Lecture Image Enhancement and Spatial Filtering

Harvey Rhody
Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology
rhody@cis.rit.edu

September 29, 2005

Abstract
Applications of point processing to image segmentation by global and regional segmentation are constructed and demonstrated. An adaptive threshold algorithm is presented. Illumination compensation is shown to improve global segmentation. Finally, morphological waterfall region segmentation is demonstrated.
Spatial Filtering

Uses of spatial filtering

• Image enhancement
• Feature detection.

Spatial filtering may be:

• Linear
• Nonlinear
• Spatially invariant
• Spatially varying
Noise Suppression by Image Averaging

In applications such as astronomy noisy images are unavoidable. Noise can be reduced by averaging several images of the same scene with independent noise. Here we simulate the process by adding copies of an image with independent noise with $\mu = 0$ and $\sigma = 64$ levels.
Effect of Averaging

Shown at the right is the histogram of the pixel noise after 8, 16, 64 and 128 images have been averaged. The standard deviation, which is proportional to the width of each curve, is reduced by $\sqrt{n}$. The reduction factors are 2.8, 4, 8, 11.3 for $n = 8, 16, 64, 128$, respectively.

Averaging is clearly a very effective tool for the reduction of noise provided that enough independent samples are available.
Region Averaging

Suppose that only a single noisy image is available. Can averaging still be employed for noise reduction?

**Strategy:**
Average the noise from pixels in a neighborhood.

This necessarily involves averaging of the image as well as the noise.
Linear Spatial Filtering

Region averaging is one form of spatial filtering. In linear spatial filtering, each output pixel is a weighted sum of pixels.

Shown at the right is a reproduction of G&W Figure 3.32, which illustrates the mechanics of spatial filtering.

In general, linear spatial filtering of an image $f$ by a filter with a weight mask $w$ of size $(2a + 1, 2b + 1)$ is

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)$$
Filters

A linear spatially invariant filter can be represented with a mask that is convolved with the image array. The weights are represented by the values \( w_i \).

\[
\begin{array}{ccc}
  w_1 & w_2 & w_3 \\
  w_4 & w_5 & w_6 \\
  w_7 & w_8 & w_9 \\
\end{array}
\]

If the gray levels of the pixels under the mask are denoted by \( z_1, z_2, \ldots, z_9 \) then the response of the linear mask is the sum

\[
R = w_1z_1 + w_2z_2 + \cdots + w_9z_9
\]

The result \( R \) is written to the output array at the position of the filter origin (usually the center of the filter).
Smoothing Filters

Smoothing filters are used for blurring and noise reduction.

Blurring is a common preprocessing step to remove small details when the objective is location of large objects.

High-frequency noise is reduced by the lowpass characteristic of smoothing filters.

Smoothing filters have all positive weights. The weights are typically chosen to sum to unity so that the average brightness values is maintained.
Smoothing Filters

Smoothing filters calculate a weighted average of the pixels under the mask. The low frequency response becomes more pronounced as the filter size is increased.

Masks are usually chosen to have odd dimensions to provide a center pixel location. The output is written to that pixel.

Larger filters do more smoothing but also produce more blurring. An example is shown below.

![Original Image](image1)
![Noisy Image](image2)
![3 × 3 smoothing](image3)
![5 × 5 smoothing](image4)
Smoothing Filters

Smoothing is linear and spatially invariant. It is equivalent to convolution of the image and the mask.

In the frequency domain this is equivalent to multiplication of the image transform with the mask transform.

The frequency response of smoothing filters of several sizes is shown below. The frequency range is in normalized units.

Note how the frequency response becomes narrower as the smoothing filter becomes larger.
Frequency Response of Smoothing Filters

M=3

M=5

M=7

M=9
Frequency Response of Smoothing Filters

The frequency response along a slice through the origin in the frequency plane is shown below for several values of $M$. 

![Frequency Response, M= 3](image1)

![Frequency Response, M= 5](image2)

![Frequency Response, M= 7](image3)

![Frequency Response, M= 9](image4)
Filter Design

The larger smoothing filters remove more high-frequency energy. This removes more of the noise and it also removes detail information from edges and other image features.

Averaging filters can emphasize some pixels more than others. Here is a mask that emphasizes the center pixels more than the edges.

\[
\begin{array}{cccccc}
1 & 1 & 2 & 1 & 1 \\
1 & 2 & 3 & 2 & 1 \\
2 & 3 & 4 & 3 & 2 \\
1 & 2 & 3 & 2 & 1 \\
1 & 1 & 2 & 1 & 1 \\
\end{array}
\]

\[
\frac{1}{25}
\]
Filter Design

The frequency response of this lowpass filter compared with the order 5 averaging filter is shown below. (LP filter: solid line; AVG filter, dashed line)

The LP filter has lower sidelobes but is wider in the main lobe. This illustrates the tradeoff that can be made by adjusting the mask weights.
Filtered Images

The effect of the lowpass filter compared with the order 5 averaging filter is illustrated below with filtered noisy images. The LP filter has a slightly broader frequency response, which shows up as slightly less blurring in the right-hand picture.
Example: Smoothing of Test Pattern

Shown at the right is the test pattern image of G&W Figure 3.35.

We will examine the effects of smoothing this image with filters of different sizes.

We will make graphs of the horizontal slice through the vertical bars and noisy box along row 125, which is indicated by the gray horizontal line.
Smoothing Filter of Size $3 \times 3$

(Right Top) original image; (Right Bottom) result of smoothing with a $3 \times 3$ filter; (Below) Plots along row 125 (top) entire row, (middle) a section of the vertical bars, (bottom) a section of the noisy box.
Smoothing Filter of Size $9 \times 9$

(Right Top) original image; (Right Bottom) result of smoothing with a $9 \times 9$ filter; (Below) Plots along row 125 (top) entire row, (middle) a section of the vertical bars, (bottom) a section of the noisy box.
Smoothing Filter of Size $25 \times 25$

(Right Top) original image; (Right Bottom) result of smoothing with a $25 \times 25$ filter; (Below) Plots along row 125 (top) entire row, (middle) a section of the vertical bars, (bottom) a section of the noisy box.
Smoothing Filter of Size $35 \times 35$

(Right Top) original image; (Right Bottom) result of smoothing with a $35 \times 35$ filter; (Below) Plots along row 125 (top) entire row, (middle) a section of the vertical bars, (bottom) a section of the noisy box.
Programming

Linear filtering programs are quite easy to write in IDL by using the built-in function, CONVOL. When all of the filter weights are equal then one can use the function, SMOOTH.

It is worthwhile using these functions because they run much faster than any user functions that would be written in IDL.

CONVOL is related to mathematical convolution and to filter weighting. How it works depends upon the choices of some keyword parameters.
CONVOL

The CONVOL function convolves an array with a kernel, and returns the result. If \( A \) is an \( N \times M \) array and \( K \) is a \( n \times m \) kernel, with \( n < N \) and \( m < M \), then \( B = \text{CONVOL}(A, K, S) \) computes an output array \( B \) by

\[
B[t, u] = \frac{1}{S} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} A[t + i - n/2, u + j - m/2] K[i, j]
\]

**Example 1:** \( B = \text{CONVOL}(A, K, 1) \) where

\[
A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

and \( K = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix} \)
Example 1 (cont)

Since $n = m = 3$ we have $n/2 = m/2 = 1$ (integer division). Since $A$ is zero everywhere except $A[2, 2] = 1$ we can write $A[r, s] = \delta[r - 2, s - 2]$. Then

$$A[t + i - 1, u + j - 1] = \delta[t + i - 3, u + j - 3]$$

so that

$$B[t, u] = \sum_{i=0}^{2} \sum_{j=0}^{2} \delta[t + i - 3, u + j - 3]K[i, j] = K[3 - t, 3 - u]$$

$$A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\quad K = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
\Rightarrow B = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 9 & 8 & 7 & 0 & 0 & 0 \\
0 & 6 & 5 & 4 & 0 & 0 & 0 \\
0 & 3 & 2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

DIP Lecture 7 22
Example 1 (cont)

The result $B$ contains the kernel pattern, reversed in rows and columns, and centered on the location of the “impulse” in the array $A$.

The kernel is reversed in the result because it is not reversed in the equation, as it would be with true convolution.

Mathematical convolution is carried out by unsetting the CENTER keyword. $C=\text{CONVOL}(A,K,S,\text{CENTER}=0)$ with $S=1$ produces

$$A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \quad K = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix} \Rightarrow B = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 3 & 0 & 0 \\
0 & 0 & 4 & 5 & 6 & 0 & 0 \\
0 & 0 & 7 & 8 & 9 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$
CONVOL with CENTER=0

When the keyword CENTER is explicitly set to zero, then the equation computed by the routine is (in 2D)

\[ C[t, u] = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} A[t - i, u - j] K[i, j] \]

In the case \( A[r, s] = \delta[r - 2, s - 2] \) we have

\[ C[t, u] = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \delta[t - i - 2, s - j - 2] K[i, j] = K[t - 2, s - 2] \]

The result has the kernel “unflipped” but shifted so that its corner is in the position of the impulse in \( A \).
Using CONVOL

CONVOL uses the keywords CENTER, EDGE_TRUNCATE, and EDGE_WRAP to control the computation. We will do several computation examples in class to illustrate the different results.

The result returned by CONVOL is always of the same size as the input array, namely, $N \times M$. True convolution produces an output of size $N + n - 1, M + m - 1$. It is necessary to use zero-padding to achieve this result. This will be discussed further in connection with the FFT.

The kernel must be smaller than the array.
SMOOTH

The IDL function SMOOTH implements a strictly uniform kernel. It is useful to achieve uniform smoothing.

The code has been written to take advantage of the uniform kernel, which makes it faster than CONVOL.
Discrete 1D Convolution

Let $f(x)$ be defined for $x = 0, 1, 2, \ldots, A - 1$ and $h(x)$ be defined for $x = 0, 1, 2, \ldots, B - 1$. Let $M$ be an integer such that

$$M \geq A + B - 1$$

Let $f_e(x)$ and $h_e(x)$ be extended by zero padding such that

$$f_e(x) = \begin{cases} 
  f(x) & 0 \leq x \leq A - 1 \\
  0 & A < x \leq M - 1
\end{cases}$$

$$h_e(x) = \begin{cases} 
  h(x) & 0 \leq x \leq B - 1 \\
  0 & B < x \leq M - 1
\end{cases}$$

for the interval $0 \leq x \leq M - 1$. Then extend $f_e(x)$ and $h_e(x)$ for all integers $x$ by repeating the basic interval with period $M$. 
Discrete 1D Convolution (cont)

Then the discrete convolution of $f$ and $h$ is

$$g(x) = f \ast h = \sum_{m=0}^{M-1} f_e(x) h_e(m - x)$$

Note that $g(x)$ is periodic with period $M$ and is defined for all $x \in \mathbb{Z}$. How would you show that?
Matrix Expression for 1D Convolution

The main period of $g(x)$, $0 \leq x \leq M - 1$ can be computed by

$$g = Hf$$

where $g$ and $f$ are column vectors of length $M$ and $H$ is a $M \times M$ matrix

$$
\begin{bmatrix}
g(0) \\
g(1) \\
\vdots \\
g(M - 1)
\end{bmatrix} =
\begin{bmatrix}
h_e(0) & h_e(-1) & h_e(-2) & \ldots & h_e(-M + 1) \\
h_e(1) & h_e(0) & h_e(-1) & \ldots & h_e(-M + 2) \\
h_e(2) & h_e(1) & h_e(0) & \ldots & h_e(-M + 3) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
h_e(M - 1) & h_e(M - 2) & h_e(M - 3) & \ldots & h_e(0)
\end{bmatrix}
\begin{bmatrix}
f_e(0) \\
f_e(1) \\
f_e(2) \\
\vdots \\
f_e(M - 1)
\end{bmatrix}
$$

By making use of periodicity of $h_e$ we have the equivalent expression

$$
\begin{bmatrix}
g(0) \\
g(1) \\
\vdots \\
g(M - 1)
\end{bmatrix} =
\begin{bmatrix}
h_e(0) & h_e(M - 1) & h_e(M - 2) & \ldots & h_e(1) \\
h_e(1) & h_e(0) & h_e(M - 1) & \ldots & h_e(2) \\
h_e(2) & h_e(1) & h_e(0) & \ldots & h_e(3) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
h_e(M - 1) & h_e(M - 2) & h_e(M - 3) & \ldots & h_e(0)
\end{bmatrix}
\begin{bmatrix}
f_e(0) \\
f_e(1) \\
f_e(2) \\
\vdots \\
f_e(M - 1)
\end{bmatrix}
$$

Each row of $H$ is a circular shift of the one above. This form is called a circulant matrix.
2D Convolution

In this case $f(x, y)$ and $h(x, y)$ are 2D discrete arrays.

\[
f_e(x, y) = \begin{cases} f(x, y) & 0 \leq x \leq A - 1 \quad 0 \leq y \leq B - 1 \\ 0 & A \leq x \leq M - 1 \quad B \leq y \leq N - 1 \end{cases}
\]

\[
h_e(x, y) = \begin{cases} h(x, y) & 0 \leq x \leq C - 1 \quad 0 \leq y \leq D - 1 \\ 0 & C < x \leq M - 1 \quad C \leq y \leq N - 1 \end{cases}
\]

where $M = A + C - 1$ and $N = B + D - 1$. The function repeats periodically in a tiling fashion with tiles of size $M \times N$.

\[
g(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_e(m, n) h_e(x - m, y - n)
\]