Image processing
(part 1)

Reading

Shirley, Ch. 4.

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. [Handout]

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.

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

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…


Original


Salt and pepper noise


Impulse noise

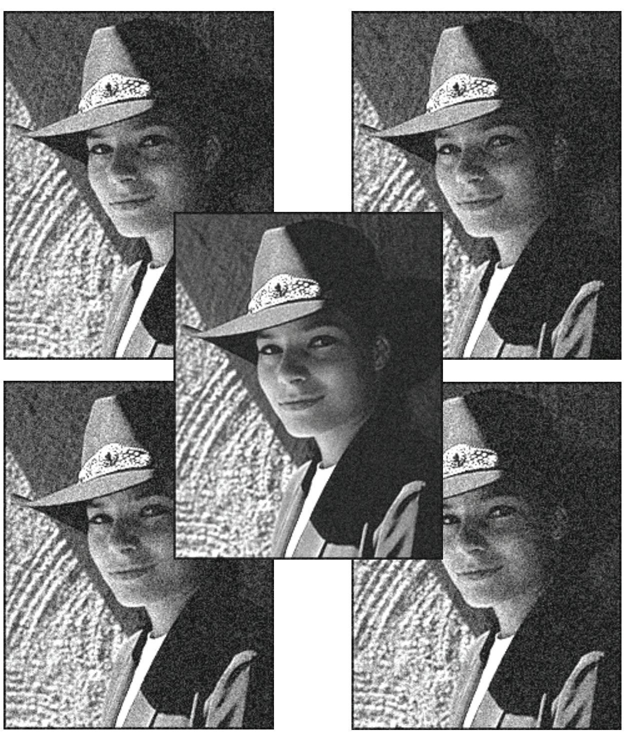

Gaussian noise

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

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

One way to write out discrete signals is in terms of sampling:

$$f(x)\mathrm{III}(x;T) = \sum_{n=-\infty}^{\infty} f(x)\delta(x-nT) = \sum_{n=-\infty}^{\infty} f(nT)\delta(x-nT)$$

Rather than refer to this complicated notation, we will just say that a sampled version of $f(x)$ is represented by a "digital signal" $f[n]$, the collection of samples of $f(nT)$ sifted out by the shah function.

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

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

where $\tilde{h}[n] = h[-n]$.

Discrete convolution, cont'd

What connection does discrete convolution have to continuous convolution?

We're essentially computing

$$f[n]*h[n] \text{ "=" } \left[f(x)\mathrm{III}(x)\right]*\left[h(x)\mathrm{III}(x)\right]$$

for some pair of functions $f(x)$ and $h(x)$ that pass through the samples $f[n]$ and $g[n]$.

It would be nice if this were the same as:

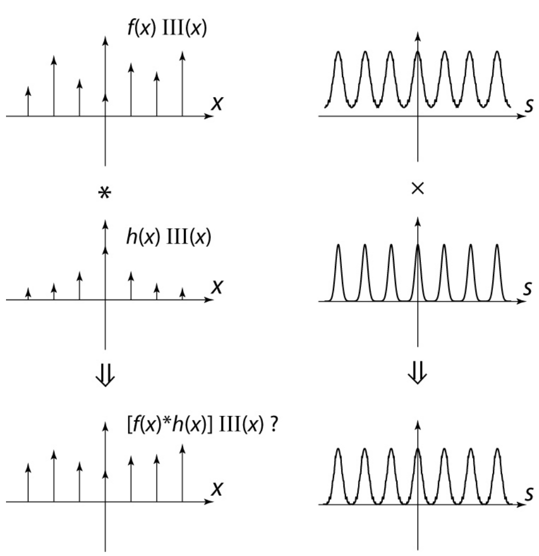
$$\left[f(x)*h(x)\right]\mathrm{III}(x)$$

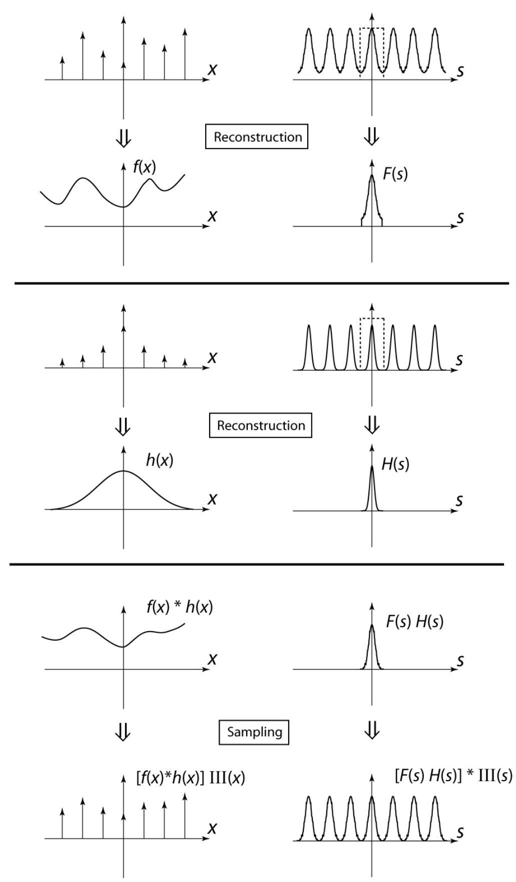
i.e., if we could think in terms of convolving continuous functions and then resampling.

But, is it the same?

Discrete convolution, cont'd

We can analyze this convolution in the Fourier domain:





Discrete Fourier Transform

Recall that the continuous 1D Fourier transform (FT) is:

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx$$

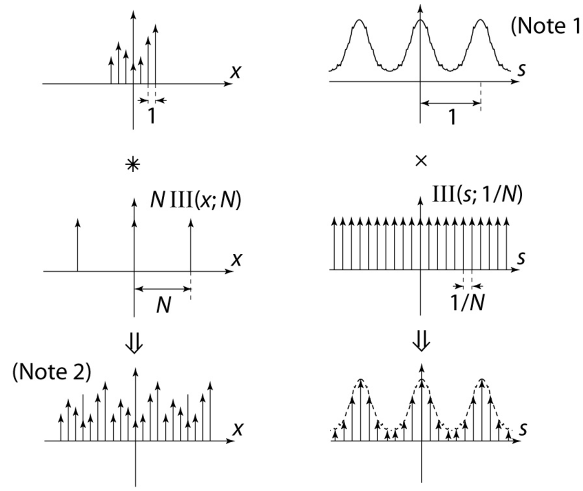
The discrete version of this is the **Discrete Fourier Transform (DFT)**:

$$F[k] = \sum_{n=0}^{N-1} f[n]\, e^{-i\frac{2\pi}{N}kn}$$

where it is assumed that the sampled signal is of finite length N.

Discrete Fourier Transform, cont'd

Is there a connection between the continuous FT and the DFT?



Note 1: horizontal axes not drawn to scale.
Note 2: amplitude scaled by N.

Yes! The DFT is essentially the FT of the input samples, after repeating them along the *x* axis.

Discrete Fourier Transform, cont'd

Summarizing, the continuous FT and inverse FT were:

Spatial domain $\longrightarrow$ $F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx$ $\longrightarrow$ Frequency domain

Spatial domain $\longleftarrow$ $f(x) = \int_{-\infty}^{\infty} F(s) e^{i2\pi sx} ds$ $\longleftarrow$ Frequency domain

and we now have the DFT and inverse DFT:

Spatial domain $\longrightarrow$ $F[k] = \sum_{n=0}^{N-1} f[n]\, e^{-i\frac{2\pi}{N}kn}$ $\longrightarrow$ Frequency domain

Spatial domain $\longleftarrow$ $f[n] = \frac{1}{N}\sum_{k=0}^{N-1} F[k]\, e^{i\frac{2\pi}{N}kn}$ $\longleftarrow$ Frequency domain

Notes:

- Properties of FT's generally apply to DFT's (e.g., convolution theorem).
- Brute force DFT computation is O(n^2).
- The Fast Fourier Transform (FFT) algorithm computes the DFT in O($n\log n$).

Discrete convolution in 2D

Similarly, discrete convolution in 2D becomes:

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

where $\tilde{h}[n,m] = h[-n,-m]$.

Further, the 2D DFT and inverse DFT are, for an N x M image:

$$F[k,l] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m]\, e^{-i2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)}$$

$$f[n,m] = \frac{1}{NM}\sum_{k=0}^{N-1}\sum_{l=0}^{M-1} F[k,l]\, e^{i2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)}$$

Convolution representation

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

128	54	9	78	100
145	98	240	233	86
89	177	246	228	127
67	90	255	237	95
106	111	128	167	20
221	154	97	123	0

X 0.1	X 0.1	X 0.1
X 0.1	X 0.2	X 0.1
X 0.1	X 0.1	X 0.1

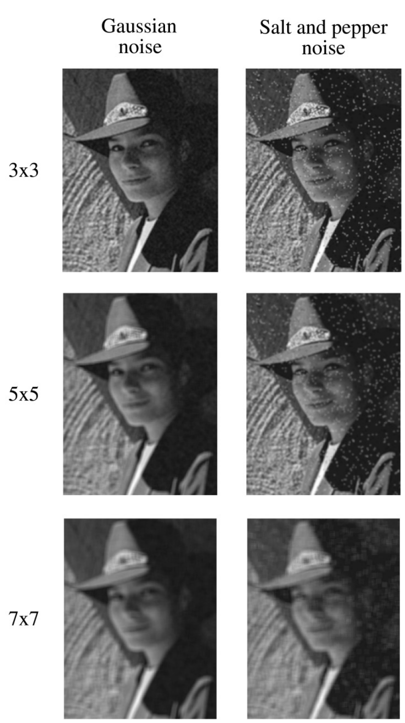
Note: *This is not matrix multiplication*!

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:

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

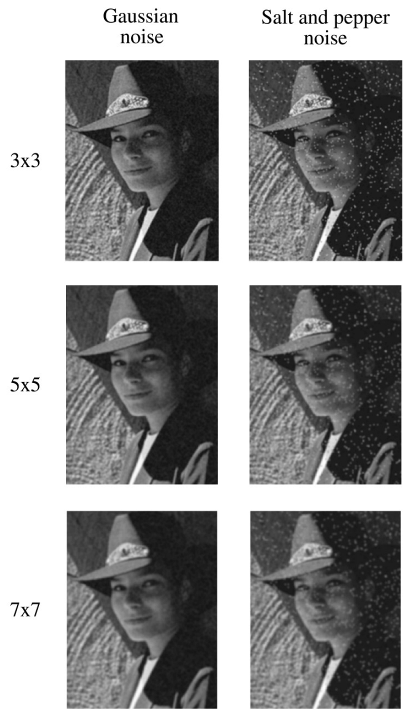
This does a decent job of blurring noise while preserving features of the image.

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 mxm 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