Image processing

Reading

- Jain, Kasturi, Schunck, *Machine Vision*. McGraw-Hill, 1995. Sections 4.2-4.4, 4.5(intro), 4.5.5, 4.5.6, 5.1-5.4.

Image processing

- An **image processing** operation typically defines a new image *g* in terms of an existing image *f*.
- The simplest operations are those that transform each pixel in isolation. These pixel-to-pixel operations can be written:

$$g(x,y) = t(f(x,y))$$

- Examples: threshold, RGB → grayscale
- Note: a typical choice for mapping to grayscale is to apply the YIQ television matrix and keep the Y.

$$\begin{bmatrix} \mathbf{Y} \\ I \\ Q \end{bmatrix} = \begin{bmatrix} \mathbf{0.299} & \mathbf{0.587} & \mathbf{0.114} \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} \mathbf{R} \\ \mathbf{G} \\ \mathbf{B} \end{bmatrix}$$

Pixel movement

- Some operations preserve intensities, but move pixels around in the image

$$g(x, y) = f(\tilde{x}(x, y), \tilde{y}(x, y))$$

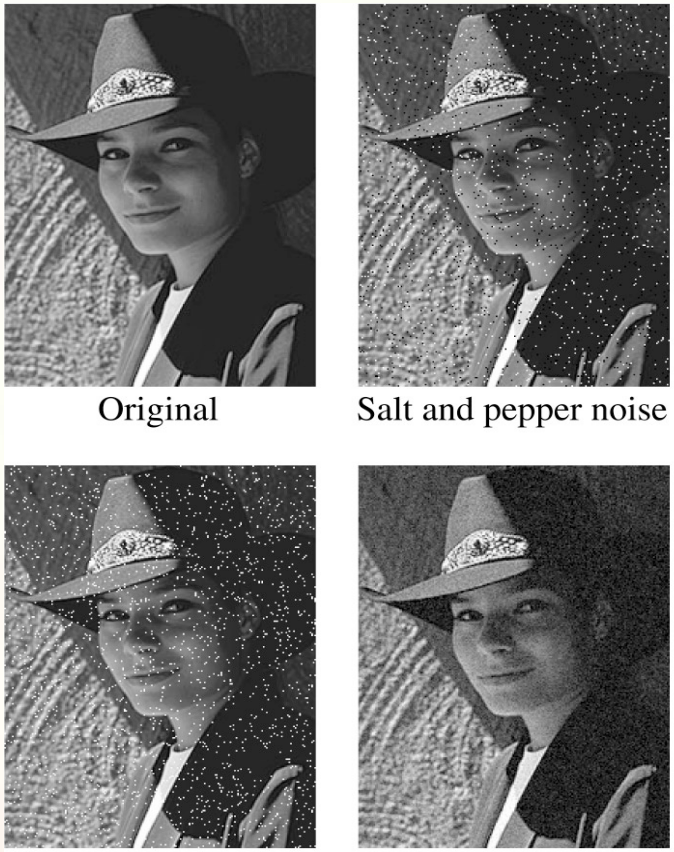
- Examples: many amusing warps of images

[Show image sequence.]

Noise

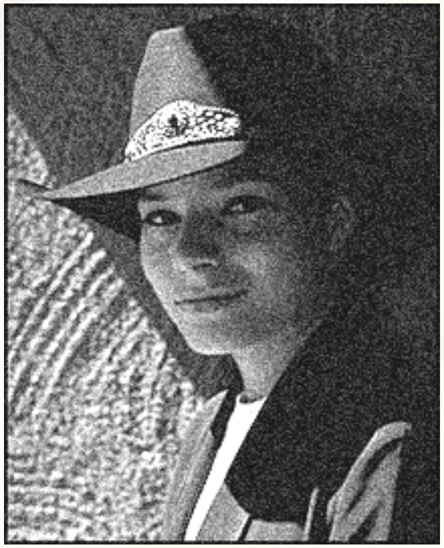
- Image processing is also useful for noise reduction and edge enhancement. We will focus on these applications for the remainder of the lecture…

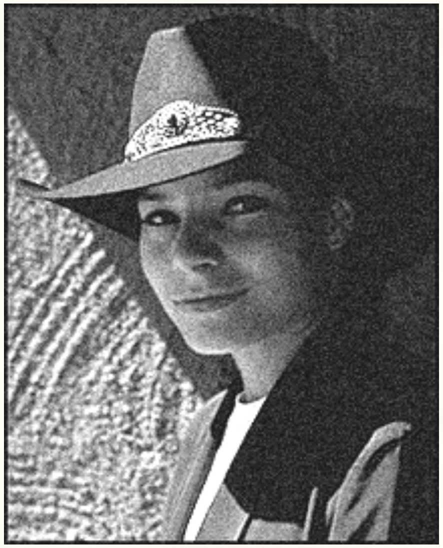
- Common types of noise:
  - **Salt and pepper noise**: contains random occurrences of black and white pixels
  - **Impulse noise:** contains random occurrences of white pixels
  - **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution



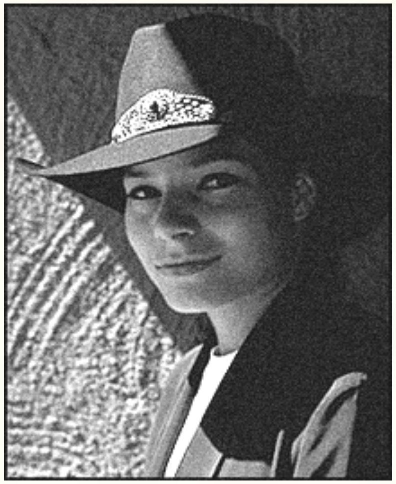
Original — Salt and pepper noise — Impulse noise — Gaussian noise

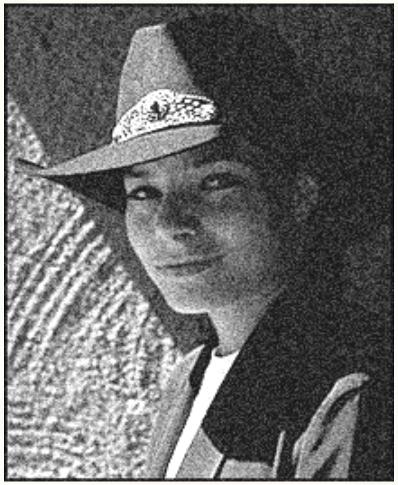
Ideal noise reduction





Ideal noise reduction





Practical noise reduction

- How can we “smooth” away noise in a single image?

- Is there a more abstract way to represent this sort of operation? *Of course there is!*

Discrete convolution

- For a digital signal, we define **discrete convolution** as:

$$
\begin{aligned}
g[i] &= f[i] * h[i] \\
&= \sum_{i'} f[i']h[i-i'] \\
&= \sum_{i'} f[i']\tilde{h}[i'-i]
\end{aligned}
$$

where $\tilde{h}[i] = h[-i]$

Discrete convolution in 2D

- Similarly, discrete convolution in 2D becomes:

$$
\begin{aligned}
g[i,j] = f[i,j] * h[i,j] \\
&= \sum_{i'} \sum_{j'} f[i',j'] h[i-i', j-j'] \\
&= \sum_{i'} \sum_{j'} f[i',j'] \widehat{h}[i'-i, j'-j]
\end{aligned}
$$

where $\widehat{h}[i,j] = h[-i,-j]$

Convolution representation

- Since f and h are defined over finite regions, we can write them out in two-dimensional arrays:

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

X .2	X 0	X .2
X 0	X .2	X 0
X .2	X 0	X .2

- Note: *This is not matrix multiplication*!
- **Q**: What happens at the edges?

Mean filters

- How can we represent our noise-reducing averaging filter as a convolution diagram (know as a **mean filter**)?

Effect of mean filters

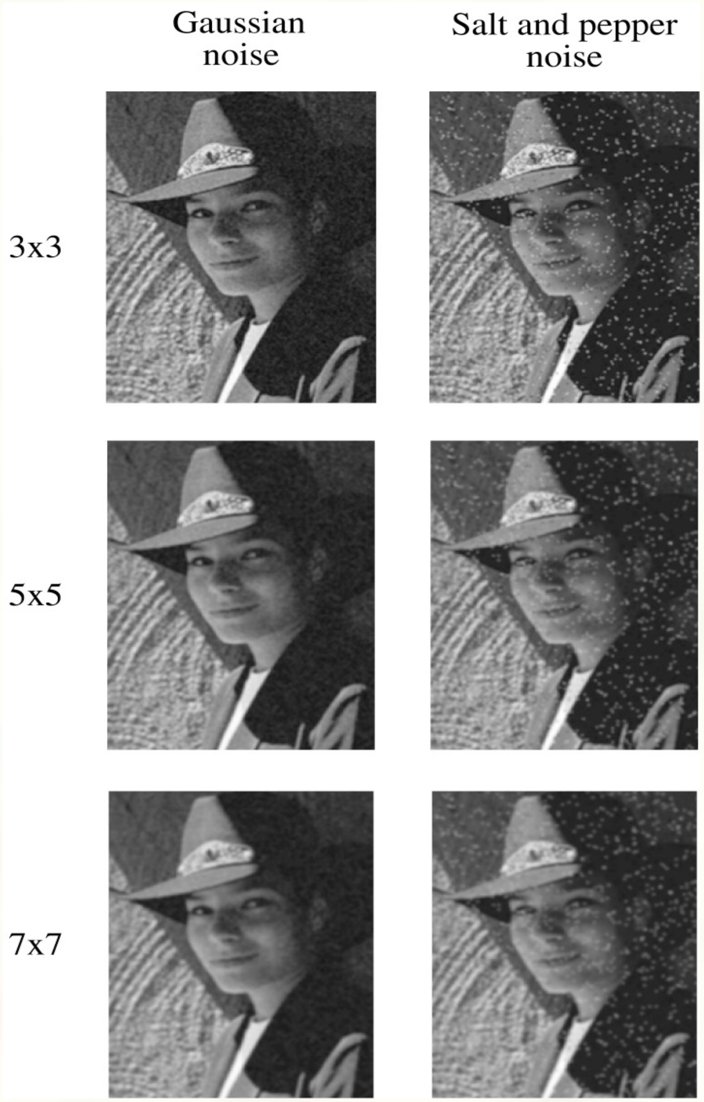
Gaussian filters

- Gaussian filters weigh pixels based on their distance from the center of the convolution filter. In particular:

$$h[i, j] = \frac{e^{-(i^2 + j^2)/(2\sigma^2)}}{C}$$

- This does a decent job of blurring noise while preserving features of the image.
- What parameter controls the width of the Gaussian?
- What happens to the image as the Gaussian filter kernel gets wider?
- What is the constant *C*? What should we set it to?

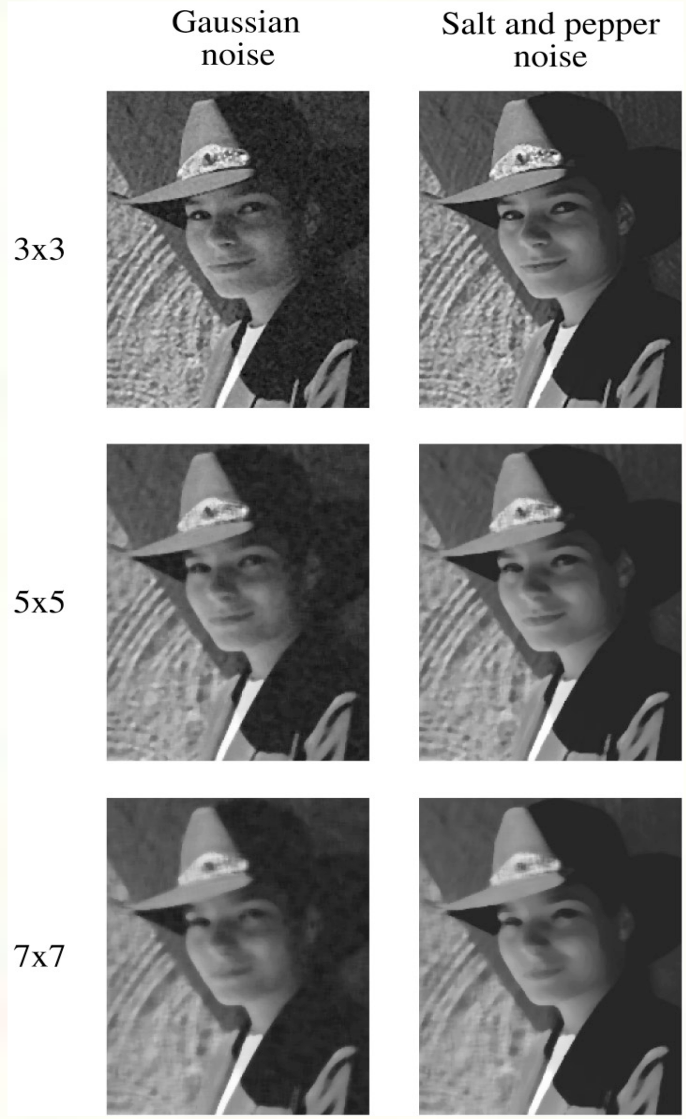
Effect of Gaussian filters



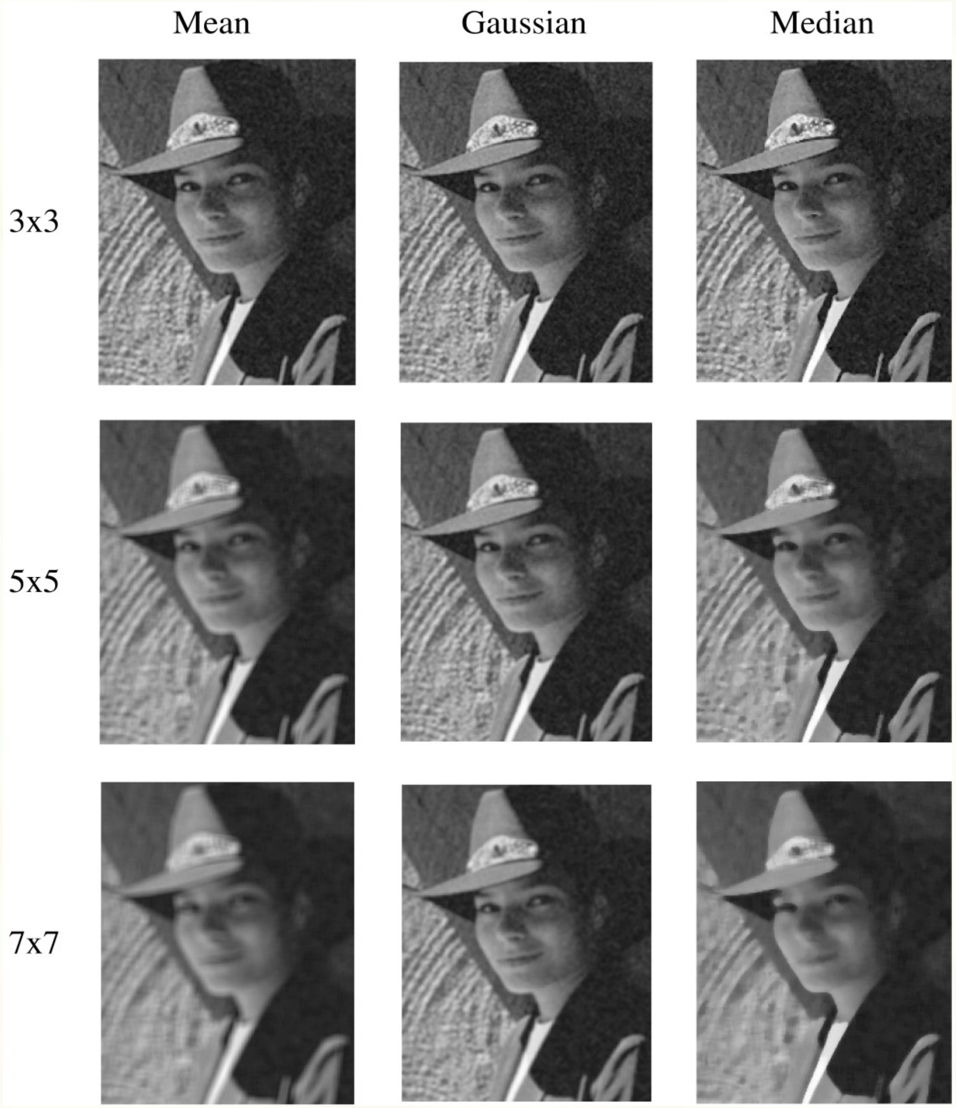
Median filters

- A **median filter** operates over an *m*x*m* region by selecting the median intensity in the region.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?

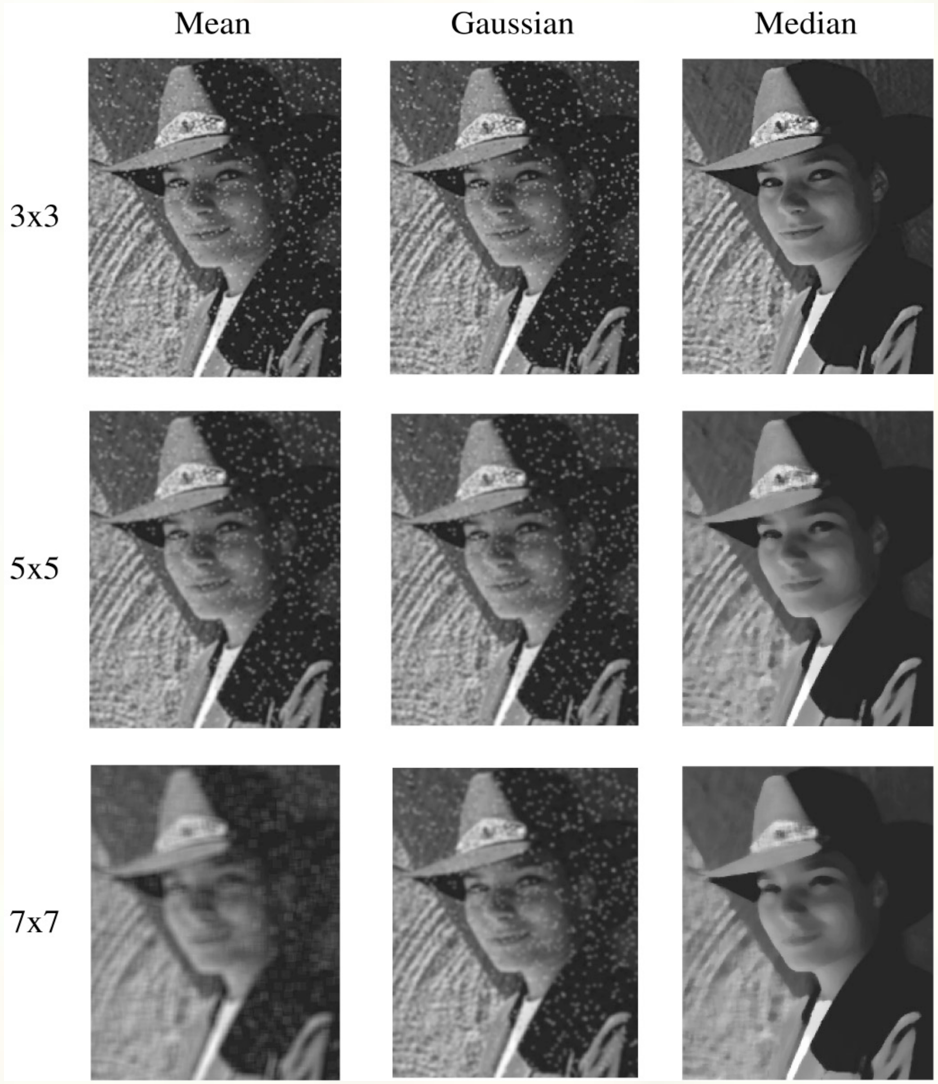
Effect of median filters



Comparison: Gaussian noise



Comparison: salt and pepper noise

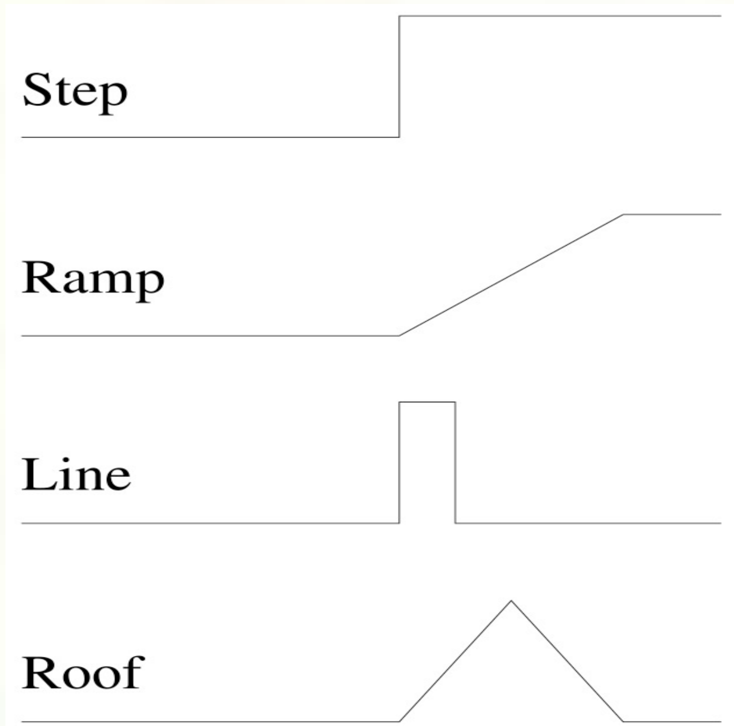


Edge detection

- One of the most important uses of image processing is **edge detection:**
  - Really easy for humans
  - Really difficult for computers

  - Fundamental in computer vision
  - Important in many graphics applications

What is an edge?



- **Q**: How might you detect an edge in 1D?

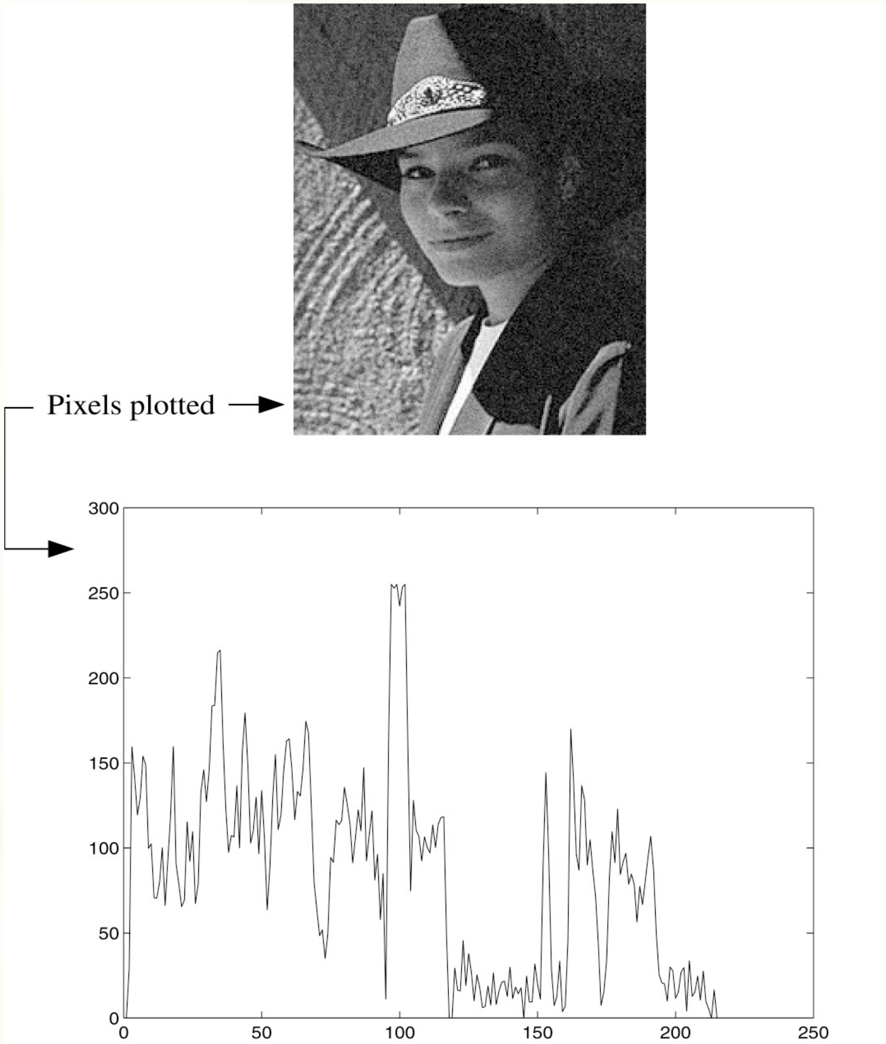
Gradients

- The **gradient** is the 2D equivalent of the derivative:

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

- Properties of the gradient
  - It's a vector
  - Points in the direction of maximum increase of f
  - Magnitude is rate of increase
- How can we approximate the gradient in a discrete image?

Less than ideal edges



Steps in edge detection

- Edge detection algorithms typically proceed in three or four steps:
  - **Filtering**: cut down on noise
  - **Enhancement**: amplify the difference between edges and non-edges
  - **Detection**: use a threshold operation
  - **Localization** (optional): estimate geometry of edges beyond pixels

Edge enhancement

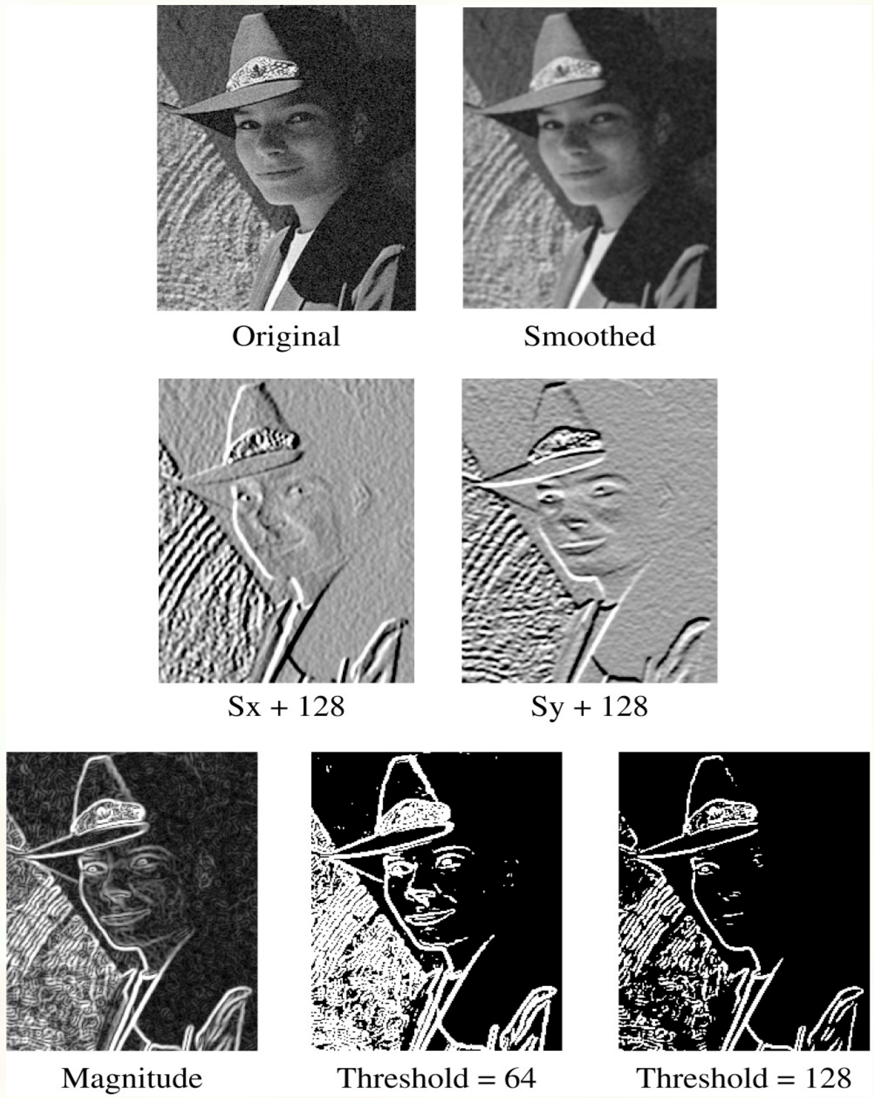
- A popular gradient magnitude computation is the **Sobel operator**:

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

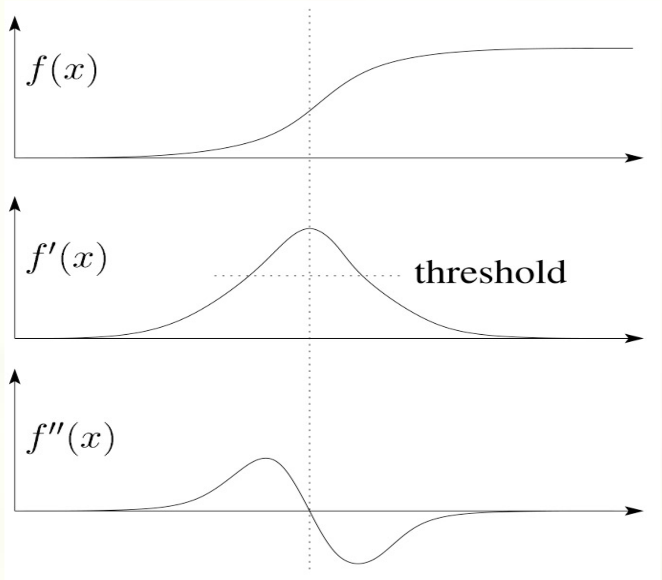
$$s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- We can then compute the magnitude of the vector (s_x, s_y).

Results of Sobel edge detection



Second derivative operators



- The Sobel operator can produce thick edges. Ideally, we're looking for infinitely thin boundaries.
- An alternative approach is to look for local extrema in the first derivative: places where the change in the gradient is highest.
- **Q**: A peak in the first derivative corresponds to what in the second derivative?
- **Q**: How might we write this as a convolution filter?

Localization with the Laplacian

- An equivalent measure of the second derivative in 2D is the **Laplacian**:

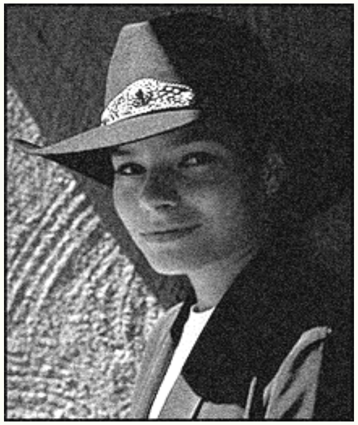
$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Using the same arguments we used to compute the gradient filters, we can derive a Laplacian filter to be:

$$\Delta^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Zero crossings of this filter correspond to positions of maximum gradient. These zero crossings can be used to localize edges.

Localization with the Laplacian



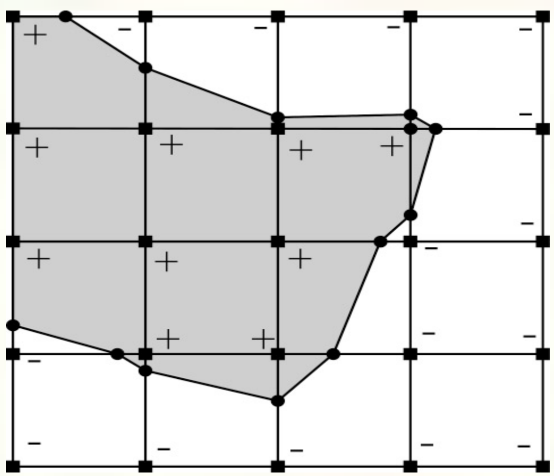
Original

Smoothed

Laplacian (+128)

Marching squares

- We can convert these signed values into edge contours using a "marching squares" technique:



Sharpening with the Laplacian


Original


Laplacian (+128)


Original + Laplacian


Original - Laplacian

Why does the sign make a difference?

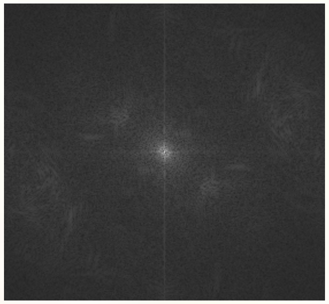
How can you write each filter that makes each bottom image?

Spectral impact of sharpening

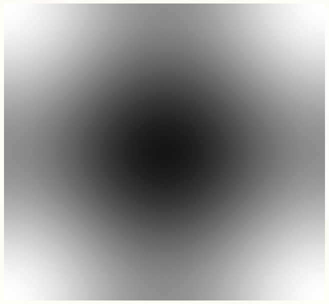
We can look at the impact of sharpening on the Fourier spectrum:

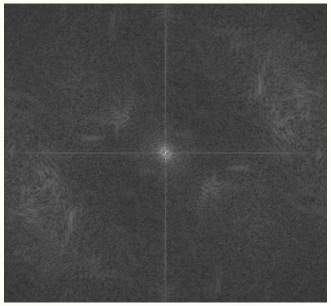
Spatial domain

Frequency domain



$$\delta - \Delta^2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$





Summary

- What you should take away from this lecture:
  - The meanings of all the boldfaced terms.
  - How noise reduction is done
  - How discrete convolution filtering works
  - The effect of mean, Gaussian, and median filters
  - What an image gradient is and how it can be computed
  - How edge detection is done
  - What the Laplacian image is and how it is used in either edge detection or image sharpening

Next time: Affine Transformations

- Topic:
- How do we represent the rotations, translations, etc. needed to build a complex scene from simpler objects?
- Read:
  - Watt, Section 1.1.

  Optional:
  - Foley, et al, Chapter 5.1-5.5.
  - David F. Rogers and J. Alan Adams, Mathematical Elements for Computer Graphics, 2nd Ed., McGraw-Hill, New York, 1990, Chapter 2.