Biological Instrumentation and Measurement, Fall 2008

Department of Biological Engineering
Massachusetts Institute of Technology

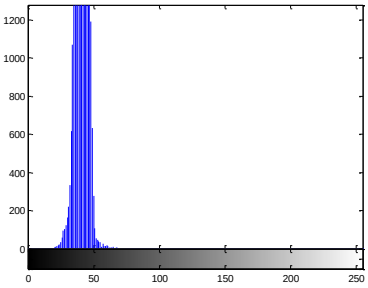
Problem Set #8 Solution

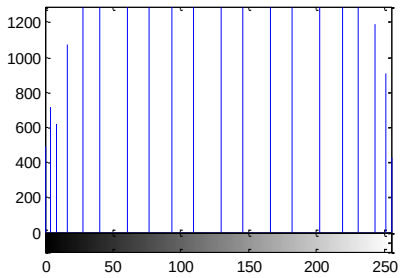
Due: Tuesday, November 25

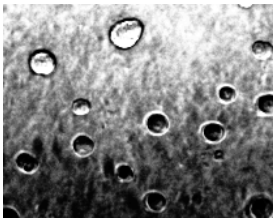
1. **Contrast and histogram** Optimize contrast for visualization of the following microscope image (dark_cells.tif) taken at too low a light level.

(a) Apply histogram equalization to this image.
(b) Plot the histograms of the image before and after equalization.
(c) Display the image after equalization.

(a) Get histogram using imhist.
A=rgb2gray(dark_cells); %first convert from rgb format to gray scale
Imhist(A)







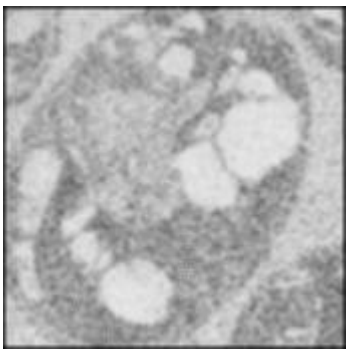
(b, c) Then do histogram equalization with histeq.
B=histeq(A);
Imshow(B);
Imhist(B);

2. **Reduce image noise by low pass filter pre-processing** Create a 3x3 and a 5x5 low pass filter. Apply these filters to remove noise from the low light level cell image (noisy_cell.jpg). Try a 5x5 median filter also?

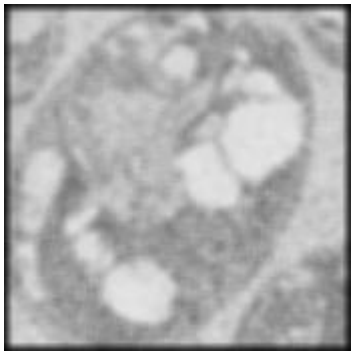
First make kernels: A=1/9*[1 1 1; 1 1 1; 1 1 1]; A2=1/25*[1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1; 1 1 1 1 1]

B=rgb2gray(noisy_cell); C=conv2(B,A); imshow(C, [0 180]); D=conv2(B,A2); imshow(D, [0 180]);

3x3



5x5



3. **Reduce image noise by median filter pre-processing**

Generating a long integration image using a CCD camera sometimes produce large intensity spikes due to cosmic rays. Try to remove the noise from this image (spiky1.tif). Can you develop a different method to remove the spikes if you have two pictures taken one after the other (spike1.tif, spike2.tif)?

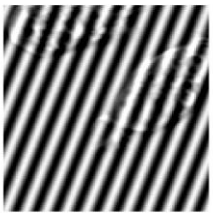
A=rgb2gray(spiky1); C=medfilt2(A,[12 ,12]); imshow(C);

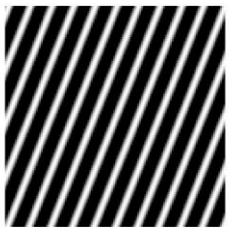
If we assume the noises from spike1 and spike2 are random and large, we identify pixels without noise by requiring the intensities of corresponding pixels to be similar. For a pixel with noise in one picture, that pixel is much brighter than the pixel in the other picture. We can use the intensity information from the lower intensity, non-noisy picture to fill in the missing information. The advantage of this approach is that there is no loss of resolution.

```
A=rgb2gray(spiky1); B=rgb2gray(spiky2);
dev = std(std(double(A)));
j=1;
for i=1 : 900
  for j=1 : 1200
    if abs(A(i,j)-B(i,j))<dev
            D(i,j)=A(i,j);
        elseif A(i,j) < B(i,j)
            D(i,j) = A(i,j);
        else D(i,j)= B(i,j);
        end;
    end;
end;
imshow(D)
```

4. **Removing periodic noise by Fourier filtering**

(a) Periodic noises often occur in biological imaging (highnoise and lownoise). Can you remove the noise if you know the noise characteristics (noise)? The data files are matrix stored in fftfiles.m.





(b) Consider the case that you do not know the noise characteristics (newnoise). Can you clean up this image?

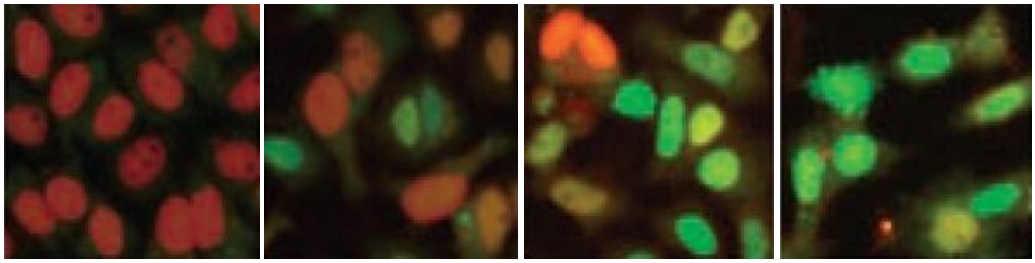
The resultant images are shown in the lecture notes. The m-files used to produce them follow:

```
clear all;
A=imread('fourier.gif');
A=A(1:128, 1:128);
A=im2double(A);
figure;
imshow(A);
for i=1:128
    for j=1:128
        B(i,j)=sin(0.5*i+0.2*j);
        B2(i,j)=sin(0.3*i+0.7*j);
    end
end
figure;
imshow(B);
noise = B;
A2=A+0.5*B;
hinoise = A2;
lownoise = A + 0.05*B;
newnoise = A + 0.5*B2;
figure;
imshow(lownoise);
C=fft2(B);
```

```
%figure;
%imshow(abs(C));
ma=max(max(abs(C)));
mi=min(min(abs(C)));
D=(abs(C)-mi)/(ma-mi);
%figure;
%imshow(D);
E=1-D;
%figure;
%imshow(E);
SE=[0 0 0 0 0 0 0 0 0; 0 1 1 1 1 1 1 1 0; 0 1 1 1 1 1 1 1 0; 0 1 1 1 1 1 1 1 0; 0 1 1 1 1
1 1 1 0; 0 1 1 1 1 1 1 1 0; 0 1 1 1 1 1 1 1 0; 0 1 1 1 1 1 1 1 0; 0 0 0 0 0 0 0 0 0];
F=imerode(E, SE);
%figure;
%imshow(F);
A2F=fft2(A2);
%RA2F=A2F-0.5*C;
%figure;
%imshow(abs(A2F));
ma2=max(max(abs(A2F)));
mi2=min(min(abs(A2F)));
A2FS=(abs(A2F)-mi)/(ma-mi);
%figure;
%imshow(abs(A2FS));
RA2F=A2F.*F;
RA2=ifft2(RA2F);
figure;
imshow(abs(RA2));
for i=1:128
    for j=1:128
        if (rand(1) >0.99)
            AA(i,j)=A(i,j)+0.25;
        else
            AA(i,j)=A(i,j);
        end
    end
end
figure;
imshow(AA);
co=55;
for i=1:128
    for j=1:128
        if ((i>64-co)& (i<64+co) & (j>64-co) & (j<64+co))
            BB(i,j)=0;
        else
            BB(i,j)=1;
        end
    end
end
%figure;
%imshow(BB);
CC=fft2(AA);
DD=CC.*BB;
AAR=ifft2(DD);
AAR=abs(AAR);
%figure;
%imshow(AAR);
SF=[1 1 1; 1 1 1; 1 1 1];
SF=1/9*SF;
AAR2=conv2(AA,SF);
%figure;
%imshow(AAR2);
dif1=AA-A;
```

```
%figure;
%imshow(dif1, [0,0.25]);
dif2=AAR-A;
%figure;
%imshow(dif2, [0,0.25]);
dif3=AAR2(2:129, 2:129)-A;
%figure;
%imshow(dif3, [0,0.25]);
o1=0;
o2=0;
o3=0;
for i=1:128
    for j=1:128
        o1=o1+dif1(i,j)*dif1(i,j);
        o2=o2+dif2(i,j)*dif2(i,j);
        o3=o3+dif3(i,j)*dif3(i,j);
    end
end
```

5. **Image segmentation and quantification** Four images are extracted from the Perlman et al. paper (Fig. 4).



(a) Segment the nuclei from the rest of the image. Show a "mask" image (2 bit) for each images that select only the nuclei.
(b) Develop an algorithm to measure the red vs the green intensities from each pixel within the nuclei.
(c) Create a scattered plot for each valid pixels (within the nuclei) as described in the Perlman paper for each image.

This M-file is modified from the work of Li Huipeng, a SMA grad student who took my course last summer.

```
%% 1.Read an image
rgb_perlman1=imread('perlman1.png');
imshow(rgb_perlman1); %show the rgb image
%% 2.Change the image into gray scale
gray_perlman1=rgb2gray(rgb_perlman1);
figure,imshow(gray_perlman1); % show the image
[row column]=size(gray_perlman1); %get the size of the matrix
%% 3.Finding the nuclues areas
level=graythresh(gray_perlman1); % finding the threshold
gray_perlman1_bw=im2bw(gray_perlman1,level); % binarising the original image
figure,imshow(gray_perlman1_bw); % show the binarised image
%% 4.Extend the binarised image matrix to 3D
%  Why extend the binarised image matrix to 3D?
%  In order to match with the original image for the convenience of  ".*"
gray_perlman1_bw_rgb=zeros(row,column,3);
for (i=1:3)
    gray_perlman1_bw_rgb(:,:,i)=gray_perlman1_bw(:,:);
end
%% 5.Use the mask on the rgb image
double_rgb_perlman1=double(rgb_perlman1); %change the type of the data
nucleus_region=gray_perlman1_bw_rgb.*double_rgb_perlman1;
%% 6.Creat the scatter plot vectors
nuclues_r=zeros(row,column);
for i=1:159
    for j=1:161
            nuclues_r(i,j)=nucleus_region(i,j,1);
    end;
end;  % stored the r's to a matrix
nuclues_g=zeros(row,column);
for i=1:159
    for j=1:161
            nuclues_g(i,j)=nucleus_region(i,j,2);
    end;
end; % stored the g's to a matrix
r_x=reshape(nuclues_r,1,row*column); %reshape the r matrix into a vector
g_y=reshape(nuclues_g,1,row*column); %reshape the g matrix into a vector
%% 7.Scatter plot to get the final result
figure,scatter(r_x,g_y);  %scatter plot the r and g vectros
```

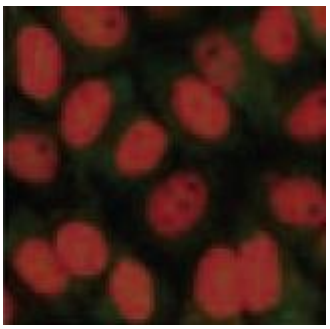


Fig. 1

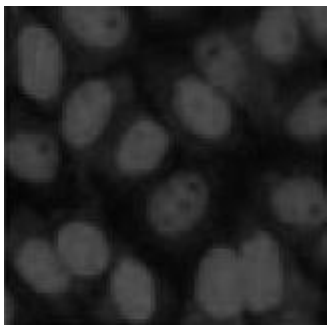


Fig.2

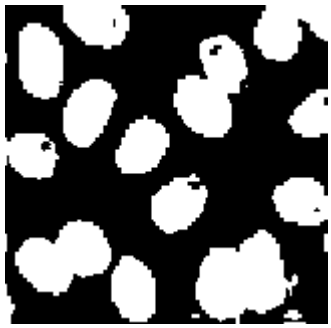


Fig. 3

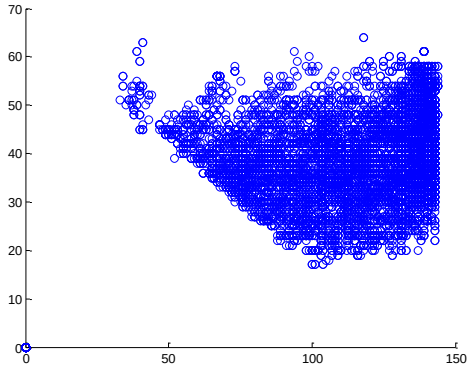


Fig.4

The results from the other three images are similar of course.