Linear Spatial Filtering

Linear spatial filtering modifies an image f by replacing the value at each pixel with some linear function of the values of nearby pixels. Moreover, this linear function is assumed to be independent of the pixel's location (i, j), where (i, j) indexes the pixels in f, which is represented as a m_r by m_c matrix. This kind of operation can be expressed as convolution or correlation. For spatial filtering, it's often more intuitive to work with correlation, which will be defined later.

As a motivating example, suppose we want to replace each pixel in f with the average intensity in a three by three neighborhood of pixels centered there. Then if g is the filtered image,

$$\begin{aligned} g(i,j) = \frac{1}{9}\,(&f(i-1,j-1) + f(i-1,j) + f(i-1,j+1) + \\ &f(i,j-1) + f(i,j) + f(i,j+1) + \\ &f(i+1,j-1) + f(i+1,j) + f(i+1,j+1))\,. \end{aligned}$$

We can think of this filtering operation in terms of the mask

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

The filtered result $g(i, j)$ is obtained by centering the mask over pixel (i, j), multiplying the elements of f with the overlapping elements of the mask and then adding them up.

Something special must be done when the center of the mask is on the boundary of f. In the above averaging example, the formula for g requires evaluating f at places where it's not defined if (i, j) is on the boundary. To remedy this, we consider f to be periodic, which means $f(i + m_r, j) = f(i, j)$ and $f(i, j + m_c) = f(i, j)$ for all integers (i, j).

This periodicity introduces another problem, however. Now filtered values at the boundary can depend on information coming from opposite sides of f, something we probably don't want to happen. A common way to remedy this is to pad the image with zeros. More precisely, let w be the mask, which for simplicity we will assume to be defined as an n by n matrix. Recall that f is a m_r by m_c matrix. Now we will pad f and w by defining $m_r + n - 1$ by $m_c + n - 1$ matrices f_p and w_p. The matrix w_p equals zero except for the n by n submatrix in the upper left corner, which equals w. In MATLAB notation, $w_p(1 : n, 1 : n) = w$. We will additionally define the matrix f_p to equal zero except for a m_r by m_c submatrix where it equals f. Instead of being in the upper left corner, this submatrix is shifted by $\frac{n}{2}$ rounded

down to the nearest integer. In MATLAB notation, $f_p(1+r:m_r+r, 1+r:m_c+r) = f$ where $r = \text{floor}(\frac{n}{2})$. This is just one of many possible ways of padding f and w. The reason for shifting by r is so the filtered image corresponds spatially to the original.

With the zero padded matrices f_p and w_p defined as above, the filtered matrix g can then be obtained by computing the correlation $w_p \star f_p$ defined by

$$(w_p \star f_p)(s,t) = \sum_{i=1}^{m_r+n-1} \sum_{j=1}^{m_c+n-1} w_p(i,j) f_p(s+i-1, t+j-1).$$

This is a $m_r + n - 1$ by $m_c + n - 1$ matrix with elements indexed by (s,t). Only the upper left m_r by m_c submatrix is needed to define g. So we can restrict $s = 1, ..., m_r$, $t = 1, ..., m_c$ and define g by

$$g(s,t) = (w_p \star f_p)(s,t).$$

Note that the above definition of $w_p \star f_p$ is consistent with MATLAB's convention of indexing starting at one instead of zero.

Taking into account where w_p equals zero, the correlation $w_p \star f_p$ could be equivalently computed by

$$(w_p \star f_p)(s,t) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_p(i,j) f_p(s+i-1, t+j-1).$$

For large n, $w_p \star f_p$ can be computed more quickly using the fast Fourier transform (FFT), a fast method for computing the discrete Fourier transform (DFT). This makes use of the fact that the DFT of $w_p \star f_p$ is the componentwise product of the DFT of f_p and the complex conjugate of the DFT of w_p.

MATLAB: We can test the accompanying MATLAB code filt.m on a test image by typing filt(f,w), where f is the image and w is the mask. For example, to apply the three by three averaging filter to elvis.bmp, we would define

```
f = double(imread('elvis.bmp'));
w = ones(3)/9;
g = filt(f,w);
```

Then g is the filtered image. The effect in this case is a slight blurring of the image, and the result is shown in Figure 1b, where Figure 1a shows the original image.

We can try other masks as well. For example, defining $w = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$ identifies vertical edges.

```
w = [1 -1 ; 1 -1];
```

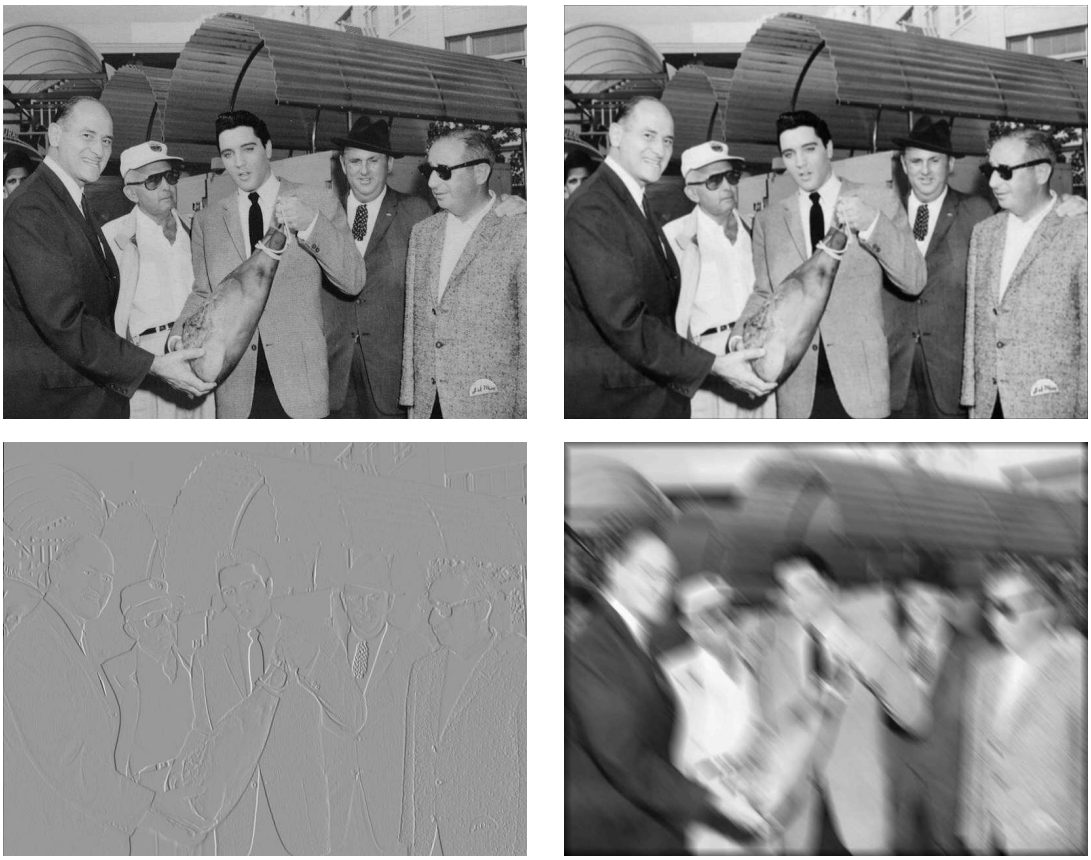

Figure 1: Three linear spatial filtering examples

a. original image b. filtered using $w = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

c. $w = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$ d. $w = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1 \end{bmatrix}$

The rescaled result is shown in Figure 1c. Another example is to let w be the 25 by 25 identity matrix.

```
w = eye(25);
```

The rescaled result looks like a motion blur in a diagonal direction and is shown in Figure 1d.

MATLAB has many built in filtering functions in its signal and image processing toolboxes. There is excellent documentation on the Mathworks website,

`http://www.mathworks.com/access/helpdesk/help/helpdesk.html`

Question: What happens if we compute $f_p \star w_p$ instead of $w_p \star f_p$?

Reference: R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Third Edition, 2008.