# Image Processing

# Reading

Course Reader:

Jain et. Al. ***Machine Vision***
Chapter 4 and 5

# Definitions

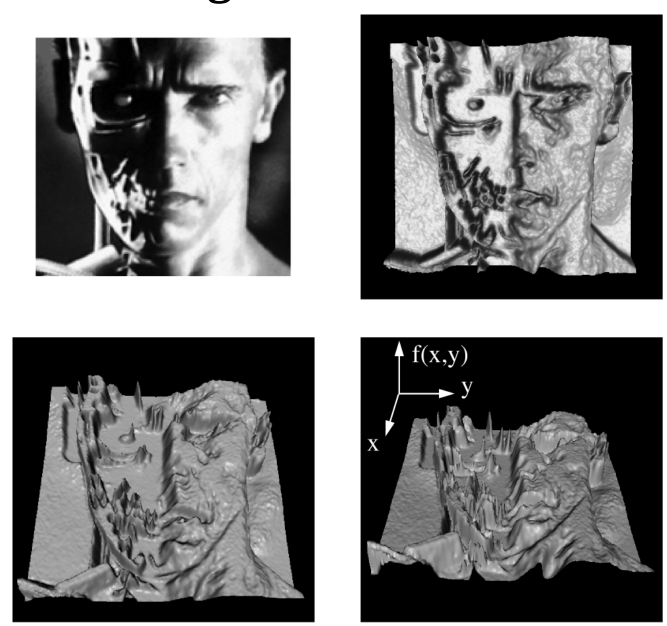
- Many graphics techniques that operate only on images
- **Image processing**: operations that take images as input, produce images as output
- In its most general form, an **image** is a function *f* from $R^2$ to R
  - $f(x, y)$ gives the intensity of a channel at position $(x, y)$ defined over a rectangle, with a finite range:

$$f: [a,b] \times [c,d] \rightarrow [0,1]$$

  - A color image is just three functions pasted together:

$$\boldsymbol{f}(x, y) = (f_r(x, y), f_g(x, y), f_b(x, y))$$

# Images as Functions

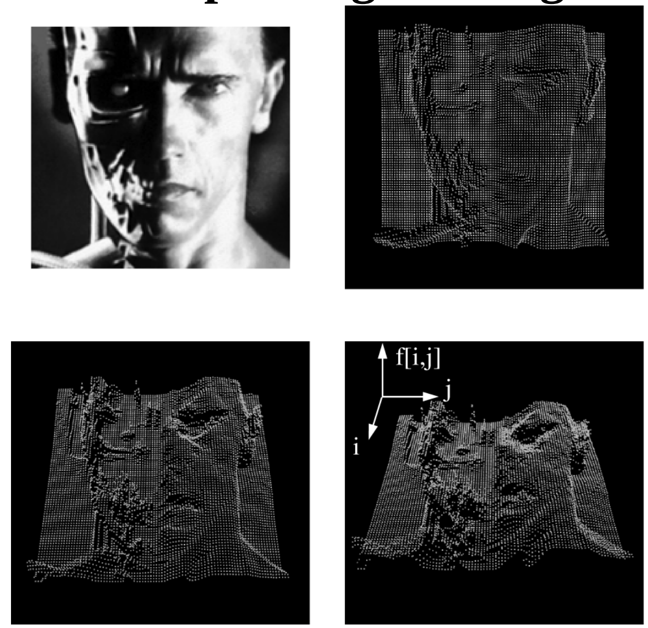


# What is a digital image?

- In computer graphics, we usually operate on **digital** (**discrete**) images:
  - **Sample** the space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- If our samples are $d$ apart, we can write this as:

$$f'[i,j] = Quantize(f(i \cdot d, j \cdot d))$$

# Sampled digital image




# 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:

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

- Example: threshold, RGB → grayscale

# Pixel Movement

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

$$g(x,y) = f(u(x,y), v(x,y))$$

- Examples: many amusing warps of images

# Multiple input images

- Some operations define a new image *g* in terms of *n* existing images ($f_1, f_2, \ldots, f_n$), where *n* is greater than 1
- Example: cross-dissolve between 2 input images

# Noise

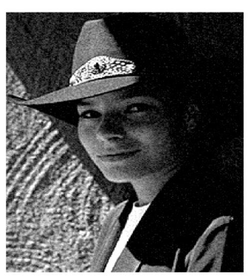
- Common types of noise:
  - **Salt and pepper noise**: contains random occurrences of black and white pixels
  - **Impulse noise:** contains random occurrences of white pixels
  - **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

# Noise Examples

Original

Salt and pepper noise

Impulse noise



Gaussian noise

# Ideal noise reduction

# Ideal noise reduction

# Noise Reduction

- How can we "smooth" away noise?

# Convolution

- Convolution is a fancy way to combine two functions.
  - Think of $f$ as an image and $h$ as a "smear" operator
  - $g$ determines a new intensity at each point in terms of intensities of a neighborhood of that point

$$
\begin{aligned}
g(x) &= f(x) * h(x) \\
&= \int_{-\infty}^{\infty} f(x')h(x-x')dx' \\
&= \int_{-\infty}^{\infty} f(x')\tilde{h}(x'-x)dx'
\end{aligned}
$$

where $\tilde{h}(x) = h(-x)$.

# Convolution in 2D

In two dimensions, convolution becomes:

$$
\begin{aligned}
g(x,y) &= f(x,y) * h(x,y) \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')h(x-x',y-y')dx'dy' \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')\tilde{h}(x'-x,y'-y)dx'dy'
\end{aligned}
$$

where $\tilde{h}(x,y) = h(-x,-y)$.

# Discrete convolution in 2D

Similarly, discrete convolution in 2D becomes:

$$g[i,j] = f[i,j] * h[i,j] = \sum_k \sum_l f[k,l]h[k-i,l-j] = \sum_k \sum_l f[k,l]\tilde{h}[i-k,j-l]$$

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

# Convolution Representation

- Since $f$ and $h$ are defined over finite regions, we can write them out in two-dimensional arrays:
- Note: *This is not matrix multiplication*!

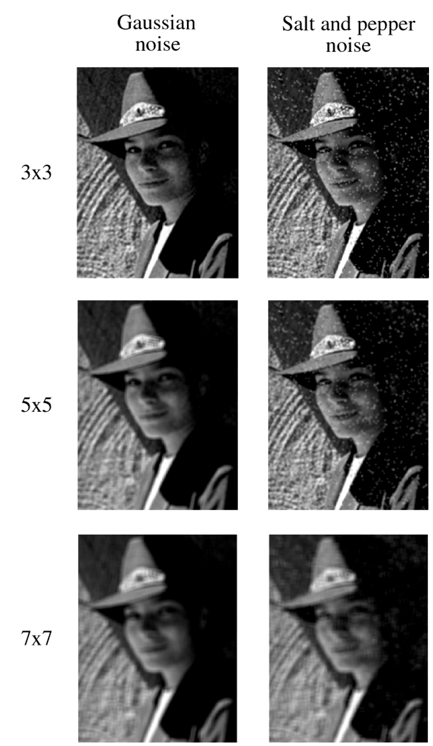
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 62 | 79 | 23 | 119 | 120 | 105 | 4 | 0 |
| 10 | 10 | 9 | 62 | 12 | 78 | 34 | 0 |
| 10 | 58 | 197 | 46 | 46 | 0 | 0 | 48 |
| 176 | 135 | 5 | 188 | 191 | 68 | 0 | 49 |
| 2 | 1 | 1 | 29 | 26 | 37 | 0 | 77 |
| 0 | 89 | 144 | 147 | 187 | 102 | 62 | 208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

| | | |
|---|---|---|
| ×.2 | ×0 | ×.2 |
| ×0 | ×.2 | ×0 |
| ×.2 | ×0 | ×.2 |

# Mean Filters

- How can we represent our noise-reducing averaging filter as a convolution diagram?

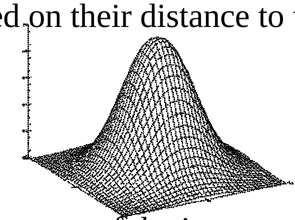
# Mean Filters



# Gaussian Filters

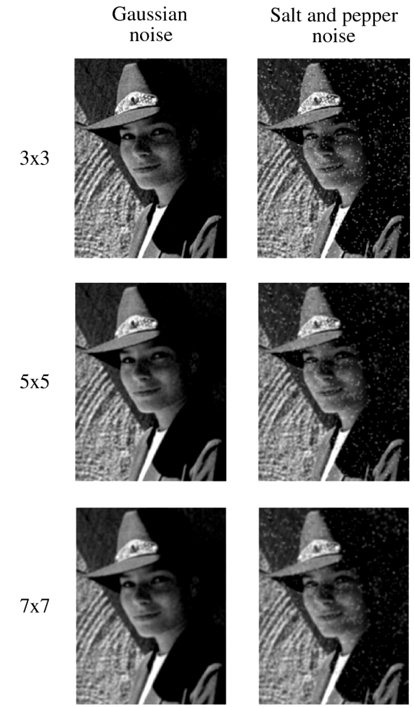
- Gaussian filters weigh pixels based on their distance to the location of convolution.

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



- Blurring noise while preserving features of the image
- Smoothing the same in all directions
- More significance to neighboring pixels
- Width parameterized by σ
- Gaussian functions are separable
- Convolving with multiple Gaussian filters results in a single Gausian filter

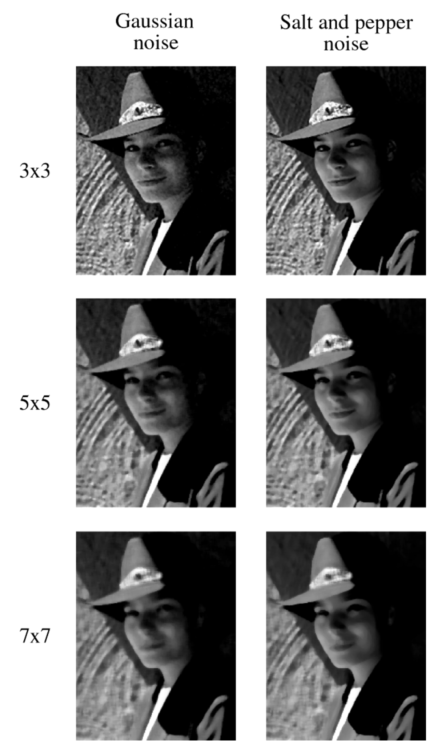
# Gaussian Filters

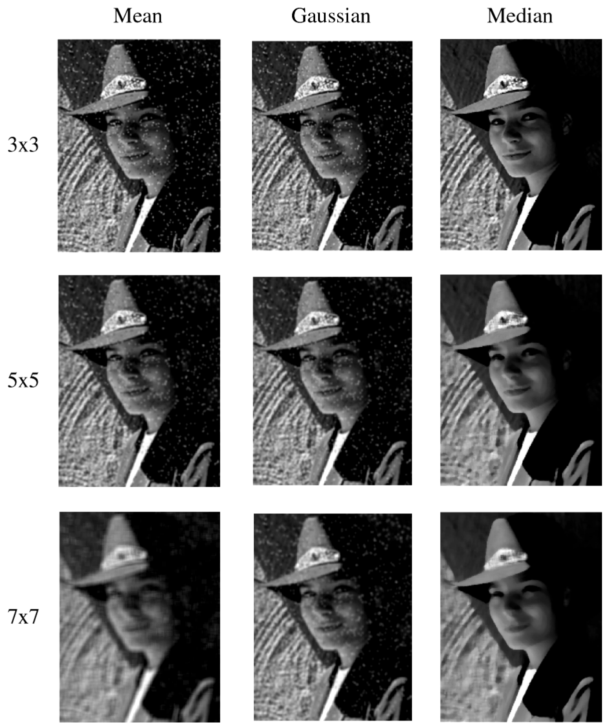



# Median Filters

- A **Median Filter** operates over a $k \times k$ 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?

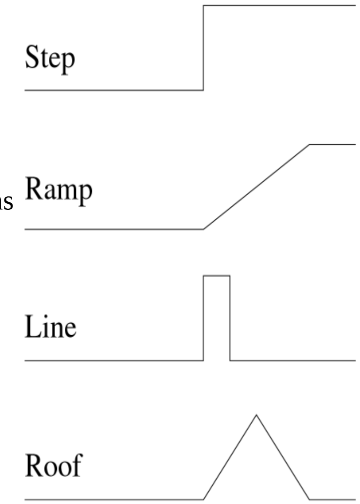
# Median Filters






# 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 defines an edge?



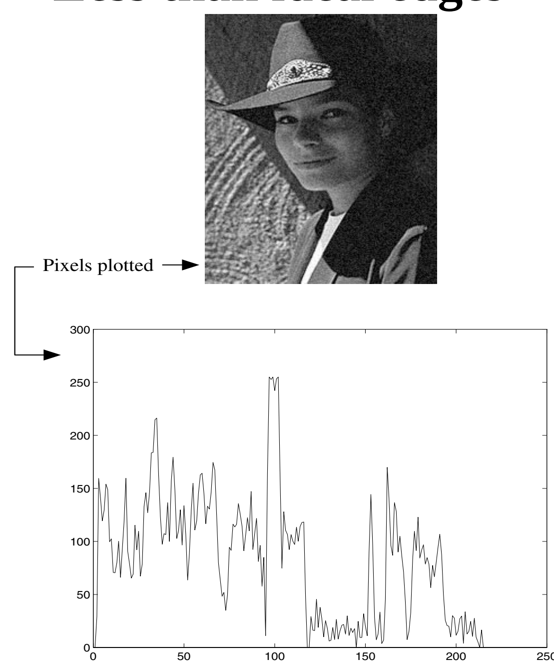

# Gradient

- 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



# Edge Detection Algorithms

- 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

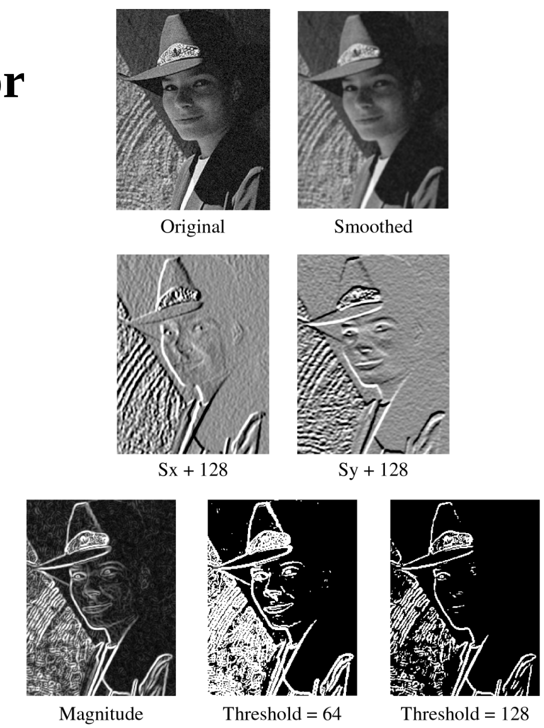
- A popular gradient magnitude computation is the **Sobel operator**:

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

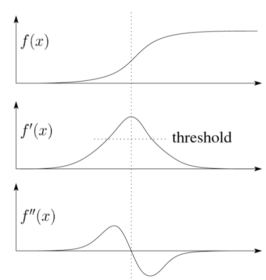
$$s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

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

# Sobel Operator




# 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?

# Localization with the Laplacian

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

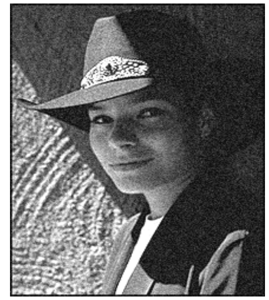
$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- 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)

# Sharpening with the Laplacian

Original

Laplacian (+128)

Original + Laplacian

Original - Laplacian

# Summary

- Formal definitions of image and image processing
- Kinds of image processing: pixel-to-pixel, pixel movement, convolution, others
- Types of noise and strategies for noise reduction
- Definition of convolution and how discrete convolution works
- The effects of mean, median and Gaussian filtering
- How edge detection is done
- Gradients and discrete approximations