Region Processing II

Prof. Eric Miller
elmiller@ece.tufts.edu
The 2D Case

- Much like before but with two variables
- We stick with odd-sized filters using above index system
- We stick with images with above index system
- Only concerned with $y[m,n]$ for $m=1,2,...M$ and $n=1,2,...,N$

$$y[m,n] = \sum_{k=-P}^{P} \sum_{l=-Q}^{Q} h[k,l] x[m-k, n-l]$$
Example

- Output image will be of size 6 by 6
- Basically we replace each pixel in $x$ by an weighted average of that pixel and its neighbors
- When $h$ spills over the edge, we assume $x$ is zero
Example (cont)

\[ y[m, n] = \sum_{k=-P}^{P} \sum_{l=-Q}^{Q} h[k, l]x[m - k, n - l] \]

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 2 & 3 & 4 & 5 & 6 \\
2 & 7 & a & b & c & 10 & 11 \\
3 & 9 & d & e & f & 8 & 3 \\
4 & 0 & g & h & j & 5 & 7 \\
5 & 10 & 12 & 11 & 1 & 0 & 0 \\
6 & 1 & 2 & 8 & 4 & 8 & 8 \\
\end{array}
\]

\[
y[3, 3] = 8a + 9b + 10c + 6d + 5e + 8f + 4g + 8h + 0j
\]

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 2 & 3 & 4 & 5 & 6 \\
2 & 7 & 8 & 9 & a & b & c \\
3 & 9 & 6 & 5 & 8 & d & e & f \\
4 & 0 & 4 & 8 & 0 & g & h & j \\
5 & 10 & 12 & 11 & 1 & 0 & 0 \\
6 & 1 & 2 & 8 & 4 & 8 & 8 \\
\end{array}
\]

\[
y[3, 6] = 11a + 10b + 0c + 3d + 9e + 0f + 5g + 7h + 0j
\]
Implementation

• Not hard to implement this, but a bit tedious. We will use the Matlab function \texttt{filter2}
  \[ y = \text{filter2}(h,x,\text{shape}) \]
• \texttt{h} and \texttt{x} are the filter and the image as we would expect
• \texttt{shape} = `same` or `valid` or `full`
  – \texttt{same} = \texttt{y} same size as \texttt{x}
  – \texttt{valid} = all pixels not impacted by edges (\texttt{y} smaller than \texttt{x})
  – \texttt{full} = any overlap of filter with image (\texttt{y} larger than \texttt{x})
What can we do with this

• People have built filters to “do” common things
  – Smooth images to removing detail or getting rid of noise
  – Enhancing features especially horizontal and vertical edges

• We’ll stick with these two
Image smoothing

• Many types of smoothing filters especially for 1D problems
• For images: consider a couple of common choices
  – Boxes: arithmetic averager
  – Gaussian: bell curve in 2D
Box filters

$h[m,n] = \frac{1}{N}$ where there are $N$ pixels in $h$

What might each of these do?

Matlab construction:

```matlab
h = fspecial('average',[M,N]);
```
creates an $M \times N$ averaging filter
Gaussian in 1D

\[ h[n] = c e^{-n^2/(2\sigma^2)} \]

\( n = -50:50; \)
\( \text{sigma} = 3; \)
\( h = \exp(-n.^2/(2*\text{sigma}*\text{sigma})); \)
\( C = 1/\text{sum}(h); \ h = c*h; \)

- C chosen to make \( h \) have area = 1

- Note role of \( \sigma \) in defining the width of the filter
- Not a uniform average. Closer points get more weight
- Many filters have similar characteristics. This one is special for reasons a bit beyond this class.
Gaussian in 2D

Simplest case

$$h[n, m] = c e^{-\frac{1}{2\sigma^2}(m^2+n^2)}$$

A bit more general

$$h[n, m] = c e^{-\frac{1}{2}(\frac{m^2}{\sigma_1^2} + \frac{n^2}{\sigma_2^2})}$$

The most general

$$h[n, m] = c \exp \left\{ -\frac{1}{2(1-\rho^2)} \left[ \frac{m^2}{\sigma_1^2} - \frac{2\rho mn}{\sigma_1\sigma_2} + \frac{n^2}{\sigma_2^2} \right] \right\}$$

In all cases

$$\sigma > 0$$

$$\sigma_1 > 0$$

$$\sigma_2 > 0$$

$$-1 < \rho < 1$$
Look at simple Gaussian

\[ h[n, m] = c e^{-\frac{1}{2\sigma^2}(m^2+n^2)} \]

- Circularly symmetric also called “isotropic” in EE-speak means all directions treated the same
- Note the role of \( \sigma \) in defining the width of the filter; how much averaging takes place
Matlab Topics

- Use of `meshgrid` for generating sets of 2D points
- Use of `fspecial` for building filters
- Use of `mesh` for displaying surfaces and `imagesc` as an alternate for `imshow` for displaying pictures