Visual Computing: Convolution and Filtering

Prof. Marc Pollefeys
Last time: Segmentation
What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.

![Local image data](image1.png)  ![Modified image data](image2.png)

Some function
Linear Shift-Invariant Filtering

- About modifying pixels based on **neighborhood**. Local methods simplest.
- Linear means **linear combination** of neighbors. Linear methods simplest.
- **Shift-invariant** means doing the same for each pixel. Same for all is simplest.
- Useful to:
  - Low-level image processing operations
  - Smoothing and noise reduction.
  - Sharpen.
  - Detect or enhance features.
Linear Filtering

• $L$ is linear operation if

$$L[\alpha l_1 + \beta l_2] = \alpha L[l_1] + \beta L[l_2]$$
Linear Operations: Weighted Sum

• Output $I'$ of linear image operation is a weighted sum of each pixel in the input $I$

$$I'_j = \sum_{i=1}^{N} \alpha_{ij} I_i, \ j = 1...N$$

(note: $N=wh$)
Linear Filtering

• Linear operations can be written:

\[ I'(x, y) = \sum_{(i, j) \in N(x, y)} K(x, y; i, j)I(i, j) \]

• \( I \) is the input image; \( I' \) is the output of the operation.

• \( K \) is the kernel of the operation. \( N(m,n) \) is a neighbourhood of \( (m,n) \).
Linear Filtering

- Linear operations can be written:

$$I'(x, y) = \sum_{(i, j) \in N(x, y)} K(x, y; i, j) I(i, j)$$

- $I$ is the input image; $I'$ is the output of the operation.
- $K$ is the kernel of the operation.
- $N(m, n)$ is a neighbourhood of $(m, n)$.

Operations are “shift-invariant” if $K$ does NOT depend on $(x, y)$: using same weights everywhere!
Correlation
(e.g. template matching)

\[ o (i,j) = c_{11} I(i-1,j-1) + c_{12} I(i-1,j) + c_{13} I(i-1,j+1) + c_{21} I(i,j-1) + c_{22} I(i,j) + c_{23} I(i,j+1) + c_{31} I(i+1,j-1) + c_{32} I(i+1,j) + c_{33} I(i+1,j+1) \]
Correlation

• Linear operation of correlation:

\[
I' = K \circ I
\]

\[
I'(x, y) = \sum_{(i, j) \in N(x, y)} K(i, j)I(x + i, y + j)
\]

• Represent the linear weights as an image, \( K \)
**Convolution**

(e.g. point spread function)

\[ I'(x,y) = K(1,1)I(x-1,y-1) + K(0,1)I(x,y-1) + K(-1,1)I(x+1,y-1) + K(1,0)I(x-1,y) + K(0,0)I(x,y) + K(-1,0)I(x+1,y) + K(1,-1)I(x-1,y+1) + K(0,-1)I(x,y+1) + K(-1,-1)I(x+1,y+1) \]
Convolution

• Linear operation of convolution:

\[ I' = K \ast I \]

\[ I'(x, y) = \sum_{(i, j) \in N(x, y)} K(i, j)I(x - i, y - j) \]

• Represent the linear weights as an image, \( K \)
• Same as correlation, but with kernel reversed
Correlation

\[ I'(x, y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i, j)I(x + i, y + j) \]

Convolution

\[ I'(x, y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i, j)I(x - i, y - j) \]

\[ = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(-i, -j)I(x + i, y + j) \]

So if \( K(i,j) = K(-i, -j) \), then Correlation == Convolution
Linear Filtering (warm-up)

Original

Slide credit: D.A. Forsyth
Linear Filtering
(warm-up)

Original

0 0 0
0 1 0
0 0 0

Filtered
(no change)

Slide credit: D.A. Forsyth
Linear Filtering

Original

(use convolution)

Slide credit: D.A. Forsyth
Linear Filtering

Original

(\text{use convolution})

Shifted left
By 1 pixel

Slide credit: D.A. Forsyth
Linear Filtering

Original

?
Linear Filtering

Original
Linear Filtering

Original

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

\[\frac{1}{9}\]

?
Linear Filtering

Original

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Blur (with a box filter)

Slide credit: D.A. Forsyth
Linear Filtering

Original

(Note that filter sums to 1)
Linear Filtering

Original

Sharpening filter
- Accentuates differences with local average

Slide credit: D.A. Forsyth
Sharpening

before

after
Correlation
(e.g. Template-matching)

Convolution
(e.g. point spread function)

\[ I' = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i, j) I(x + i, y + j) \]

(matlab default)

\[ I' = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i, j) I(x - i, y - j) \]

\[ = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(-i, -j) I(x + i, y + j) \]
Example

\[
K = \text{ones}(9, 9);
\]

\[
I_2 = \text{conv2}(I, K);
\]
Example

\[ K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \]
Yucky details

• What about near the edge?
  – the filter window falls off the edge of the image
  – need to extrapolate
  – methods:
    • clip filter (black)
    • wrap around
    • copy edge
    • reflect across edge
    • vary filter near edge
Separable Kernels

• Separable filters can be written

\[ K(m,n) = f(m)g(n) \]

• For a rectangular neighbourhood with size 
\((2M+1) \times (2N+1)\),

\[ I'(m,n) = f \ast (g \ast I(N(m,n))) \]

\[ I''(m,n) = \sum_{j=-N}^{N} g(j)I(m,n-j) \]

\[ I'(m,n) = \sum_{i=-M}^{M} f(i)I''(m-i,n) \]
Separable Kernels

• Separable filters can be written

\[ K(m,n) = f(m)g(n) \]

• For a rectangular neighbourhood with size \((2M+1)\times(2N+1),\)

\[
I'(m,n) = f \ast (g \ast I(N(m,n)))
\]

\[
I''(m,n) = \sum_{j=-N}^{N} g(j)I(m,n-j)
\]

\[
I'(m,n) = \sum_{i=-M}^{M} f(i)I''(m-i,n)
\]

computational advantage?
Smoothing Kernels
(Low-pass filters)

Mean filter:
\[
\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

Weighted smoothing filters:
\[
\frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}
\]
Gaussian Kernel

- Idea: Weight contributions of neighboring pixels by nearness

\[
G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}
\]

- Constant factor at front makes volume sum to 1

5 x 5, \( \sigma = 1 \)

Slide credit: Christopher Rasmussen
Smoothing with a Gaussian

Slide credit: D.A. Forsyth
Smoothing with a box filter
Gaussian Smoothing Kernels

\[ g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{(x^2 + y^2)}{2\sigma^2}\right] \]

\[ = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{x^2}{2\sigma^2}\right] \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{y^2}{2\sigma^2}\right] \]

\[ = g(x) g(y) \]

Separable!
Gaussian Smoothing Kernels

- Amount of smoothing depends on $\sigma$ and window size.
- Width > $3\sigma$

- $7 \times 7; \sigma = 1$
- $7 \times 7; \sigma = 9$
- $19 \times 19; \sigma = 1$
- $19 \times 19; \sigma = 9$
Scale Space

- Convolution of a Gaussian with standard deviation $\sigma$ with itself is a Gaussian standard deviation $\sigma\sqrt{2}$.

- Repeated convolution by a Gaussian filter produces the scale space of an image.
Scale Space Example

11x11; \( \sigma = 3 \).
Gaussian Smoothing Kernel Top-5

1. Rotationally symmetric
2. Has a single lobe
   - Neighbor’s influence decreases monotonically
3. Still one lobe in frequency domain
   - No corruption from high frequencies
4. Simple relationship to $\sigma$
5. Easy to implement efficiently
Differential Filters

Prewitt operator:

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]

Sobel operator:

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]
High-pass filters

Laplacian operator:
\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

High-pass filter:
\[
\begin{bmatrix}
-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1
\end{bmatrix}
\]
High-pass filters

Laplacian

High pass
Differentiation and convolution

- Recall, for 2D function, $f(x,y)$:

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \left( \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon} \right)$$

- This is linear and shift invariant, so must be the result of a convolution.

- We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

(which is obviously a convolution)

Slide credit: D.A. Forsyth
Vertical gradients from finite differences
Filters are templates

• Filter at some point can be seen as taking a dot-product between the image and some vector

• Filtering the image is a set of dot products

- filters look like the effects they are intended to find
- filters find effects they look like
Image Sharpening

- Also known as Enhancement
- Increases the high frequency components to enhance edges.
- $I' = I + \alpha |k*I|$, where $k$ is a high-pass filter kernel and $\alpha$ is a scalar in [0,1].
Sharpening Example

original

$\alpha = 0.5$
Integral images

- Integral images (also known as summed-area tables) allow for efficient computation of the convolution with a constant rectangle.

\[
\mathcal{I}(x, y) = \int_{0}^{x} \int_{0}^{y} I(x', y') \, dx' \, dy' 
\]

Figures from Viola and Jones 2001
Viola-Jones cascade face detection

- Very efficient face detection using integral images

Figures from Viola and Jones 2001
• Also possible along diagonal
Thursday:

Image Features