National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering and Computer Science

Faculty Member: Dr. Ahmad Salman   Dated: 20/02/2023

Semester: 6th   Section: BEE 12C

EE-330 Digital Signal Processing

Lab 3: Digital Images: A/D and D/A

Group Members

Name	Reg. No	PLO4 - CLO4		PLO5 - CLO5	PLO8 - CLO6	PLO9 - CLO7
		Viva / Quiz / Lab Performance	Analysis of data in Lab Report	Modern Tool Usage	Ethics and Safety	Individual and Teamwork
		5 Marks	5 Marks	5 Marks	5 Marks	5 Marks
Danial Ahmad	331388					
Muhammad Umer	345834					
Tariq Umar	334943					

1 Table of Contents

2 Complex Exponentials and Sinusoids

2.1 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. 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

2.2 Introduction

Digital images have become an integral part of our lives, from capturing precious moments to being used in medical imaging and scientific research. Understanding how to work with digital images is an essential skill for many fields, and this lab aims to introduce students to the fundamentals of digital image processing.

2.3 Software

MATLAB is a high-level programming language and numerical computing environment. Developed by MathWorks, it provides an interactive environment for numerical computation, visualization, and programming. MATLAB is widely used in various fields, including engineering, science, and finance, due to its capabilities for matrix and vector operations, implementation of algorithms, and creation of graphical representations of data. The objective of this lab is to provide a hands-on experience with the A-to-D sampling and the D-to-A reconstruction processes that are essential for digital image processing. We will also demonstrate a commonly used method of image zooming (reconstruction) that produces “poor” results, which will help illustrate the importance of understanding the underlying principles of digital image processing.

2.4 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

3 Lab Procedure

3.1 Getting Test Images

To probe your understanding of image display, do the following simple displays:

a) 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 array `ww/xx`. Use whos to check the size of ww 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.

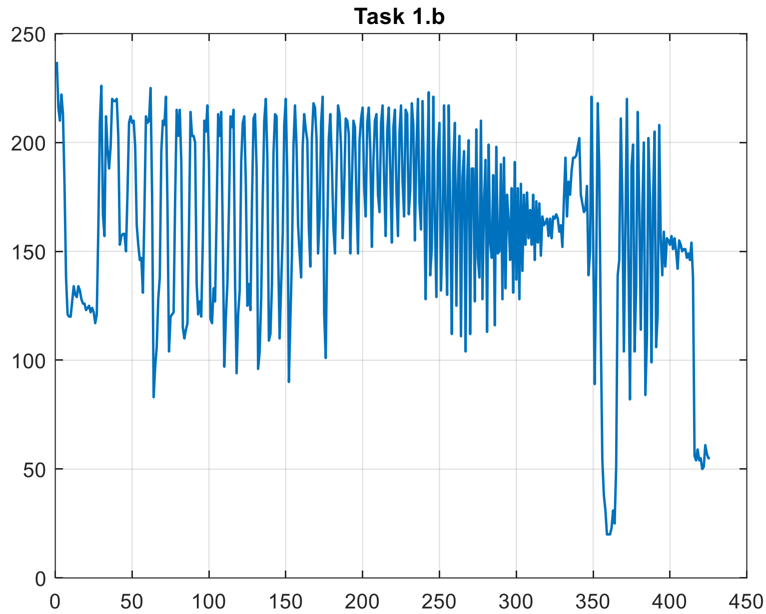
```
ww200 = ww(200, :);
```

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?

```
%% Task 1.a
load lighthouse
imshow(ww)
whos ww
title('Task 1.a')
                                              Output
Name          Size                  Bytes  Class     Attributes
ww          326x426                138876  uint8
```

Task 1.a

```
%% Task 1.b
ww200 = ww(200, :);
figure
plot(ww200, 'LineWidth', 1.15)
grid on
title('Task 1.b')
```



Answer: The values near to 255 represent the brighter (white) pixels whereas the ones near to zero represent dim (black) pixels. The 200th row crosses the fence in the interval (350, 400) of the column channel, where the highest frequency occurs.

3.2 Sampling of Images

Images that are stored in digital form on a computer must 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). 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.

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. Which parts of the image show the aliasing effects most dramatically?

```
%% Task 2
load lighthouse
p = 2;
wp = ww(1:p:end, 1:p:end);
figure
subplot(121)
imshow(ww)
title('Original Image')
subplot(122)
imshow(wp)
title('Downsampled Image')
```

Downsampled Image

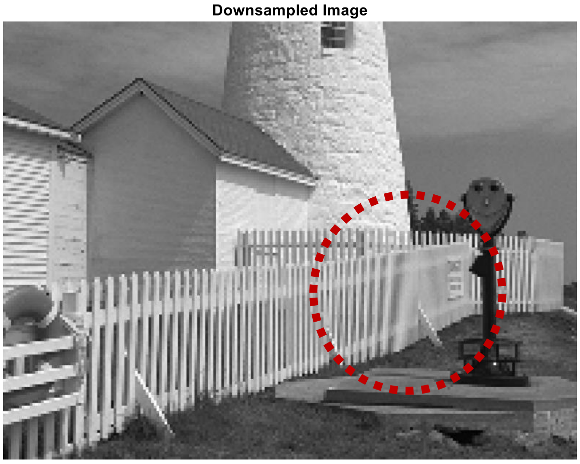
Answer: The parts of the image where white and black pixels change rapidly, in other words, the fence where the highest frequency occurs observes the highest aliasing effects.

3.2.1 Down-Sampling

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?

Original Image

Downsampled Image



3.3 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.

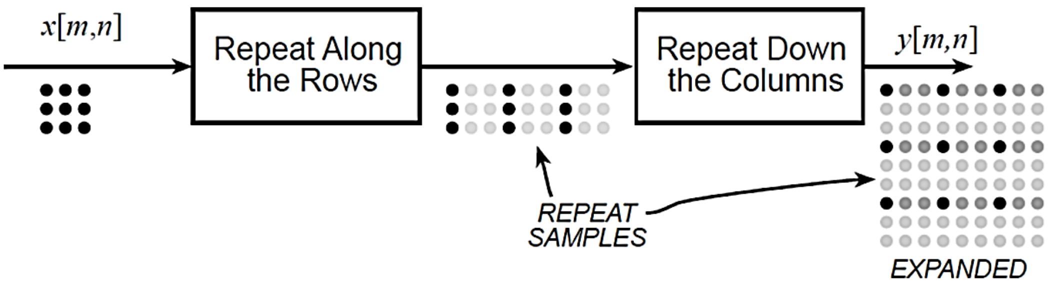


Figure 1: 2-D Interpolation broken down into row and column operations: the gray dots indicate repeated data values created by a zero-order hold; or, in the case of linear interpolation, they are the interpolated values.

For these reconstruction experiments, use the lighthouse image, down sampled by a factor of 3 (like 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.

```
%% Task 3;
load lighthouse
p = 3;
ww3 = ww(1:p:end, 1:p:end);
```

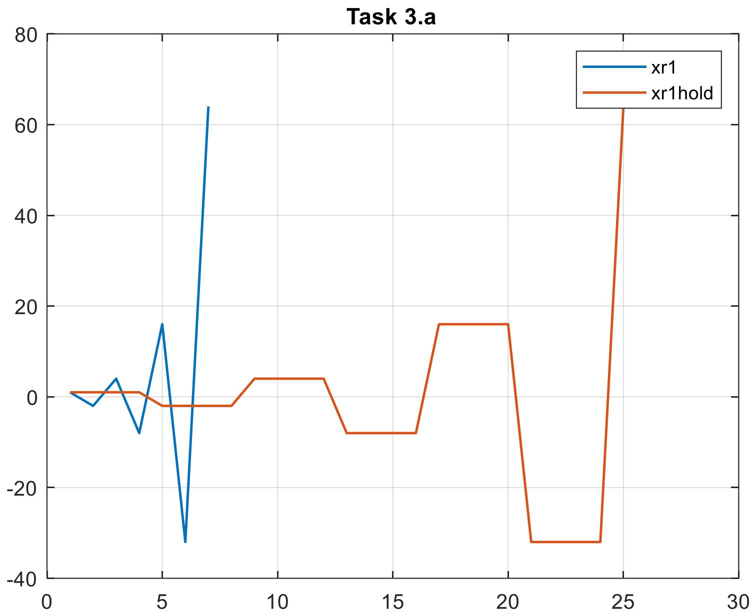
a) The simplest interpolation would be reconstruction with a square pulse which produces a "zero-order hold." Here is a method that works for a one-dimensional signal (i.e., one row or one column of the image), if we start with a row vector xr1, and the result is the row vector xr1hold.

```
xr1 = (-2).^(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 xr1hold is 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.

```
%% Task 3.a
xr1 = (-2) .^ (0:6);
L = length(xr1);
nn = ceil((0.999:1:4 * L) / 4);
xr1hold = xr1(nn);
```

```
figure
plot(xr1, 'LineWidth', 1.15)
hold on
plot(xr1hold, 'LineWidth', 1.15)
hold off
grid on
title('Task 3.a')
legend('xr1', 'xr1hold')
```

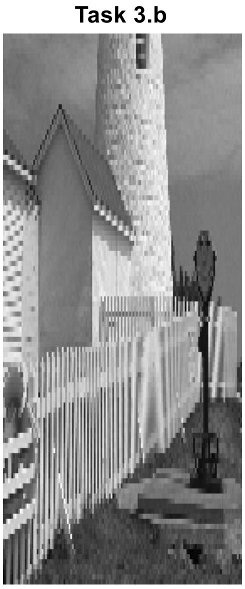


Answer: An interpolation factor of 4 is observed, as distinct point of xr1 is held on for 4 samples. The nn vectors contains the index of xr1 but repeated for 4 intervals each, i.e., the index [1 2 …] would be observed as [1 1 1 1 2 2 2 2 …] in the nn vector.

```
nn =
Columns 1 through 17
1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5
Columns 18 through 28
5 5 5 6 6 6 6 7 7 7 7
```

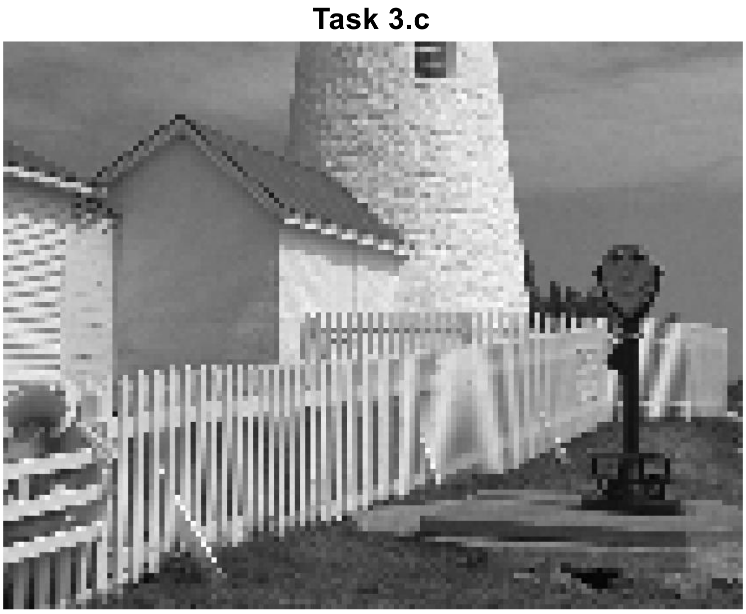
b) Now return to the down-sampled lighthouse image, and process all the rows of xx3 to fill in the missing points. Use the zero-order hold idea but do it for an interpolation factor of 3. Call the result xholdrows. Display xholdrows as an image and compare it to the down sampled image xx3; compare the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

```
%% Task 3.b
[r, c] = size(ww3);
rr = ceil((0.999:1:3 * r) / 3);
wholdrows = ww3(rr, :);
figure
imshow(wholdrows)
title('Task 3.b')
```



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.

```
%% Task 3.c
cc = ceil((0.999:1:3 * c) / 3);
whold = wholdrows(:, cc);
figure
imshow(whold)
title('Task 3.c')
```

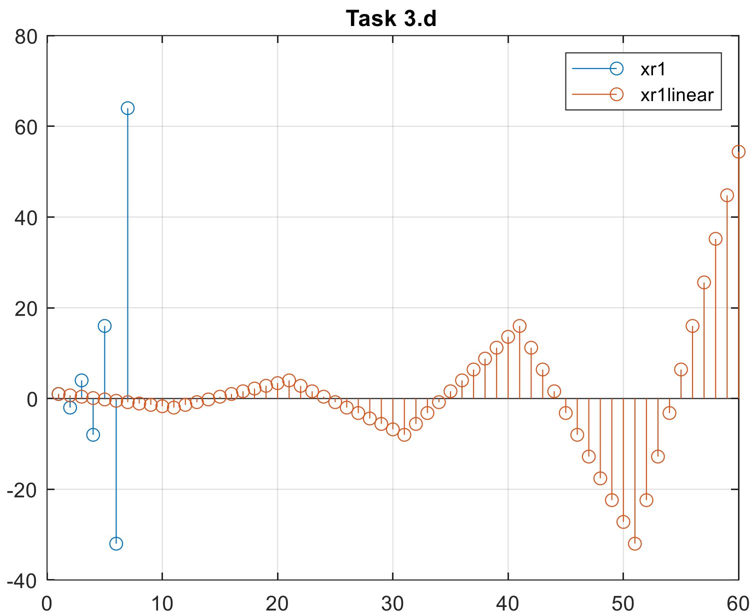


d) Linear interpolation can be done in MATLAB using the interp1function (that's "interp-one"). When unsure about a command, use help. Its default mode is linear interpolation, which is equivalent to using the'*linear' option, but interp1can also do other types of polynomial interpolation. Here is an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

```
n1 = 0:6;
xr1 = (-2).^n1;
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?

```
%% Task 3.d
n1 = 0:6;
xr1 = (-2) .^ n1;
tti = 0:1/10:6;
xr1linear = interp1(n1, xr1, tti(1:end-1));
figure
stem(xr1)
hold on
stem(xr1linear)
grid on
title('Task 3.d')
legend('xr1', 'xr1linear')
```



Answer: As the length of the interpolated signal xr1linear is 10 times that of the original signal xr1, the interpolation factor $L = \frac{output\ rate}{input\ rate}$ is 10.

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. Name the interpolated output image xxlinear. Include your code for this part in the lab report.

```
%% Task 3.e
[r, c] = size(ww3);

nr = 1:r;
```

```
nc = 1:c;
rq = 1:1/3:r;
rc = 1:1/3:c;

wrowlinear = interp1(single(ww3(nr, :)), rq);
wwlinear = interp1(single(wrowlinear(:, nc)'), rc);
wwlinear = uint8(wwlinear');

figure
imshow(wwlinear)
title('Task 3.e')
```

Task 3.e

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?

Answer: The reconstructed image appears smoother and less distorted compared to the down-sampled image. However, there is still some loss of detail due to the down-sampling process, which cannot be completely recovered through reconstruction.

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?

Answer: the quality of the interpolation result can be dependent on the local frequency content of the image. High-frequency features such as edges and fine details may require a more sophisticated interpolation method, such as linear interpolation, to produce satisfactory results. Meanwhile, low-frequency features such as smooth color transitions may be adequately captured by simpler methods such as zero-order hold.

4 Conclusion

In conclusion, this lab has provided us with a comprehensive introduction to digital image processing. Using MATLAB, we were able to demonstrate the A-to-D sampling and the D-to-A reconstruction processes for digital images. We also showed a commonly used method of image zooming (reconstruction) that produced relatively poor results, which illustrated the importance of understanding the underlying principles of digital image processing.