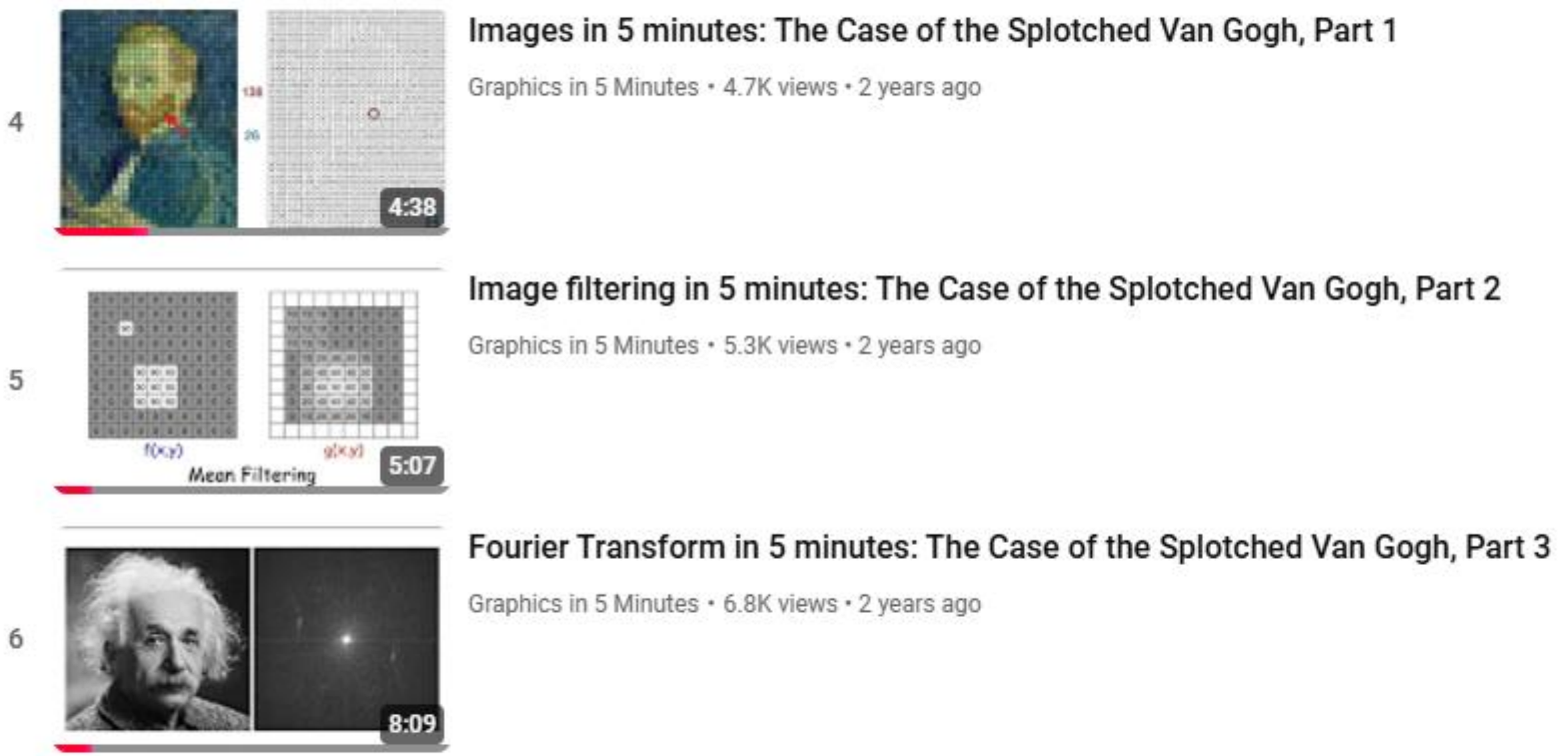
Videos to Watch (by Steve Seitz)

https://www.youtube.com/playlist?list=PLWfDJ5nla8UpwShx-lzLJqcp575fKpsSO



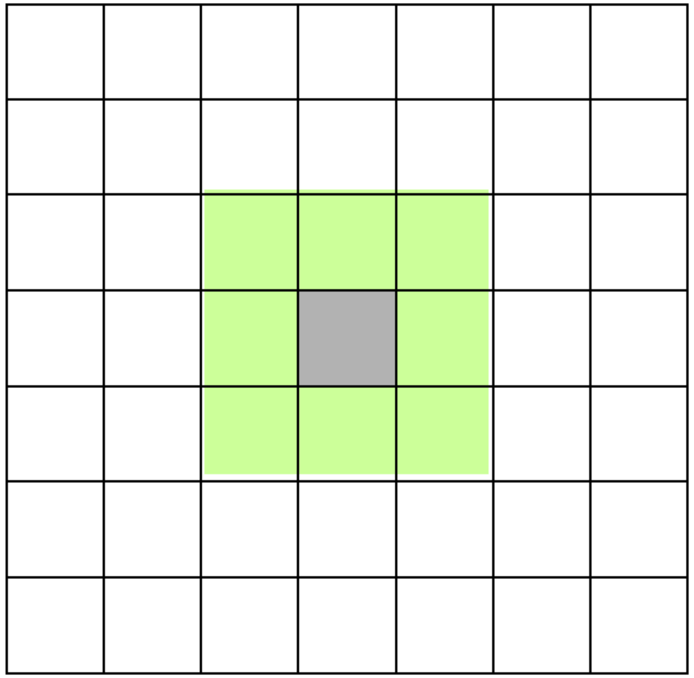
4) Sampling, Moire

5) Convolutions, Blur before Sample

6) Fourier – Explaining the sampling effect

Convolution

- A linear operator (weighted sum of neighbors) applied identically on all pixels.
- Example - **Blur**: Average a pixel with its 3×3 neighborhood



$$\underset{\text{Blurred Pixels}}{h(x,y)} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \underset{\text{Kernel}}{f(i,j)} \; \underset{\text{Original Pixels}}{g(x-i,y-j)}$$

- Image g is blurred by kernel $f =$ giving image h (uniform average)

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

(x,y) of center = (0,0)

Convolution

$$h(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(i,j)\ g(x-i, y-j)$$

- A linear operator (weighted sum of neighbors) applied identically on all pixels.
  - Blur: Average a pixel with its neighbors (weighted average)

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

(x,y) of center = (0,0)

  - Why is blur important?

0	0	0
0	1	0
0	0	0

  - Do Nothing kernel

Convolution

$$h(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(i,j)\ g(x-i, y-j)$$

– <u>Shift</u> the image to the left (reflection of kernel)

Only a single non-zero $h(x,y) = f(-1,0)\ g(x+1,y)$

1		

(x,y) of center = (0,0)

– <u>Edge</u>: Compute a difference between neighbors of a pixel (~derivative)

- Vertical edges

1	-1

1/2		-1/2

- Horizontal edges

1
-1

	1/2	
	-1/2	

- 1D convolution, 2D convolution

Convolution - Blur

– Blur: Replace a pixel with an average of its neighbors

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

- Horizontal blur

1/3	1/3	1/3

- Vertical Blur

	1/3	
	1/3	
	1/3	

- Diagonal Blur

		1/3
	1/3	
1/3		

Convolutions: Smoothing, Noise Cleaning

Smoothing

($\sum$ weights = 1)

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Original Image

Noisy Image

Filtered Image

1-D Discrete Convolution ($*$)

$f = (f_0, f_1, f_2, f_3, \ldots)$, $g = (g_0, g_1, g_2, g_3, \ldots)$, are 1-D arrays

$$h = f * g; \qquad h(x) = (f * g)(x) = \sum_{a=1}^{n} f(a)g(x-a)$$

Let f, g be defined in the coordinate range [0..5]. Assume cyclic boundaries.

$f = (0,0,0,1,0,0)$

$g = (0,0,0,1,-1,0)$

$$h(6) = h(0) = \sum_{a=0}^{5} f(a) \cdot g(6-a) = f(3) \cdot g(3) = 1$$

$$h(7) = h(1) = \sum_{a=0}^{5} f(a) \cdot g(7-a) = f(3) \cdot g(4) = -1$$

Boundary Handling

$$f = (0,0,0,1,0,0)$$

- **Q:** *f* is defined in the range of [0..5] . What is *f(-3); f(9)* ?

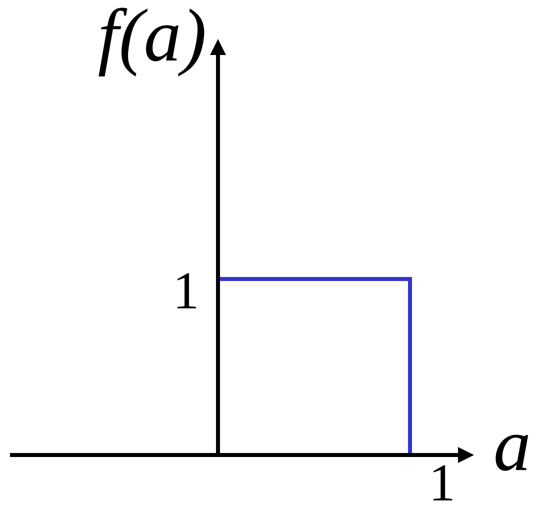
- **A1:** *f(-3) = f(9) = 1;* Cyclic approach (Fourier). *-3 = 9 = 3 (Mod. 6)*
  [Rarely Used in practice]

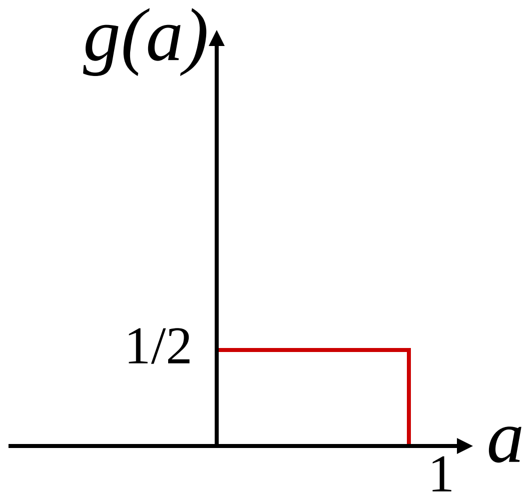
- **A2:** *f(-3) = f(9) = 0;* Every index out of range is zero.
  [Zero Padding]

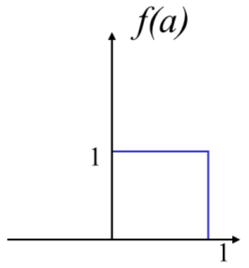
- **A3:** Reflection. *F(-a) =F(a). F(N+a)-F(N-1-a);*

1-D Continuous Convolution

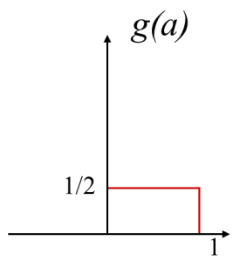
$$(f*g)(x)=\int_{-\infty}^{\infty} f(a)g(x-a)\,da$$

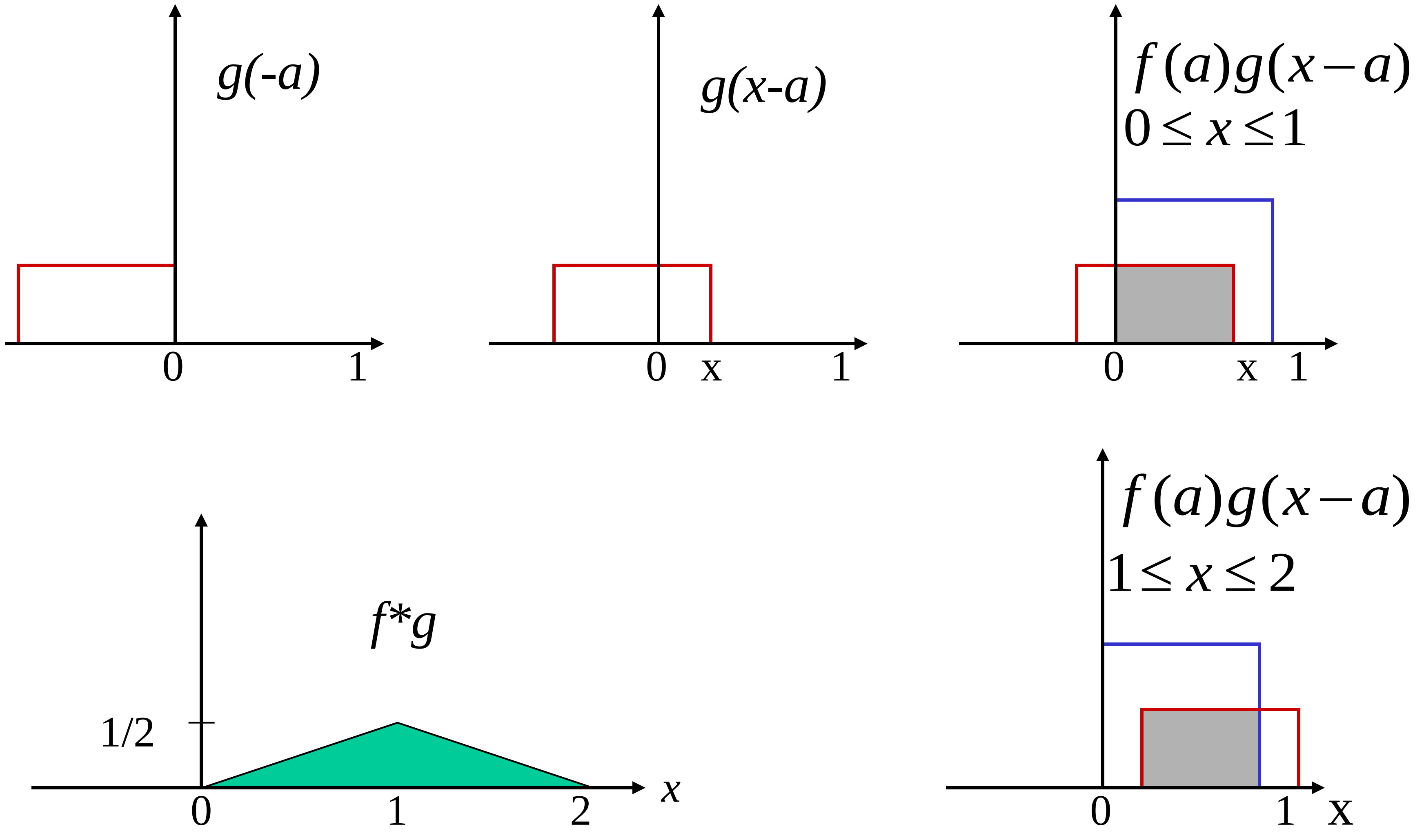






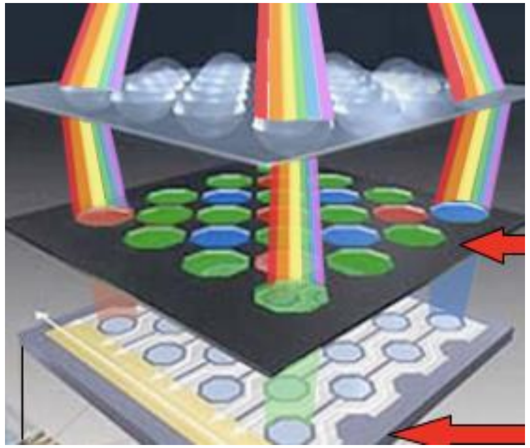
$$(f*g)(x)=\int_{-\infty}^{\infty} f(a)g(x-a)\,da$$

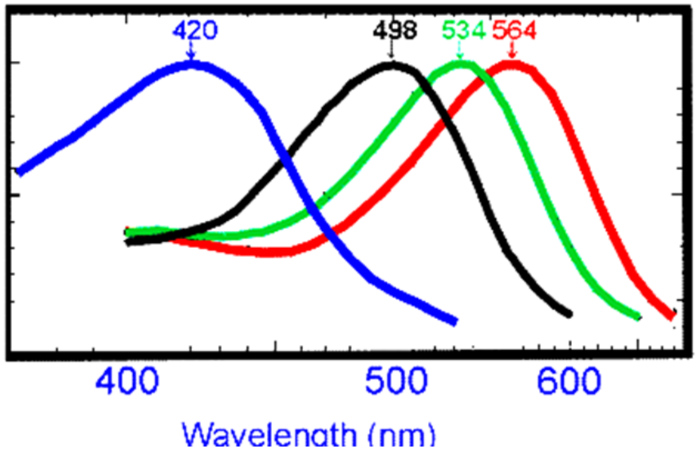




Why Continuous Convolution?

- The physical world is continuous (sensors)
  - Response of sensors to color (color space – electromagnetic wavelength)
  - Area response of sensors (image space)





Convolution Questions

(i) What other courses use convolutions?

Algorithms (polynom, FFT), Deep Learning (CNN)

(i) What is the complexity of discrete convolution?

$O(N^2)$;   With FFT it is *$O(N \log N)$*

(i) Why "reflect"?

Many good properties, e.g. Commutative

Convolution Theorem (Φ is Transform Fourier)

$$\Phi(f * g) = F \cdot G$$

Pointwise Multiplication

$$\Phi(f \cdot g) = F * G$$

Convolution in spatial domain (*f(x,y), g(x,y)*) is equivalent to **pointwise multiplication** in frequency domain (*F(u,v), G(u,v)*)

Performing Convolution by using Fourier:

$$f * g = \Phi^{-1}(F \cdot G) = \Phi^{-1}\big(\Phi(f) \cdot \Phi(g)\big)$$

FFT (Fast Fourier Transform) reduces complexity of convolution:

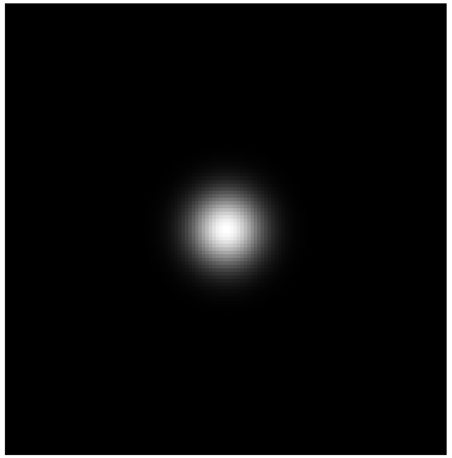
$O(N^2) \rightarrow O(N \log N)$

Convolutions in the Frequency Domain

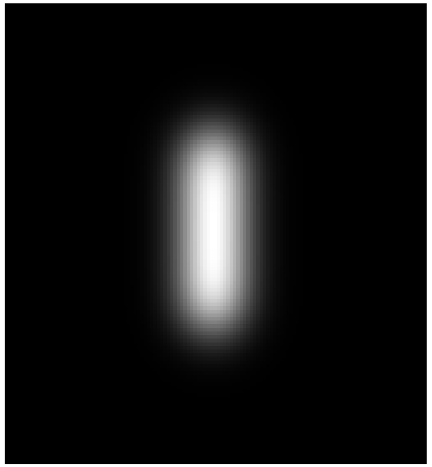
Convolution

Image $f(x,y)$

$g(x,y)$

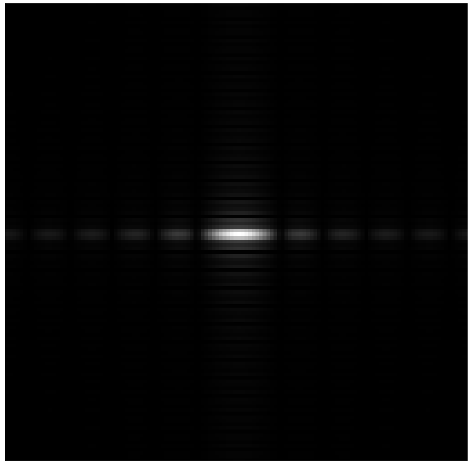


$f * g\ (x,y)$

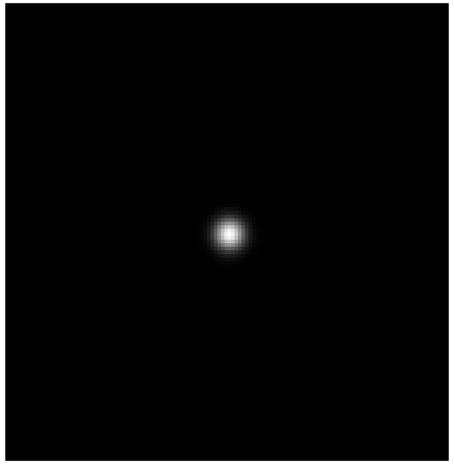


Fourier Spectrum

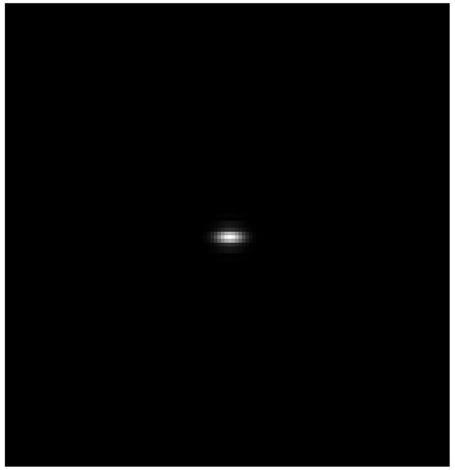
$F(u,v)$



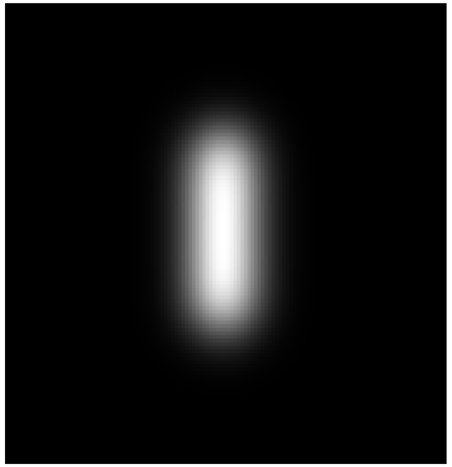
$G(u,v)$



$F(u,v) \cdot G(u,v)$



$\Phi^{-1}[F(u,v) \cdot G(u,v)]$



Properties of Convolution

Commutative: $$f * g = g * f$$

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

Distributive: $$f * (g + h) = f * g + f * h$$

Convolution is a Linear Operation over 1D vectors and 2D images

Convolution as Matrix Multiplication (Cyclic)

A Linear Operator can be expressed as matrix multiplication

$$(x_1, x_2, x_3, x_4, x_5, x_6) * \frac{1}{4}\begin{pmatrix}1 & 2 & 1\end{pmatrix}$$

$$(x_1, x_2, x_3, x_4, x_5, x_6) \cdot \frac{1}{4}\begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 \\ 1 & 0 & 0 & 0 & 1 & 2 \end{pmatrix}$$

Circulant Matrix

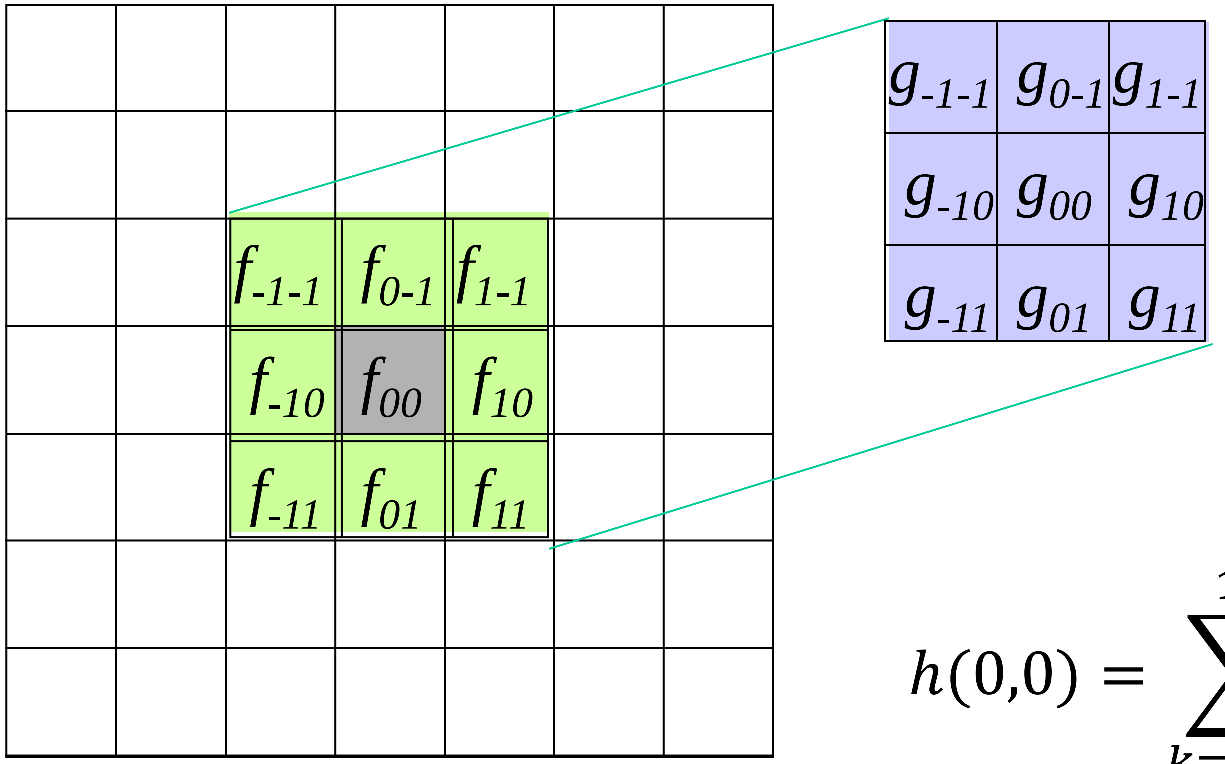
2-D Discrete Convolution

$$\boxed{h = f * g}$$

$$h(x,y) = \sum_{k=0}^{n} \sum_{l=0}^{m} f(k,l)\ g(x-k, y-l)$$

Blurred Pixel Kernel Original Pixels

2D Discrete Convolution



$$h(0,0) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} f(k,l) \cdot \; g(0-k, 0-l)$$

$$f(-1,-1)g(1,1) + f(-1,0)g(1,0) + f(-1,1)g(-1,1) + f(0,-1)g(0,1) + \ldots$$

Simple Convolutions (Empty is zero)

$$\begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$$

[1 [1 [1 [1 1]
[1 [1 [1 [1 1]

$$\begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

[1 [1 [1 [1 [1 1]

Simple Convolutions (Empty is zero)

$$\begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

$$\begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix}$$

$$\begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$$

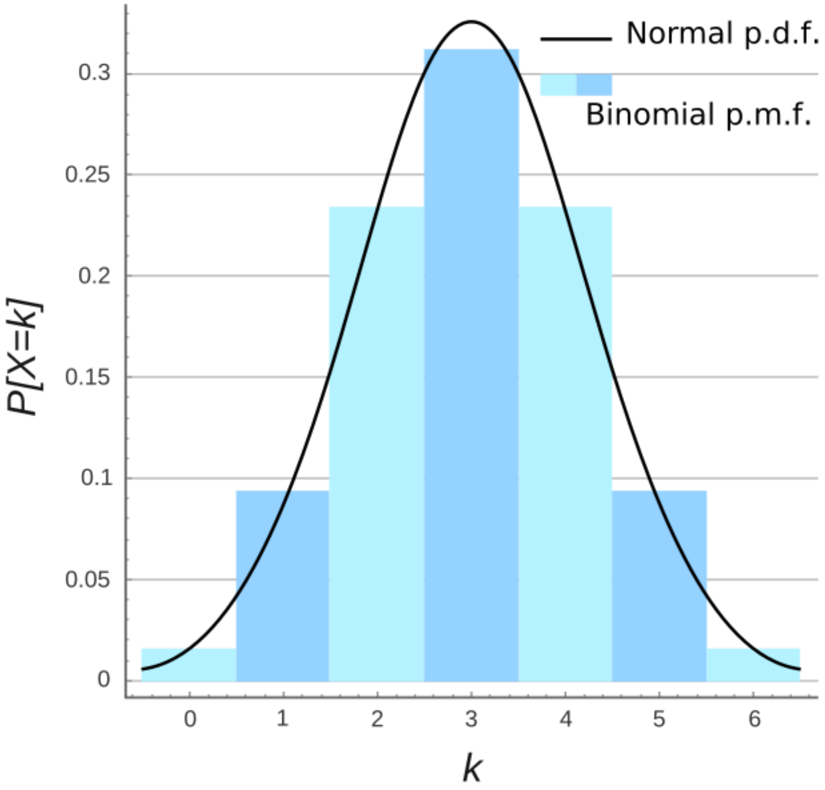
$$\begin{bmatrix}1 & 2 & 1\end{bmatrix} * \begin{bmatrix}1\\2\\1\end{bmatrix} = \begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

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

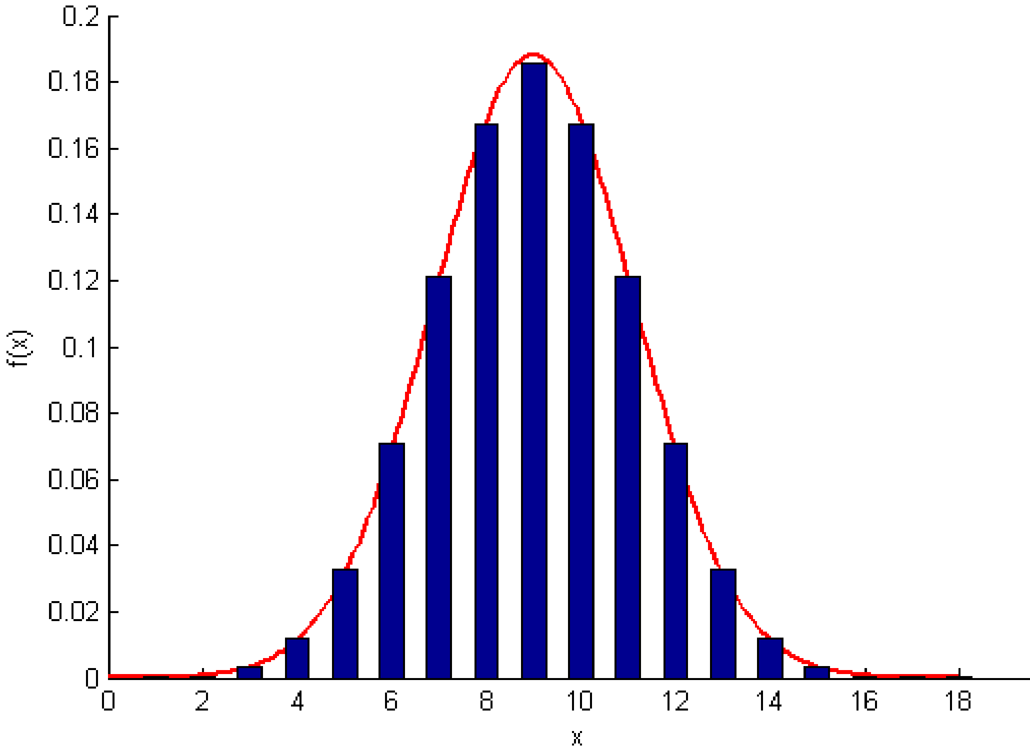
$$\begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix}$$

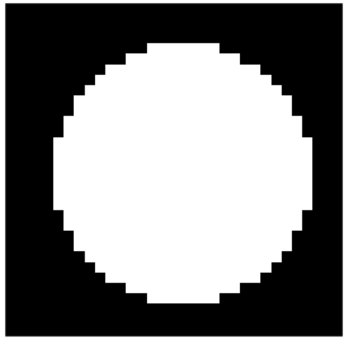
Simple Convolutions (Empty is zero)

- Repetitive convolutions of [1 1] giving binomial coefficients (Pascal triangle)
- For large n, binomial coefficients approximate a Gaussian $G(x) = \frac{1}{2\pi\sigma^2} e^{\frac{(x-\mu)^2}{2\sigma^2}}$



$$
\begin{bmatrix}1 & 1\end{bmatrix}^n
$$
$$
\begin{bmatrix}1 & 1\end{bmatrix}
$$
$$
\begin{bmatrix}1 & 2 & 1\end{bmatrix}
$$
$$
\begin{bmatrix}1 & 3 & 3 & 1\end{bmatrix}
$$
$$
\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}
$$

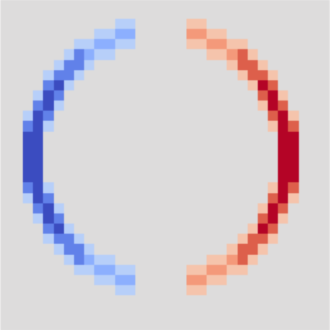




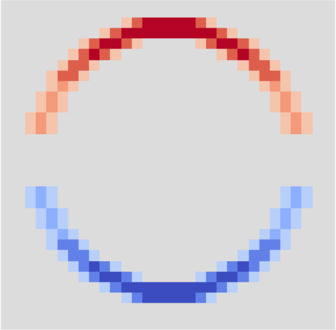
Edge Detection

Edge: Large difference between neighboring pixels

- Vertical edges
- right-left diff



- Horizontal edges
- below-above diff



$$\begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} * \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1/6 & 0 & -1/6 \\ 1/6 & 0 & -1/6 \\ 1/6 & 0 & -1/6 \end{bmatrix}$$

$$\begin{bmatrix} 1/2 \\ 0 \\ -1/2 \end{bmatrix} * \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} = \begin{bmatrix} 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 \\ -1/6 & -1/6 & -1/6 \end{bmatrix}$$

– Difference, a derivative, high pass - across the desired edge

– Blur, low pass, noise cleaning - along the desired edge

Convolutions: Smoothing, Noise Cleaning

Q: What is the average gray level after smoothing?

Smoothing

($\sum$ weights = 1)

$$\frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Original Image

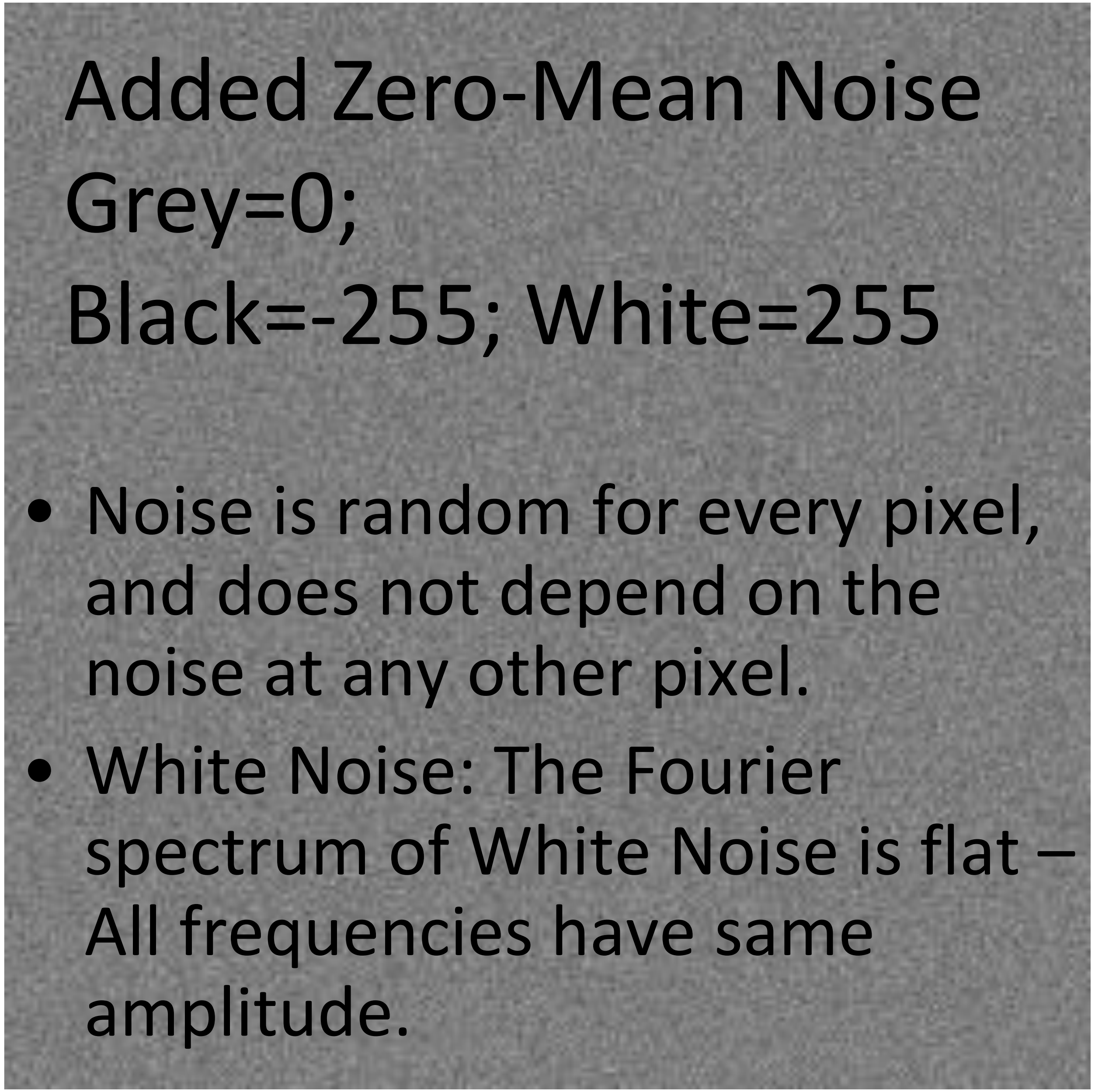
Noisy Image

Filtered Image

Noise Assumption: Additive, Zero-Mean, Independent

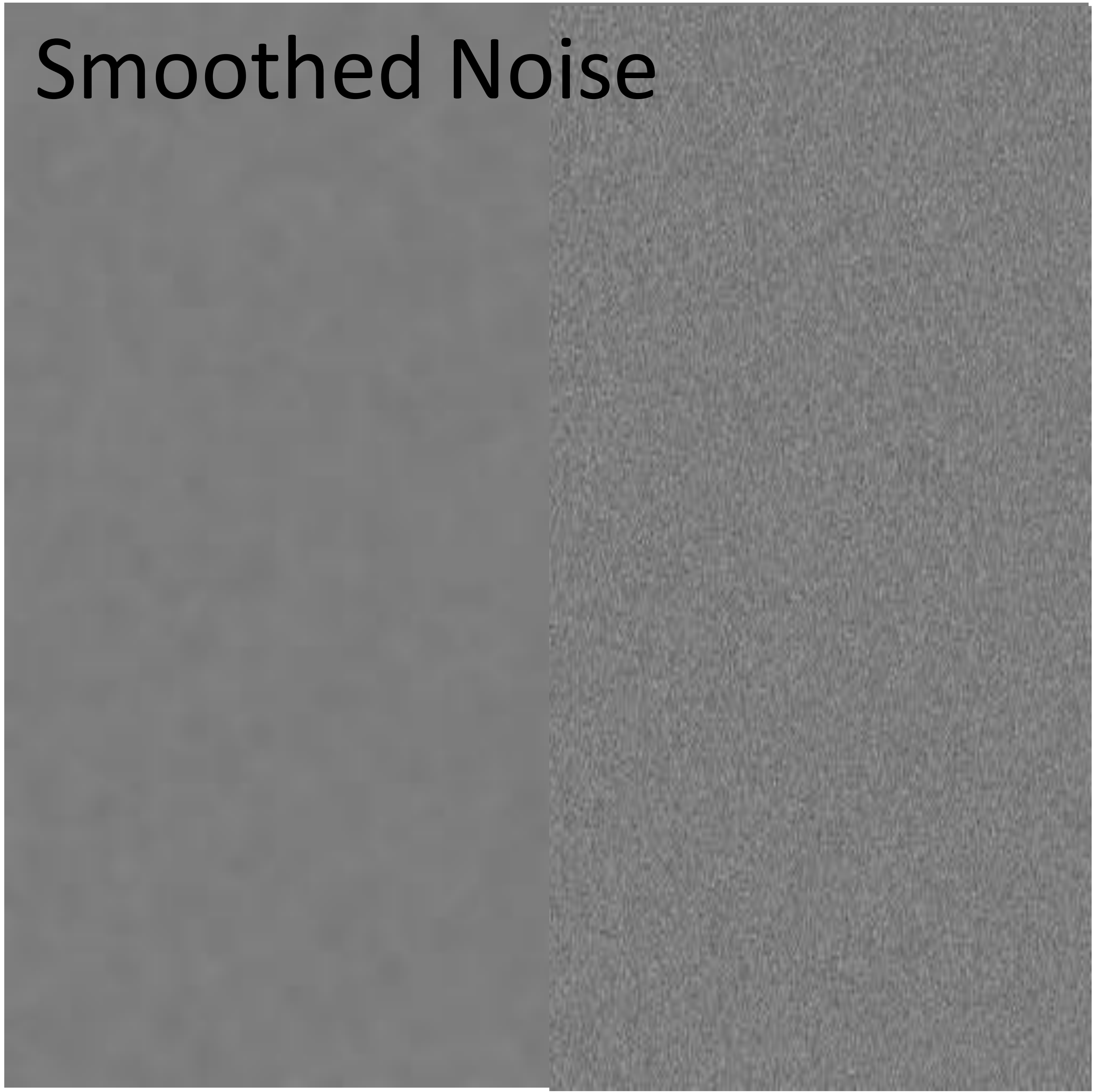
Convolutions: Smoothing, Noise Cleaning

Convolutions: Smoothing, Noise Cleaning



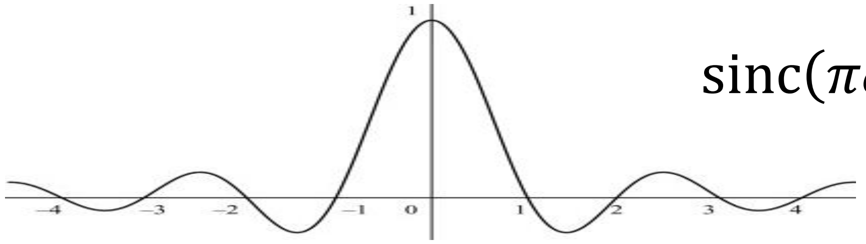
Convolutions: Smoothing, Noise Cleaning

Convolutions: Smoothing, Noise Cleaning



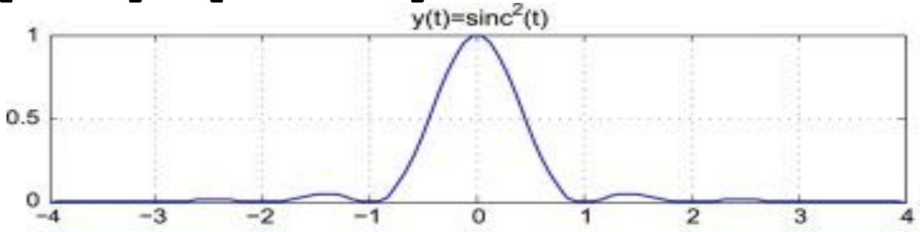
Low Pass: Fourier Domain & Image Domain

- Image: convolve f by box Fourier: Multiply F by Sinc
- [1 1]



$$\mathrm{sinc}(\pi\omega) = \frac{\sin(\pi\omega)}{\pi\omega}$$

- Multiply Fourier F by box $\Leftrightarrow$ Convolve image f with Sinc
- Image: convolve by [1 1] * [1 1]= [1 2 1]
- Fourier: Multiply F by Sinc^2



- Image: Gaussian $\Leftrightarrow$ Fourier: Gaussian
- Blur image f w. Gaussian $\Leftrightarrow$ Multiply Fourier F w. Gaussian

Convolution: Spatial filtering

Smoothing by Convolution

$\frac{1}{9}\times$

1	1	1
1	1	1
1	1	1

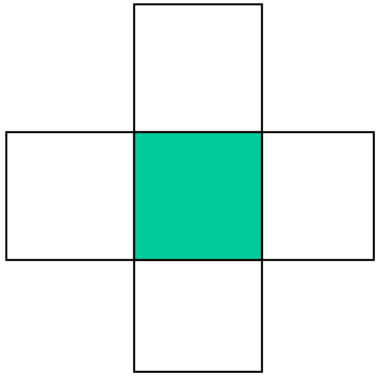
$\frac{1}{16}\times$

1	2	1
2	4	2
1	2	1

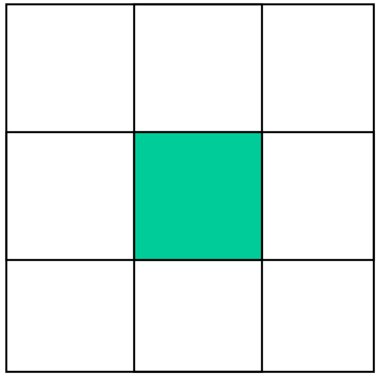
Smoothing by Spatial Filtering

Using larger filter

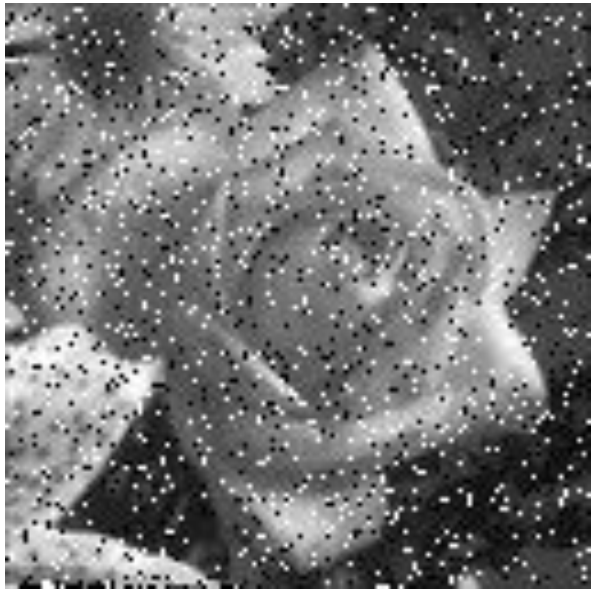
Noise Cleaning by Median Filtering



- Replace the value of a pixel with the MEDIAN of its neighborhood
- Depends on the definition of "neighborhood"



Original



Salt & Pepper Noise

Median

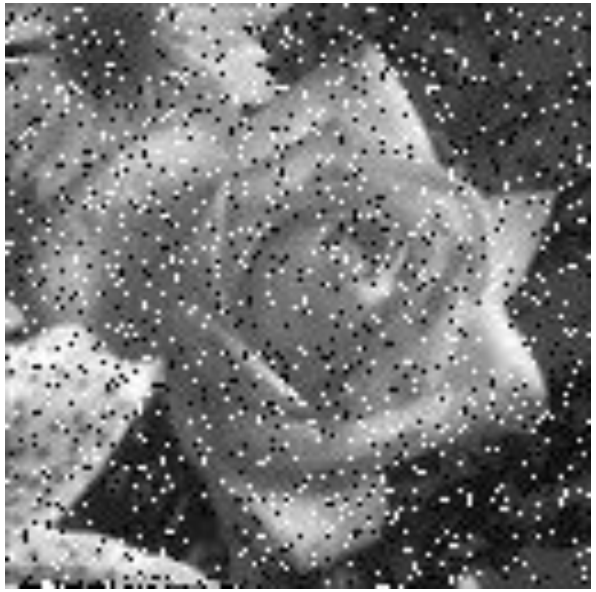
Median

- Given a set of numbers, **half** are above the median and **half** are below the median.
- 1, 3, 3, **6**, 7, 8, 9 Median = 6 [Mean: 5.2]
- 1, 2, 3, **4, 5**, 6, 8, 9 Median = 4.5 [Mean: 4.75]

- 1, 2, 1, 2, 10^6 Mean = (1,000,006)/7 = 200,001
- 1, 1, **2**, 2, 10^6 Median = 2
- Median in **robust** to <u>outliers</u>

Noise Cleaning

- Averaging / Smoothing - Loss of Detail
- Median - Blockiness



Salt & Pepper Noise
Black or white, ignores original value

Median

Blur (Convolution)

Spatial Approximation to Derivatives

$$\frac{\partial}{\partial x} f(i,j) = \lim_{h \to 0} \frac{f(i,j) - f(i-h,j)}{h} \cong f(i,j) - f(i-1,j)$$

$$\cong \frac{f(i+1,j) - f(i-1,j)}{2}$$

convolution with

$$\begin{pmatrix} 1 & -1 \end{pmatrix}$$

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

$\begin{pmatrix} 1 & -1 \end{pmatrix}$ Derivative between 2 pixels

$\frac{1}{2}\begin{pmatrix} 1 & 0 & -1 \end{pmatrix}$ Derivative at center pixel

Spatial Approximation to Derivatives

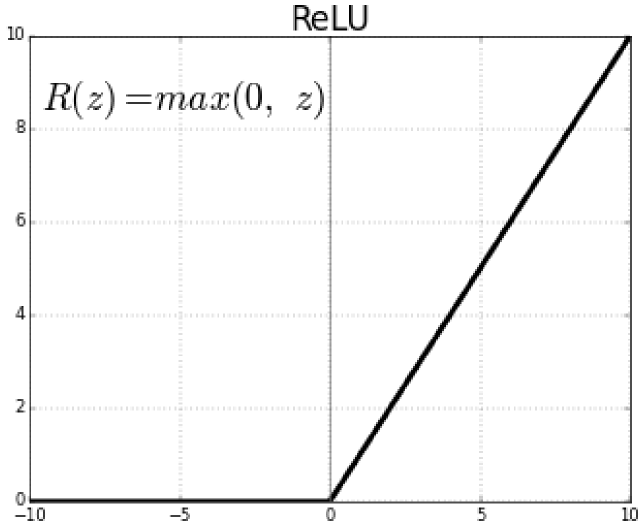
$$\frac{\partial}{\partial x} f(i,j) = \lim_{h \to 0} \frac{f(i,j) - f(i-h,j)}{h} \qquad \cong f(i,j) - f(i-1,j)$$

convolution with $\begin{pmatrix} 1 & -1 \end{pmatrix}$

$$\cong \quad \frac{f(i+1,j) - f(i-1,j)}{2}$$

Convolution with $\frac{1}{2}\begin{pmatrix} 1 & 0 & -1 \end{pmatrix}$

Detection of Vertical Edges (ReLU: Negative → 0)



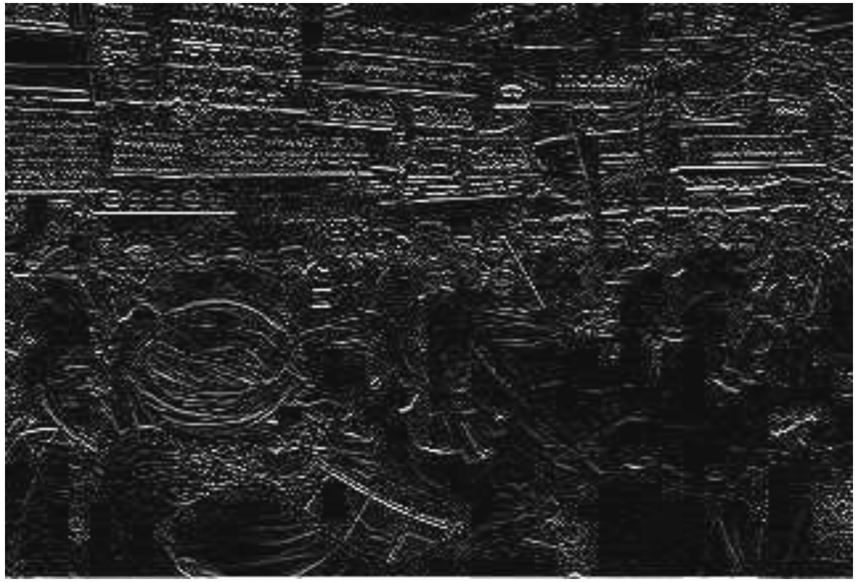
Spatial Approximation to Derivatives

Q: What is the average derivative in all pixels?

$$\frac{\partial}{\partial y} f(i,j) \cong f(i,j) - f(i,j-1)$$

convolution with $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$

or with $\frac{1}{2}\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$



Horizontal Edge Detection (ReLU)

Derivative Example

Image x derivative y derivative

Negative: Black. Zero: grey. Positive: White.

No absolute value. No RelU

Image Derivatives (cont')

Since a picture is not continuous, there are many **approximations** to the derivative

Popular blur kernels (Sobel):

$$\frac{\partial f}{\partial x} = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \frac{\partial f}{\partial y} = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

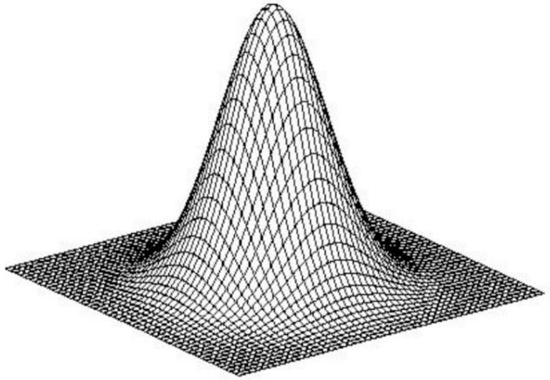
Note: Good derivative filters are edge in one direction and **blur** in the orthogonal direction.

Compare to:

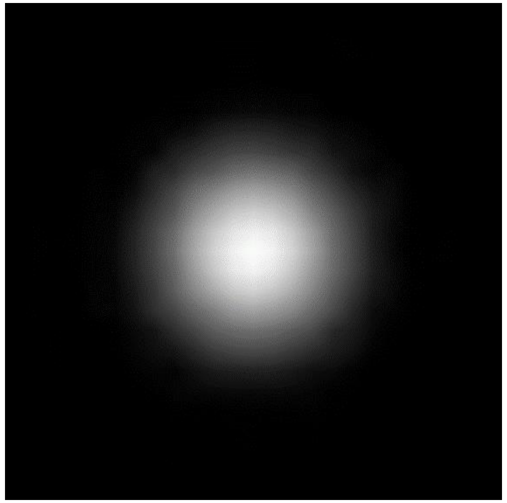
$$\frac{\partial f}{\partial x} = \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad \frac{\partial f}{\partial y} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

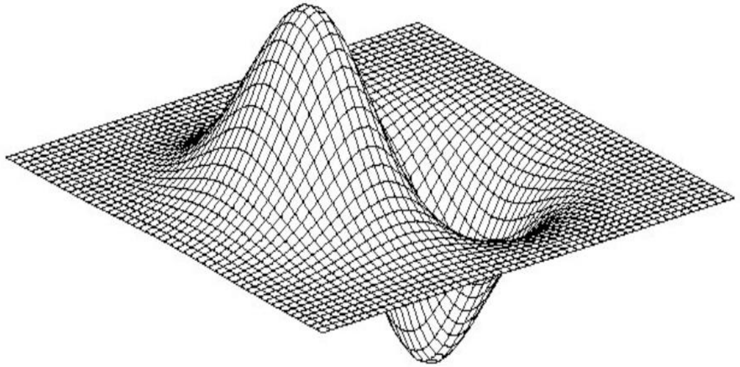
Gaussian Blur & Its Derivatives

- Blur image & derivative = Convolve with derivative of blur

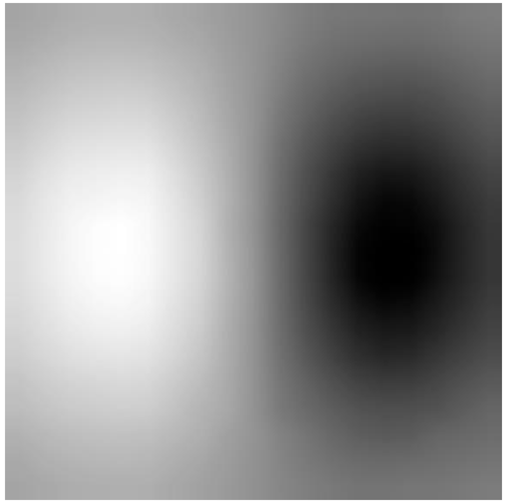


Gaussian

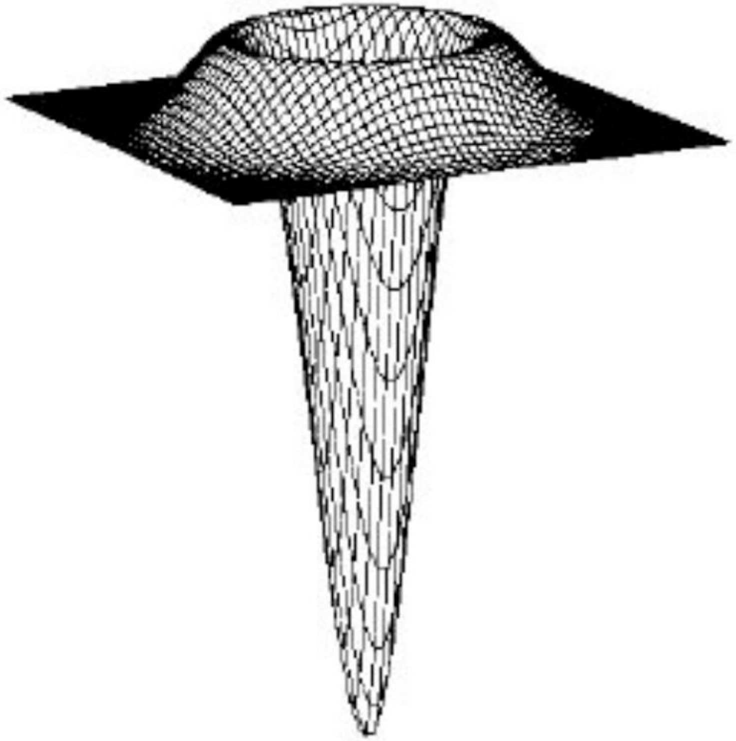




Derivative of Gaussian



Laplacian of Gaussian



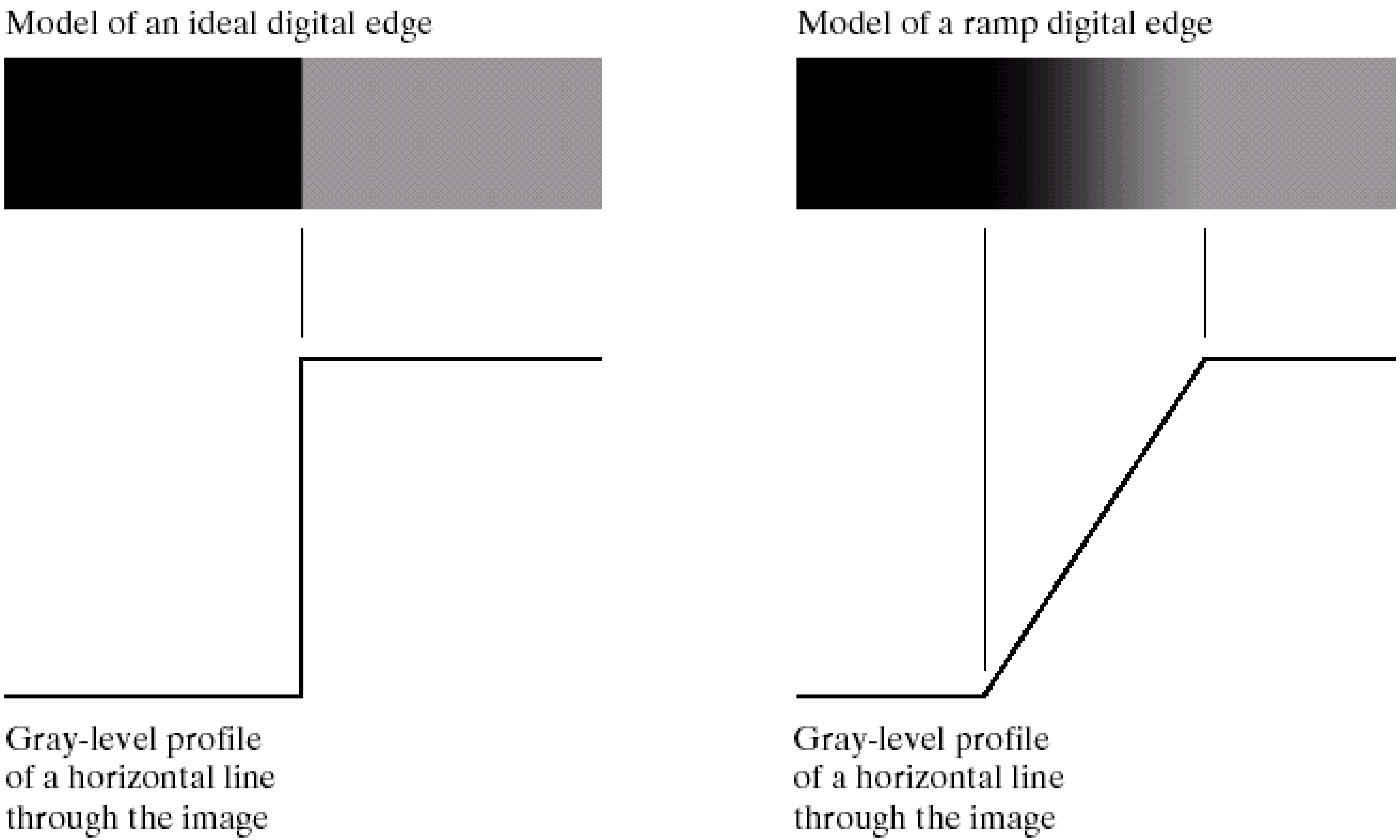
Repeated Convolutions: Blur & Derivatives

$$\frac{\partial f}{\partial x} = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \frac{\partial f}{\partial y} = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

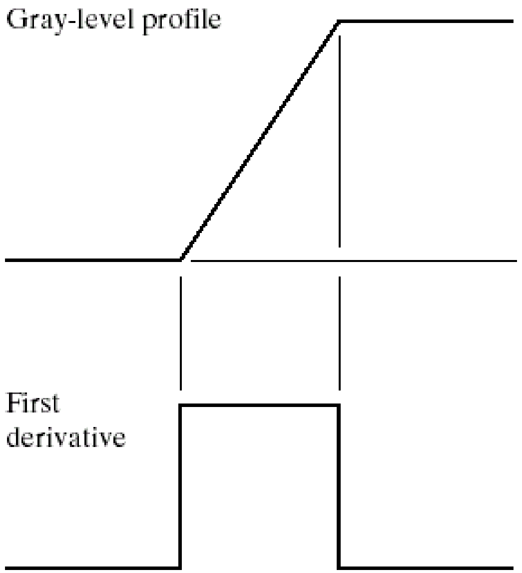
Home Exercise: Compute ***A*** *,* ***B*** *,* ***C*** *, and* ***D***

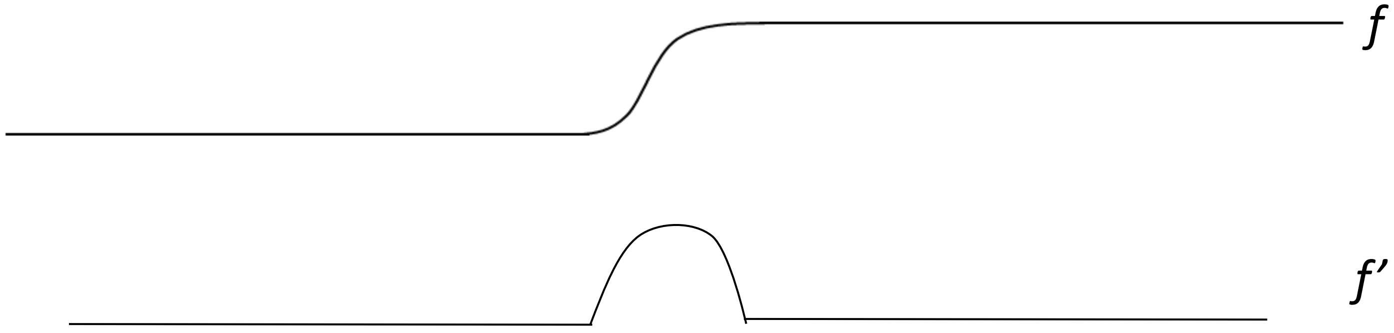
- $f * (½\ \ ½) * (½\ \ ½) = f * A$
- $f * (½\ \ ½) * (1\ -1) = f * B$
- $f * (½\ \ ½) * (½\ \ ½)^T = f * C$
- $f * \quad A \quad * \quad B^T \quad = f * D$

Edges in a picture



Derivative Response to Edges





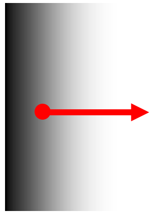
The Gradient

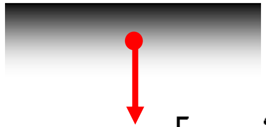
Gradient: The vector of x & y derivatives

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

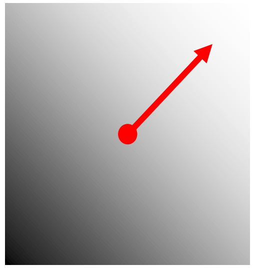
The direction of most rapid change in intensity

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





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



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

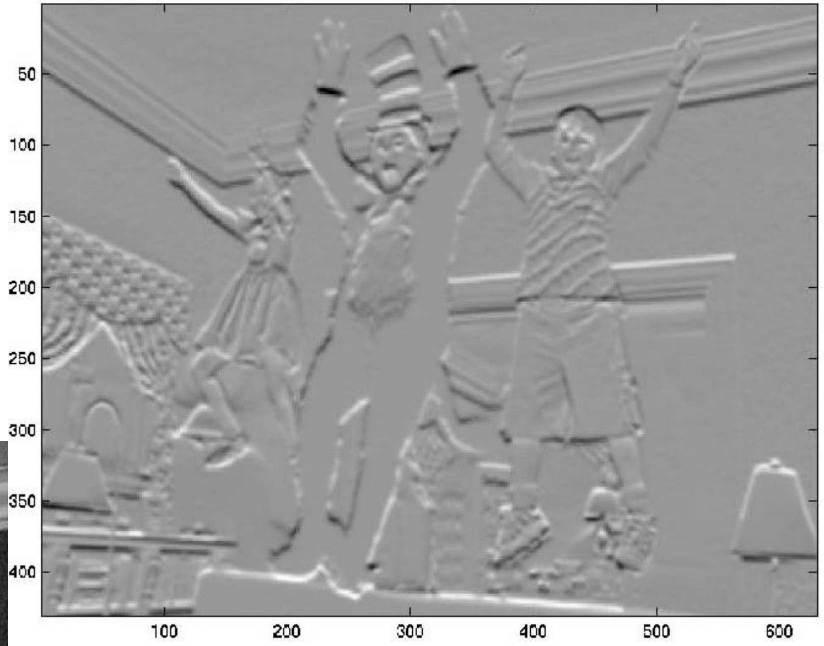
Edge (Gradient) Magnitude

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

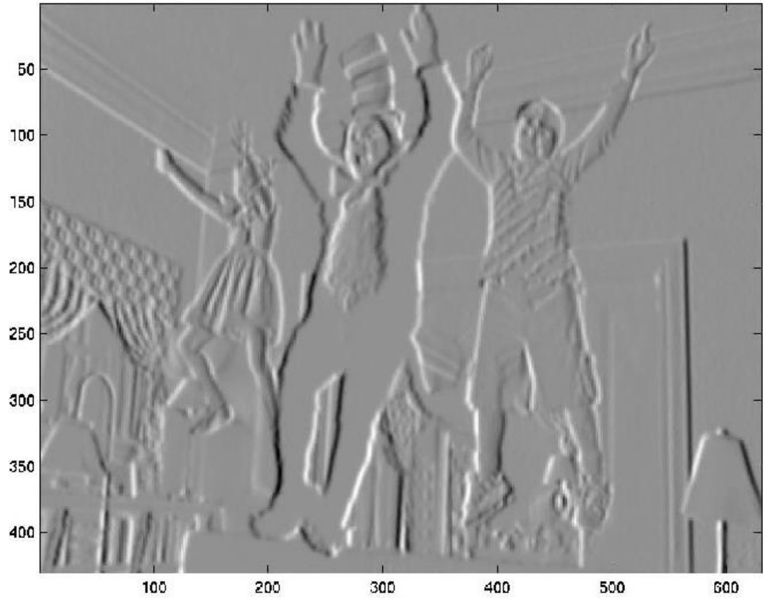
Edge (Gradient) Direction

$$\alpha = \tan^{-1}\left(\left(\frac{\partial f}{\partial y}\right) / \left(\frac{\partial f}{\partial x}\right)\right)$$

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



Zero is gray



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

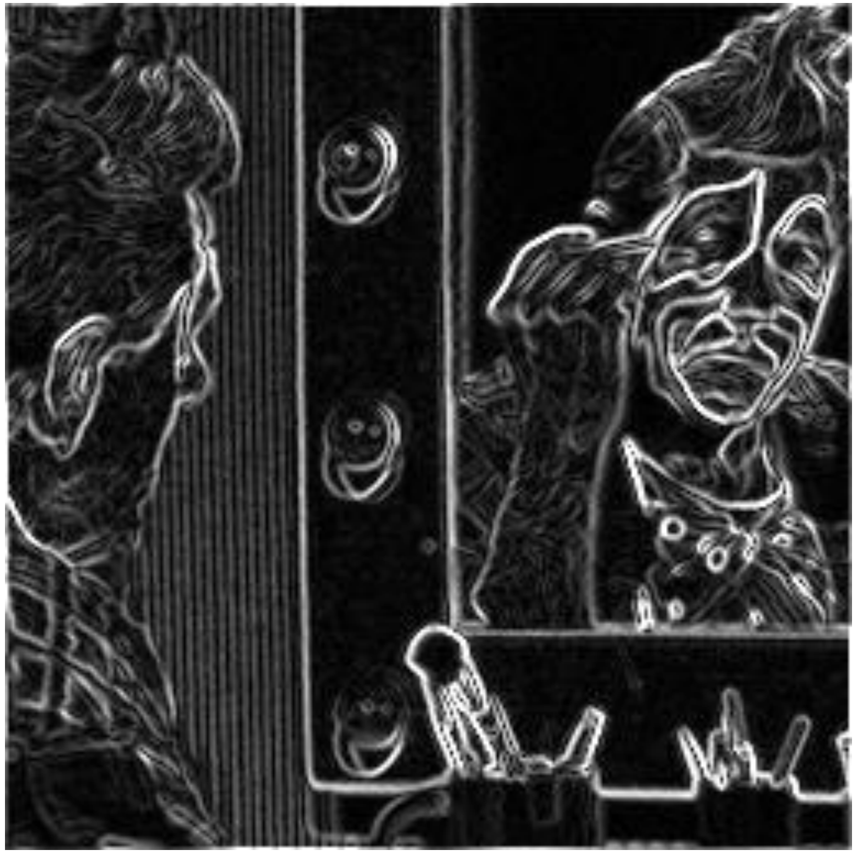
$$|\nabla f| = \sqrt{(I_x)^2 + (I_y)^2}$$

Zero is black

Edge Detection using Gradient

Original

|Gradient|



Second Derivatives

1^{st} x Derivative, convolution with $\begin{pmatrix} 1 & -1 \end{pmatrix}$

2^{nd} x Derivative, convolve again with $\begin{pmatrix} 1 & -1 \end{pmatrix}$ giving $\begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$

1^{st} y Derivative, convolution with $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$

2^{nd} y Derivative, convolve again with $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ giving $\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$

Convolution $\frac{\partial}{\partial^2 x}$: $\begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$ Convolution $\frac{\partial}{\partial^2 y}$: $\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$

Laplacian

Equation: $$\nabla^2 f = \frac{\partial}{\partial x^2} f + \frac{\partial}{\partial y^2} f$$

Convolution: $$\begin{pmatrix}1 & -2 & 1\end{pmatrix} + \begin{pmatrix}1 \\ -2 \\ 1\end{pmatrix} = \begin{pmatrix}0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0\end{pmatrix}$$

Difference between a pixel and its neighborhood

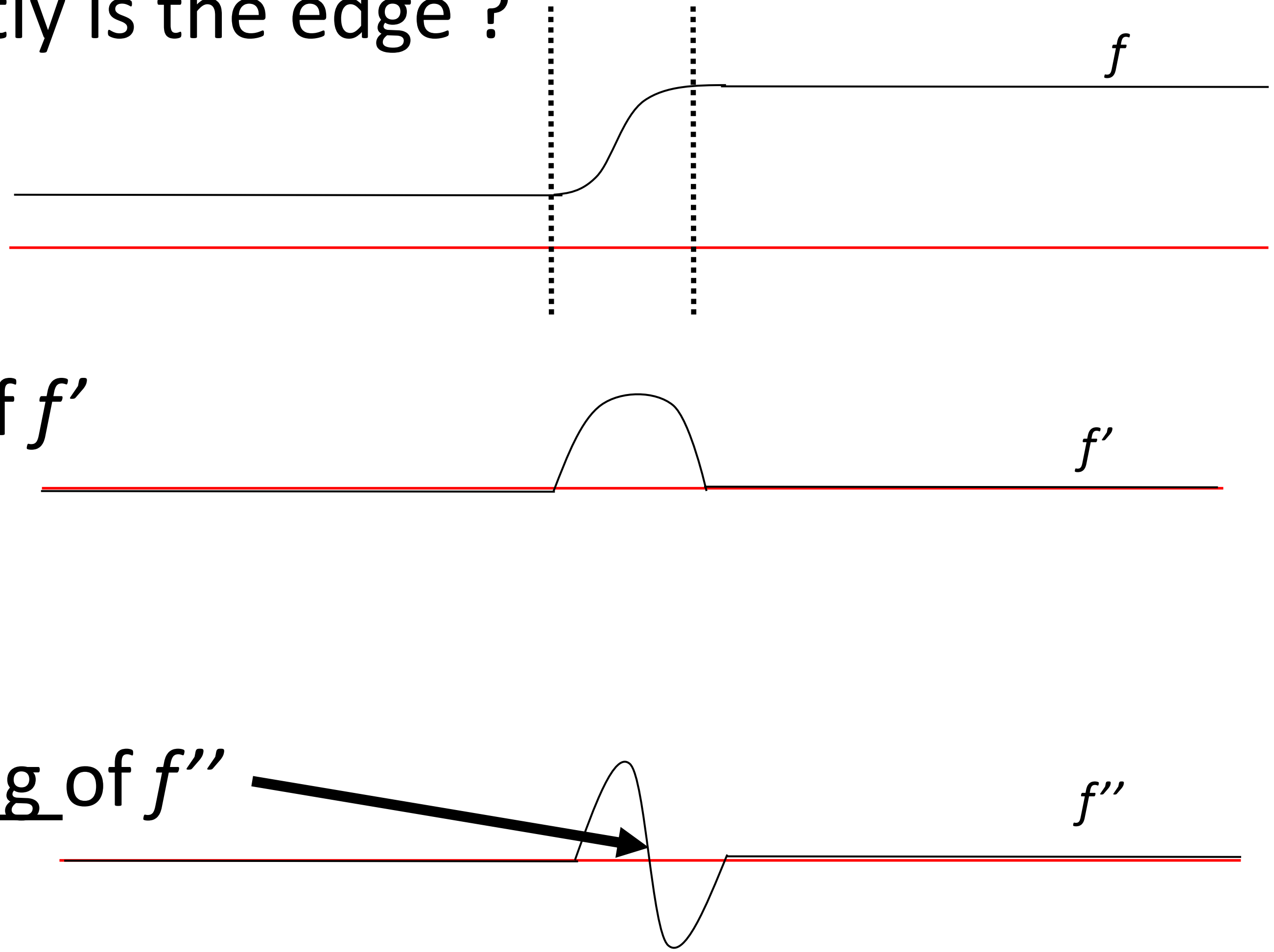
Alternative Laplacian: $$\begin{pmatrix}1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1\end{pmatrix}$$

Edge Localization with Zero Crossing

Where exactly is the edge ?

Maximum of f'

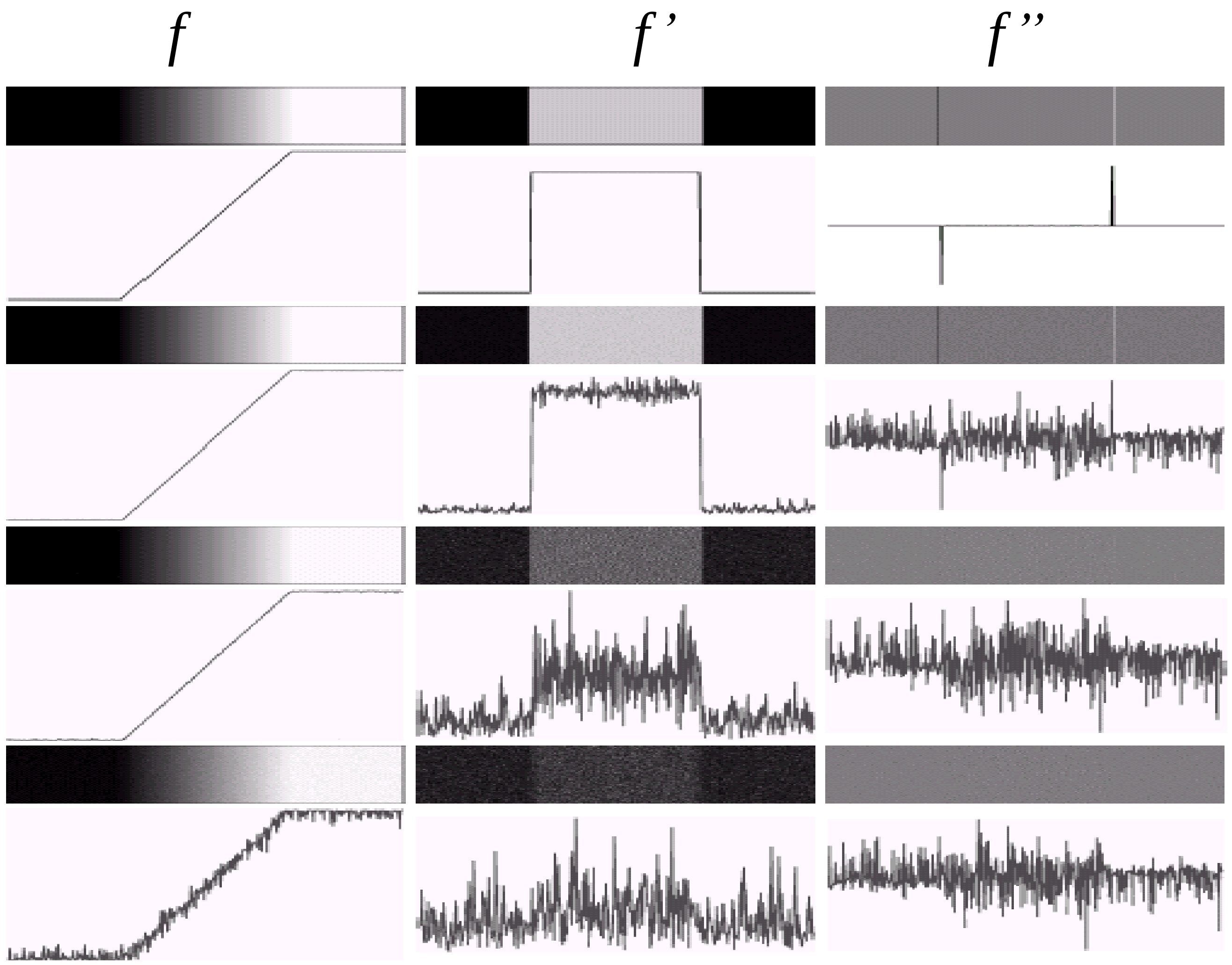
Zero crossing of f''



Problem: f'' is very noisy ➡ Smooth f first !

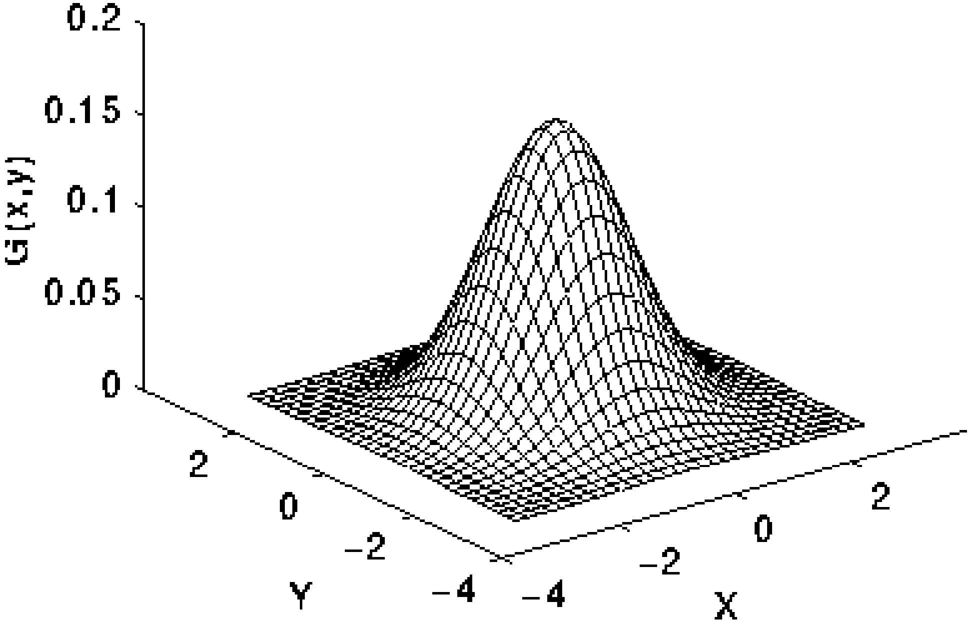
Effect of Noise on Derivatives

Derivatives amplify noise



Smoothing with a 2D Gaussian

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



1 1
1 2 1
1 3 3 1
1 4 6 4 1

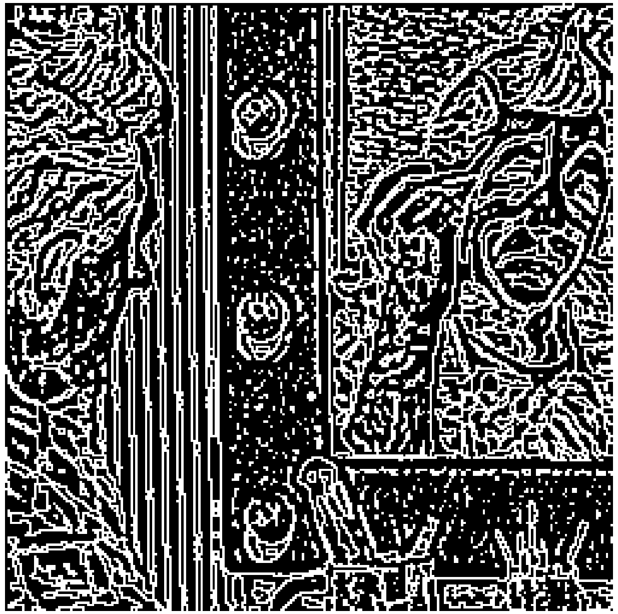
(Gaussian is approximated by binomial coefficients. **Why?**)

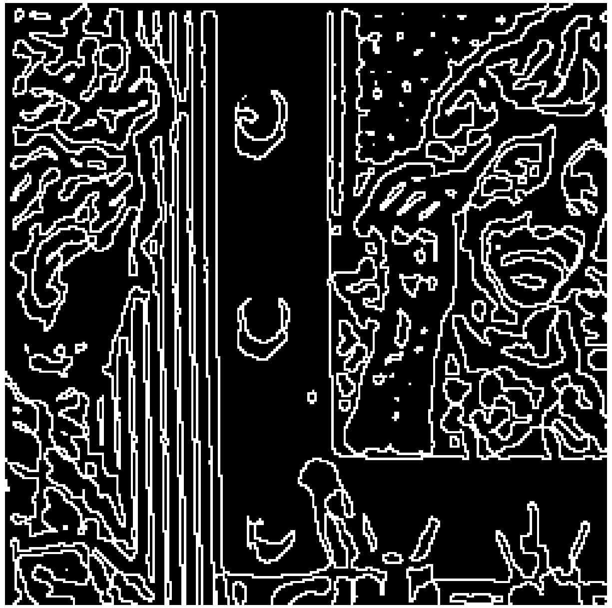
Zero Crossing: Smoothing Effect

Why are all lines closed curves?

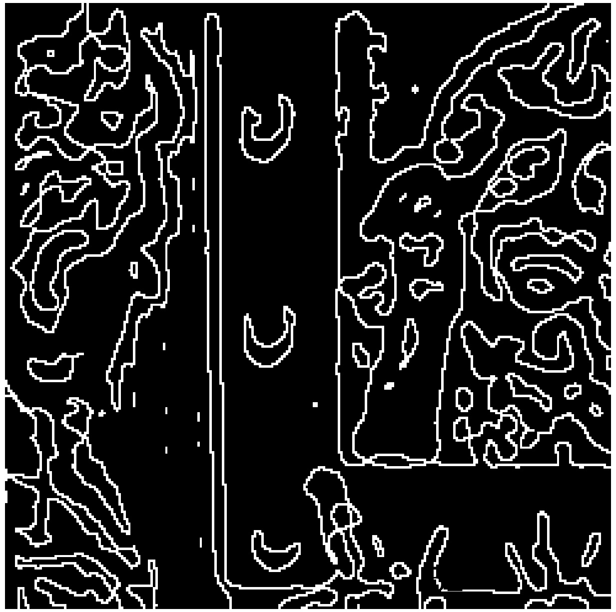
Level Curves – All pixels with same value
Like elevation curves in topological map

Little Blur

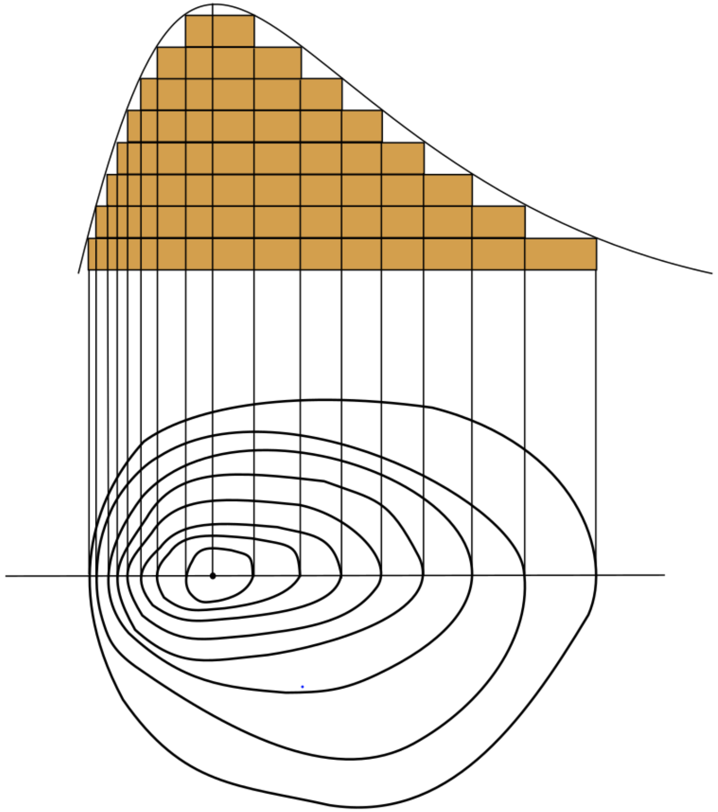


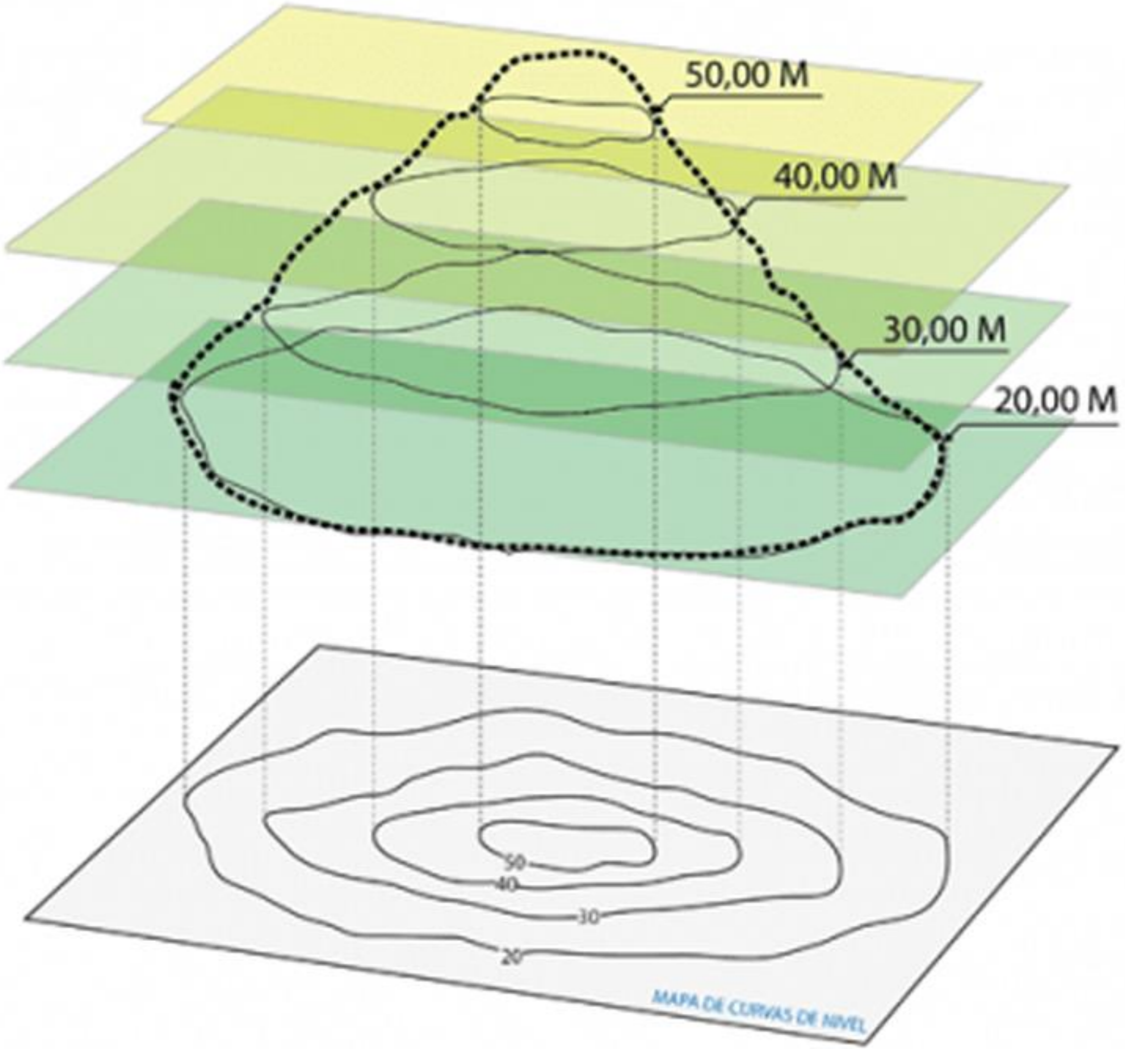


Much Blur



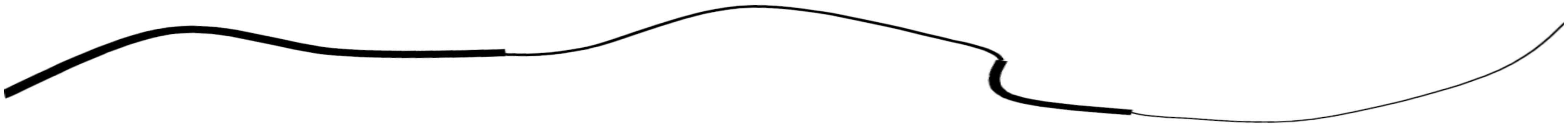
Level Curves - Contour Lines





Canny Edge Detection

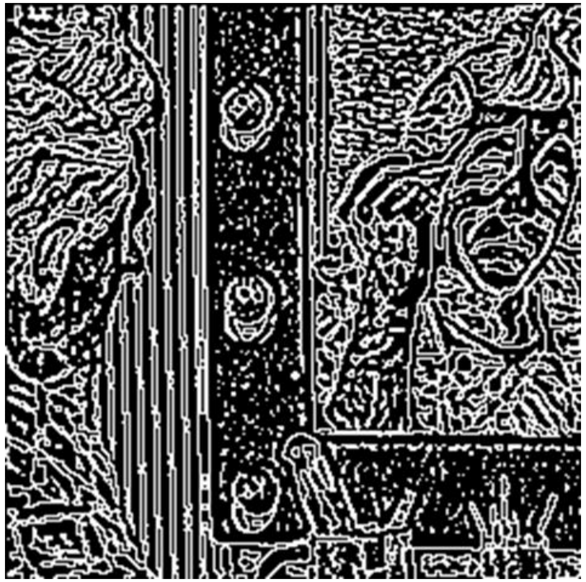
- Computing image derivatives f_x, f_y
  - Smoothing with a Gaussian.
  - Using simple derivative kernels *(1, -1), ½(1, 0, -1).*
- Computing edge direction: $\tan(\alpha) = f_y / f_x$
- Edge point is the <u>local maxima</u> in edge direction.
  - E.g. zero crossing (to get an edge with width 1)
- Use only edge points with gradient <u>above threshold</u>
- <u>Hysteresis</u>: Edge linking with two thresholds: high and low
  - Low threshold accepted only if neighbor accepted



- Blur with different Gaussian widths will give different results

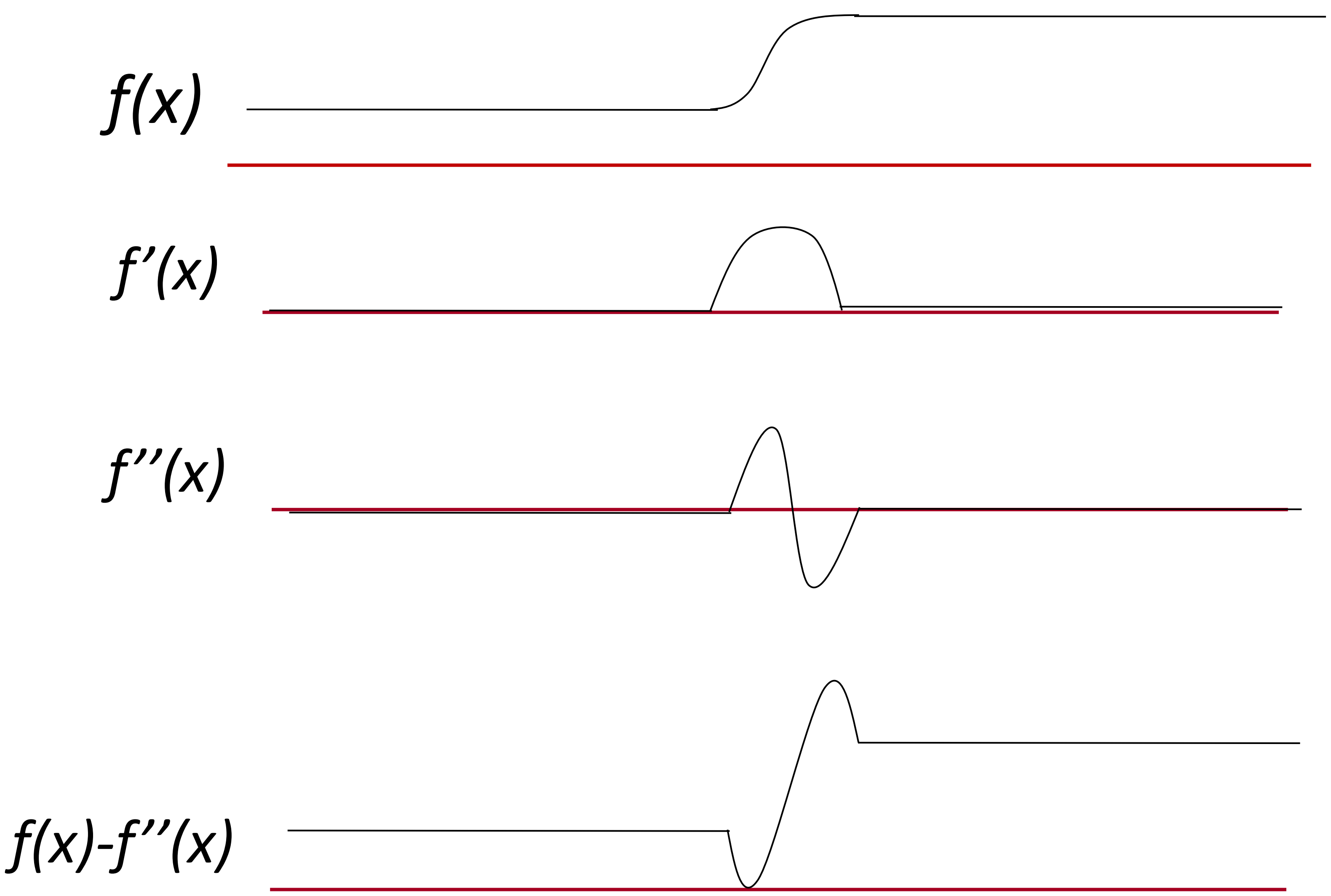
Canny Example

Zero Crossings



Canny

Sharpening by Subtracting the Laplacian



Sharpening by Subtracting the Laplacian

Equation:

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

Convolution:

$$\begin{pmatrix} 1 & -2 & 1 \end{pmatrix} + \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Subtracting the Laplacian from the image ($0 < a < 1$): (Check $a = 0.25$)

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - a \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -a & 0 \\ -a & 1 + 4a & -a \\ 0 & -a & 0 \end{pmatrix}$$

Sharpening Example

Convolutional Neural Networks (CNN)

- Historically, people designed convolution kernels for many image processing tasks ("hand crafted features")
- Recently, CNN's were introduced to **learn** convolution kernels from desired input-output data samples.
- A common choice is to keep the kernel size at $3{\times}3$ or $5{\times}5$.
- When input has several layers (E.g. RGB = 3 layers), a different kernel is used for each layer, and the results are added together.
- CNN includes different layers, Convolution, Pooling (sampling), activation (e.g. RelU)