GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

**ECE 2026 Fall 2013**

# Lab #8: FIR Filtering of Digital Images

Date: 28 Oct-31 Oct 2013

Each Lab assignment in ECE2026 consists of three parts: Pre-Lab, In-lab Tasks, and Take-home Questions. It requires you to come into lab prepared. Be sure to read the entire lab carefully before arriving.

**Pre-Lab:** You should read the Pre-Lab section of the lab and go over all exercises in this section before going to your assigned lab session. Although you do not need to turn in results from the Pre-Lab, doing the exercises therein will help make your in-lab experience more rewarding and go more smoothly with less frustration and panicking.

**In-lab Tasks and Verification:** There are a number of designated tasks for each student to accomplish during the lab session. Students are encouraged to read, prepare for and even attempt at these tasks beforehand. These tasks must be completed **during your assigned lab time** and the steps marked *Instructor Verification* must also be signed off **during the lab time**. One of the laboratory instructors must verify the appropriate steps by signing on the **Instructor Verification** line. When you have completed a step that requires verification, simply put a plastic cup on top of your PC and demonstrate the step to one of the TAs or the professor. (You can also use the plastic cups to indicate if you have a more general question, i.e. you can use it to get our attention even if you don't have an Instructor Verification ready.)

**Take-home Questions:** At the end of each lab sheet below all the verification steps, several questions are to be answered by the student, who can choose to complete the answers while in the lab or after leaving the lab.

The lab sheet with all verification signatures and answers to the questions needs to be turned in to the Lab-grading TA at the beginning of the next lab session.

## 1. Introduction

The goal of this lab is to learn how to implement FIR filters in MATLAB, and then study the response of FIR filters to various signals, including images or speech. As a result, you should learn how filters can create interesting effects such as blurring and echoes. In addition, we will use FIR filters to study the convolution operation and properties such as linearity and time-invariance.

In the experiments of this lab, you will use **`firfilt()`**, or **`conv()`,** to implement 1-D filters and **`conv2()`** to implement two-dimensional (2-D) filters. The 2-D filtering operation actually consists of 1D filters applied to all the rows of the image and then all the columns.

### 1.1 Discrete-time Convolution GUI

This lab involves the use of a MATLAB GUI for convolution of discrete-time signals, **`dconvdemo`**. This is exactly the same as the MATLAB functions **`conv()`** and **`firfilt()`** used to implement FIR filters. This demo is part of the SP-First Toolbox.

### 1.2 Overview of Filtering

For this lab, we will define an FIR filter as a discrete-time system that converts an input signal $x[n]$ into an output signal $y[n]$ by means of the weighted summation formula:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] \tag{1}$$

Equation (1) gives a rule for computing the $n^{\text{th}}$ value of the output sequence from present and past values of the input sequence. The filter coefficients $\{ b_k \}$ are constants that define the filter's behavior. As an example, consider the system for which the output values are given by

$$\begin{aligned} y[n] &= \tfrac{1}{3}x[n] + \tfrac{1}{3}x[n-1] + \tfrac{1}{3}x[n-2] \\ &= \tfrac{1}{3}\{x[n] + x[n-1] + x[n-2]\} \end{aligned} \tag{2}$$

This equation states that the $n^{\text{th}}$ value of the output sequence is the average of the $n^{\text{th}}$ value of the input sequence $x[n]$ and the two preceding values, $x[n-1]$ and $x[n-2]$. For this example, the $b_k$'s are $b_0 = \frac{1}{3}, b_1 = \frac{1}{3}$, and $b_2 = \frac{1}{3}$.

MATLAB has two built-in functions, **`conv()`** and **`filter()`**, for implementing the operation in (1), and the SP-First toolbox supplies another M-file, called **`firfilt()`**, for the special case of FIR filtering. The **`firfilt.m`** function filter implements a wider class of filters than just the FIR case. Technically speaking, both the **`conv`** and the **`firfilt`** function implement the operation called *convolution*. The following MATLAB statements implement the three-point averaging system of (2):

```
nn = 0:99;               %<--Time indices
xx = cos( 0.08*pi*nn ); %<--Input signal
bb = [1/3 1/3 1/3];     %<--Filter coefficients
yy = firfilt(bb, xx);   %<--Compute the output
```

In this case, the input signal **`xx`** is contained in a vector defined by the cosine function. In general, the vector **`bb`** contains the filter coefficients $\{b_k\}$ needed in (1). The **`bb`** vector is defined in the following way:

```
bb = [b0, b1, b2, ... , bM].
```

In MATLAB, all sequences have finite length because they are stored in vectors. If the input signal has $L$ nonzero samples, we would normally store only the $L$ nonzero samples in a vector, and would assume that $x[n] = 0$ for n outside the interval of $L$ samples, i.e., don't store any zero samples unless it suits our purposes. If we process a finite-length signal through (1), then the output sequence $y[n]$ will be longer than $x[n]$ by $M$ samples. Whenever **`firfilt()`** implements (1), we will find that

```
length(yy) = length(xx)+length(bb)-1
```

In the experiments of this lab, you will use **`firfilt()`** to implement FIR filters and begin to understand how the filter coefficients define a digital filtering algorithm. In addition, this lab will introduce examples to show how a filter reacts to different frequency components in the input.

## 2. Pre-Lab: Run the GUI

The first objective of this lab is to demonstrate usage of the **`dconvdemo`** GUI. If you have installed the SP-First Toolbox, you will already have this demo on the **`matlabpath`**.

### 2.1 Discrete-Time Convolution Demo

In this demo, you can select an input signal $x[n]$, as well as the impulse response of the filter $h[n]$. Then the demo shows the sliding window view of FIR filtering, where one of the signals must be flipped and shifted along the axis when convolution is computed. Figure 1 shows the interface for the **`dconvdemo`** GUI. In the pre-lab, you should perform the following steps with the **`dconvdemo`** GUI.

(a) Click on the **`Get x[n]`** button and set the input to a finite-length pulse: $x[n] = (u[n] - u[n-10])$. Note the length of this pulse.

(b) Set the filter to a three-point averager by using the **`Get h[n]`** button to create the correct impulse response for the three-point averager. Remember that the impulse response is identical to the $b_k$'s for an FIR filter. Also, the GUI allows you to modify the length and values of the pulse.

(c) Observe that the GUI produces the output signal in the bottom panel.

(d) When you move the mouse pointer over the index "*n*" below the signal plot and do a click-hold, you will get a hand tool that allows you to move the "*n*"-pointer to the left or right. By moving the pointer horizontally you can observe the sliding window action of convolution. You can even move the index beyond the limits of the window and the plot will scroll over to align with "*n*.

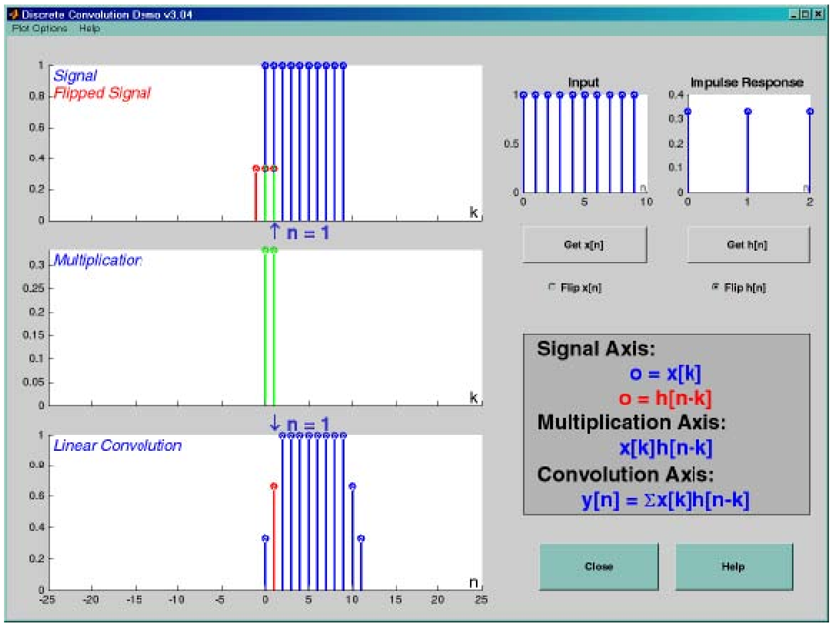

Figure 1: Interface for discrete-time convolution GUI called **`dconvdemo`**. This is the convolution of a three-point averager with a ten-point rectangular pulse.

## 2.2 Filtering via Convolution

You can perform the same convolution as done by the **`dconvdemo`** GUI by using the MATLAB function **`firfilt`**, or **`conv`**. For ECE-2026, the preferred function is **`firfilt`**.

(a) For the Pre-Lab, you should do some filtering with a three-point averager. The filter coefficient vector for the three-point averager is defined via:

```
bb = 1/3*ones(1,3);
```

Use **`firfilt`** to process an input signal that is a length-10 pulse:

$$x[n] = \begin{cases} 1, & \text{for } n = 0,1,2,3,4,5,6,7,8,9 \\ 0, & \text{elsewhere} \end{cases}$$

Note: in MATLAB indexing can be confusing. Our *mathematical* signal definitions start at $n = 0$, but MATLAB starts its indexing at "1". Nevertheless, we can ignore the difference and pretend that MATLAB is indexing from zero, as long as we don't try to write $x[0]$ in MATLAB. For this experiment, generate the length-10 pulse and put it inside of a longer vector with the statement **`xx = [ones(1,10),zeros(1,5)]`**. This produces a vector of length 15, which has 5 extra zero samples appended.

(b) To illustrate the filtering action of the three-point averager, it is informative to make a plot of the input signal and output signal together. Since $x[n]$ and $y[n]$ are discrete-time signals, a stem plot is needed. One way to put the plots together is to use **`subplot(2,1,*)`** to make a two-panel display:

```
nn = first:last; %---use first=1 and last=length(xx)
subplot(2,1,1);
```

```
stem(nn-1,xx(nn))
subplot(2,1,2);
stem(nn-1,yy(nn),'filled') %--Make black dots
xlabel('Time Index (n)')
```

This code assumes that the output from **firfilt** is called **yy**. Try the plot with first equal to the beginning index of the input signal, and last chosen to be the last index of the input. In other words, the plotting range for both signals will be equal to the length of the input signal, even though the output signal is longer. Notice that using **nn-1** in the two calls to **stem()** causes the x-axis to start at zero in the plot; that has been an indexing convention since the early days of computing.

(c) Explain the filtering action of the three-point averager by comparing the plots in the previous part. This averaging filter might be called a "smoothing" filter, especially when we see how the transitions in $x[n]$ from zero to one, and from one back to zero, have been "smoothed."

## 3. Lab Exercise

### 3.1 Discrete-Time Convolution

In this section, you will generate filtering results needed in a later section. Use the discrete-time convolution GUI, **dconvdemo**, to do the following:

(a) The convolution of two impulses, $\delta[n-1]*\delta[n-4]$.

(b) Filter the input signal $x[n]=u[n-9]-u[n-2]$ with a first-difference filter. Make $x[n]$ by selecting the "Pulse" signal type from the drop-down menu within **Get x[n],** and also use the text box "**Delay**." Set the impulse response to match the filter coefficients of the first-difference. Enter the impulse response values by selecting "**User Signal**" from the drop-down menu within **Get h[n].** The GUI will produce the convolution output. Write a simple formula for the output $y[n]$ using unit impulses.

(c) The GUI allows you to "grab" the index arrow and to move it around to see how individual output values are created. Use this feature to explain why $y[n]$ from the previous part is zero for most $n$s.

(d) Convolve two rectangular pulses: one with amplitude 1.5 and length 4, the other with amplitude 2 and length 5. Make a sketch of the output signal, showing its length and its maximum amplitude. What is the shape of the convolution result?

(e) Record the length and maximum amplitude of the convolved result in (d) above.

(f) Filter the sinusoidal signal, $x[n] = \cos(\pi n/2)$ with a first-difference filter. Make sure that the length of the sinusoid is somewhat large, e.g., 30. Determine the period of the convolution output and sketch one period of the output signal. (Ignore the first output point, which is called "transient" in this case.)

(g) Try to determine a formula for the output signal $y[n]$ that is good for $n \geq 1$. In other words, find the amplitude $A$, phase $\phi$, and frequency of the output sinusoid. Hint: you should be able to get the phase from a peak of the output sinusoid. Note: If you look ahead to the frequency response definition, you would expect your result to match $H(e^{j\hat{\omega}})=1-e^{-j\hat{\omega}}$ at $\hat{\omega}=\pi/2$ .

Instructor Verification (separate page)

### 3.2 Filtering Images: 2-D Convolution

One-dimensional FIR filters, such as running averagers and first-difference filters, can be used to process one-dimensional signals such as speech or music. These same filters can be applied to images if we regard each row (or column) of the image as a one-dimensional signal. For example, the 50th row of an image is the

N-point sequence **`xx[50,n]`** for $1 \le n \le$ N, so we can filter this sequence with a 1-D filter using the **`conv`** or **`firfilt`** operator.

One objective of this lab section is to show how simple 2-D filtering can be accomplished with 1-D row and column filters. It might be tempting to use a **`for`** loop when writing an M-file that would filter all the rows. For a first-difference filter, this would create a new image made up of the filtered rows:

$$y_1[m,n] = x[m,n] - x[m,n-1]$$

However, this image $y_1[m,n]$ would only be filtered in the horizontal direction. Filtering the columns would require another for loop, and then you would finally have the completely filtered image:

$$y_2[m,n] = y_1[m,n] - y_1[m-1,n]$$

In this case, the image $y_2[m,n]$ has been filtered in both directions by a first-difference filter

These filtering operations involve a lot of **`conv`** calculations, so the process can be slow. Fortunately, MATLAB has a built-in function **`conv2()`** that will do this with a single call. It performs a more general filtering operation than row/column filtering, but since it can do these simple 1-D operations it will be very helpful in this lab.

(a) Load in the image **`echart.png`** (in **`Lab08-2013F-images.zip`** from **`t-square`**) with the **`imread`** command. Assign the image data to any array, say **`xim`**. We can filter all the rows of the image at once with the **`conv2()`** function. To filter the image in the horizontal direction using a first-difference filter, we form a row vector of filter coefficients and use the following MATLAB statements:

```
bdiffh = [1, -1];
yim1 = conv2(xim, bdiffh);
```

In other words, the filter coefficients in **`bdiffh`** for the first-difference filter are stored in a row vector and will cause **`conv2()`** to filter all rows in the *horizontal* direction. Display the input image **`xim`** and the output image **`yim1`** on the screen at the same time. Compare the two images and give a qualitative description of what you see.

*Note*: In some newer version of MATLAB, **`conv2`** uses data that must be in **`double`** format. If you have such a version, you may need to use something like

```
yim1 = conv2(double(xim), double(bdiffh));
```

because the image data loaded by **`imread`** may be of a different format.

(b) Now filter the **`echart.png`** image in the *vertical* direction with a first-difference filter to produce the image **`yim2`**. This is done by transposing the **`bdiffh`** filter coefficients and calling **`yim2 = conv2(xim,bdiffh','same')`** with a column vector of filter coefficients. Display the image **`yim2`** on the screen and describe in words how the output image compares to the input.

(c) Repeat (a) and (b) for the other two images in the collection, **`Lab08-2013F-images.zip`**.

Instructor Verification (separate page)

# Lab #8
# ECE-2026 Fall-2013
# Lab Sheet

For each verification, be prepared to explain your answer and respond to other related questions that the lab TA's or professors might ask. Turn this page in at the beginning of your next lab session.

Name: ______________________________________________ Date of Lab: ______________________

| Part | Observations |
|---|---|
| 3.1(a) | Convolve impulses: $\delta[n-1] * \delta[n-4] =$ |
| 3.1(b) | Rectangular Pulse through a First-difference filter: y[n] = |
| 3.1(c) | Explain why y[n] is zero for most values of n. |
| 3.1(d) | Convolve two rectangles, sketch result. State the shape of the convolution result. |
| 3.1(e) | Maximum amplitude and length of the convolved rectangles. |
| 3.1(f) | Sinusoid through a First-difference filter: sketch y[n] = |
| 3.1(g) | Amplitude and Phase of the output sinusoid: y[n] = |

Verified: _____________________________ Date/Time:______________________________

Part 3.2(a), (b) Process the images with a 2-D filter that filters in both the horizontal and vertical directions with a first difference filter. Explain how the filter changes the "image signal."

Verified: _____________________________ Date/Time:______________________________