Filters

CMPUT 206
Instructor: Nilanjan Ray

Source: www.imagingbook.com
Filter: a basic image processing operation

$R_{u,v}$ is called a neighborhood of a pixel location $(u, v)$. It is a set of pixel values on the source image. The pixel value at $(u, v)$ on the filtered image is computed from this set of pixel values from the source image.
A first example of image filter

- Replace every pixel by the average of its 8-neighboring pixel values:

\[
I'(u, v) \leftarrow \frac{1}{9} \cdot \left[ I(u-1, v-1) + I(u, v-1) + I(u+1, v-1) + \\
I(u-1, v) + I(u, v) + I(u+1, v) + \\
I(u-1, v+1) + I(u, v+1) + I(u+1, v+1) \right]
\]

Or, compactly expressed with summation signs as:

\[
I'(u, v) \leftarrow \frac{1}{9} \cdot \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u + i, v + j)
\]

This is an example of a linear filter
Linear filters

• The averaging filter can be expressed by a filter matrix:

\[ H(i, j) = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \]

• Here’s the filter operation we saw before, now expressed with filter matrix:

\[ I'(u, v) \leftarrow \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u + i, v + j) \cdot H(i, j) \]

Convention: \( H(0, 0) \) coincides with current image position \((u, v)\)
Do until all pixel locations are visited:

Step 1: At current location point-wise multiply and add image sub-window and filter matrix

Step 2: Shift current location, and go to step 1
Linear filters: image borders

Why does image borders pose problems in filtering? How about point operations?

Let’s discuss some strategies to handle image borders in linear filtering.
Methods for extending the image to facilitate filtering along the borders. The assumption is that the (nonexisting) pixels outside the original image are either set to some constant value (a), take on the value of the closest border pixel (b), are mirrored at the image boundaries (c), or repeat periodically along the coordinate axes.
Linear filters: filter size

- Typically a filter matrix is of size \((2K+1)\)-by-\((2K+1)\), i.e., odd matrix size—why?
- Computational complexity increases with filter size
- For a really large filter matrix, a trick called frequency domain processing is employed—this is computationally cheaper

In Photoshop you can do a custom linear filtering
Types of linear filters

• Essentially of two types
  – Smoothing filters, typically used in
    • Noise reduction
    • Segmentation
  – Difference filters, typically used in
    • Edge detection
    • deblurring
Types of linear filters

- Often, visualization of a filter matrix reveals the nature of a linear filter

(a) Box filter, (b) Gaussian filter, (c) Laplace filter: Can you guess their types?
Some properties of linear filters

• Commutative

\[ I \ast H = H \ast I \]

"\ast" expresses linear filtering

• Linear

\[ (s \cdot I) \ast H = I \ast (s \cdot H) = s \cdot (I \ast H) \]

\[ (I_1 + I_2) \ast H = (I_1 \ast H) + (I_2 \ast H) \]

• Associative

\[ A \ast (B \ast C) = (A \ast B) \ast C \]
Filters in ImageJ

A 3x3 linear filter

```java
1 import ij.plugin.filter.Convolver;
2 ...
3 public void run(ImageProcessor I) {
4     float[] H = { // filter array is one-dimensional!
5         0.075f, 0.125f, 0.075f,
6         0.125f, 0.200f, 0.125f,
7         0.075f, 0.125f, 0.075f }
8     Convolver cv = new Convolver();
9     cv.setNormalize(false); // do not use filter normalization
10    cv.convolve(I, H, 3, 3); // apply the filter H to I
11 }
```

A Gaussian Filter: Circular filter shape

```java
1 import ij.plugin.filter.GaussianBlur;
2 ...
3 public void run(ImageProcessor ip) {
4     GaussianBlur gb = new GaussianBlur();
5     double radius = 2.5;
6     gb.blur(ip, radius);
7 }
```
Why the name filter?

• Can anyone guess?
Nonlinear Filters

• One important use of linear filter is noise reduction; however, linear (smoothing) filters have a significant drawback – along with the noise, it also smoothes out important structures, edges, etc. in the image:

• We now want to examine if some nonlinear filters can offer a better solution here
What is a nonlinear filter?

• It’s a filter—resulting pixel value is a function of the pixel values in a corresponding neighborhood, i.e., sub-window around the pixel location

• It does not maintain linearity property
Median filter

- Median filter replaces every image pixel by the median of the pixels in the corresponding image sub-window $R$:

\[ I'(u, v) \leftarrow \text{median} \{ I(u+i, v+j) \mid (i, j) \in R \} \]

- The median of $2K+1$ pixel values $p_i$ is defined as:

\[ \text{median} (p_0, p_1, \ldots, p_K, \ldots, p_{2K}) \triangleq p_K \]

where $p_i$ are sorted, i.e., $p_0 \leq p_1 \leq \ldots \leq p_{2K}$
Median filter: pictorially

Computation of a 3x3 median filter. The 9 pixel values extracted from the 3x3 image sub-window are arranged as a sorted vector. The resulting center value is taken.
Weighted median filter

Each pixel value is inserted into the extended pixel vector multiple times, as specified by the weight matrix.

Mimics voting
Some illustrations

(a) Lena image with ‘salt and pepper’ noise; (b) linear 3x3 box filter; (c) median filter result