**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |
| **Course/Section:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Semester: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |

**EE-330 Digital Signal Processing**

**Lab #5 Digital Images: A/D and D/A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. no.** | **Report Marks / 10** | **Viva Marks / 5** | **Total/15** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Lab5: Digital Images: A/D and D/A**

**Objectives**

The objective in this lab is to introduce digital images as a second useful type of signal. We will show how the A-to-D sampling and the D-to-A reconstruction processes are carried out for digital images. In particular, we will show a commonly used method of image zooming (reconstruction) that gives “poor” results.

* Familiarization with digital images.
* Working with images in Matlab.
* Sampling of images
* Familiarization with reconstruction of images

**Lab Instructions**

* This lab activity comprises of three parts: Pre-lab, Lab Exercises, and Post-Lab Viva session.
* The lab report shall be uploaded on LMS three days before next scheduled lab. The Pre-lab tasks should be completed before coming to the lab and hard copy of Pre-lab session should be deposited with teacher/lab engineer at start of the lab for necessary evaluation. Alternatively, the reports can be submitted in PDF format on LMS.
* The students should perform and demonstrate each lab task separately for step-wise evaluation (please ensure that course instructor/lab engineer has signed each step after ascertaining its functional verification)
* Only those tasks that completed during the allocated lab time will be credited to the students. Students are however encouraged to practice on their own in spare time for enhancing their skills.

**Lab Report Instructions**

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

* Lab objectives
* MATLAB codes
* Results (graphs/tables) duly commented and discussed
* Conclusion

# Digital Images: A/D and D/A

## Pre-Lab

## Digital Images

In this lab we introduce digital images as a signal type for studying the effect of sampling, aliasing and reconstruction. An image can be represented as a function *x*(*t*1*,t*2) of two continuous variables representing the horizontal (*t*2) and vertical (*t*1) coordinates of a point in space.1 For monochrome images, the signal *x*(*t*1*,t*2)would be a scalar function of the two spatial variables, but for color images the function *x*(: , :) would have to be a vector-valued function of the two variables.2 Moving images (such as TV) would add a time variable to the two spatial variables. Monochrome images are displayed using black and white and shades of gray, so they are called *gray-scale* images. In this lab we will consider only sampled gray-scale still images. A sampled gray-scale still image would be represented as a two-dimensional array of numbers of the form

*x[m,n]=x(mT1,nT2) 1≤m≤M, and 1 ≤n≤ N*

Where *T*1and *T*2 are the sample spacing in the horizontal and vertical directions. Typical values of *M* and *N* are 256 or 512 for e.g. a 512x512 image which has nearly the same resolution as a standard TV image. In MATLAB we can represent an image as a matrix, so it would consist of *M* rows and *N* columns. The matrix entry at (*m,n*)is the sample value *x*[*m,n*]—called a *pixel* (short for picture element).An important property of light images such as photographs and TV pictures is that their values are always non-negative and ﬁnite in magnitude; i.e.

0 ≤ *x*[*m,n*] ≤ *X*max*<∞*

This is because light images are formed by measuring the intensity of reﬂected or emitted light which must always be a positive ﬁnite quantity. When stored in a computer or displayed on a monitor, the values of *x*[*m,n*]have to be scaled relative to a maximum value *X*max. Usually an eight-bit integer representation is used. With 8-bit integers, the maximum value (in the computer) would be *X*max=28-1=255, and there would be 28 = 256 different gray levels for the display, from 0 to 255.

### Displaying Images

### MATLAB Function to Display Images

**you can use imshow() to display image type “help imshow” in command window of Matlab to see how to use imshow**

### Get Test Images

In order to probe your understanding of image display, do the following simple displays:

1. Load and display the 326\*426 “lighthouse” image from lighthouse.mat. This image can be ﬁnd in the MATLAB ﬁles link. The command “ load lighthouse” will put the sampled image into the a matrix as matlab variable . Use whos to check the size of created variable after loading.

(b) Use the colon operator to extract the 200th row of the “lighthouse” image, and make a plot of that row as a 1-D discrete-time signal.

lighthouse200 = lighthouse(200,:);

Observe that the range of signal values is between 0 and 255. Which values represent white and which ones black? Can you identify the region where the 200th row crosses the fence?

## LAB TASKS: Sampling, Aliasing and Reconstruction

## Lab Task 1:

### Synthesize a Test Image

In order to probe your understanding of the relationship between MATLAB matrices and image display, you can generate a synthetic image from a mathematical formula.

(a) Generate a simple test image in which all of the columns are identical by using the following

outer product: xpix = ones(256,1)\*cos(2\*pi\*(0:255)/16);

Display the image and explain the gray-scale pattern that you see. How wide are the bands in number of pixels? How can you predict that width from the formula for xpix?

(b) In the previous part, which data value in xpix is represented by white? which one by black?

(c) Explain how you would produce an image with bands that are horizontal. Give the formula that would create a 400\*400 image with 5 horizontal black bands separated by white bands. Write the MATLAB code to make this image and display it.

### Lab Task 2:

### Sampling of Images

Images that are stored in digital form on a computer have to be sampled images because they are stored in an MxN array (i.e., a matrix). The sampling rate in the two spatial dimensions was chosen at the time the image was digitized (in units of samples per inch if the original was a photograph). For example, the image might have been “sampled” by a scanner where the resolution was chosen to be 300 dpi (dots per inch).7 If we want a different sampling rate, we can simulate a *lower* sampling rate by simply throwing away samples in a periodic way. For example, if every other sample is removed, the sampling rate will be halved (in our example, the 300 dpi image would become a 150 dpi image). Usually this is called *sub-sampling* or *down-sampling*.8

*Down-sampling* throws away samples, so it will shrink the size of the image. This is what is done by the following scheme *wp = ww(1:p:end,1:p:end);* when we are down sampling by a factor of p.

(a) One potential problem with down-sampling is that aliasing might occur. This can be illustrated in a dramatic fashion with the lighthouse image.

Load the lighthouse.mat ﬁle which has the image stored in a variable called ww. When you check the size of the image, you’ll ﬁnd that it is not square. Now down sample the lighthouse image by factor 2.What is the size of the down-sampled image? Notice the aliasing in the down-sampled image, which is surprising since no new values are being created by the down-sampling process. Describe how the aliasing appears visually.9 Which parts of the image show the aliasing effects most dramatically?

#### Down-Sampling

For the lighthouse picture, down sampled by two in the preveious section:

(a) Describe how the aliasing appears visually. Compare the original to the down sampled image. Which parts of the image show the aliasing effects most dramatically?

(b) This part is challenging: explain why the aliasing happens in the lighthouse image by using a “frequency domain” explanation. (Note that the fence provides a sort of “spatial chirp” where the spatial frequency increases from left to right.)

### Lab Task 3:

### Reconstruction of Images

When an image has been sampled, we can ﬁll in the missing samples by doing interpolation. For images, this would be analogous to the sine-wave interpolation which is part of the reconstruction process in a D-to-A converter. We could use a “square pulse” or a “triangular pulse” or other pulse shapes for the reconstruction.



Figure 1: 2-D Interpolation broken down into row and column operations: the gray dots indicate repeated data values created by a zero-order hold; or, in the case of linear interpolation, they are the interpolated values.

For these reconstruction experiments, use the lighthouse image, down-sampled by a factor of 3 (similar to what you did in Section 2.3). You will have to generate this by loading in the image from lighthouse.mat to get the image which is in the array called xx. A down-sampled lighthouse image should be created and stored in the variable xx3. The objective will be to reconstruct an approximation to the original lighthouse image, which is 256x256, from the smaller down-sampled image.

(a) The simplest interpolation would be reconstruction with a square pulse which produces a “zero-order hold.” Here is a method that works for a one-dimensional signal (i.e., one row or one column of the image), assuming that we start with a row vector xr1, and the result is the row vector xr1hold.

*xr1 = (-2).ˆ(0:6);*

*L = length(xr1);*

*nn = ceil((0.999:1:4\*L)/4); %<--Round up to the integer part*

*xr1hold = xr1(nn);*

Plot the vector xr1hold to verify that it is a zero-order hold version derived from xr1. Explain what values are contained in the indexing vector nn. If xr1holdis treated as an interpolated version of xr1, then what is the *interpolation factor*? Your lab report should include an explanation for this part, but plots are optional—use them if they simplify the explanation.

(b) Now return to the down-sampled lighthouse image, and process all the rows of xx3 to ﬁll in the missing points. Use the zero-order hold idea from part (a), but do it for an interpolation factor of 3. Call the result xholdrows. Display xholdrows as an image, and compare it to the down sampled image xx3; compare the size of the images as well as their content.

(c) Now process all the columns of xholdrows to ﬁll in the missing points in each column and call the result xhold. Compare the result (xhold) to the original image lighthouse. Include your code for parts (b) and (c) in the lab report.

(d) *Linear interpolation* can be done in MATLAB using the interp1function (that’s “interp-one”).When unsure about a command, use help. Its default mode is linear interpolation, which is equivalent to using the ’\*linear’ option, but interp1can also do other types of polynomial interpolation. Here is an example on a 1-D signal:

*n1 = 0:6;*

*xr1 = (-2).ˆn1;*

*tti = 0:0.1:6; %--locations between the n1 indicesxr1*

*linear = interp1(n1,xr1,tti); %--function is INTERP-ONE*

*stem(tti,xr1linear)*

For the example above, what is the interpolation factor when converting xr1to xr1linear?

(e) In the case of the lighthouse image, you need to carry out a linear interpolation operation on both the rows and columns of the down-sampled image xx3. This requires two calls to the interp1 function, because one call will only process all the columns of a matrix.10 Name the interpolated output image xxlinear. Include your code for this part in the lab report.

(f) Compare xxlinear to the original image lighthouse. Comment on the visual appearance of the “reconstructed” image versus the original; point out differences and similarities. Can the reconstruction (i.e., zooming) process remove the aliasing effects from the down-sampled lighthouse image?

(g) Compare the quality of the linear interpolation result to the zero-order hold result. Point out regions where they differ and try to justify this difference by estimating the local frequency content. In other words, look for regions of “low-frequency” content and “high-frequency” content and see how the interpolation quality is dependent on this factor. A couple of questions to think about: Are edges low frequency or high frequency features? Are the fence posts low frequency or high frequency features? Is the background a low frequency or high frequency feature?

*Comment:* You might use MATLAB’s zooming feature to show details in small patches of the output image. However, be careful because zooming does its own interpolation, probably a zero-order hold.

### Lab Task 3

### More about Images in MATLAB (Optional)

This section11 is included for those students who might want to relate these MATLAB operations to previous experience with software such as *Photoshop.* There are many image processing functions in MATLAB. For example, try the help command:

*help images* for more information, but keep in mind that the Image Processing Toolbox may not be on your computer.

#### Zooming in Software

If you have used an image editing program such as Adobe’s *Photoshop,* you might have observed how well or how poorly image zooming (i.e., interpolation) is done. For example, if you try to blow up a JPEG ﬁle that you’ve downloaded from the web, the result is usually disappointing. Since MATLAB has the capability to read lots of different formats, you can apply the image zooming via interpolation to any photograph that you can acquire. The MATLAB function for reading JPEG images is imread( )which would be invoked as follows:

xx = imread(’foo.jpg’,’jpeg’);

Since imread( )is part of the image processing toolbox, this test can be done in the CoC computer labs, but may not be possible on your home computer.

#### Warnings

Images obtained from JPEG ﬁles might come in many different formats. Two precautions are necessary:

1. If MATLAB loads the image and stores it as 8-bit integers, then MATLAB will use an internal data type called uint8. The function show img( )cannot handle this format, but there is a conversion function called double( )that will convert the 8-bit integers to double-precision ﬂoating-point for use with ﬁltering and processing programs.

*yy = double(xx);*

You can convert back to 8-bit values with the function uint8().

2. If the image is a color photograph, then it is actually composed of three “image planes” and MATLAB will store it as a 3-D array. For example, the result of whosfor a 545x668 color image would give:



In this case, you should use MATLAB’s image display functions such as imshow( )to see the color image. Or you can convert the color image to gray-scale with the function rgb2gray( ).

**Notes:**

1. The variables t1 and t2 *do not denote time, they represent spatial dimensions.* Thus, their units would be inches or some other unit of length.
2. For example, an RGB color system needs three values at each spatial location: one for red, one for green and one for blue.
3. If you have the MATLAB Image Processing Toolbox, then the function imshow.m can be used instead.
4. If the MATLAB function imagesc.m is used to display the image, two features will be missing: (1) the color map may beincorrect because it will not default to gray, and (2) the size of the image will not be a true pixel-for-pixel rendition of the image on the computer screen.
5. The MATLAB function show\_img has an option to perform this scaling while making the image display.
6. An alternative is to use the free program called IRFANVIEW, which can do image editing and also has screen capture capability.It can be obtained from <http://www.irfanview.com/english.htm>.
7. For this example, the sampling periods would be T1 = T2 = 1/300 inches.
8. The Sampling Theorem applies to digital images, so there is a *Nyquist Rate* that depends on the maximum *spatial* frequency in the image.
9. One difficulty with showing aliasing is that we must display the pixels of the image exactly. This almost never happens because most monitors and printers will perform some sort of interpolation to adjust the size of the image to match the resolution of the device. In MATLAB we can override these size changes by using the function truesize which is part of the Image Processing Toolbox. In the *SP First Toolbox,* an equivalent function called trusize.m is provided.
10. Use a matrix transpose in between the interpolation calls. The transpose will turn rows into columns.
11. Optional means that you don’t have to include this in a lab report. This section provided in case you are curious and want to learn more on your own.