Filtering Images in the Spatial Domain
Chapter 3b G&W

Ross Whitaker
(modified by Guido Gerig)
School of Computing
University of Utah

Overview

- Correlation and convolution
- Linear filtering
  - Smoothing, kernels, models
  - Detection
  - Derivatives
- Nonlinear filtering
  - Median filtering
  - Bilateral filtering
  - Neighborhood statistics and nonlocal filtering

Cross Correlation

- Operation on image neighborhood and small …
  - "mask", "filter", "stencil", "kernel"
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Cross Correlation

- Operation on image neighborhood and small …
  - "mask", "filter", "stencil", "kernel"
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

```
  0.0*87 + 0.1*95 + 0.0*103
+ 0.1*50 + 0.6*36 + 0.1*150
+ 0.0*20 + 0.1*47 + 0.0*205 = 55.8
```

Cross Correlation

- Operation on image neighborhood and small …
  - "mask", "filter", "stencil", "kernel"
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	55.8	.	.
.	.	.	.

```
  0.0*87 + 0.1*95 + 0.0*103
+ 0.1*50 + 0.6*36 + 0.1*150
+ 0.0*20 + 0.1*47 + 0.0*205 = 55.8
```

Cross Correlation

- Operation on image neighborhood and small …
  - "mask", "filter", "stencil", "kernel"
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	55.8	.	.
.	.	.	.

```
  0.0*87 + 0.1*95 + 0.0*103
+ 0.1*50 + 0.6*36 + 0.1*150
+ 0.0*20 + 0.1*47 + 0.0*205 = 55.8
```

Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	55.8	.	.
.	.	.	.

Cross Correlation

- Operation on image neighborhood and small …
  - "mask", "filter", "stencil", "kernel"
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

→

Output image

.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Cross Correlation

- Operation on image neighborhood and small …
  - "mask", "filter", "stencil", "kernel"
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

```
  0.0*95 + 0.1*103 + 0.0*150
+ 0.1*36 + 0.6*150 + 0.1*104
+ 0.0*47 + 0.1*205 + 0.0*77 = 134.8
```

Cross Correlation

- Operation on image neighborhood and small …
  - "mask", "filter", "stencil", "kernel"
- Linear operations within a moving window

Input image

100	130	104	99	…
87	95	103	150	…
50	36	150	104	…
20	47	205	77	…
.	.	.	.	
.	.	.	.	

Filter

0.0	0.1	0.0
0.1	0.6	0.1
0.0	0.1	0.0

Output image

.	.	.	.
.	.	.	.
.	.	134.8	.
.	.	.	.

```
  0.0*95 + 0.1*103 + 0.0*150
+ 0.1*36 + 0.6*150 + 0.1*104
+ 0.0*47 + 0.1*205 + 0.0*77 = 134.8
```

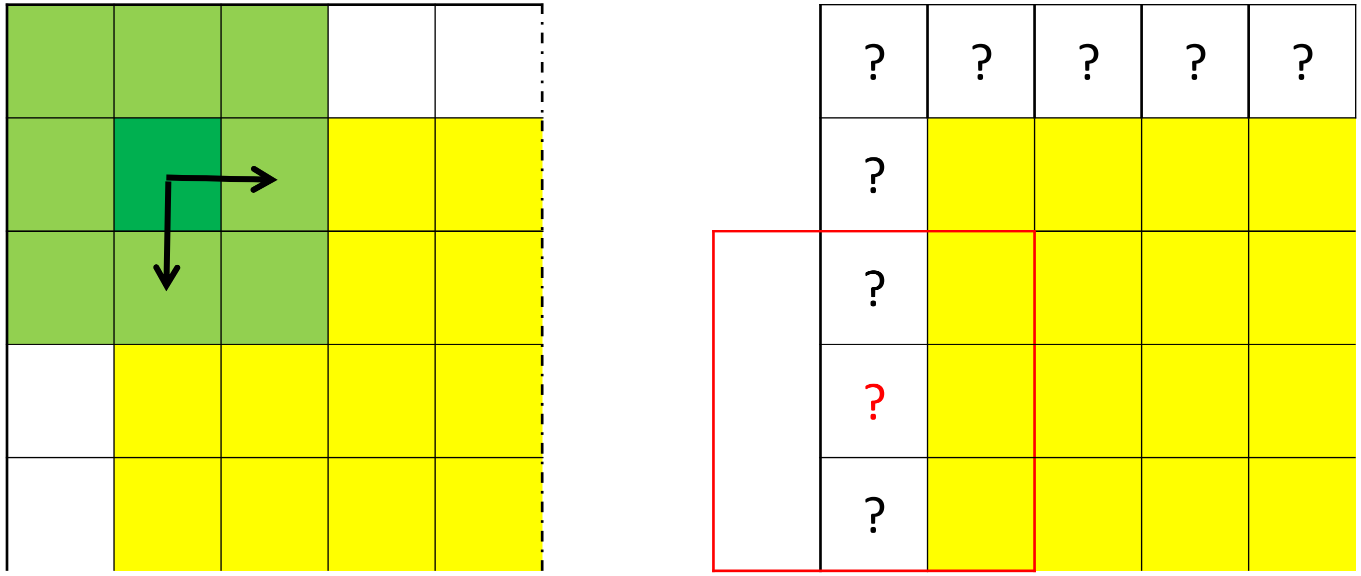
Cross Correlation

- 1D $$g(x) = \sum_{s=-a}^{a} w(s) f(x+s)$$
- 2D $$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$w(s,t) = \begin{array}{|ccccc|} w(-a,-b) & \cdots & & \cdots & w(a,-b) \\ \vdots & & & & \vdots \\ & \cdots & w(0,0) & \cdots & \\ \vdots & & & & \vdots \\ w(-a,b) & \cdots & & \cdots & w(a,b) \end{array}$$

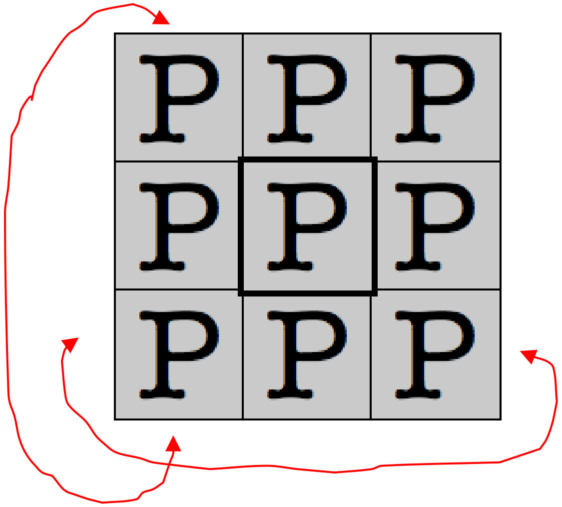
Correlation: Technical Details

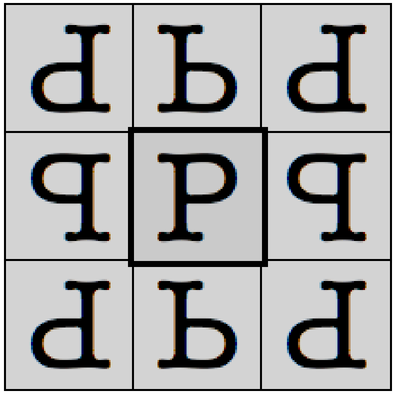
- How to filter boundary?



Correlation: Technical Details

- Boundary conditions
  - Boundary not filtered (keep it 0)
  - Pad image with amount (a,b)
    - Constant value or repeat edge values
  - Cyclical boundary conditions
    - Wrap or mirroring





Correlation: Technical Details

- Boundaries
  - Can also modify kernel – no longer correlation
- For analysis
  - Image domains infinite
  - Data compact (goes to zero far away from origin)

$$g(x,y) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w(s,t) f(x+s, y+t)$$

Correlation: Properties

- Shift invariant

$$g = w \circ f \qquad g(x,y) = w(x,y) \circ f(x,y)$$

$$w(x,y) \circ f(x-x_0, y-y_0) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w(s,t) f(x-x_0+s, y-y_0+t) = g(x-x_0, y-y_0)$$

Correlation: Properties

- Shift invariant

$$g = w \circ f \qquad g(x,y) = w(x,y) \circ f(x,y)$$

$$w(x,y) \circ f(x-x_0, y-y_0) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w(s,t) f(x-x_0+s, y-y_0+t) = g(x-x_0, y-y_0)$$

- Linear $w \circ (\alpha e + \beta f) = \alpha w \circ e + \beta w \circ f$

Compact notation

$$C_{wf} = w \circ f$$

Filters: Considerations

- Normalize
  - Sums to one
  - Sums to zero (some cases, see later)
- Symmetry
  - Left, right, up, down
  - Rotational
- Special case: auto correlation

$$C_{ff} = f \circ f$$

Examples 1

0	0	0
0	1	0
0	0	0

Examples 1

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

Examples 1

0	0	0
0	1	0
0	0	0

1/9 *

1	1	1
1	1	1
1	1	1

Examples 1

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

$$1/9 * \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

Examples 1

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

Examples 2

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

$$\frac{1}{25} * \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}$$

Smoothing and Noise

Noisy image

5x5 box filter

Noise Analysis

- Consider an a simple image I() with additive, uncorrelated, zero-mean noise of variance s
- What is the expected rms error of the corrupted image?
- If we process the image with a box filter of size 2a+1 what is the expected error of the filtered image?

$$\mathrm{RMSE} = \left(\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \left(\tilde{I}(x,y) - I(x,y) \right)^2 \right)^{\frac{1}{2}}$$

Other Filters

- Disk
  - Circularly symmetric, jagged in discrete case
- Gaussians
  - Circularly symmetric, smooth for large enough stdev
  - Must normalize in order to sum to one
- Derivatives – discrete/finite differences
  - Operators

Gaussian Kernel

$$\sigma = 1;\ \text{Plot}\left[\frac{1}{\sqrt{2\,\pi\,\sigma^2}}\,E^{-\frac{x^2}{2\,\sigma^2}},\ \{x,\ -4,\ 4\},\ \text{ImageSize} \rightarrow\right.$$

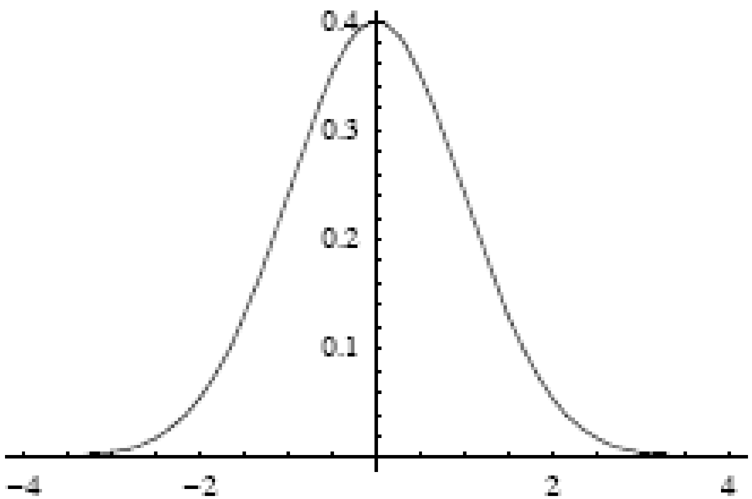


Figure 2.1 The Gaussian kernel with unit standard deviation in 1D.

Gaussian Kernel

$$G_{1D}(x;\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-\frac{x^2}{2\sigma^2}},\ G_{2D}(x, y;\sigma) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}},\ G_{ND}(\vec{x};\sigma) = \frac{1}{\left(\sqrt{2\pi}\,\sigma\right)^N}\, e^{-\frac{|\vec{x}|^2}{2\sigma^2}}$$

Normalization to 1.0

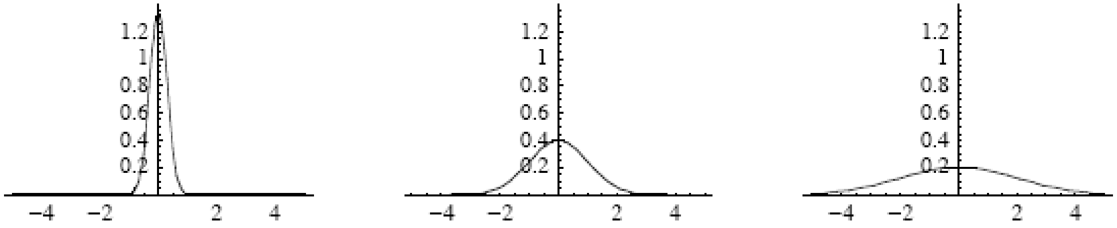
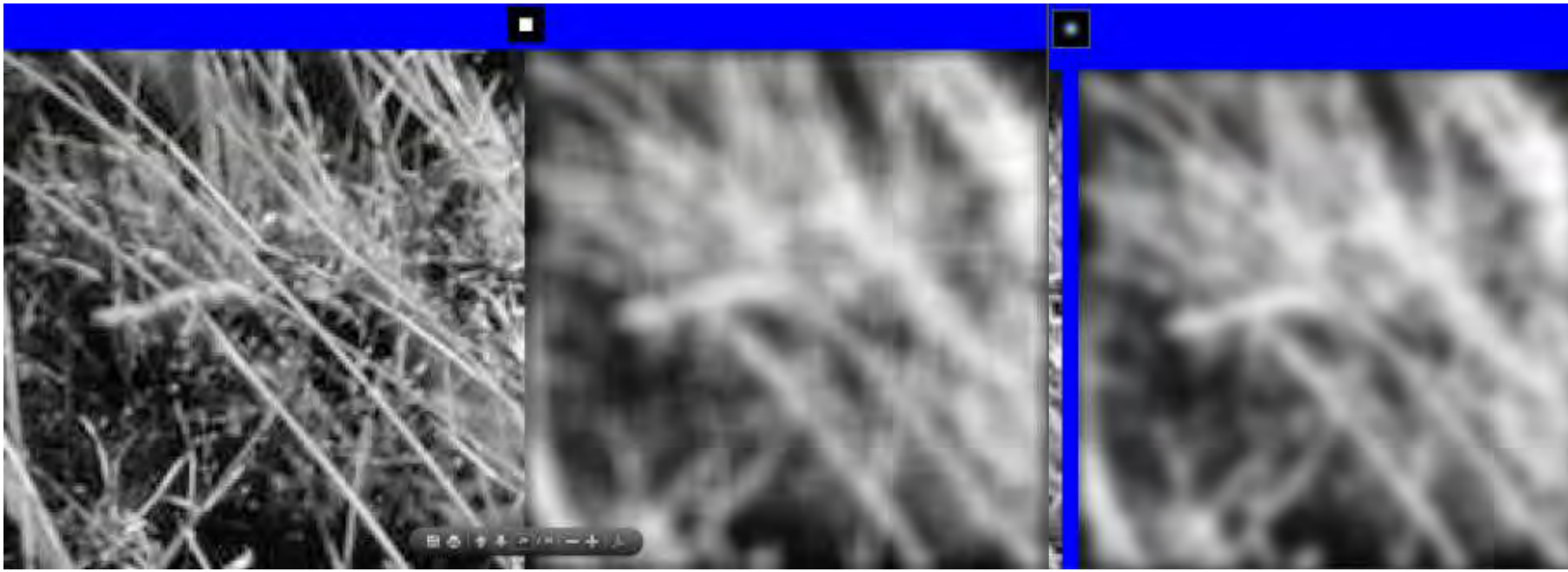


Figure 3.2 The Gaussian function at scales $\sigma = .3$, $\sigma = 1$ and $\sigma = 2$. The kernel is normalized, so the total area under the curve is always unity.

Box versus Gaussian



Convolution

Java demo: http://www.jhu.edu/signals/convolve/
http://www.jhu.edu/signals/discreteconv2/index.html

- Discrete

$$g(x,y) = w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

- Continuous

$$g(x,y) = w(x,y) * f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(s,t) f(x-s, y-t) ds dt$$

- Same as cross correlation with kernel transposed around each axis
- The two operations (correlation and convolution) are the same if the kernel is symmetric about axes

$$g = w \circ f = w^* * f$$

w^* reflection of w

Convolution: Properties

- Shift invariant, linear
- Commutative

$$f * g = g * f$$

- Associative

$$f * (g * h) = (f * g) * h$$

- Others (discussed later):
  - Derivatives, convolution theorem, spectrum…

Computing Convolution

Computing Convolution

- Compute time
  - MxM mask
  - NxN image

Computing Convolution

- Compute time
  - MxM mask
  - NxN image

} $O(M^2N^2)$ "for" loops are nested 4 deep

Computing Convolution

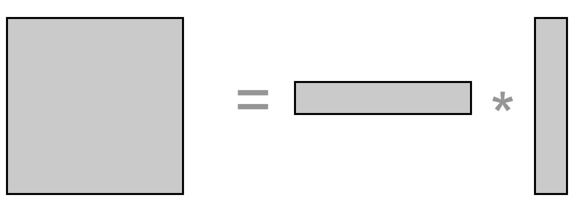
- Compute time
  - MxM mask
  - NxN image

$O(M^2N^2)$ "for" loops are nested 4 deep

- Special case: *separable*

Two 1D kernels

$$w = \overbrace{w_x * w_y}^{\text{Two 1D kernels}}$$



$$w * f = \underbrace{(w_x * w_y) * f}_{O(M^2N^2)} = \underbrace{w_x * (w_y * f)}_{O(MN^2)}$$

Separable Kernels

- Examples
  - Box/rectangle
  - Bilinear interpolation
  - Combinations of partial derivatives
    - $d^2f/dxdy$
  - Gaussian
    - Only filter that is both circularly symmetric and separable
- Counter examples
  - Disk
  - Cone
  - Pyramid

Separability

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

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

```
DisplayTogetherArray[{Plot[gauss[x, σ = 1], {x, -3, 3}],
   Plot3D[gauss[x, σ = 1] gauss[y, σ = 1], {x, -3, 3}, {y, -3, 3}]},
  ImageSize -> 440];
```

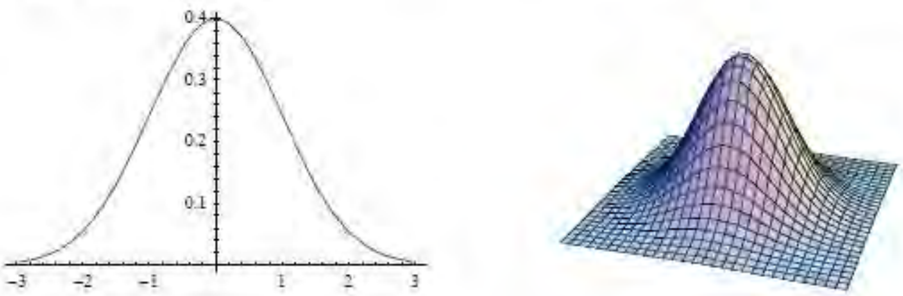
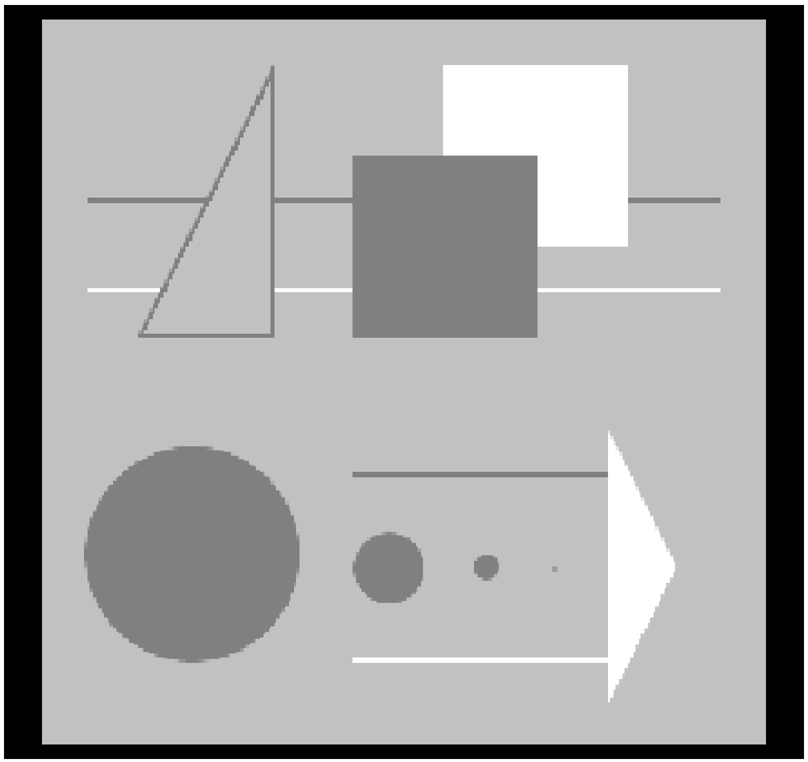


Figure 3.7 A product of Gaussian functions gives a higher dimensional Gaussian function. This is a consequence of the separability.

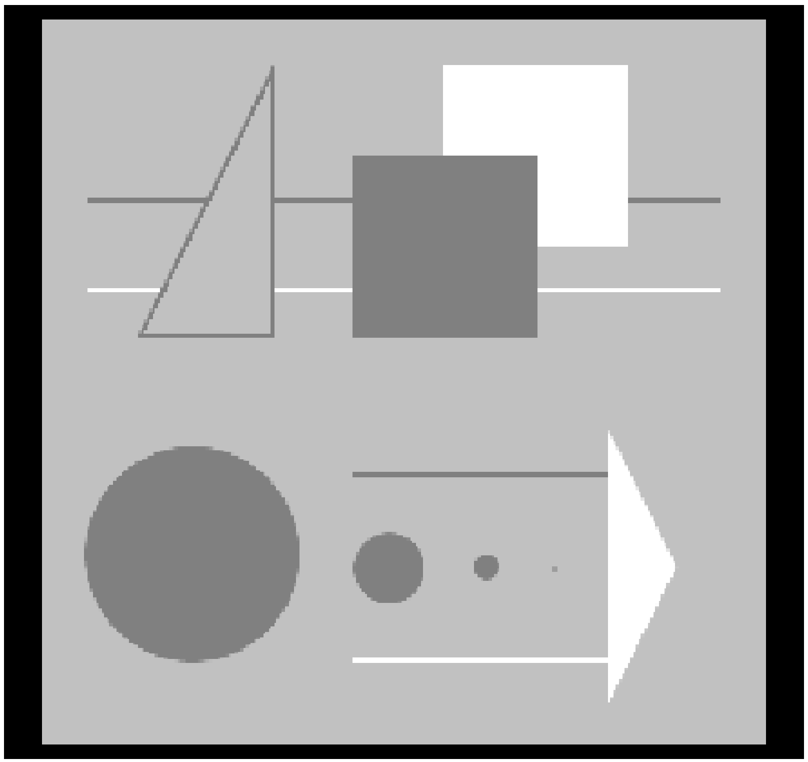
Digital Images: Boundaries are “Lines” or “Discontinuities”

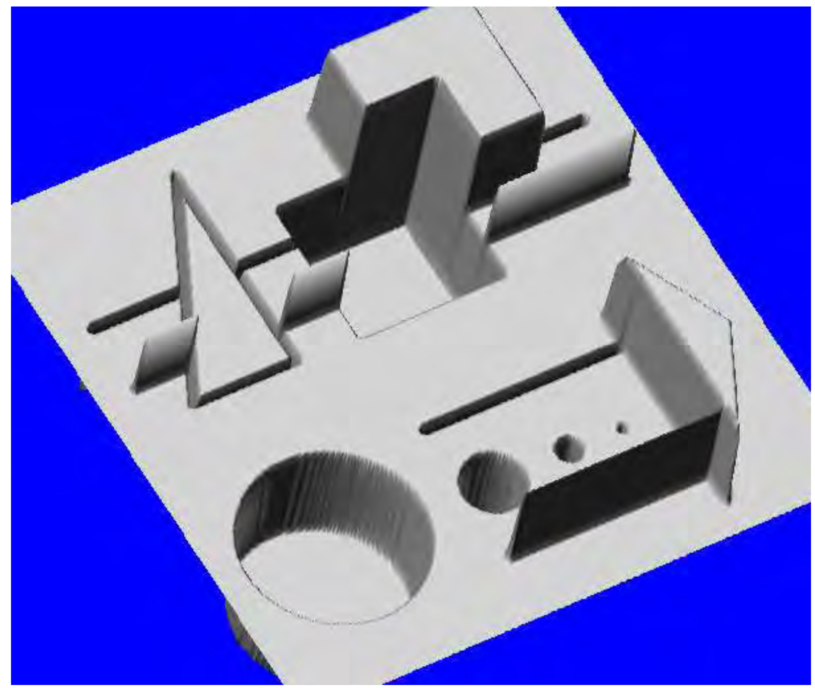


Example: Characterization of discontinuities?

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Digital Images: Boundaries are “Lines” or “Discontinuities”



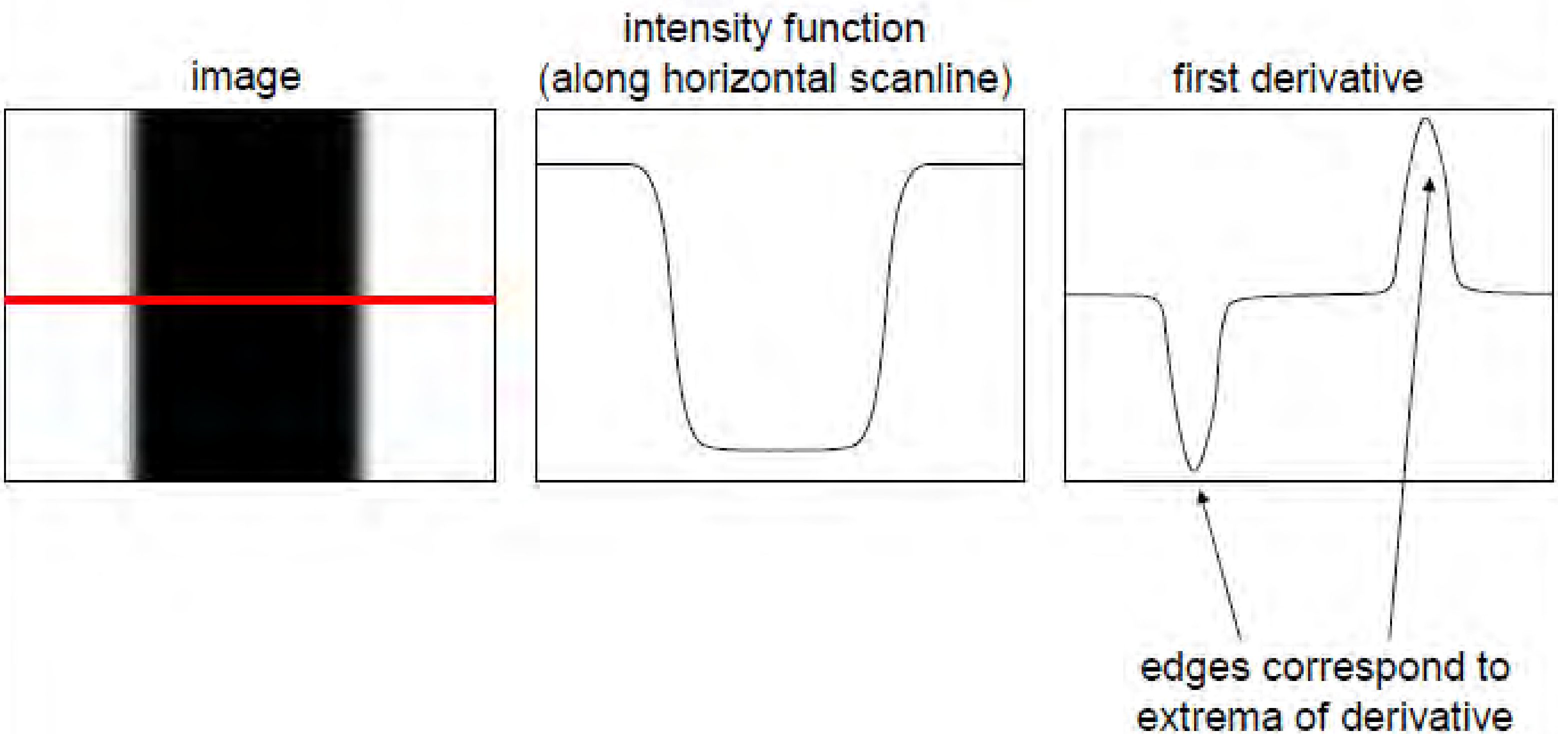


Example: Characterization of discontinuities?

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Characterizing edges

- An edge is a place of rapid change in the image intensity function



Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Differentiation and convolution

- Recall, for 2D function, f(x,y):

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon}\right)$$

- This is linear and shift invariant, so must be the result of a convolution.

- We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

- (which is obviously a convolution)

-1	1

Source: D. Forsyth, D. Lowe

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Derivatives: Finite Differences

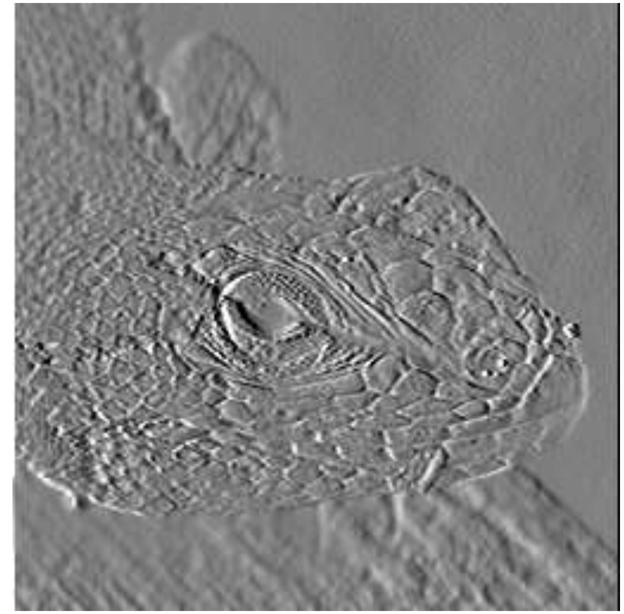
$$\frac{\partial f}{\partial x} \approx \frac{1}{2h}\left(f(x+1,y) - f(x-1,y)\right)$$

$$\frac{\partial f}{\partial x} \approx w_{dx} \circ f \qquad w_{dx} = \begin{array}{|c|c|c|}\hline -\frac{1}{2} & 0 & \frac{1}{2} \\ \hline\end{array}$$

$$\frac{\partial f}{\partial y} \approx w_{dy} \circ f \qquad w_{dy} = \begin{array}{|c|}\hline -\frac{1}{2} \\ \hline 0 \\ \hline \frac{1}{2} \\ \hline\end{array}$$

Derivative Example

0	0	0
−1	0	1
0	0	0



0	−1	0
0	0	0
0	1	0

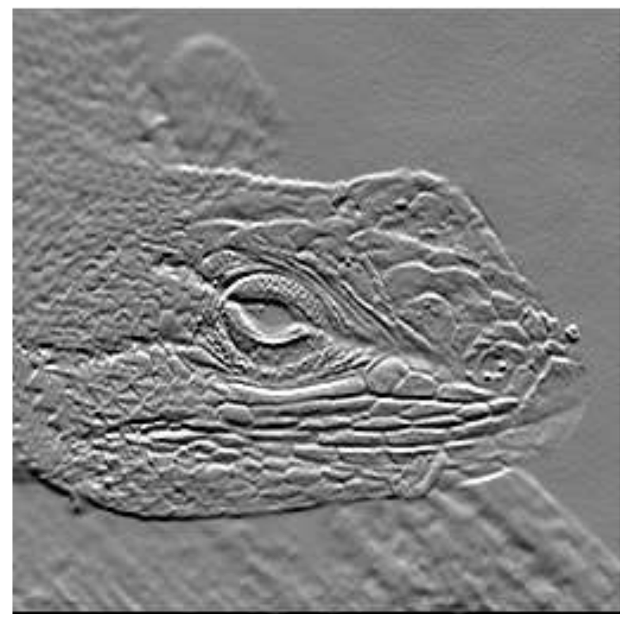
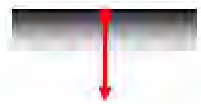


Image gradient

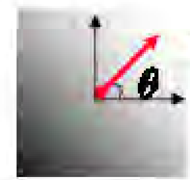
- The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- The gradient points in the direction of most rapid change in intensity

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



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

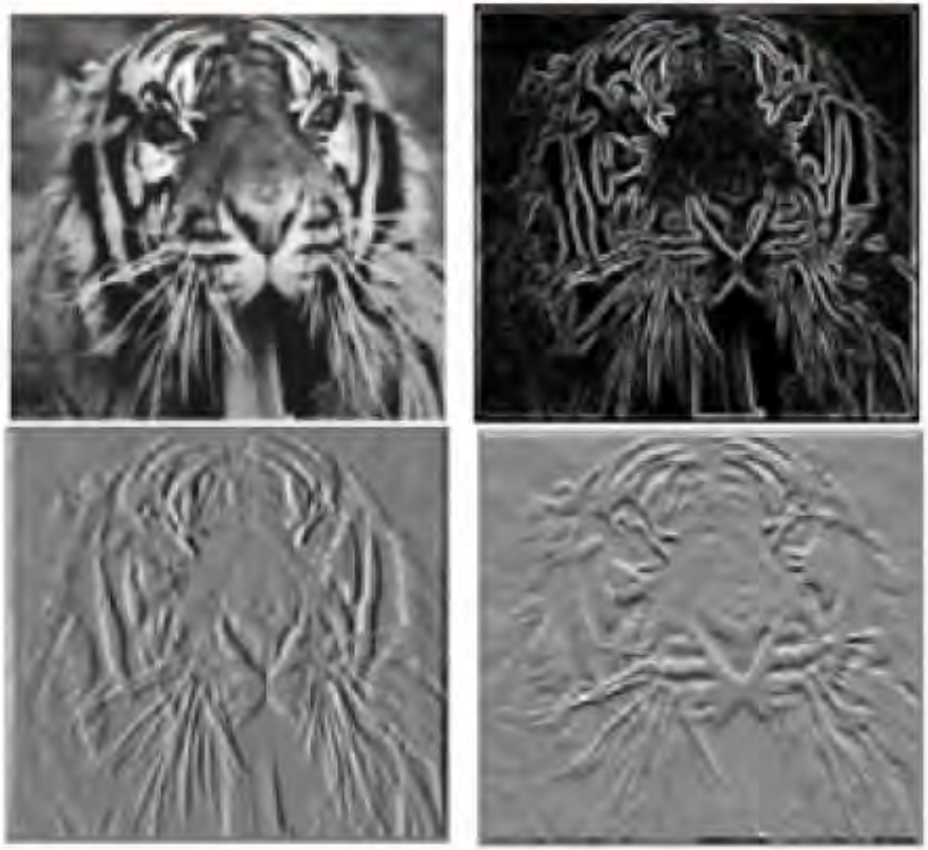


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- The gradient direction is given by:
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$
  - how does this relate to the direction of the edge? *perpendicular*
- The *edge strength* is given by the gradient magnitude
$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Finite differences: example



- Which one is the gradient in the x-direction (resp. y-direction)?

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Finite difference filters

- Other approximations of derivative filters exist:

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

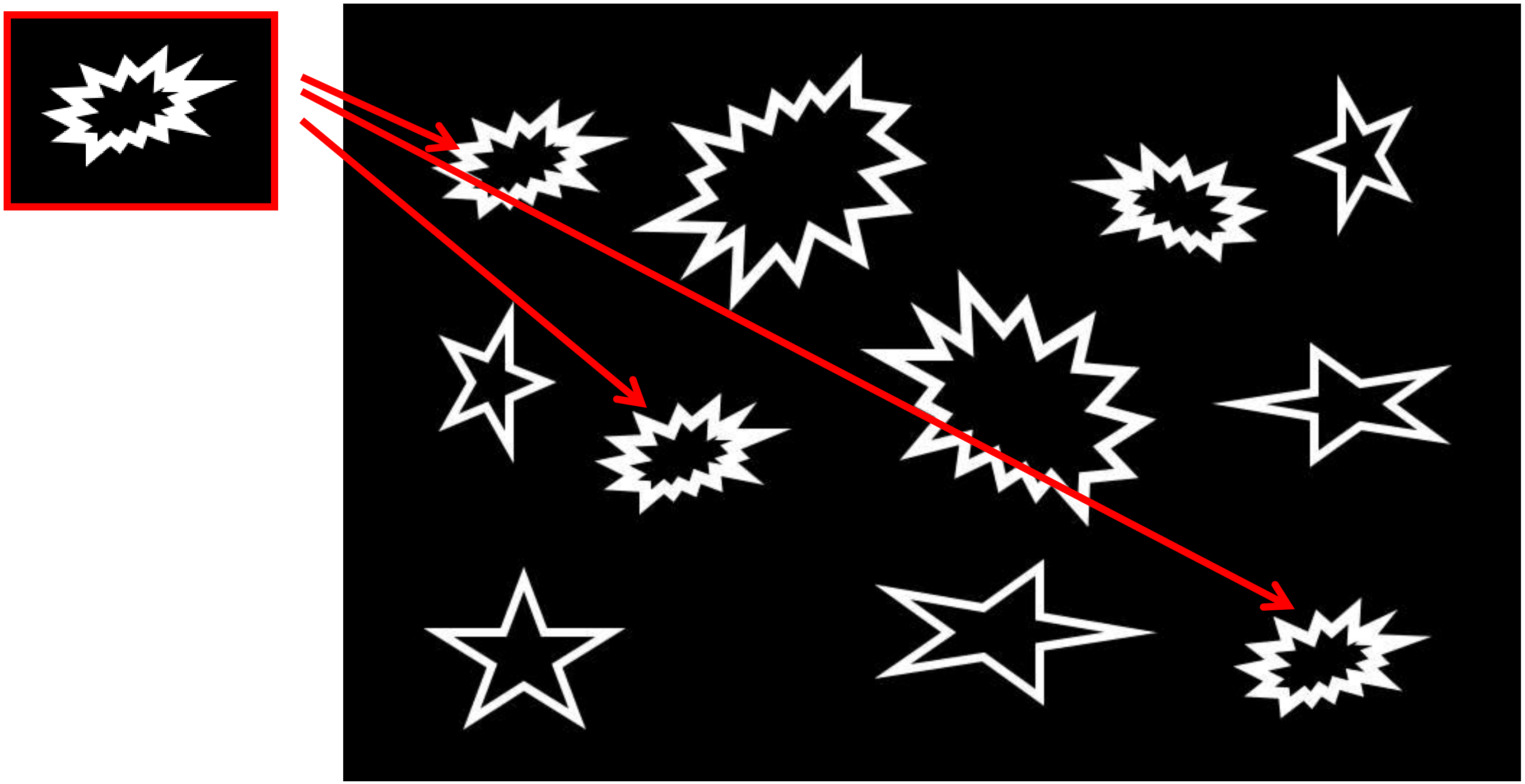
Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Source: K. Grauman

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Pattern Matching



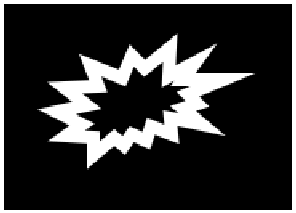
Pattern Matching/Detection

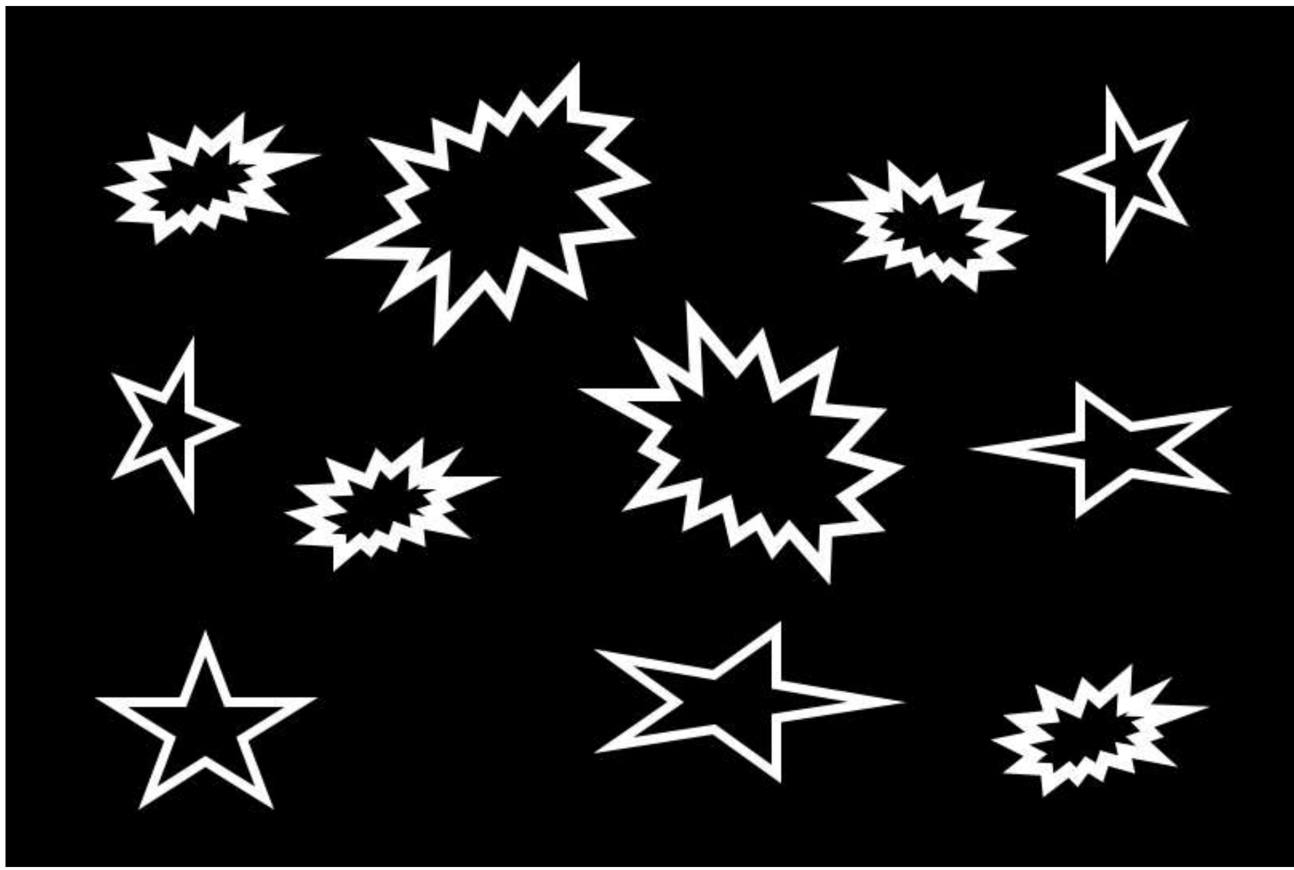
- The optimal (highest) response from a filter is the autocorrelation evaluated at position zero

$$\max_{\bar{x}} C_{ff}(\bar{x}) = C_{ff}(0) = \int f(\bar{s}) f(\bar{s}) d\bar{s}$$

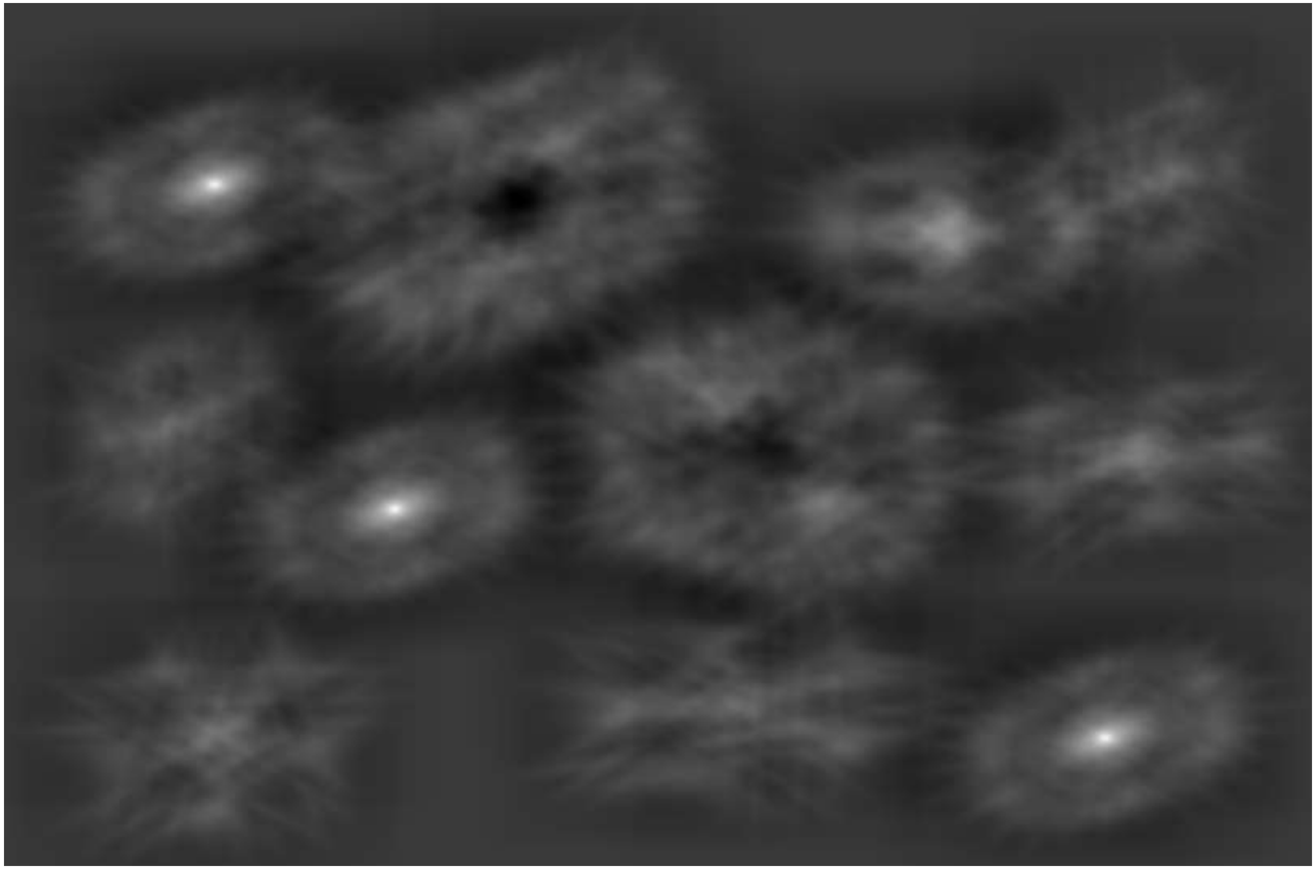
- A filter responds best when it matches a pattern that looks itself
- Strategy
  - Detect objects in images by correlation with “matched” filter

Matched Filter Example

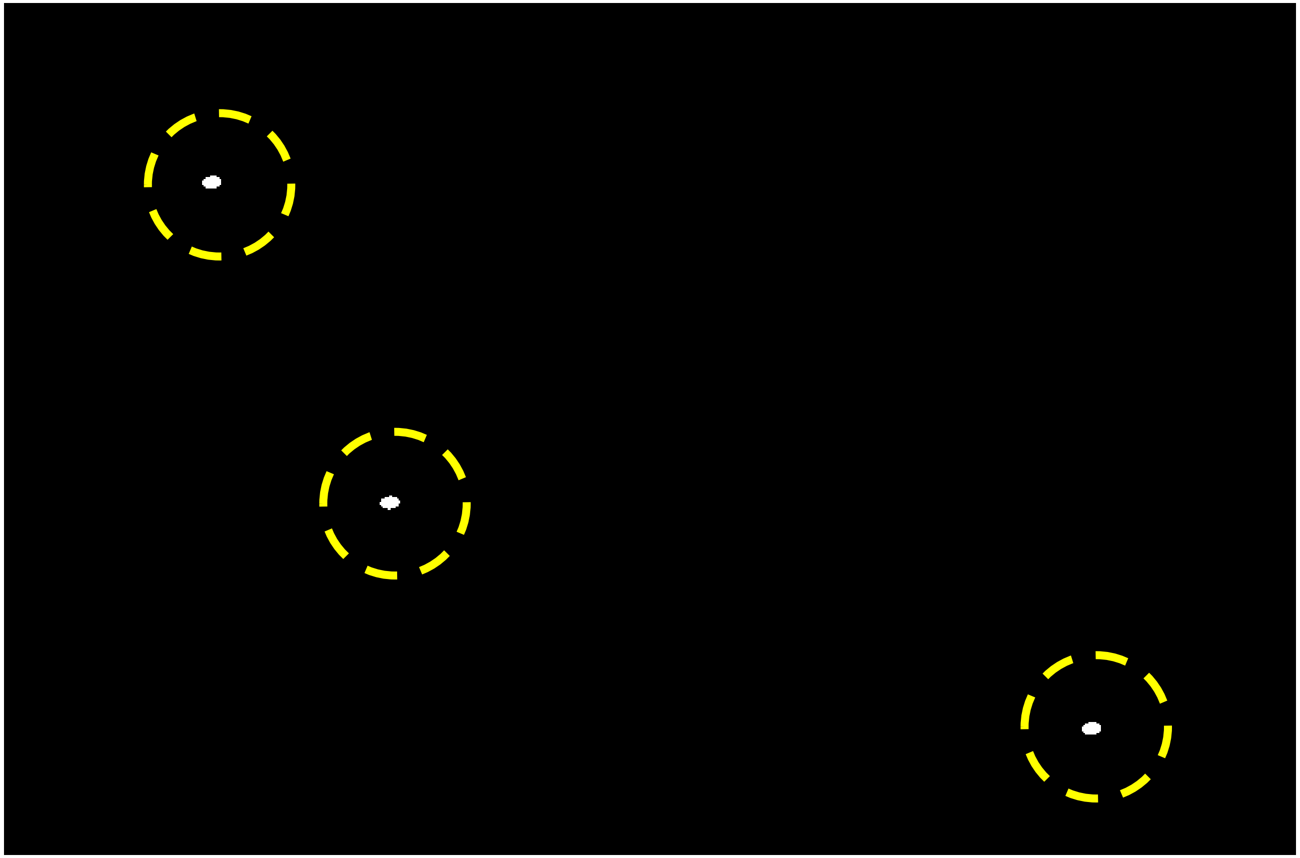




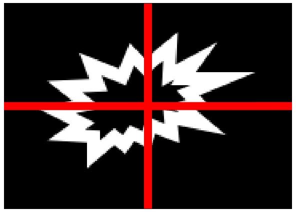
Matched Filter Example: Correlation of template with image

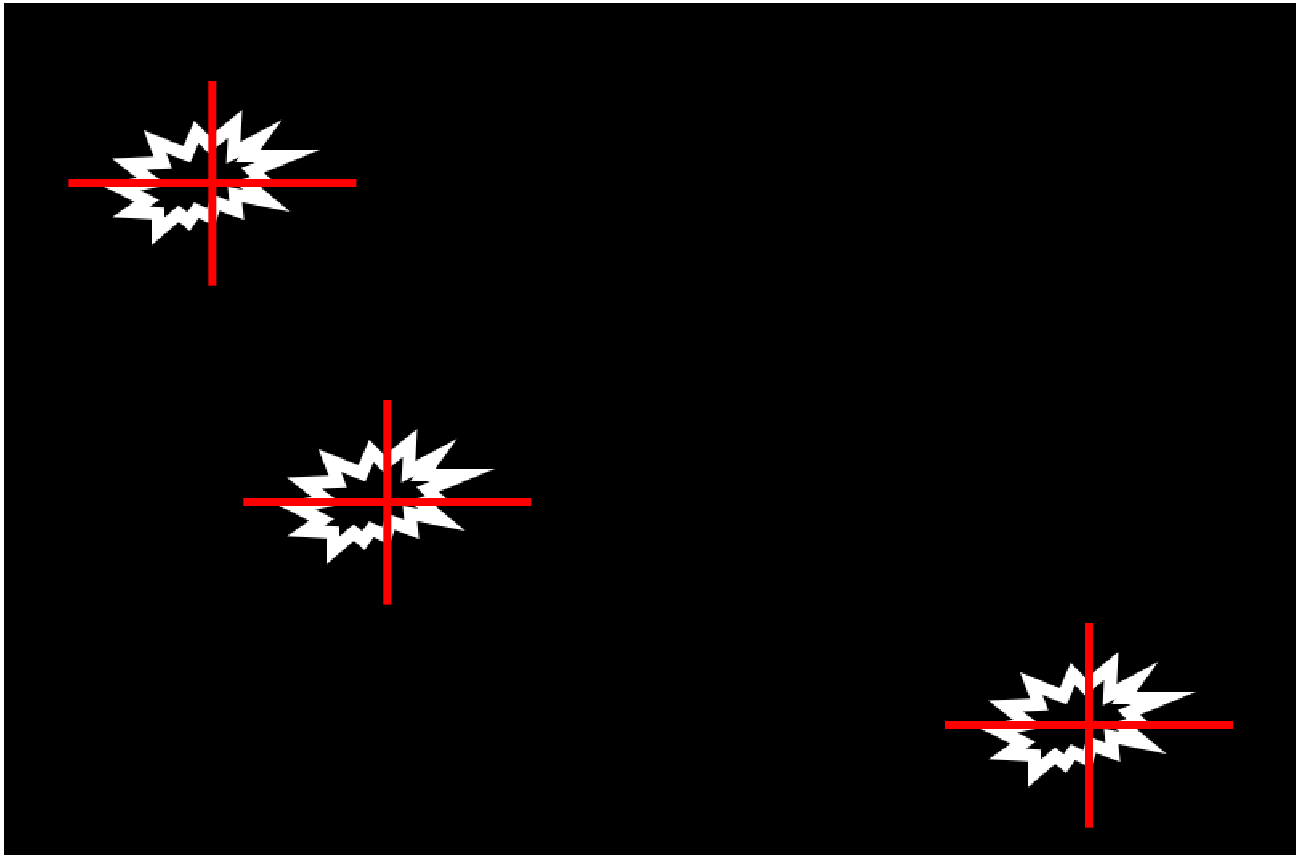


Matched Filter Example: Thresholding of correlation results



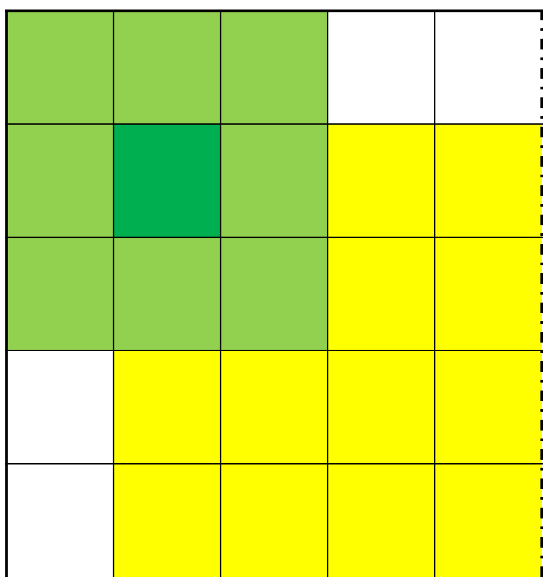
Matched Filter Example: High correlation → template found

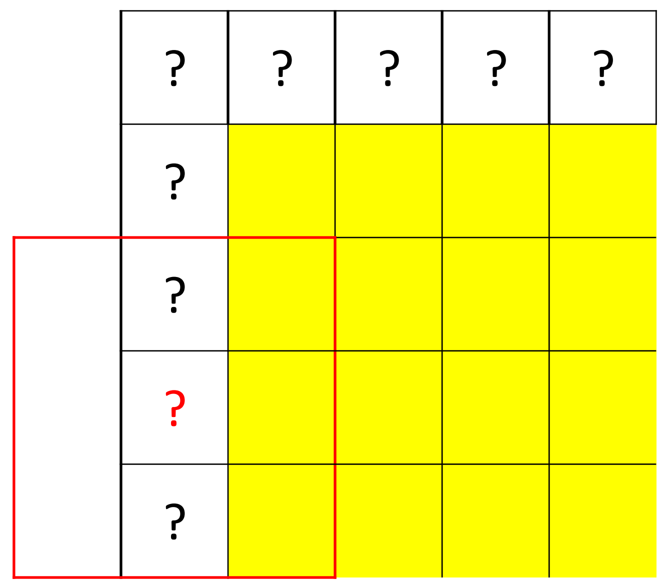




Summary of 9/15

- Spatial Filtering
  - Consider neighborhood information
  - Special consideration at boundary





Summary of 9/15

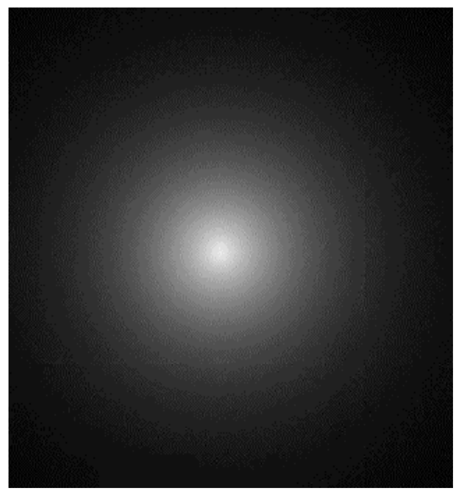
- Common Filters

Box

1/9 *

1	1	1
1	1	1
1	1	1

Gaussian

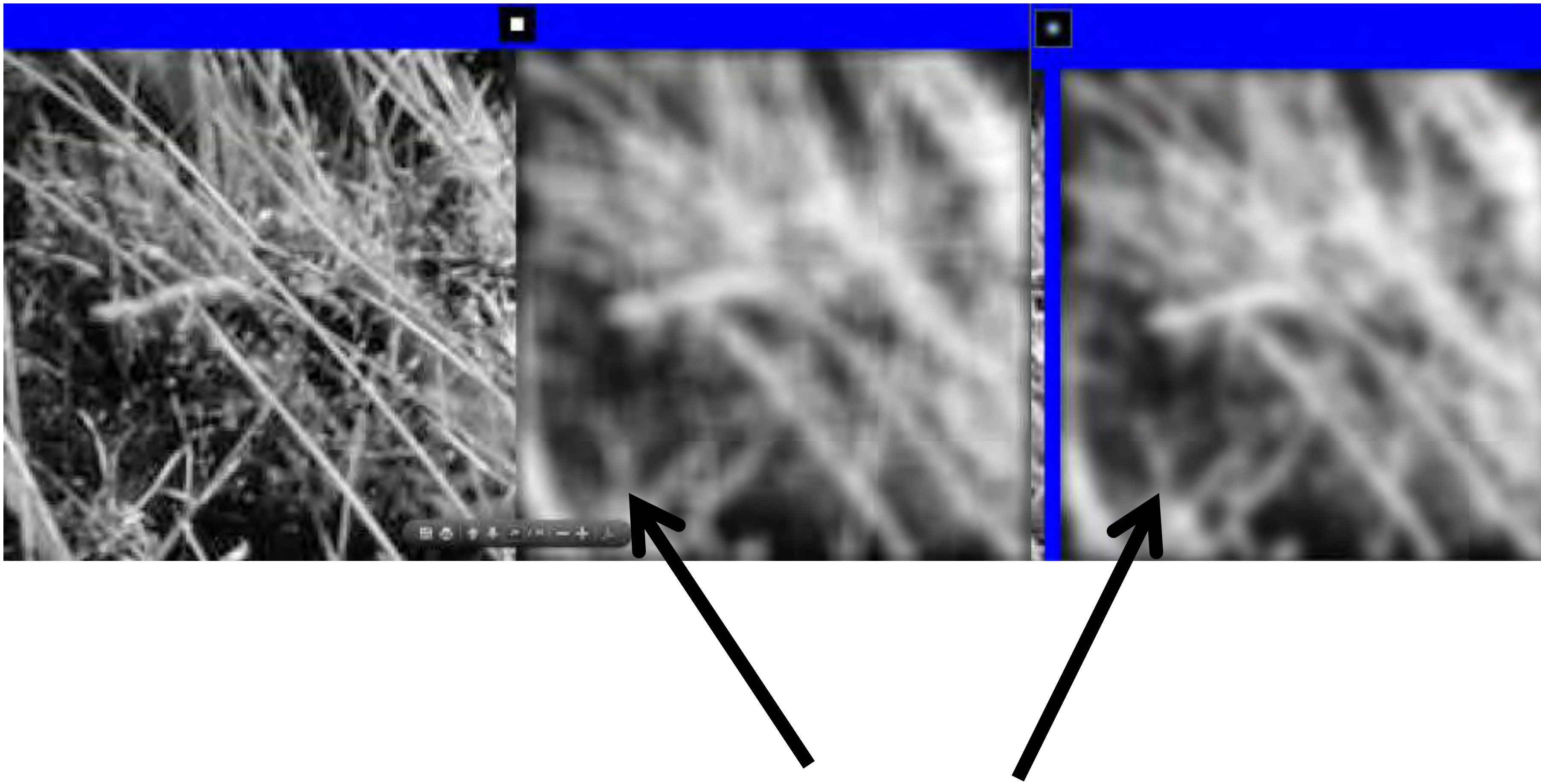


Derivative

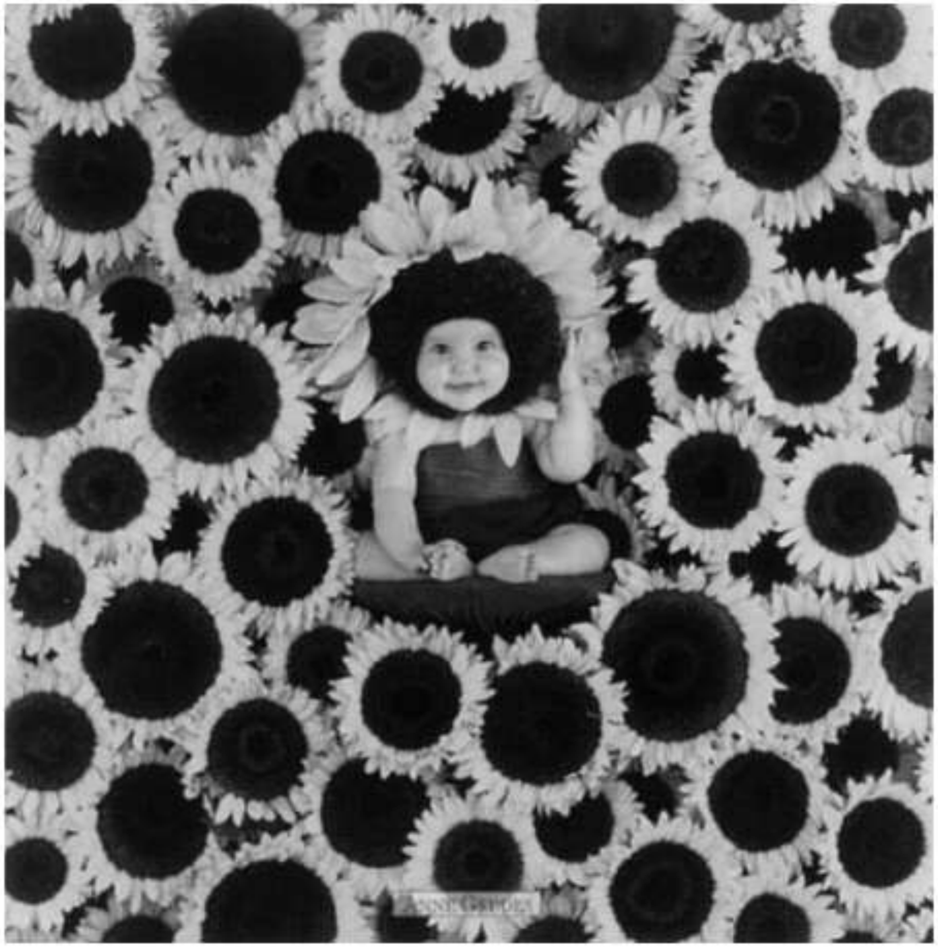
$$w_{dx} = \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

$$w_{dy} = \begin{bmatrix} -\frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix}$$

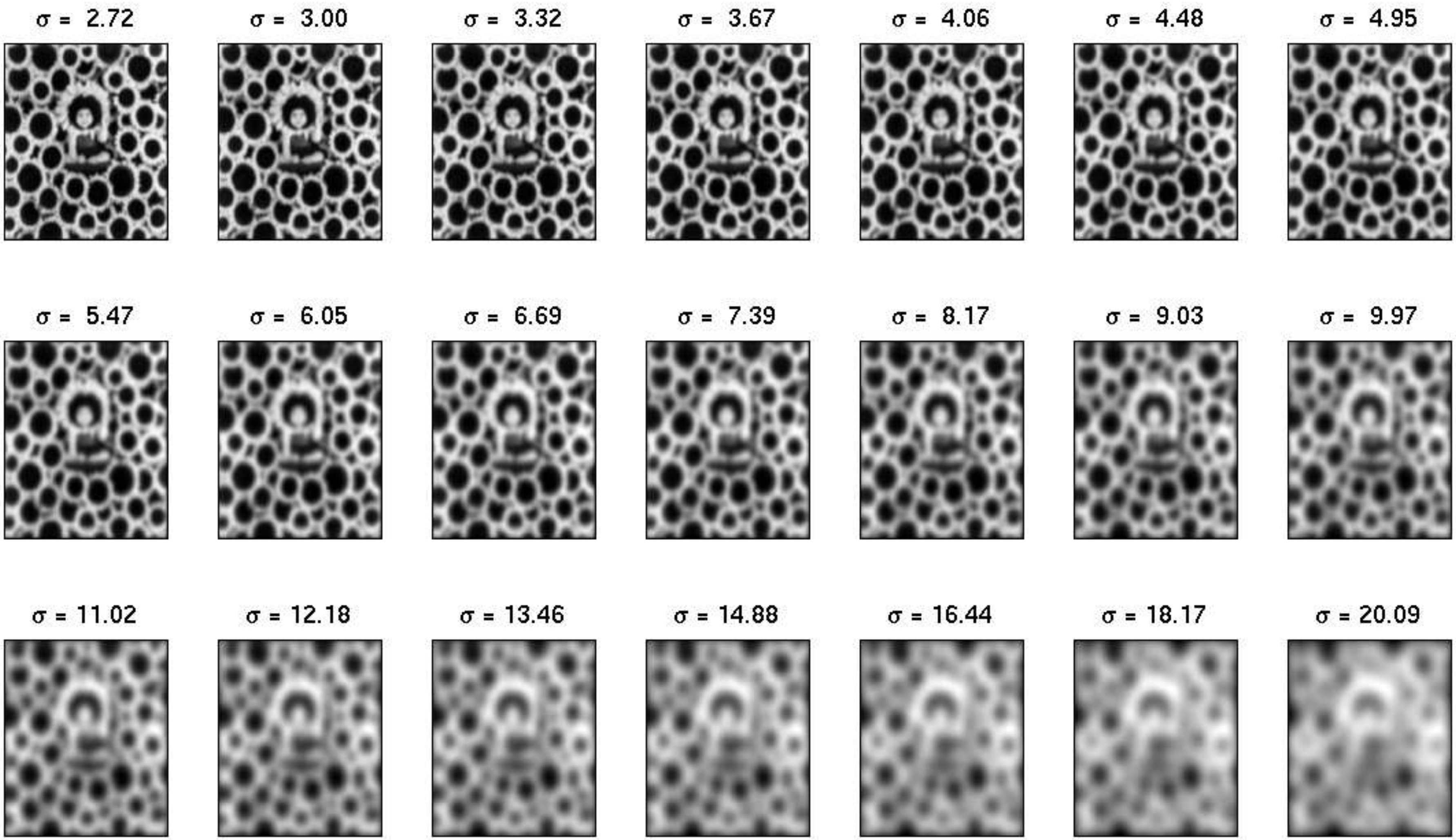
Box versus Gaussian



Gaussian Filtering



Gaussian Filtering



Finite differences: example

- Which one is the gradient in the x-direction (resp. y-direction)?

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

Cross-correlation and Convolution

- Cross-correlation

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

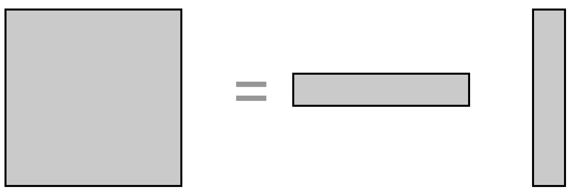
- Convolution

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Key Concepts

- Separability

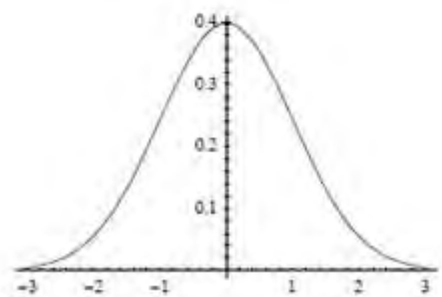
$$w = w_x * w_y$$

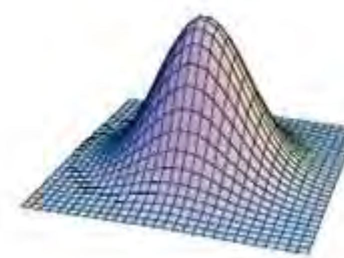


$$w * f = (w_x * w_y) * f = w_x * (w_y * f)$$

$(w_x * w_y) * f$: O(M^2N^2) $\qquad$ $w_x * (w_y * f)$: O(MN^2)

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





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

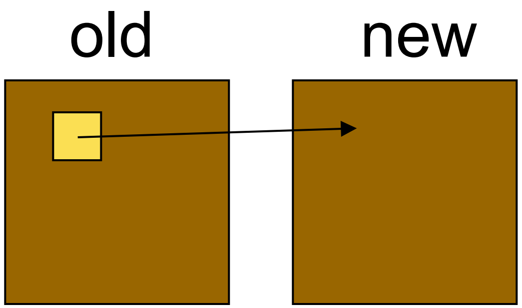
In this case, the two functions are the (identical) 1D Gaussian

Nonlinear Methods For Filtering

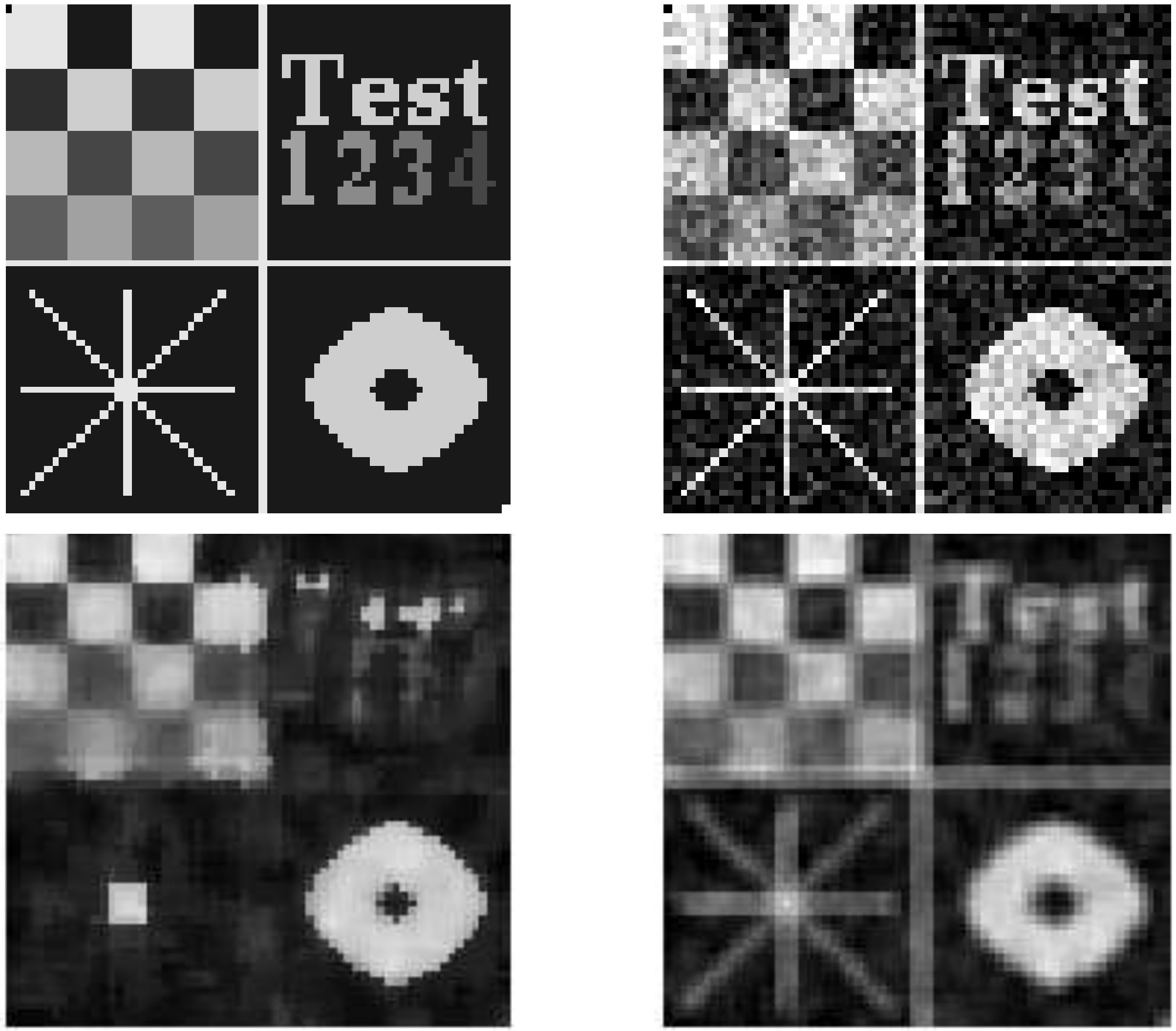
- Median filtering
- Bilateral filtering
- Neighborhood statistics and nonlocal filtering

Median Filtering

- For each neighborhood in image
  - Sliding window
  - Usually odd size (symmetric) 5x5, 7x7,…
- Sort the greyscale values
- Set the center pixel to the median
- Important: use “Jacobi” updates
  - Separate input and output buffers
  - All statistics on the original image



Median vs Gaussian



Original

+
Gaussian
Noise

3x3
Median

3x3 Box

Median Filter

- Issues
  - Boundaries
    - Compute on pixels that fall within window
  - Computational efficiency
    - What is the best algorithm?
- Properties
  - Removes outliers (replacement noise – salt and pepper)
  - Window size controls size of structures
  - Preserves straight edges, but rounds corners and features

Replacement Noise

- Also: “shot noise”, “salt&pepper”
- Replace certain % of pixels with samples from pdf
- Best filtering strategy: filter to avoid outliers

Smoothing of S&P Noise

- It's not zero mean (locally)
- Averaging produces local biases

Smoothing of S&P Noise

- It's not zero mean (locally)
- Averaging produces local biases

Median Filtering

Median 3x3

Median 5x5

Median Filtering

Median 3x3

Median 5x5

Median Filtering

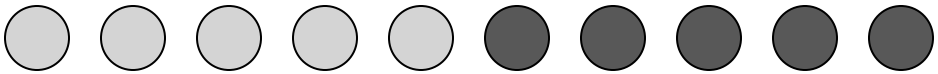
- Iterate

Median 3x3

2x Median 3x3

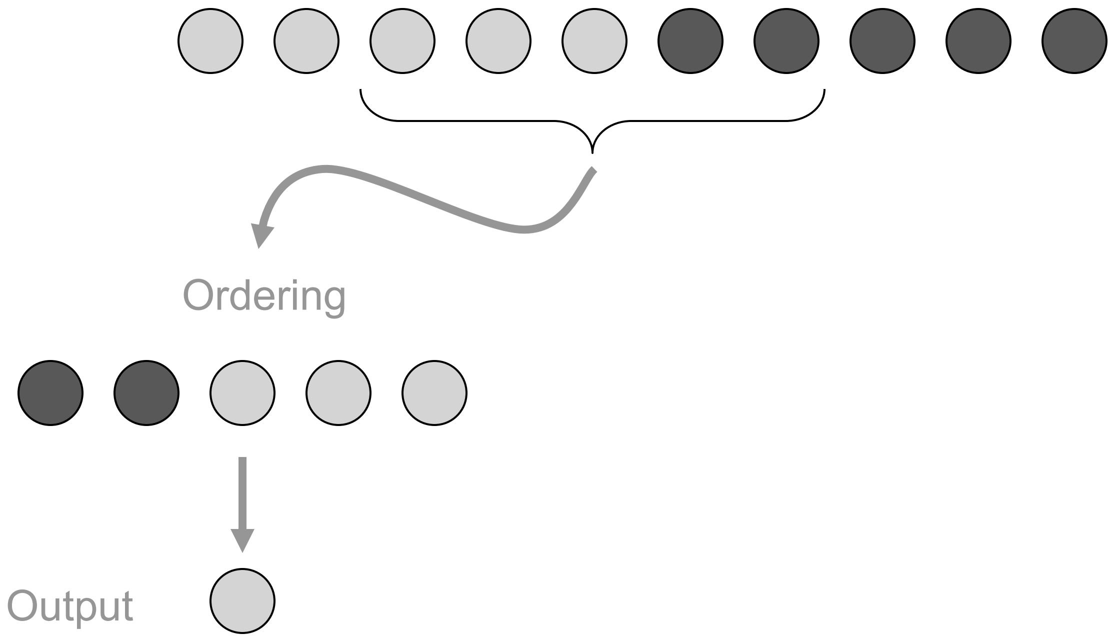
Median Filtering

- Image model: piecewise constant (flat)



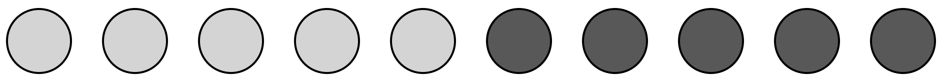
Median Filtering

- Image model: piecewise constant (flat)



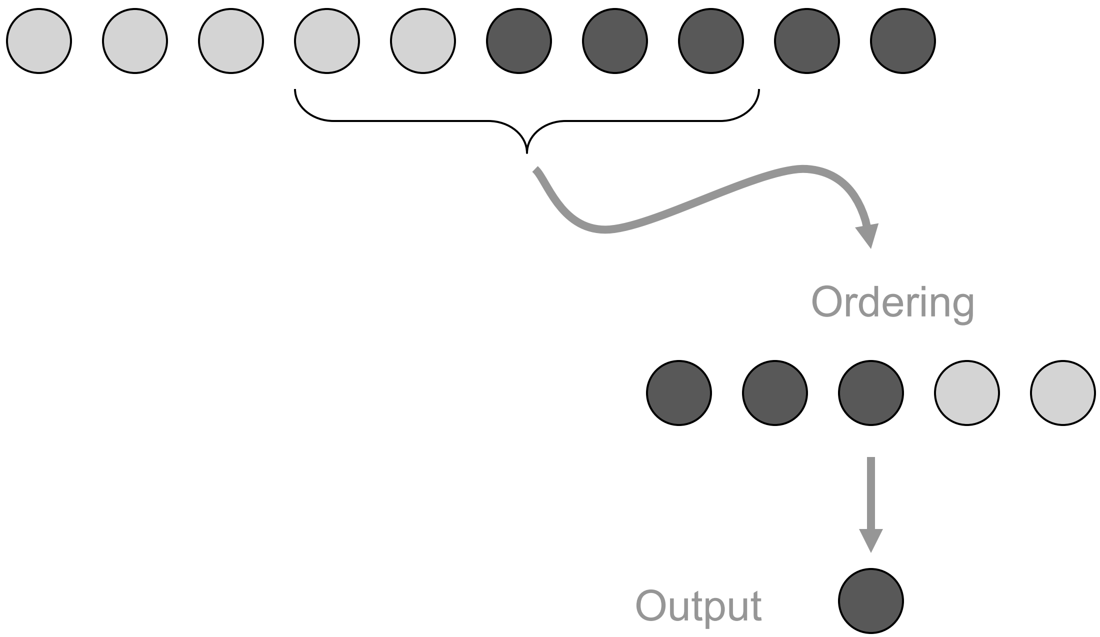
Median Filtering

- Image model: piecewise constant (flat)



Median Filtering

- Image model: piecewise constant (flat)



Order Statistics

- Median is special case of order-statistics filters
- Instead of weights based on neighborhoods, weights are based on ordering of data

Neighborhood

$$X_1, X_2, \ldots, X_N$$

Ordering

$$X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(N)}$$

Filter

$$F(X_1, X_2, \ldots, X_N) = \alpha_1 X_{(1)} + \alpha_2 X_{(2)} + \ldots + \alpha_N X_{(N)}$$

Neighborhood average (box)

$$\alpha_i = 1/N$$

Median filter

$$\alpha_i = \begin{cases} 1 & i = (N+1)/2 \\ 0 & \text{otherwise} \end{cases}$$

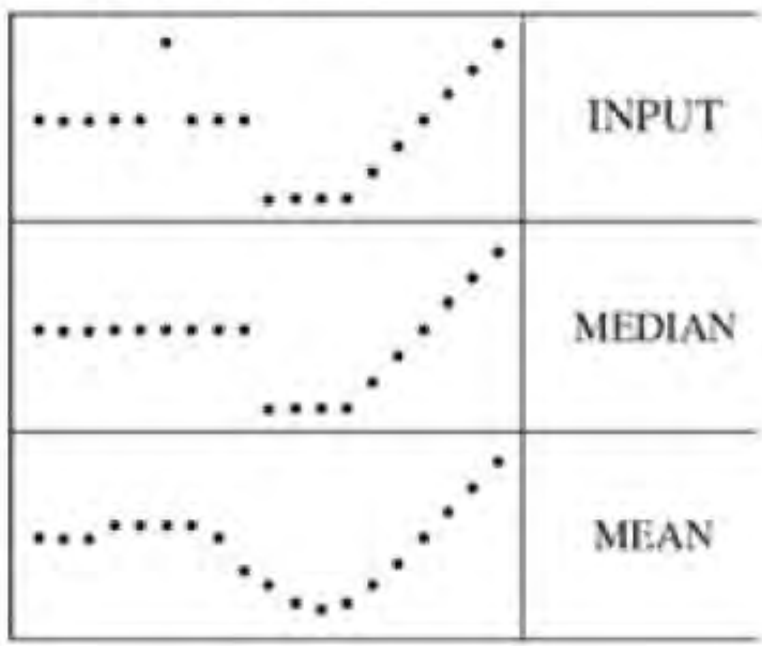
Trimmed average (outlier removal)

$$\alpha_i = \begin{cases} 1/M & (N-M+1)/2 \leq i \leq (N+M+1)/2 \\ 0 & \text{otherwise} \end{cases}$$

Median filter

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

filters have width 5 :



Source: K. Grauman

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf

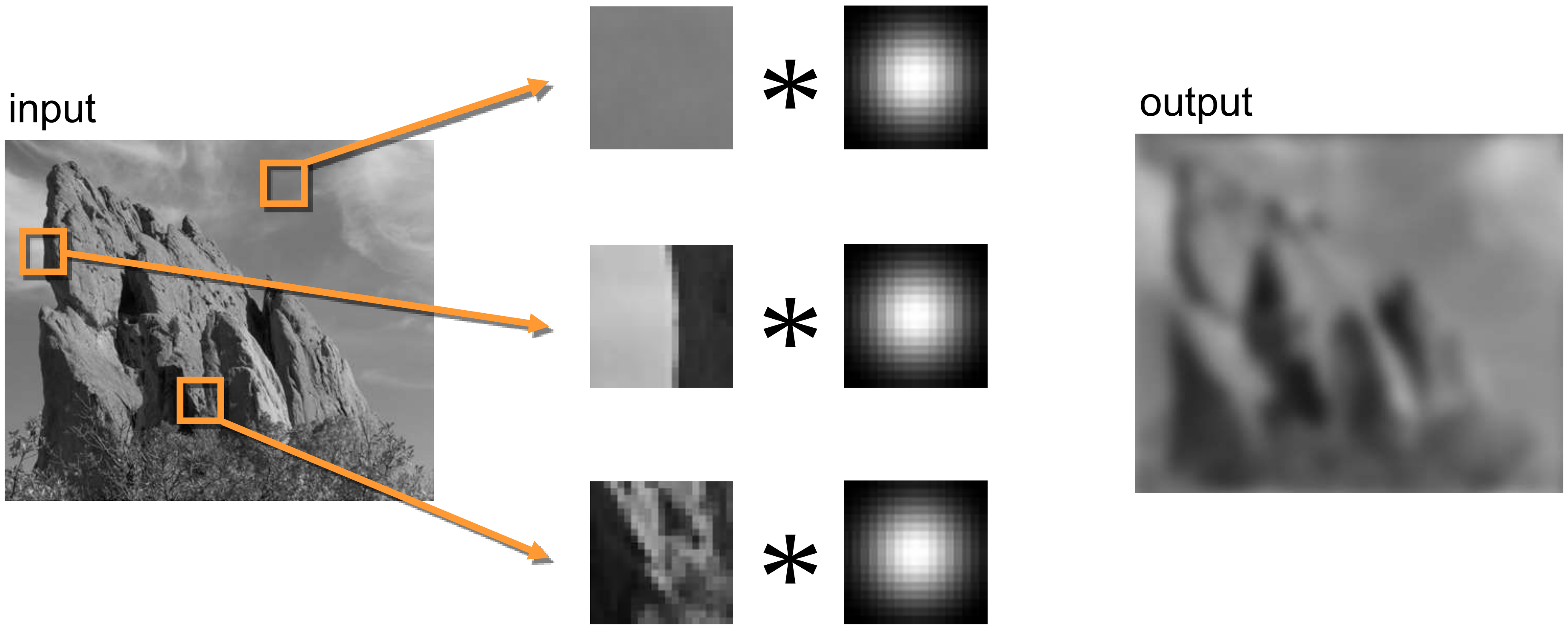
Piecewise Flat Image Models

- Image piecewise flat -> average only within similar regions
- Problem: don't know region boundaries

Piecewise-Flat Image Models

- Assign probabilities to other pixels in the image belonging to the same region
- Two considerations
  - Distance: far away pixels are less likely to be same region
  - Intensity: pixels with different intensities are less likely to be same region

Gaussian: Blur Comes from Averaging across Edges

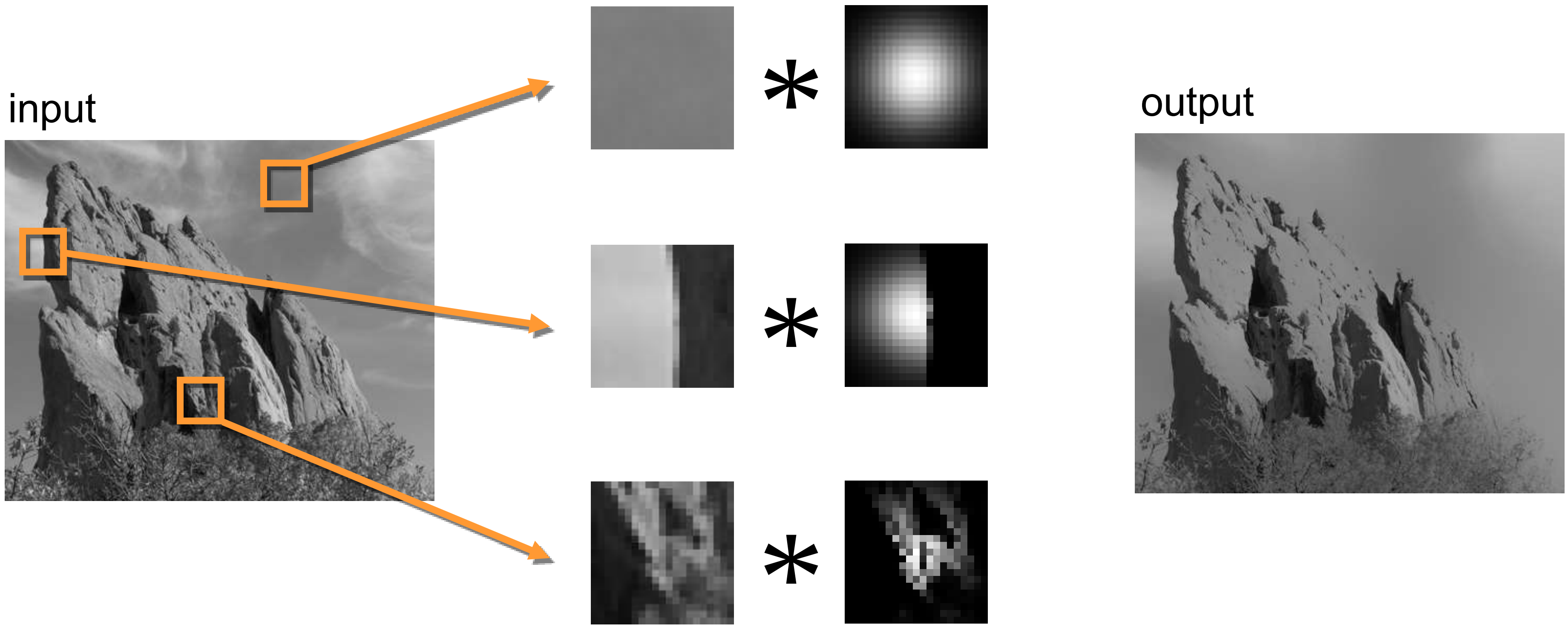


Same Gaussian kernel everywhere.

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt

Bilateral Filter
No Averaging across Edges

[Aurich 95, Smith 97, Tomasi 98]



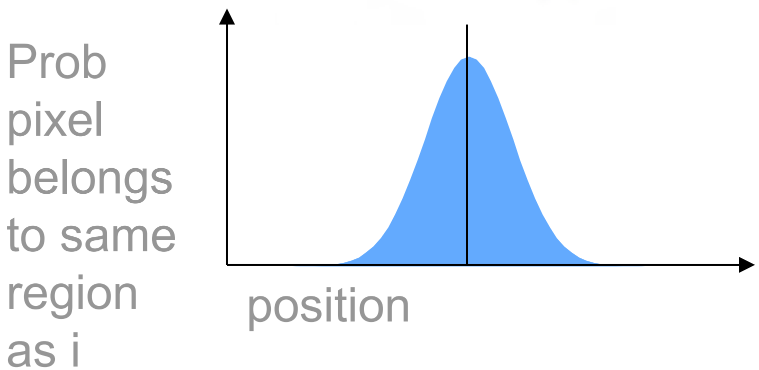
The kernel shape depends on the image content.

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt

Main Idea

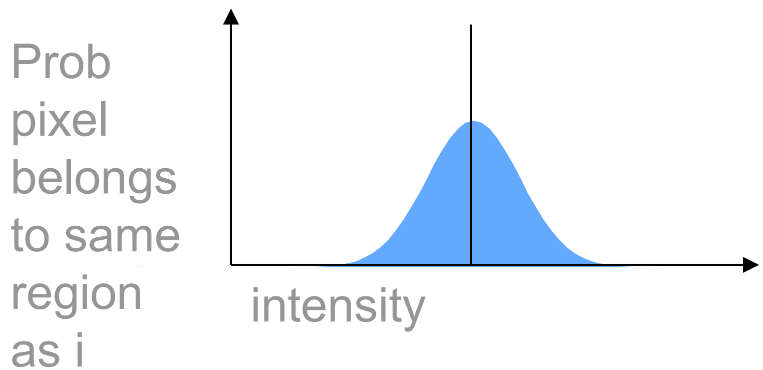
Distance (kernel/pdf)

$$G(\mathbf{x}_i - \mathbf{x}_j)$$



Distance (pdf)

$$H(f_i - f_j)$$



Bilateral Filter Definition: an Additional Edge Term

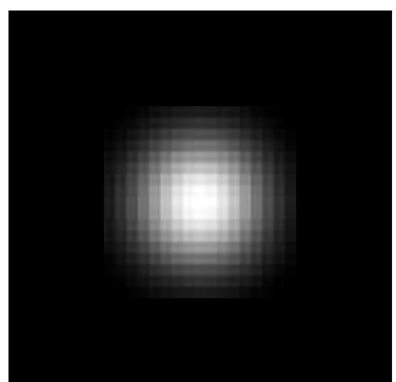
Same idea: **weighted average of pixels**.

new | not new | new

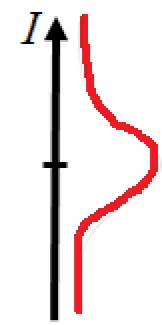
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

normalization factor

space weight



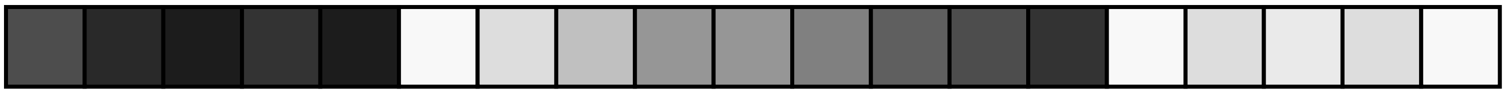
range weight



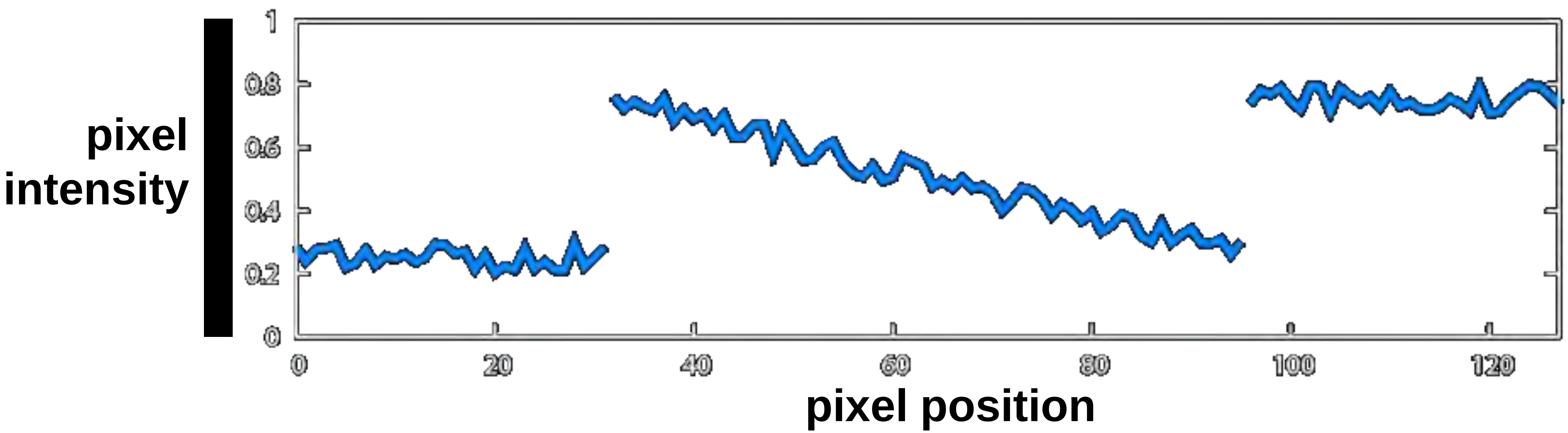
Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt

Illustration a 1D Image

- 1D image = line of pixels



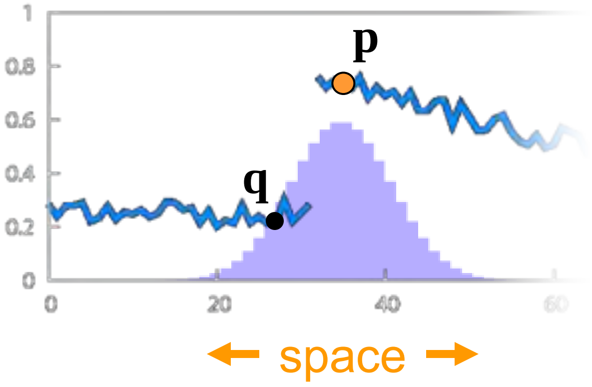
- Better visualized as a plot



Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt

Gaussian Blur and Bilateral Filter

Gaussian blur



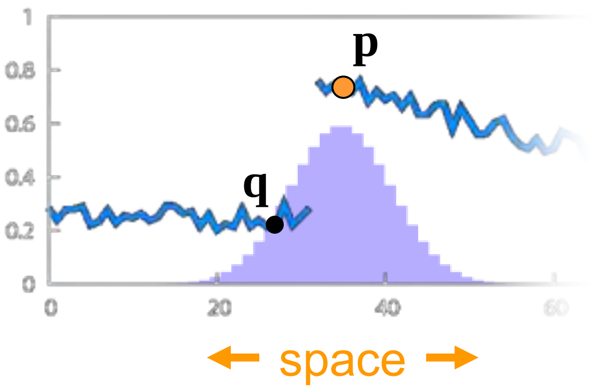
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\| \mathbf{p} - \mathbf{q} \|) I_{\mathbf{q}}$$

space

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt

Gaussian Blur and Bilateral Filter

Gaussian blur

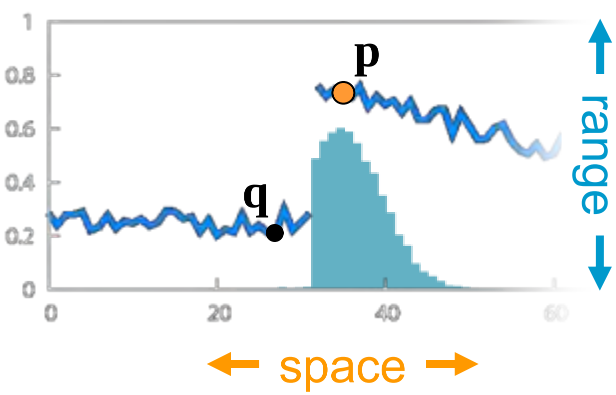


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\| \mathbf{p} - \mathbf{q} \|)\, I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|)\, G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |)\, I_{\mathbf{q}}$$

space range

normalization

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt

Bilateral Filter

- Neighborhood – sliding window
- Weight contribution of neighbors according to:

$$f_i \leftarrow k_i^{-1} \sum_{j \in N} f_j G(\mathbf{x}_i - \mathbf{x}_j) H(f_i - f_j)$$

$$k_i = \sum_{j \in N} G(\mathbf{x}_i - \mathbf{x}_j) H(f_i - f_j)$$

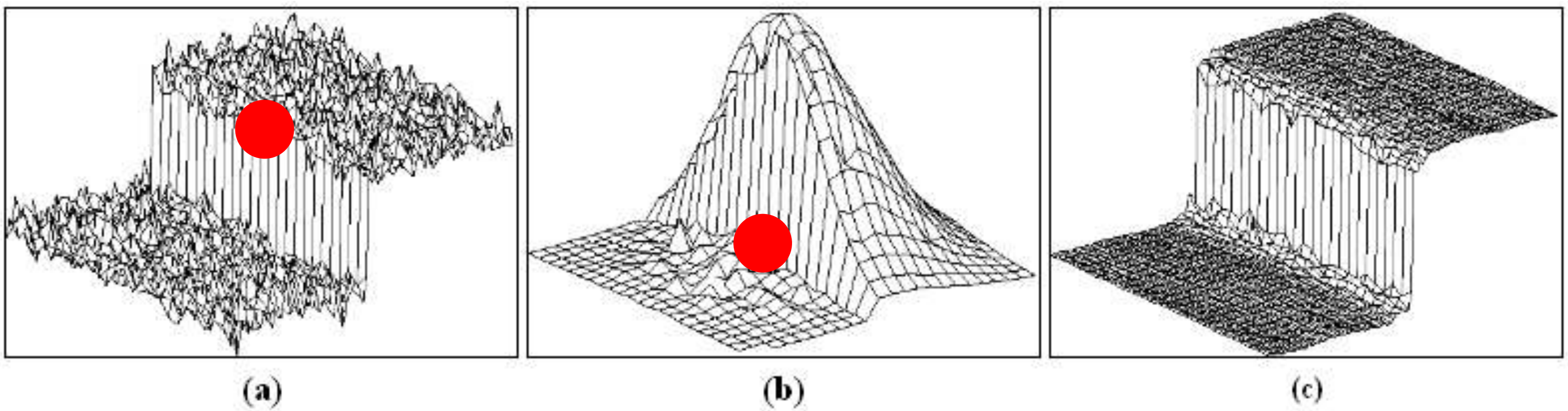
normalization: all weights add up to 1

- G is a Gaussian (or lowpass), as is H, N is neighborhood,
  - Often use $G(r_{ij})$ where r_{ij} is distance between pixels
  - Update must be normalized for the samples used in this (particular) summation
- Spatial Gaussian with extra weighting for intensity
  - Weighted average in neighborhood with downgrading of intensity outliers

Tomasi, Manduchi: http://en.wikipedia.org/wiki/Bilateral_filter
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html

Bilateral Filter

Replaces the pixel value at **x** with an average of similar and nearby pixel values.



When the bilateral filter is centered, say, on a pixel on the bright side of the boundary, the similarity function *s* assumes values close to one for pixels on the same side, and values close to zero for pixels on the dark side. The similarity function is shown in figure 1(b) for a 23x23 filter support centered two pixels to the right of the step in figure 1(a).

Bilateral Filtering

Replaces the pixel value at **x** with an average of similar and nearby pixel values.

Gaussian Blurring

Bilateral

Bilateral Filtering

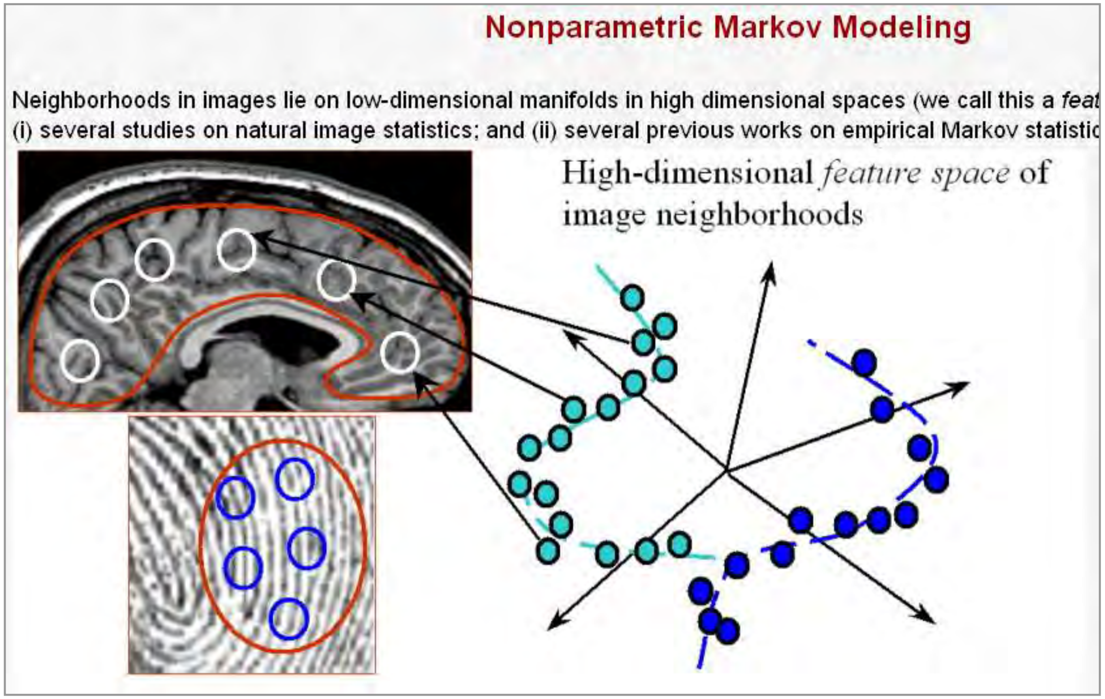
Gaussian Blurring

Bilateral

Nonlocal Averaging

- Recent algorithm
  - NL-means, Baudes et al., 2005
  - UINTA, Awate & Whitaker, 2005
- Different model
  - No need for piecewise-flat
  - Images consist of some set of pixels with similar neighborhoods → average several of those
    - Scattered around
      - General area of a pixel
      - All around
- Idea
  - Average sets of pixels with similar neighborhoods

UINTA: Unsupervised Information-Theoretic Adaptive Filtering : Excellent Introduction and Additional Readings (Suyash P. Awate)

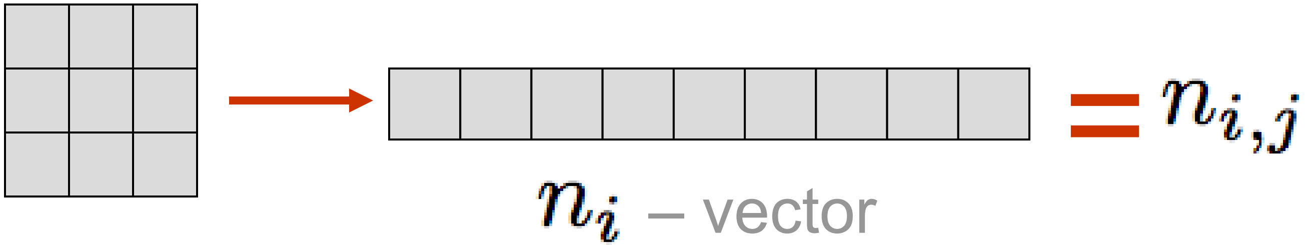


http://www.cs.utah.edu/~suyash/pubs/uinta/

Suyash P. Awate, Ross T. Whitaker
Unsupervised, Information-Theoretic, Adaptive Image Filtering with Applications to Image Restoration
IEEE Trans. Pattern Analysis & Machine Intelligence (TPAMI) 2006, Vol. 28, Num. 3, pp. 364-376

Nonlocal Averaging

- Strategy:
  - Average pixels to alleviate noise
  - Combine pixels with similar neighborhoods
- Formulation
  - $n_{i,j}$ – vector of pixels values, indexed by j, from neighborhood around pixel i



Nonlocal Averaging Formulation

- Distance between neighborhoods

$$d_{i,k} = d(n_i, n_k) = ||n_i - n_k|| = \left(\sum_{j=1}^{N} (n_{i,j} - n_{k,j})^2 \right)^{\frac{1}{2}}$$

- Kernel weights based on distances

$$w_{i,k} = K(d_{i,k}) = e^{-\frac{d_{i,k}^2}{2\sigma^2}}$$

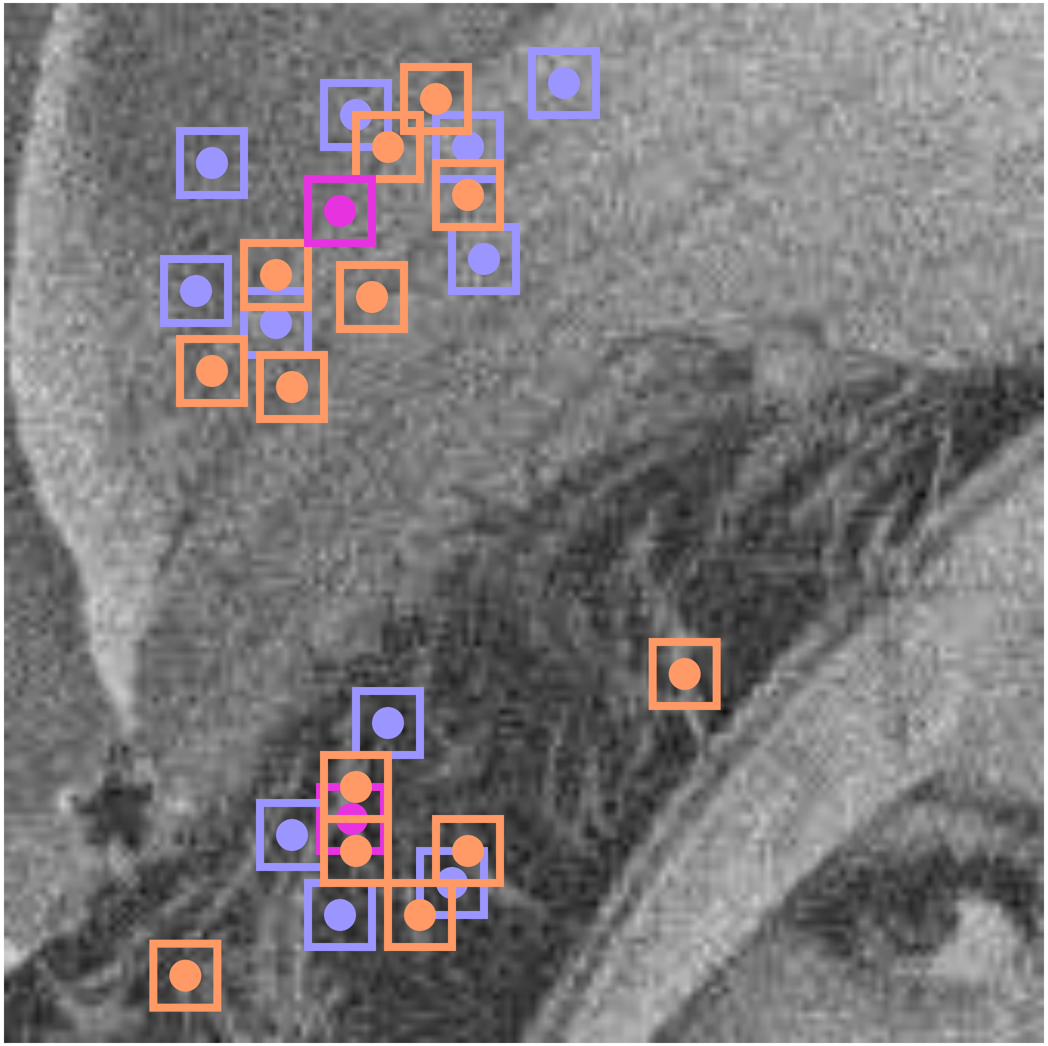
- Pixel values of k neighborhoods: f_k

Averaging Pixels Based on Weights

- For each pixel, i, choose a set of pixel locations k:
  - k = 1, ..., M
  - Average them together based on neighborhood weights (prop. to intensity pattern difference)

$$g_i \leftarrow \frac{1}{\sum_{k=1}^{M} w_{i,k}} \sum_{k=1}^{M} w_{i,k} f_k$$

Nonlocal Averaging



Some Details

- Window sizes: good range is 5x5->11x11
- How to choose samples:
  - Random samples from around the image
    - UINTA, Awate&Whitaker
  - Block around pixel (bigger than window, e.g. 51x51)
    - NL-means
- Iterate
  - UNITA: smaller updates and iterate

NL-Means Algorithm

- For each pixel, p
  - Loop over set of pixels nearby
  - Compare the neighborhoods of those pixels to the neighborhood of p and construct a set of weights
  - Replace the value of p with a weighted combination of values of other pixels
- Repeat… but 1 iteration is pretty good

Results

Noisy image (range 0.0-1.0)

Bilateral filter (3.0, 0.1)

Results

Bilateral filter (3.0, 0.1)

NL means (7, 31, 1.0)

Results

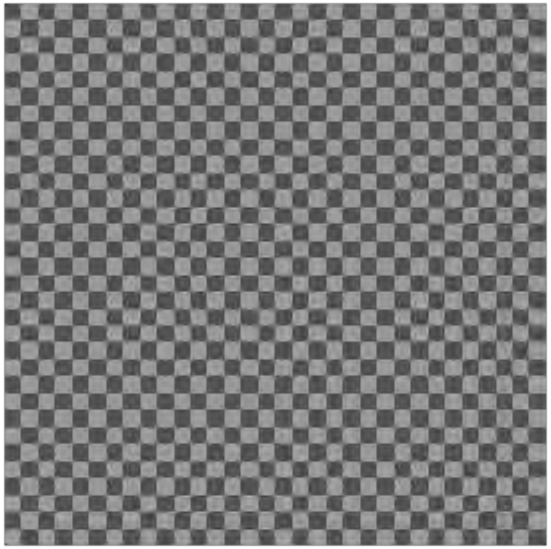
Bilateral filter (3.0, 0.1)

NL means (7, 31, 1.0)

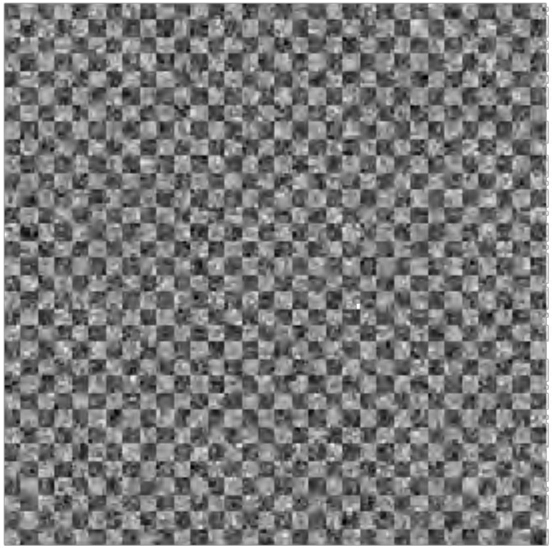
Less Noisy Example

Less Noisy Example

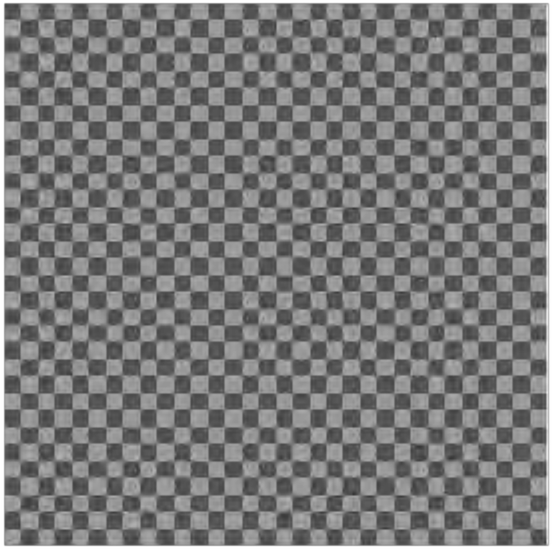
Results



Original

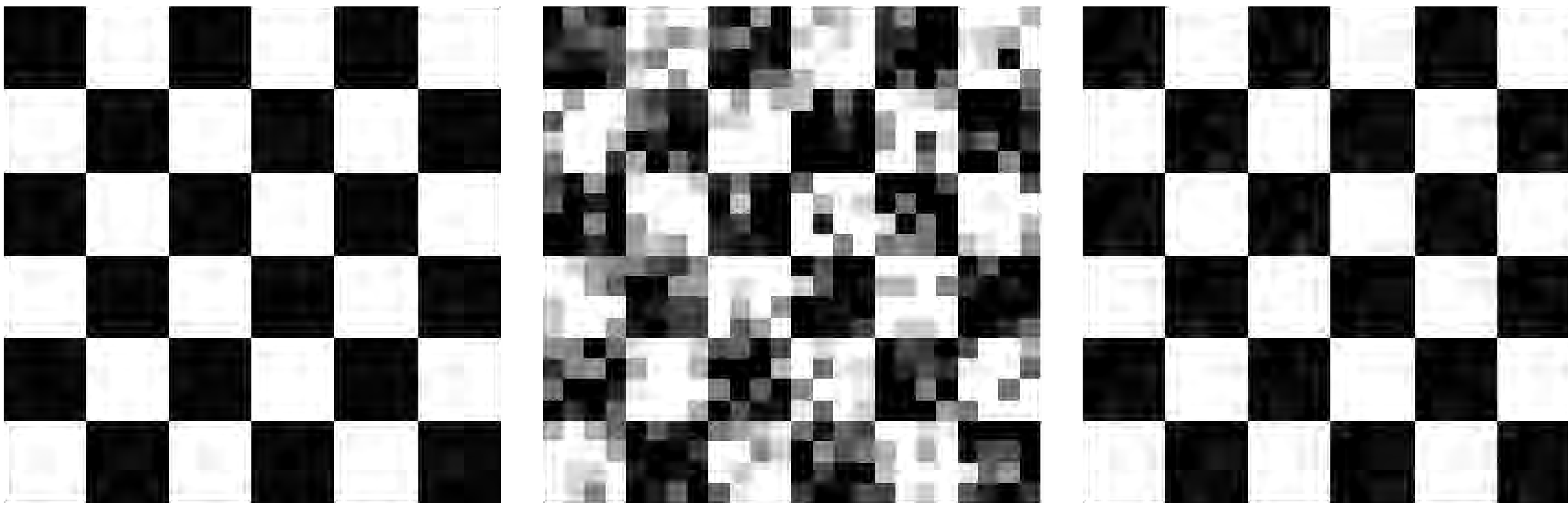


Noisy



Filtered

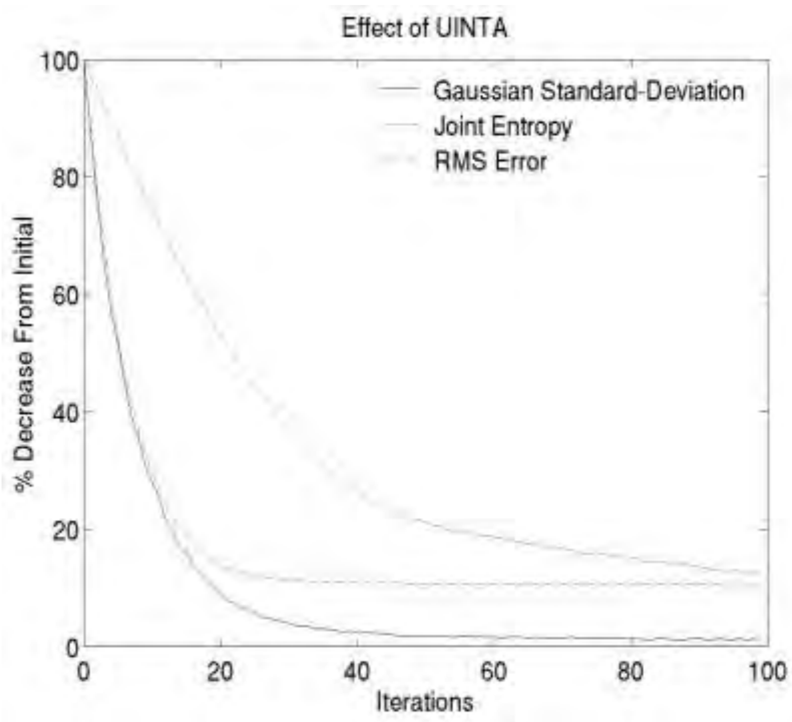
Checkerboard With Noise

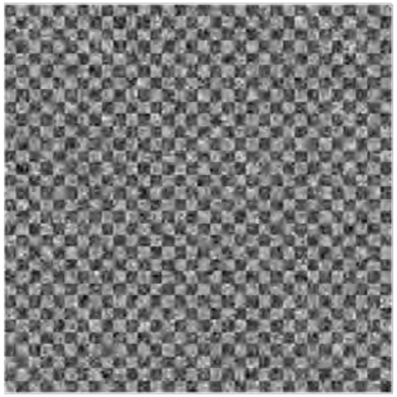


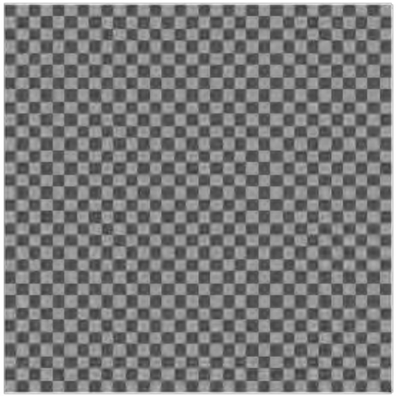
Original Noisy Filtered

Quality of Denoising

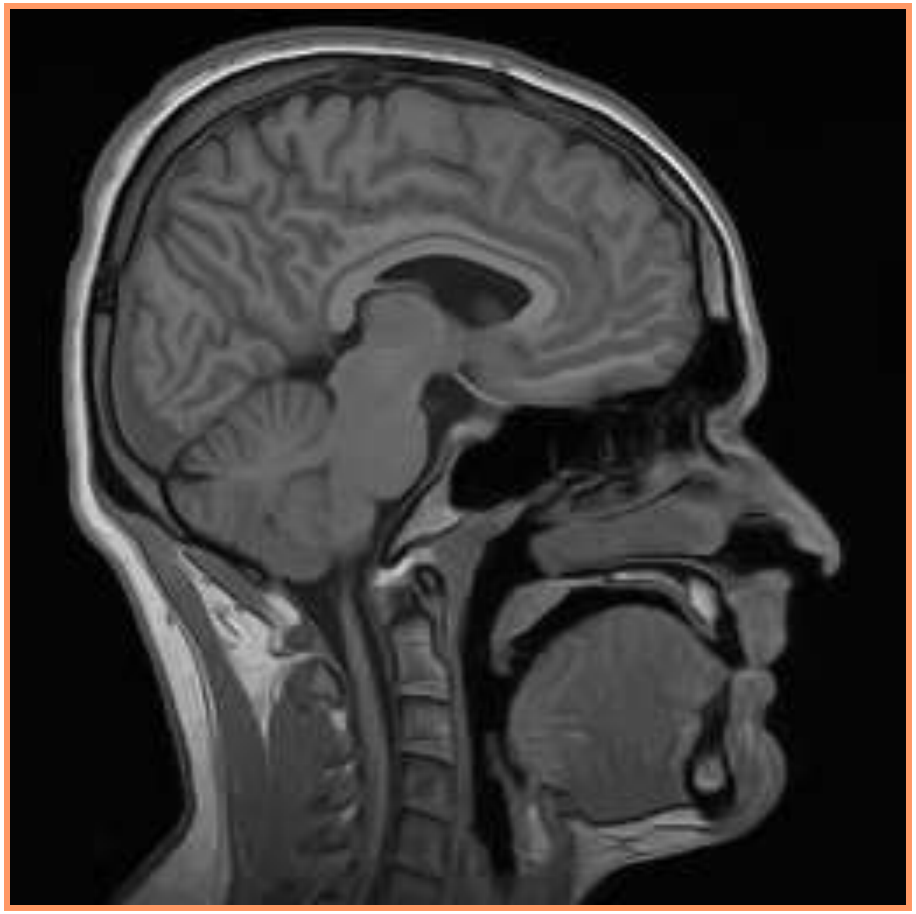
- ♦, joint entropy, and RMS- error vs. number of iterations



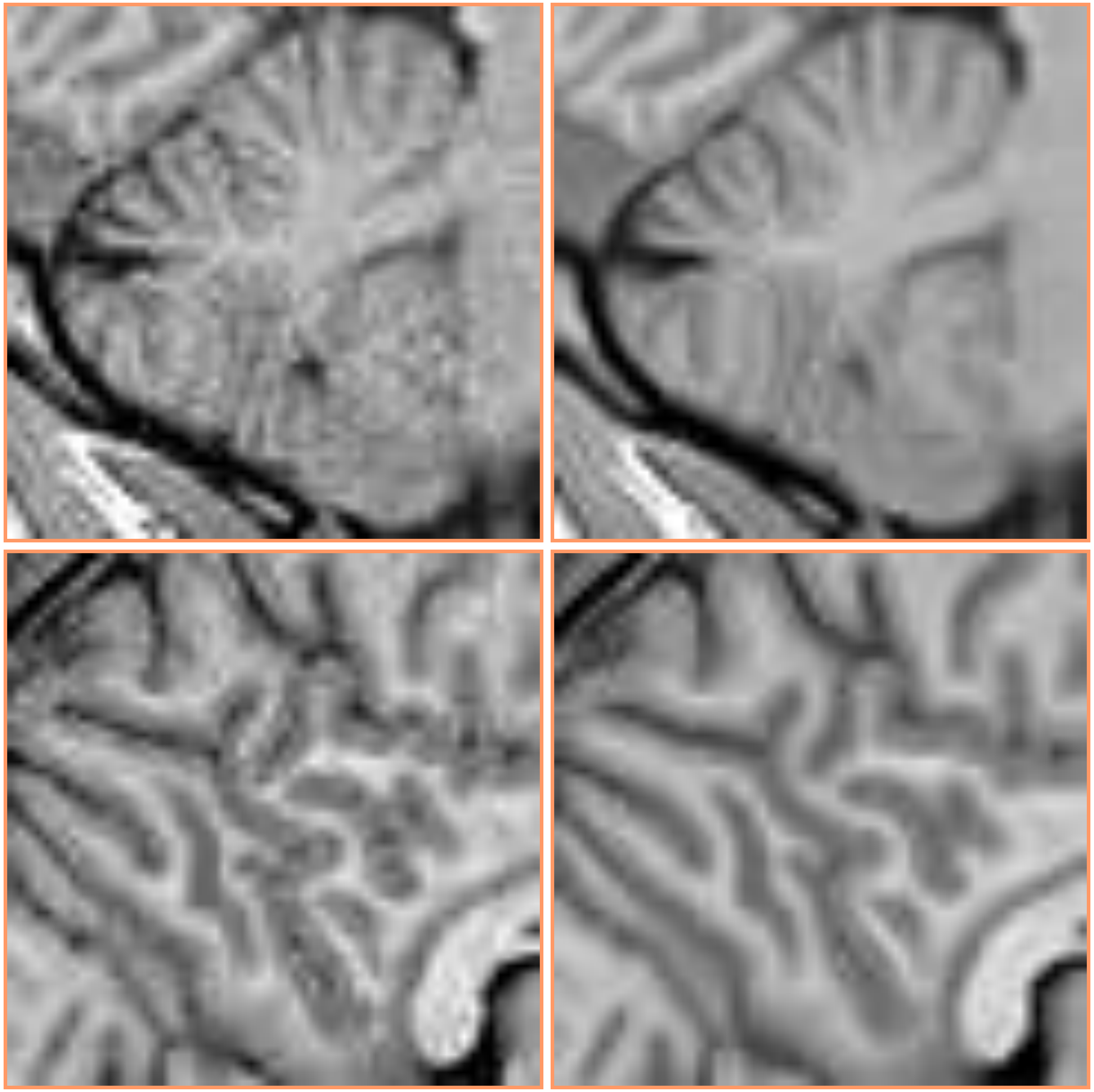




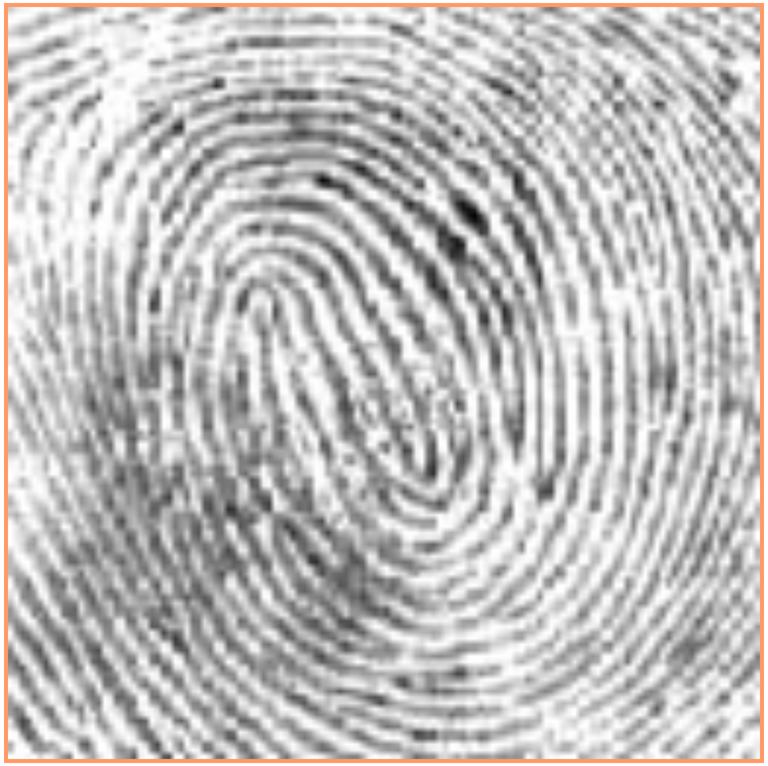
MRI Head



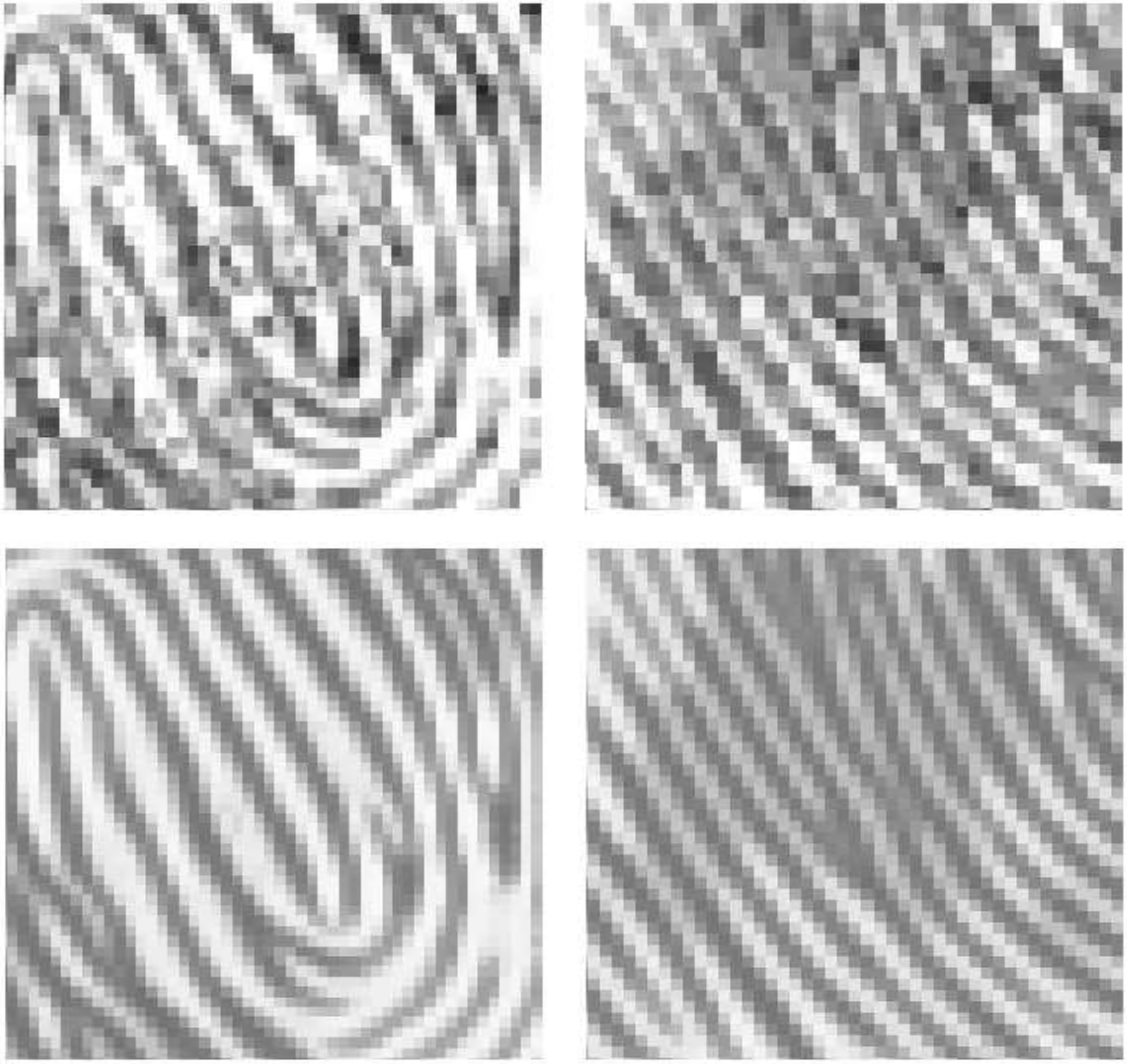
MRI Head



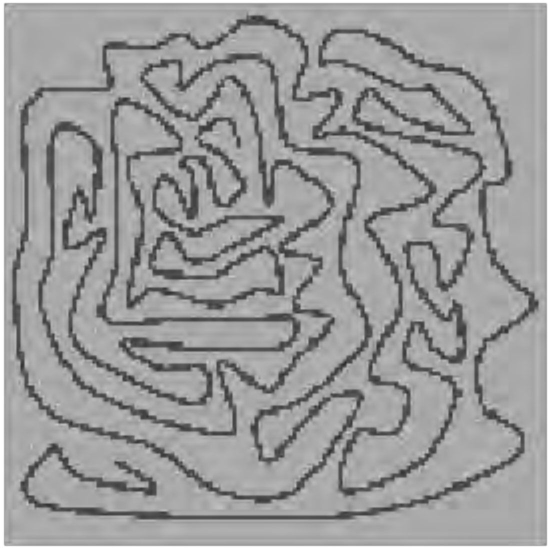
Fingerprint



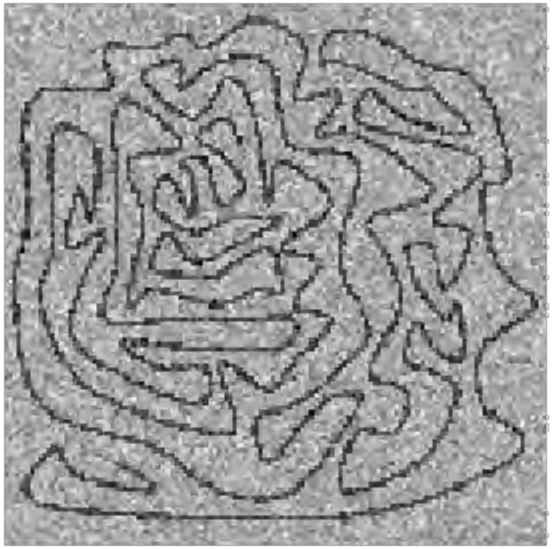
Fingerprint



Results



Original

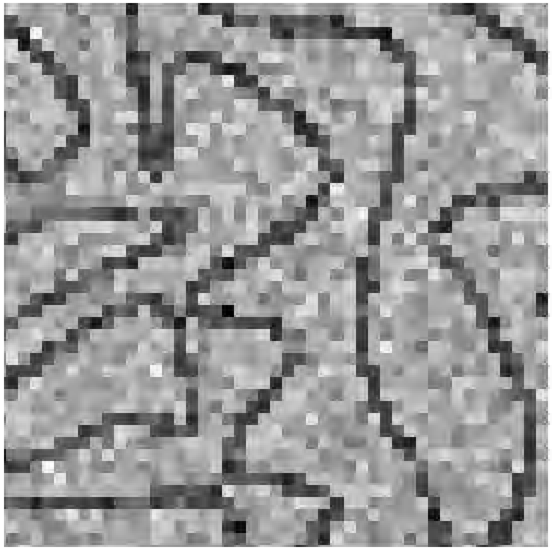


Noisy

Filtered

Results

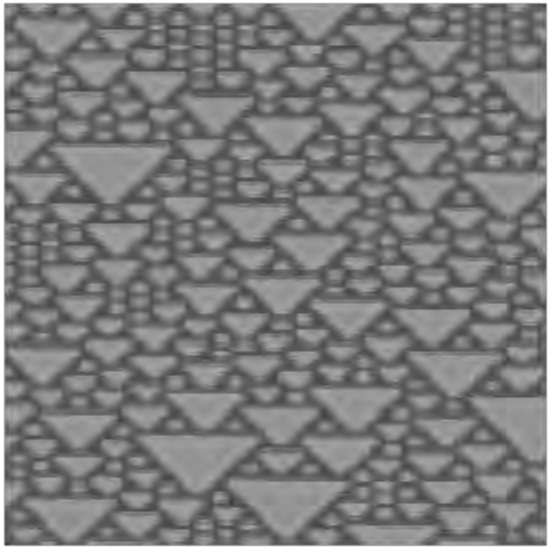
Original



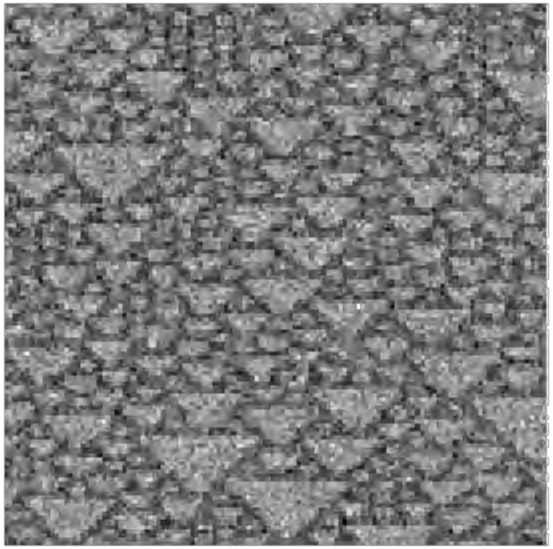
Noisy

Filtered

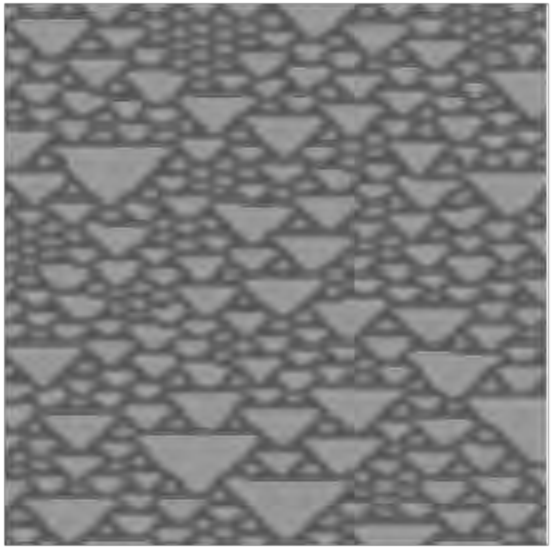
Results



Original

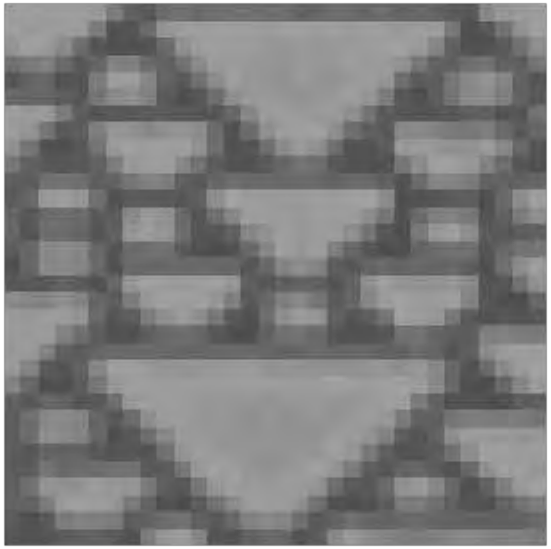


Noisy

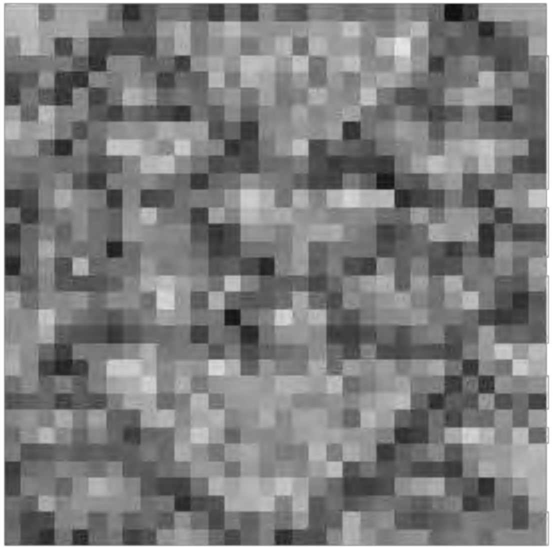


Filtered

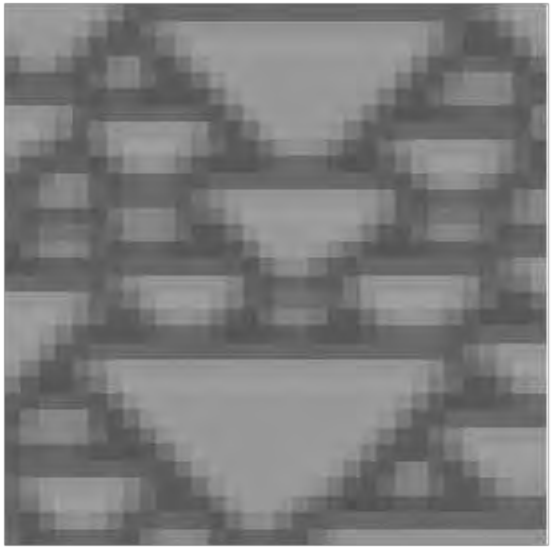
Fractal



Original



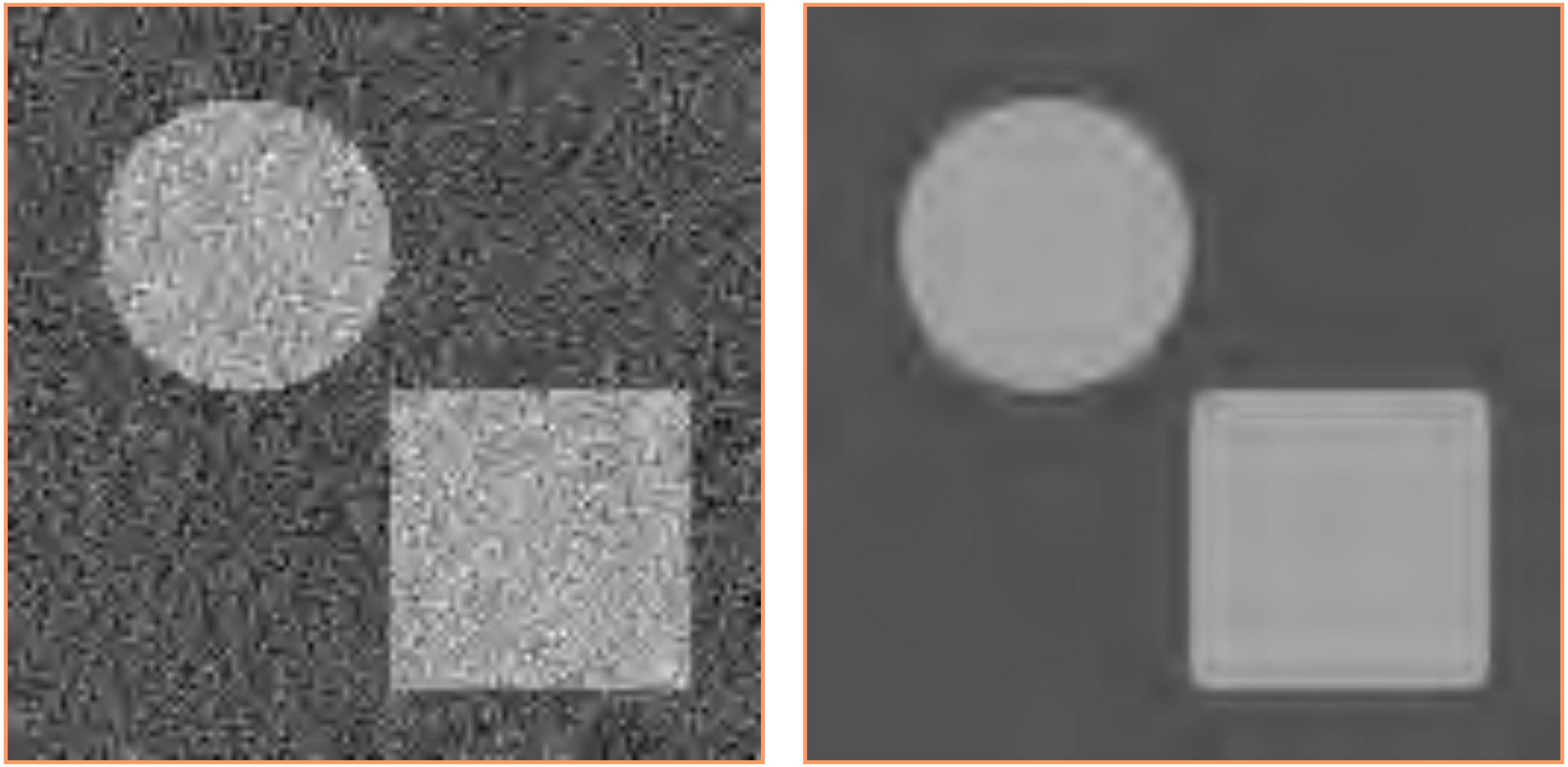
Noisy



Filtered

Piecewise Constant

- Several 10s of Iterations
- Tends to obliterate rare events (e.g. corners)



Texture, Structure