Neighborhood Operations

SS 2008 – Image Processing

Multimedia Computing, Universität Augsburg
Rainer.Lienhart@informatik.uni-augsburg.de
www.multimedia-computing.{de,org}
Overview

• Combining neighboring pixels to form a new image
  → neighboring operations are called ‘filters’
  – Detect simple structures such as edges, corners, lines, constant areas
  – Smoothing, Sharpening, Warping
  – Texture analysis
  – Motion determination
  – Etc.

• Two (out of many) principle ways:
  – Linear shift-invariant (LSI) filters
  – Rank value filters
What is Image Filtering?

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

<table>
<thead>
<tr>
<th>10</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Some function

7
Linear Filtering

• Linear case is simplest and most useful
  – Replace each pixel with a linear combination of its neighbors.

• The prescription for the linear combination is called the convolution kernel.

\[
\begin{bmatrix}
10 & 5 & 3 \\
4 & 5 & 1 \\
1 & 1 & 7 \\
\end{bmatrix} \times
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0.5 & 0 \\
0 & 1.0 & 0.5 \\
\end{bmatrix} =
\begin{bmatrix}
\end{bmatrix}
\]

kernel

\[
7
\]
Linear Filter = Convolution

- **Window/filter mask**: size/shape of local neighborhood
  Examples: rectangular, circular neighborhoods.
- **Pixel positions** must be specified relative to center of mask and image pixel
- Let \( r \) specify the rectangular region encompassing all pixels in the local neighborhood and \( g(x,y) \) the filter mask with the corresponding weighting factors

\[
I'[x, y] = I \otimes g \\
= \sum_{k=-r}^{r} \sum_{l=-r}^{r} I[x-k, y-l] g[k, l] \\
= \sum_{k,l} I[x-k, y-l] g[k, l] \\
= \sum_{k,l} I[x+k, y+l] g[-k,-l]
\]

**Mask size**: \((2r+1) \times (2r+1)\)
3x3 Convolution

\[ m \ast \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \]
Filtering Examples

original

coefficient

Pixel offset

1.0

shifted

© 2005-2008 Prof. Dr. Rainer Lienhart, Head of Multimedia Computing, Institut für Informatik, Universität Augsburg
Eichleitnerstr. 30, D-86135 Augsburg, Germany; email: Rainer.Lienhart@informatik.uni-augsburg.de
Filtering Examples

original

Blurred (filter applied in both dimensions).

© 2005-2008 Prof. Dr. Rainer Lienhart, Head of Multimedia Computing, Institut für Informatik, Universität Augsburg
Eichleitnerstr. 30, D-86135 Augsburg, Germany; email: Rainer.Lienhart@informatik.uni-augsburg.de
Point Spread Function

:= response of a filter to a point image (also called impulse response)

\[ I'[x, y] = \sum_{m=-r}^{r} \sum_{n=-r}^{r} I[x-n, y-m] \cdot g[n, m] \]

\[ = g(x, y) \]

where

\[ I[x, y] = \begin{cases} 
1 & x = 0, y = 0 \\
0 & \text{otherwise}
\end{cases} \]

If the response to a point image is known, then the response to any image can be computed. This is a direct consequence of the fact that a convolution is linear and shift-invariant.

*Transfer function* := DFT of point Spread Function

Remember: a convolution is only a multiplication in the DFT domain
Computation

For all pixels \( p \) in the image
- Center mirrored mask at \( p \)'s position \((x,y)\)
- Calculate weighted sum
- Write result back at \((x,y)\)
  (same or new image)

Issues:
- In-place operations need special care!
- Special handling for the border of the image
  (mirroring vs. extrapolation)

Cyclic convolution
Properties of LSI Ops

Commutativity:
\[ \mathbf{H} \mathbf{H}' = \mathbf{H}' \mathbf{H} \]
\[ \rightarrow \text{Easy to prove in FT domain} \]

Associativity:
\[ \mathbf{H}' \mathbf{H} = \mathbf{H} \]

Distributivity over Addition:
\[ \mathbf{H}' + \mathbf{H}'' = \mathbf{H} \]

\[ \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} \]

Separable operator $\mathcal{B}$ in 3D: $\mathcal{B} = \mathcal{B}_x \mathcal{B}_y \mathcal{B}_z$

25 muls, 24 adds

10 muls, 8 adds
Rank Value Filters

:= comparing and selection (non-linear filter)
   – Sort all pixel values within the mask (ascending)
   – Pick one of the values according to some rule and write it back as the new center pixel value

Examples: Median filter = pick medium value; min/max filter = pick min/max value
Figure 5.5: Gaussian pyramid: \(a\) schematic representation, the squares of the checkerboard corresponding to pixels; \(b\) example.
Figure 5.6: Construction of the Laplacian pyramid (right column) from the Gaussian pyramid (left column) by subtracting two consecutive planes of the Gaussian pyramid