See Figure 6.3 on page 4 below.

6.2 Image Processing with One-Dimensional Filters

A two-dimensional (2D) discrete-time signal $x[m, n]$ has two index variables rather than one. An image is a 2D discrete-time signal which is nonzero only over a finite region, e.g., the N-by-N square $0 \leq m, n \leq N-1$. For such images, m corresponds to the row index while n corresponds to the column index. Two 32-by-32 pixel images are illustrated in Figure 6.3. (A pixel is the name for a sample of an image, and is short for *picture element.*) Many of the time-domain and frequency-domain tools used for designing and analyzing one-dimensional (1D) signals apply equally well to 2D signals.[1] In particular, a widely-used method for filtering 2D signals is to process the signal with a cascade of two 1D filters, one filter for each direction. The cascade of these two 1D filters, called a separable 2D filter, is given by the following two steps: (i) for each n compute the output $z[m, n]$ of the LTI system which satisfies

$$\sum_{k=-K}^{K} a_k z[m-k, n] = \sum_{j=-K}^{K} b_j x[m-j, n]\,; \tag{6.5}$$

(ii) for each m compute the output $y[m, n]$ of the LTI system which satisfies

$$\sum_{k=-K}^{K} c_k y[m, n-k] = \sum_{j=-K}^{K} d_j z[m, n-j]\,. \tag{6.6}$$

The output $y[m, n]$ is the filtered version of $x[m, n]$.
Note that both the column-wise filter defined by the coefficients a_k and b_j and the row-wise filter defined by the coefficients c_k and d_j are noncausal. In the context of image processing, causality is not a constraint, since one is often given the entire image before it is to be processed.
In this exercise, you will lowpass filter an image using a cascade of two 1D filters. For simplicity, the two cascaded 1D filters are identical, i.e., $c_k = a_k$ and $d_j = b_j$. You will see that the time-domain and frequency-domain tools you have learned for 1D signals can provide considerable insight into 2D filtering.
For this exercise, you will need to load the data file `plus.mat`, which is provided in the Computer Explorations Toolbox. If this file is already in your MATLABPATH, you can load the data by typing `load plus`. If loaded correctly, your workspace will then contain a matrix `x`, which contains the image to be processed in this exercise, and a noise-corrupted version of `x` in the matrix `xn`, which will be filtered in the Advanced Problems. To display the image, type

```
>> colormap(gray);
>> image(64*x);
```

Your image should be identical to the one in Figure 6.3(a). The command `colormap(gray)` is used to select the mapping from the values of `x` to the colors displayed by `image`. Also, because the colormap has 64 color levels, the input to `image` is scale by 64.

Basic Problems

In these problems you will create three 1D lowpass filters, each of which will eventually be used to lowpass filter the image contained in `x`. Analyzing the properties of these filters in 1D will be useful for interpreting the images filtered by a cascade of two 1D filters.

(a). The functions `butter` and `remez` both return the coefficients of difference equations which define 1D discrete-time filters. The filters determined by `butter` have infinite-length impulse responses, while the filters determined by `remez` have finite-length impulse responses. Type

```
>> wc = 0.4;
>> n1 = 10; n2 = 4; n3 = 12;
>> [b1,a1] = butter(n1,wc);
>> a2 = 1; b2 = remez(n2,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
>> a3 = 1; b3 = remez(n3,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
```

where `wc` is the cutoff frequency of each filter (normalized by π) and `n1`, `n2`, and `n3` are the orders of the three filters.

(b). Use `freqz`, which is described in Tutorial 3.2, to plot the magnitude and phase of each of the three 1D filters determined in Part (a). As a check, verify that `wc` is the approximate cutoff frequency of each filter. Which filters have linear phase?

(c). Use `filter` to determine the step response of each filter for `n=[0:20]`. Plot each of these step responses. Which step response has the largest overshoot, i.e., the difference between the steady-state value and the maximum value of the step response? You will see that large overshoots or ringing in the step response can lead to undesirable artifacts in the filtered images.

Intermediate Problems

For image processing applications, noncausal filters can be used and are in fact desired. One reason for using noncausal filters is that the contents of an image should not be shifted in location after processing. In the following problems, you will first learn how to implement 1D noncausal filters using `filter`. Then you will write an M-file which will repeatedly call `filter` to process the columns and rows of the image stored in `x`.

The function `filter` assumes that the filter coefficients stored in the vectors `a` and `b` correspond to a causal filter. For example, if `x16` contains the column of the image for $n = 16$, i.e., $x[m, 16]$ for $0 \le m \le N-1$, then `y16=filter(b,a,x16)` stores in `y16` the output of the causal filter

$$\sum_{k=0}^{2*\mathtt{d}} \mathtt{a}(k+1)\, y[m-k, 16] = \sum_{j=0}^{2*\mathtt{d}} \mathtt{b}(j+1)\, x[m-j, 16] \qquad (6.7)$$

on the interval $0 \le m \le N-1$, where the vectors `a` and `b` have `2*d+1` elements. For this exercise, however, you will need to implement the following noncausal filter using `filter`:

$$\sum_{k=-\mathtt{d}}^{\mathtt{d}} \mathtt{a}(k+1+\mathtt{d})\, y[m-k, 16] = \sum_{j=-\mathtt{d}}^{\mathtt{d}} \mathtt{b}(j+1+\mathtt{d})\, x[m-j, 16]\,. \qquad (6.8)$$

This filter is in the same form as the noncausal filters given by Eqs. (6.5) and (6.6). Noting the similarity between Eqs. (6.7) and (6.8), you can see that `y16=filter(b,a,x16)` computes the output of the noncausal filter on the interval $-\mathtt{d} \le m \le N-\mathtt{d}-1$. In other words, `d` is equal to the advance induced by `filter`. If the vector returned by `filter` is to contain the output of the noncausal filter for $n = 16$ and $0 \le m \le N-1$, the input to `filter` must contain $x[m, 16]$ for $0 \le m \le N + \mathtt{d} - 1$. Since $x[m, 16]$ is not known outside the interval $0 \le m \le N-1$, the values outside this interval can assumed to be zero. Therefore, the output of the noncausal filter on the interval $0 \le m \le N-1$ can be extracted from the last N samples the signal produced by `filter(b,a,[x16; zeros(d,1)]`.

(d). Create the column vector `x16=x(:,16)`, which provides a 1D signal defined on the interval $0 \le m \le 31$. Use `filter` to compute the output of the noncausal filter given by Eq. (6.8) for `a=a1` and `b=b1`, assuming the input is given by `x16` for $0 \le m \le 31$ and is zero elsewhere. Store in `y16` the filter response on the interval $0 \le m \le 31$. Plot `x16` versus `y16`. If `y16` is computed correctly, the discontinuities in `x16` line up with the "smoothed" discontinuities in `y16`.

(e). For filters two and three, determine the response of these 1D filters to the column vector `x16`. For each filter, you will have to append zeros to `x16` before calling `filter`. Again plot the responses versus `x16` to ensure that the discontinuities are aligned and there is no delay or advance in your filter implementation.

(f). Write an M-file for the function `filt2d` which noncausally filters the 2D image stored in the matrix `x`. The first line of the your M-file `filt2d.m` should read

```
function y = filt2d(b,a,d,x)
```

The vectors `a` and `b` contain the coefficients of the 1D filter and `d` contains the "delay" associated with a causal implementation of the filter. (Note that `d` is just `n/2`, where `n` is the order of the 1D filter). Your M-file should essentially consist of two steps: (i) filtering each column of an N-by-N matrix `x` and storing the results in an N-by-N matrix `z`, followed by (ii) filtering each row of `z`. Note that you will have to append an appropriate number of zeros to each column of `x` and each row of `z` before filtering. The matrix `y` returned by `filt2d` should have the same dimensions as `x`.

(g). Use `image` to display the filtered image given by `filt2d` for each of the three filters. (Remember to scale the input to `image` by 64.) Make sure that the image has not been delayed, i.e., the filtered "plus" contained in `y` should have the same center location as the "plus" in `x`.

(h). Which filter leads to more distortion in the shape of the original image? Pay attention to any destruction of the symmetries present in the input image. This distortion is due in part to the nonlinearity of the phase in one of the filters.

Advanced Problems

Now you will compare the outputs of the second and third filters. These filters both have finite-length impulse responses but require different numbers of coefficients.

(i). Store in `y2` and `y3` the matrices returned by filtering `x` with `filt2d` using the second and third 1D filters.

(j). Compare `y2` and `y3` using `image`. Which output image has more oscillations? How could you have predicted the oscillations from the step responses of the two filters?

For many image processing applications, filters with smooth transition bands are preferred over filters which more closely approximate the frequency response of an ideal lowpass filter, which has a sharp transition band. One application which illustrates the advantages of a smoother transition band is the removal of noise from images.

(k). Display the noisy image using `image(64*xn)`. Your image should be identical to the one in Figure 6.3(b).

(l). Determine the output of `filt2d` for the second and third filters for the noisy input `xn`. Display both outputs using `image`. Which filter does a better job of removing the noise? Pay particularly close attention to the output signals in areas for which `x` is zero. You may wish to explore the effect of changing the parameters of the filters returned by `butter` and `remez`.

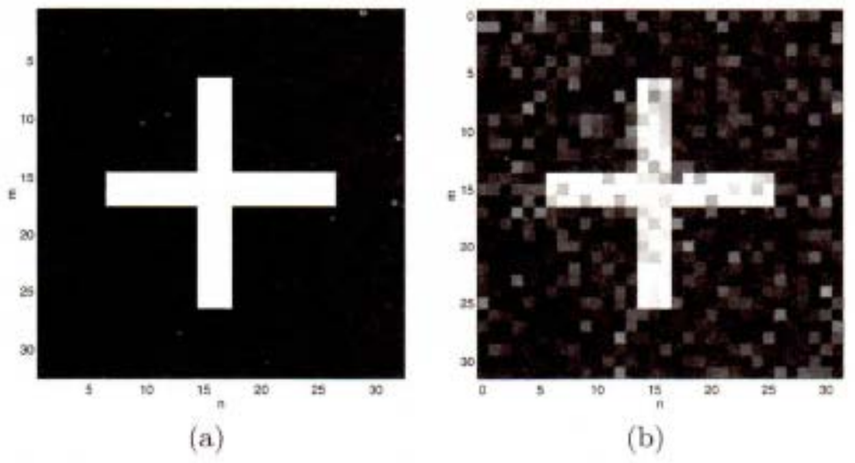

Figure 6.3. Two 32-by-32 pixel images: (a) the "plus sign", and (b) a noisy version of the "plus sign".