EE3662 DSP Lab
Unit II – Visual Signal

Instructor: 黃朝宗、孫民

TA: 汪立偉、丁友鈞、鄭安佑、鄭敬儒

National Tsing Hua University

Department of Electrical Engineering

Syllabus

- 1st Week: 3-hr lecture + take-home lab (Lab 5)
- 2nd Week: 3-hr lab on hybrid image (Lab 6)
- 3rd Week: 3-hr lab on corner detection (Lab 7)
- 4th Week: 3-hr lab on seam carving (Lab 8)

* MATLAB is used for all labs

Lecture Outline

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

Lab Map

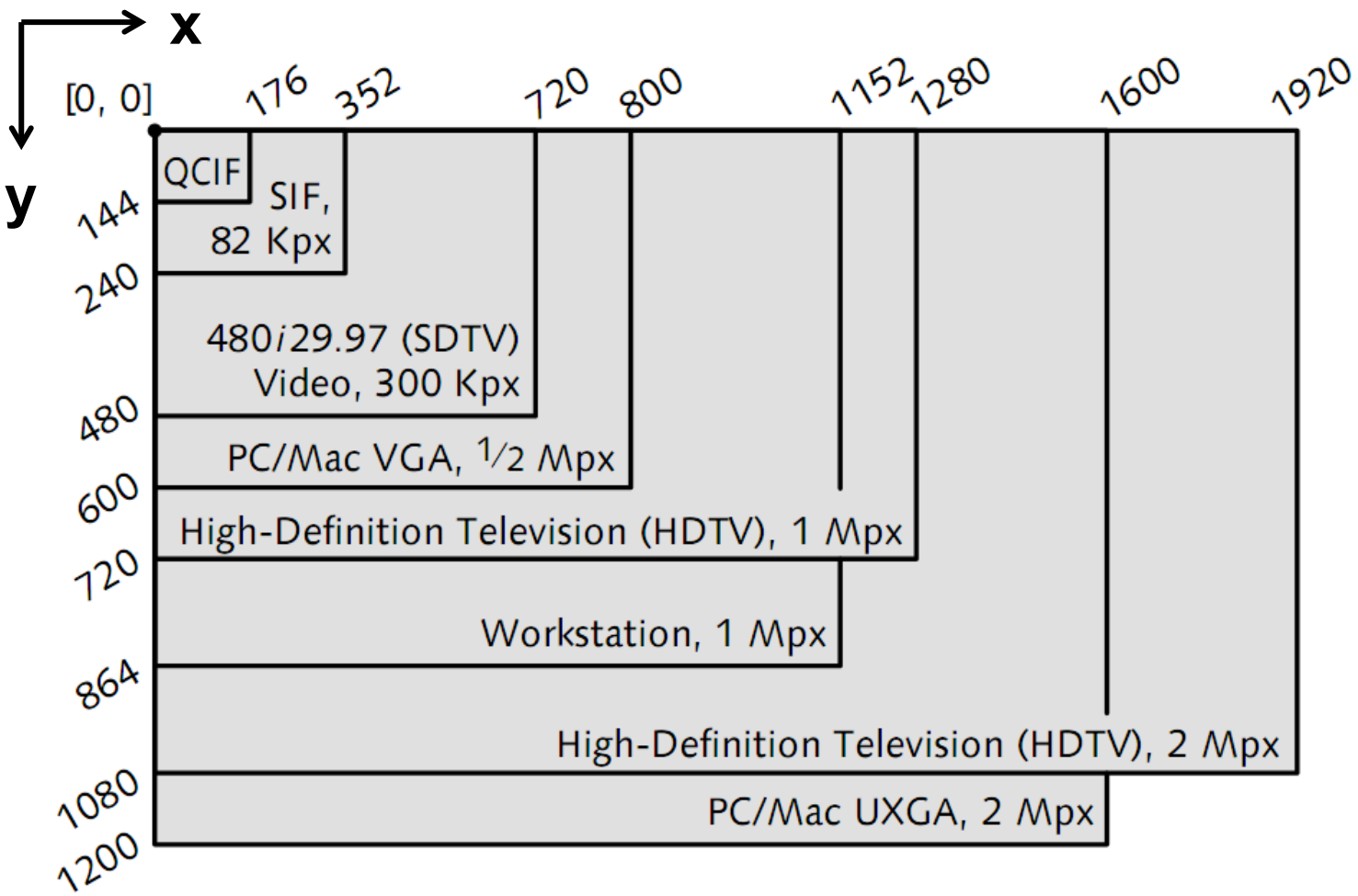
Pixel array manipulation	(Lab 5)
Image filtering	(Lab 6)
Image feature	(Lab 7)
Image Manipulation	(Lab 8)

Basics Representation

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

Pixel Array

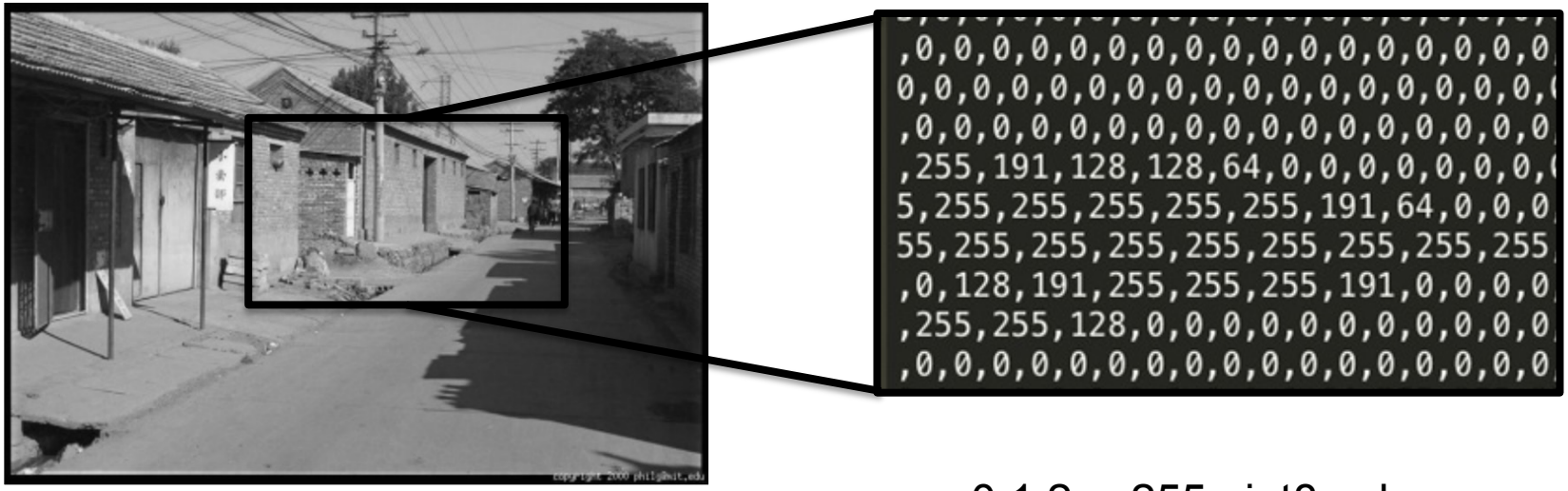
- Picture Element, Pel or Pixel, in (x,y)



Ref: *Digital Video and HDTV*

Pixel Value

Grayscale (灰階) image



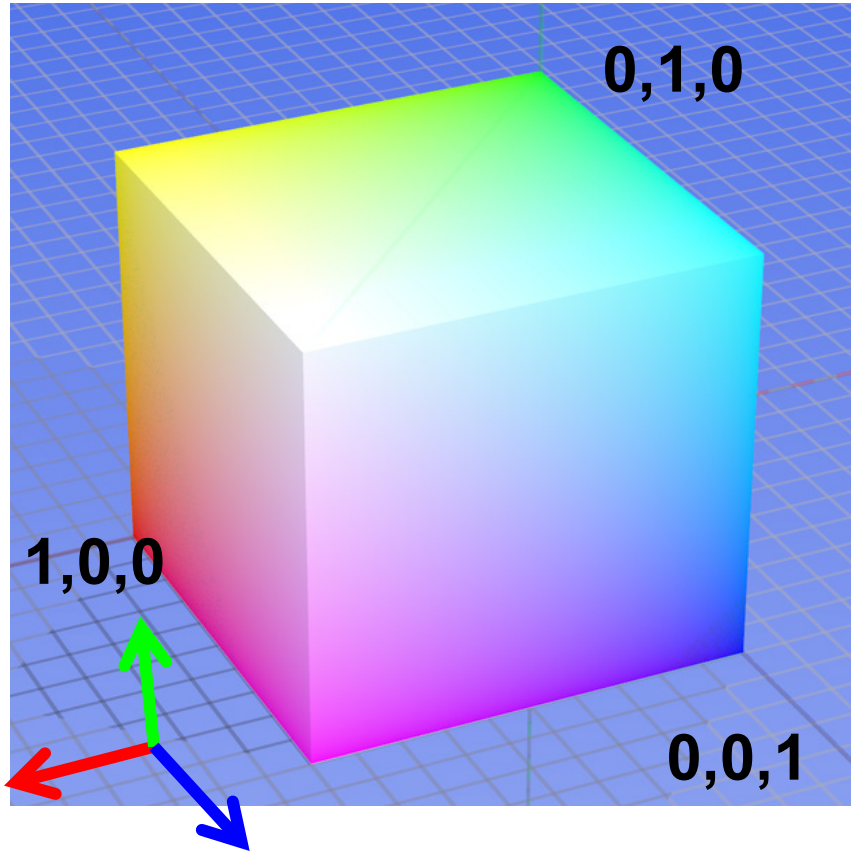
0,1,2,…255 uint8 values

RGB color space

Color image

Color spaces: RGB

Default color space



R
(G=0,B=0)

G
(R=0,B=0)

B
(R=0,G=0)

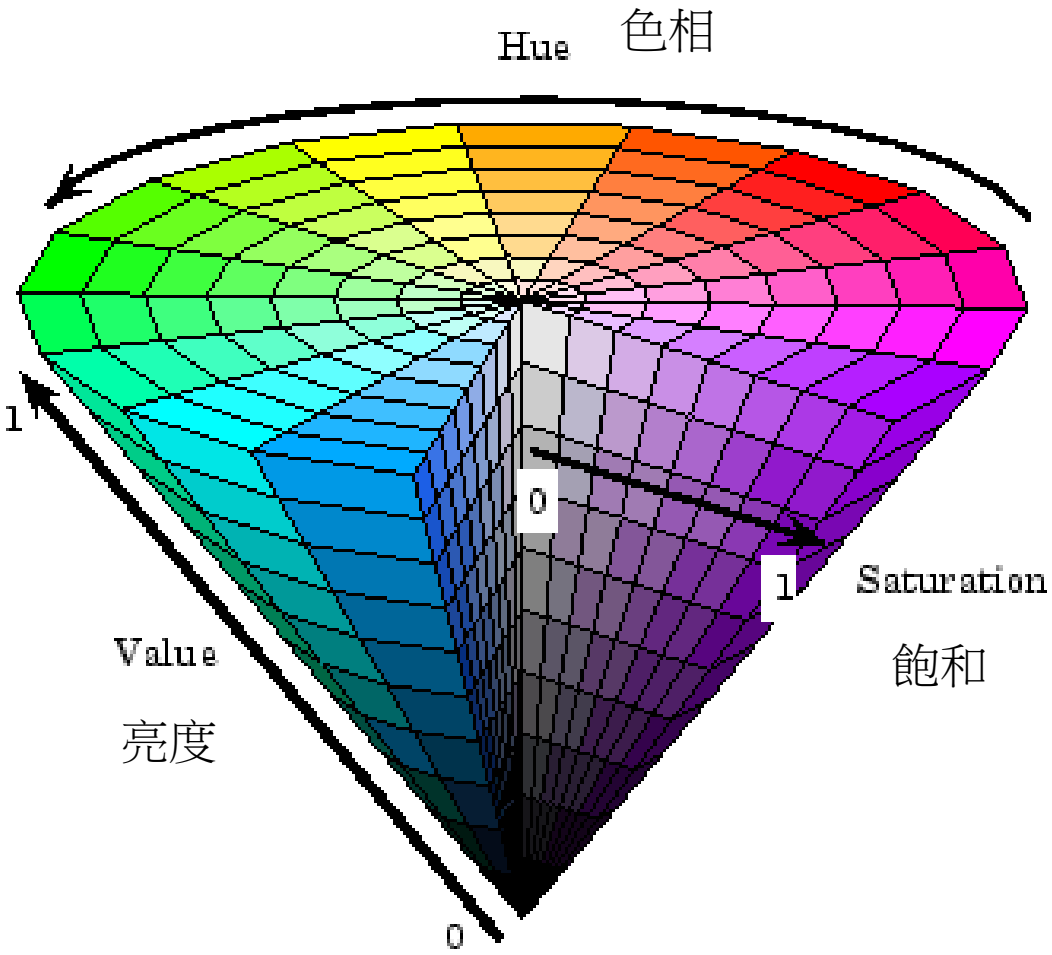
Some drawbacks

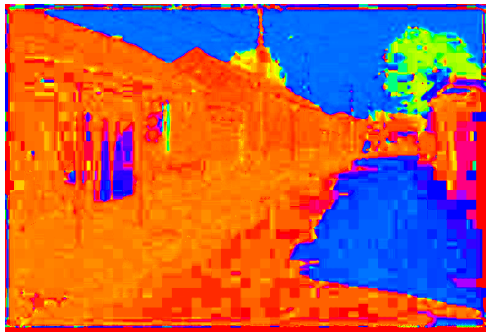
- Strongly correlated channels
- Non-perceptual

Image from: http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png

Color spaces: HSV

Intuitive color space





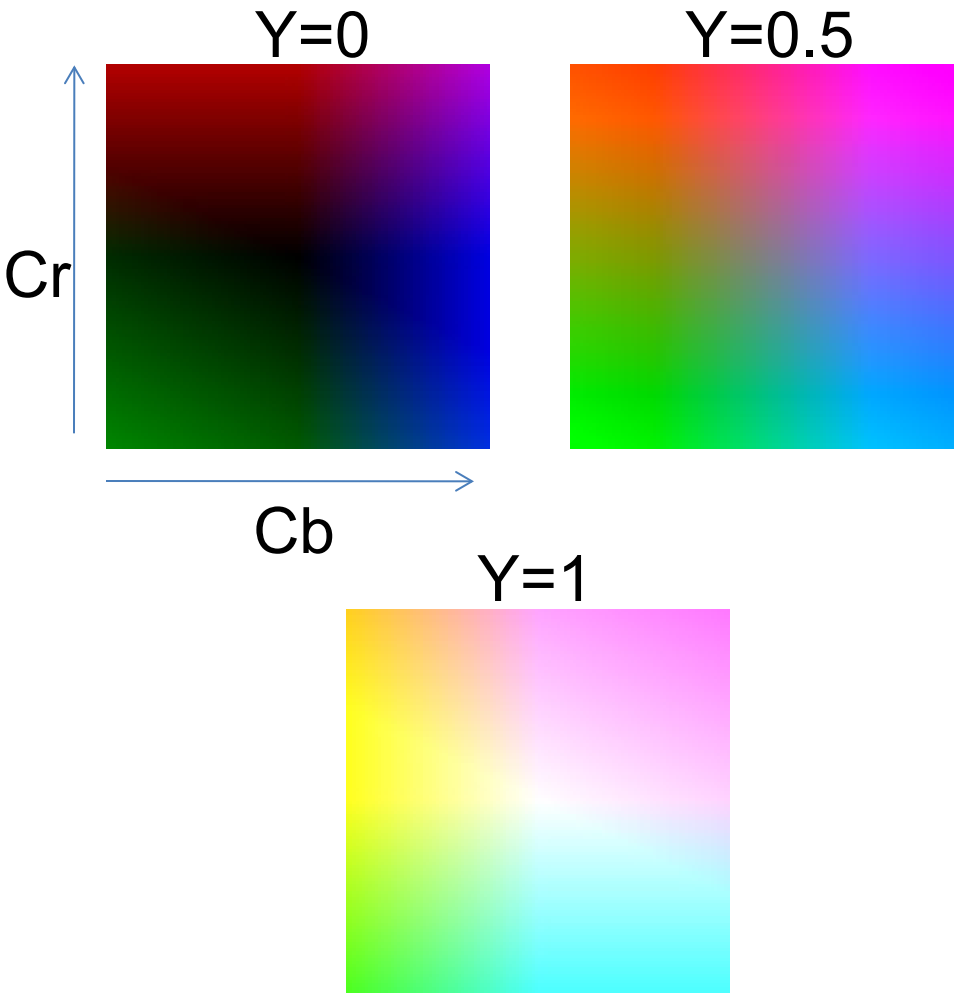
H
(S=1,V=1)

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)

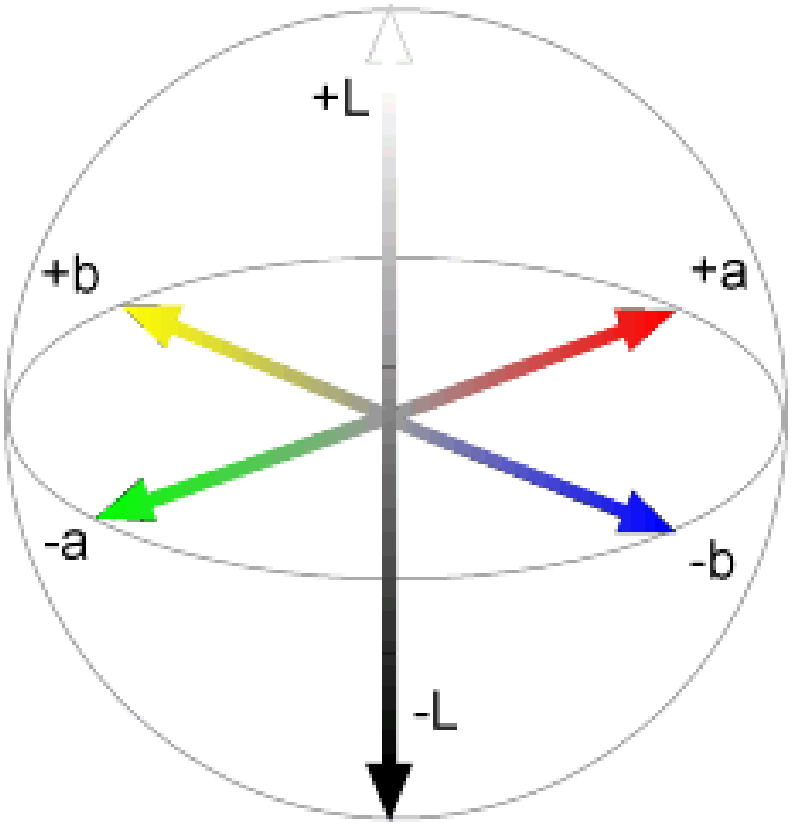
Cb
(Y=0.5,Cr=0.5)

Cr
(Y=0.5,Cb=05)

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

Color spaces: L*a*b*

"Perceptually uniform"* color space

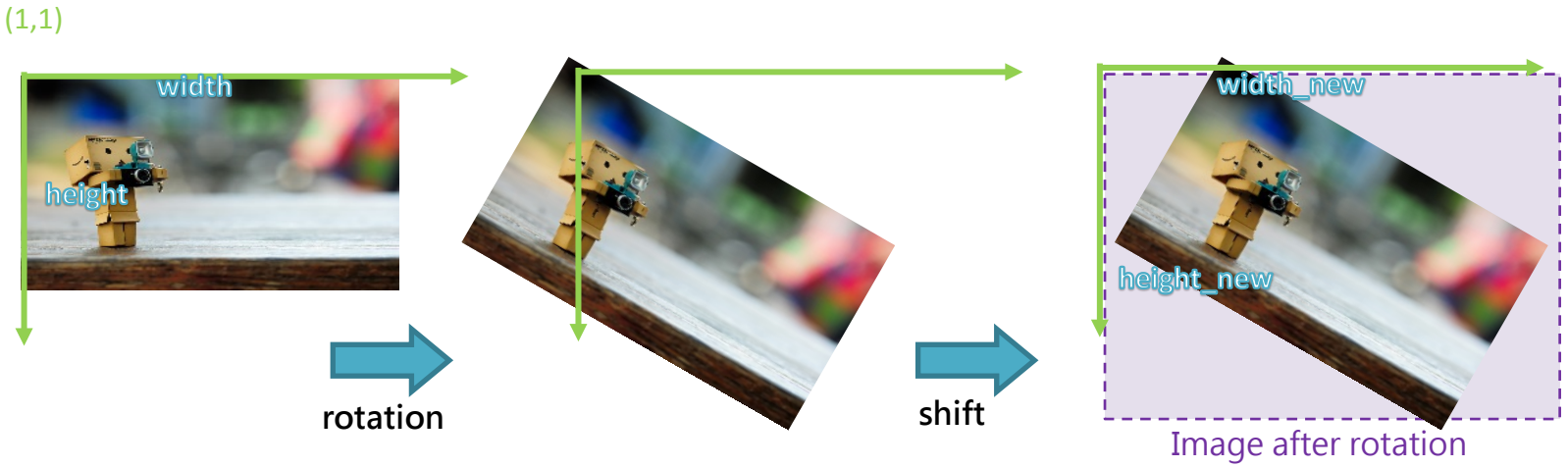


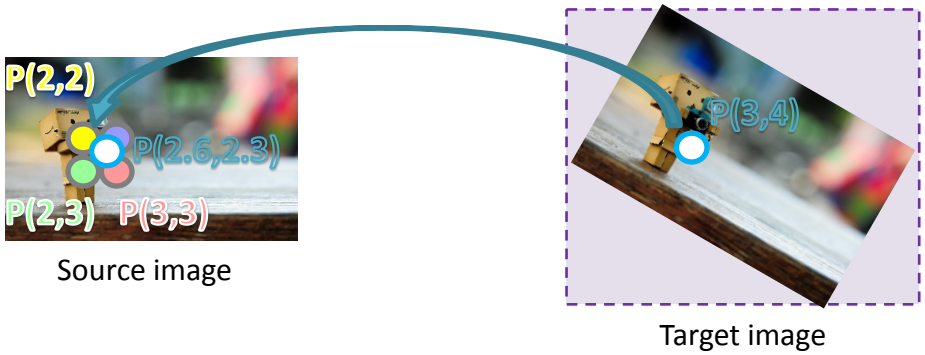
L
(a=0,b=0)

a
(L=65,b=0)

b
(L=65,a=0)

Example: Rotation (Lab 5)

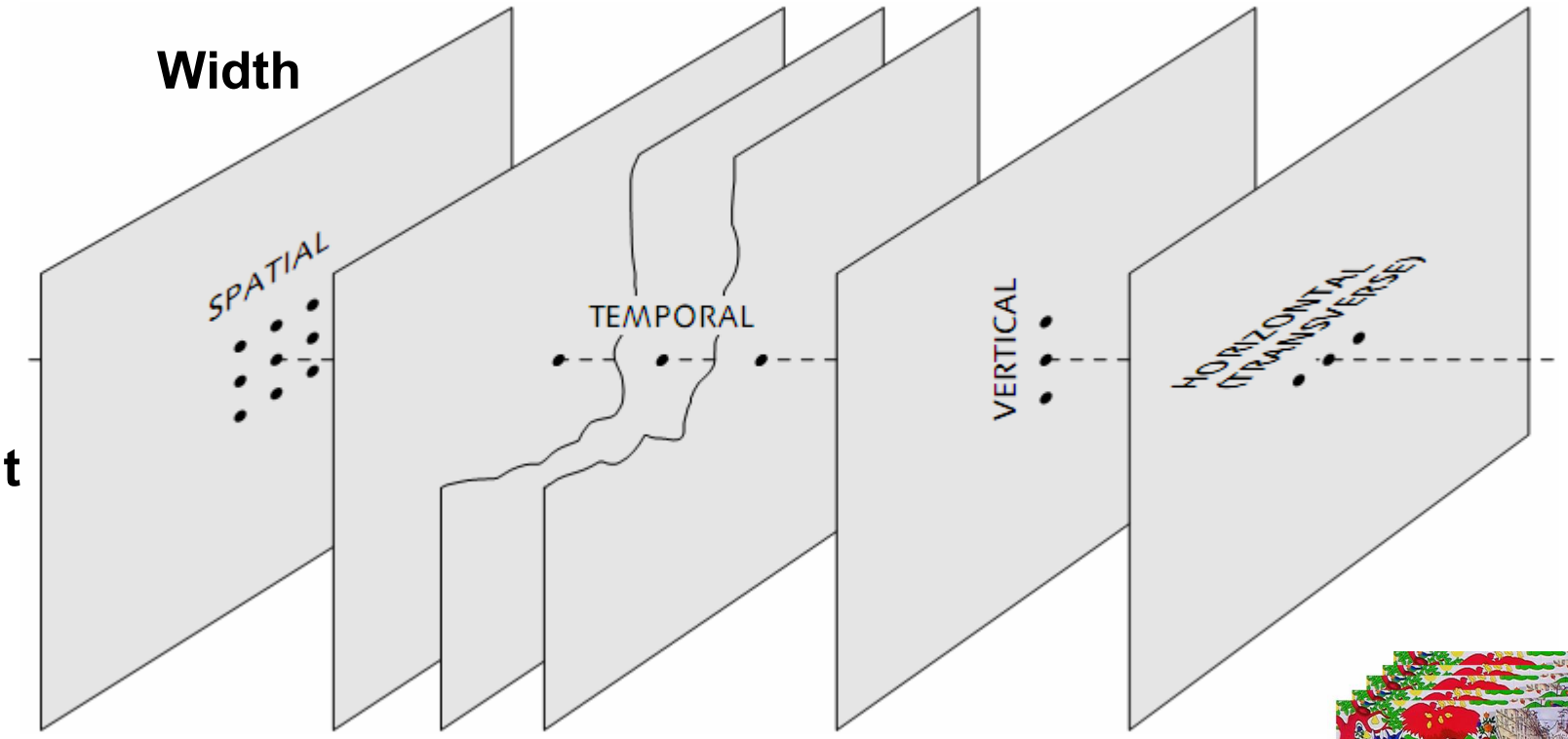




e.g.
Backward warping +
bilinear interpolation

Video = Sequence of Images

3-D index: (x, y, t)



Ref: *Digital Video and HDTV*

Refresh Rate v.s. Frame Rate

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

Up Conversion

Ref: *Digital Video and HDTV*

At least high enough to avoid flicker.
Higher for fast-moving objects.

Ex. for 120Hz LCD TV:
Encoded Video: 30Hz (frame per second, fps)
HDMI 1080p : 60Hz (progressive)
Refresh : 120Hz

Resolution Explosion

85” 8Kx4K (sharp)

monitor

(Eizo, ViewSonic, IBM, Sharp, …)

20”~36” 4Kx2K

84” 4Kx2K (LG)

3.5” 320p
(Apple)

2009

3.5” 640p
(Apple)

2010

4” 640p
(Apple)

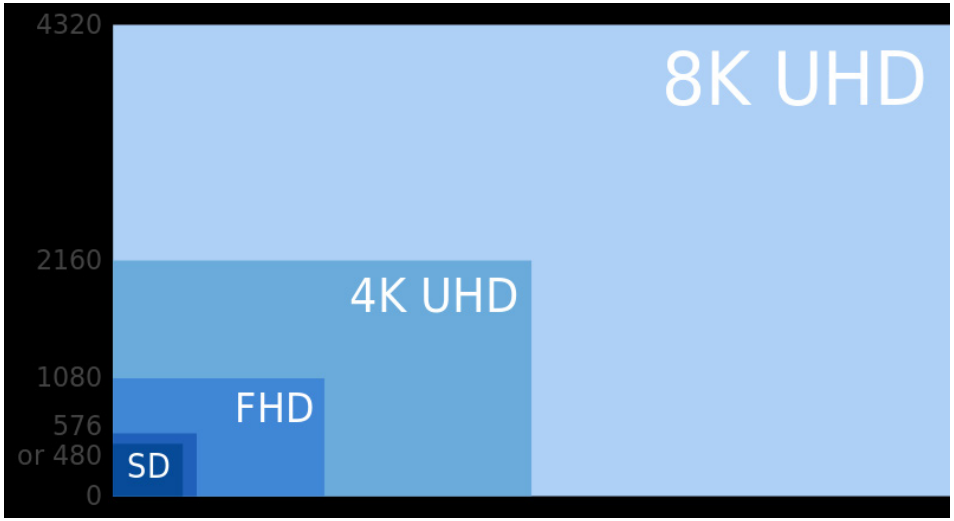
5” 1080p
(HTC)

2011

2012

1080p
iPhone6
2014

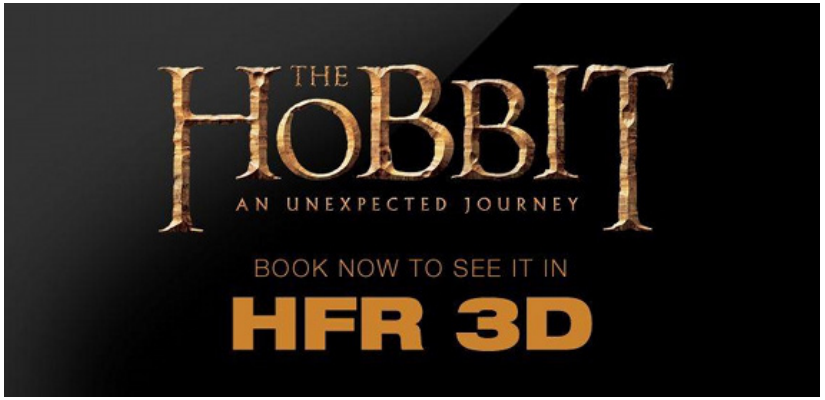
Frame Rate Going Up



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 => 3.8Gpixels/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

$g[\cdot,\cdot]$

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

Slide credit: David Lowe (UBC)

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

0

$h[.,.]$

	0								

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

Image filtering

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

$f[.,.]$

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

$h[.,.]$

	0	10							

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20						

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

Credit: S. Seitz

Image filtering

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

$f[.,.]$

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

$h[.,.]$

	0	10	20	30					

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
				?					

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

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

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

Credit: S. Seitz

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

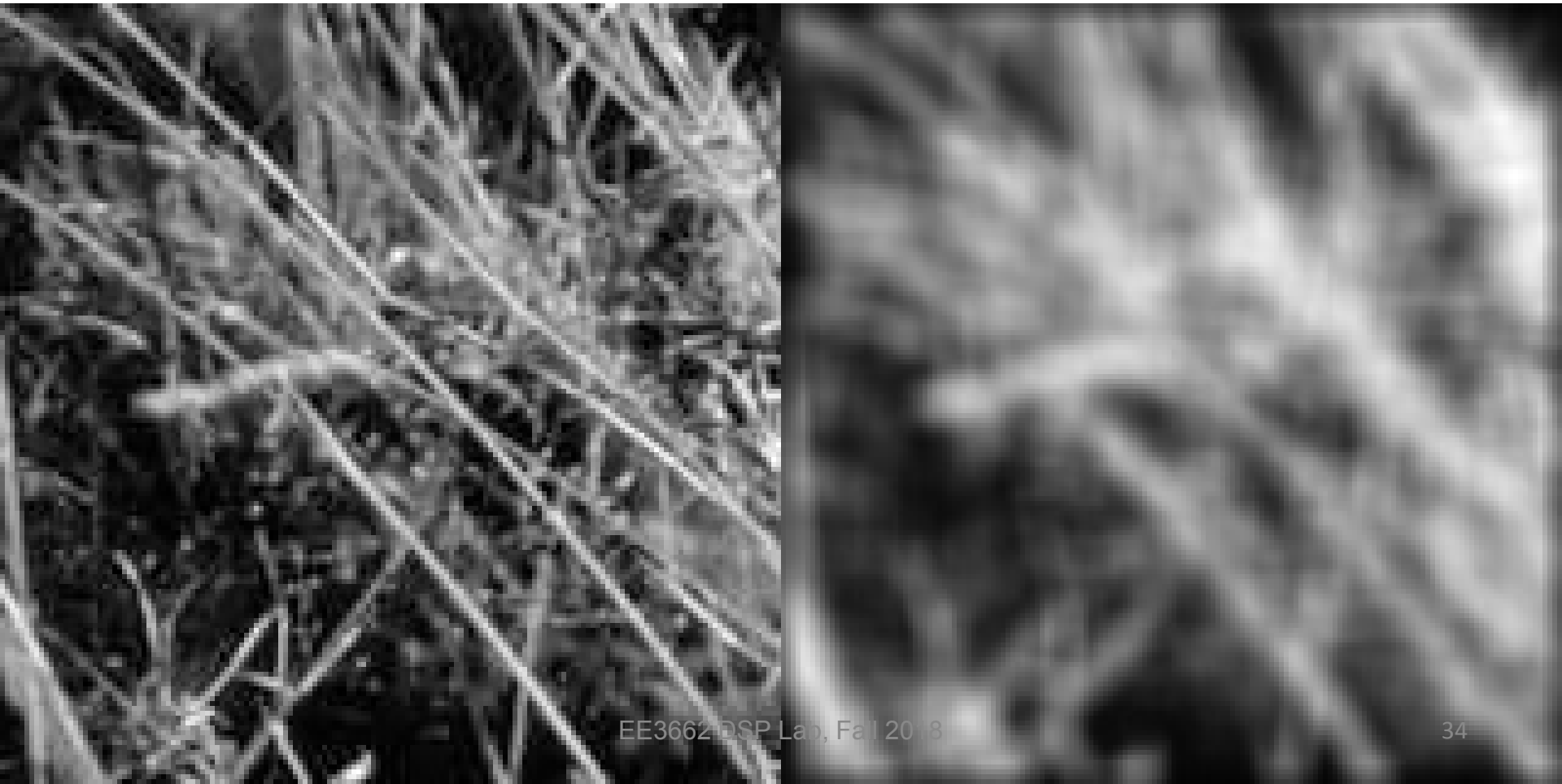
- Achieve smoothing effect (remove sharp features)

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

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

Slide credit: David Lowe (UBC)

Smoothing with box filter



Practice with linear filters

Original

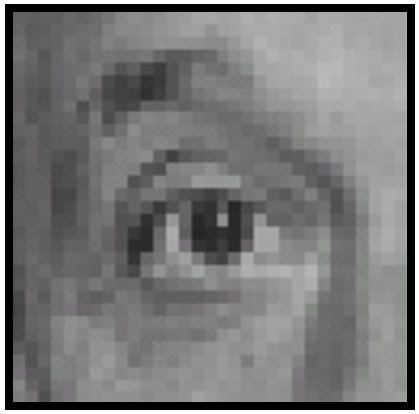
0	0	0
0	1	0
0	0	0

?

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters

Original

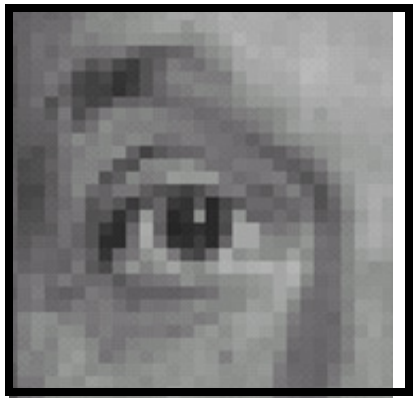
0	0	0
0	0	1
0	0	0

?

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Practice with linear filters

Original

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

(Note that filter sums to 1)

Practice with linear filters

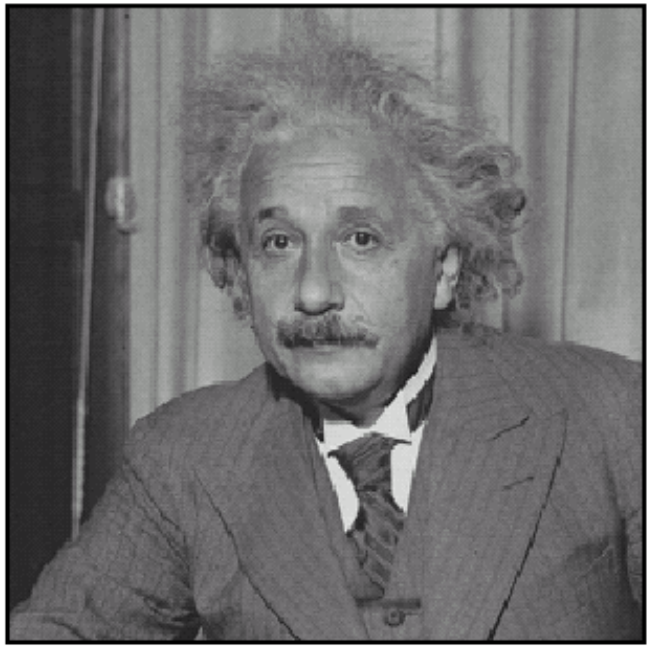
Original

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

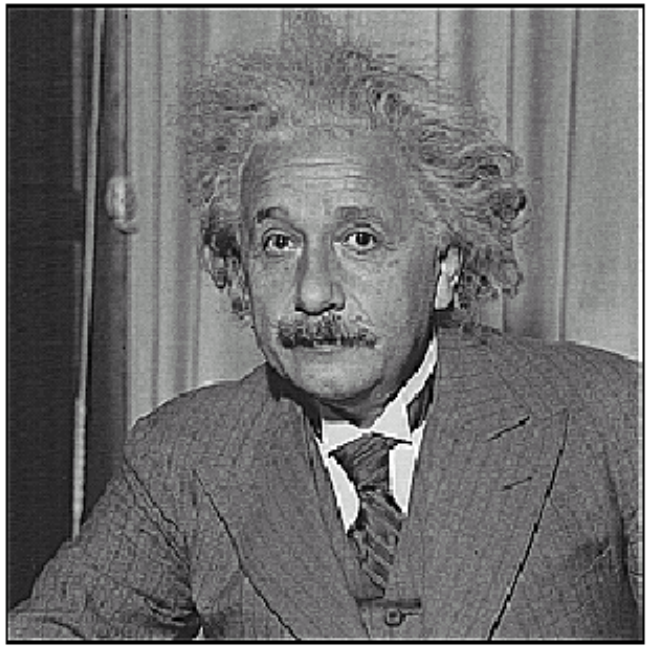
Sharpening filter

- Accentuates (突顯) differences with local average

Sharpening



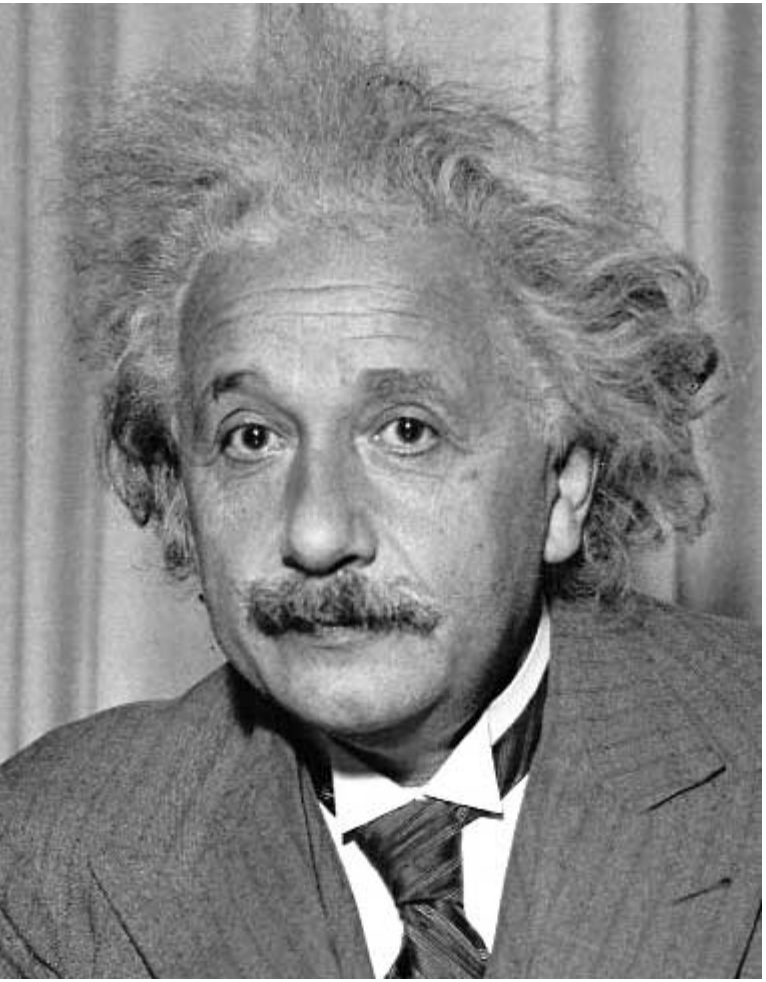
before



after

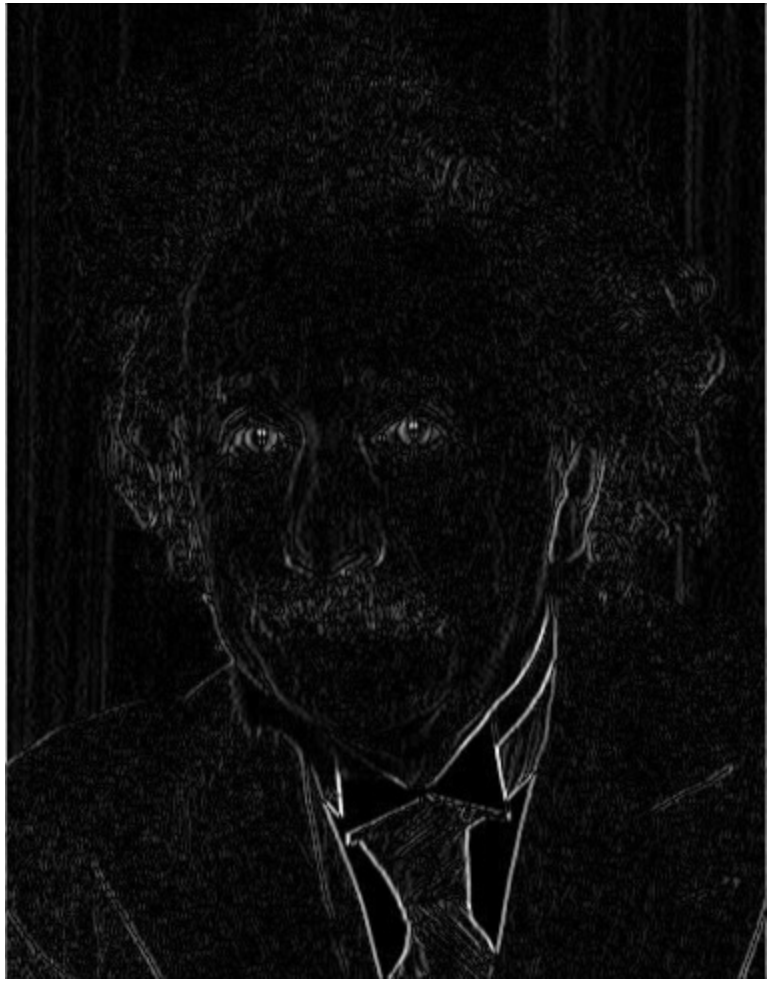
Source: D. Lowe

Other filters



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

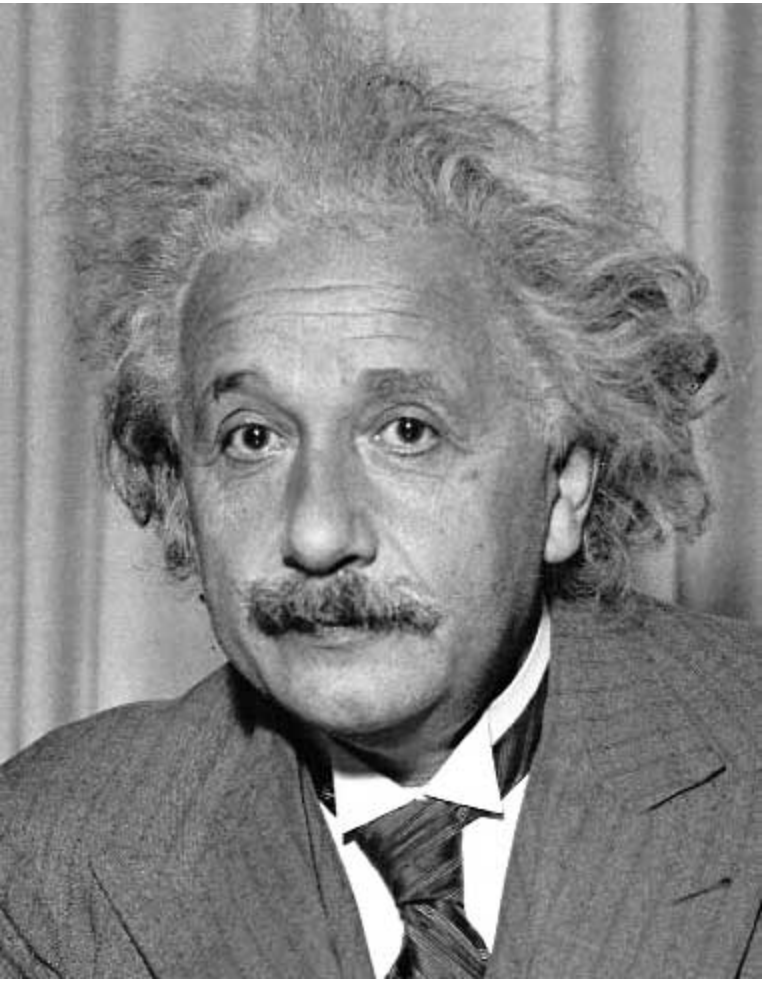
Sobel



Vertical Edge
(absolute value)

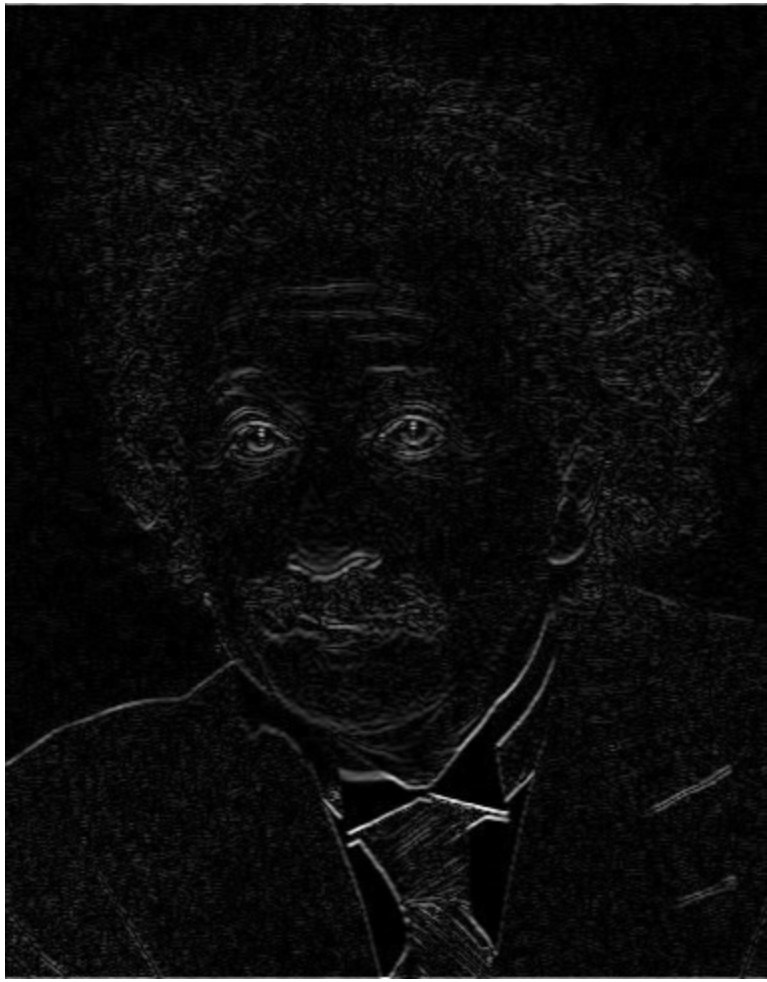
http://en.wikipedia.org/wiki/Sobel_operator

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);

g=filter f=image

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

- 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:

Conv2(g_1 + g_2, f) =

conv2(g_1, f) + conv2(g_2, f)

Shift invariance: same behavior regardless of pixel location

Conv2(shift(g),f) = shift(conv2(g,f))

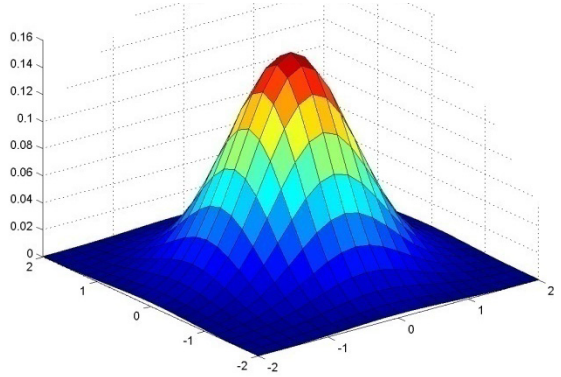
Any linear, shift-invariant operator can be represented as a **convolution**

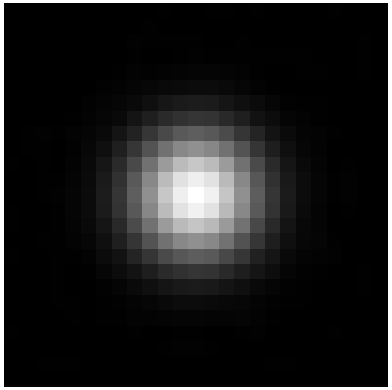
More properties

- 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





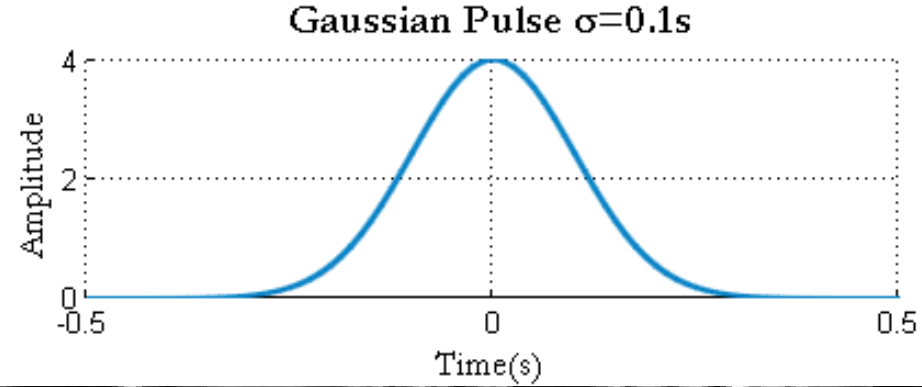
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

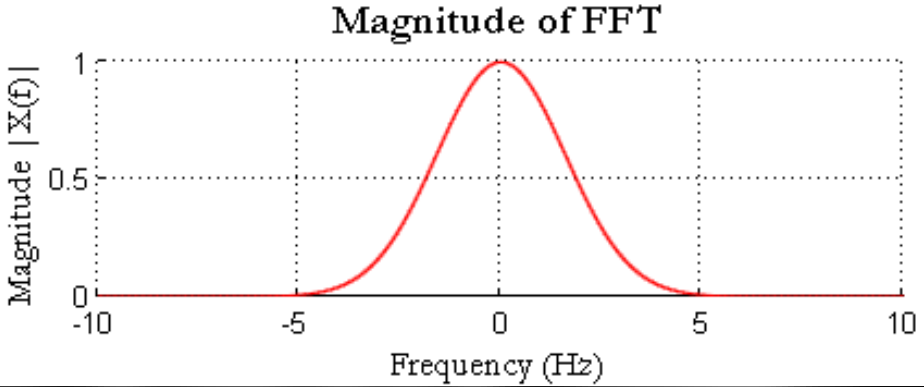
5 x 5, $\sigma = 1$

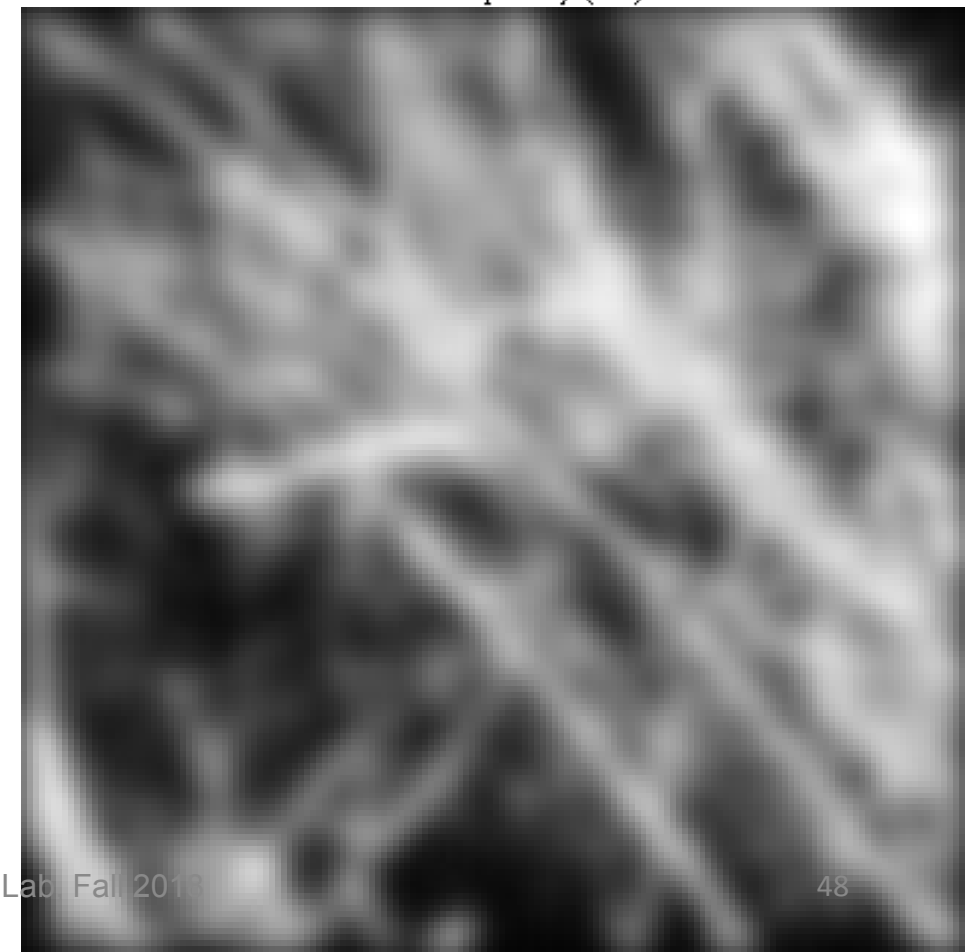
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

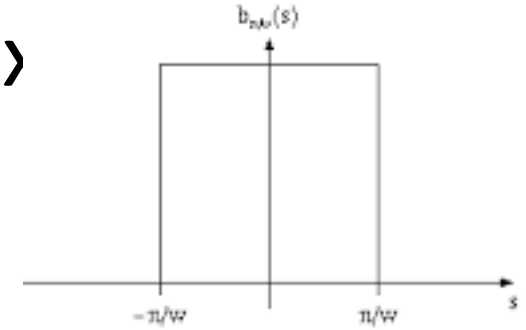
Smoothing with Gaussian filter

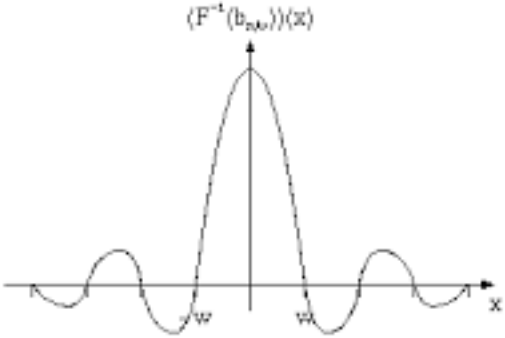


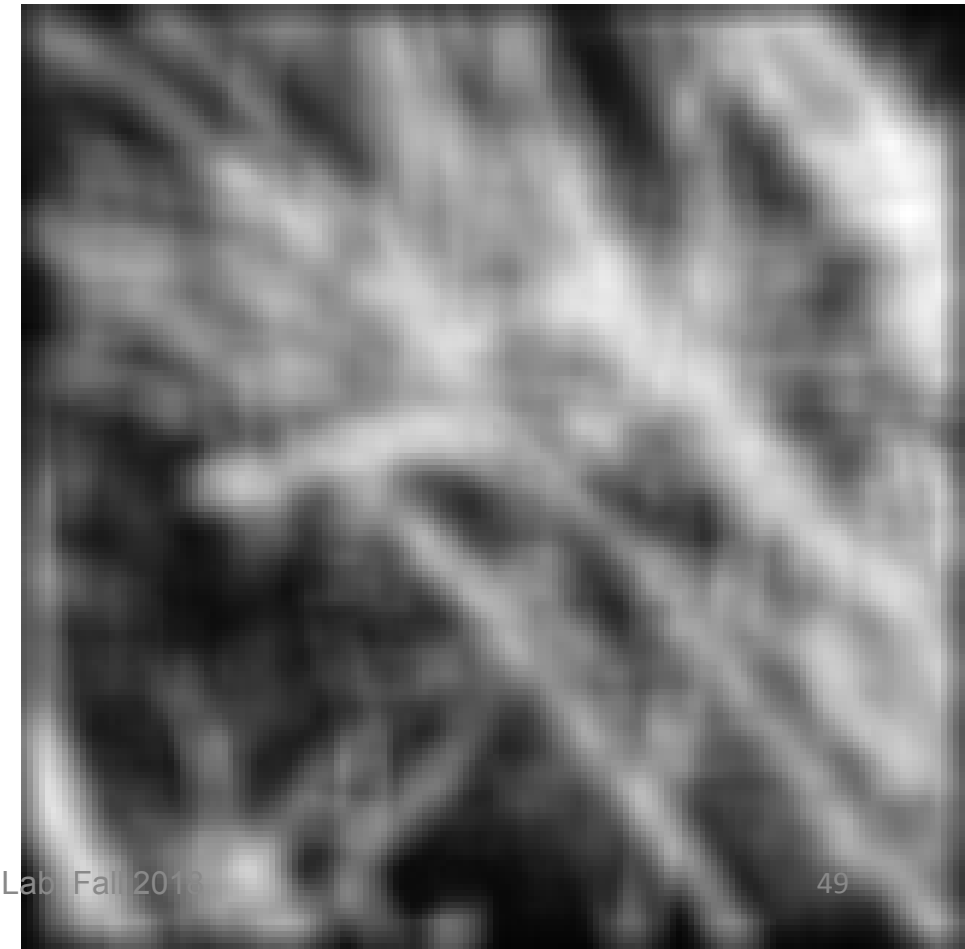




Smoothing with box







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 σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

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)

1	2	1
2	4	2
1	2	1

$*$

2	3	3
3	5	5
4	4	6

The filter factors
into a product of 1D
filters:

1	2	1
2	4	2
1	2	1

$=$

1
2
1

x

1	2	1

Perform convolution
along rows:

1	2	1

$*$

2	3	3
3	5	5
4	4	6

$=$

	11	
	18	
	18	

Followed by convolution
along the remaining column:

Source: K. Grauman

Separability

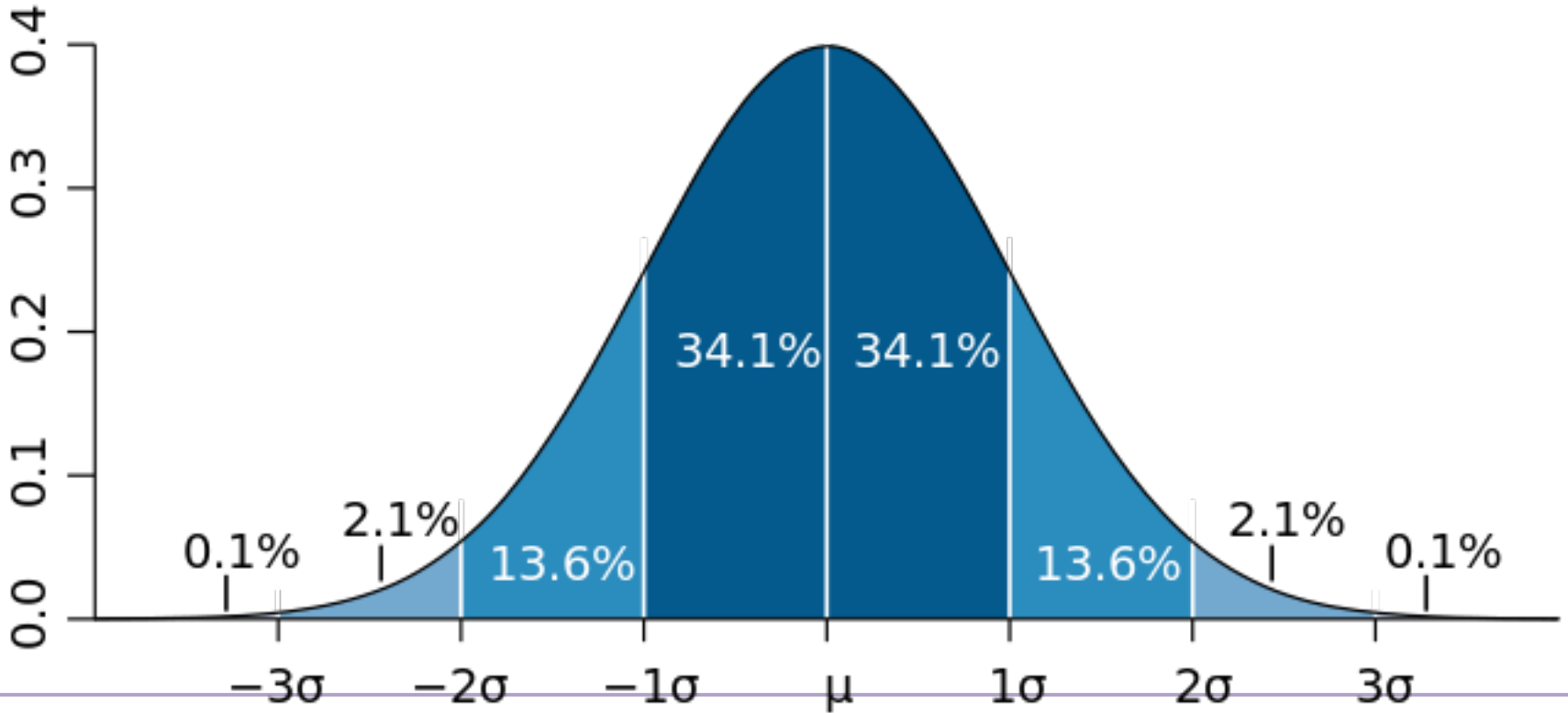
- Why is separability useful in practice?
  - Full filter: Filtering an M-by-N image with a P-by-Q filter kernel requires roughly MNPQ multiplies and adds.
  - Separable filter: The first step requires about MNP multiplies and adds. The second requires about MNQ multiplies and adds, for a total of MN(P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



Practical matters

- What about near the 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

Source: S. Marschner

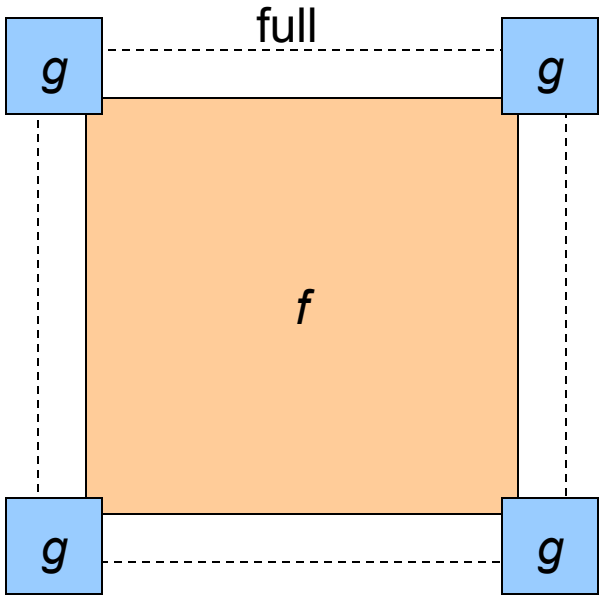
Practical matters

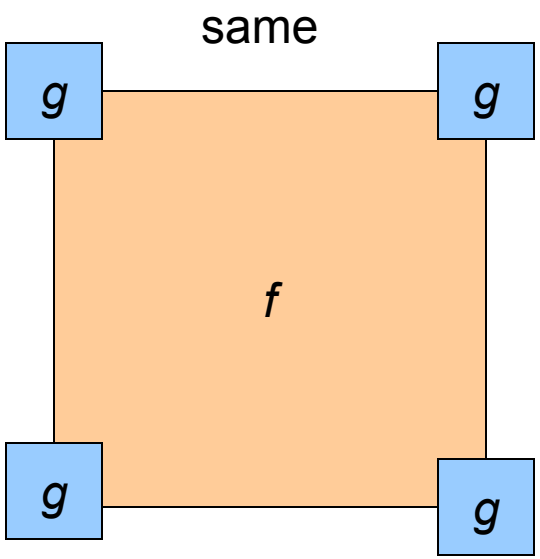
– methods (MATLAB):
  - clip filter (black): imfilter(f, g, 0)
  - wrap around: imfilter(f, g, ‘circular’)
  - copy edge: imfilter(f, g, ‘replicate’)
  - reflect across edge: imfilter(f, g, ‘symmetric’)

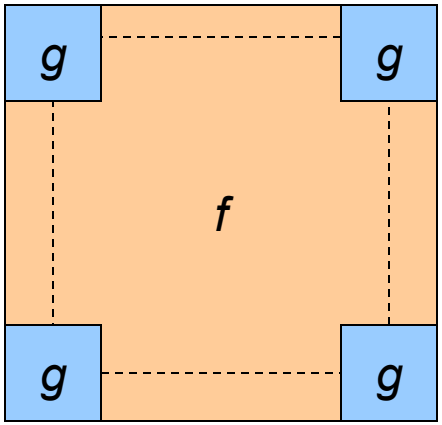
Source: S. Marschner

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



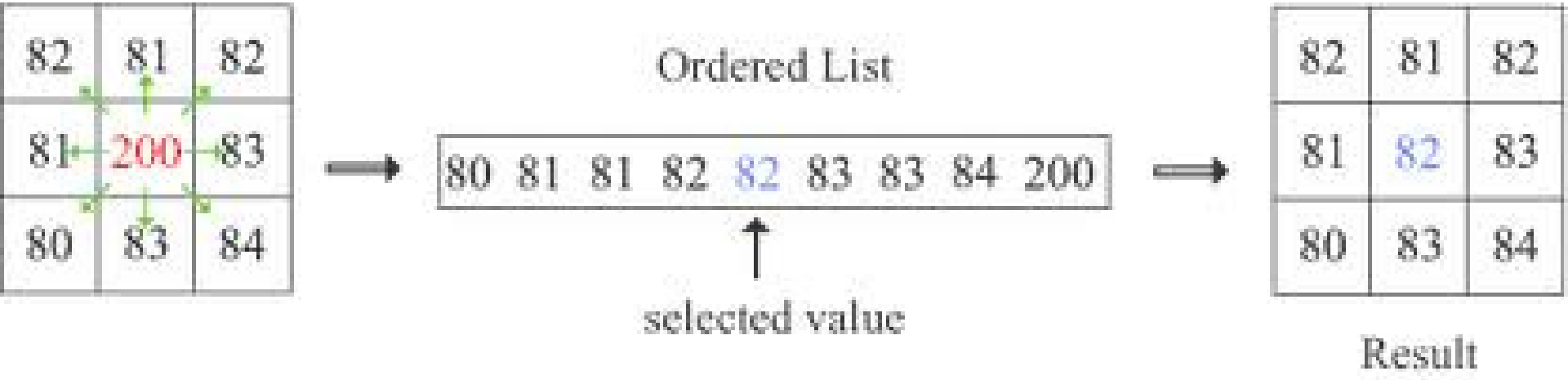




Source: S. Lazebnik

Median filters

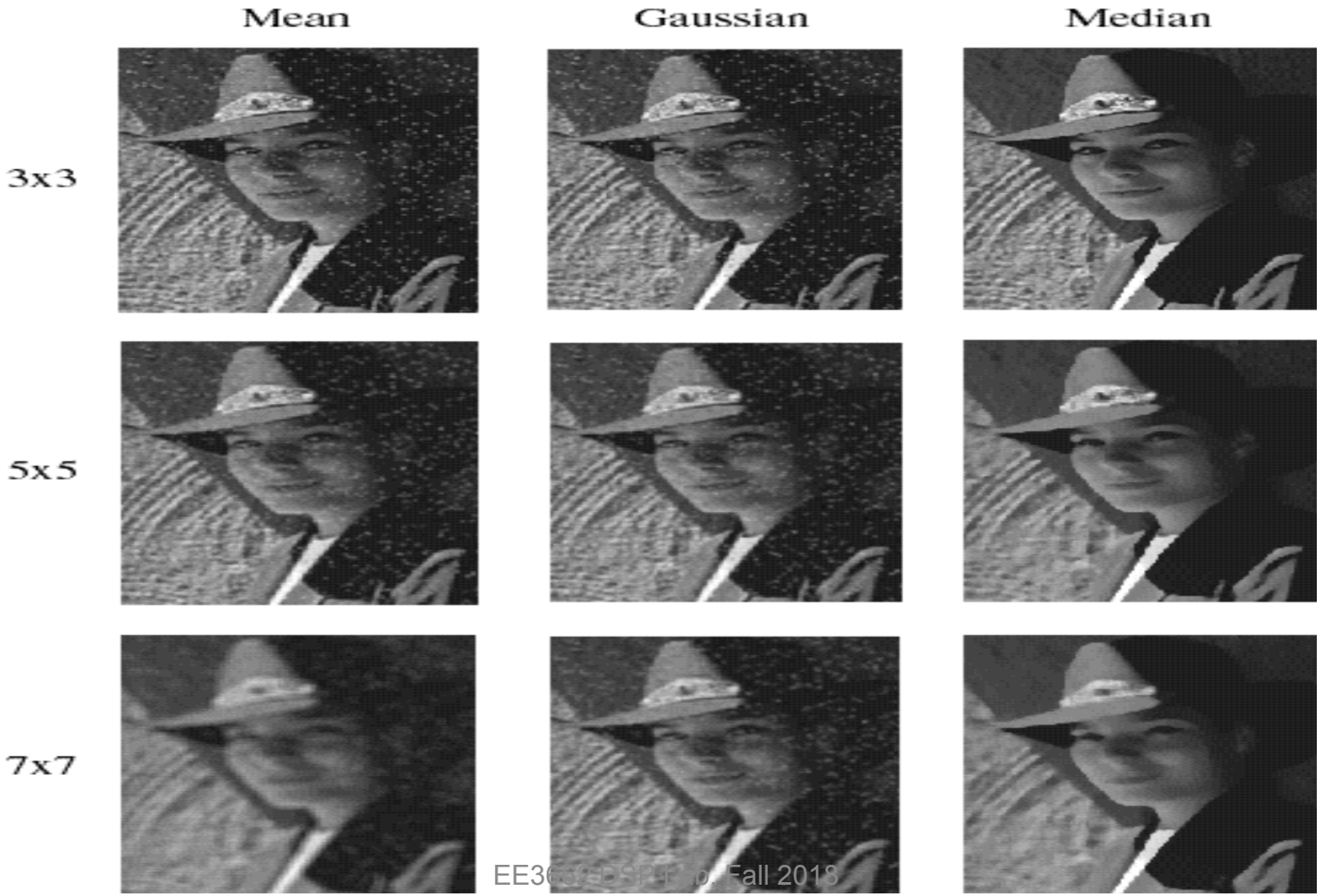
- A **Median Filter** operates over a window by selecting the median intensity in the window.



B = ordfilt2(A,order,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?

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

$$h[m,n] = median_{kl}(f[m+k, n+l])$$

1	1	1
2	1	1
1	1	1

1	1	1
5	1	1
1	1	1

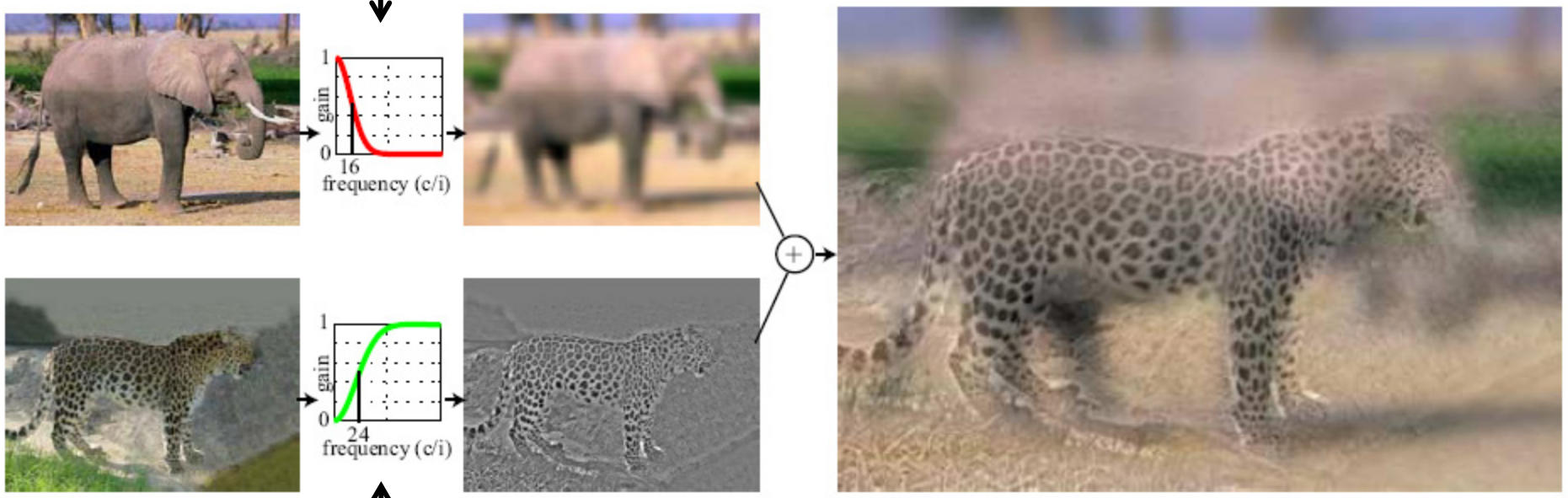
1	5	1
1	1	1
1	1	1

Median: 1 1 1

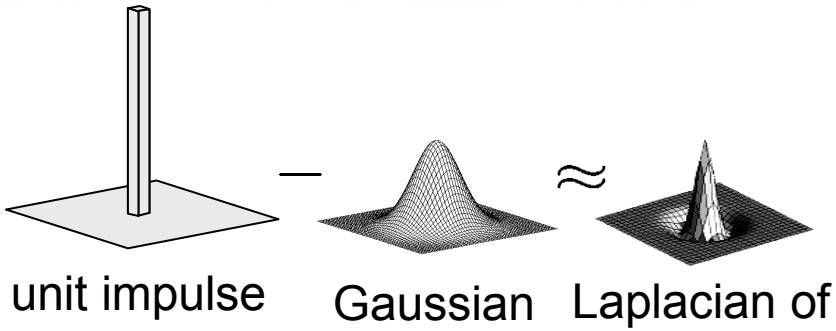
Lab 6: Hybrid Images

Gaussian Filter!

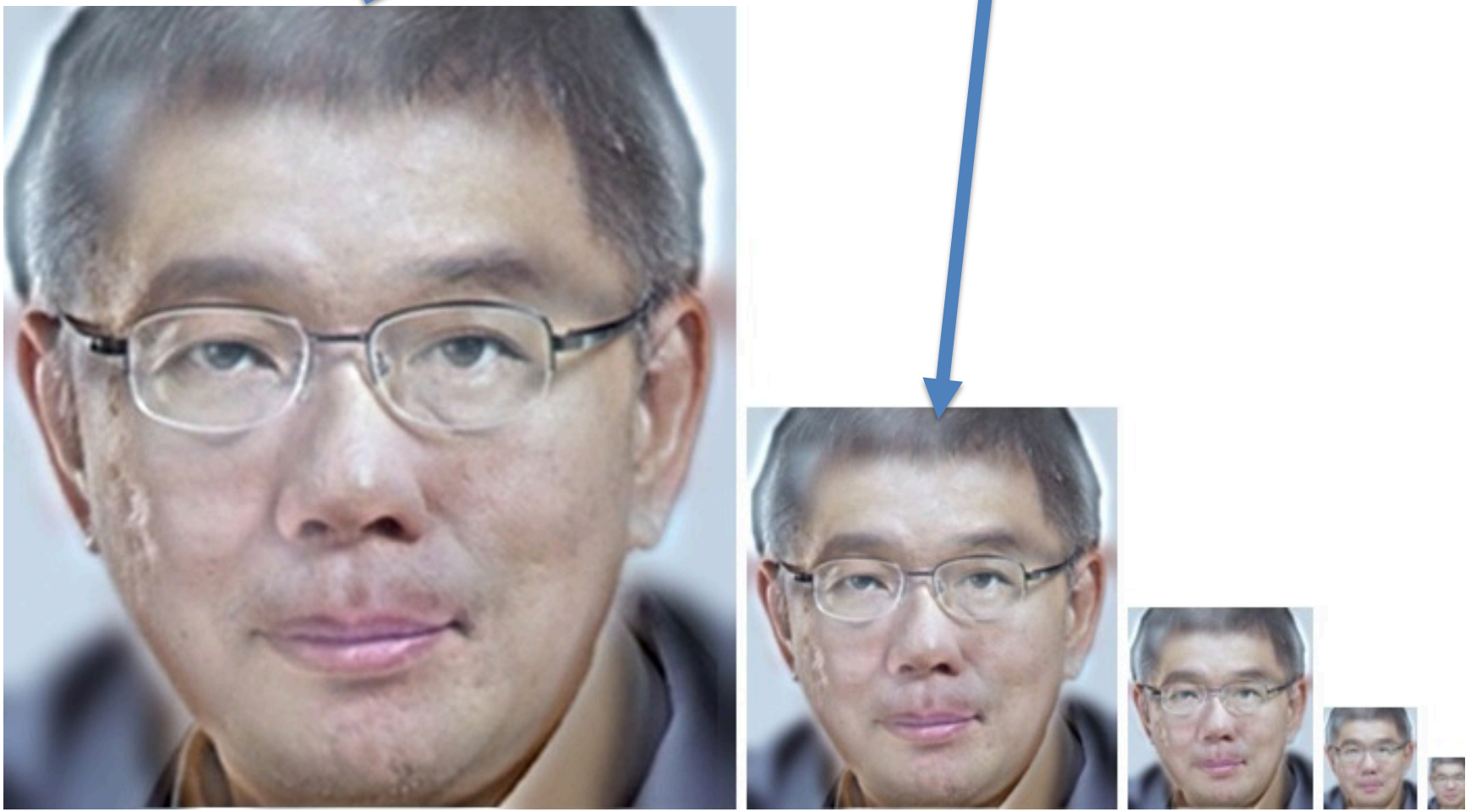
A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006



Laplacian Filter!



Gaussian Filter
+Sampling



Lecture Outline

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

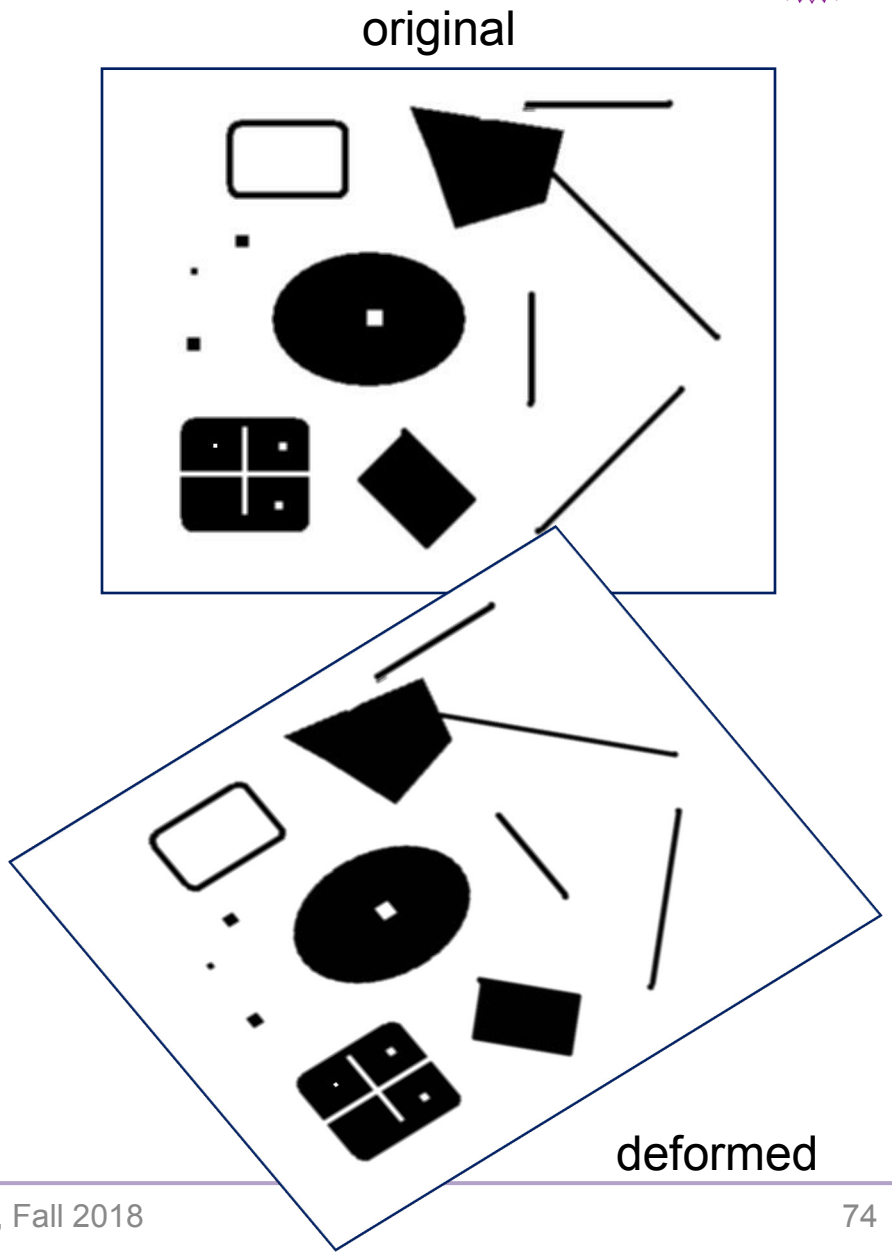
Interest points

- Note: “interest points” = “keypoints”, also sometimes called “features”

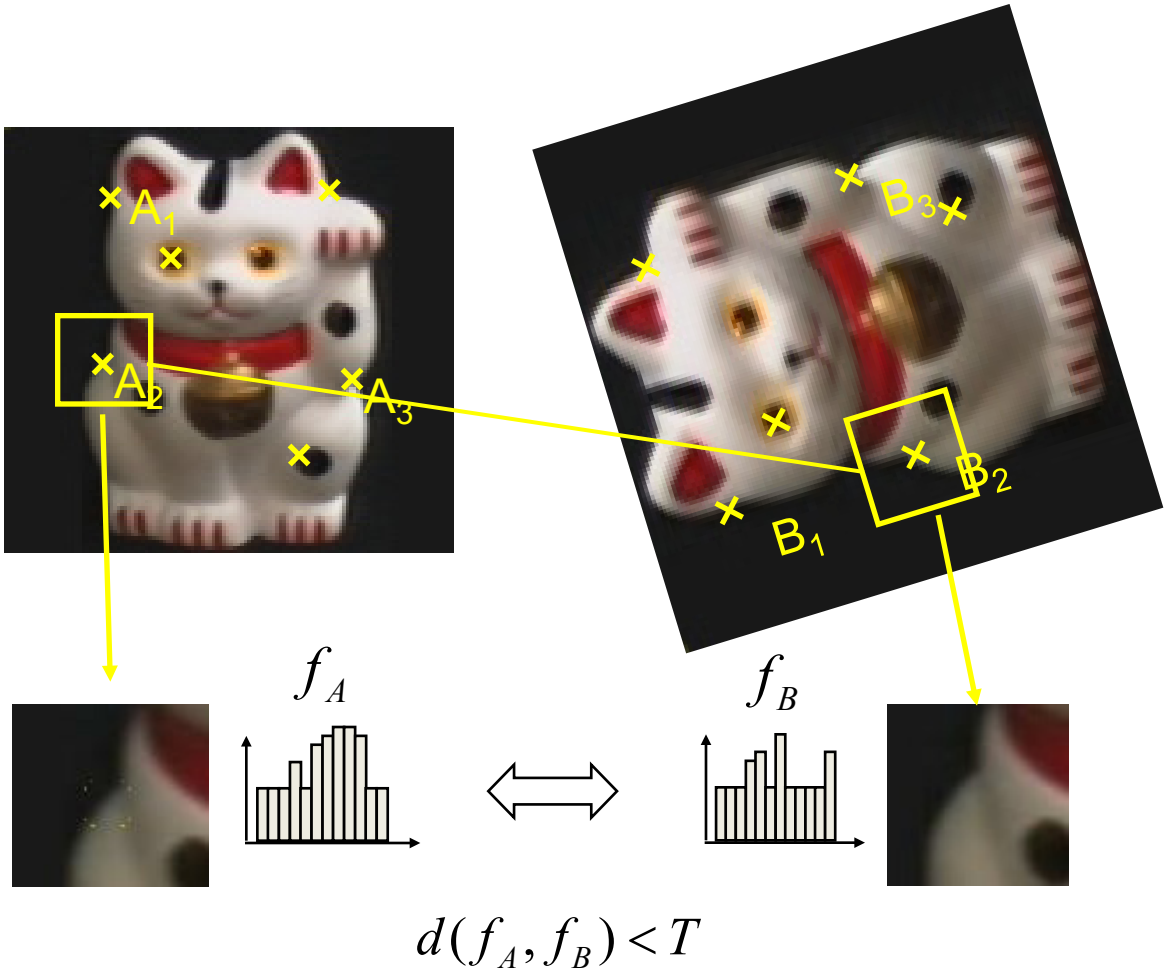
- Many applications
  - tracking: which points are good to track?
  - recognition: find patches likely to tell us something about object category
  - 3D reconstruction: find correspondences across different views

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



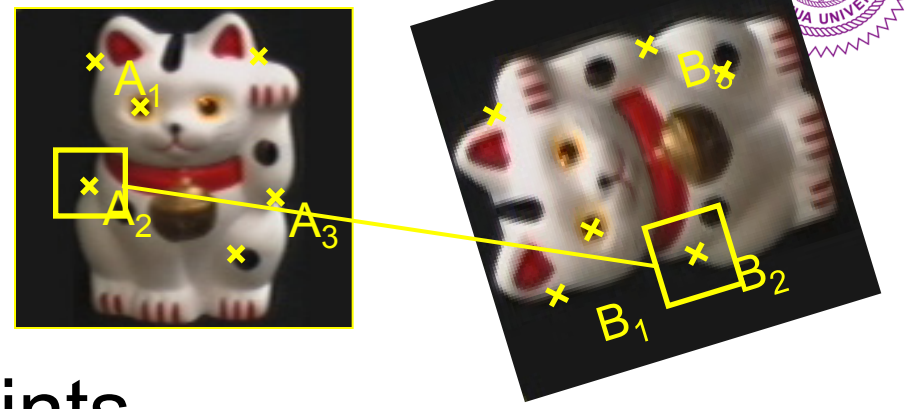
1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

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

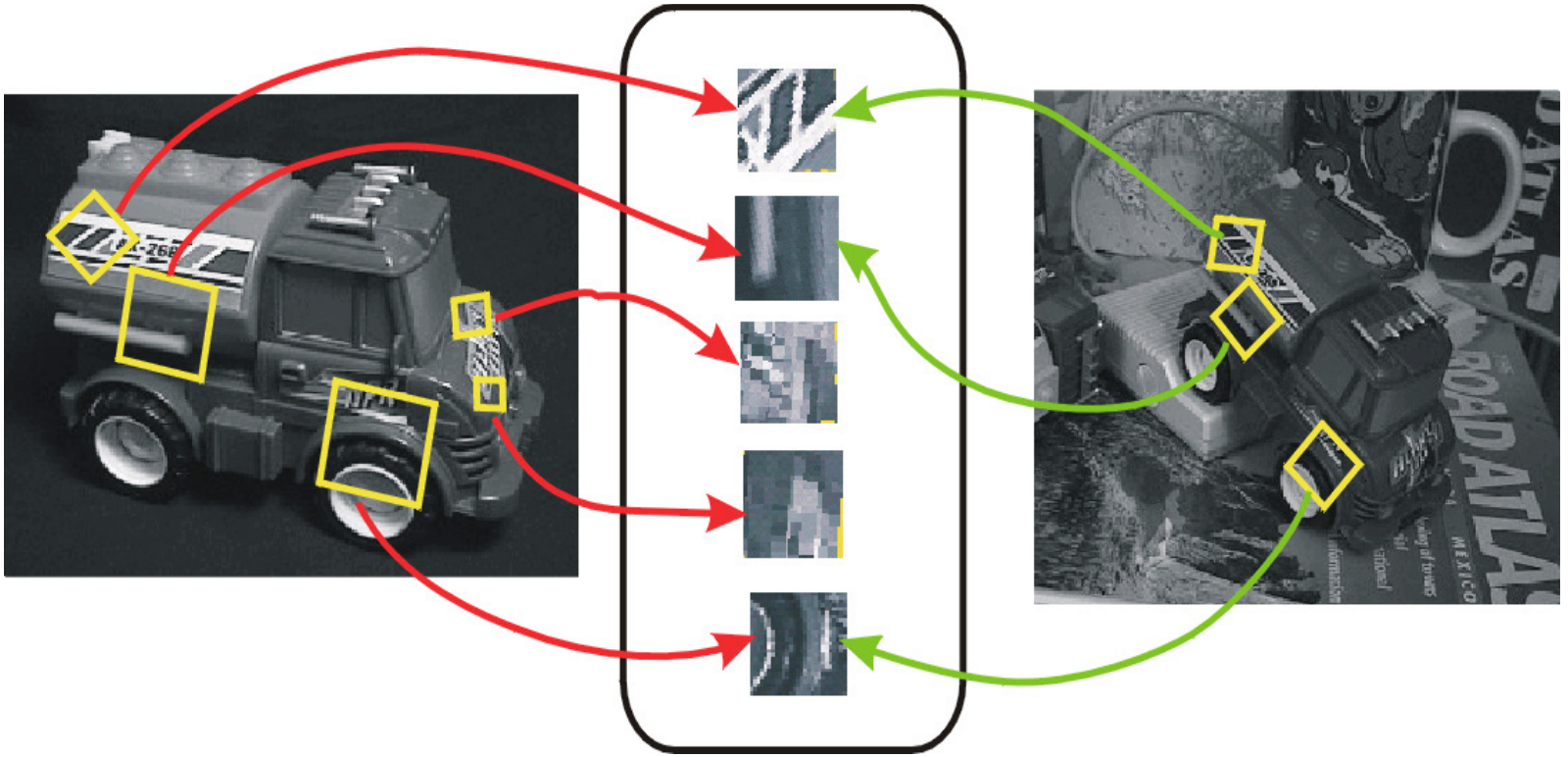
Minimize wrong matches

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



Features Descriptors

Feature extraction: Corners

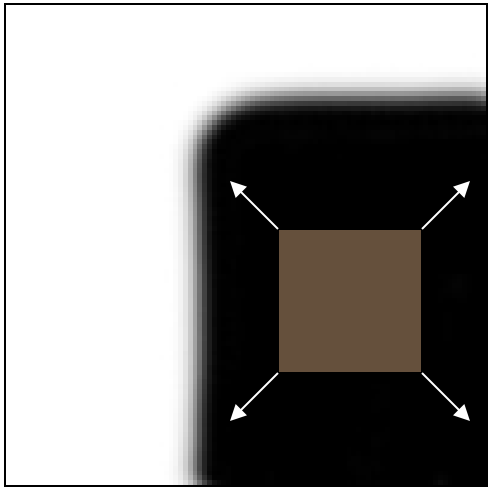
Slides from Rick Szeliski, Svetlana Lazebnik, and Kristin Grauman

Many Existing Detectors Available

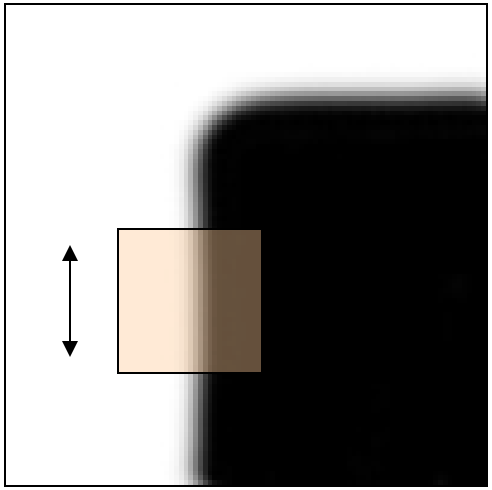
Hessian & Harris	[Beaudet ‘78], [Harris ‘88]
Laplacian, DoG	[Lindeberg ‘98], [Lowe 1999]
Harris-/Hessian-Laplace	[Mikolajczyk & Schmid ‘01]
Harris-/Hessian-Affine	[Mikolajczyk & Schmid ‘04]
EBR and IBR	[Tuytelaars & Van Gool ‘04]
MSER	[Matas ‘02]
Salient Regions	[Kadir & Brady ‘01]
Others…	

Corner Detection (Lab 7): Basic Idea

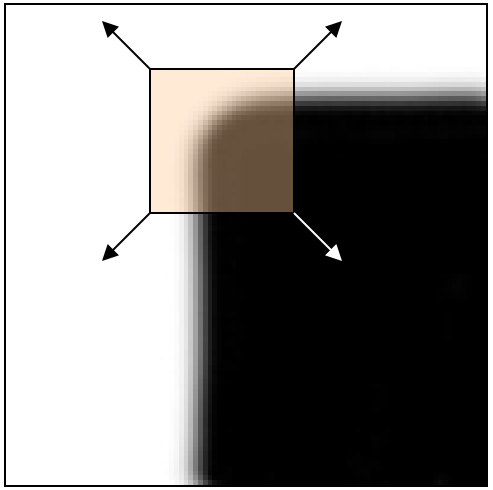
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



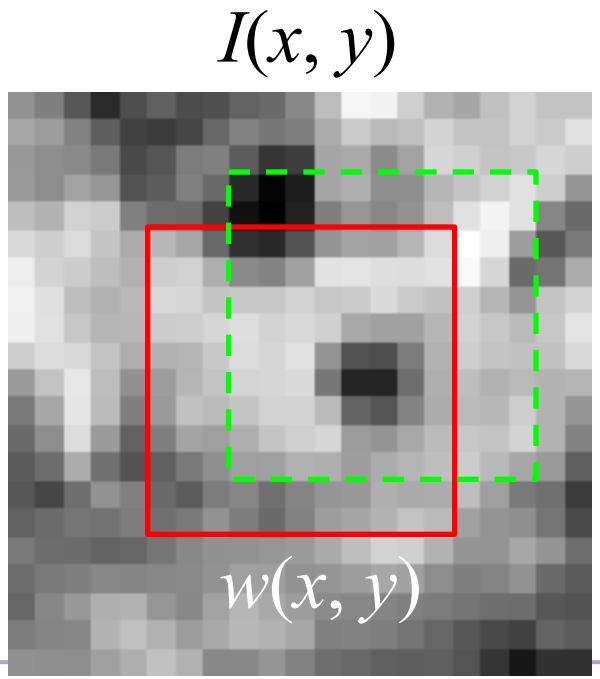
"corner":
significant change in all directions

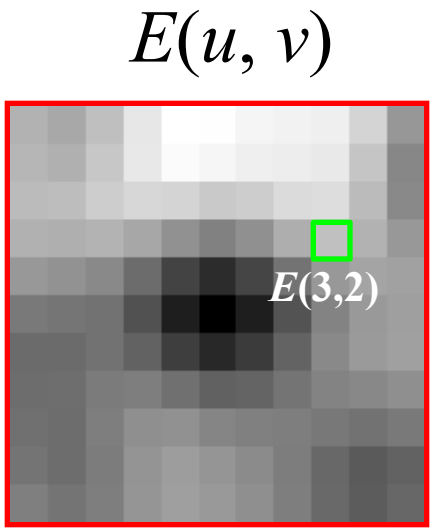
Source: A. Efros

Corner Detection: Mathematics

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

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

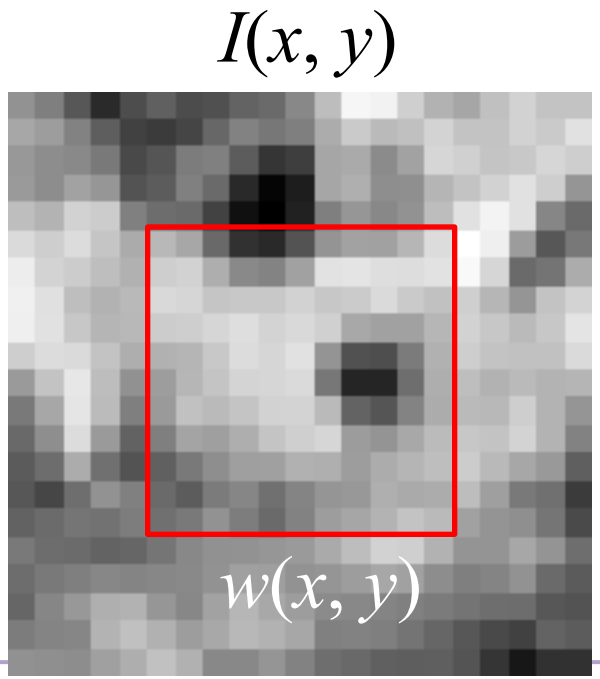


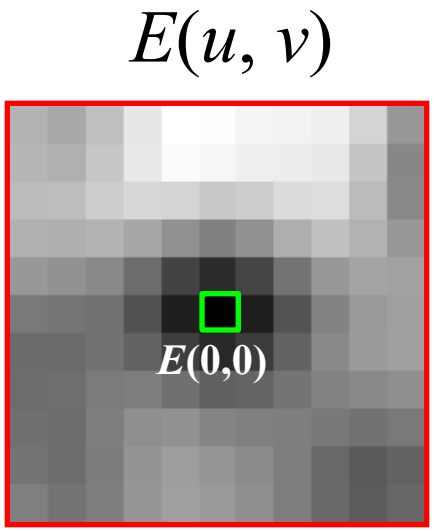


Corner Detection: Mathematics

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

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





Corner Detection: Mathematics

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

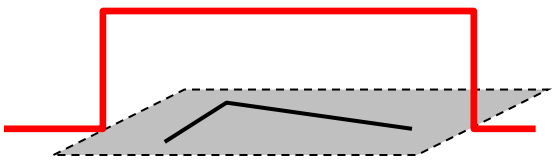
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

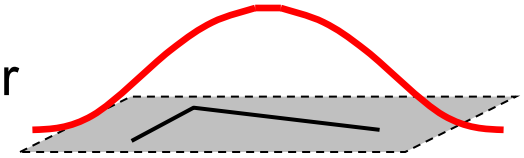
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Source: R. Szeliski

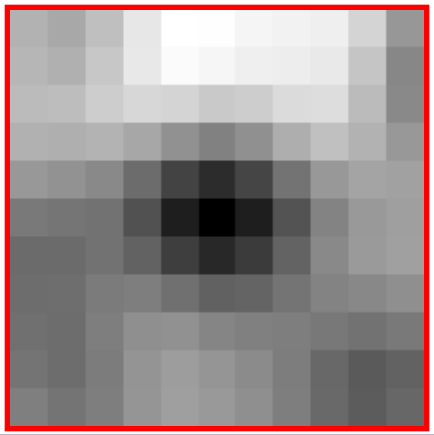
Corner Detection: Mathematics

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

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

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 $[u,v]$:

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

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*:

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

Corner Detection: Mathematics

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

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

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

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

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

$$+\sum_{x,y} 2w(x,y)\left[I(x+u,y+v)-I(x,y)\right]I_{xx}(x+u,y+v)$$

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

$$+\sum_{x,y} 2w(x,y)\left[I(x+u,y+v)-I(x,y)\right]I_{xy}(x+u,y+v)$$

Corner Detection: Mathematics

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

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

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

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

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

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

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

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

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

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

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

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:

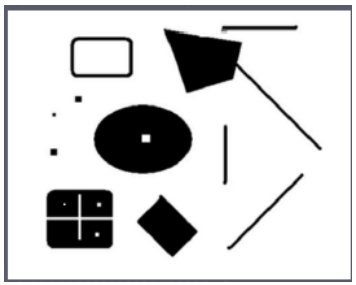
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

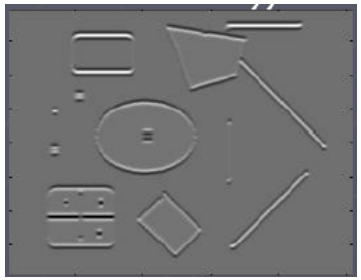
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

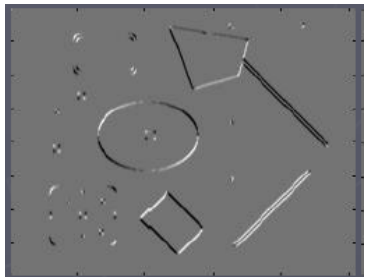
Corners as distinctive interest points

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

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







Notation:

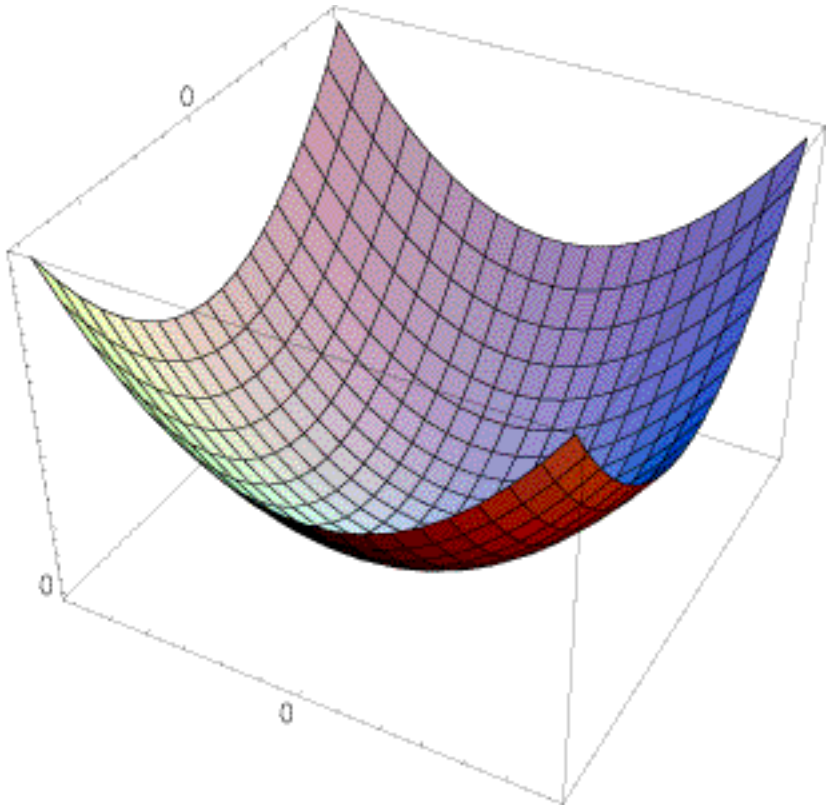
$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_y \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Interpreting the second moment matrix

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

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

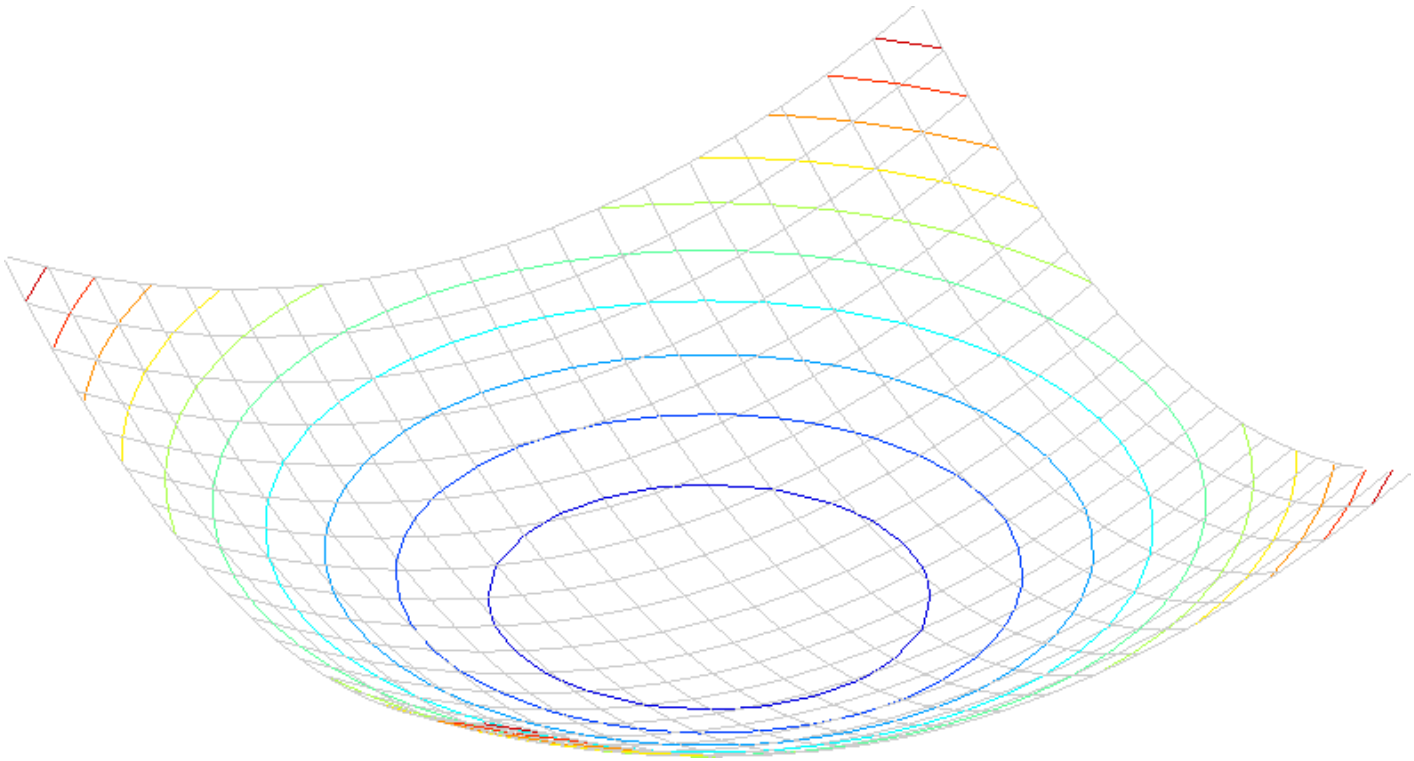
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Interpreting the second moment matrix

Consider a horizontal "slice" of $E(u, v)$: $\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.



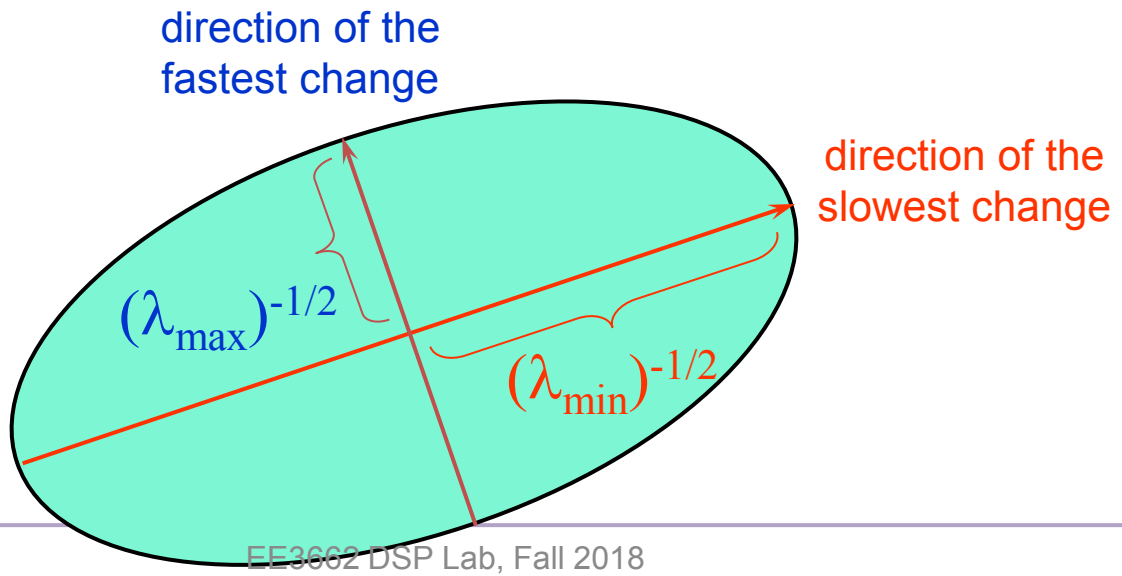
Interpreting the second moment matrix

Consider a horizontal "slice" of $E(u, v)$: $\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.

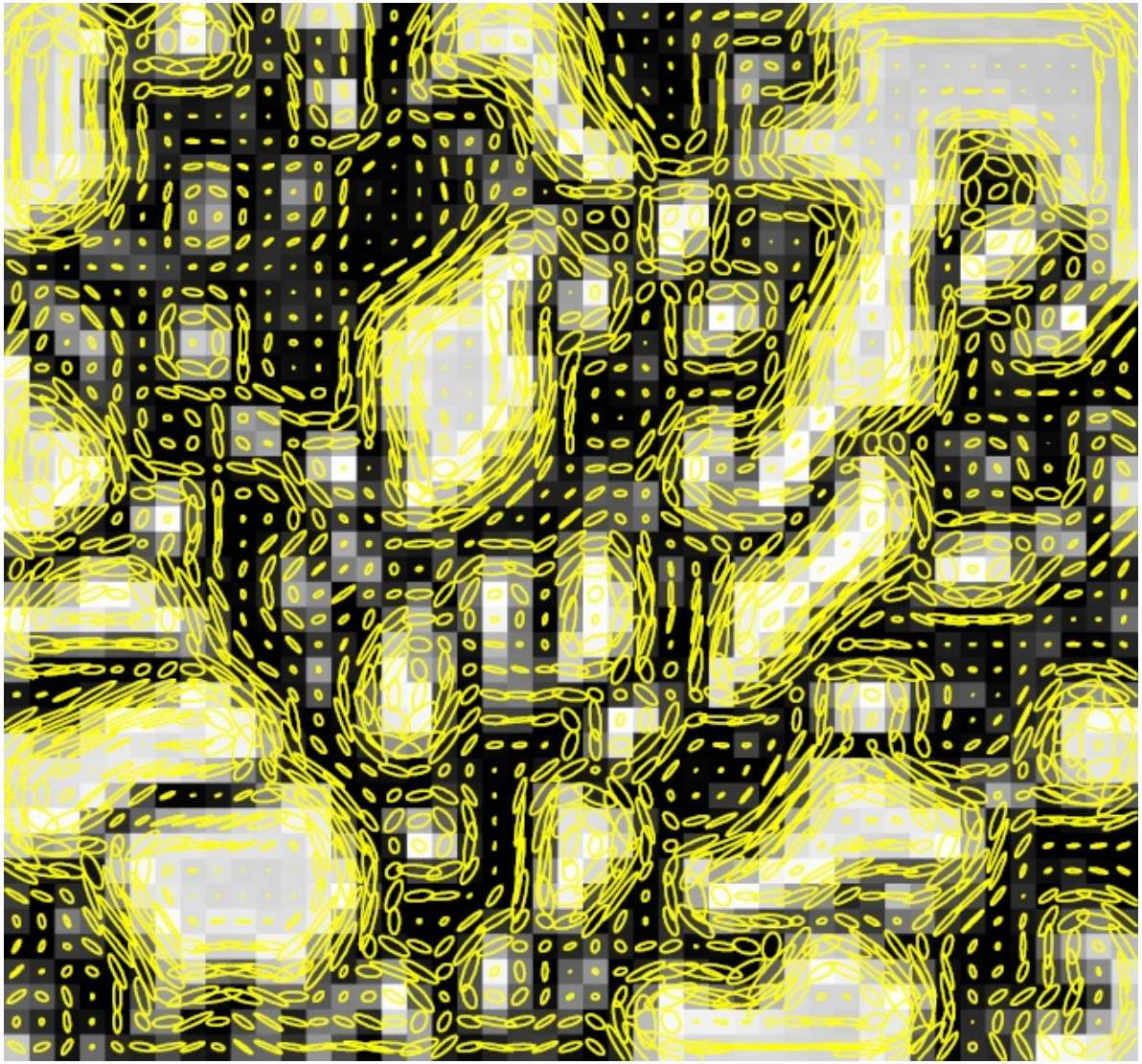
Diagonalization of M: $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

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



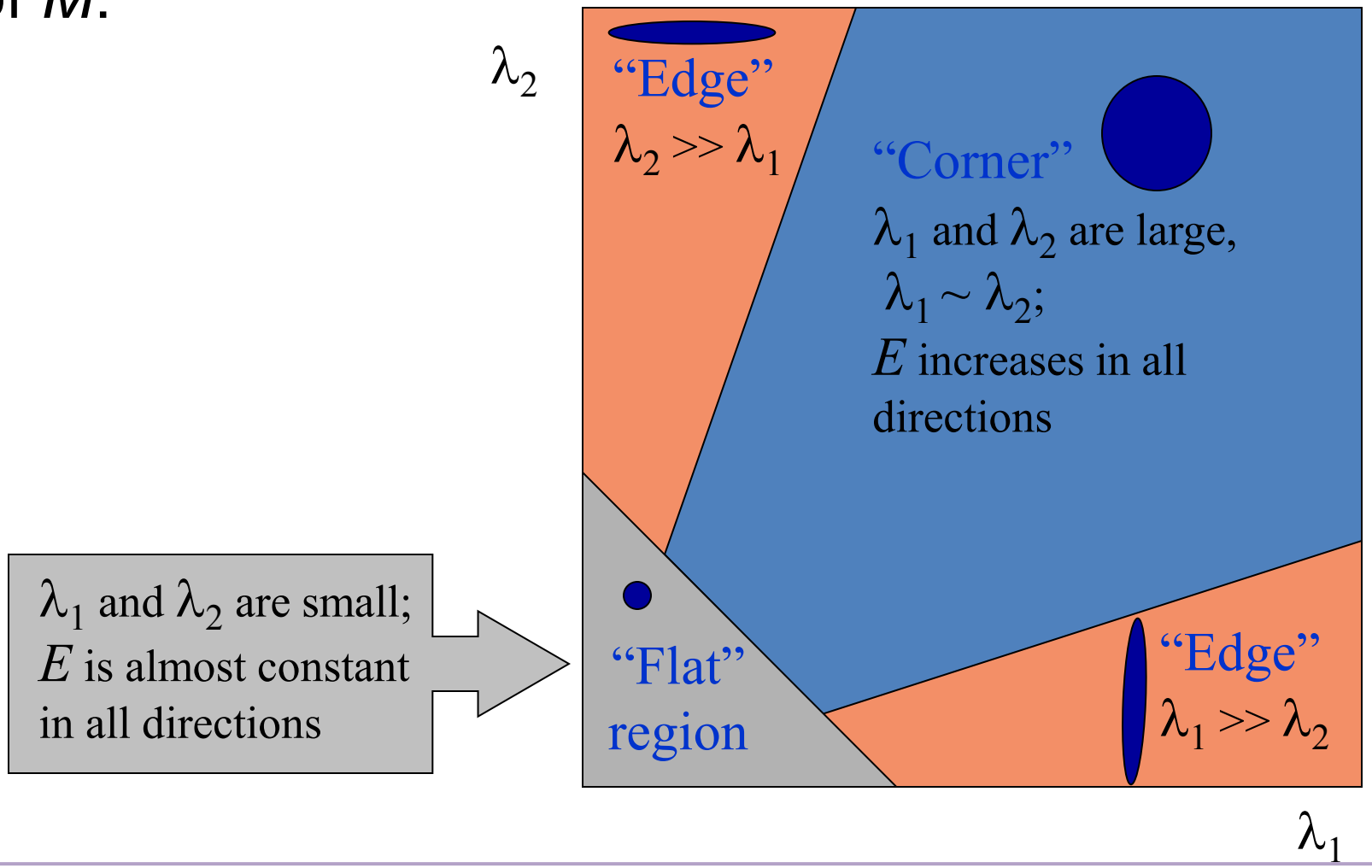
Visualization of second moment matrices

Visualization of second moment matrices



Interpreting the eigenvalues

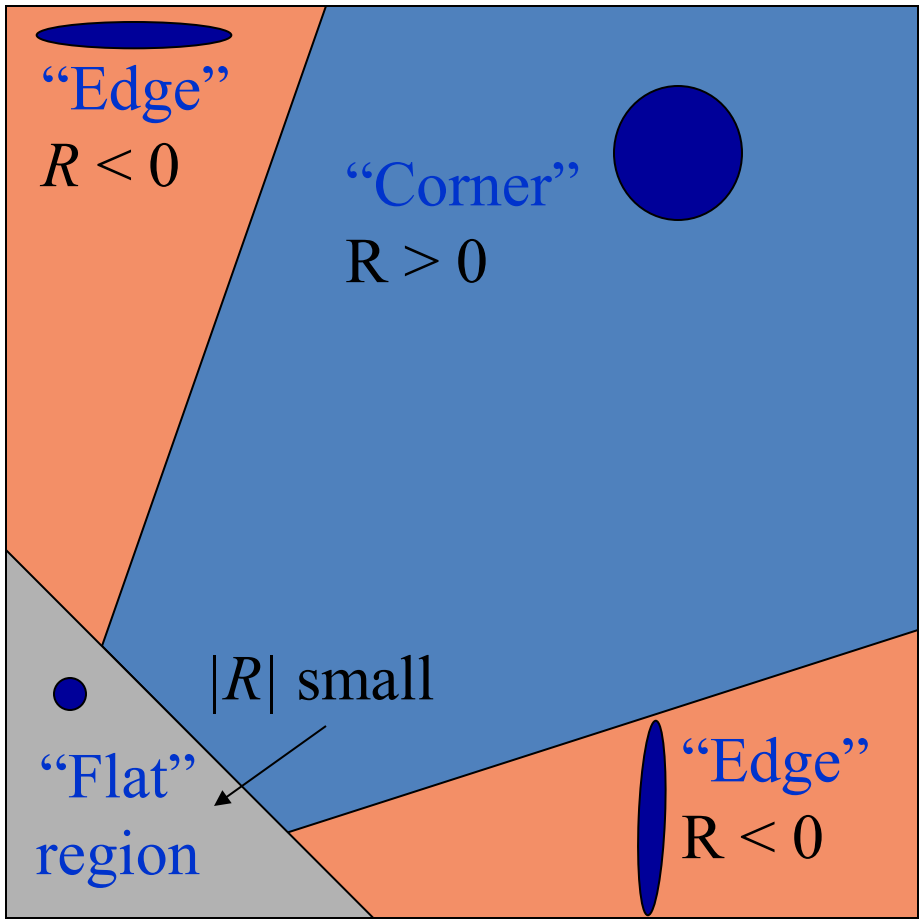
Classification of image points using eigenvalues of *M*:



Corner response function

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

α: constant (0.04 to 0.06)



Harris corner detector

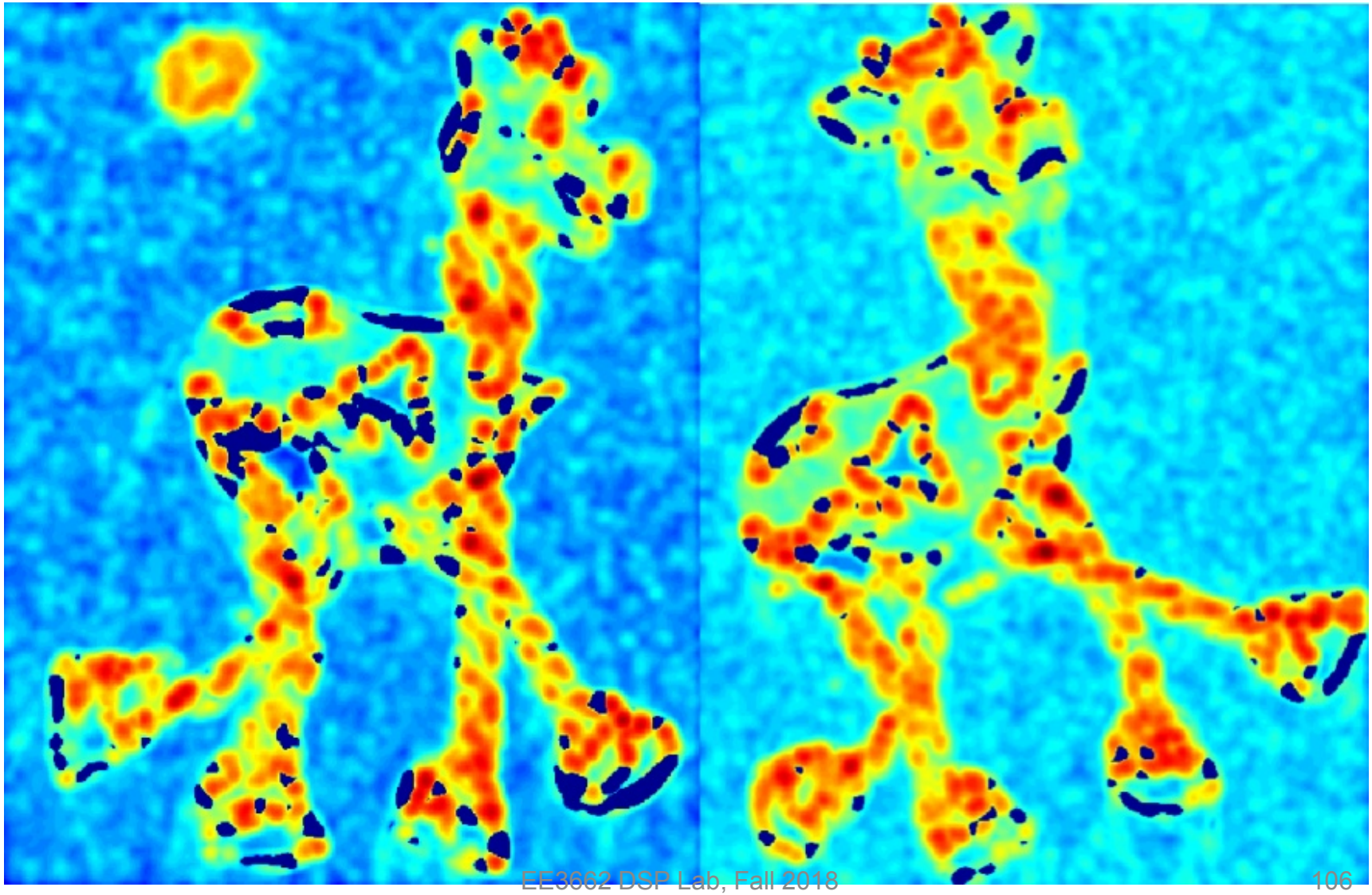
1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*R*> threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

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

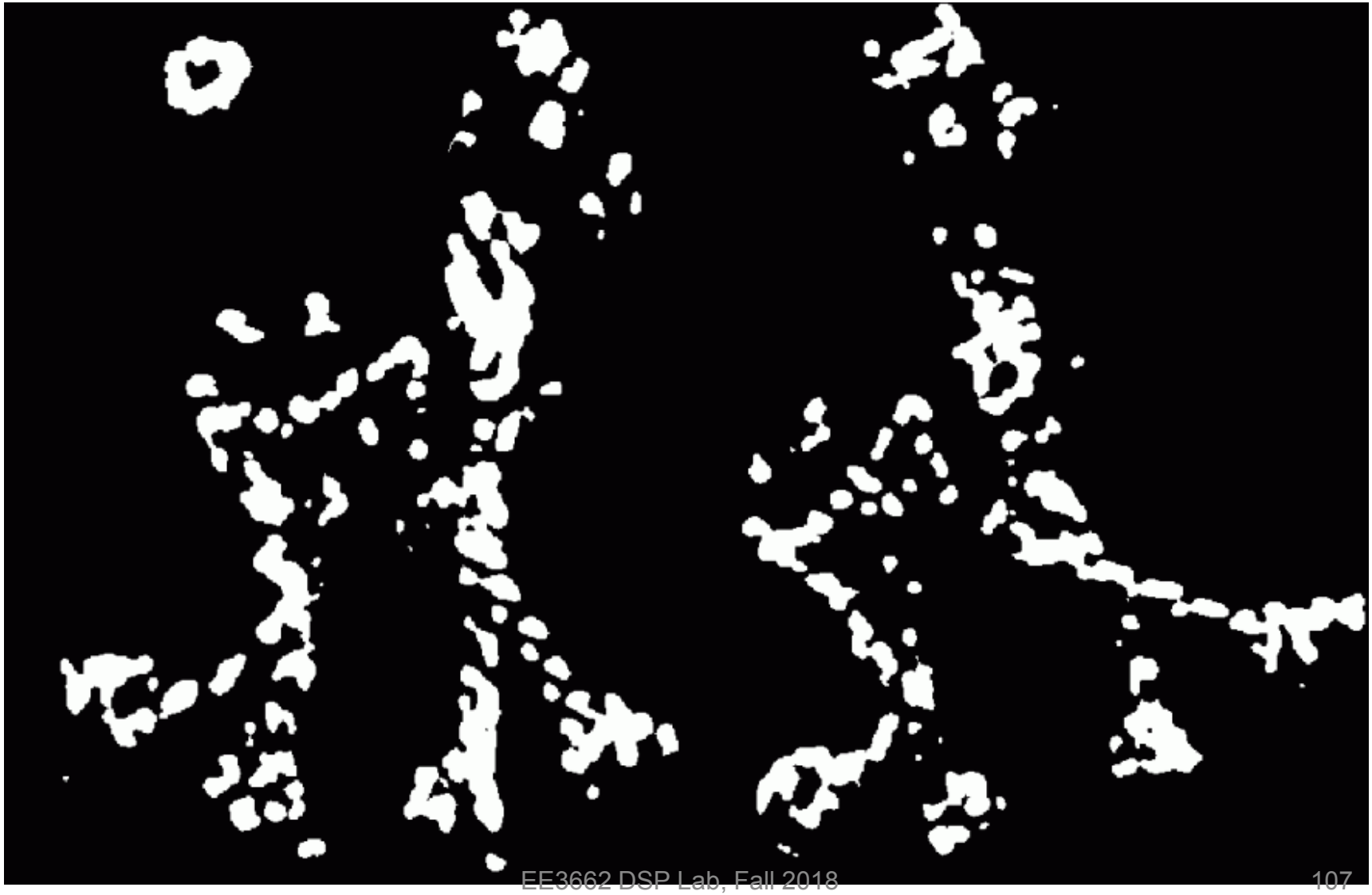
Harris Detector: Steps

Compute corner response R



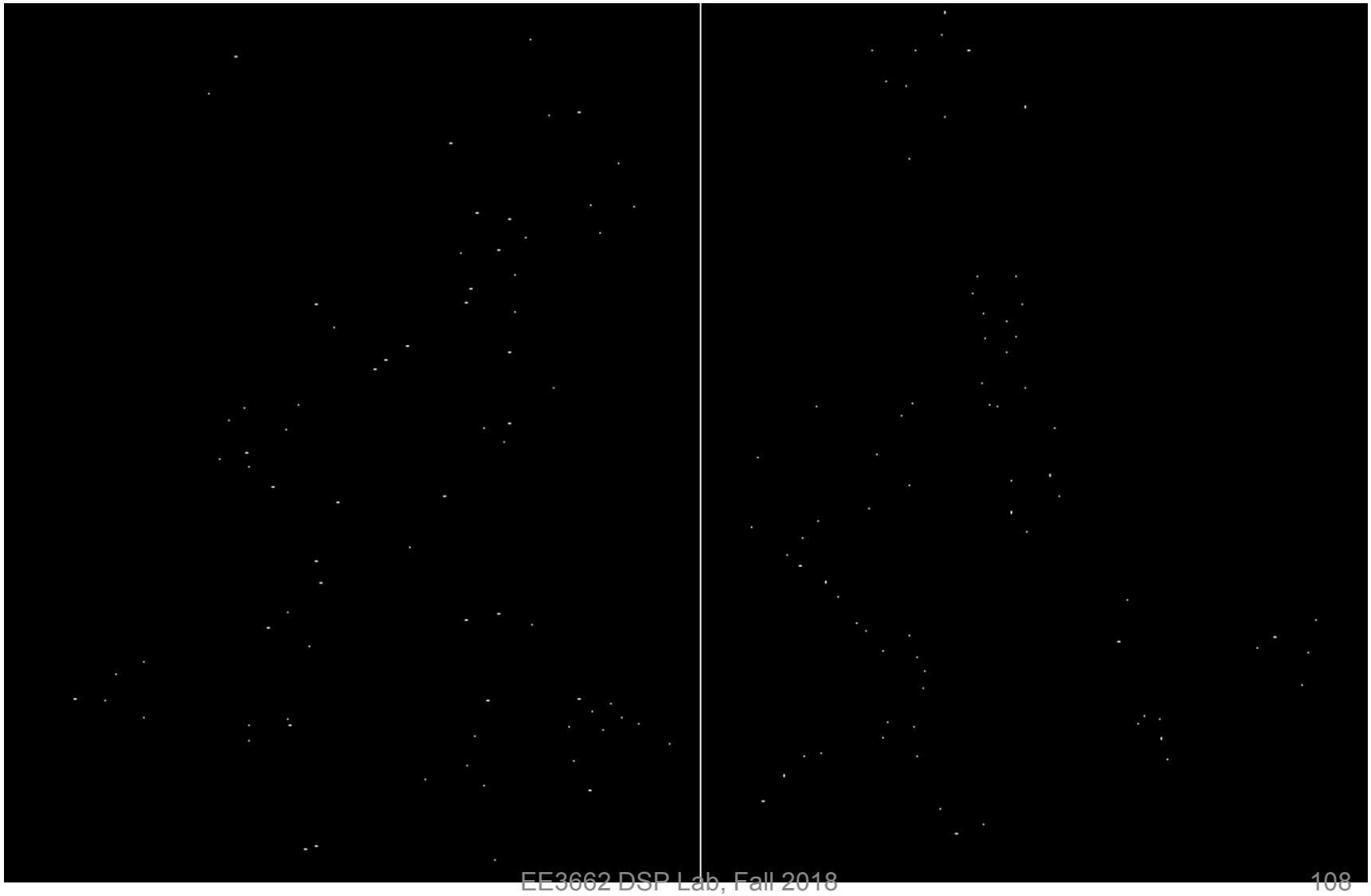
Harris Detector: Steps

Find points with large corner response: R>threshold



Harris Detector: Steps

Take only the points of local maxima of R

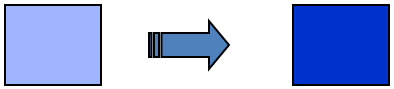


Harris Detector: Steps

Invariance and covariance

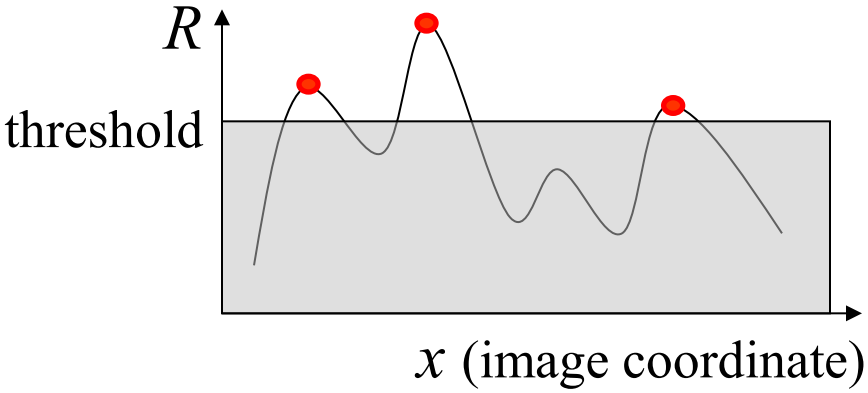
- We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance:** image is transformed and corner locations do not change
  - **Covariance:** if we have two transformed versions of the same image, features should be detected in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

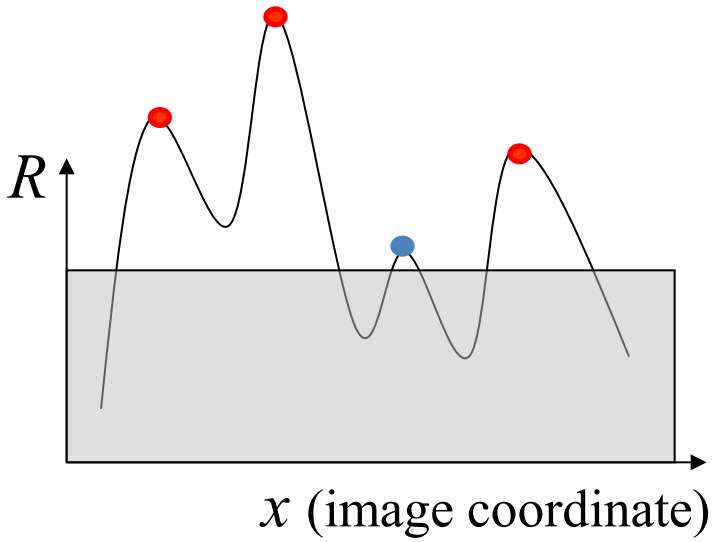
Affine intensity change



$$I \rightarrow a\,I + b$$

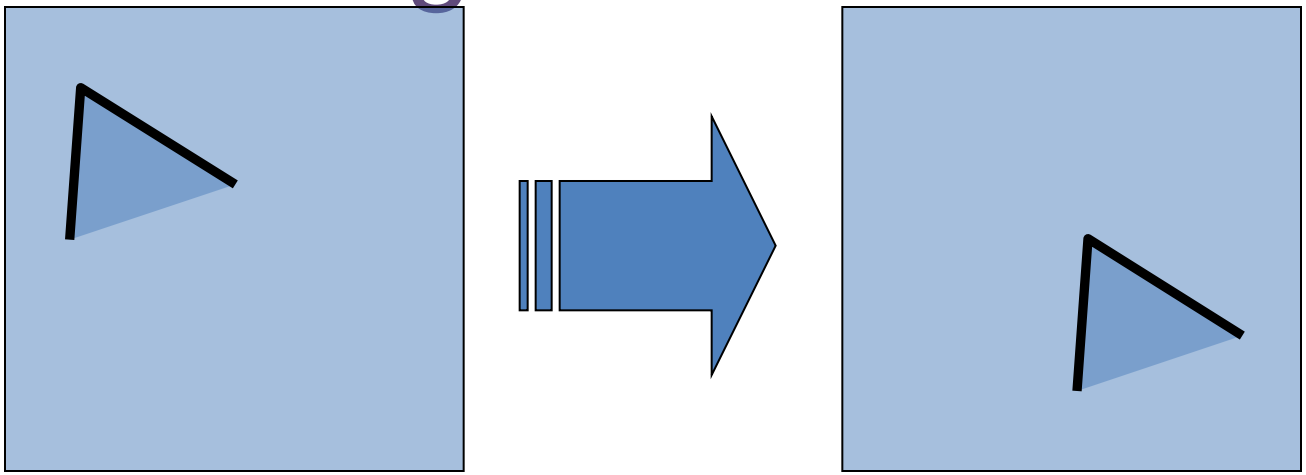
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant to affine intensity change

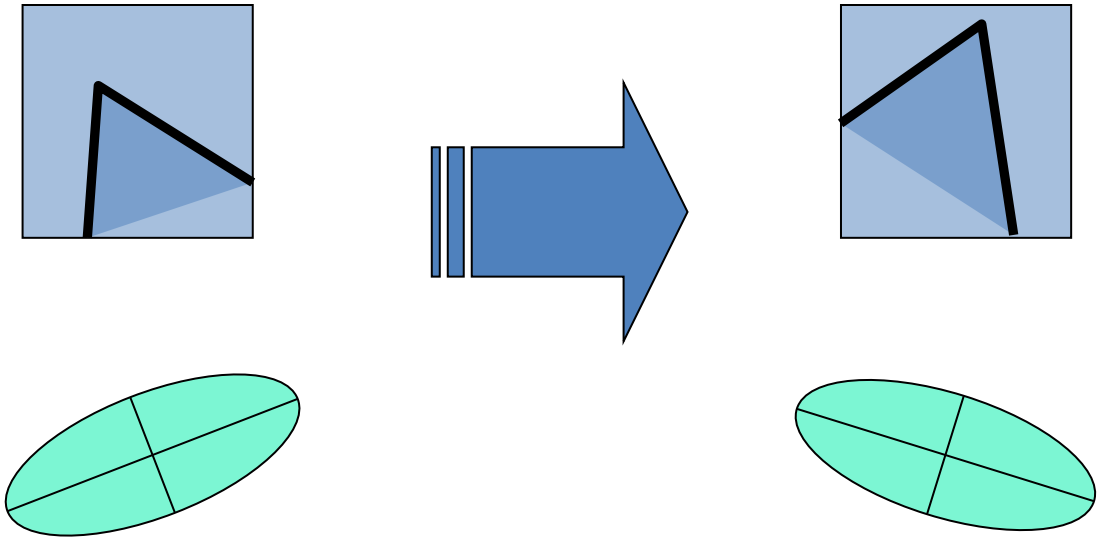
Image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

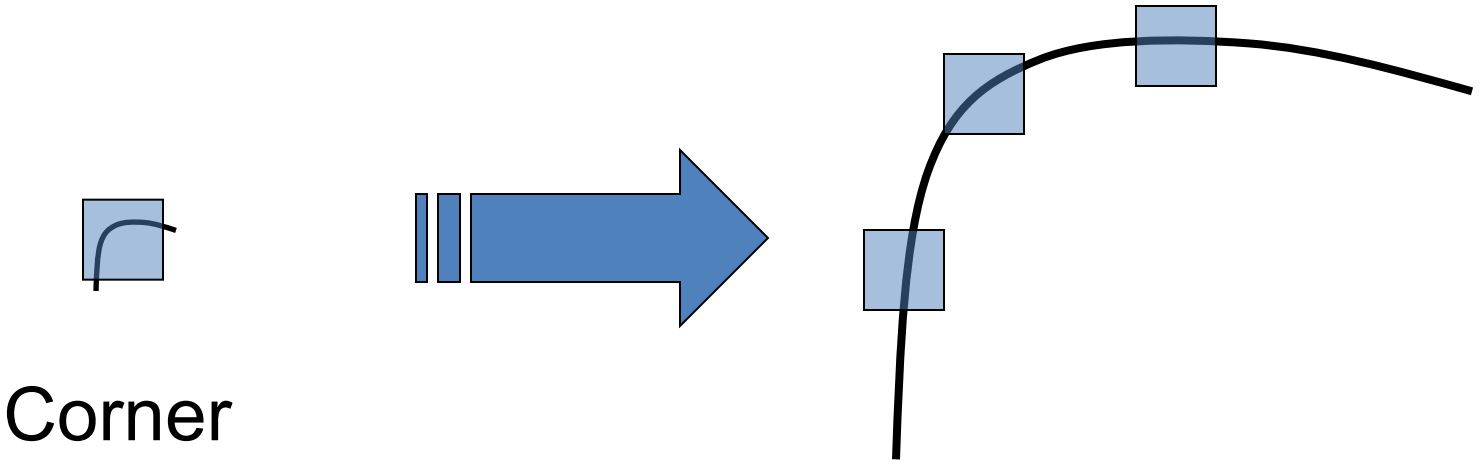
Image rotation



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

Corner location is covariant w.r.t. rotation

Scaling



Corner

All points will be classified as edges

Corner location is not covariant to scaling!

Lab 7: Image Filtering and Corner Detection

(Left: originla 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

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

Energy for pixel I

Energy-based pixel removal

Crop

Remove the lowest-energy pixels per row

Remove the lowest-energy columns

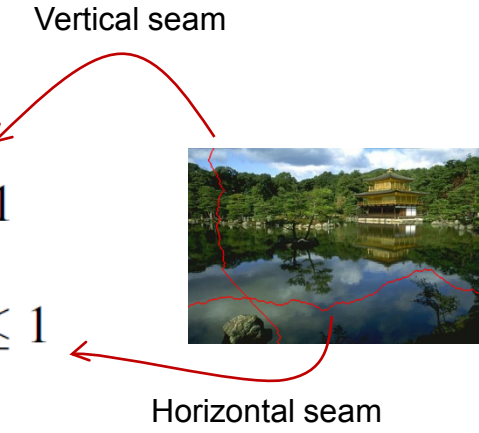
Remove the lowest-energy seams

Seam carving

- Define seams

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

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



- 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

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

Extenstion

- Seam insertion

- Object removal