3. Image Processing (continued).

Last time, we started the discussion of basic notions and notation in image processing.
We finished with an example of image filtering which showed that 2-D convolution is very similar to 1-D convolution.

3.2. Image Filtering.

Example 1.

$$
\begin{array}{c|cccccc}
m \backslash n & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 0 & 0.2 & 0 & 0.2 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 1 & 1 & 1 \\
3 & 0 & 0 & 1 & 0.9 & 1 & 1 \\
4 & 0 & 0 & 1 & 1.1 & 1 & 1 \\
5 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
\;**\;
\begin{array}{c|ccc}
m \backslash n & -1 & 0 & 1 \\
\hline
-1 & 0 & 1 & 0 \\
0 & 1 & 4 & 1 \\
1 & 0 & 1 & 0
\end{array}
\cdot \frac{1}{8}
=
\begin{bmatrix}
2 & 8 & 4 & 8 & 2 & 0 \\
0 & 1 & 10 & 11 & 10 & 10 \\
0 & 10 & 6 & 69 & 70 & 60 \\
0 & 10 & 69 & 77 & 79 & 70 \\
0 & 10 & 71 & 85 & 81 & 70 \\
0 & 10 & 60 & 71 & 70 & 60
\end{bmatrix}
$$

$f(m,n)$ $\qquad$ $h(m,n)$ $\qquad$ $g(m,n) = f**h(m,n)$.

Blurring

border effects

In general,

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

In this example,

$$g(m,n) = \frac{1}{2} f(m,n) + \frac{1}{8} f(m-1,n) + \frac{1}{8} f(m+1,n) + \frac{1}{8} f(m,n-1) + \frac{1}{8} f(m,n+1)$$

Various $\boxed{\text{remarks}}$ on the filtering example:

1. To reduce the border effects, we can symmetrically extend the image, instead of padding with zeros.

2. This is an averaging filter - since $g(m,n)$ is a weighted average of values of f in a neighborhood of (m,n).

3. Just like a 1-D averaging filter (e.g., $\frac{1}{2}\;\frac{1}{2}$), this is a low-pass filter.

Frequency domain:

$$F(\omega_1,\omega_2)=\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} f(m,n)\,e^{-j\omega_1 m-j\omega_2 n}$$ - discrete-space Fourier transform

Inverse transform:

$$f(m,n)=\left(\frac{1}{2\pi}\right)^2\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}F(\omega_1,\omega_2)\,e^{j\omega_1 m+j\omega_2 n}\,d\omega_1\,d\omega_2$$

Convolution theorem:

$$g(m,n)=h**f(m,n) \Leftrightarrow G(\omega_1,\omega_2)=H(\omega_1,\omega_2)F(\omega_1,\omega_2).$$

(Similarly, in continuous-space,

$$F(\omega_x,\omega_y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)\,e^{-j\omega_x x-j\omega_y y}\,dx\,dy$$ - continuous-space Fourier transform

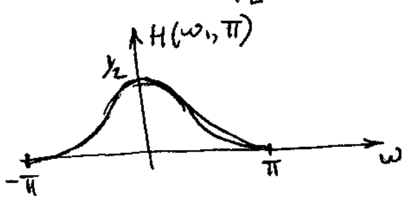
$$f(x,y)=\left(\frac{1}{2\pi}\right)^2\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}F(\omega_x,\omega_y)\,e^{j\omega_x x+j\omega_y y}\,d\omega_x\,d\omega_y,$$

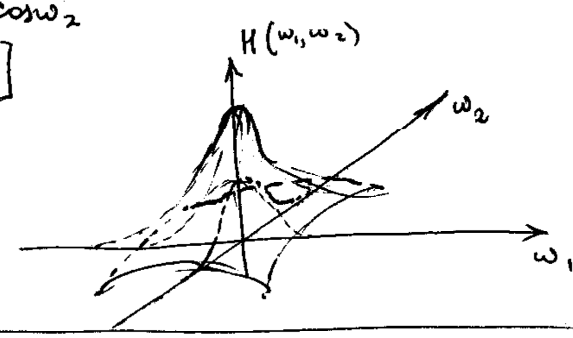
and $g(x,y)=h**f(x,y) \Leftrightarrow G(\omega_x,\omega_y)=H(\omega_x,\omega_y)F(\omega_x,\omega_y)$.)

For our filter,

$$H(\omega_1,\omega_2)=\frac{1}{2}+\underbrace{\frac{1}{8}e^{-j\omega_1}+\frac{1}{8}e^{j\omega_1}}_{\frac{1}{4}\cos\omega_1}+\underbrace{\frac{1}{8}e^{-j\omega_2}+\frac{1}{8}e^{j\omega_2}}_{\frac{1}{4}\cos\omega_2}$$

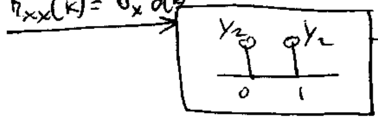
$$=\frac{1}{4}\left[(1+\cos\omega_1)+(1+\cos\omega_2)\right]$$





4. Note that a 1-D averaging filter, e.g., $\underset{}{\overset{1/2\ 1/2}{\text{⫯ ⫯}}}$, can be used to reduce noise!

$X(n)$ = white noise, $r_{xx}(k) = \sigma_x^2 \delta(k)$



$Y(n)$, with $r_{yy}(k) = h * h_- * r_{xx}(k)$
$= \sigma_x^2 \, h * h_-(k)$

$$\sigma_y^2 = r_{yy}(0) = \sigma_x^2 \sum_{m=-\infty}^{\infty} (h(m))^2 = \sigma_x^2\left(\frac{1}{4}+\frac{1}{4}\right) = \frac{\sigma_x^2}{2}$$

I.e., noise variance is reduced while the underlying signal is (hopefully!) not distorted too much. Same thing in 2-D (Lab 10).

(It is often easier to manipulate and visualize signals than 2-D images. One can often resort to 1-D analysis in order to gain insight into image processing algorithms.)

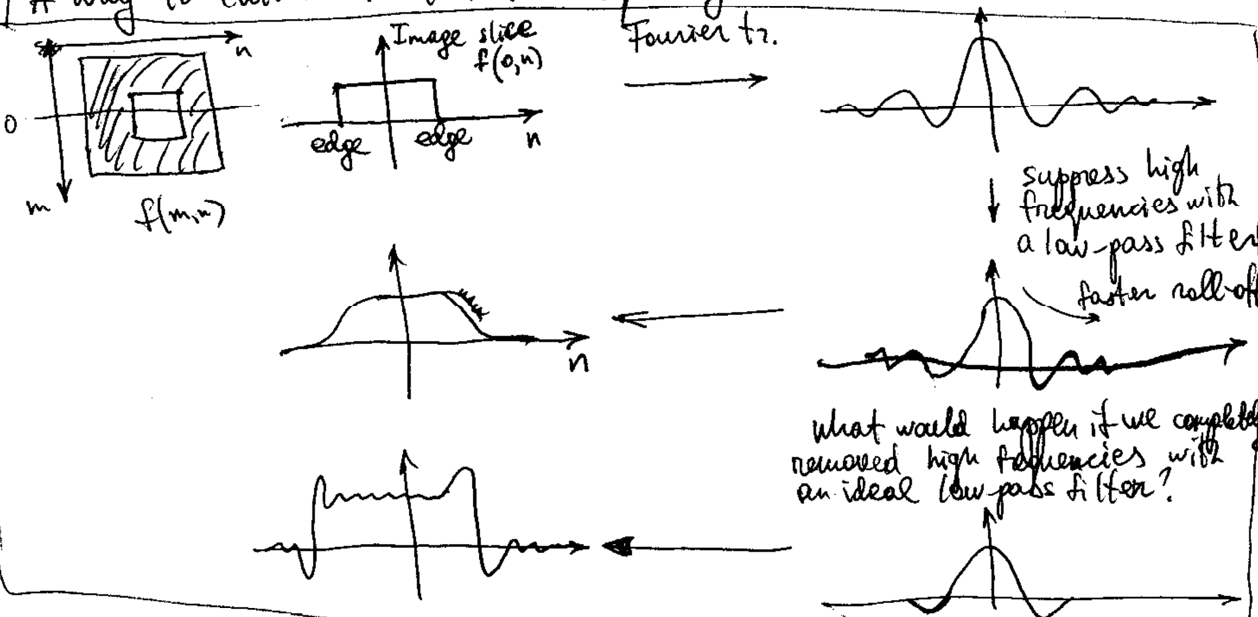
5. Unfortunately, the useful signal/image <u>is</u> distorted, in a way which is, for images, visually annoying:

an averaging filter introduces blurring – i.e., edges between objects are smeared.

A way to think of this in time domain:

to get an output pixel in the vicinity of an edge, we are averaging input pixels from both sides of the edge.

A way to think of this in frequency domain:



[DEMO]

remove noise ⇒ remove high frequencies preserve sharp edges ⇒ preserve high frequencies	an LTI or linear space-invariant filter cannot ~~~~ achieve both.

⇒ We need to look for nonlinear filters.

Example 2 (Median Filtering)

The median of $\{x_1, \ldots, x_{2i+1}\}$ is such x_p that i x's are $> x_p$ and i x's are $< x_p$.

E.g., median$\{1,2,8,3,0\} = 2$.

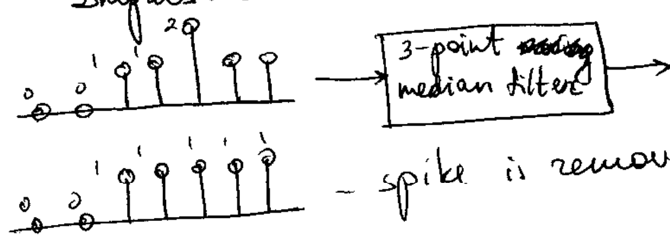
In a class of 69 students, "median score is 55" means that 34 people scored above 55 and 34 people scored below 55.

$f(m,n) \to$ Median filter with window $\to g(m,n) = \text{median}\{f(m,n), f(m-1,n), f(m+1,n), f(m,n+1), f(m,n-1)\}$

For the image of Example 1, the result of median filtering is:

0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	1	1	1
0	0	1	1	1	1
0	0	1	1	1	1
0	0	1	1	1	1

- No blurring for small window sizes
- "Impulsive" noise is removed:



– spike is removed

- But does not do well with Gaussian noise:
  - small window ⇒ poor noise removal
  - large window ⇒ remove small objects; blur.

[CAR DEMO]