**Hints and tips for first exercise**

Here are some example solutions to the first two problems from Lab #1, with some hints on the matlab commands needed for the other problems. Use the help command to learn more about these commands.

Problem 1 - Raster displays.

Sample solution code:

f=fopen('raster');

[pix,count]=fscanf(f,'%d',[18,3]); pix=pix'; pix

imagesc(pix); axis('equal'); axis('image');

Problem 2 - Resolution.

Sample solution code: pix=zeros(512,512); circle=ones(33,33);

for i=1:33, for j=1:33,

if(sqrt((i-17)^2+(j-17)^2))> 16 circle(i,j)=0; end; end; end;

for i=1:33, for j=1:33,

pix(i+200,j+200)=circle(i,j); pix(i+232,j+200)=circle(i,j); pix(i+200,j+300)=circle(i,j); pix(i+248,j+300)=circle(i,j); pix(i+200,j+400)=circle(i,j); pix(i+264,j+400)=circle(i,j); end; end;

h=imagesc(pix'); axis ([0 512 0 512]); axis image;

## The filtering can be done with lines like the following: ##

filter=ones(33,33); pix2=filter2(filter,pix,'same');

h=imagesc(pix2'); axis ([0 512 0 512]); axis image;

Problem 3 - Resolution in an image.

Hints:

You will need to read in a file now in binary format. That is, the data are in integer 8-bit format on the disk, rather than as text or ascii representations. The read command for raw data such as these is:

f=fopen('campusdrive.raw'); a=fread(f,[580 inf],'uint8');

where, once again, f is a file pointer and a contains the input array. “fread” replaces fscanf, and we do not use a format description such as %d. The [580 inf] parameter tells the program to place the data in an array of line length 580 and the inf argument says to read until it runs out of data. ‘uint8’ means the data are stored on disk as unsigned integers of length 8 bits.

You can then filter and display as in the previous problem.

Problem 4 - Quantization.

Hints

In this problem you need to select only the high order bits in the pixels. We can do this with logical and’s using the matlab bitand command:

b=bitand(b,128+64);

which, for example, logically and’s every element of matrix b with a number consisting of 1’s in the 128 and 64 places of a binary number, thus saving the top two bits of an eight-bit number. Open the binary file as in the previous problem, and display as usual using imagesc and axis.

Problem 5 - An unknown image.

Hints: Try this one yourself!