To: Prof. Ross Snider

From: Jeff Meirhofer

Regarding: Lab #9 – Sampling and Zooming of Images

Date: March 26, 2009

**Summary:**

The purpose of this lab is to explore the concepts of sampling and zooming images. In order to get high-quality zoomed images, we must use interpolation. This interpolation can be implemented by using FIR filters. The lab will help us further visualize how these actions are performed on images.

**Sampling and Zooming**

**3 a)** First, we will do some sub-sampling or down-sampling. This is when we sample an image at a rate lower than it was originally sampled at. We are given a function, “imsample.m”. It takes an input parameter, P, and an image as its inputs. It then sub-samples the image, taking the data from every Pth pixel. The new image will contain only these pixels and zeros where the other pixels used to be.

**3 b)** We load the “zone” image and down-sample it by 2, but leave the pixels in the same spatial locations as before. When we show this image we can see that the images that are “thrown away” are set to zero and are therefore black. We can also see that the image is still the same size. The result is shown in Figure 1.



Figure 1. Original zone image and sampled zone image with pixels thrown away

**3 c)** Now we will do true down-sampling of the image. To do this, we create a new image that consists of only the pixels that have been sampled, and not the ones that have been set to zero. This produces a smaller image. The function “xp = xx(1:p:M, 1:p:N)” creates the new image(xp) from the sampled image xx. The image comparisons are in Figures 2 and 3.



Figure 2. Original and down-sampled image



Figure 3. Original image of “lenna” and down-sampled version

We can see that obvious aliasing occurs in Figure 2. This makes the image appear much different than the original. The down-sampled image appears to have repeated patterns at places where the original image does not. This shows the effects of aliasing.When we perform the same process on the “Lenna” image, we do not see these same effects. The image appears to be relatively unchanged. The main reason for this is because the Lenna image does not have pixels that are periodically repeating along its horizontal and vertical axis, and the zone image does (or at least in Lenna they are not as blatant as in zone).

**3.1)** Now we will use interpolation to construct the missing samples of an image that is sub-sampled by 4.

**3.1 a)** First, we will try using a square wave for our interpolation. This is known as a zero-order hold. We will down-sample the image, and then implement the zero-order hold horizontally (on the rows). The result still has gaps (samples thrown away) running horizontally through it. It is in Figure 4.

**3.1 b)** Now, we will implement the same process on the columns. This fills in the gaps. When compared to the original image, the new image looks pixilated and blurry.

**3.1 c)** In this part, we do a linear interpolation of the samples. We use an FIR filter to do the interpolation. The coefficients make a triangular pulse.

**3.1 d)** When we compare the linearly interpolated image to the others, it looks much better than the square image. However, it still looks much worse than the original image.

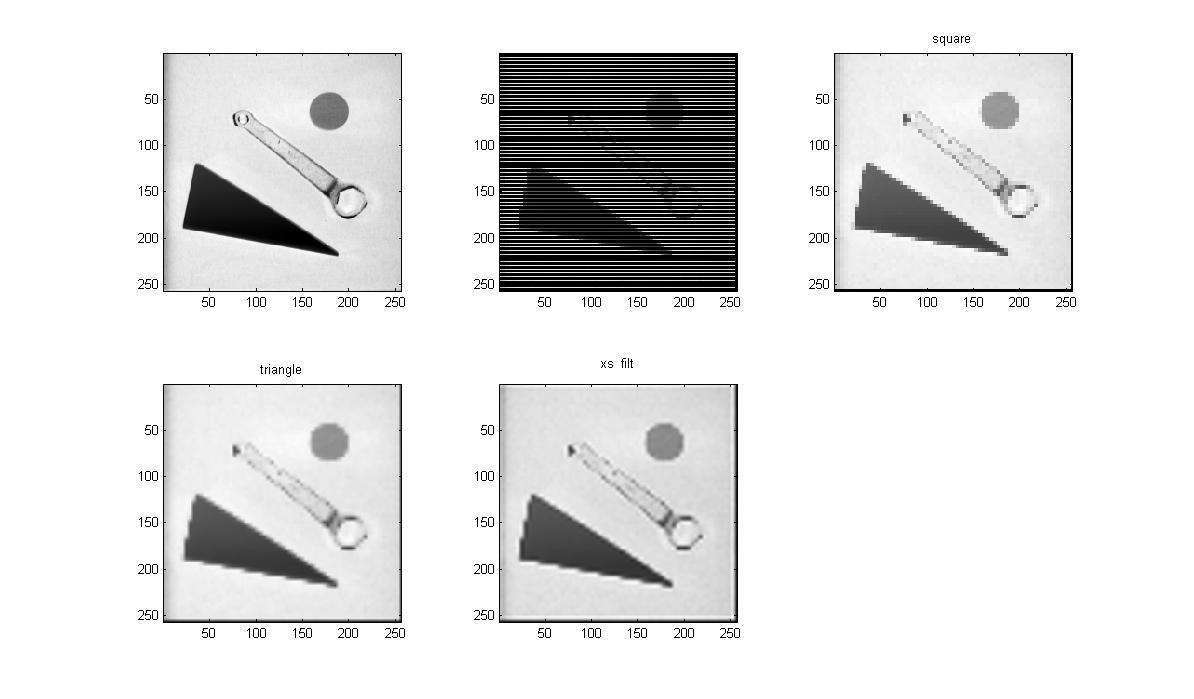


Figure 4. Reconstructed images

**3.1 e)** The frequency response of the FIR filter used in the triangle reconstruction is shown below.



Figure 5. Frequency response of triangle FIR filter

You can see that the system looks a lot like a lowpass filter

**3.1 f)** Now, we filter the image with a 23-point lowpass filter. The coefficients of this filter form the shape of a modified sinc function. They are given by , where for k = 0,1,2,…,22 . We have to use L’Hopital’s rule to compute the value at k=11, and we come up with 1. We can see in Figure 4 that this method does a better job of reconstruction than the triangle method. It is still blurry compared to the original image, but it does a very good job of reconstruction. The frequency response of this filter is shown below.



Figure 6. Frequency response of 23-point FIR filter

You can see that this function is very similar to the sinc function.

**3.2)** In this section, we will look at ways to zoom in on an area of an image. We will use a 50 X 50 section from the Lenna image, and zoom in to make it 200 X 200.

**3.2 a)** First, we will do the zooming by simply taking each pixel value and repeating it four times in each direction. This method is shown in Figure 7.

**3.2 b)** Next, we will make the 50 X 50 image 200 X 200 by inserting zeros in between the real pixels. This is used later for doing interpolations.

**3.3 c)** Finally, we filter the image from “b” with our triangle filter and our sinc filter from before. The results of each of the three zooming operations are shown in Figure 7.



Figure 7. A 50 X 50 image, zoomed to 200 X 200 by three different methods.

**3.3 d)** The first method does not work very well. It looks pixilated and it not smooth. The triangle method works fairly well. It can show edges and a little bit of detail. The sinc method works the best of all. The sinc method produces smoother edges than either of the other two, which makes it look more natural. It also does a better job of showing the edges. This is because the filter that implements the triangle experiences its largest gain at nearly DC only, while the sinc filter includes a large range of frequencies that experience its highest gain (approximately). Or said in a different way, the sinc filter’s frequency response rises from negative to positive quicker than the triangle’s. This allows for the DC content to be less overpowering.

**3.3 e)** N/A

**Appendix**

**2.1**

close all

clear all

%2

%2.a

xss = zeros(1,19);

samp = [1 3 -2 4 2 -1 -3];

xss(1:3:19) = samp;

%2.b

coeffs = [1/3 2/3 1 2/3 1/3];

output = firfilt(xss,coeffs);

figure(1)

subplot(2,1,1),plot(1:19,xss),title('xss'),axis([0 25 -4 4])

subplot(2,1,2),plot(1:length(output),output),title('output'),axis([0 25 -4 4])

**3**

close all

clear all

%3

%3.b

load zone

xs = imsample(zone,2);

figure(1)

subplot(1,2,1),show\_img(zone,0)

subplot(1,2,2),show\_img(xs,0)

%3.c

p = 2;

[M,N] = size(zone);

xp = zone(1:p:M,1:p:N);

figure(2)

subplot(1,2,1),show\_img(zone,0)

subplot(1,2,2),show\_img(xp,0)

load lenna

p = 2;

[M,N] = size(xx);

xp = xx(1:p:M,1:p:N);

figure(3)

subplot(1,2,1),show\_img(xx,0)

subplot(1,2,2),show\_img(xp,0)

**3.1**

close all

clear all

%3.1

%3.1.a

load tools

xs = imsample(xx,4);

bs = ones(1,4);

yhold = filter2(bs,xs);

figure(1)

subplot(2,3,1),show\_img(xx,0)

%subplot(2,2,2),show\_img(xs,0)

subplot(2,3,2),show\_img(yhold,0)

%3.1.b

yhold = filter2(bs(:),yhold);

subplot(2,3,3),show\_img(yhold,0),title('square')

%3.1.c

coeffs = [1/4 1/2 3/4 1 3/4 1/2 1/4];

ylin = filter2(coeffs(:),filter2(coeffs,xs));

subplot(2,3,4),show\_img(ylin,0),title('triangle')

%3.1.e

w = -pi:pi/100:pi;

figure(2),freqz(coeffs, 1, w)

%3.1.f

k = 0:22;

wk = 0.54-0.46\*cos(2\*pi\*k/22);

bk = sin(pi\*(k-11)/4)./(pi\*(k-11)/4).\*wk;

bk(12) = 1;

xs\_filt = filter2(bk(:),filter2(bk,xs));

figure(1)

subplot(2,3,5),show\_img(xs\_filt,0),title('xs filt')

figure(3),freqz(bk,1,w)

**3.2**

close all

clear all

%3.2

%3.2.a

load lenna

xz = xx(101:150,101:150);

xz2 = zoom\_img(xz);

figure(1)

subplot(2,2,1),show\_img(xz,0)

subplot(2,2,2),show\_img(xz2,0)

%3.2.b

L = length(xz);

yy = zeros(4\*L,4\*L);

yy(4:4:4\*L, 4:4:4\*L) = xz;

% subplot(2,2,2),show\_img(yy, 0)

%3.2.c

coeffs = [1/4 1/2 3/4 1 3/4 1/2 1/4];

ylin = filter2(coeffs(:),filter2(coeffs,yy));

subplot(2,2,3),show\_img(ylin, 0),title('Linear Interpolation')

k = 0:22;

wk = 0.54-0.46\*cos(2\*pi\*k/22);

bk = sin(pi\*(k-11)/4)./(pi\*(k-11)/4).\*wk;

bk(12) = 1;

ysinc = filter2(bk(:),filter2(bk,yy));

subplot(2,2,4),show\_img(ysinc, 0),title('Sinc Interpolation')

**imsample.m**

function [ yy ] = imsample( xx, P )

%IMSAMPLE Function for sub-sampling an image

% usage: yy = imsample(xx, P)

% xx : input image to be sampled

% P : sub-sampling period (an integer like 2,3,etc)

% yy : output image

[M,N] = size(xx);

S = zeros(M,N);

S(1:P:M,1:P:N) = ones(length(1:P:M), length(1:P:N));

yy = xx .\* S;