Biomedical Informatics 260

Image Filtering

Lecture 4

David Paik, PhD

Spring 2019

Last Lecture: Image Segmentation

1. How to find semantically meaningful regions within an image
2. Algorithms that operate directly on image pixel values
3. But we can add some processing steps to provide additional image features beyond the pixel intensity

Today: Image Filtering

- Goal is to develop a solid understanding of the fundamentals leading to insightful intuition about imaging algorithms
- What is image filtering?
  - Image in, image out
- Why perform filtering?
  - Filter out unwanted noise
  - Detect desirable features
- Methods:
  - Convolution
  - Non-linear Convolution-like Methods
  - Diffusion Methods

Fourier Analysis

Fourier Series

Periodic signals can be decomposed into sine waves

$$s(x) = \frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n cos\left(\frac{2\pi nx}{P}\right) + b_n sin\left(\frac{2\pi nx}{P}\right) \right)$$

or alternatively,

$$s(x) = \sum_{n=-N}^{N} c_n e^{i\frac{2\pi nx}{P}}$$

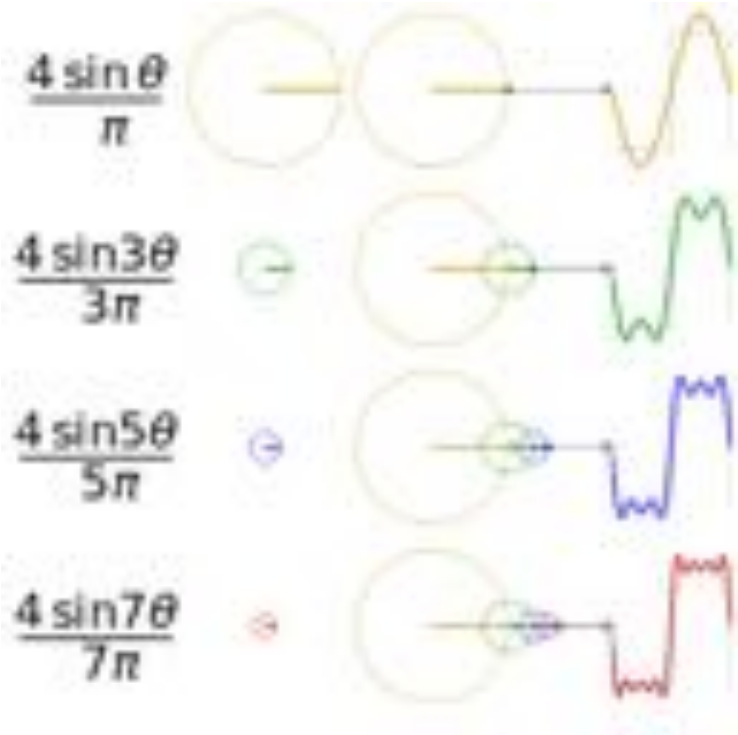
How do you compute the coefficients?

$$a_n = \int_0^P s(x) cos\left(\frac{2\pi nx}{P}\right) dx$$

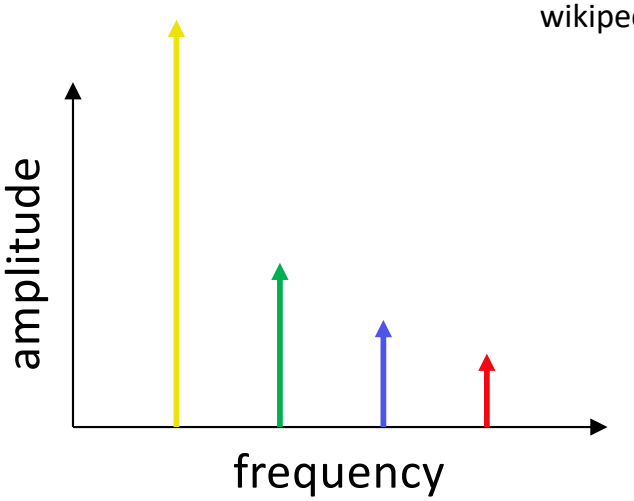
$$b_n = \int_0^P s(x) sin\left(\frac{2\pi nx}{P}\right) dx$$

or alternatively,

$$c_n = \int_0^P s(x) e^{-i\frac{2\pi nx}{P}} dx$$



wikipedia



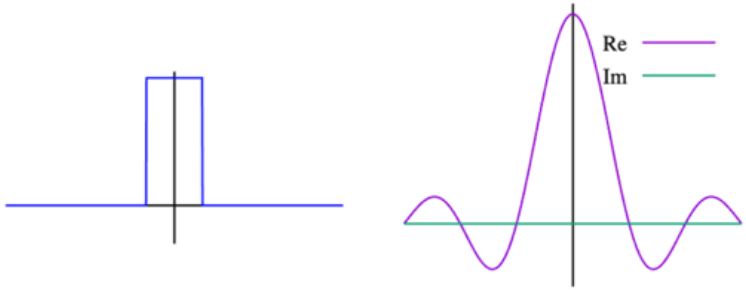
Fourier Transform

Aperiodic signals can be decomposed into a continuum of sine waves

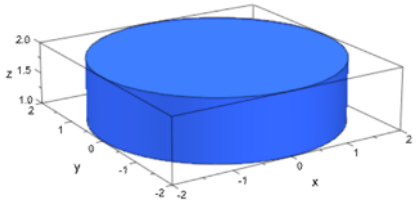
$$f(x) = \int_{-\infty}^{\infty} \mathrm{F}(u) \cdot e^{i2\pi ux} dx$$

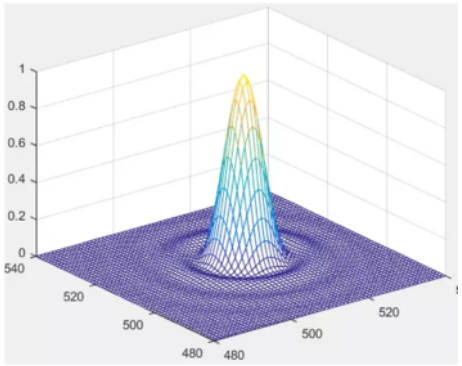
multi-dimensional:

$$f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{F}(u,v) \cdot e^{i2\pi(ux+vy)} dx\, dy$$



$f(x)$ $F(u)$





$f(x,y)$ $F(u,v)$

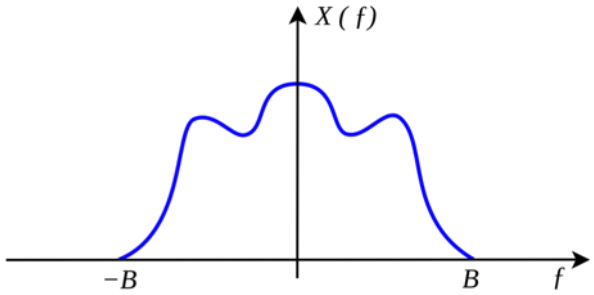
How do you find the amplitude at each frequency?

$$\mathrm{F}(u) = \int_{-\infty}^{\infty} f(x) \cdot e^{-i2\pi ux} dx$$

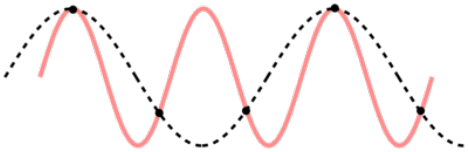
$$\mathrm{F}(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \cdot e^{-i2\pi(ux+vy)} dx\, dy$$

Nyquist-Shannon Sampling Theorem

If a signal $s(x)$ contains no frequencies higher than B, it is completely determined by sampling with a frequency $f_s \geq 2B$

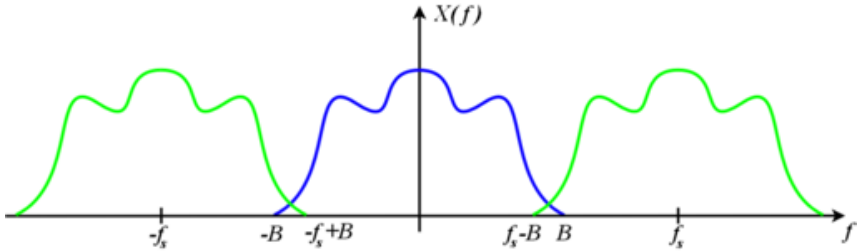


Aliasing



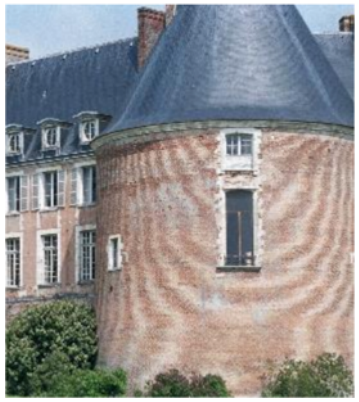
Sampling rate too low

One frequency looks like another

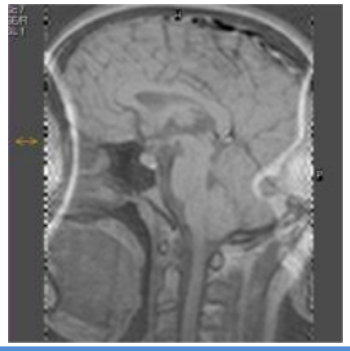


In Fourier domain, sampling creates duplicates spaced f_s apart. They overlap if less than Nyquist frequency

temporal aliasing



spatial aliasing



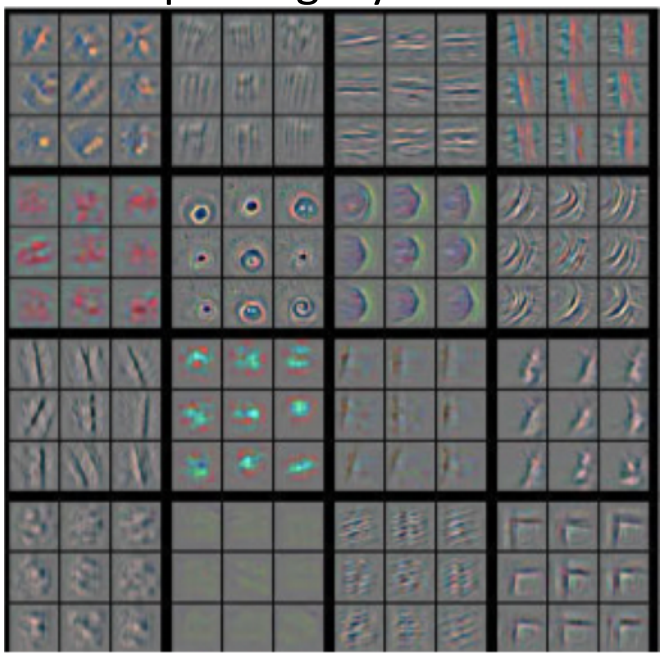
wikipedia

Downsampling, Upsampling, & Aliasing

- Upsampling (as we saw in visualization) uses interpolation
  - Low order methods are generally ok

- Downsampling can cause issues if the lower sampling rate is below the Nyquist rate
  - Limit the bandwidth of the signal (low pass filter) before sampling
  - In deep learning, the integer multiple of sampling period is called the stride

Features after stride 4 pooling layer



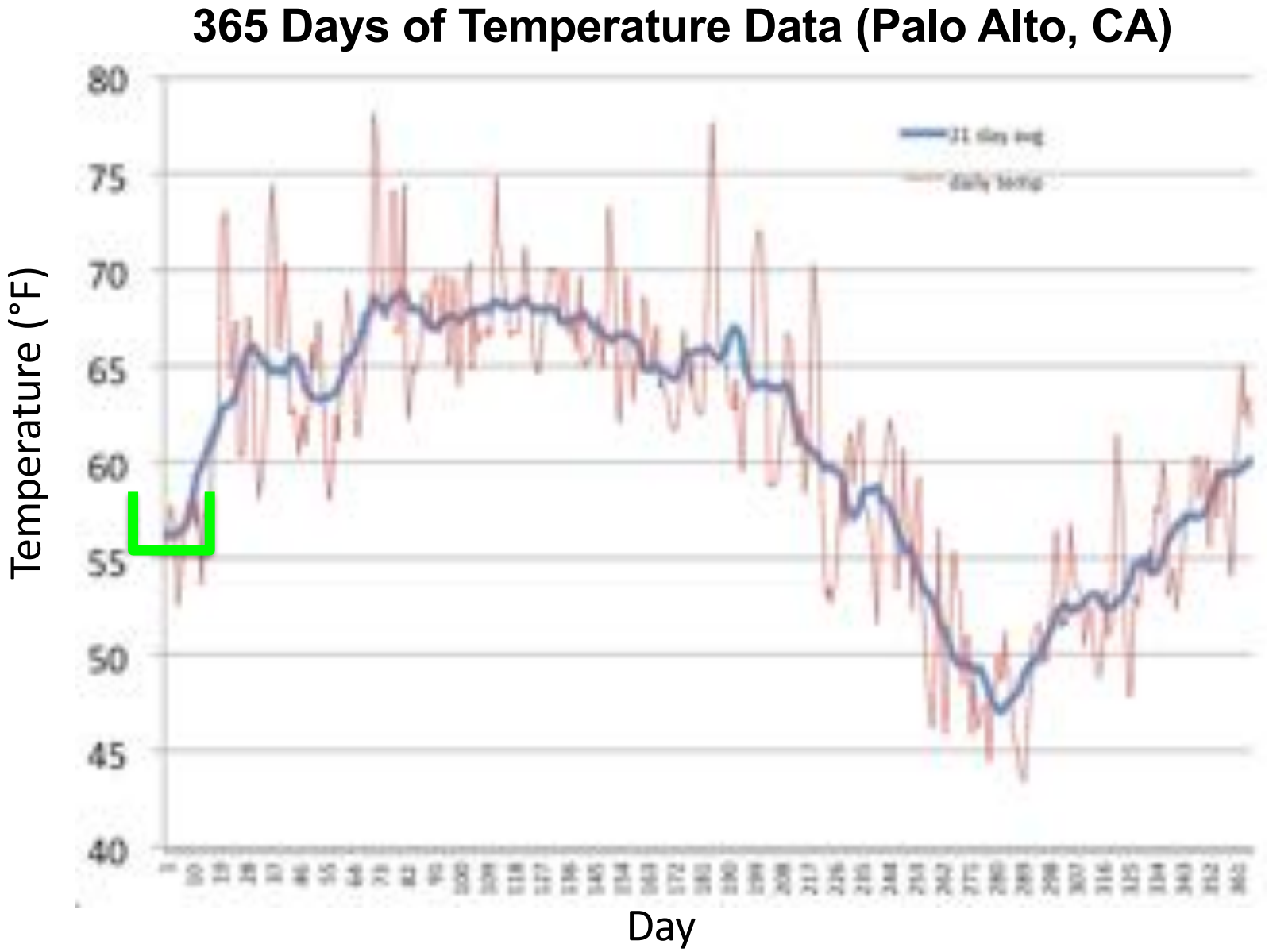
Zeiler and Fergus, ECCV 2014

Convolution and Smoothing

How does one “smooth” data
in order to reduce noise?

Noise Reduction by Moving Window Averages

(AKA 1D convolution)



Continuous Domain 1D Convolution

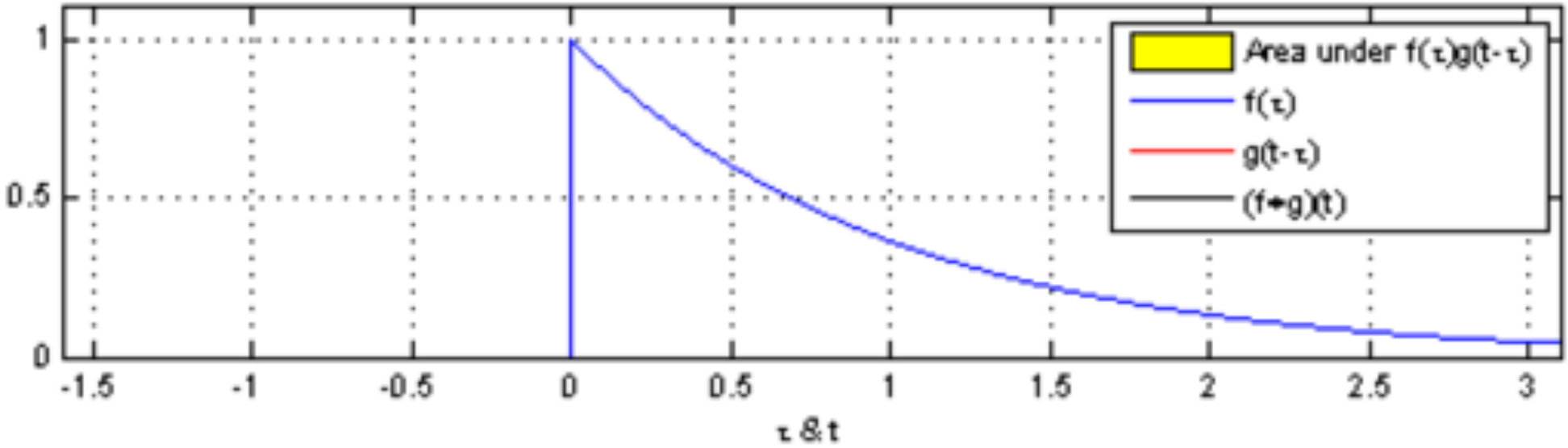
$$f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

data kernel

area under the curve

Kernel is usually compact in its support (i.e., spatial extent)

$f(t)*g(t)$ is a modified (or improved) version of $f(t)$



wikipedia.org

Continuous Domain 2D Convolution

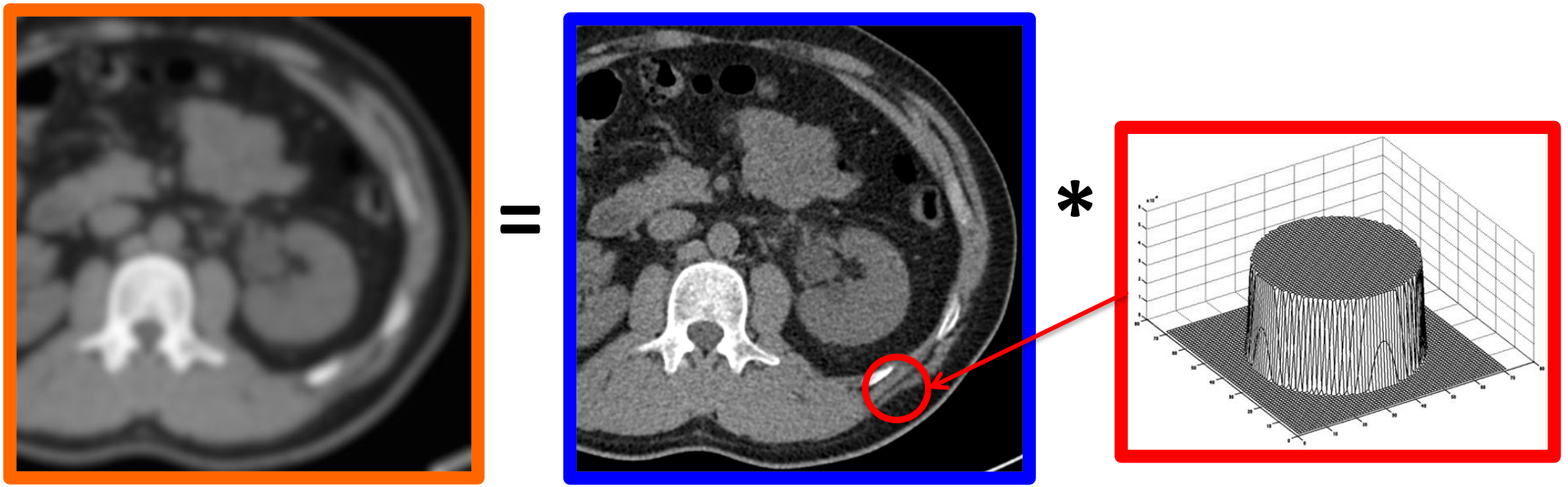
$$f(x,y) * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')g(x-x',y-y')\,dx'\,dy'$$

result

data

kernel

volume under the surface



Discrete 2D Convolution

$$O = K * I$$

3x3 Averaging Kernel, *K*

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

5	3	2	7	4	6	8	7	4	4
7	6	4	7	4	8	9	6	3	7
9	4	1	4	8	5	2	1	7	5
2	5	8	6	3	5	8	7	4	2
2	5	7	5	3	4	6	7	7	4
3	6	8	6	3	3	5	7	5	2
2	4	7	9	8	5	4	3	6	7
6	6	9	8	1	3	6	9	8	0
7	5	4	8	2	3	7	5	9	3
5	4	3	9	6	7	3	7	6	2

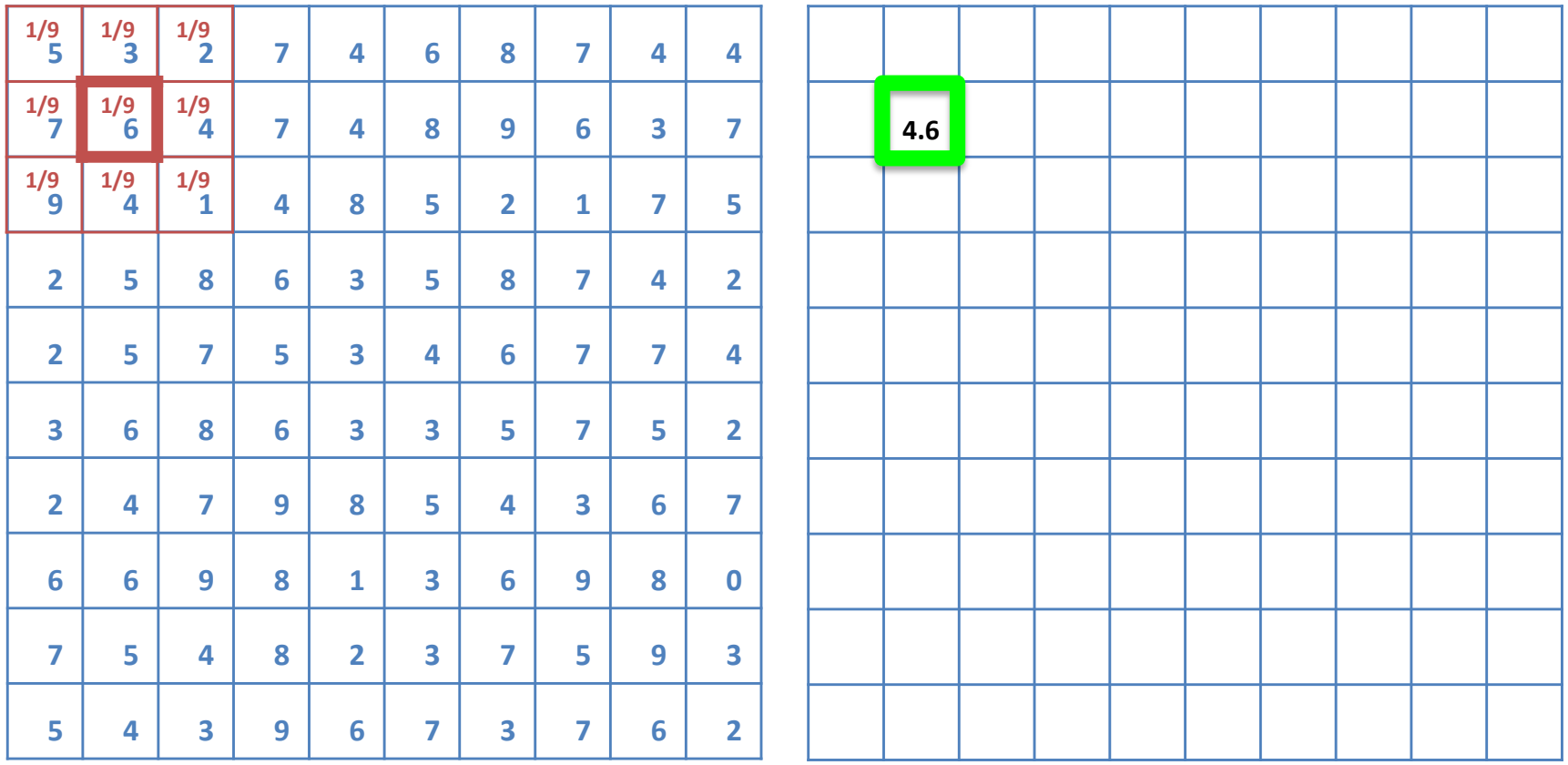
Input Image, *I*

Output Image, *O*

Discrete 2D Convolution

3x3 Averaging Kernel, *K*

$$O = K * I$$



Input Image, *I*

Output Image, *O*

Discrete 2D Convolution

3x3 Averaging Kernel, *K*

$$O = K * I$$

How would you handle the edges where part of the kernel is outside the image?

5	1/9 3	1/9 2	1/9 7	4	6	8	7	4	4
7	1/9 6	1/9 4	1/9 7	4	8	9	6	3	7
9	1/9 4	1/9 1	1/9 4	8	5	2	1	7	5
2	5	8	6	3	5	8	7	4	2
2	5	7	5	3	4	6	7	7	4
3	6	8	6	3	3	5	7	5	2
2	4	7	9	8	5	4	3	6	7
6	6	9	8	1	3	6	9	8	0
7	5	4	8	2	3	7	5	9	3
5	4	3	9	6	7	3	7	6	2

Input Image, *I*

	4.6	4.2							

Output Image, *O*

Example Boundary Conditions

- Constant
  - Pixels outside image are all constant, often 0
- Neumann boundary condition
  - Specifies derivative at the boundary, typically 0
  - Edge pixel values get extended beyond the image
- Periodic boundary condition
  - Edge of the image wraps around to the other side
- Or just let the result be smaller

Smoothing Kernels

(aka low pass filtering, noise reduction)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2D rectangular window
(boxcar or Dirichlet)

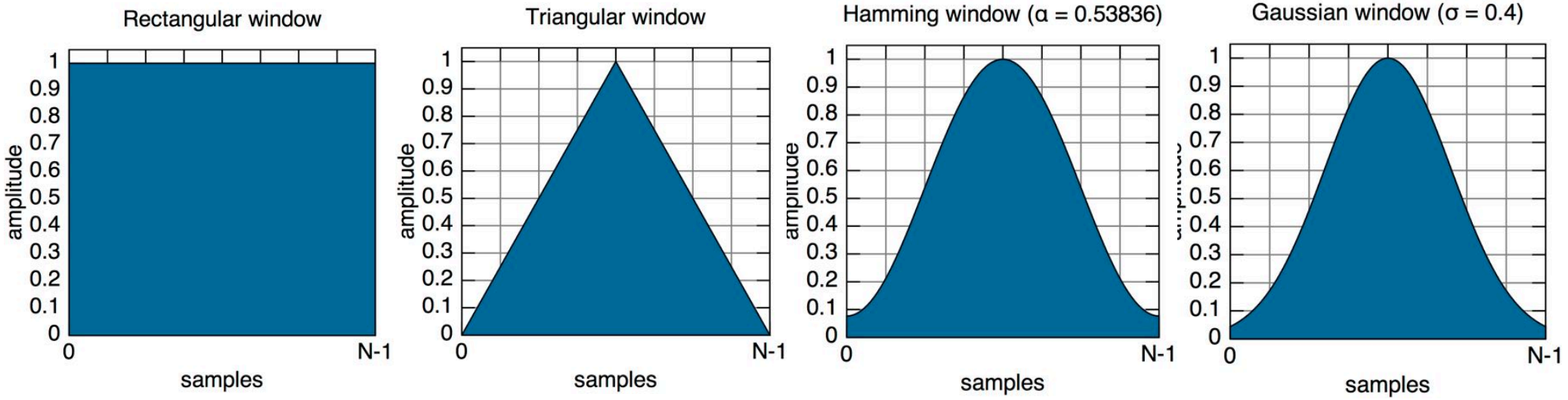
Size of the kernel can vary
(usually odd #)

Any dimensionality

Usually symmetric

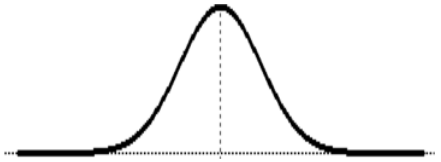
Steepness of drop-off can vary
(broader kernel = heavier smoothing)

Many variants exist (shown here in 1D):

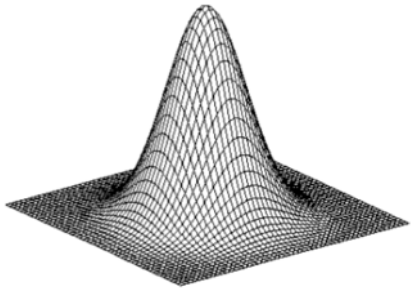


Gaussian Smoothing

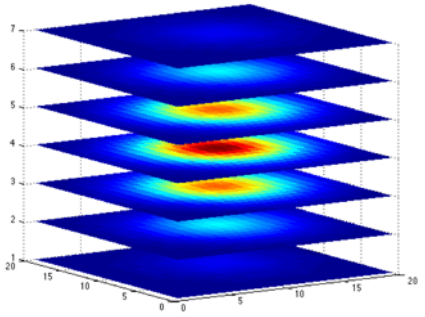
1D: $$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$



2D: $$G(x,y) = \frac{1}{\sigma^2 2\pi} e^{-\frac{\left(x^2+y^2\right)}{2\sigma^2}}$$



3D: $$G(x,y,z) = \frac{1}{\left(\sigma\sqrt{2\pi}\right)^3} e^{-\frac{\left(x^2+y^2+z^2\right)}{2\sigma^2}}$$

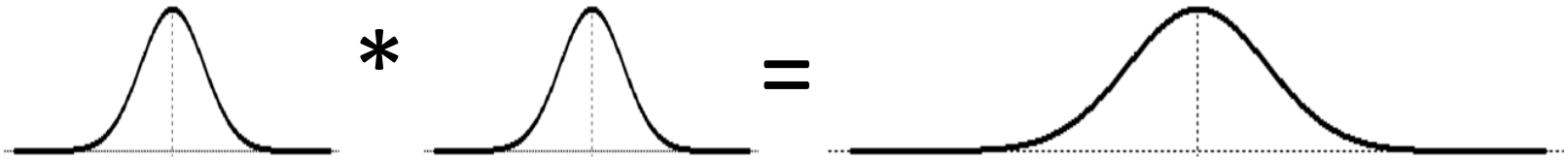


- Larger σ leads to wider kernel and more smoothing
- In discrete domain, kernel should extend ±2-3σ in order to capture > 95% of the area under the Gaussian curve
- Kernel should be normalized to sum to 1
- Kernel is separable so that 3D convolution can be performed as a series of 1D convolutions (more on this later…)

Gaussian Smoothing

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

Convolution is associative



Repeated Gaussian smoothing is equivalent to one Gaussian smoothing with a larger σ

We'll see an interesting application of this later in the lecture

Kernel is radially symmetric

$$G(x,y,z) = \frac{1}{\left(\sigma\sqrt{2\pi}\right)^3} e^{-\frac{\left(x^2+y^2+z^2\right)}{2\sigma^2}}$$

$$r^2 = x^2 + y^2 + z^2$$

Why is this is a desirable property?

Edge Detection Convolution Kernels

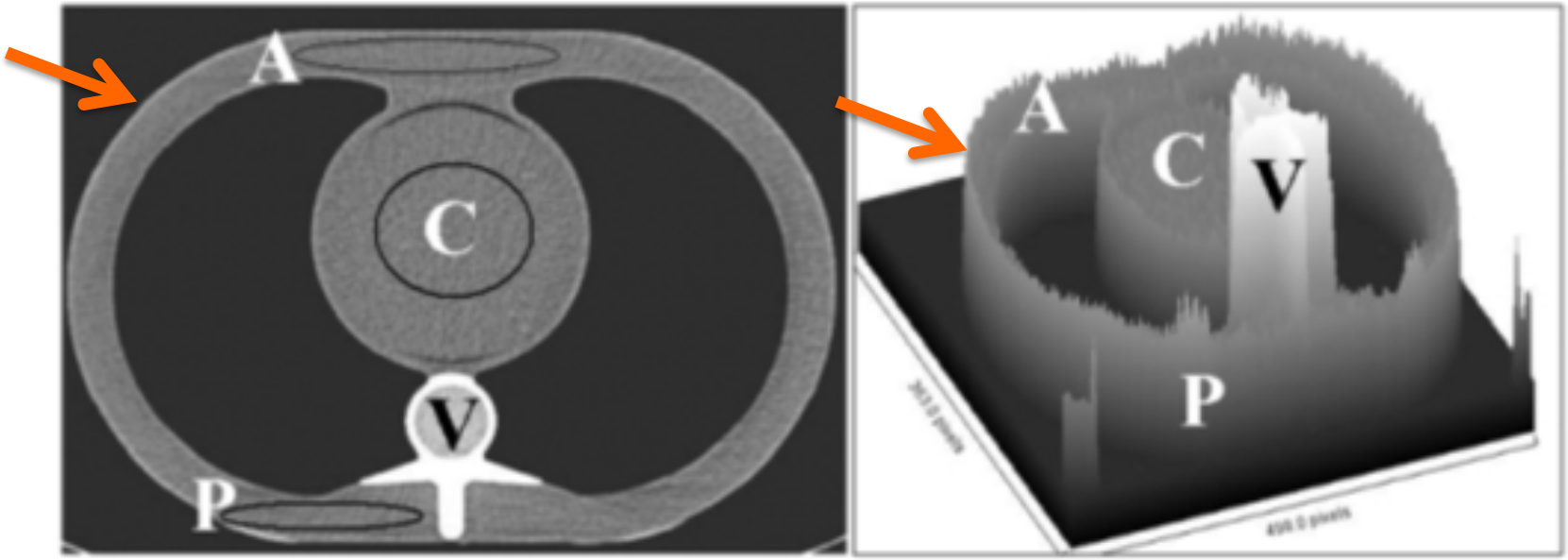
How does one find edges between distinct image regions?

How are image edges defined?

Step edges in an image are steep changes in voxel values

Steepness is defined by spatial derivatives $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}\left(\text{and } \frac{\partial I}{\partial z} \text{ in } 3D\right)$

Image edges (gradients) have direction and magnitude

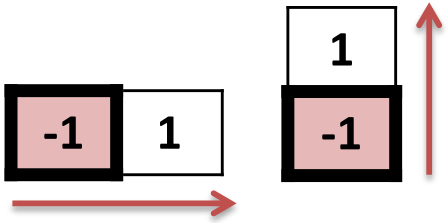


Kaira et al., KJR 2009

Edge Detector Convolution Kernels

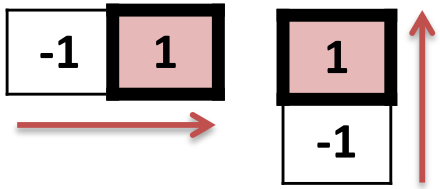
Calculating Gradients

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$



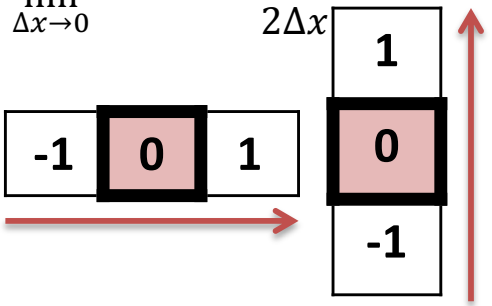
Forward Finite Difference

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

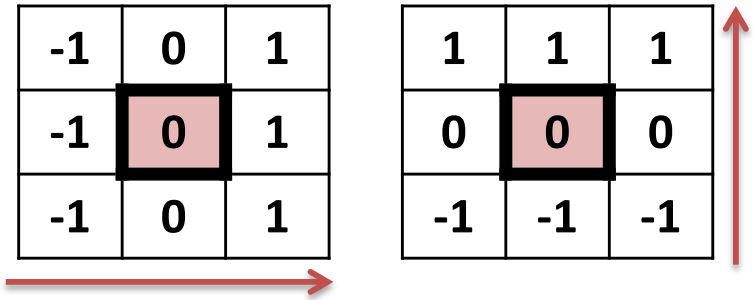


Backward Finite Difference

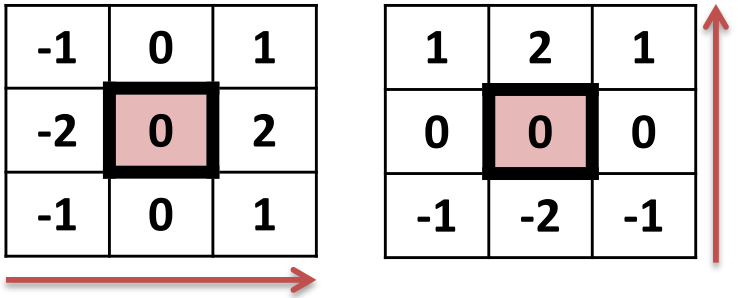
$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}$$



Central Finite Difference



Prewitt Operator



Sobel Operator

Prewitt and Sobel compute derivatives in one direction while smoothing in the other direction

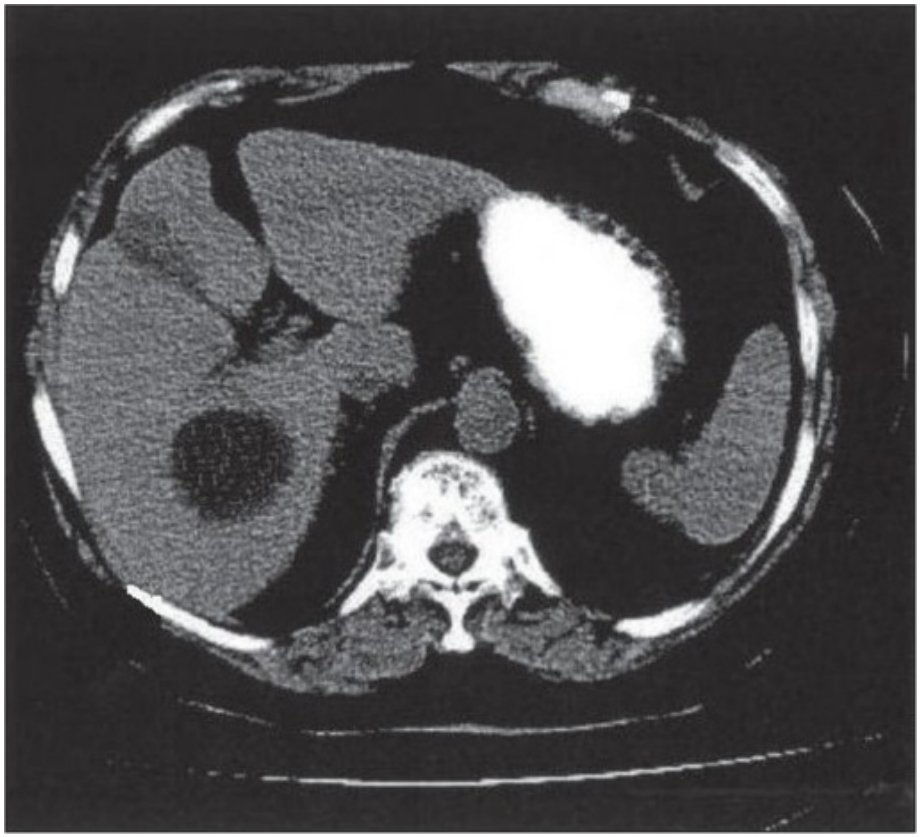
These kernels perform well on very clean images but do not work as well in the presence of noise (kernels are too small!)

Canny Edge Detector

How does one find edges between image regions when noise is a problem?

Canny Edge Detector

Classic Algorithm for Finding Edges in Images



Original CT Image

Canny Edge Detector Output

Sharma et al., JMP 2010

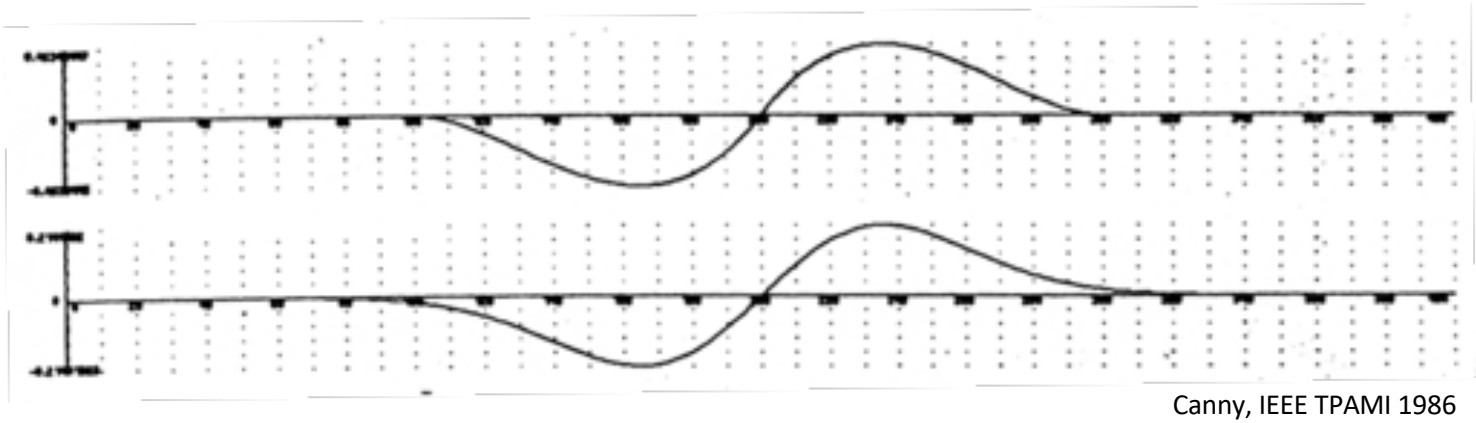
Canny Edge Detector

Classic Algorithm for Finding Edges in Images

- Want a kernel to simultaneously maximize SNR and localization for a step edge under white Gaussian noise
- Solution by numerical optimization is *very* close to derivative of Gaussian kernel

Numerically Optimized

Derivative of Gaussian

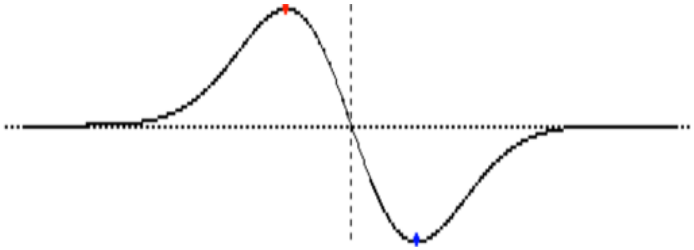
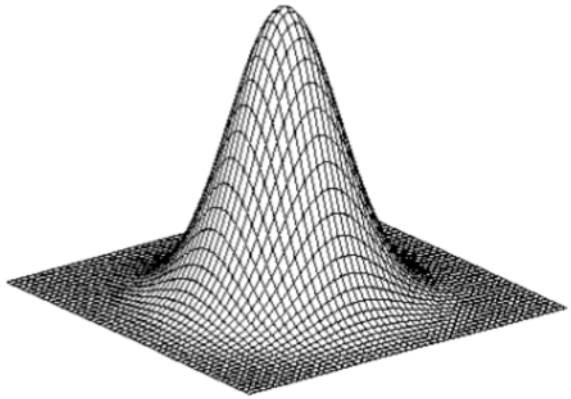
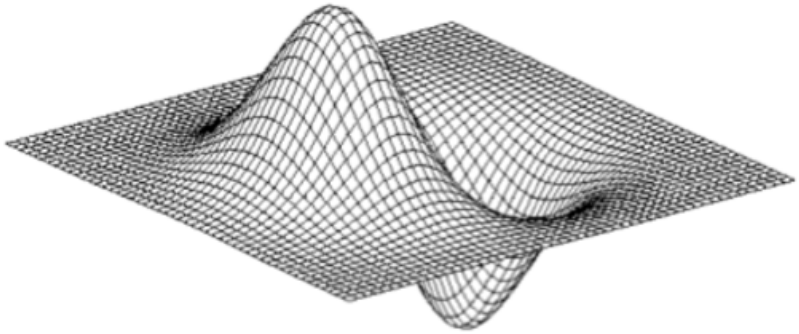


Canny, IEEE TPAMI 1986

Image gradient approximated by convolution with kernel = derivative of Gaussian:

$$\nabla I \cong \nabla G * I = \left(\frac{\partial G}{\partial x} * I, \frac{\partial G}{\partial y} * I, \frac{\partial G}{\partial z} * I \right)$$

(Partial) Derivative of Gaussian Kernel

	$G(\cdot)$	$\frac{\partial G(\cdot)}{\partial x}$
1D		
2D		
3D	$\frac{1}{\left(\sigma\sqrt{2\pi}\right)^3} e^{-\frac{\left(x^2+y^2+z^2\right)}{2\sigma^2}}$	$-\frac{x}{\sigma^2\left(\sigma\sqrt{2\pi}\right)^3} e^{-\frac{\left(x^2+y^2+z^2\right)}{2\sigma^2}}$

Separable Gaussian Kernel

Gaussian

$$G(x,y,z) = G(x)\cdot G(y)\cdot G(z)$$

$$= \frac{1}{\left(\sigma\sqrt{2\pi}\right)^3} e^{-\frac{\left(x^2+y^2+z^2\right)}{2\sigma^2}} = \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \cdot \frac{e^{-\frac{y^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \cdot \frac{e^{-\frac{z^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

Derivative Of Gaussian

$$\frac{\partial G(x,y,z)}{\partial x} = \frac{dG(x)}{dx}\cdot G(y)\cdot G(z)$$

$$= -\frac{xe^{-\frac{x^2}{2\sigma^2}}}{\sigma^3\sqrt{2\pi}} \cdot \frac{e^{-\frac{y^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \cdot \frac{e^{-\frac{z^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

Why does this matter?
Three 1D convolutions is *much* faster than one 3D convolution

Another Way of Looking at It

$$\left(\frac{\partial G}{\partial x} * I, \frac{\partial G}{\partial y} * I, \frac{\partial G}{\partial z} * I\right) = \left(\frac{\partial}{\partial x} G * I, \frac{\partial}{\partial y} G * I, \frac{\partial}{\partial z} G * I\right)$$

$$(\nabla G) * I = \nabla (G * I)$$

Convolving with a derivative of Gaussian is the same as convolving with a Gaussian and then taking a derivative (but taking the exact derivative of an image is not well defined)

We can think of this as equivalent to first blurring in order to reduce noise, then taking an exact derivative

Final Steps of Canny Edge Detector:

Hysteresis thresholding

First, global threshold with an upper threshold (on $|\nabla I|$)
Then, region grow starting from those regions using a lower threshold

Non-maximum suppression

Eliminate edges that are not local maxima (along the direction of the gradient)

Canny Edge Detector Example Results

1. Original

2. Gradient

3. Hysteresis Thresholding

4. Non-max Suppression

Note the breaks in the edges

Laplacian of Gaussian Kernel

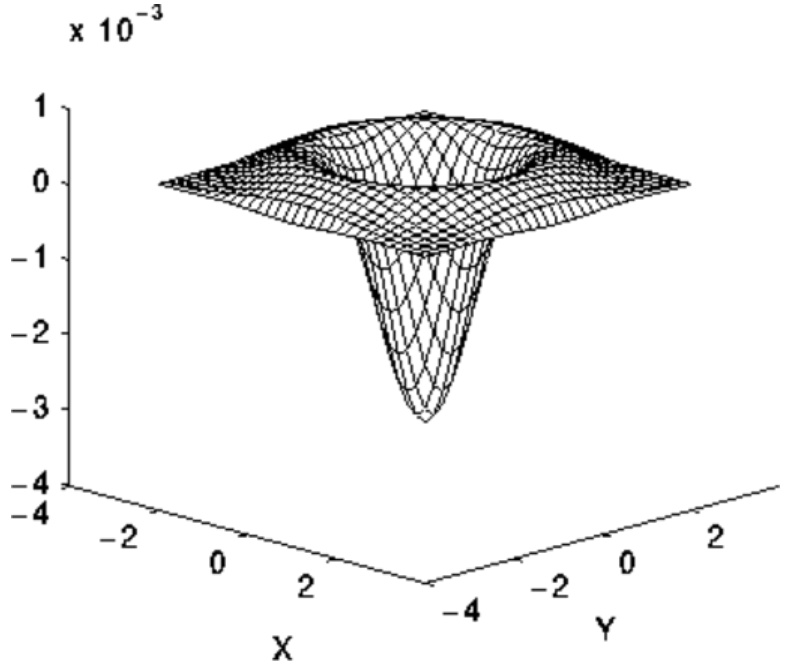
What other features can be detected with derivatives of Gaussians?

Laplacian of Gaussian (LoG) Kernel

$$\mathrm{div}\nabla = \nabla \cdot \nabla = \nabla^2 = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) \cdot \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

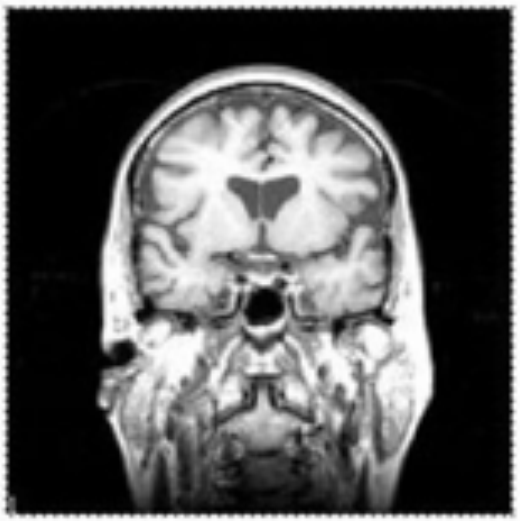
$$\nabla^2 G = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$



You can think of this as adding up N neighbors around the brim of the hat and subtracting N times the center of the hat. We'll see this again!

Think of this as an omnidirectional edge finder as compared to other gradient kernels that find edges in a specific orientation

Edge Detection with LoG

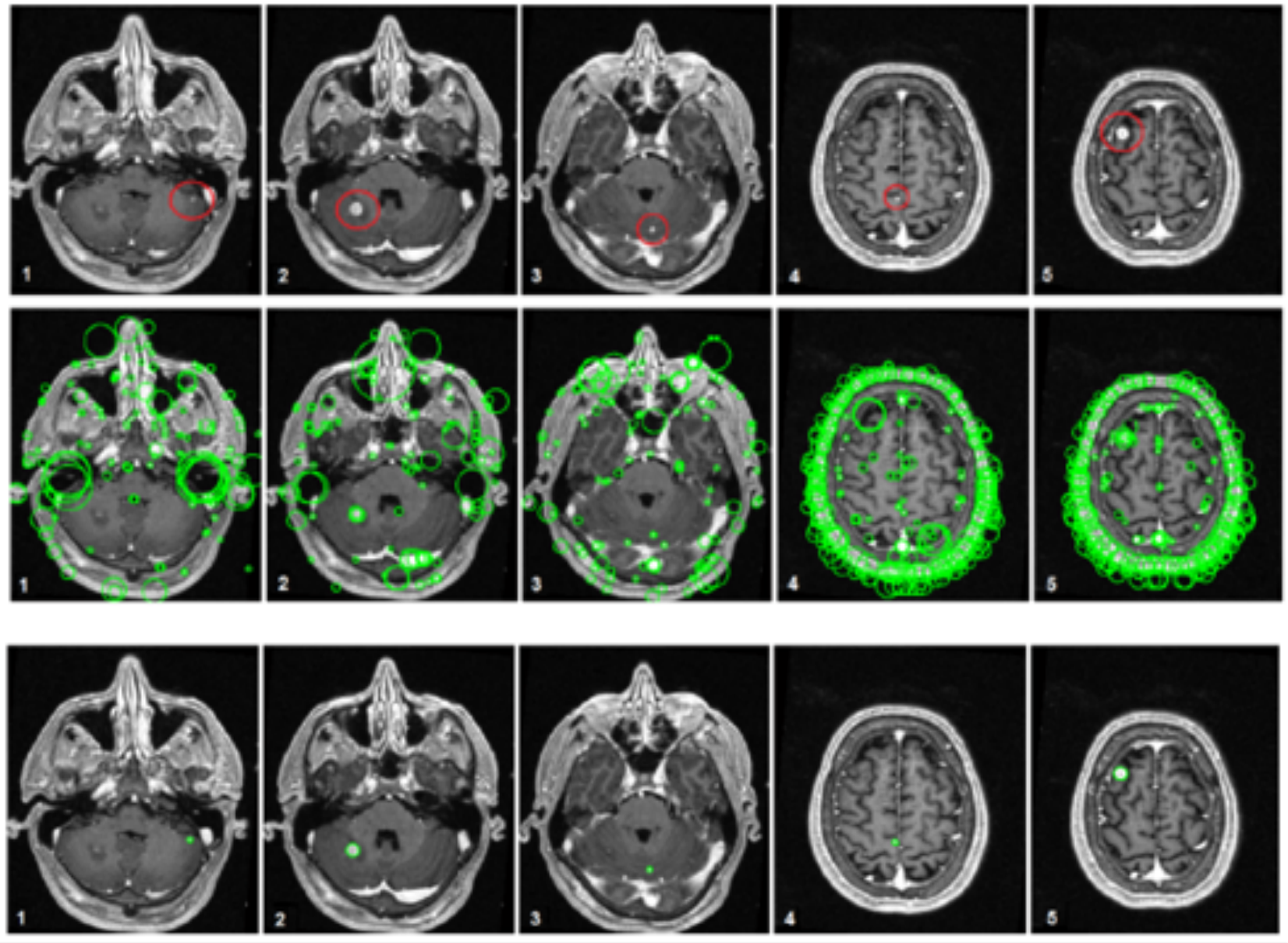


(A) Original MR image

(B) Laplacian results

(C) Extraction of the zero crossing of the Laplacian (object edges)

Blob Detection with LoG



Yu et al, IEEE ISBI 2014

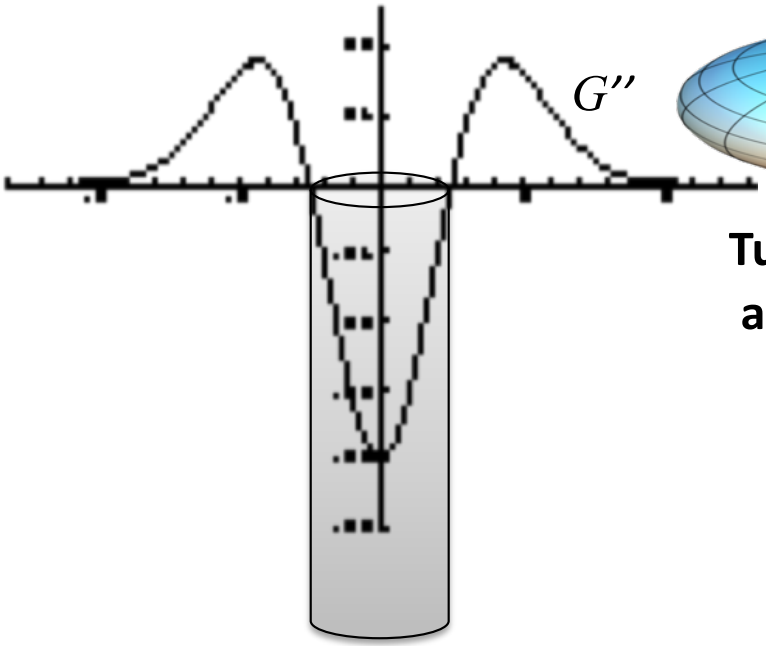
Shape Detection Filtering

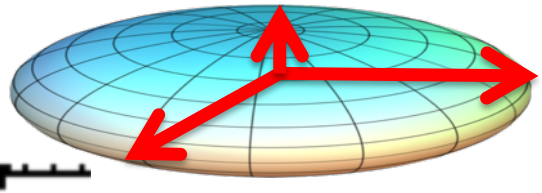
How do you identify tube-like and sheet-like shapes in an image?

Hessian Matrix

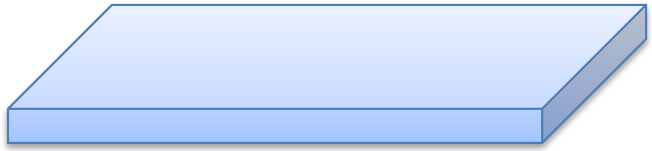
$$H = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} & \frac{\partial^2 I}{\partial x \partial z} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} & \frac{\partial^2 I}{\partial y \partial z} \\ \frac{\partial^2 I}{\partial z \partial x} & \frac{\partial^2 I}{\partial z \partial y} & \frac{\partial^2 I}{\partial z^2} \end{bmatrix}$$

$$\approx \begin{bmatrix} \frac{\partial^2 G}{\partial x^2} & \frac{\partial^2 G}{\partial x \partial y} & \frac{\partial^2 G}{\partial x \partial z} \\ \frac{\partial^2 G}{\partial y \partial x} & \frac{\partial^2 G}{\partial y^2} & \frac{\partial^2 G}{\partial y \partial z} \\ \frac{\partial^2 G}{\partial z \partial x} & \frac{\partial^2 G}{\partial z \partial y} & \frac{\partial^2 G}{\partial z^2} \end{bmatrix} * I$$





Tubes 2nd derivatives as prolate spheroids



Sheet 2nd derivatives as oblate spheroids

LoG kernel has no directionality

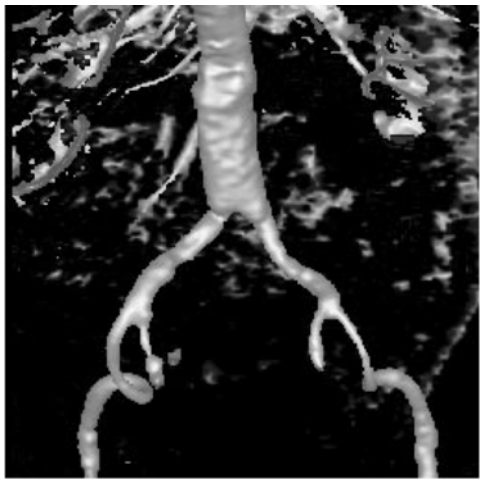
Eigenanalysis can find an orthonormal basis that optimizes the filter response

Hessian Eigenvalues

$$|\lambda_1| \le |\lambda_2| \le |\lambda_3| \quad \textit{by definition}$$

$$\left.\begin{array}{l} \lambda_2 \approx \lambda_3 \\ |\lambda_1| \approx 0 \\ |\lambda_1| << |\lambda_2| \end{array}\right\} \quad \textit{assumption for tubular structures}$$

$\lambda < 0$ *for bright structure*
$\lambda > 0$ *for dark structure*



Original

Vesselness

Hessian Vesselness

$$R_A = \frac{|\lambda_2|}{|\lambda_3|} \qquad R_B = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}}$$

$$S = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}$$

$$V = \begin{cases} 0 & \textit{if } \lambda_2 > 0 \textit{ or } \lambda_3 > 0 \ (\textit{bright}) \\ \left(1 - e^{-\frac{R_A^2}{2\alpha^2}}\right) e^{-\frac{R_B^2}{2\beta^2}} \left(1 - e^{-\frac{S^2}{2c^2}}\right) & \end{cases}$$

Analysis can be performed across multiple scales (values of σ)

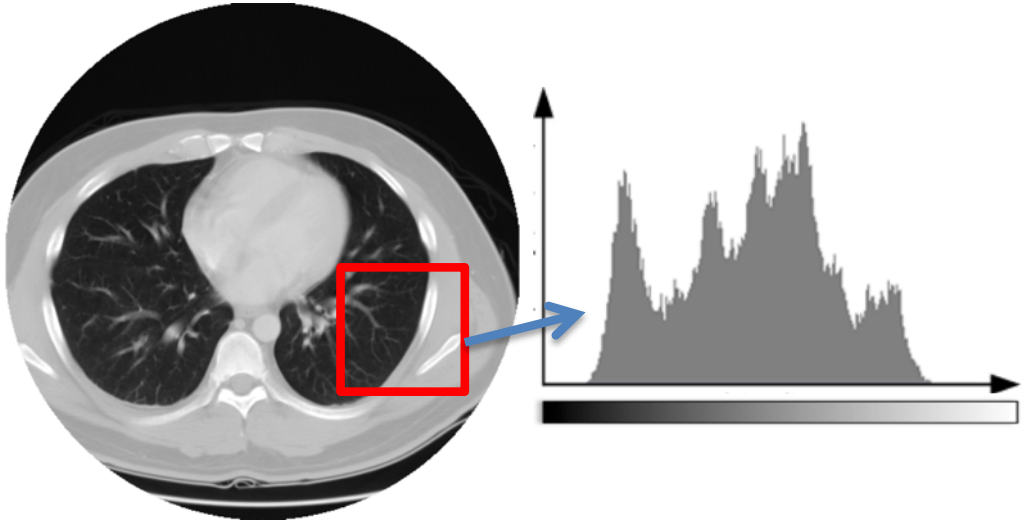
Frangi et al, MICCAI 1998

Non-linear Noise Reduction Filtering

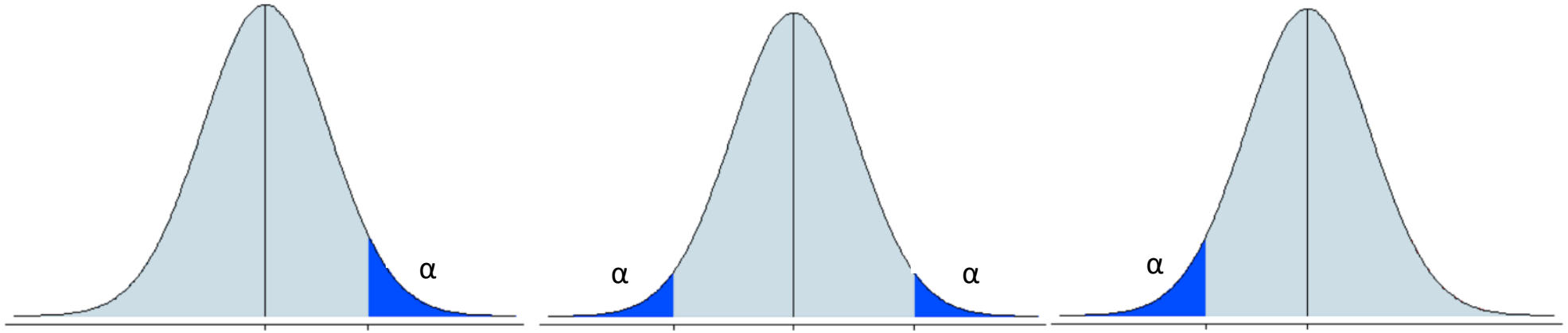
How does one “smooth” away
image noise without
blurring away the real edges?

Smoothing with Alpha-Trimmed Mean

(Weighted) means are not the only statistics than can be computed over a moving kernel. Although it'll no longer be true convolution.



Middle percentile α=0 → Mean
Middle percentile α=0.5 → Median
Lower percentile α=1 → Minimum
Upper percentile α=1 → Maximum

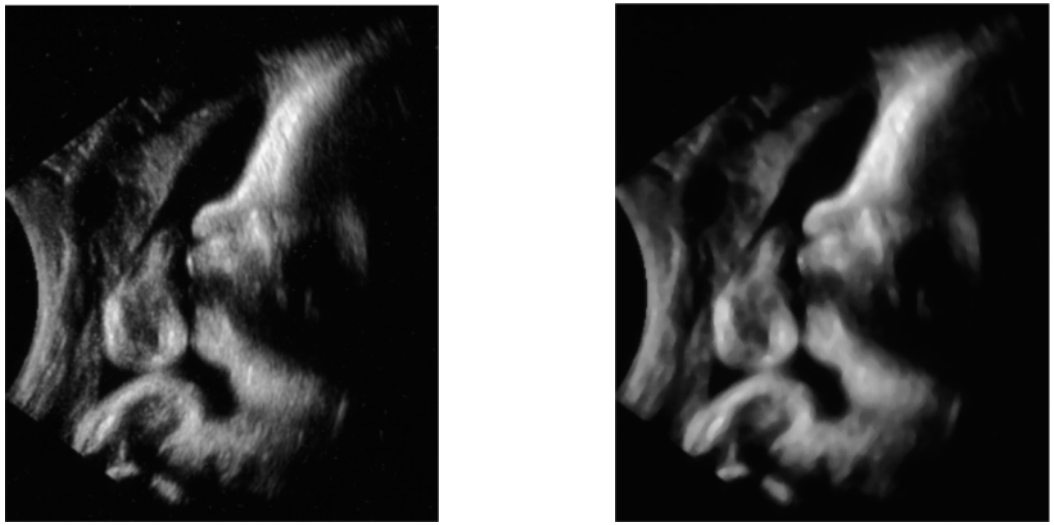


Lower Percentile
(light blue)

Middle Percentile
(light blue)

Upper Percentile
(light blue)

Results of Median Filtering



Works particularly well against “salt and pepper” outlier noise

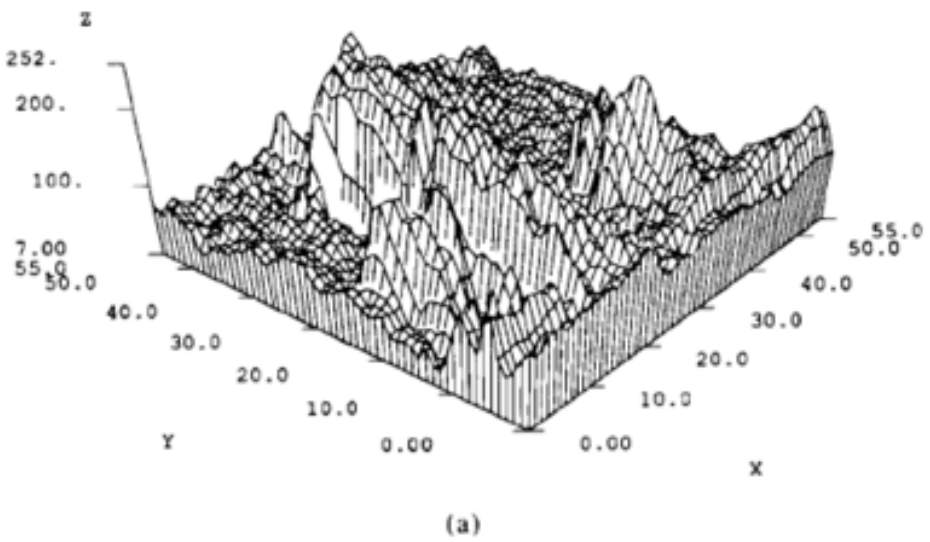
http://www-mmdb.iai.uni-bonn.de/lehre/BIT/ss03_DSP_Vorlesung/matlab_demos/index.html
http://reference.wolfram.com/mathematica/ref/MedianFilter.html

Anisotropic Diffusion

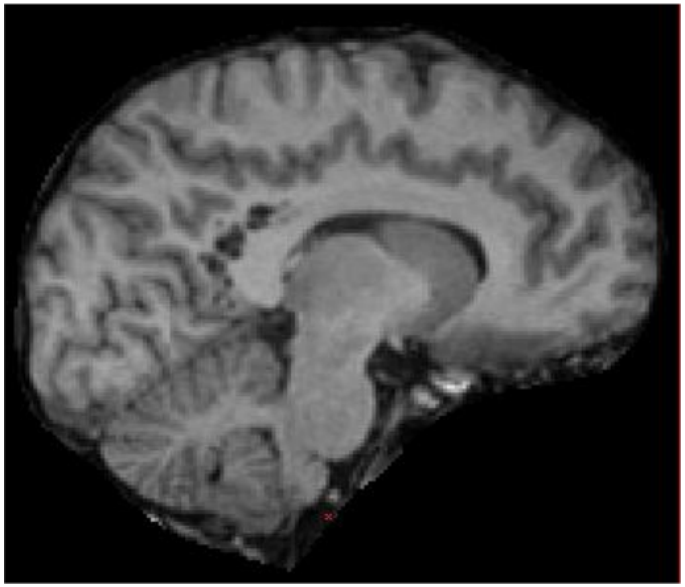
What physical processes can be emulated in order to achieve edge preserving noise reduction?

Edge Preserving Smoothing

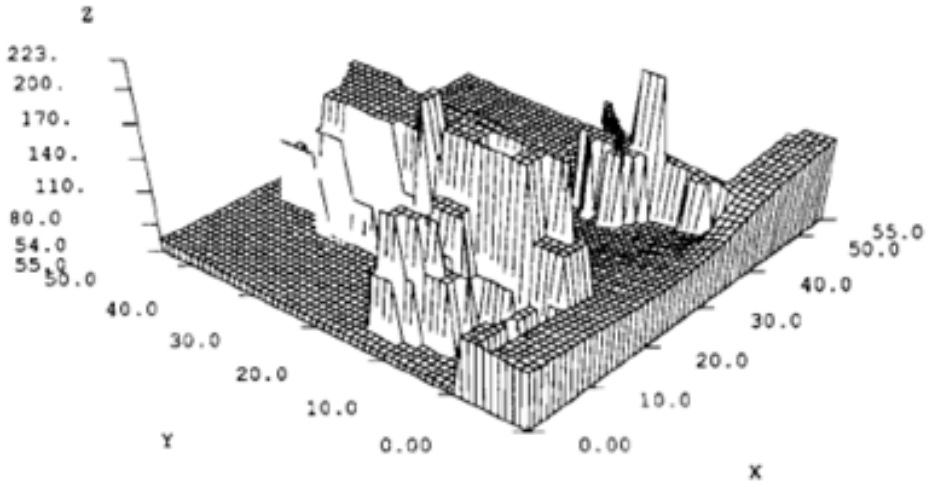
Anisotropic Diffusion



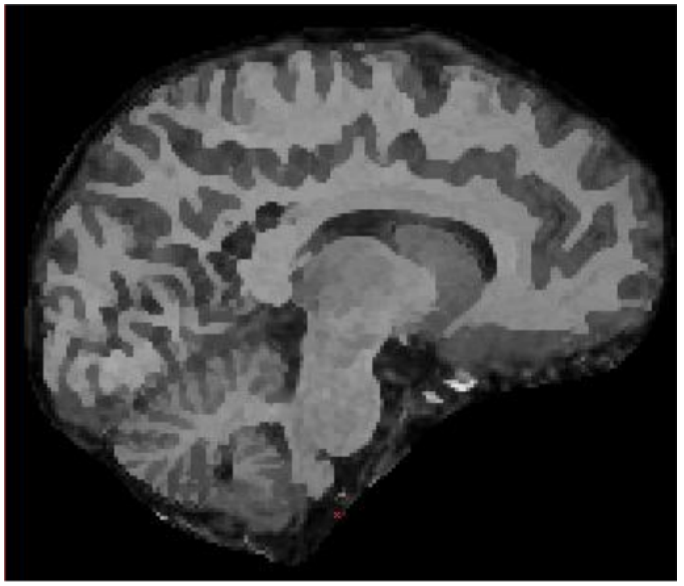
(a)



Banner.500.3D



(b)

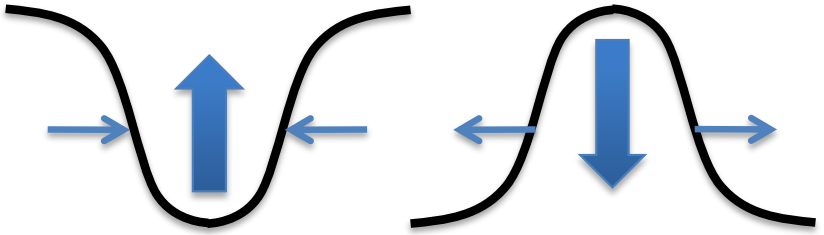


Perona and Malik, IEEE PAMI 1990
http://sharp.bu.edu/~fischl/fischl.html

(*Plain Old*) Isotropic Diffusion

Diffusion in 1D

$$\frac{\partial I}{\partial t} = c\frac{\partial^2 I}{\partial x^2}$$



Diffusion in 2D/3D

$$\frac{\partial I}{\partial t} = \operatorname{div}\left(c\nabla I\right) = c\nabla^2 I = c\left(\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} + \frac{\partial^2 I}{\partial z^2}\right)$$

Incidentally:

isotropic diffusion = Gaussian smoothing *where*

length of time of diffusion = Gaussian variance, σ^2

$$O(x,y,t) = G_{\sigma^2=t}(x,y) * I(x,y)$$

Lindeberg, Encyclopedia of Computer Science and Engineering

Perona-Malik Anisotropic Diffusion

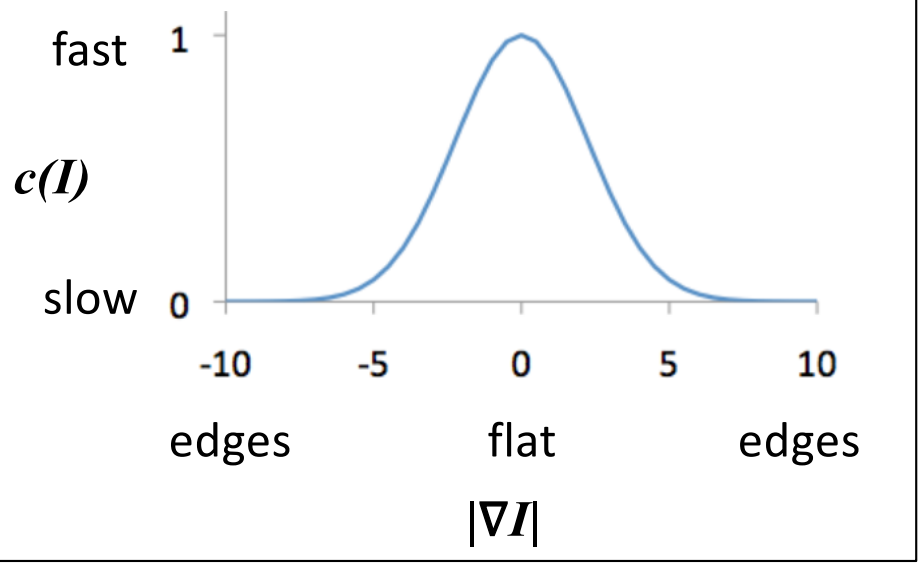
Key idea: slow down diffusion near image edges

Anisotropic Diffusion: $$\frac{\partial I}{\partial t} = \operatorname{div}\left(c(I)\nabla I\right)$$

Why anisotropic?
c no longer a constant, it's a function of the image

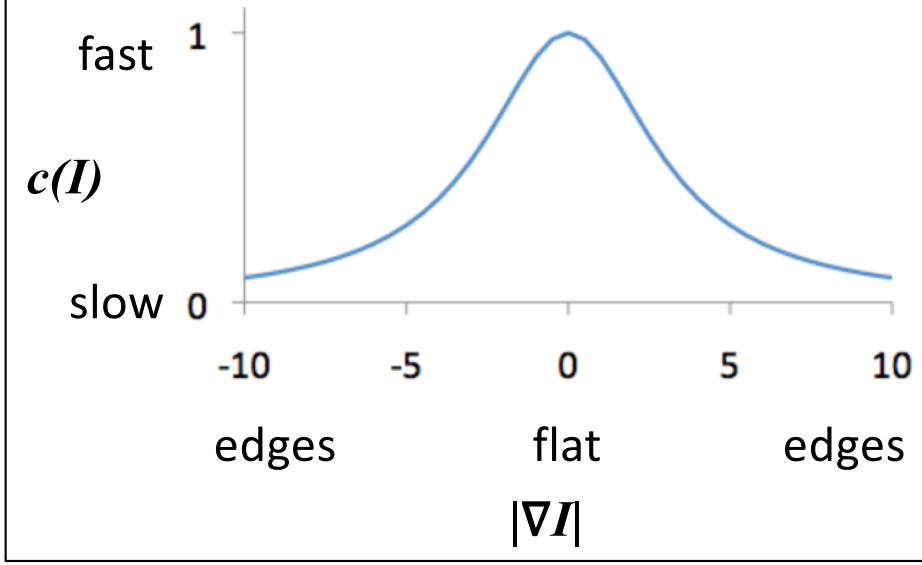
Two different diffusion coefficient schemes:

$$c(I) = e^{-\left(|\nabla I|/K\right)^2}$$



OR

$$c(I) = \frac{1}{1+\left(|\nabla I|/K\right)^2}$$



Perona-Malik Implementation

Definition of divergence

$$\operatorname{div}\left(G_x\mathbf{i}+G_y\mathbf{j}\right)=\frac{\partial G_x}{\partial x}+\frac{\partial G_y}{\partial y}$$

Central Finite Difference

$$\frac{\partial f}{\partial x}=\frac{1}{\Delta x}\left(f\left(x+\frac{\Delta x}{2}\right)-f\left(x-\frac{\Delta x}{2}\right)\right)$$

$$\frac{\partial I(x,y,t)}{\partial t}=\operatorname{div}\left(c(I(x,y,t))\nabla I\right)$$

Use definition of div

$$=\frac{\partial}{\partial x}\left[c(I(x,y,t))\frac{\partial I}{\partial x}\right]+\frac{\partial}{\partial y}\left[c(I(x,y,t))\frac{\partial I}{\partial y}\right]$$

Replace gradient in x and y with central differences

$$=\frac{\partial}{\partial x}\left[c(I(x,y,t))\frac{1}{\Delta x}\left(I\left(x+\frac{\Delta x}{2},y,t\right)-I\left(x-\frac{\Delta x}{2},y,t\right)\right)\right]$$

$$+\frac{\partial}{\partial y}\left[c(I(x,y,t))\frac{1}{\Delta y}\left(I\left(x,y+\frac{\Delta y}{2},t\right)-I\left(x,y-\frac{\Delta y}{2},t\right)\right)\right]$$

Replace outer derivatives with central differences

$$=\frac{1}{\Delta x^2}\left[c(I(x+\frac{\Delta x}{2},y,t))\left(I\left(x+\Delta x,y,t\right)-I\left(x,y,t\right)\right)\right]-\frac{1}{\Delta x^2}\left[c(I(x-\frac{\Delta x}{2},y,t))\left(I\left(x,y,t\right)-I\left(x-\Delta x,y,t\right)\right)\right]$$

$$+\frac{1}{\Delta y^2}\left[c(I(x,y+\frac{\Delta y}{2},t))\left(I\left(x,y+\Delta y,t\right)-I\left(x,y,t\right)\right)\right]-\frac{1}{\Delta y^2}\left[c(I(x,y-\frac{\Delta y}{2},t))\left(I\left(x,y,t\right)-I\left(x,y-\Delta y,t\right)\right)\right]$$

Adapted from G Lohmann, Volumetric Image Analysis

Perona-Malik Implementation

$$= \frac{1}{\Delta x^2}\left[c(I(x+\frac{\Delta x}{2},y,t))\left(I\left(x+\Delta x,y,t\right)-I\left(x,y,t\right)\right)\right]-\frac{1}{\Delta x^2}\left[c(I(x-\frac{\Delta x}{2},y,t))\left(I\left(x,y,t\right)-I\left(x-\Delta x,y,t\right)\right)\right]$$
$$+\frac{1}{\Delta y^2}\left[c(I(x,y+\frac{\Delta y}{2},t))\left(I\left(x,y+\Delta y,t\right)-I\left(x,y,t\right)\right)\right]-\frac{1}{\Delta y^2}\left[c(I(x,y-\frac{\Delta y}{2},t))\left(I\left(x,y,t\right)-I\left(x,y-\Delta y,t\right)\right)\right]$$

Notational convenience

$e.g., \nabla_N I = I\left(x,y+\Delta y,t\right)-I\left(x,y,t\right)$

(neighbor minus center)

$$= \frac{1}{\Delta x^2}\left[c(I(x+\frac{\Delta x}{2},y,t))\left(\nabla_E I\right)\right]-\frac{1}{\Delta x^2}\left[c(I(x-\frac{\Delta x}{2},y,t))\left(\nabla_W I\right)\right]$$
$$+\frac{1}{\Delta y^2}\left[c(I(x,y+\frac{\Delta y}{2},t))\left(\nabla_N I\right)\right]-\frac{1}{\Delta y^2}\left[c(I(x,y-\frac{\Delta y}{2},t))\left(\nabla_S I\right)\right]$$

Approximate gradient magnitude with projection of gradient in N,S,E,W dir

(*preserves image brightness*)

$$= \frac{1}{\Delta x^2}\left[g(\left|\nabla_E I\right|)\left(\nabla_E I\right)\right]-\frac{1}{\Delta x^2}\left[g(\left|\nabla_W I\right|)\left(\nabla_W I\right)\right]$$
$$+\frac{1}{\Delta y^2}\left[g(\left|\nabla_N I\right|)\left(\nabla_N I\right)\right]-\frac{1}{\Delta y^2}\left[g(\left|\nabla_S I\right|)\left(\nabla_S I\right)\right]$$

(neighbors minus center with some coefficients)

Notational convenience

$where\ c(I) = g(\left|\nabla I\right|)$

$$= \Phi_E - \Phi_W + \Phi_N - \Phi_S$$

Shorthand notation

Adapted from G Lohmann, Volumetric Image Analysis

Perona-Malik Implementation

$$\frac{\partial I}{\partial t} = \operatorname{div}\left(c(I)\nabla I\right)$$

We've calculated all the spatial derivatives on the **RHS**, now let's do the time derivative on the **LHS**

Forward Finite Difference Approximation of Time Derivative

$$I\left(x,y,t+\Delta t\right) = I\left(x,y,t\right) + \Delta t \frac{\partial I\left(x,y,t\right)}{\partial t}$$

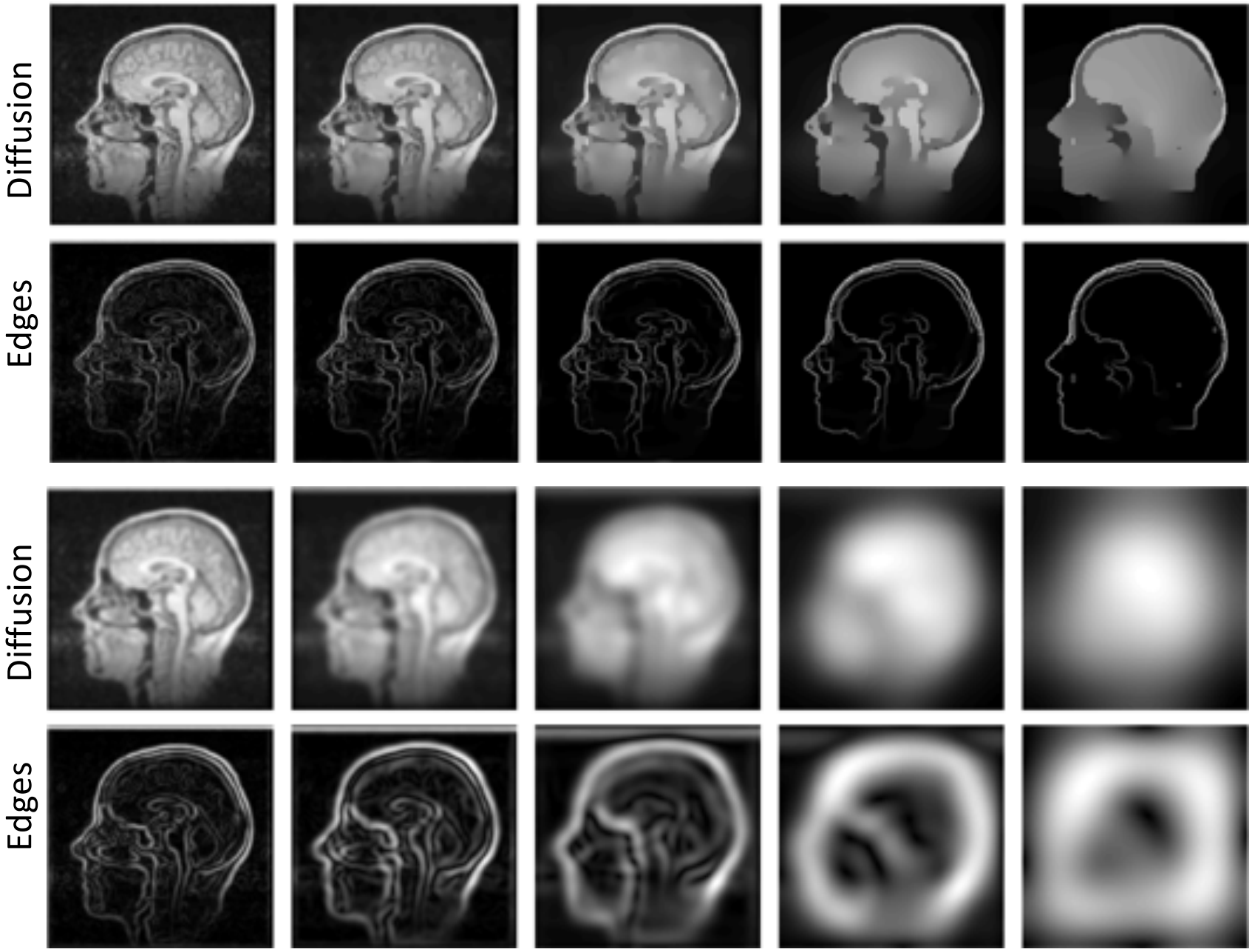
$$I_{i+1}\left(x,y\right) = I_i\left(x,y\right) + \Delta t \frac{\partial I_i\left(x,y\right)}{\partial t}$$

$$= I_i\left(x,y\right) + \Delta t\left[\Phi_E - \Phi_W + \Phi_N - \Phi_S\right]$$

Anisotropic diffusion is an iterative process that is terminated when 'enough' smoothing has been performed

Adapted from G Lohmann, Volumetric Image Analysis

Anisotropic Diffusion Results



James Fishbaugh http://www.cs.utah.edu/~jfishbau/advimproc/project2/

What is the take-home message?

- Noise vs. real features (edges) is the key struggle
- Application of physics analogies (e.g., diffusion) is not uncommon
- The key to working with grayscale images is dealing effectively with the noise
  - Taking derivatives makes it worse [finite differences]
  - You can average it out [convolution, Canny edge detection]
  - You can statistically remove outliers [Alpha-trimmed mean]
  - You can selectively blur it out [anisotropic diffusion]

What does it mean for me?

- You have a very solid foundation in image filtering techniques
  - Convolution and Smoothing
    - Continuous Convolution
    - Discrete Convolution
    - Gaussian Smoothing
  - Edge Detection
    - Canny Edge Detector
    - Laplacian of Gaussian Edge Detector
  - Non-linear Noise Reduction
    - Alpha-Trimmed Mean
    - Perona-Malik Anisotropic Diffusion

Next Lecture:

Geometric Features