Filtering of Images

Objectives:

- To perform image blurring operations.
- To use MATLAB to perform one-dimensional filtering of an image.
- To investigate several kinds of image filters.
- To investigate image scaling.

Things to Remember:

1) An important property of images such as photographs and TV pictures is that their values are always non-negative and finite in magnitude. This is because light images are formed by measuring the intensity of reflected or emitted light, which is always a positive finite quantity.

2) Filtering may introduce negative values, especially if high pass filters are used.

3) The default format for most gray-scale displays is 8 bits, so the pixel values f [i, j] are integers in the range from 0 to 255.

4) The MATLAB functions `max` and `min` can be used to find the largest and smallest values in the image.

5) The MATLAB functions `round`, `fix` and `floor` can be used to quantize pixel values to integers.

6) When the image values lie outside the range [0, 255] or when the image is scaled so that it occupies only a small portion of the range [0, 255], the display may have poor quality. We can analyze this condition using the MATLAB function `hist` (called histogram) to plot how often each pixel occurs.

Procedure A: Warm-up

1. Create a test image that is 256×256 by making each horizontal line a discrete time sinusoid with a period of 50 samples.

2. Extract one row from the image and make a plot to verify that the horizontal line is a sinusoid.

3. Display the image from step 1.

4. If the sinusoid was generated with values between –1 and +1, scale it by writing your own MATLAB code.

5. Now load and display the `lenna` image from `lenna.mat`. Use the `whos` command, if necessary, to find the variable name into which the `load lenna` command put the `lenna` image. Most likely it will be array `xx`.

6. Make a plot of the 200th row of the `lenna` image. Observe the maximum and minimum values.

7. Consider a way in which you might display the negative of the `lenna` image. Try it.

Procedure B: One-dimensional Filtering

1. Load the 256 × 256 chessboard image you made earlier in the course. Extract row 33 from the bottom of the image. Filter this one-dimensional signal with a 7-point avarager:

```
h=ones(1,7)/7;        %suppose that your chessboard image is xx
y1=conv(xx33,h);
```

   Plot both the input and the output signal in the same figure using a two-panel subplot. Observe whether or not the filtered waveform is "smoother" or "rougher" than the input. Explain.

2. Now extract the 99th column of the image and filter it with a first order difference filter:

   $y[n] = x[n] - x[n-1]$ (in MATLAB use vector [1 –1])

3. Plot the input and output together for comparison. Once again observe whether or not the filtered waveform is "smoother" or "rougher" than the input. Explain.

Procedure C: Blurring an image

From the above procedure, perhaps you can see how you could use a `for loop` to write an m-file that would filter all of the rows. However, this would filter your image only in the horizontal direction. Filtering the columns would require another loop. Thus the image would be filtered in both directions by a 7-point averager. Nested `for loops` are at the core of any 2D convolution program.

Fortunately, two-dimensional convolution is available in MATLAB. This function would accept a filter that is one-dimensional (either horizontal or vertical). The filtering of the image will be done in the direction of the filter vector.

1) Filter the image in the horizontal direction using a 7-point averager stored in the row vector *h*. Use `conv2`. Display the input image *xx* and the output image *y1* on the screen using the first two subplot positions of a 2×2 subplot figure. Compare the two images and give qualitative descriptions of what you see. Extract row 33 from the bottom of the output image and compare it to the output obtained in item 1 of Procedure B.

2) Now filter the image in the vertical direction with a 7-point averager to produce image y2. This is done by calling convolution with the transpose of vector *h*. The transpose of vector *h* is *h'*. Plot all three images on the screen. Describe what you see.

3) What filter do you think will cause a more severe degradation of the original image, a 7-point averager or a 21-point averager? Try it and comment.

Procedure D: More Image Filters

Load the `lenna` image into MATLAB and display it.

Filter the `lenna` image with each of the following filters. Remember to filter both the rows and columns, unless otherwise directed.

a) `a1 = [1 1 1]/3;`
b) `a2 = ones(1,7)/7;`
c) `a3 = [1 -1];` For this filter, filter only the rows. Note that this filter will yield negative values. Before displaying the resulting image it must be scaled to fit back into the allowable range [0 255]. (optionally, you can do the scaling).
d) `a4 = [-1 3 -1];` For this filter, filter only the rows.
e) `a5 = [-1 1 1 1 -1];`

Comment on the effect of each filter, paying special attention to regions of the image with plenty of detail, such as feathers. Answer the following questions by showing representative images that exhibit the characteristics of high-pass or low-pass filtering.

- Which filters appear to be low-pass filters?
- What is the general effect of a low-pass filter on an image?
- Which ones are high-pass filters?

Procedure E

Using your own MATLAB program that performs 2D convolution (not the conv2 function of MATLAB), apply the following filters to Lenna and examine the results.

Smoothing Filters 1 & 2:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian Filters:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gradient Filters 1 & 2:

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

Gradient Filter 3:

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 1 & -1 \\ 1 & 1 & 0 \end{bmatrix}$$

Sobel Filter 1:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel Filter 2:

$$\begin{bmatrix} 0 & -1 & -1 & -1 & 0 \\ -1 & -2 & -2 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 2 & 2 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Prewitt Filter:

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Laplacian Filters 1 & 2:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 9 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

In addition to the individual comment that should accompany the result of each filter, you should include an overall discussion of these filters. Compare the various filters and discuss effects such as highlighting versus edge extraction and edge thickness.

To be economical with paper and toner, please insert the output figures into an MS Word file, instead of printing them individually. This has the added advantage of saving you write-up time.