CMT107 Visual Computing

VII.1 Image Processing in Java

Xianfang Sun

School of Computer Science & Informatics
Cardiff University

Overview

- Images in Java
- Read (Load) an Image
- Draw an Image
- Process an Image
- Write (Save) an Image

Images in Java

- An image is typically a rectangular two-dimensional array of pixels
  - each pixel represents the colour at that position
  - dimensions represent the horizontal extent (width) and vertical extent (height) of the image as it is displayed
- Most important image class in Java 2D API
  - java.awt.image.BufferedImage
- Image programming Tasks:
  - Load an external image file
  - Draw an image onto a drawing surface
  - Manipulate the pixels of an image
  - Save the contents of an image to an external image file

Read (Load) an Image

- Use javax.imageio package

```
BufferedImage img = null;
try {
    img = ImageIO.read(new File("Daffodil.jpg"));
} catch (IOException e) {
}
```

Draw An Image

- Use Graphics Function

```
boolean Graphics.drawImage(Image img, int x, int y,
                           ImageObserver observer);
```

- Example

```
public void paint(Graphics g) {
    g.drawImage(img, 0, 0, null);
}
```

Process an Image

- The width and height of the image can be obtained by

```
width = img.getWidth();
height = img.getHeight();
```

- The pixel colour at (x, y) can be retrieved and set by

```
Color pixel = new Color(img.getRGB(x, y));
img.setRGB(x, y, pixel.getRGB());
```

- Example: convert a colour image to a grayscale image

```
for (int y = 0; y < height; y++)
    for (int x = 0; x < width; x++) {
        Color pixel = new Color(in.getRGB(x, y));
        int r = pixel.getRed();
        int g = pixel.getGreen();
        int b = pixel.getBlue();
         r = g = b = (int) (0.299*r + 0.587*g + 0.114*b); //grayscale
        out.setRGB(x, y, (new Color(r, g, b)).getRGB());
    }
}
```

Write (Save) an Image

- Use javax.imageio package

```
BufferedImage out = getMyImage(); //Retrieve image
try {
   ImageIO.write(out, "jpg", new File("DaffodilG.jpg"));
} catch (IOException ex) {
}
```

Summary

- What is an image?
- What is a pixel?
- How to load and save an image?
- How to draw an image?
- How to access and set the pixels of an image?

CMT107 Visual Computing

VII.2 Image Filtering

Xianfang Sun

School of Computer Science & Informatics
Cardiff University

Overview

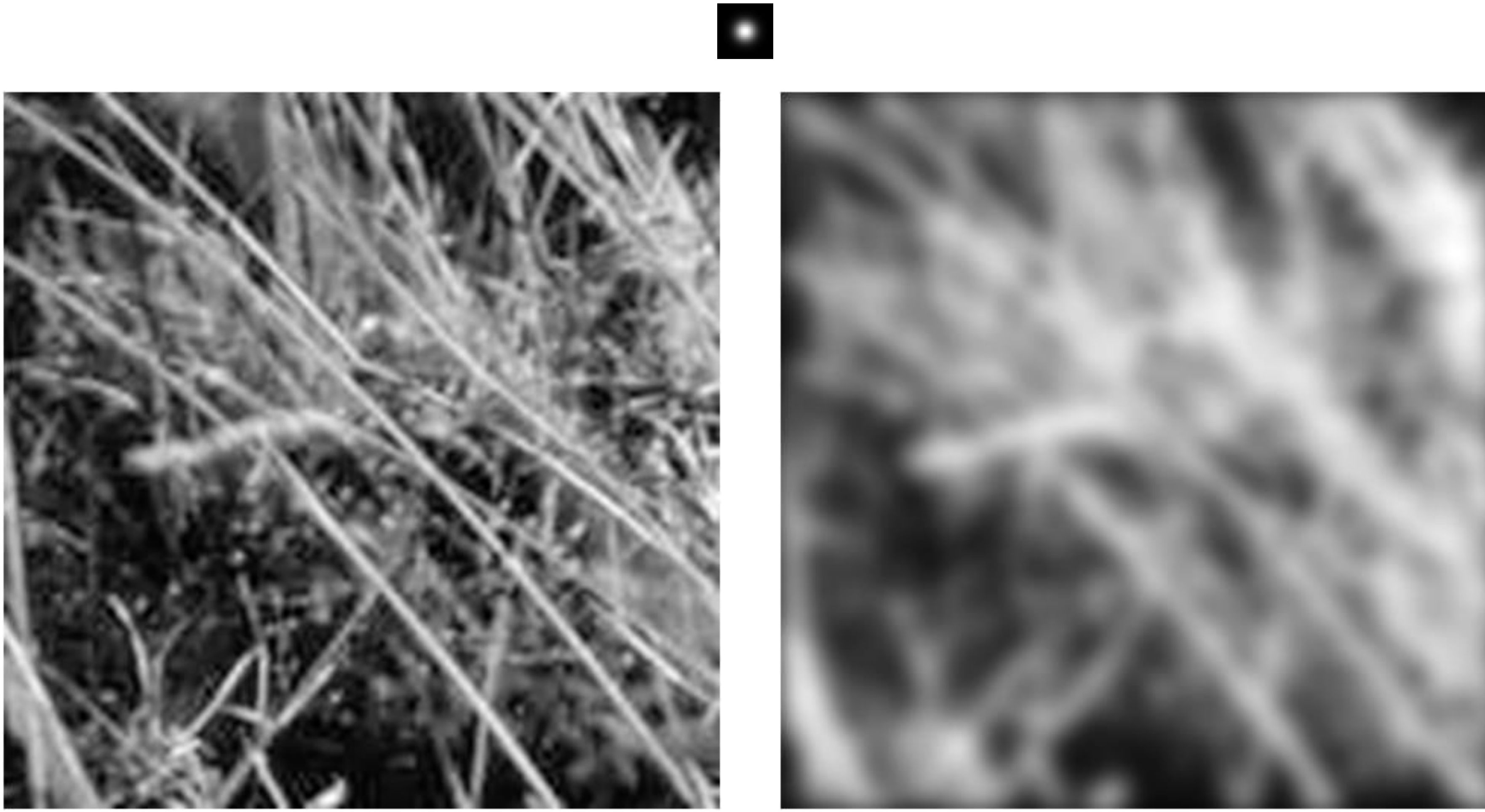
- Linear filtering
- Convolution
- Box Filtering
- Gaussian Filtering
- Separable Kernel
- Median Filter
- Sharpening

Acknowledgement
The majority of the slides in this section are from Svetlana Lazebnik at University of Illinois at Urbana-Champaign

Image Filtering

- Filtering is a technique for modifying or enhancing an image.
  - Emphasise certain features or remove other features
- Filtering is a neighbourhood operation
  - The output value of any given pixel is determined by the values of the pixels in the neighbourhood of the corresponding input pixel
- Linear filtering is filtering in which the value of an output pixel is a linear combination (weighted average) of the values of the pixels in the input pixel's neighbourhood
  - Linear filtering can be represented by convolution

Linear Filtering



Motivation: Image Denoising

- How can we reduce noise in a photograph?

Moving average

- Let's replace each pixel with a *weighted* average of its neighborhood
- The weights are called the *filter kernel*
- What are the weights for the average of a 3x3 neighbourhood?

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

"box filter"

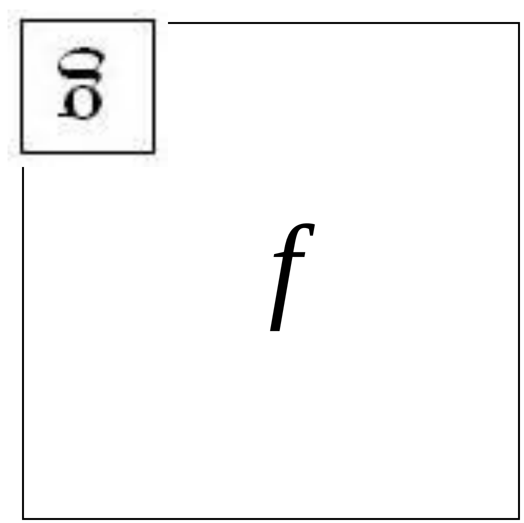
Source: D. Lowe

Convolution

- Let *f* be the image and *g* be the kernel. The output of convolving *f* with *g* is denoted *f* * *g*.

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

Convention:
kernel is "flipped"



Source: F. Durand

Key properties

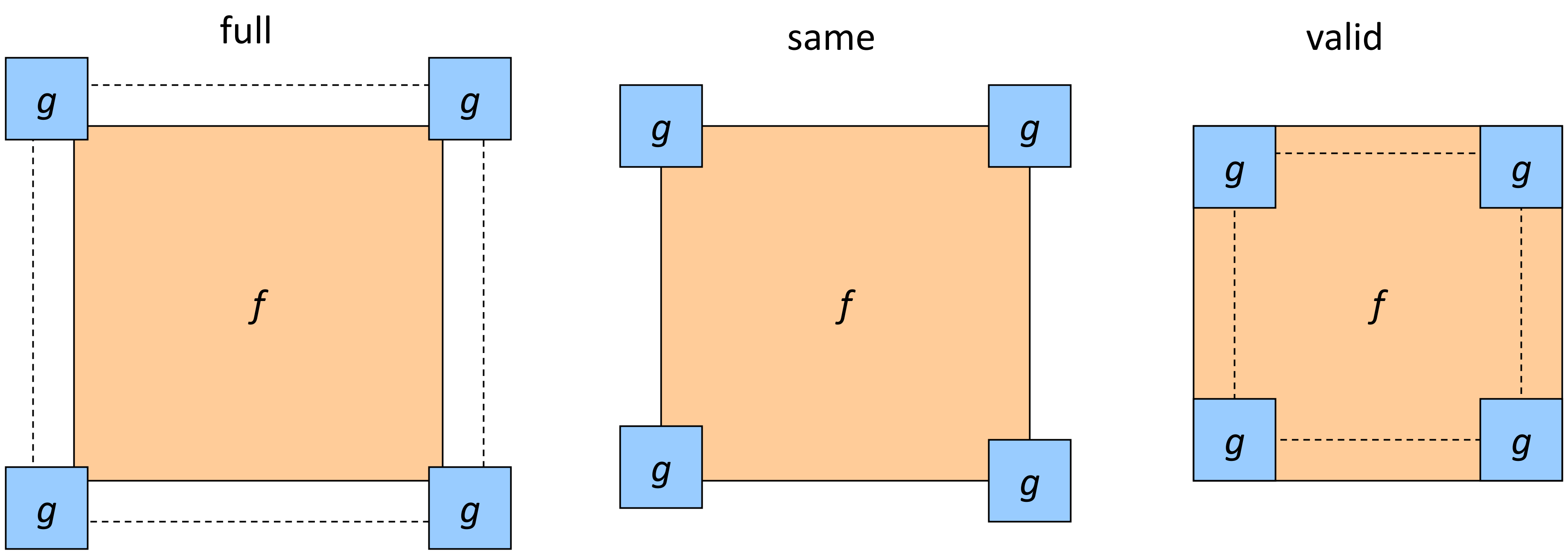
- Linearity: filter($f_1 + f_2$) = filter(f_1) + filter(f_2)
- Shift invariance: same behavior regardless of pixel location: filter(shift(f)) = shift(filter(f))
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

More Properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
- 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: $a * (b_1 * b_2 * b_3)$
- Distributive over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Size of the Output

- ‘full’: output size is the sum of sizes of f and g -1
- ‘same’: output size is the same as f
- ‘valid’: output size is the difference of the sizes of f and g



Boundary Pixels

- 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

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
1	0	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
1	0	0
0	0	0

Shifted *left*
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

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

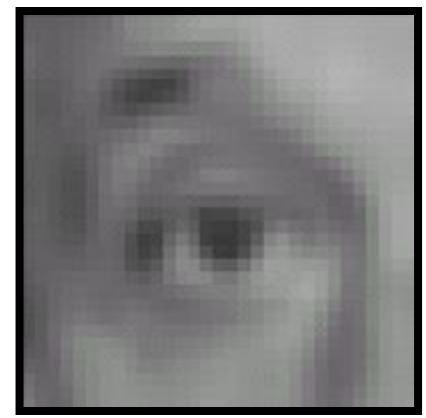
?

Source: D. Lowe

Practice with linear filters

Original

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



Blur (with a box filter)

Source: D. Lowe

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)

Source: D. Lowe

Practice with linear filters

Original

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

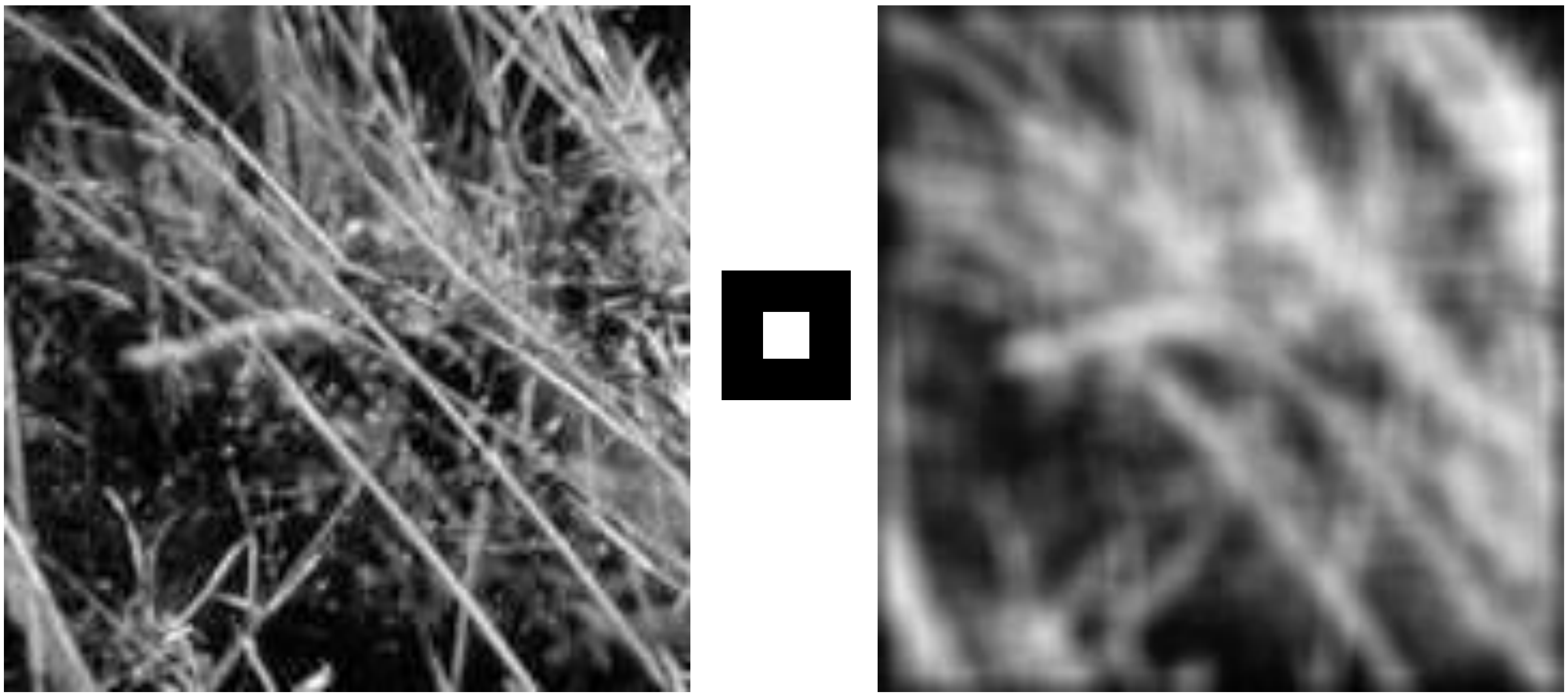
Sharpening filter

- Accentuates differences with local average

Source: D. Lowe

Smoothing with box filter revisited

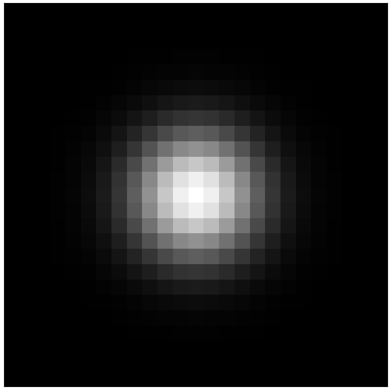
- What’s wrong with this picture?
- What’s the solution?



Source: D. Forsyth

Smoothing with box filter revisited

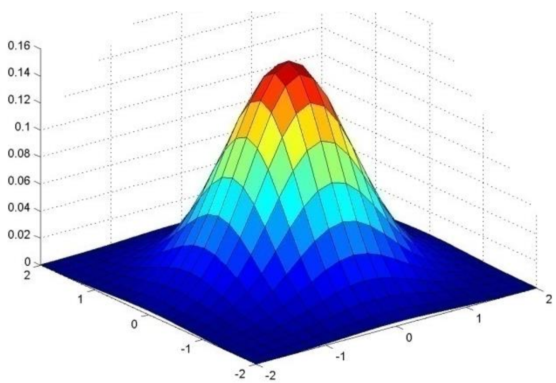
- What's wrong with this picture?
- What's the solution?
  - To eliminate edge effects, weight contribution of neighbourhood pixels according to their closeness to the center

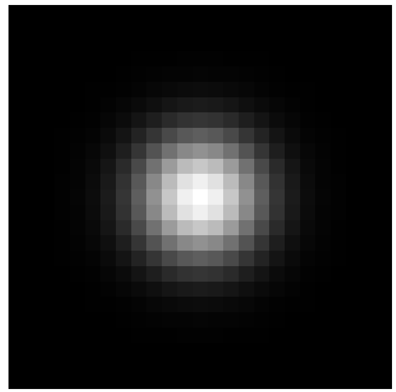


"fuzzy blob"

Gaussian Kernel

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





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

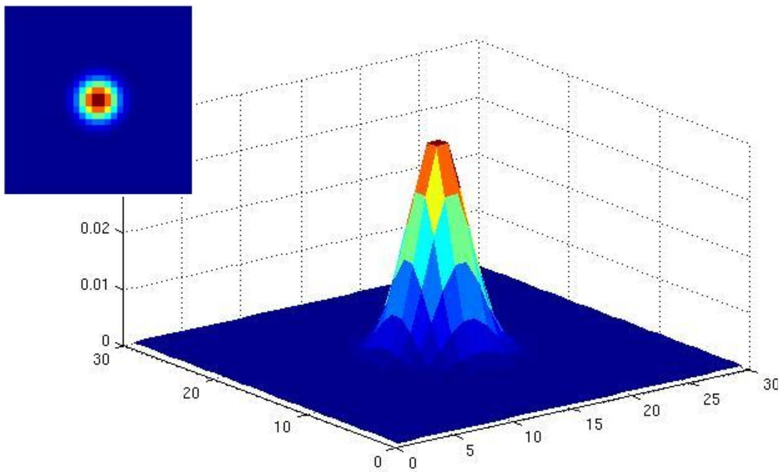
5 x 5, σ = 1

- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

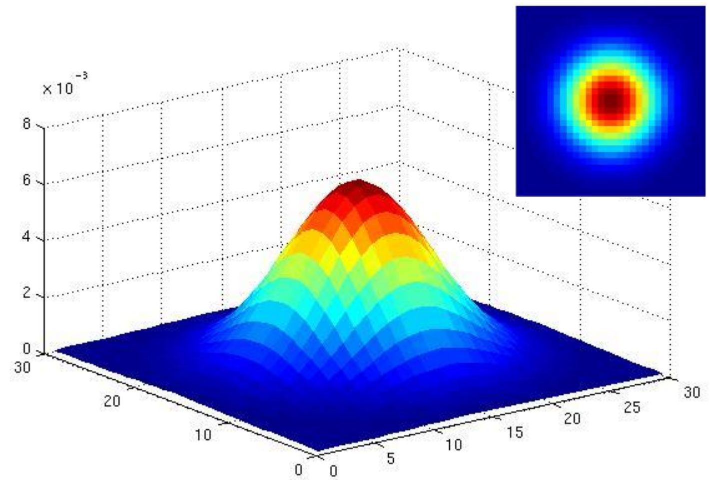
Source: C. Rasmussen

Gaussian Kernel

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



σ = 2 with 30 x 30 kernel



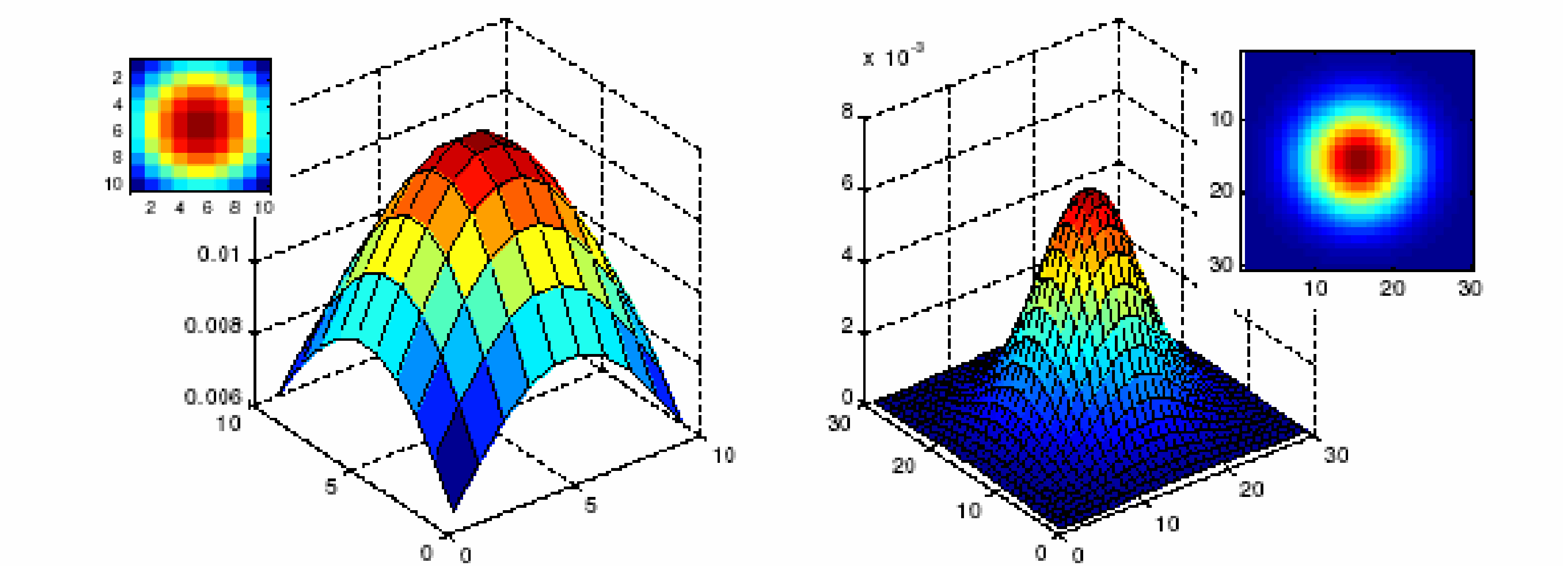
σ = 5 with 30 x 30 kernel

- Standard deviation σ: determines extent of smoothing

Source: K. Grauman

Choosing kernel width

- The Gaussian function has infinite support, but discrete filters use finite kernels



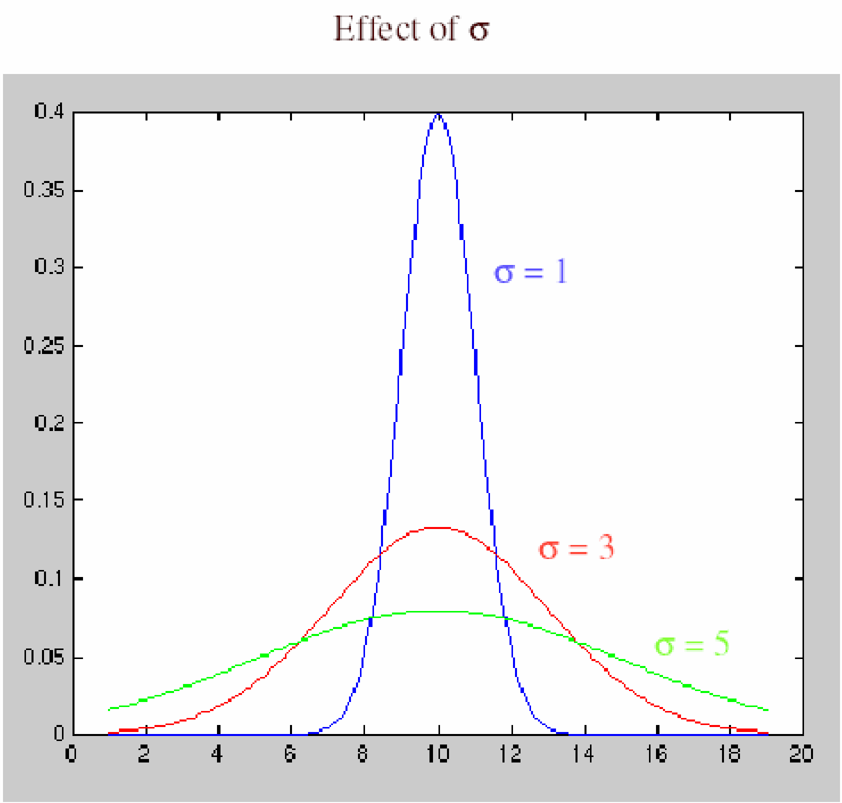
$\sigma = 5$ with 10x10 kernel

$\sigma = 5$ with 30x30 kernel

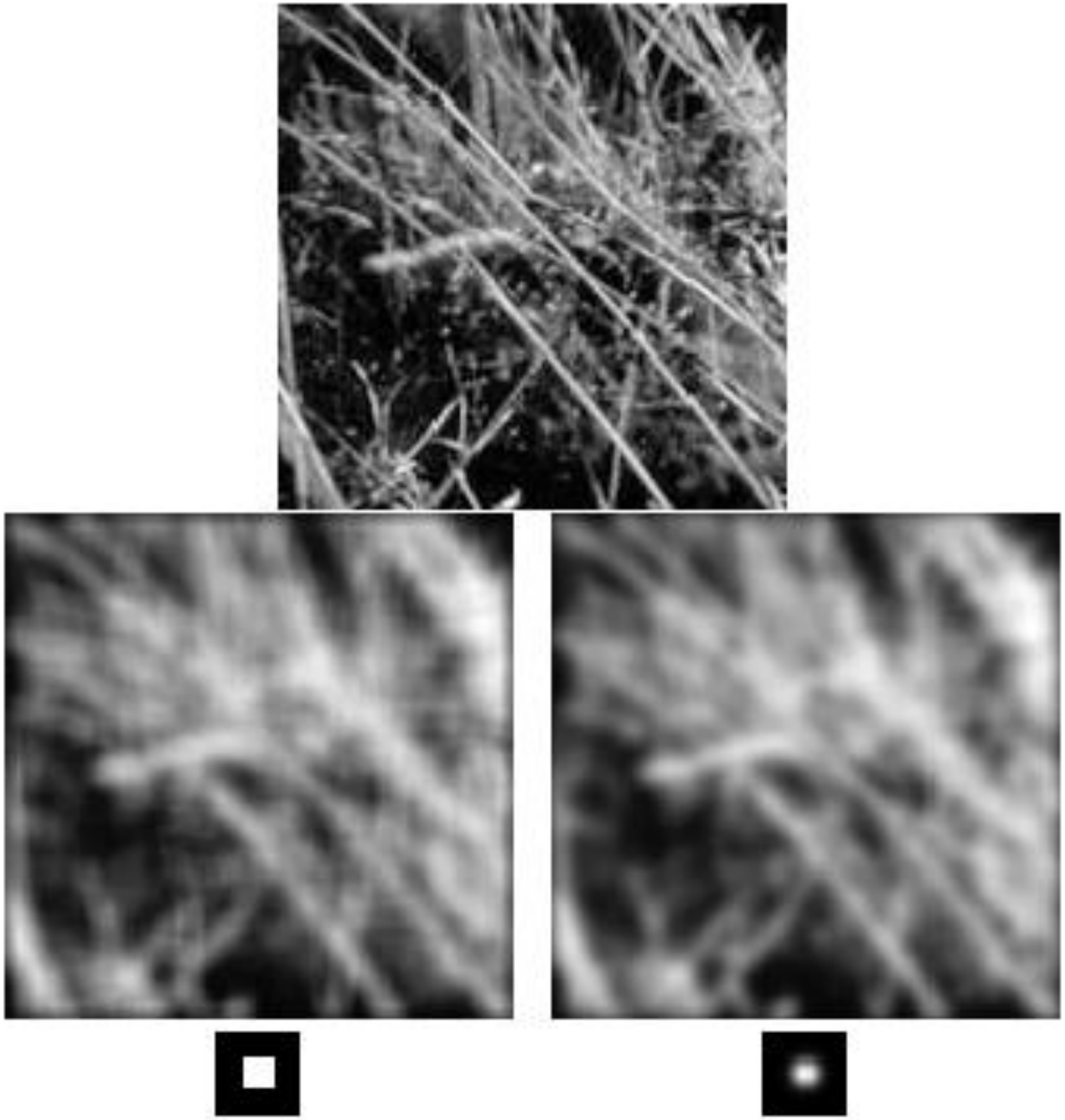
Source: K. Grauman

Choosing kernel width

- Rule of thumb: set filter half-width to about 3σ



Gaussian vs. box filtering



Gaussian filters

- Remove "high-frequency" components from the image (low-pass filter)
- Convolution with self is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *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

Source: D. Lowe

Separability example

2D convolution
(center location only)

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

The filter factors
into a product of 1D
filters:

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

Perform convolution
along rows:

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

Followed by convolution
along the remaining column:

Source: K. Grauman

Why is separability useful?

- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
  - $O(n^2 m^2)$
- What if the kernel is separable?
  - $O(n^2 m)$

Noise

Original

Salt and pepper noise

Impulse noise

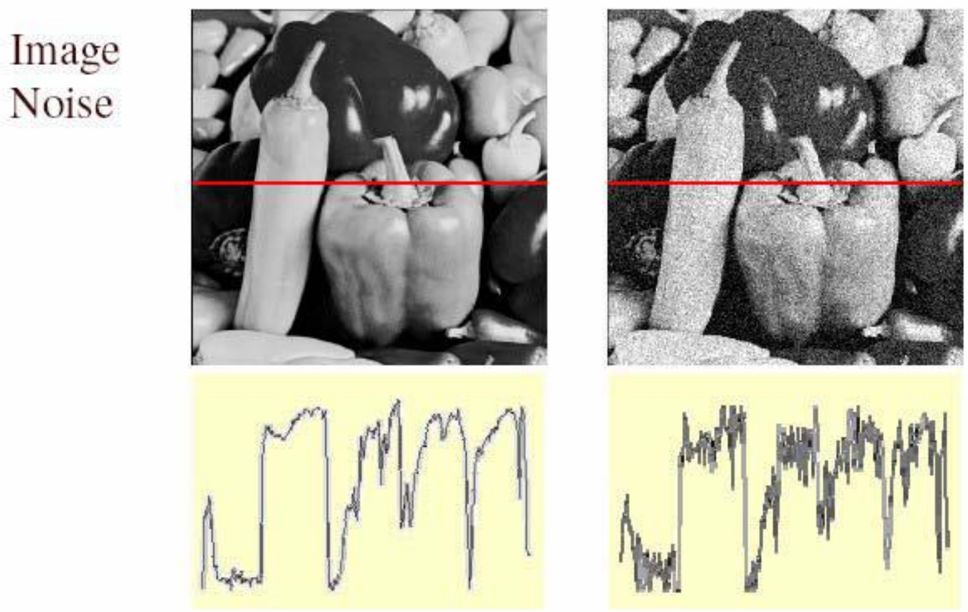
Gaussian noise

- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz

Gaussian Noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise

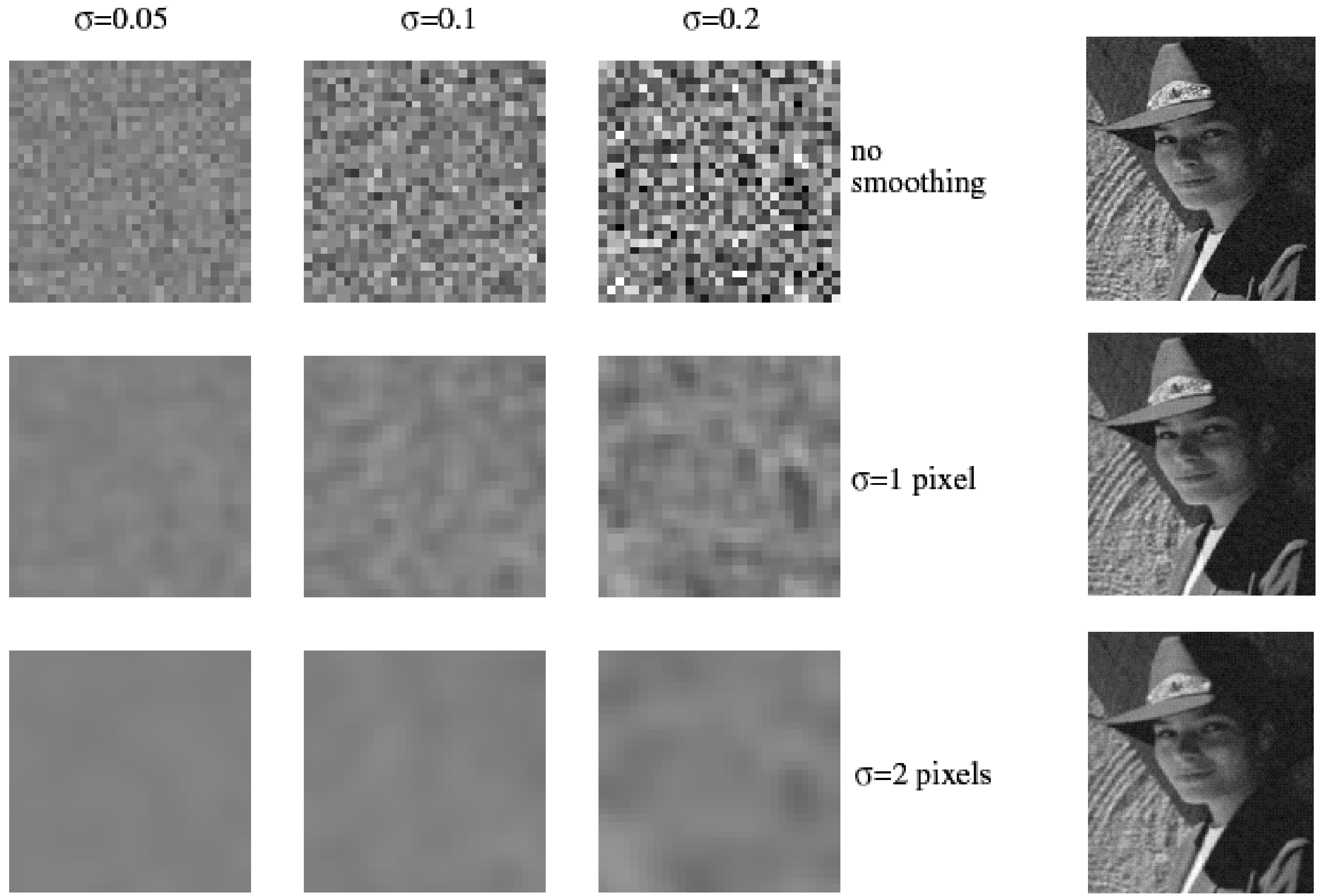


$$f(x,y) = \overbrace{\tilde{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

Source: M. Hebert

Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing salt-and-pepper noise

3x3

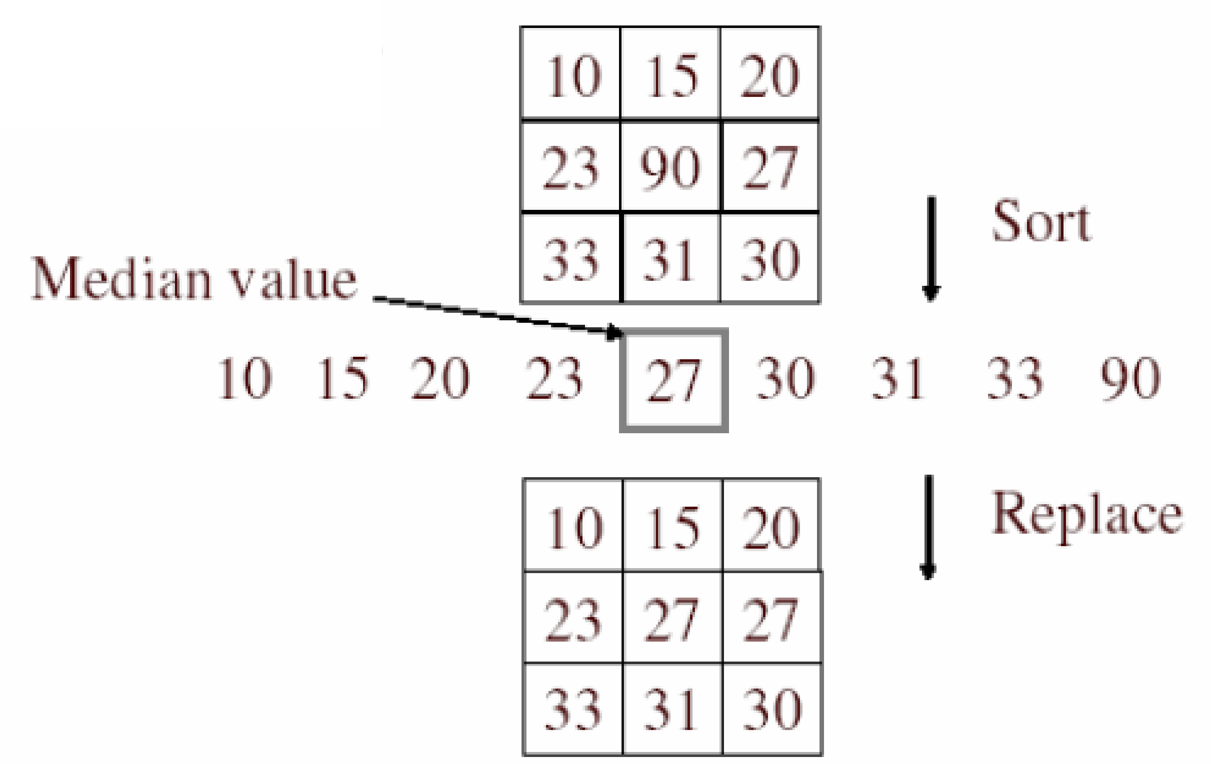
5x5

7x7

- What’s wrong with the results?

Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window

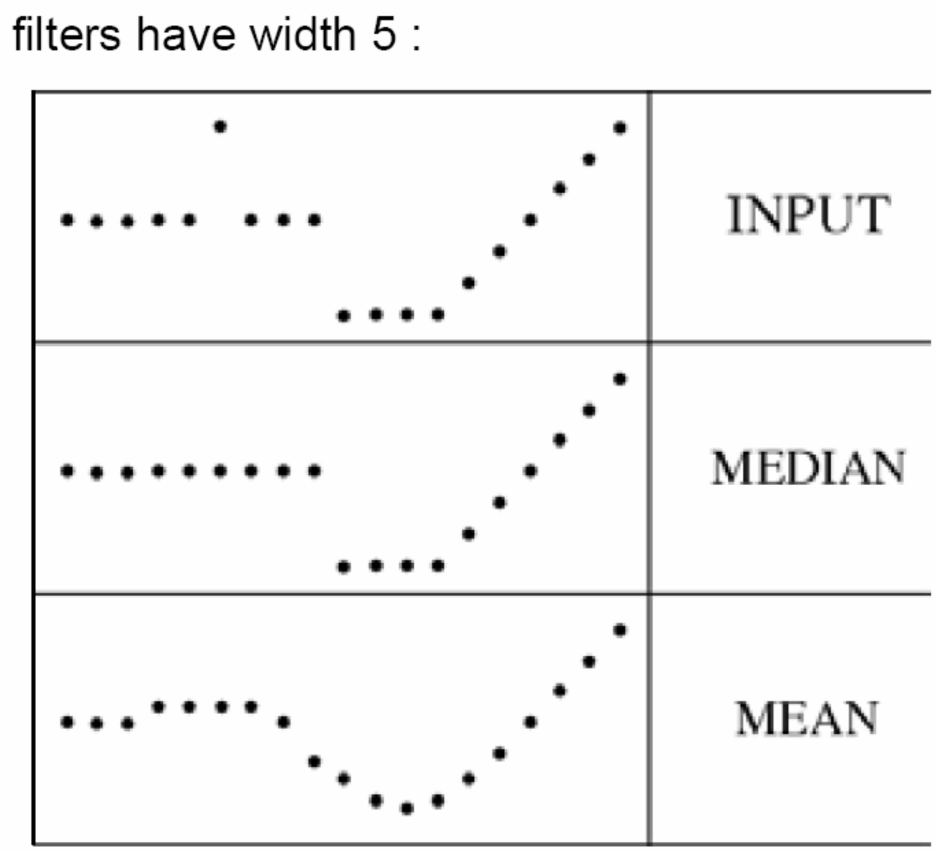


- Is median filtering linear?

Source: K. Grauman

Median filter

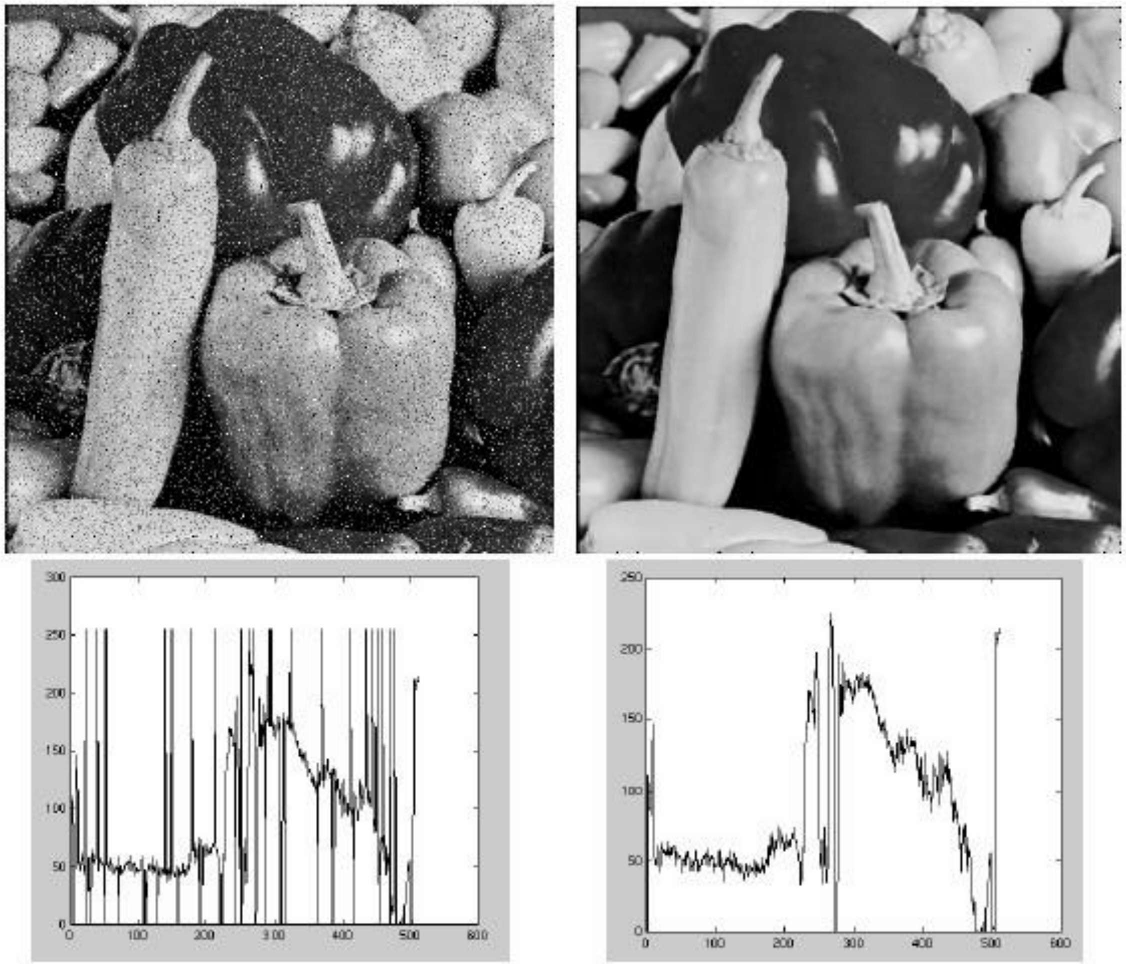
- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers



Source: K. Grauman

Median filter

Salt-and-pepper noise Median filtered



Source: M. Hebert

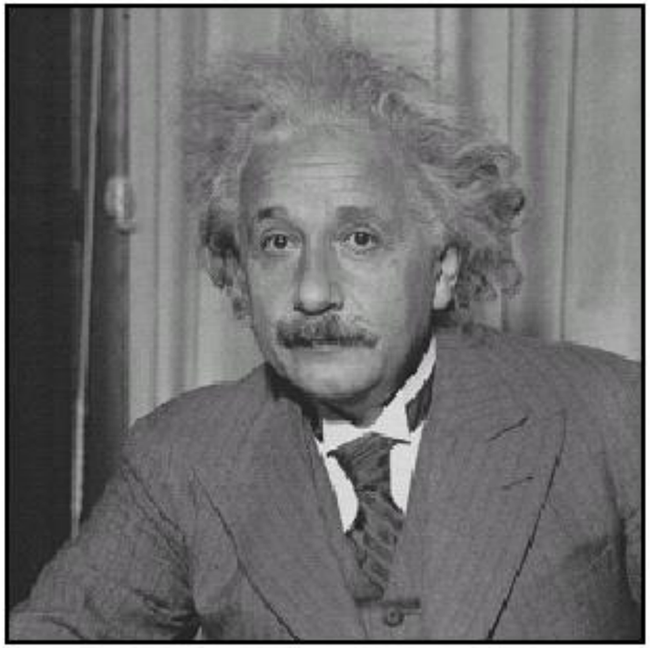
Gaussian vs. median filtering

3x3 5x5 7x7

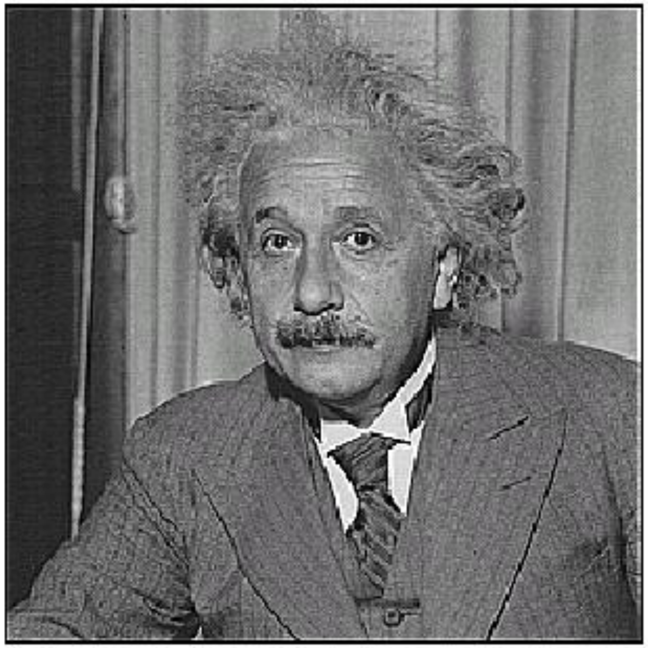
Gaussian

Median

Sharpening revisited



before



after

Source: D. Lowe

Sharpening revisited

- What does blurring take away?

−

=

Let’s add it back:

+ α

=

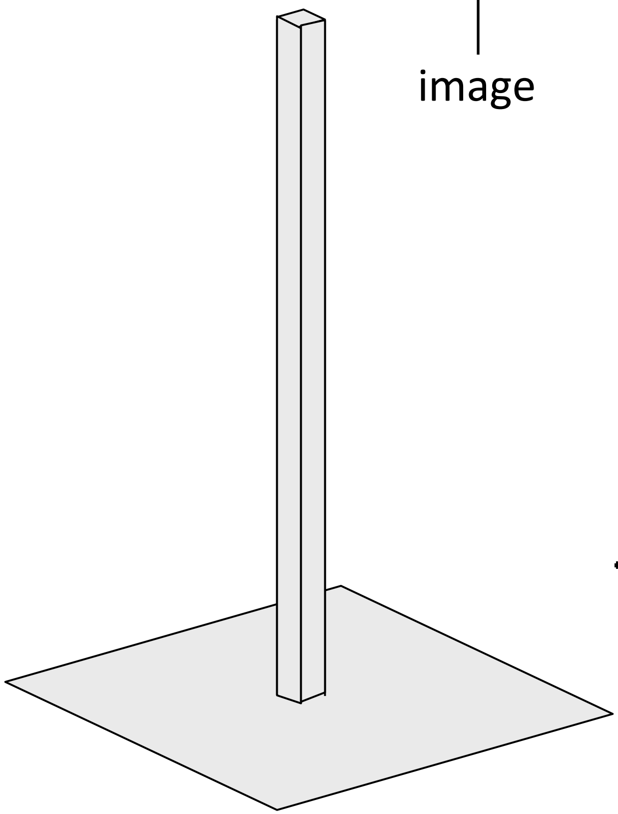
Unsharp mask filter

$$f + \alpha(f - f * g) = (1+\alpha) f - \alpha f * g = f * ((1+\alpha)e - g)$$

image

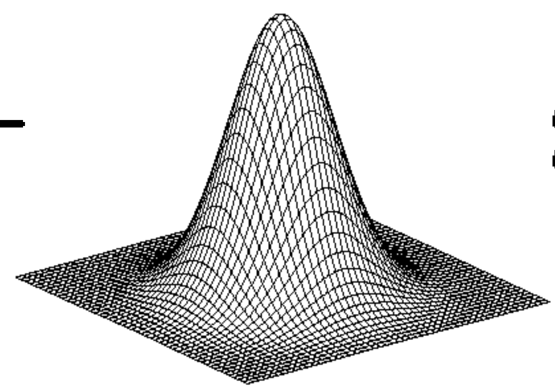
blurred image

unit impulse (identity)



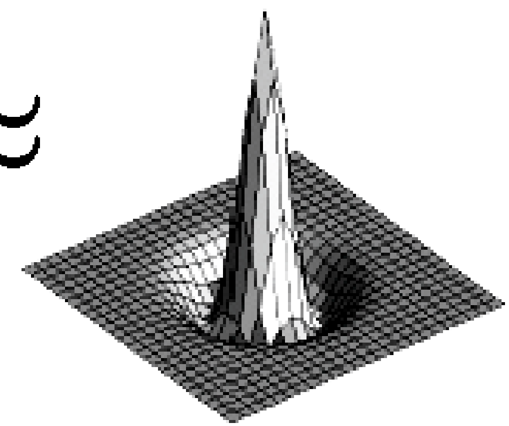
unit impulse

$-$



Gaussian

$\approx$



Laplacian of Gaussian

Image Filtering with Java

- Use filter() function in BufferedImageOp
- Implement filtering without using filter() function

Use filter() Function

- Define a filter kernel

```
float[] km = {          // low-pass filter kernel
    0.1f, 0.1f, 0.1f,
    0.1f, 0.2f, 0.1f,
    0.1f, 0.1f, 0.1f
};
Kernel kernel = new Kernel(3, 3, km);
```

- Define an Operator

```
BufferedImageOp op = null;
op = new ConvolveOp(kernel, ConvolveOp.EDGE_NO_OP, null);
```

  - ConvolveOp(Kernel kernel, int edgeCondition, RenderingHints hints)
    - edgeCondition: ConvolveOp.EDGE_NO_OP or
      ConvolveOp.EDGE_ZERO_FILL

- Call the filter() function

```
out = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
op.filter(in, out);
```

Not Use filter() Function

- Define a filter kernel matrix

```
float[] km = {           // low-pass filter kernel
    0.1f, 0.1f, 0.1f,   // Suppose the matrix has been flipped
    0.1f, 0.2f, 0.1f,
    0.1f, 0.1f, 0.1f
};
```

- Calculate convolution on each pixel

```
int[] rArray = new int[width*height];  //
for each pixel {
    get the neighbourhood colours of the pixel
    calculate the colour according to the convolution formula
    set the pixel colour in the output image
}
```

  - More details in Lab session 6

Summary

- What is filtering? What is linear filtering?
- What is convolution?
- How to do sharpening of image?
- What is box filtering, Gaussian filtering, and median filtering?
- What is separable kernel? Why use separable kernel?

CMT107 Visual Computing

VII.3 Corner Extraction

Xianfang Sun

School of Computer Science & Informatics
Cardiff University

Overview

- Feature Extraction
  - Characteristics of Good Features
  - Applications
- Corner Detection
  - Basic Idea
  - Mathematics
- Harris Detector
- Invariance and Covariance

Acknowledgement

The majority of the slides in this section are from Svetlana Lazebnik at University of Illinois at Urbana-Champaign

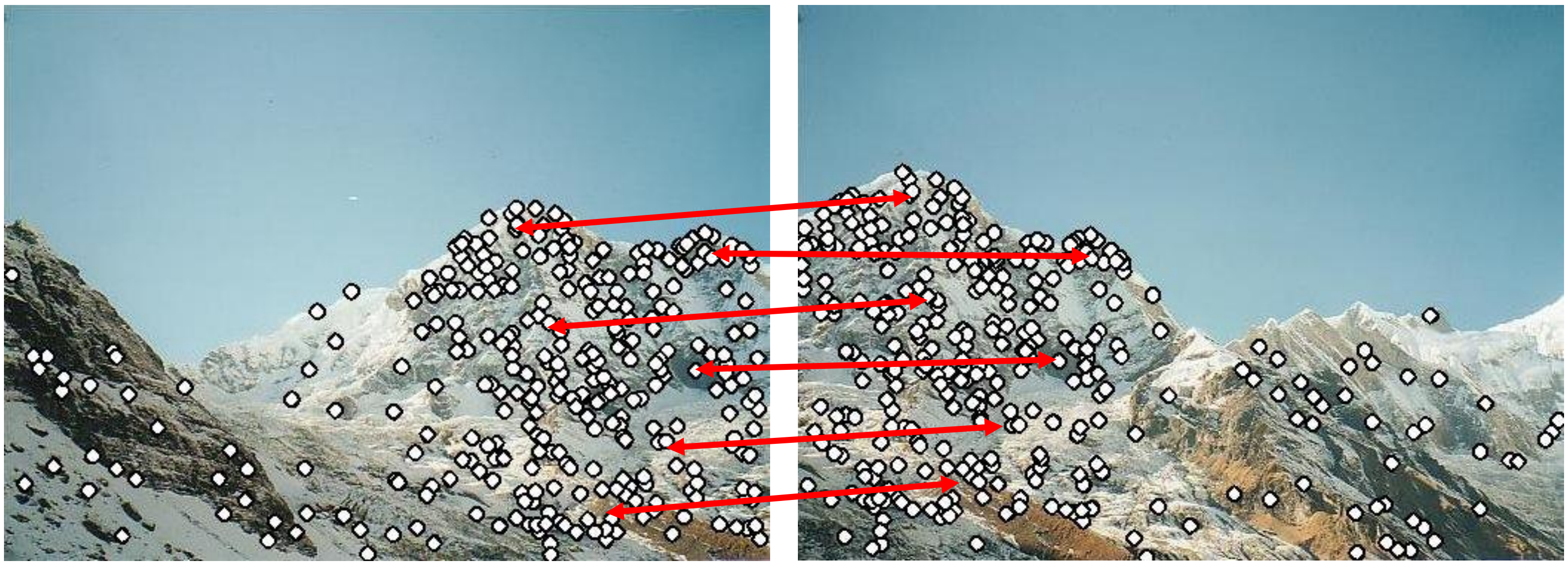
Feature Extraction: Corners

Why Extract Features?

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

Why Extract Features?

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



Step 1: extract features

Step 2: match features

Why Extract Features?

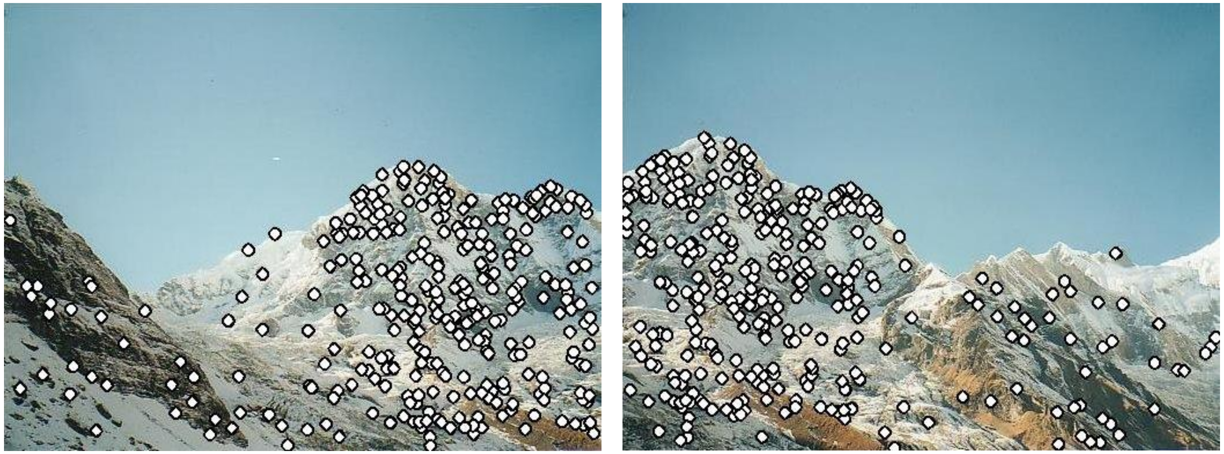
- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features

Step 2: match features

Step 3: align images

Characteristics of Good Features

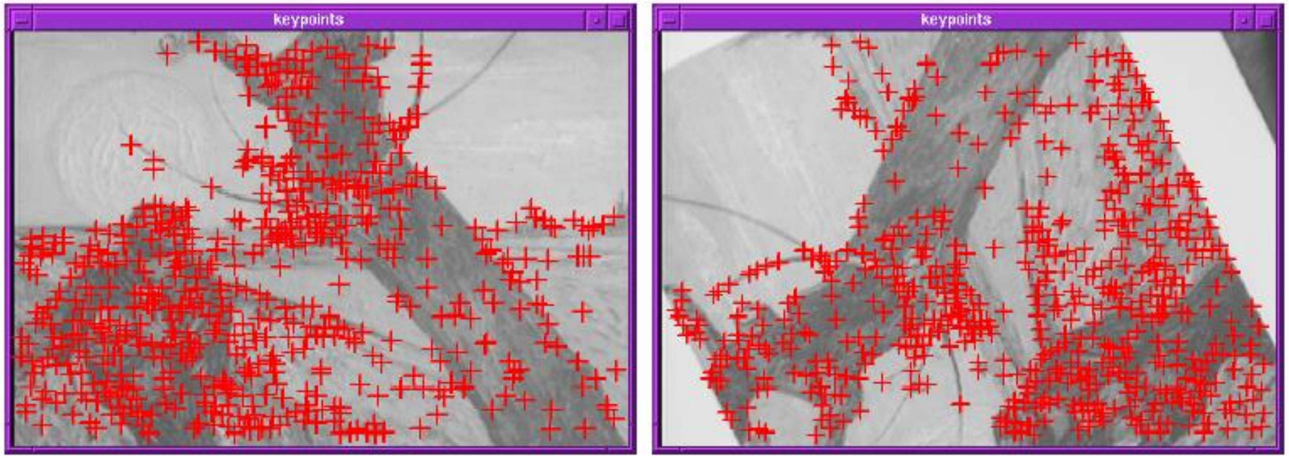


- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Applications

- Feature points are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition

Finding Corners

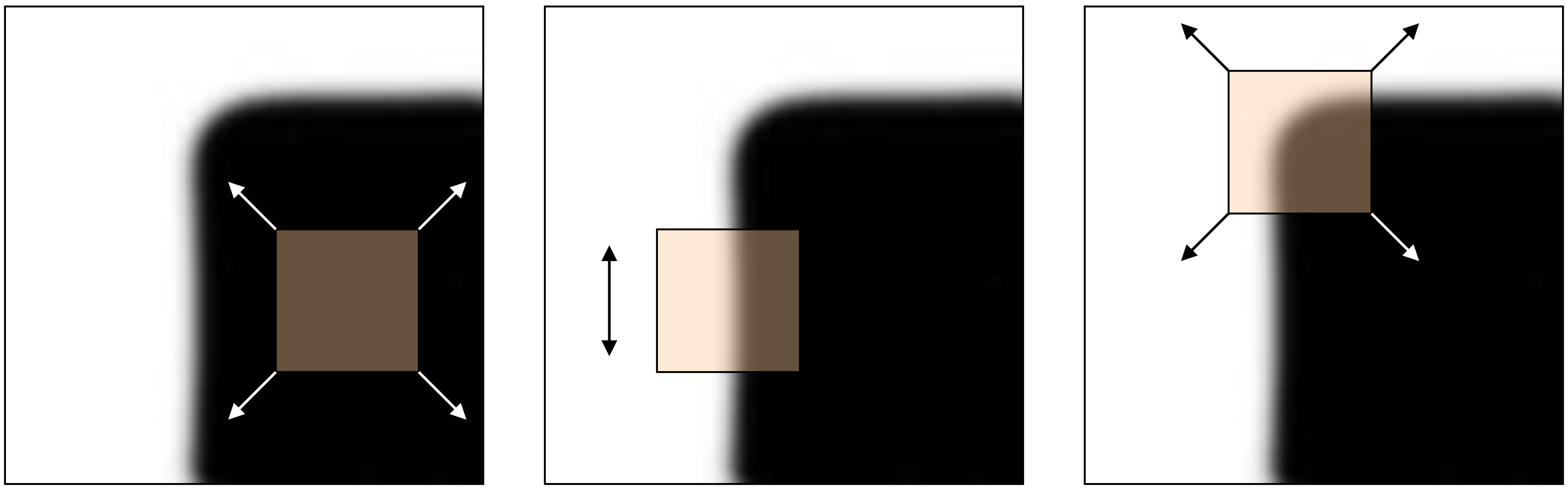


- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

C.Harris and M.Stephens. "A Combined Corner and Edge Detector.“ *Proceedings of the 4th Alvey Vision Conference, 1988*: pages 147--151.

Corner Detection: 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

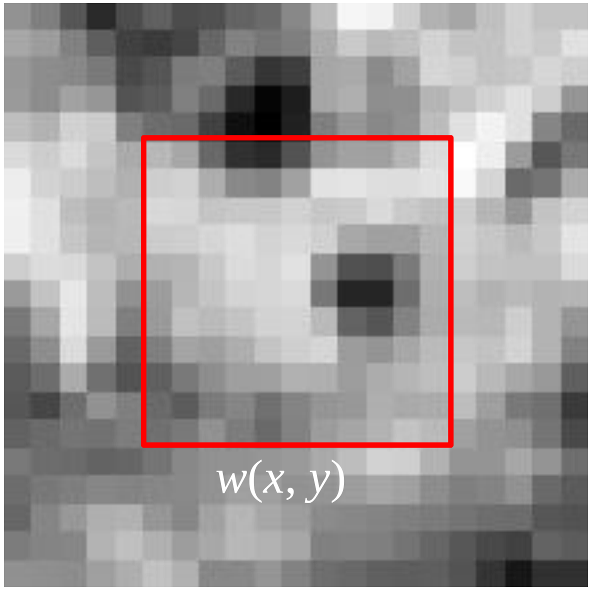
Source: A. Efros

Corner Detection: Mathematics

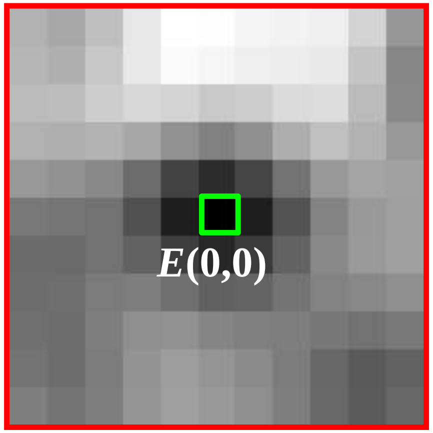
Change in appearance of window *w*(*x,y*)
for the shift [*u,v*]:

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

$I(x, y)$



$E(u, v)$

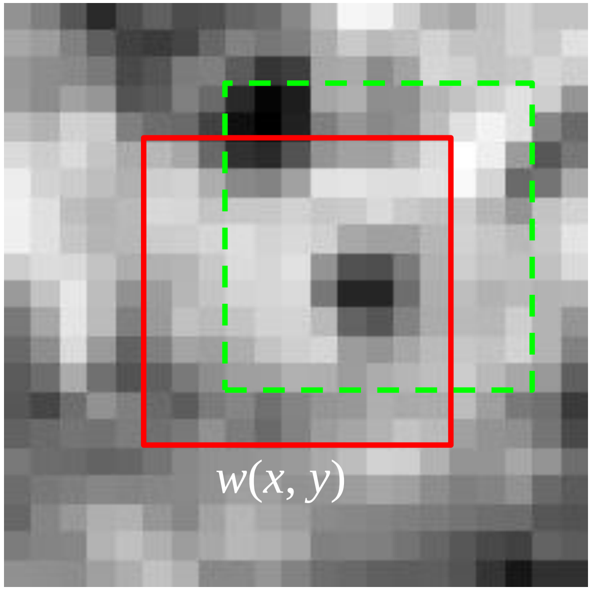


Corner Detection: Mathematics

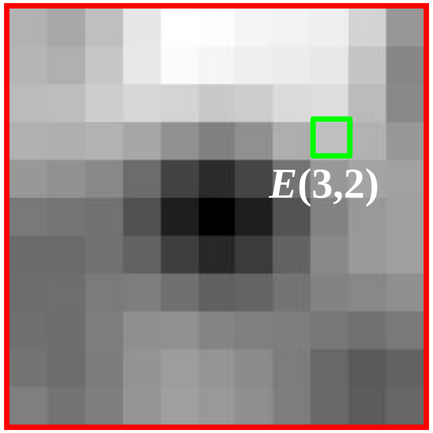
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

$I(x, y)$



$E(u, v)$



Corner Detection: Mathematics

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

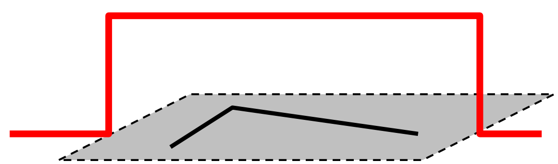
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

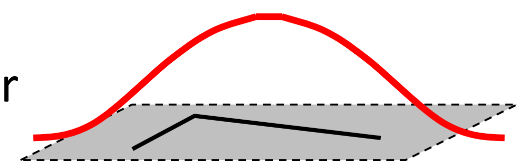
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

Source: R. Szeliski

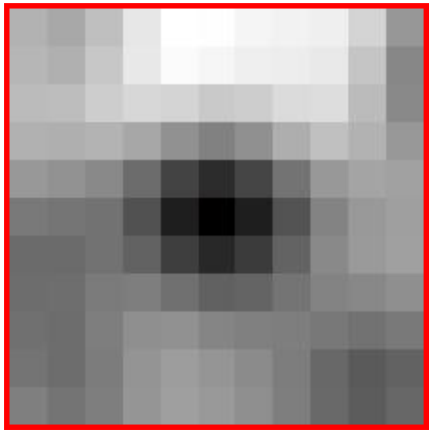
Corner Detection: Mathematics

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*:

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

Corner Detection: Mathematics

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

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

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

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

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

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

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

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

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):

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

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

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

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

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

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

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

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:

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

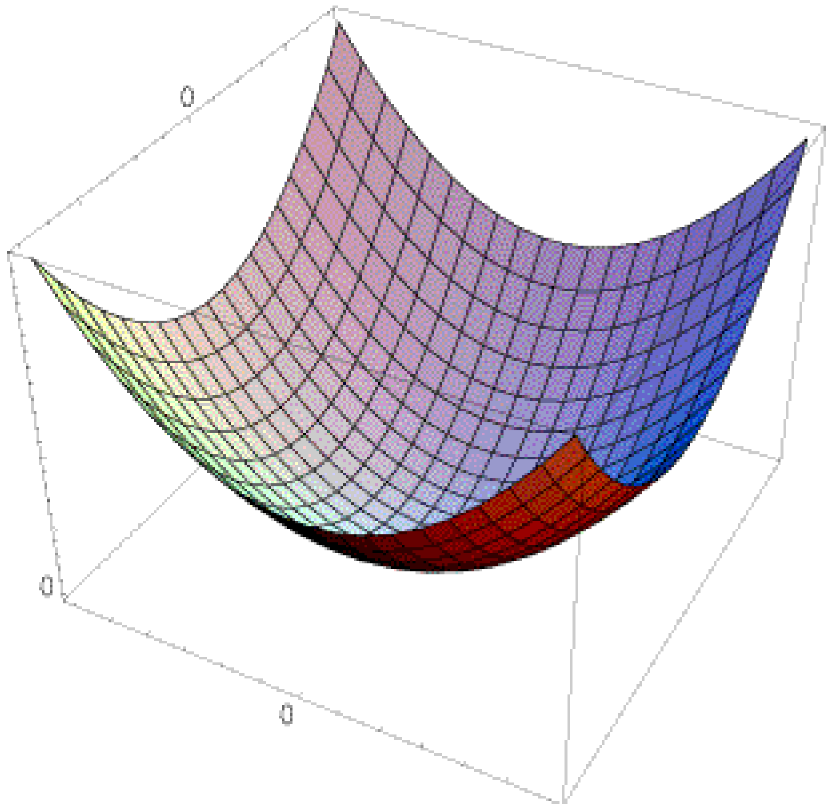
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

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

First, consider the axis-aligned case (gradients are either horizontal or vertical)

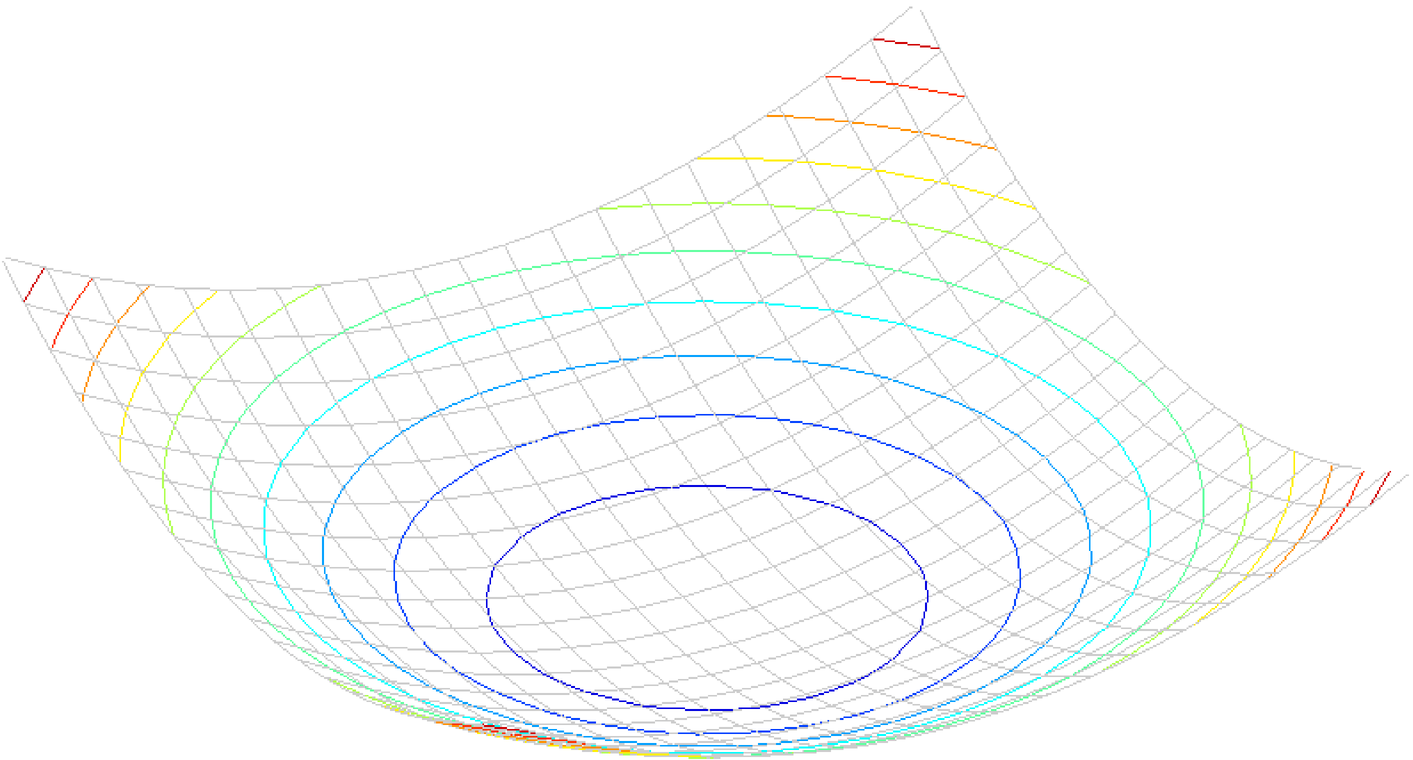
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

If either λ is close to 0, then this is not a corner, so look for locations where both are large.

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.



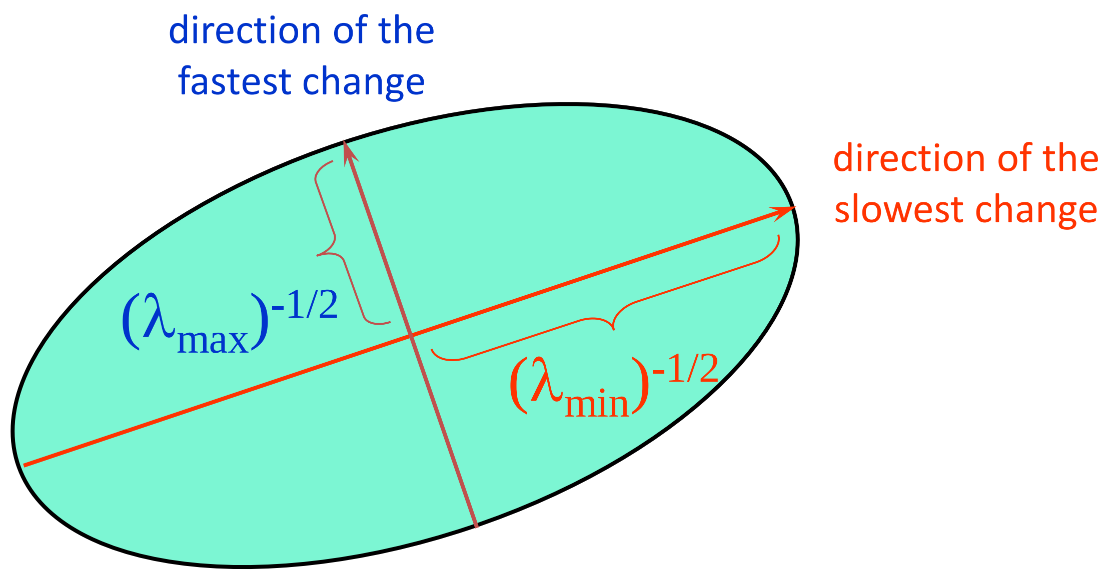
Interpreting the Second Moment Matrix

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

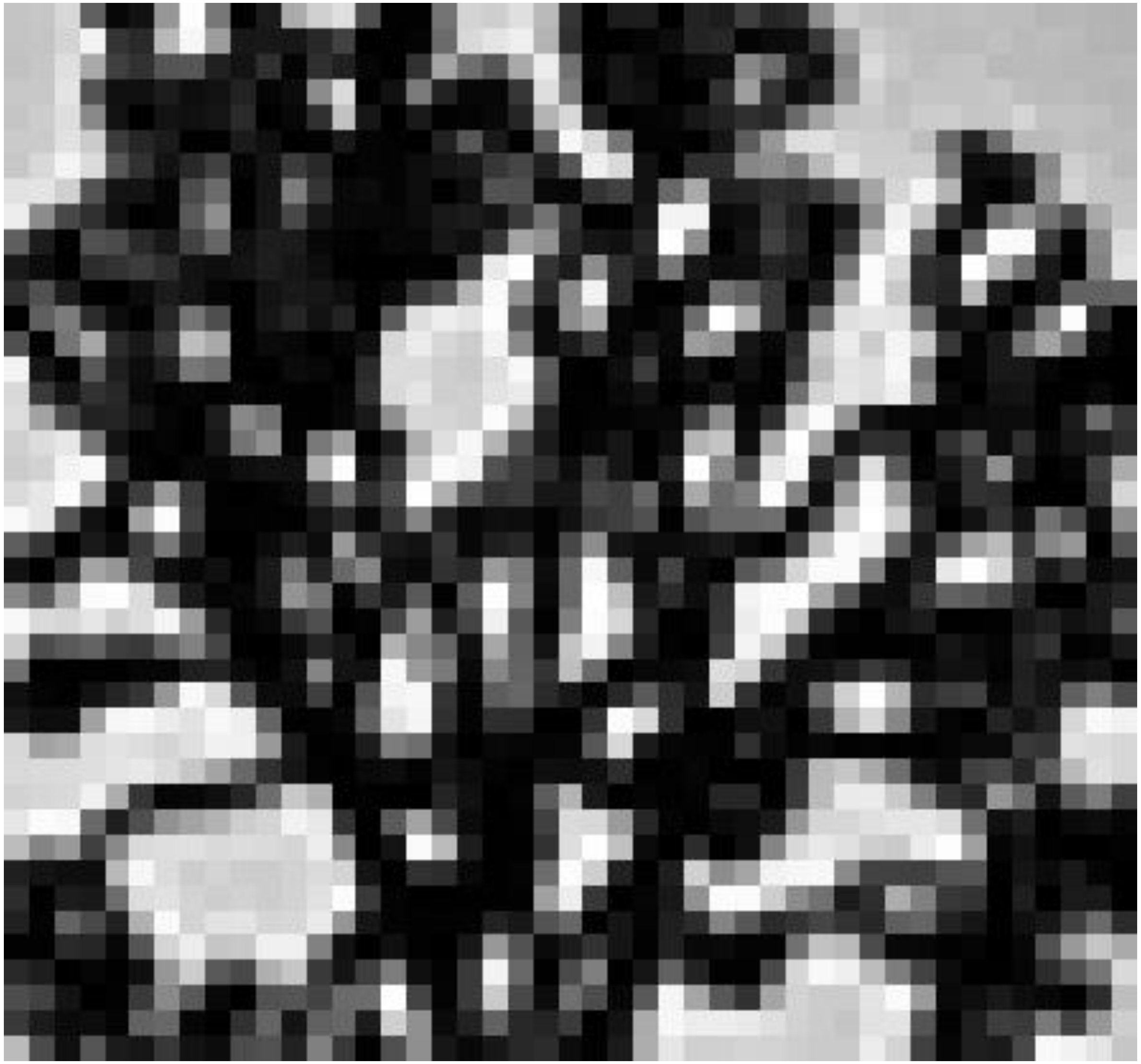
This is the equation of an ellipse.

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

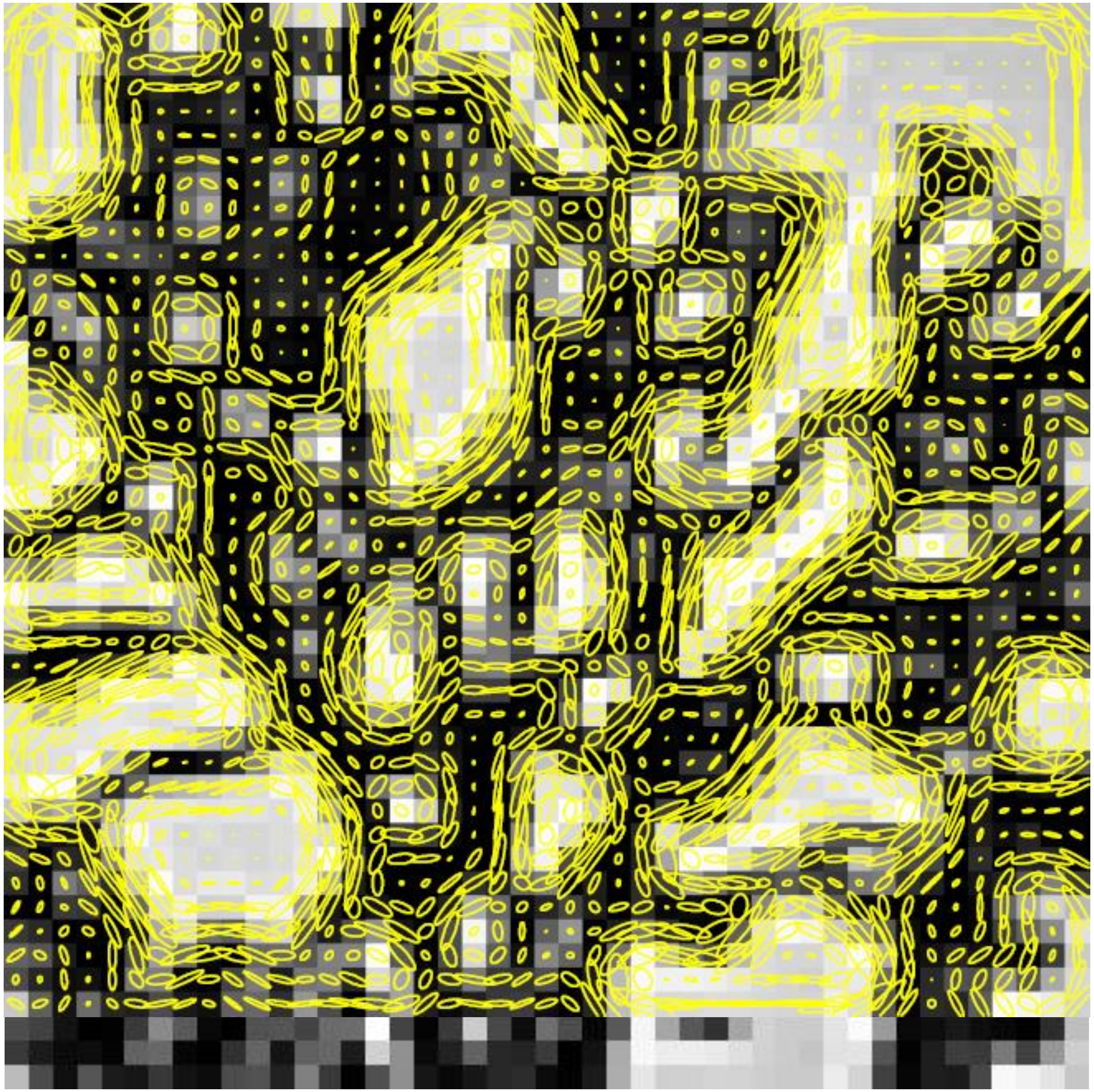
The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by *R*



Visualization of Second Moment Matrices

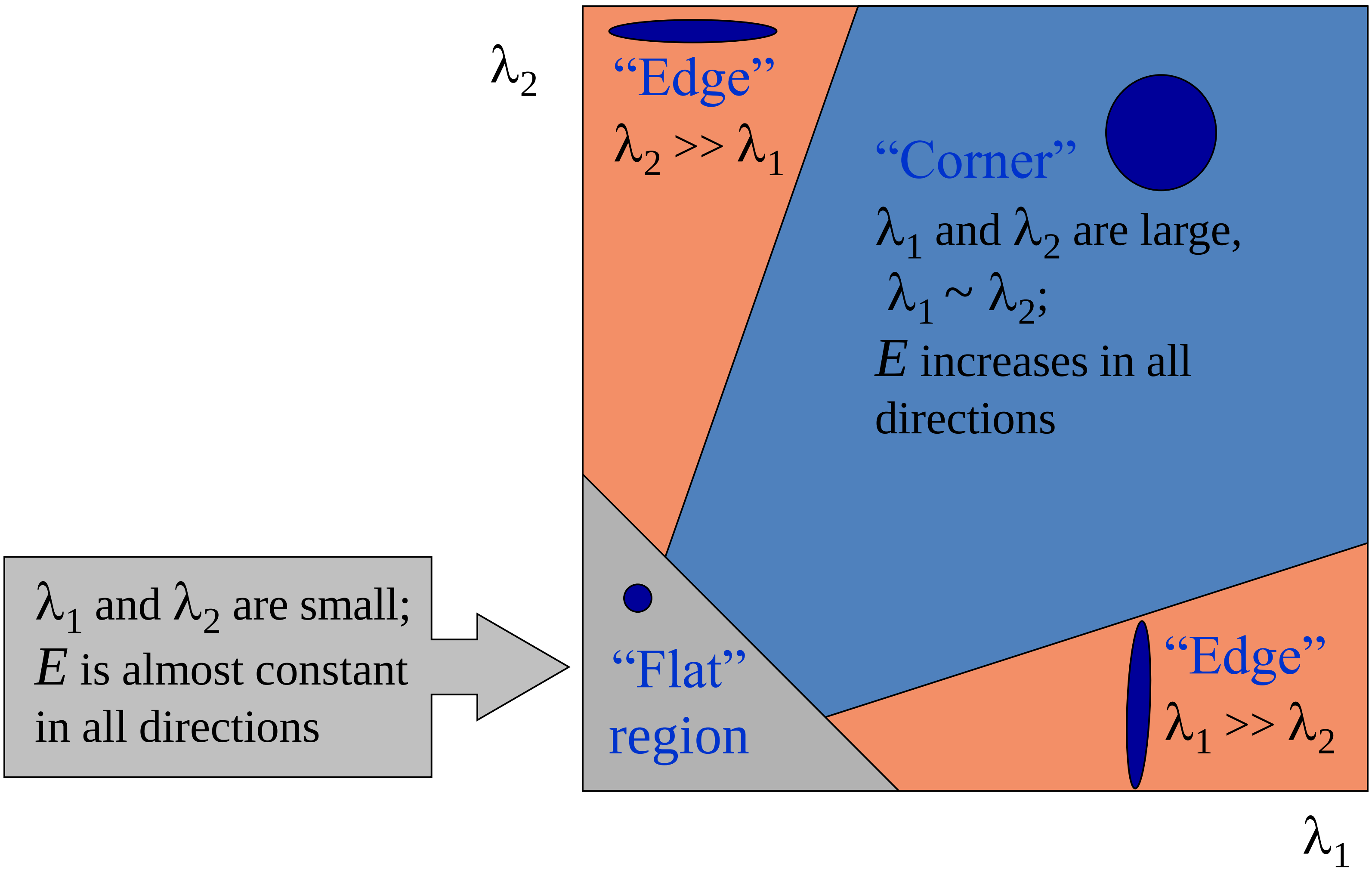


Visualization of Second Moment Matrices



Interpreting the Eigenvalues

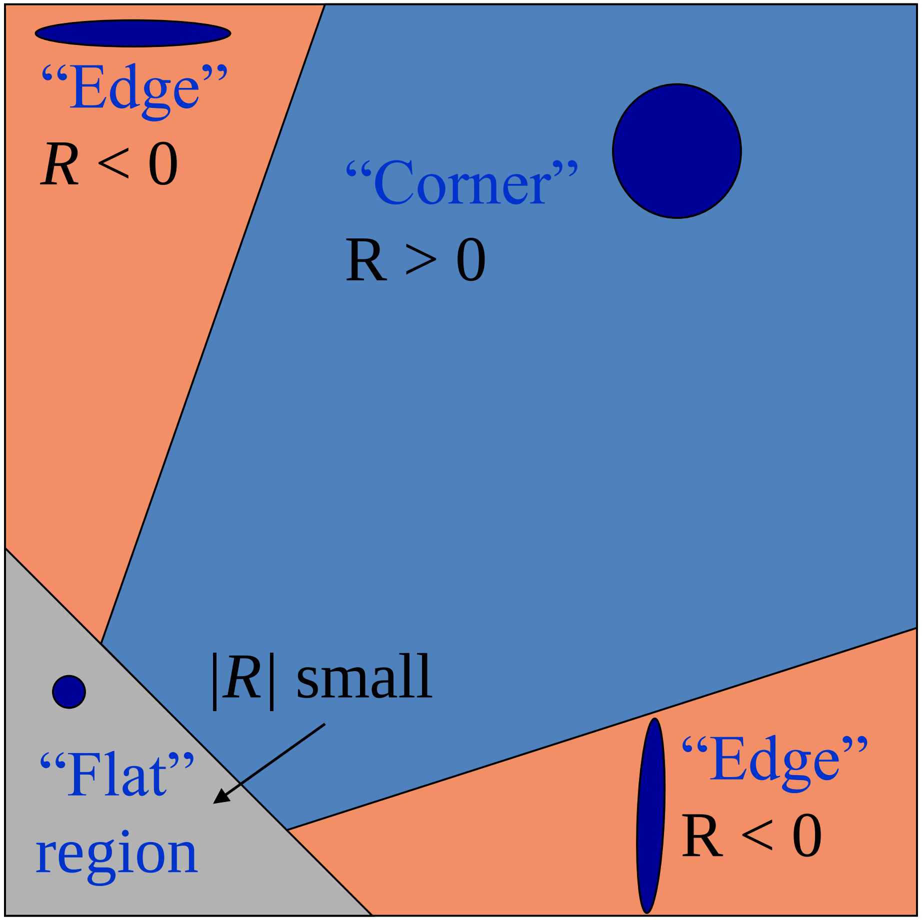
Classification of image points using eigenvalues of *M*:



Corner response function

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

α: constant (0.04 to 0.06)



Harris Detector: Steps

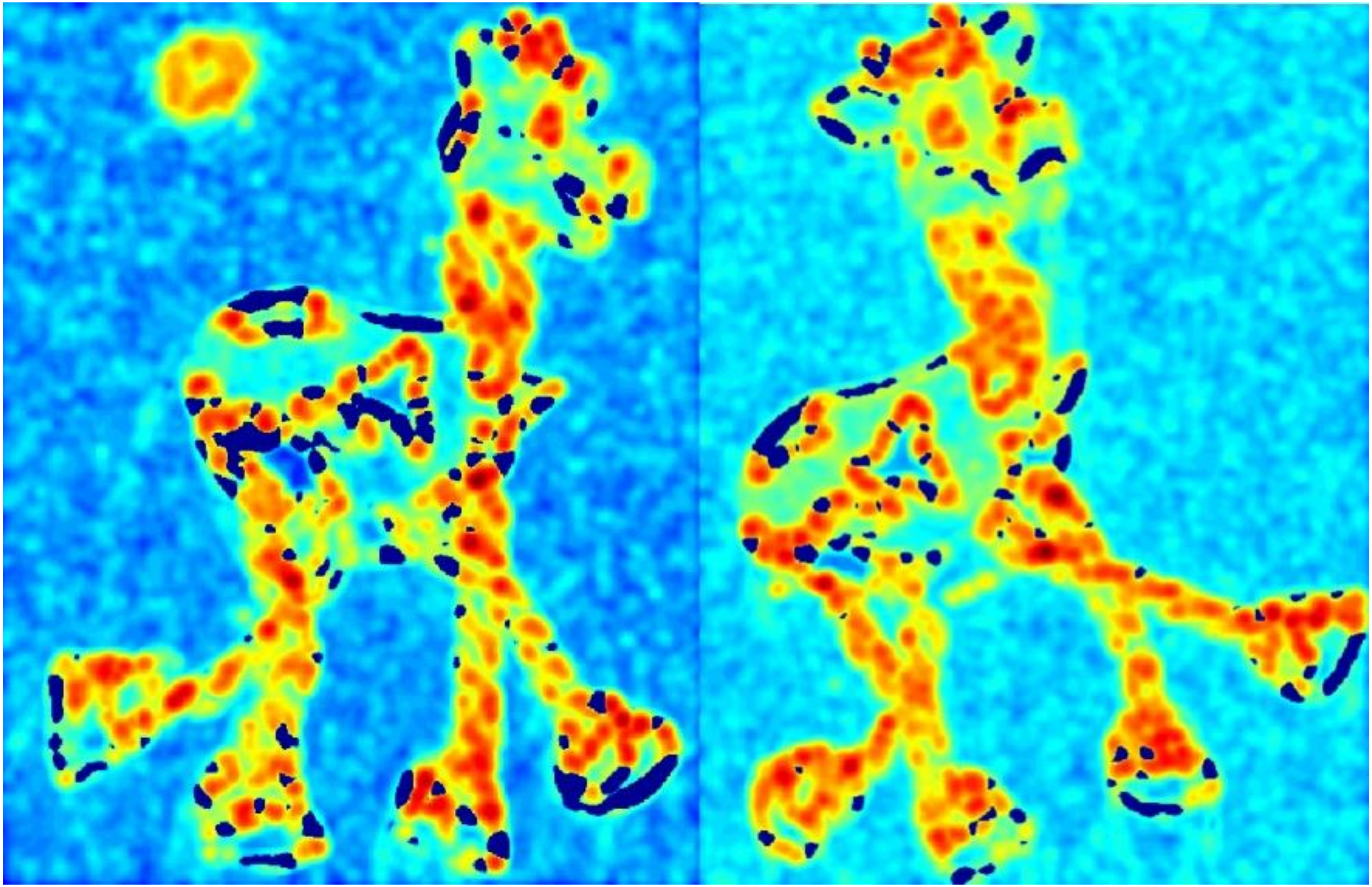
1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel
3. Compute corner response function *R*
4. Threshold *R*
5. Find local maxima of response function (nonmaximum suppression)

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

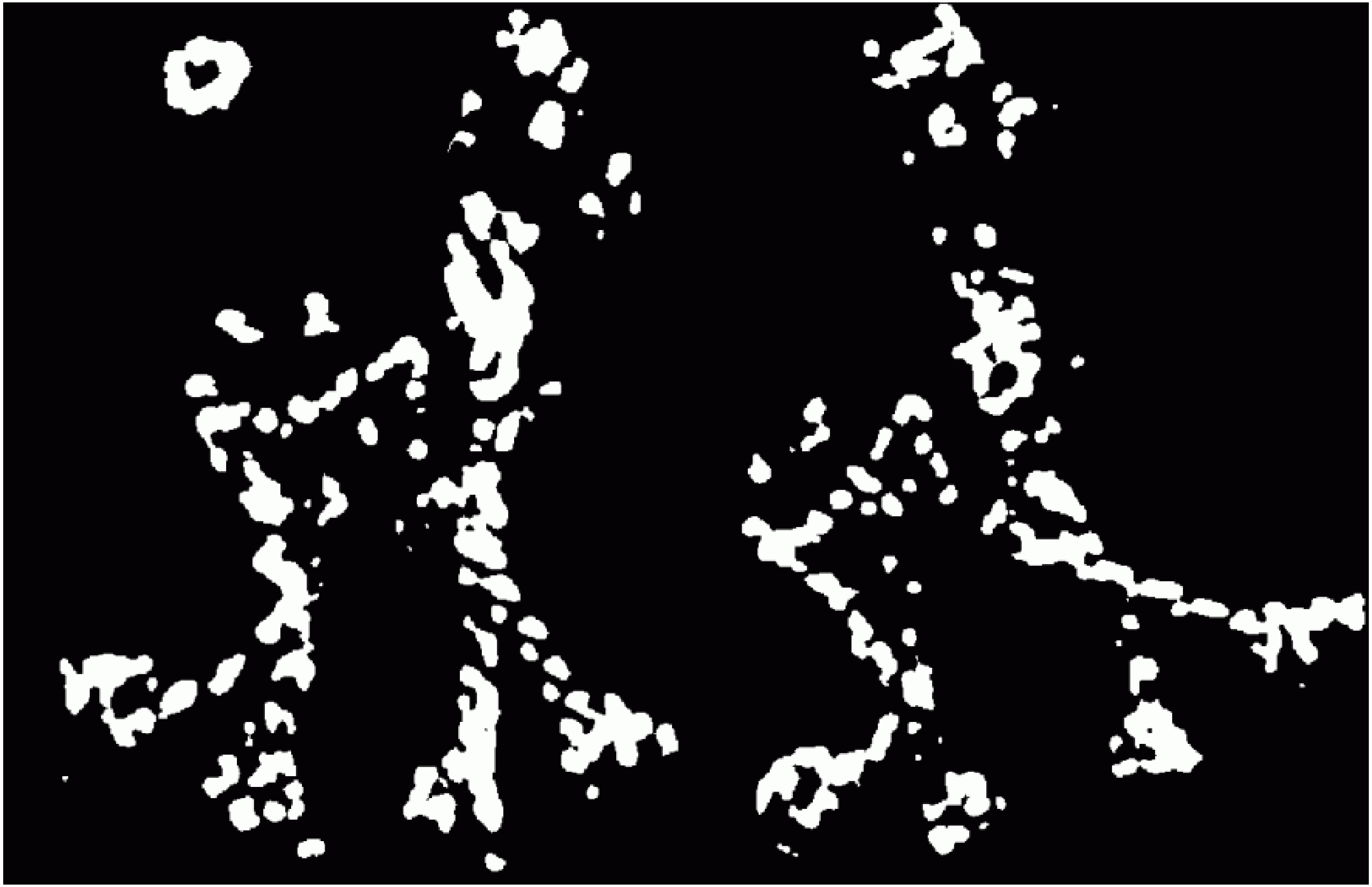
Harris Detector: Steps

Compute corner response R



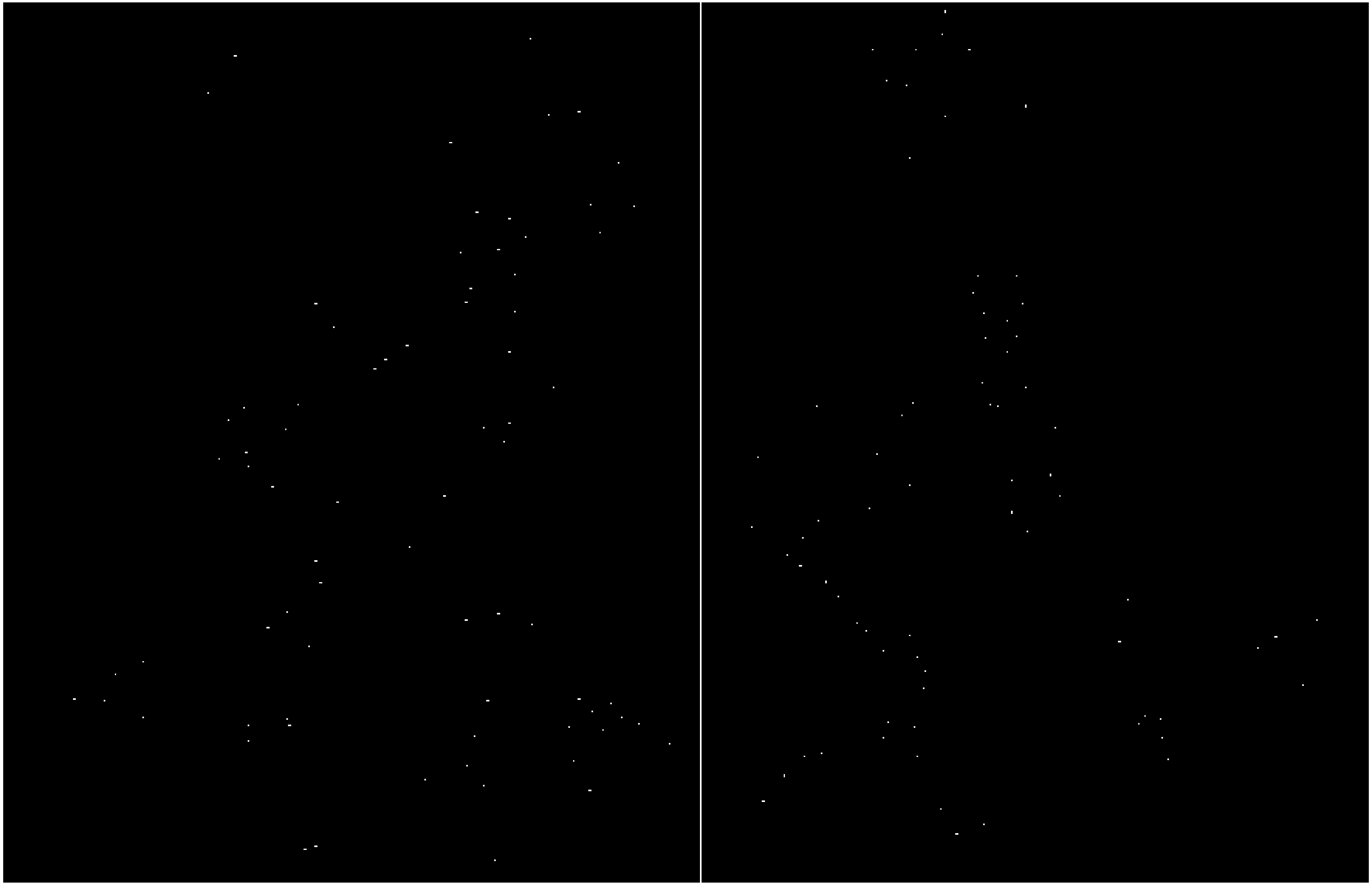
Harris Detector: Steps

Find points with large corner response: *R*>threshold



Harris Detector: Steps

Take only the points of local maxima of *R*

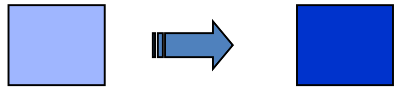


Harris Detector: Steps

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

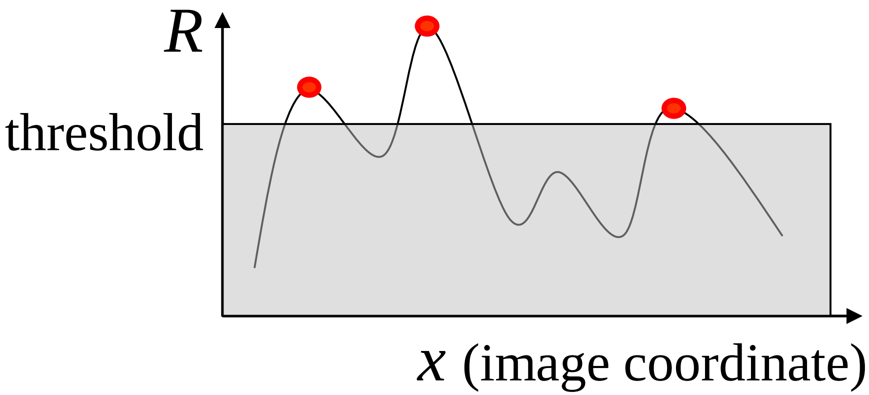
Affine Intensity Change

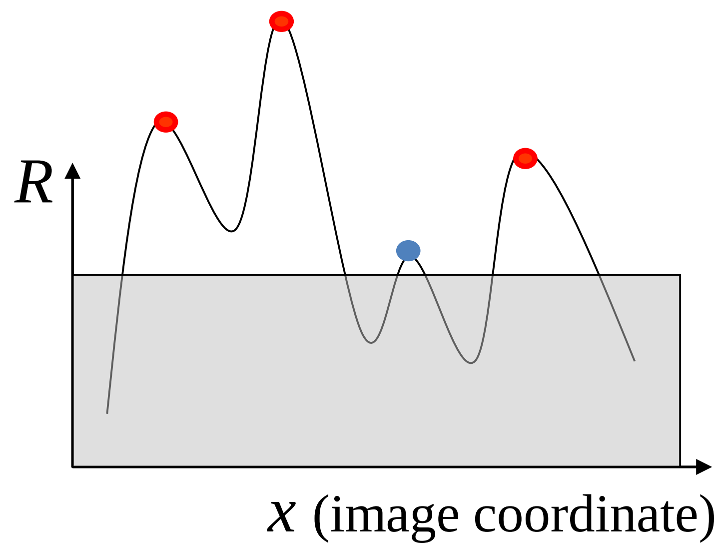


$I \to a\,I + b$

- Only derivatives are used => invariance to intensity shift $I \to I + b$

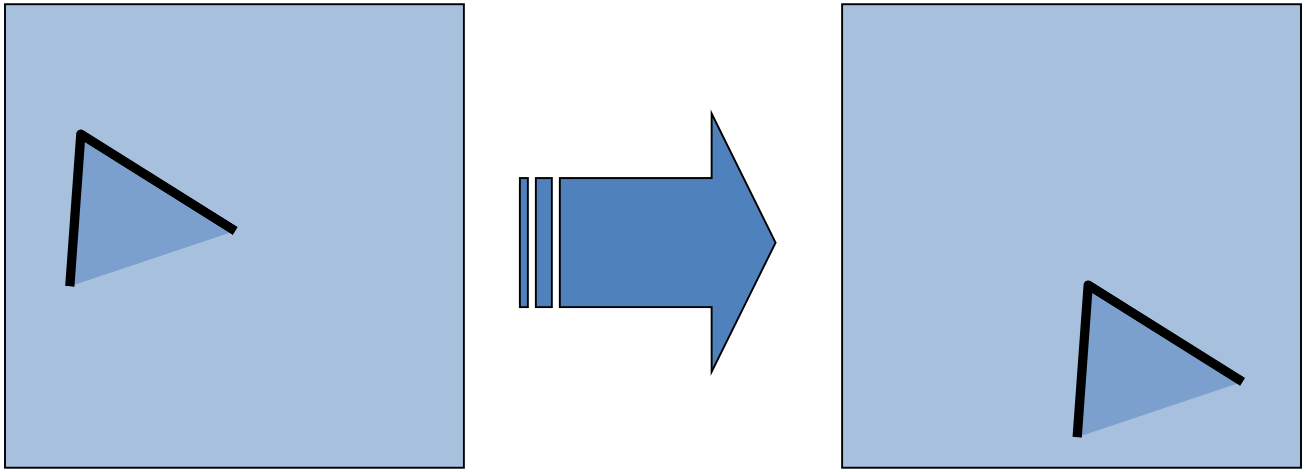
- Intensity scaling: $I \to a\,I$





Partially invariant to affine intensity change

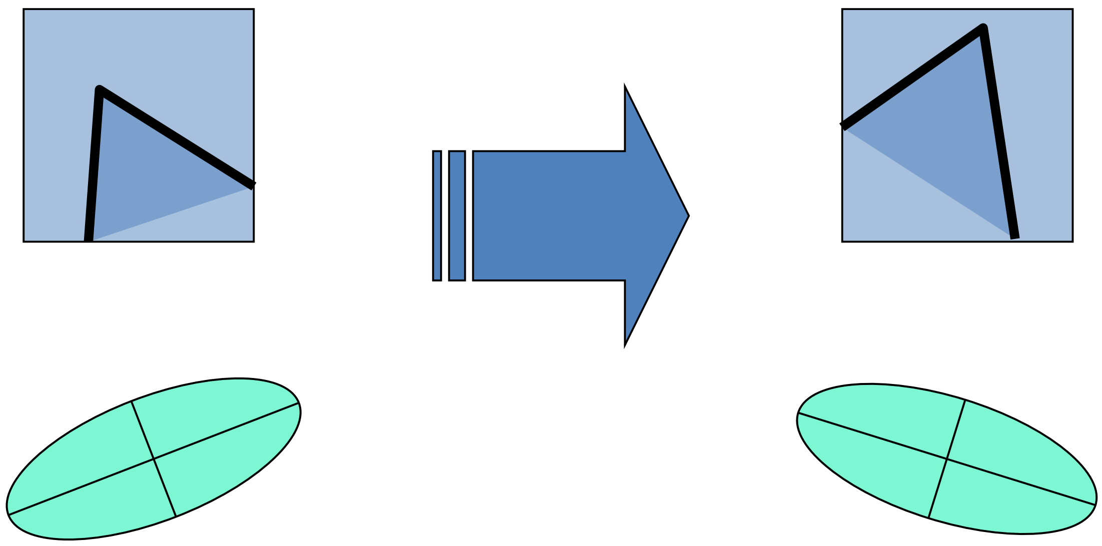
Image Translation



➢ Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

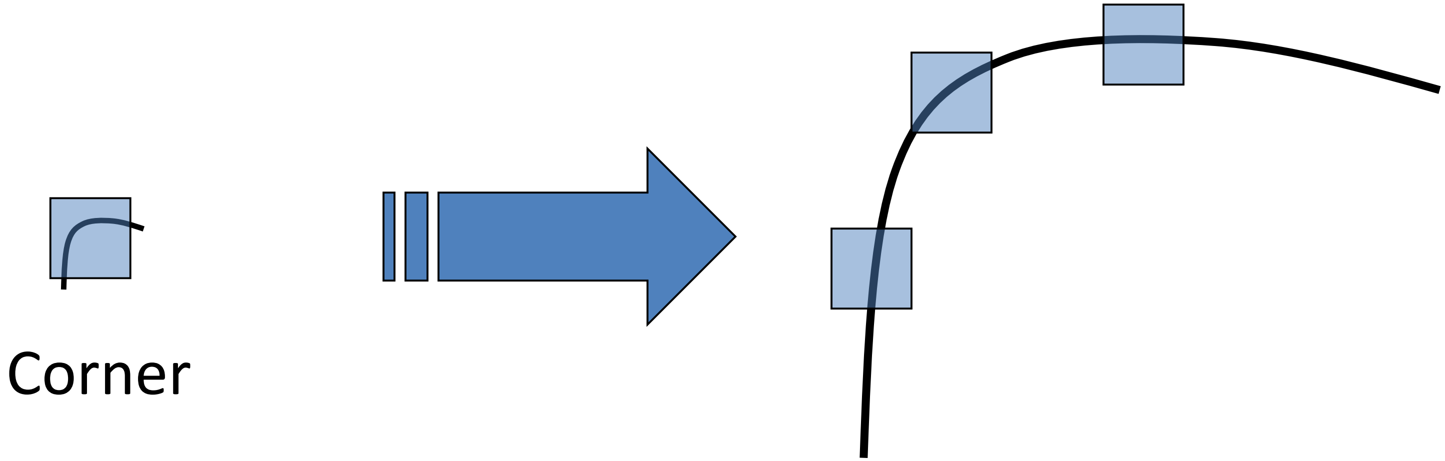
Image Rotation



- 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!

Summary

- Why we need feature extraction? What are the applications of feature extraction?
- What are Characteristics of Good Features?
- Describe the basic idea of corner detection.
- How to decide whether a point is in a flat region, on an edge, or corner according to the two eigenvalues of the second moment matrix?
- Describe steps of Harris detector
- What is Invariance and Covariance?
- Is affine intensity change invariant? Is image translation, rotation, or scaling covariant?