Outlines

- What are Filters
- Linear Filters
- Convolution operation
- Properties of Linear Filters
- Application of filters
- Nonlinear Filter
- Normalized Correlation and finding patterns in images
- Sources:
  - Burger and Burge “Digital Image Processing” Chapter 6
  - Forsyth and Ponce “Computer Vision a Modern approach”
What is a Filter

- Point operations are limited (why)
- They cannot accomplish tasks like sharpening or smoothing

Smoothing an image by averaging

- Replace each pixel by the average of its neighboring pixels
- Assume a 3x3 neighborhood:

\[
I'(u, v) \leftarrow \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9}
\]
In general a filter applies a function over the values of a small neighborhood of pixels to compute the result.

The size of the filter = the size of the neighborhood: 3x3, 5x5, 7x7, ..., 21x21...

The shape of the filter region is not necessarily square, can be a rectangle, a circle...

Filters can be linear or nonlinear.

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

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

\[ I'(u, v) \leftarrow \sum_{(i,j) \in R_H} I(u + i, v + j) \cdot H(i, j) \]

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

Averaging filter

\[ I'(u, v) \leftarrow \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9} \]

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

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

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

- The filter matrix $H$ moves over the original image $I$ to compute the convolution operation.
- We need an intermediate image storage!
- We need 4 for loops!
- In general a scale is needed to obtain a normalized filter.
- Integer coefficient is preferred to avoid floating point operations.

Another smoothing filter

```java
public void run(ImageProcessor orig) {
    int w = orig.getWidth();
    int h = orig.getHeight();
    // 3 x 3 filter matrix:
    double[][] filter = {
        {0.075, 0.125, 0.075},
        {0.125, 0.2, 0.125},
        {0.075, 0.125, 0.075}
    };
    ImageProcessor copy = orig.duplicate();
    for (int v = 1; v <= h-2; v++) {
        for (int u = 1; u <= w-2; u++) {
            // compute filter result for position (u,v)
            double sum = 0;
            for (int j = -1; j <= 1; j++) {
                for (int i = -1; i <= 1; i++) {
                    int p = copy.getPixel(u+i, v+j);
                    // get the corresponding filter coefficient:
                    double c = filter[i+j][i+1];
                    sum += c * p;
                }
            }
            int q = (int) Math.round(sum);
            orig.putPixel(u, v, q);
        }
    }
}
```
Integer coefficient

\[
H(i, j) = \begin{bmatrix}
0.075 & 0.125 & 0.075 \\
0.125 & 0.200 & 0.125 \\
0.075 & 0.125 & 0.075 \\
\end{bmatrix} = \frac{1}{40} \begin{bmatrix}
3 & 5 & 3 \\
5 & 8 & 5 \\
3 & 5 & 3 \\
\end{bmatrix}
\]

- Ex: linear filter in Adobe photoshop

\[
I'(u, v) \leftarrow \text{Offset} + \frac{1}{\text{Scale}} \sum_{j=-2}^{1} \sum_{i=-2}^{2} I(u+i, v+j) \cdot H(i, j)
\]
For a filter of size \((2K+1) \times (2L+1)\), if the image size is \(M \times N\), the filter is computed over the range:

\[
K \leq u' \leq (M - K - 1) \quad \text{and} \quad L \leq v' \leq (N - L - 1)
\]
Smoothing by Averaging vs. Gaussian

Flat kernel: all weights equal $1/N$

- Smoothing with an average actually doesn’t compare at all well with a defocussed lens
  - Most obvious difference is that a single point of light viewed in a defocussed lens looks like a fuzzy blob; but the averaging process would give a little square.

- A Gaussian gives a good model of a fuzzy blob
An Isotropic Gaussian

- The picture shows a smoothing kernel proportional to

\[ \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]

(which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing with a Gaussian
Gaussian smoothing

- Advantages of Gaussian filtering
  - rotationally symmetric (for large filters)
  - filter weights decrease monotonically from central peak, giving most weight to central pixels
  - Simple and intuitive relationship between size of $\sigma$ and the smoothing.
  - The Gaussian is separable:

Advantage of separability

- First convolve the image with a one dimensional horizontal filter
- Then convolve the result of the first convolution with a one dimensional vertical filter
- For a $k \times k$ Gaussian filter, 2D convolution requires $k^2$ operations per pixel
- But using the separable filters, we reduce this to $2k$ operations per pixel.
Separability

Advantages of Gaussians

- Convolution of a Gaussian with itself is another Gaussian
  - so we can first smooth an image with a small Gaussian
  - then, we convolve that smoothed image with another small Gaussian and the result is equivalent to smoothing the original image with a larger Gaussian.
  - If we smooth an image with a Gaussian having sd $\sigma$ twice, then we get the same result as smoothing the image with a Gaussian having standard deviation $(2\sigma)^{1/2}$
Mathematical Properties of Linear Convolution

- For any 2D discrete signal, convolution is defined as:

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

\[ I' = I \ast H \]

**Properties**

- **Commutativity**

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

- **Linearity**

  \[ (s \cdot I) \ast H = I \ast (s \cdot H) = s \cdot (I \ast H) \]
  \[ (I_1 + I_2) \ast H = (I_1 \ast H) + (I_2 \ast H) \]

  (notice) \( (b + I) \ast H \neq b + (I \ast H) \)

- **Associativity**

  \[ A \ast (B \ast C) = (A \ast B) \ast C \]
Properties

- Separability

\[ H = H_1 \ast H_2 \ast \ldots \ast H_n \]
\[ I \ast H = I \ast (H_3 \ast H_2 \ast \ldots \ast H_n) \]
\[ = \ldots ((I \ast H_1) \ast H_2) \ast \ldots \ast H_n) \]

Nonlinear Filters

- Linear filters have a disadvantage when used for smoothing or removing noise: all image structures are blurred, the quality of the image is reduced.

- Examples of nonlinear filters:
  - Minimum and Maximum filters

\[ I'(u, v) \leftarrow \min \{I(u+i, v+j) \mid (i, j) \in R\} \]
\[ I'(u, v) \leftarrow \max \{I(u+i, v+j) \mid (i, j) \in R\} \]
Median Filter

- Much better in removing noise and keeping the structures

\[ I'(u, v) \leftarrow \text{median} \{ I(u+i, v+j) \mid (i, j) \in R \} \]
Weighted median filter
Linear Filters: convolution

Convolution as a Dot Product

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector.
- Convoluting an image with a filter is equivalent to taking the dot product of the filter with each image window.
- Largest value when the vector representing the image is parallel to the vector representing the filter
- Filter responds most strongly at image windows that looks like the filter.
- Filter responds stronger to brighter regions! (drawback)

Insight:
- filters look like the effects they are intended to find
- filters find effects they look like

**Ex:** Derivative of Gaussian used in edge detection looks like edges

---

**Normalized Correlation**

- Convolution with a filter can be used to find templates in the image.
- Normalized correlation output is filter output, divided by root sum of squares of values over which filter lies.
- Consider template (filter) $M$ and image window $N$:

$$C = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} M(i,j)N(i,j)}{[\sum_{i=1}^{n} \sum_{j=1}^{n} M(i,j)^2 \sum_{i=1}^{n} \sum_{j=1}^{n} N(i,j)^2]^{1/2}}$$

**Diagram:**

- Original image
- Filtered image (Normalized Correlation Result)

**Table:**

<table>
<thead>
<tr>
<th>Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtered image (Normalized Correlation Result)</td>
</tr>
</tbody>
</table>
Normalized Correlation

$$C = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} M(i,j)N(i,j)}{\left[\sum_{i=1}^{n} \sum_{j=1}^{m} M(i,j)^2 \sum_{i=1}^{n} \sum_{j=1}^{m} N(i,j)^2\right]^{1/2}}$$

- This correlation measure takes on values in the range [0,1]
- it is 1 if and only if $N = cM$ for some constant $c$
- so $N$ can be uniformly brighter or darker than the template, $M$, and the correlation will still be high.
- The first term in the denominator, $\sum M^2$ depends only on the template, and can be ignored
- The second term in the denominator, $\sum N^2$ can be eliminated if we first normalize the grey levels of $N$ so that their total value is the same as that of $M$ - just scale each pixel in $N$ by $\frac{\sum M}{\sum N}$

Positive responses
Zero mean image, -1:1 scale
Zero mean image, -max:max scale
Positive responses

Zero mean image, -1:1 scale

Zero mean image, -max:max scale