EE 112: Final Project

Spring 2017

Instructor: Professor Santacruz

Umar Sohi
student#: LEMME SMASH, PLZ

Alex Yee
student#: no ron go find becky

Anthony Watson
student#: who the hell cares

Qinglun Huang
student#: truckasaurusrex

May 13, 2017

Contents

List of Figures

You can add footnotes that look like this. Check out below.[1]

Pre-Lab Section

- images are represented as matrices where each elem is a pixel
- images are MxN matrices where $x[m, n]$ represents our matrix
- we use $x[m, n] = x(mT_1, nT_2)$ for $1 \leq m \leq M$ and $1 \leq n \leq N$ to select values in the image.
- we will restrict our project to monochrome images
- the value of the pixel are intensity values. This means that all pixel values $p_i = n$ for $n \in [0, 255] \subset \mathbb{Z}$.

Section 1.4: Getting Test Images

Basic code used on (5/4)

```
% loading in a saved .mat data file
load lighthouse

% check what variable name and info the original author saved lighthouse as
whos    % outputs ww and more info on its size etc

% if we used "who" instead of "whos" it will just list all active var names
who

% for indexing ":" usually means all indices if by itself. Syntax for range of
% vals.
% creating a row vector holding all the pixels in the 200th row
ww200 = ww(200,:);

% vals 0 -> Black and 255 -> white (insert Umar's Physics Explaination here)
% above => smaller values are darker and larger values are lighter

% finding where the 200th row crosses the fence
% the following will plot the connected plot of the sampled pixel values
plot(ww200)
```

Looking at the 200th row plot we can see that early on there is a sudden drop in intensity values. If one looks at the corresponding x-axis value for this part of the plot, one will see the darker color life-saver. This

[1] i'm a footnote

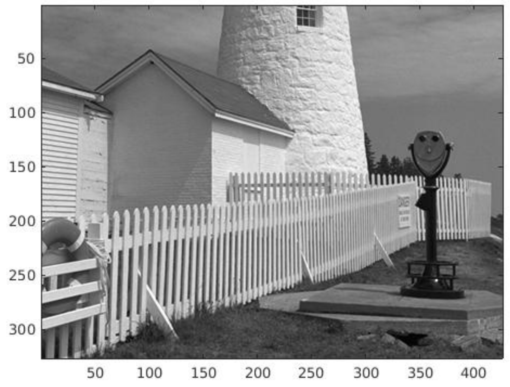


Figure 1: *LighthouseOriginalImage*

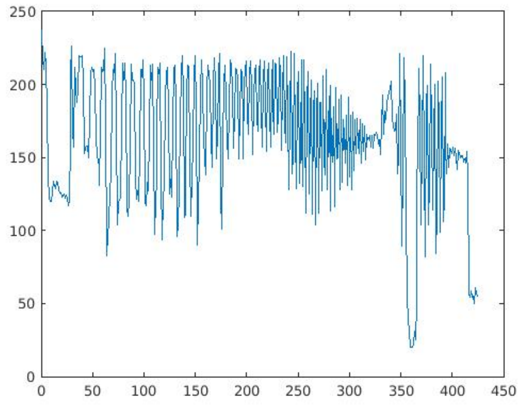


Figure 2: $200 th Row Plot$

life-saver's darker color corresponds to the 200th row going from an initial bright background color to the dark grey life-saver color.

The oscillatory values of the plot from $n \simeq 50$ to $n \simeq 300$ indicates the fence's spaced set up in the image. The fence has gaps that show the darker background adjacent to the bright fence. This corresponds to the 200th row oscillating between low and high values in the plot.

At $n \simeq 325$ the intensity values of the image seem to stablize. This is due to orientation of the fence in the image. The fence is at an angle to the viewer. As it gets farther away, we become unable to distinguish the gaps in the fence. Hence, the value stablizes about a common value until $n \simeq 350$.

At $n \simeq 350$, there is a large dip in the intensity value. This is because the 200th row is detecting the dark stationary binoculars on the right side of the image. Once, the column values n have passed the region with the binocular it starts to oscillate again due to the fence. Again, it will stablize because of the orientation of the fence at such a far distance.

At $n \simeq 412$ the intensity value drops suddenly. This is where the 200th row passes the fence. We are no longer seeing the white fence and only the dark background shows though. Thus, we get the lower intensity values with $x \simeq 60$.

Warm-up Section

Synthesize a Test Image

```
% outer product to create a matrix of cosine values
xpix = ones(256,1) * cos(2 * pi * (0:255) / 16);
```

$$xpix = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} \cos(\frac{2\pi \cdot 0}{16}) \cdots \cos(\frac{2\pi \cdot k}{16}) \cdots \cos(\frac{2\pi \cdot 255}{16}) \end{bmatrix}$$

$$xpix = \begin{bmatrix} \cos(\frac{2\pi \cdot 0}{16}) & \cdots & \cos(\frac{2\pi \cdot k}{16}) & \cdots & \cos(\frac{2\pi \cdot 255}{16}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \cos(\frac{2\pi \cdot 0}{16}) & \cdots & \cos(\frac{2\pi \cdot k}{16}) & \cdots & \cos(\frac{2\pi \cdot 255}{16}) \end{bmatrix}$$

The outer product essentially creates a matrix, that is 256 copies of the cosine vector stacked on top of each other. It is because of this structure that the resulting image is a bunch of vericle bands.

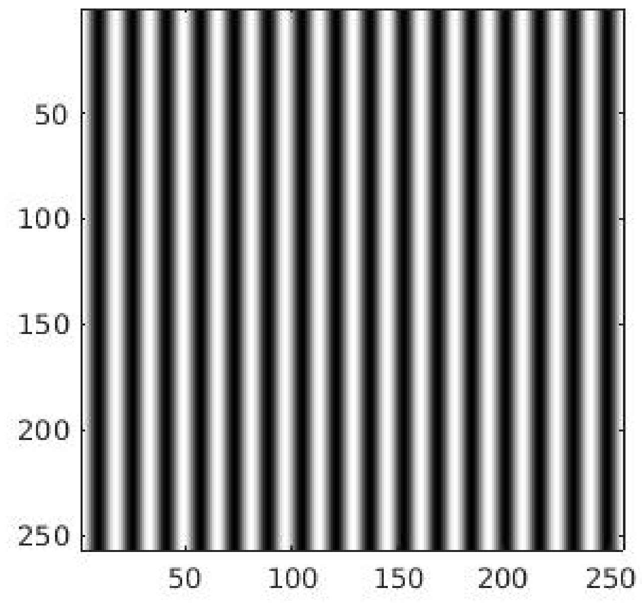


Figure 3: *outer product of 256x1 vector and a 1x256 cosine valued vector*

We can predict the width of the bands based upon cosine's period. In one period we should see one black and one white band. This is because a sinusoidal periodic function, like cosine, will transition through its lowest to highest then back to its lowest value. We find the period and band widths as follows:

$$\begin{aligned}\cos(\omega k) &= \cos(2\pi f k) \\ &= \cos\left(\frac{2\pi k}{T}\right)\end{aligned}$$

$$\cos\left(\frac{2\pi k}{T}\right) = \cos\left(\frac{2\pi k}{16}\right)$$
$$\Rightarrow T = 16px$$
$$Bandwidth = \frac{1}{2}T = 8px$$

Using this formula we can predict the value of the bandwidth.

We can examine what values create the dark and light bands of the cosine plot:

```
% look at one row of the cosine matrix
% we do this since each row is the same. Fix the row index. I chose 1
% arbitrarily.
twne = xpix(1,1:17);
plot(twne)
show_img(xpix) % zoom in to see the pixel colors

% twne will have 1 period of values, plus one extra value to verify that we
% indeed cycle back to the starting value.
```

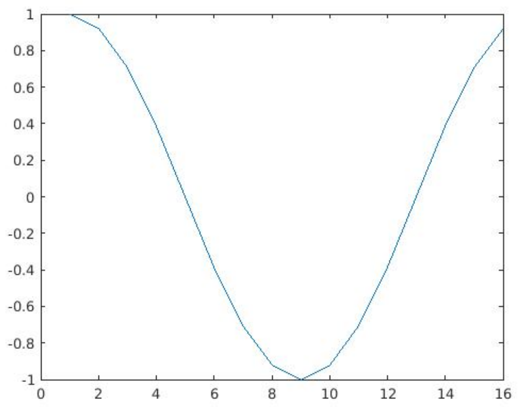
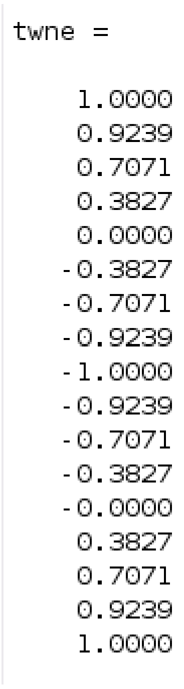


Figure 4: $CosinePlot$



```
twne =

    1.0000
    0.9239
    0.7071
    0.3827
    0.0000
   -0.3827
   -0.7071
   -0.9239
   -1.0000
   -0.9239
   -0.7071
   -0.3827
   -0.0000
    0.3827
    0.7071
    0.9239
    1.0000
```

Figure 5: $CosineValues$

Looking at the graph we want to find out what the values +1 and -1 look like. We know that at 17 and 9 we should have these values respectively.

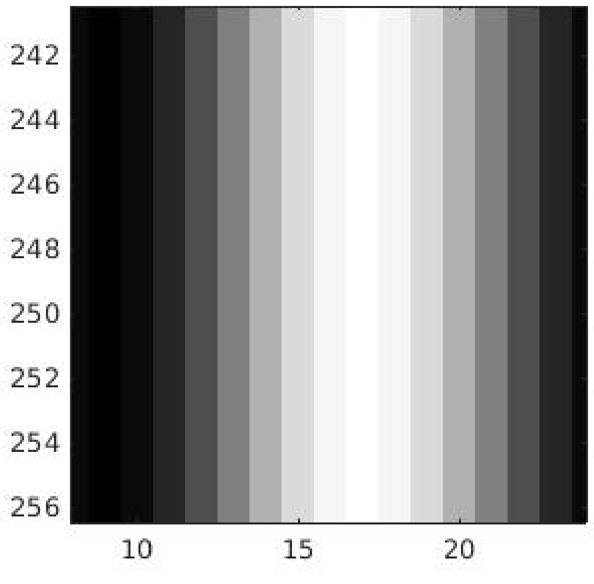


Figure 6: Zoomed In View of Cosine Image

After inspecting the zoomed in image of the cosine matrix, we can see that $9 = black$ and $17 = white$. These column index values correspond to $-1 = black$ and $+1 = white$.

```
% since we want 5 black bands w/ white band separations => 5 periods
% working with pixels => 400x400 matrix = 400x400 image
% 400 / 80 = 5 => use 80 as your period to get 5 periods in a 400px span
ypix = ones(400,1) * cos(2 * pi * (0:399) / 80);
ypix = ypix'; % transpose of vert lines gives horizontal lines
show_img(ypix);
```

The transpose will rotate the vericle lines 90 degrees giving us horizontal lines.

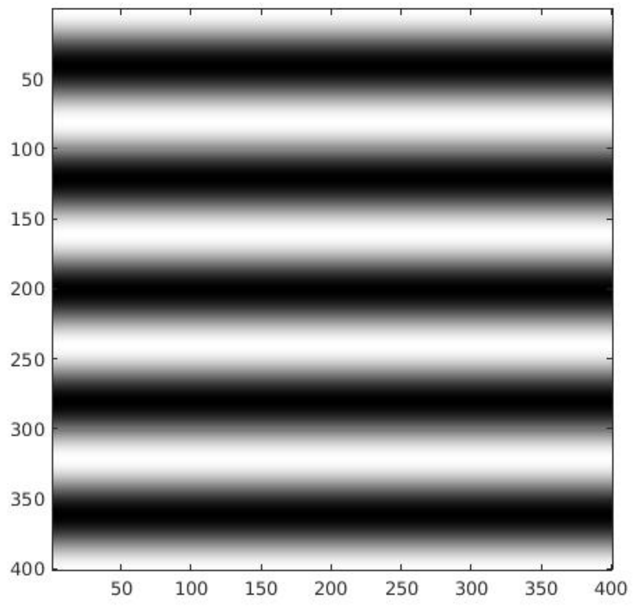


Figure 7: Horizontal Cosine Plot

```
% downsampling
wp = ww(1:2:end,1:2:end);
show_img(wp)
```