NAME: GIDEON ADJEI

INDEX NUMBER: 3019520

# DSP LAB 3B REPORT

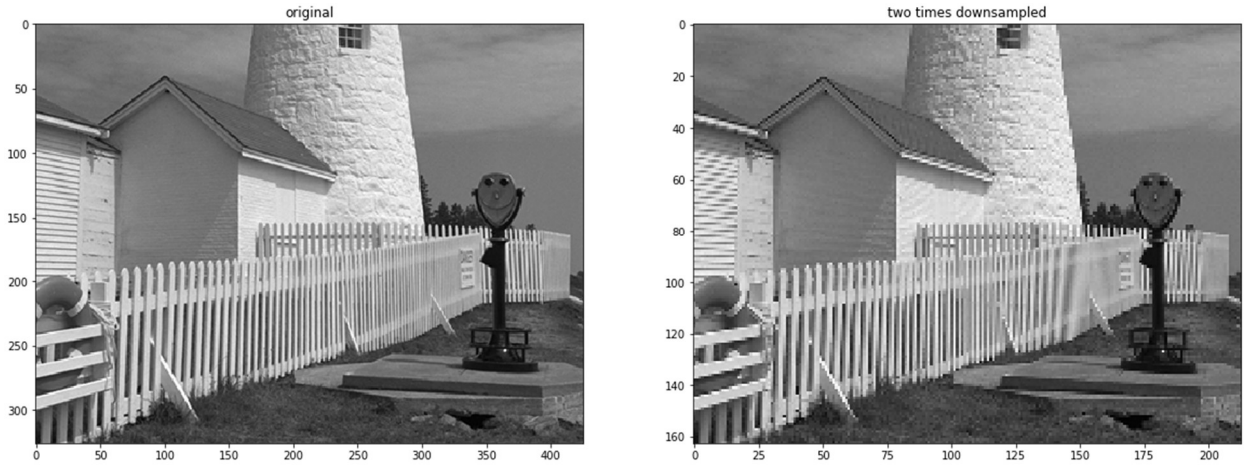
# DIGITAL IMAGES: A/D AND D/A

## 3.1 Down Sampling

In [1]:

```
%%warn up
colormap(gray(256));
%% since 400/80 = 5
xpix = ones(400, 1)*cos(2*pi*(0:399)/80);
%%use transpose to find the horizontal
show_img(xpix.', 1 ,1, gray(256));
trusize();


%% 3.1
%%myprint('Name', 'title', 'hight', 'length', '3');
load lighthouse.mat
show_img(ww, 2, 1, gray(256));
title('original');
trusize();

p = 2;
wp = ww(1:p:end,1:p:end);
show_img(wp, 3, 1, gray(256));
title('two times down sampled');
trusize();
```

Out[1]:




(a)

The aliasing can be seen clearly from the fact that, although the second image contains two times less information than the first image, they look pretty much similar. Because both images contain the same group of points, which in signal terms corresponds to the identical discrete-time sequence of points. The parts of the images with the rapid color change (from white to black) show the aliasing effects most dramatically. To be specific if take a closer look at the part, where the distance between two separate fence posts becomes too small, it can be seen that in the down-sampled image some of the fence posts merged, which is the consequence of the aliasing effect. In other words, even though the frequency of the black spots between the posts changed, there exists a number of white and black points that are contained in both images and make the second image look similar to the first, which makes the frequencies in both images' aliases.

(b)

To estimates the frequencies zoomed plots of the original pictures 199th row and the down-sampled pictures 99th row were taken. Similar to the Sampling Theorem, almost the largest frequencies from the pictures were taken. From the figure below:

Sampling them with

$$fsd = \frac{1}{35 - 29} = 0.16667$$

$$foriginal = \frac{1}{71 - 61} = 0.1$$

$$if\ fs = 0.01$$

$$\omega 1 = 2\pi \times \frac{0.17}{0.01} = 34\pi$$

$$\omega 2 = 2\pi \times \frac{0.1}{0.01} = 20\pi$$

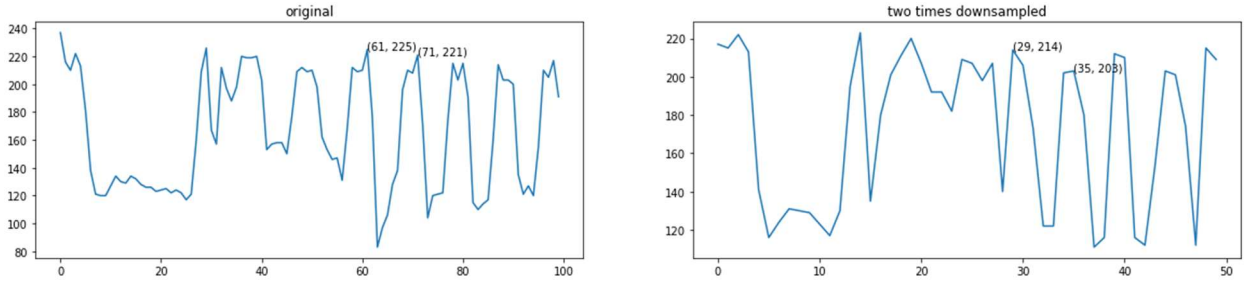
$$\omega 1 = \omega 2 + 14\pi \rightarrow aliasing$$

3.2

In [2]

```
p = 3;

load lighthouse.mat

%%down sample by 3
ww3 = ww(1:p:end,1:p:end);

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
plot(xr1);
title('original');
plot(xr1hold);
title('interpolated')
```

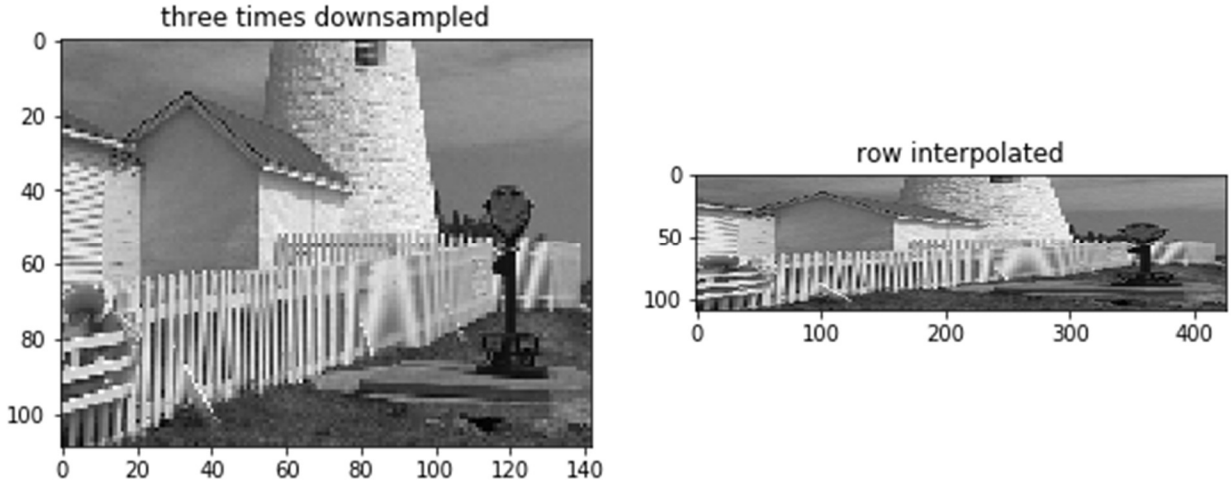
Output [2]:




nn contains xr1's indices incremented by 1 and each index is repeated 4 times. Since sampling rate became 4 times faster, interpolation factor is 4. Computation:

$$Interpolation\ factorm = \frac{output\ sample\ rate}{input\ sample\ rate}$$

```
% Load or generate your image 'xx' here
% xx = imread('path_to_your_image.png'); % Example of loading an image
xx = randi([0, 255], 256, 256); % Example of a random image for testing

threshold_values = [16, 32, 128, 224];
figure;
set(gcf, 'Position', [100, 100, 1500, 1000]); % Set figure size in pixels

for j = 1:length(threshold_values)
    i = threshold_values(j);
    k = (i > 32) + 1; % k is 1 if i <= 32, 2 if i > 32
    subplot(2, 2, (k-1)*2 + mod(j-1, 2) + 1);
    imshow(threshold(xx, i), []);
    title(['Threshold = ', num2str(i)]);
end




function thresholded = threshold(im, threshold)
    thresholded = uint8((im > threshold) * 255);
end

% Load or generate your image 'xx' here
% xx = imread('path_to_your_image.png'); % Example of loading an image
xx = randi([0, 255], 256, 256); % Example of a random image for testing

threshold_values = [16, 32, 128, 224];
figure;
set(gcf, 'Position', [100, 100, 1500, 1000]); % Set figure size in pixels

for j = 1:length(threshold_values)
    subplot(2, 2, j);
    imshow(threshold(xx, threshold_values(j)), []);
    title(['Threshold = ', num2str(threshold_values(j))]);
end
```




<Figure size 432x288 with 0 Axes>

From the comparison of both images, it can be seen that height (number of rows) of the xholdrows remained the same, but its width (number of columns) returned to the original size. Objects in the image became wider and the intervals between fence posts became more visible, but not clearer. The difference can be seen in the parts where the color changes rapidly (fence posts, window), but almost the same in the parts where the color remains almost the same (background, side of the house).



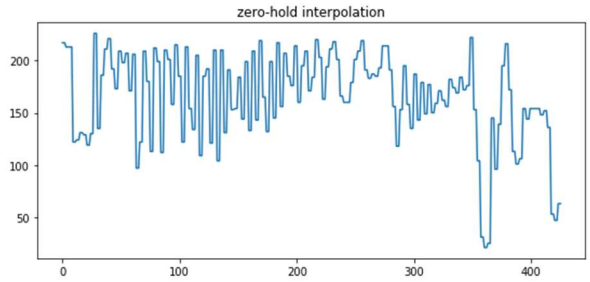
One obvious thing is that the sizes of both images are the same (+1 or -1 pixel difference). The quality of the image dropped, but the main objects can be seen. The parts of the image that did not have rapid color change (low frequency) such as the background and the ground did not change a lot and in these parts, the differences between the images are not obvious. However, in the parts where there is a rapid color change (high frequency) such as the fence, there can be seen clear distortions (black parts became white and vice versa).

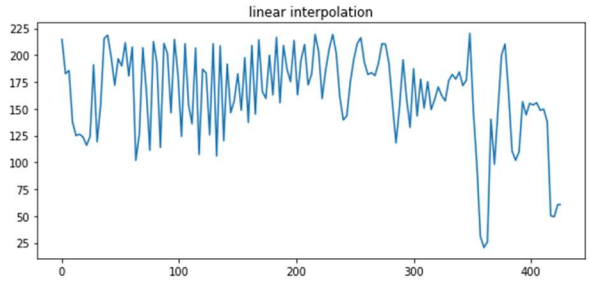


(f)

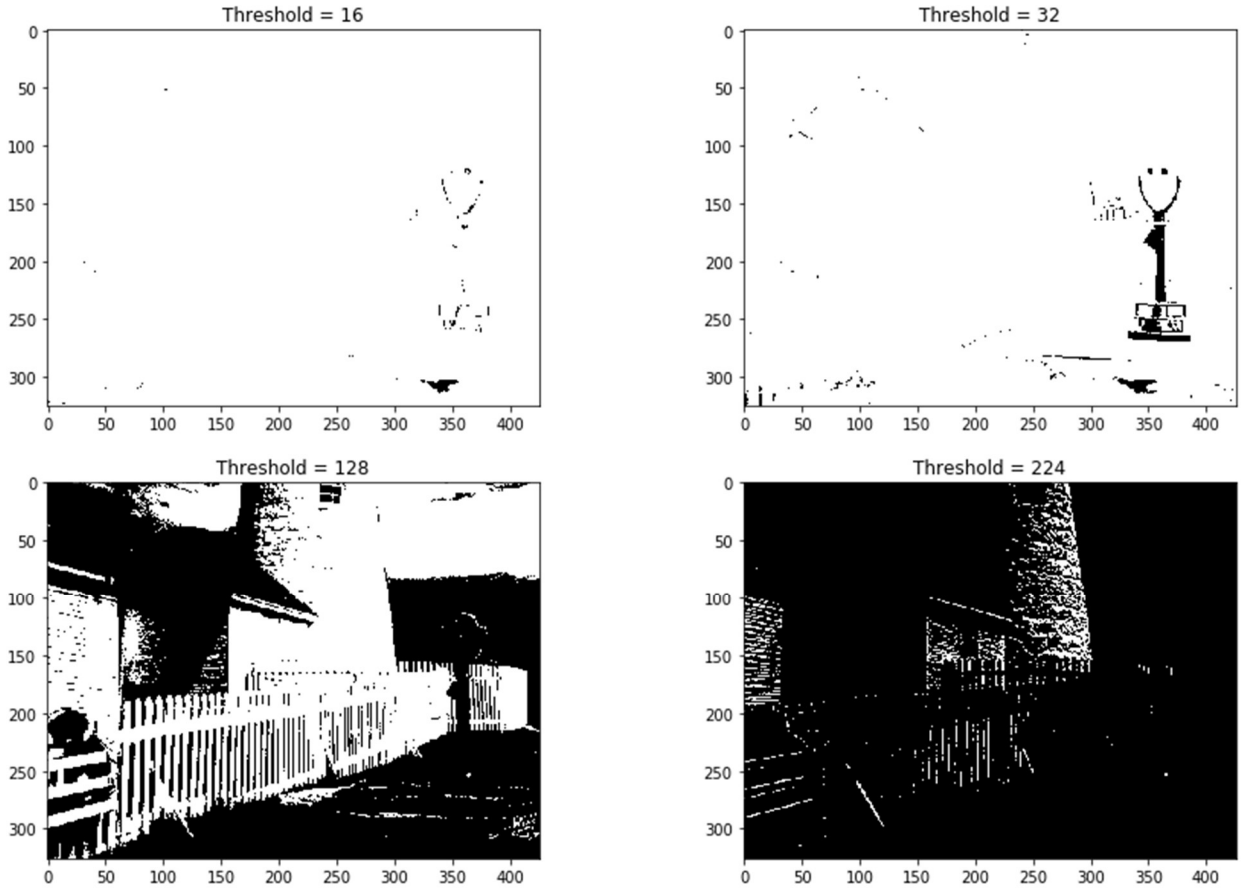
The linearly “reconstructed” image looks blurred, compared to this the original image (obviously) is more detailed, but still the main objects can be easily differentiated. The parts that are made of the different shades of the same color look almost identical, but the parts that include both white and black

colors look different. The reconstruction process most probably will remove the aliasing effects, because there still be the sequence of points that will be included in both images and because reconstruction does not remove these points unless it removes principal alias.



```
function thresholded = threshold(im, threshold)
    thresholded = uint8((im > threshold) * 255);
end

% Load or generate your image 'xx' here
% xx = imread('path_to_your_image.png'); % Example of loading an
image
xx = randi([0, 255], 256, 256); % Example of a random image for
testing

threshold_values = [16, 32, 128, 224];
figure;
set(gcf, 'Position', [100, 100, 1500, 1000]); % Set figure size in
pixels

for j = 1:length(threshold_values)
    i = threshold_values(j);
    k = (i > 32) + 1; % k is 1 if i <= 32, 2 if i > 32
    subplot(2, 2, (k-1)*2 + mod(j-1, 2) + 1);
    imshow(threshold(xx, i), []);
    title(['Threshold = ', num2str(i)]);
end
```




From the image comparison above, it can be seen that the function changes the level of brightness of the image. As we increase the threshold the image becomes darker. It varies from very bright to very dark and there is a mathematical explanation for these results. For the small values of the threshold (im>threshold) returns 1 for almost all the values in the image's matrix and transforming it to the matrix with a lot of 255 values which corresponds to white color, but as we increase threshold (im>threshold) starts to return 0 for more and more values and transforming the matrix to one with a lot of 0 values which corresponds to black color.