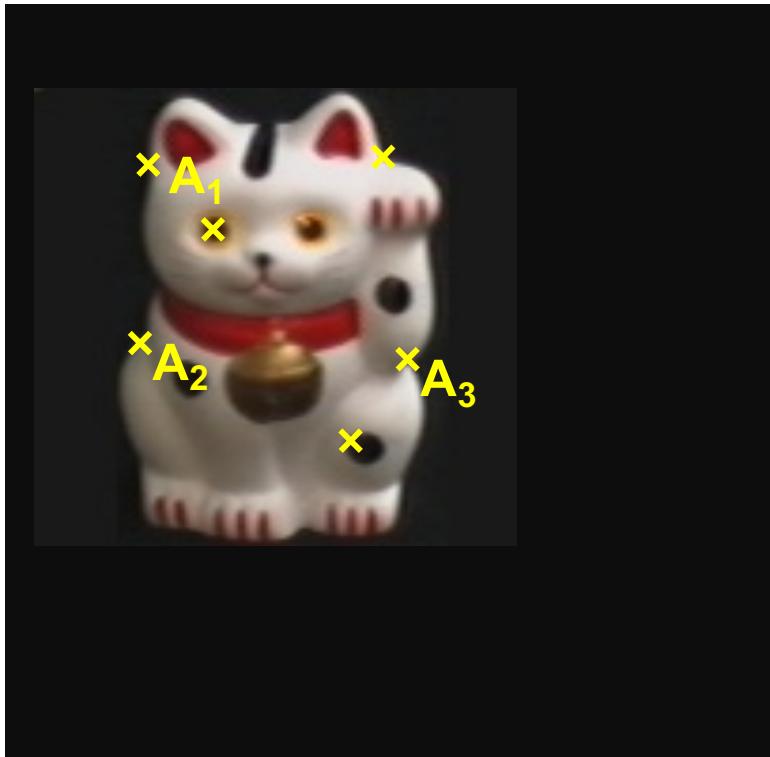
# Example: solving for translation



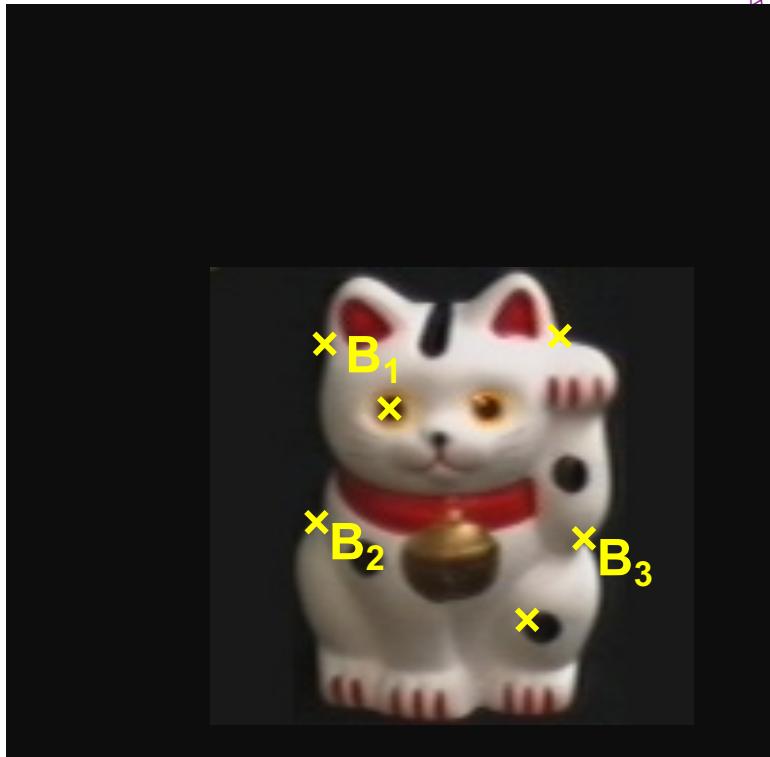
Given matched points in {A} and {B}, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

# Example: solving for translation



$(t_x, t_y)$



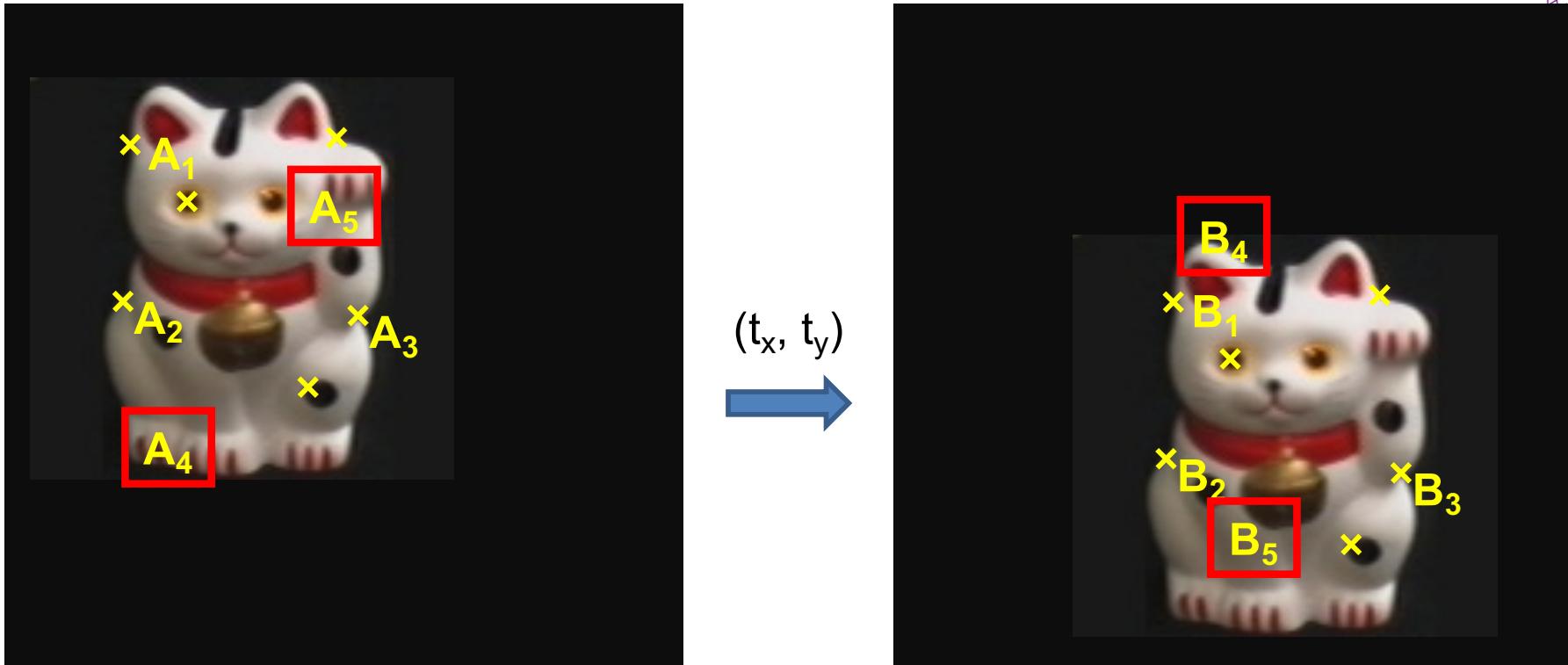
## Least squares solution

1. Write down objective function
2. Derived solution
  - a) Compute derivative
  - b) Compute solution
3. Computational solution
  - a) Write in form  $Ax=b$
  - b) Solve using pseudo-inverse or eigenvalue decomposition

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

# Example: solving for translation



Problem: outliers

## RANSAC solution

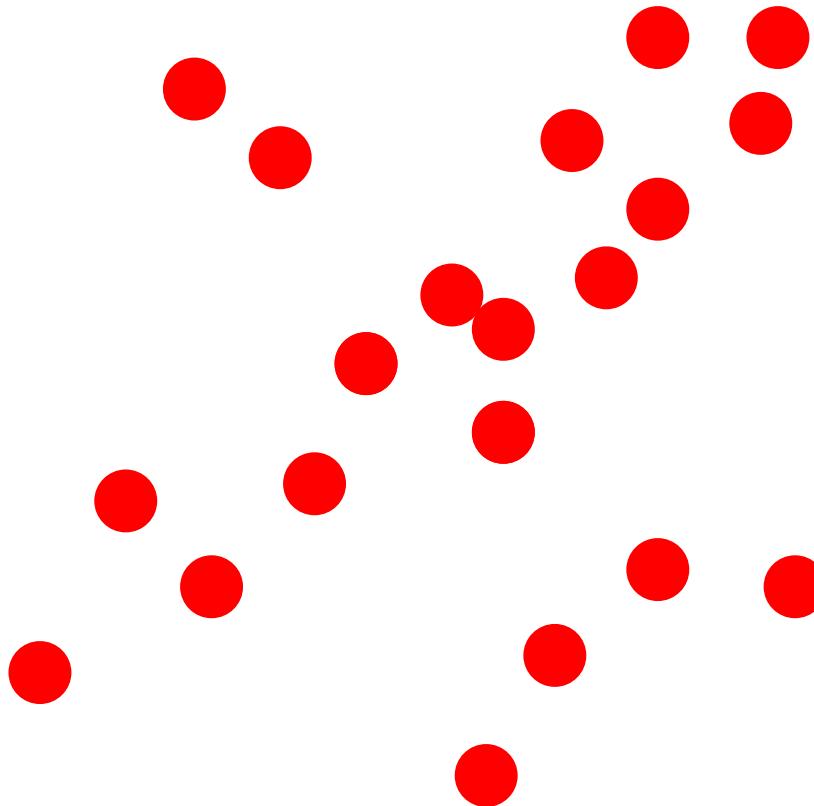
1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

# RANSAC

(RANdom SAmples Consensus) :

Fischler & Bolles in '81.



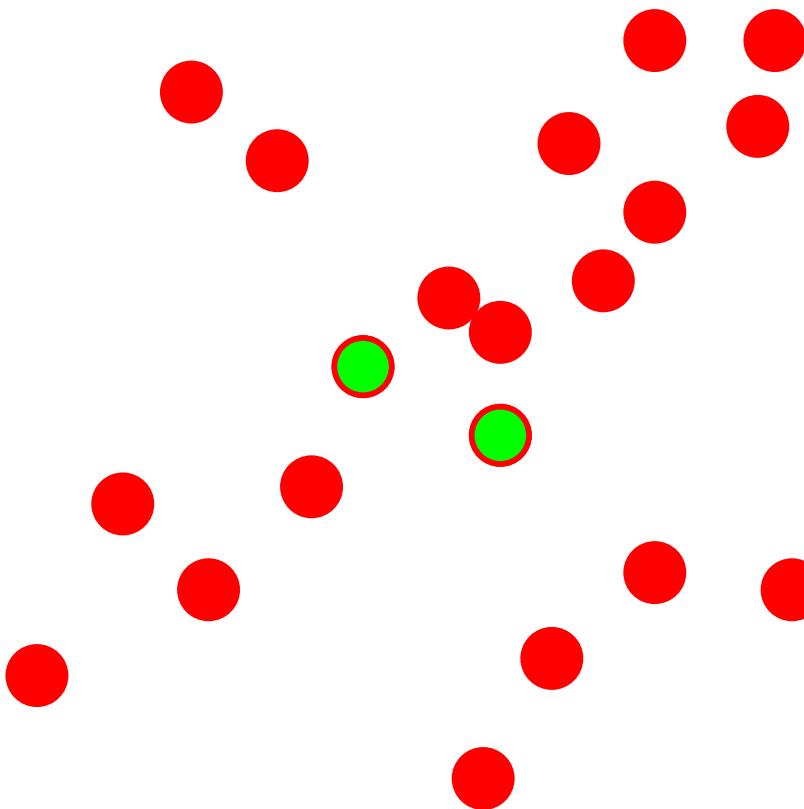
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

**Repeat 1-3 until the best model is found with high confidence**

# RANSAC

Line fitting example



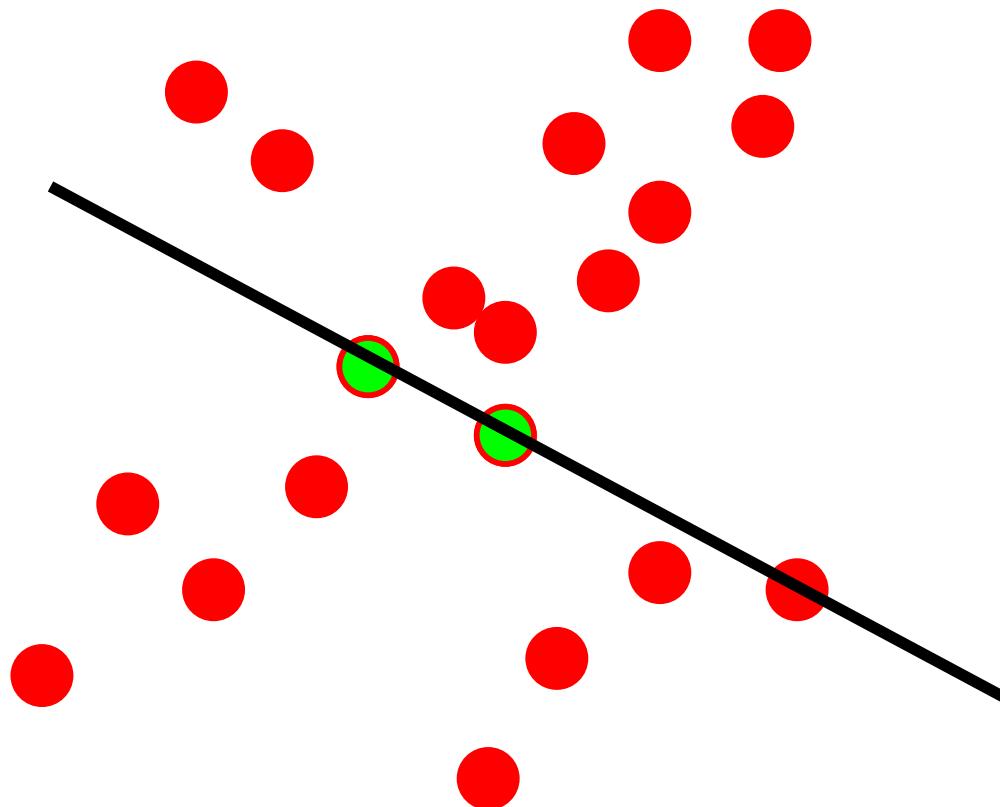
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

# RANSAC

Line fitting example



Algorithm:

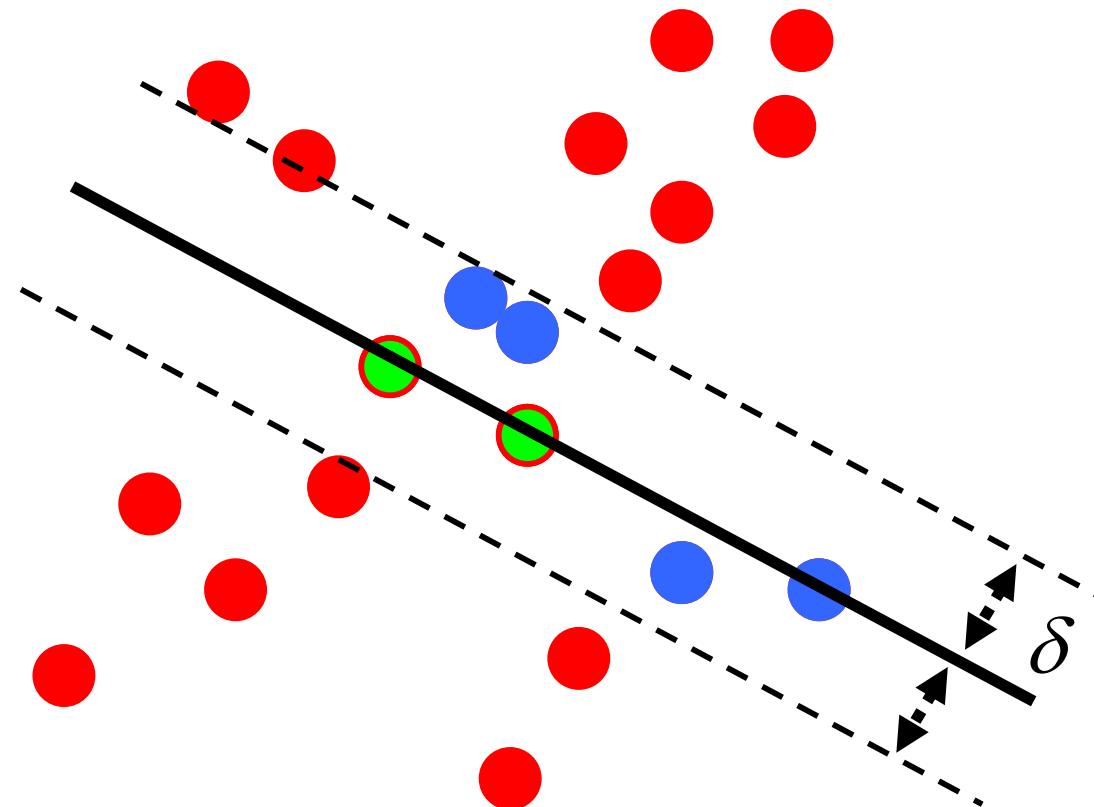
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

# RANSAC

Line fitting example

$$N_I = 6$$

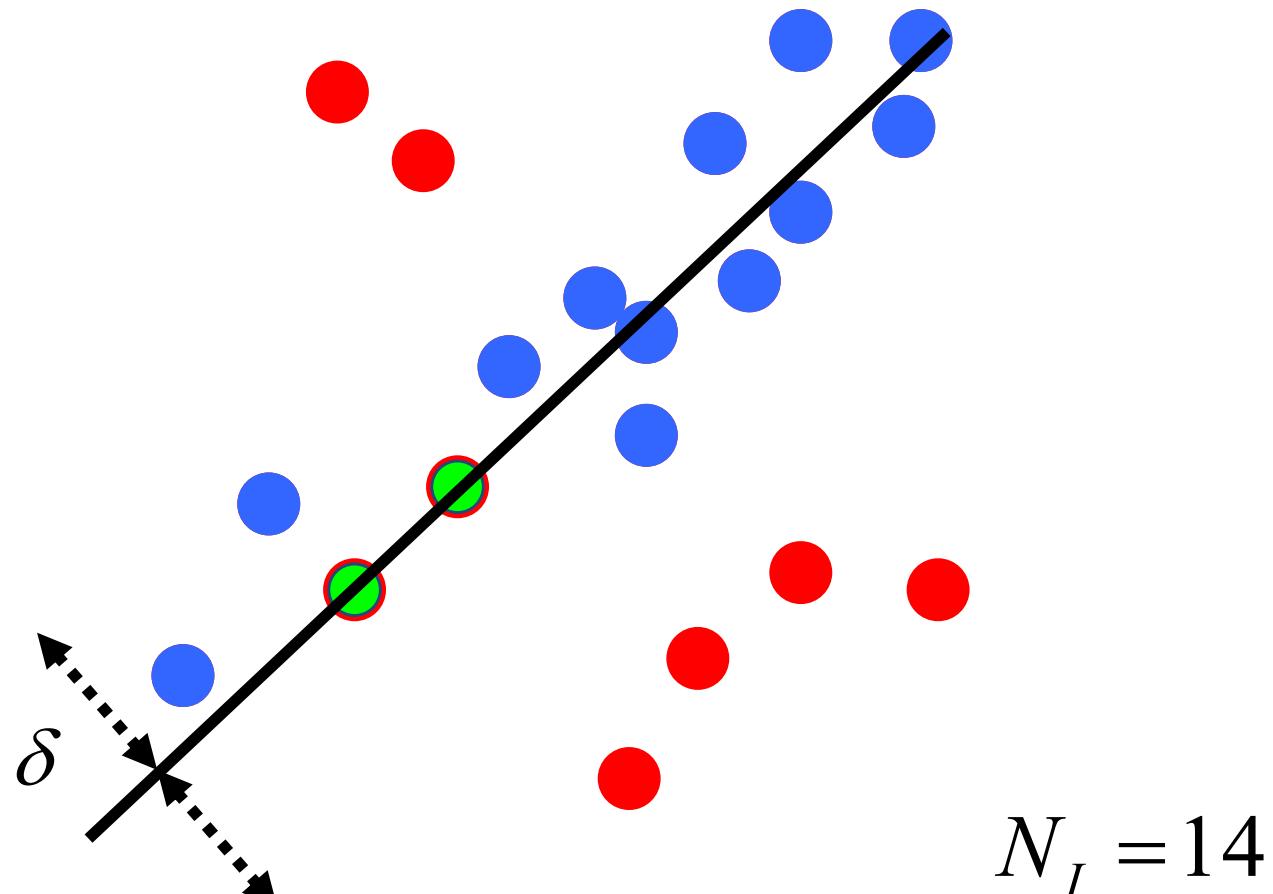


Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

# RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

**Repeat 1-3 until the best model is found with high confidence**



# How to choose parameters?

- Number of samples  $N$ 
  - Choose  $N$  so that, with probability  $p$ , at least one random sample is free from outliers (e.g.  $p=0.99$ ) (outlier ratio:  $e$ )
- Number of sampled points  $s$ 
  - Minimum number needed to fit the model
- Distance threshold  $\delta$ 
  - Choose  $\delta$  so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev.  $\sigma$ :  $t^2=3.84\sigma^2$

$$N = \log(1-p)/\log(1-(1-e)^s)$$

| $s$ | proportion of outliers $e$ |     |     |     |     |     |      |  |
|-----|----------------------------|-----|-----|-----|-----|-----|------|--|
|     | 5%                         | 10% | 20% | 25% | 30% | 40% | 50%  |  |
| 2   | 2                          | 3   | 5   | 6   | 7   | 11  | 17   |  |
| 3   | 3                          | 4   | 7   | 9   | 11  | 19  | 35   |  |
| 4   | 3                          | 5   | 9   | 13  | 17  | 34  | 72   |  |
| 5   | 4                          | 6   | 12  | 17  | 26  | 57  | 146  |  |
| 6   | 4                          | 7   | 16  | 24  | 37  | 97  | 293  |  |
| 7   | 4                          | 8   | 20  | 33  | 54  | 163 | 588  |  |
| 8   | 5                          | 9   | 26  | 44  | 78  | 272 | 1177 |  |



# RANSAC conclusions

## Good

- Robust to outliers
- Applicable for larger number of objective function parameters than Hough transform
- Optimization parameters are easier to choose than Hough transform

## Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not good for getting multiple fits

## Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)



# Optional hw: Panorama Stitching

