1) Generate a 3-level Gaussian pyramid (original image is level-0) and the corresponding Laplacian pyramid of an image (select one from the web, make it grayscale). Use the formula in the notes to first determine a viable image size (use N=3, and pick N_C and N_R), and crop the image (if needed) to test the pyramid code. Use a=0.4 for the Gaussian mask – **use separable masks!** Write/use functions for properly reducing and expanding an image. Write your own interpolation function - do not use Matlab/Python in-built interpolation functions (e.g., interp2). Lastly, perform a reconstruction of the original (cropped) image using the Laplacian pyramid. [8 pts]

Step1: Covert my image into grayscale. Use the formula in the notes to first determine a viable image size (use N=3, and pick NC=100 and NR=120), and crop the image to test the pyramid code. Left image is original image (hw3.jpg) and the right is cropIm.png.

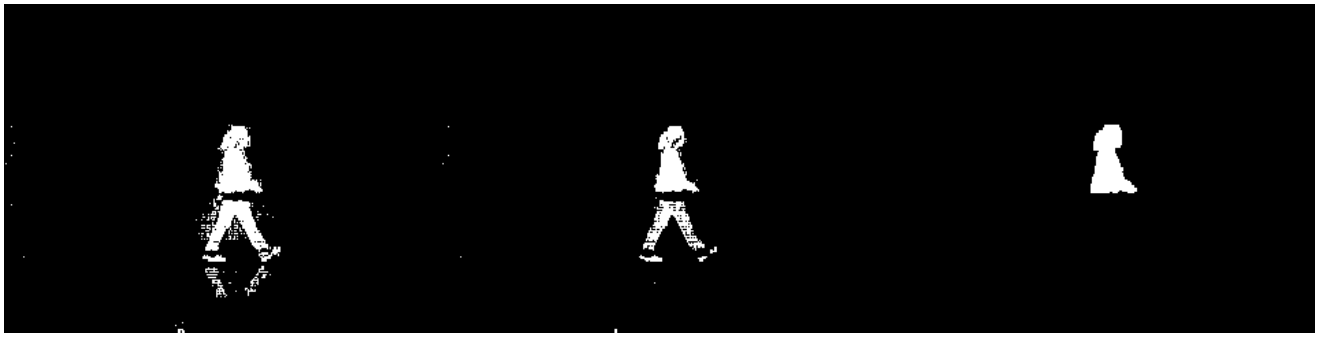
Step2: Use a=0.4 for the Gaussian mask to reduce an image. The top three images are reduceIm1.png, reduceIm2.png and reduceIm3.png. Write my own interpolation function to expand an image.The bottom three images are expandIm3.png, expandIm2.png and expandIm1.png.

Step3: laplacian = cropIm - expandIm; reconstruction = laplacian + expandIm. The top three images are laplacian1.png, laplacian2.png and laplacian3.png. The bottom three images are Reconstruction.png, Reconstruction2.png and Reconstruction3.png.

2) Using the grayscale images (walk.bmp, bg000.bmp) provided on the WWW site, perform background subtraction 1 (abs diff) to extract the object. Make sure your image is of type *double*! Experiment with thresholds and discuss. [2 pts]

Use abs diff to extract the object with T=10, 20 and 30.The larger the T, the clearer the image. Because as T gets bigger and bigger, there are fewer and fewer points that can satisfy this big absolute difference, so the noise goes down.

3) Using the grayscale images (walk.bmp, bg[000-029].bmp) provided on the WWW site, perform background subtraction 2 using statistical distances. Experiment with thresholds and discuss. [5 pts]



Use statistical distances to extract the object with T=10, 20 and 30.The larger the T, the clearer the image. When T is too large, there will be less noise, but information will be lost.

4) Dilate your best binary image resulting from problem 3) using: [1 pt]

I dilate the images from Q3. As we can see, the details of the image become less sharp as we dialte images. But the missing part of the image from the question 3 does not appear in this question. This is because small pieces join together to form a large area. So as T gets bigger, the picture gets clearer and the problem of information loss is mitigated.

5) Next perform a connected components algorithm, and keep only the largest region in L (save/display as an image). [1 pt]

Because only the largest area is retained, there is less background noise. But the problem is that when T is very large, the picture information gets lost.

Turn in a report containing all code, printouts of images, and discussion of results. Make a script to do all tasks and call needed functions. Upload your report, code, and selected images to Carmen. [2 pts]

```
%Problem 1
myIm = imread('hw3.jpg');
grayIm = double(rgb2gray(myIm));
N = 3;
C = 100 * 2^N +1;
R = 120 * 2^N + 1;
cropIm = imcrop(grayIm,[0,0,R,C]);
imwrite(uint8(grayIm), 'grayIm.png');
imwrite(uint8(cropIm), 'cropIm.png');

%Interpolation
function [expandIm, laplacian] = interpolation(cropIm, sampleIm)

[rows, columns] = size(sampleIm);


expand_row = 2*rows -1;
expand_col = 2*columns -1;
expand_row_Im = zeros(expand_row,columns);
expandIm = zeros(expand_row,expand_col);
for i = 1:columns-1
    for j = 1:rows-1
        expand_row_Im(j*2-1,i) = sampleIm(j,i);
        expand_row_Im(j*2,i) = mean([sampleIm(j,i),sampleIm(j+1,i)]);
        if i == columns -1
            expand_row_Im(j*2-1,columns) = sampleIm(j,columns);
            expand_row_Im(j*2,columns) =
mean([sampleIm(j,columns),sampleIm(j+1,columns)]);
        end
    end

    expand_row_Im(rows*2-1,i) = sampleIm(rows,i);
    expand_row_Im(rows*2-1,columns) = sampleIm(rows,columns);


end
for i = 1:columns-1
    for j = 1:expand_row-1
        expandIm(j,i*2-1) = expand_row_Im(j,i);
        expandIm(j,i*2) = mean([expand_row_Im(j,i),expand_row_Im(j,i+1)]);
        if i == columns -1
            expandIm(j,columns*2-1) = expand_row_Im(j,columns);
        end
    end

    expandIm(expand_row,i*2-1) = expand_row_Im(expand_row,i);
    expandIm(expand_row,i*2) =
mean([expand_row_Im(expand_row,i),expand_row_Im(expand_row,i+1)]);
    expandIm(expand_row,expand_col) = expand_row_Im(expand_row,columns);
```

```
end

laplacian = cropIm - expandIm;

end


%level 1
a = 0.4;
wR = [.25-.5*a, .25,a,.25,.25 -.5*a];
wC = transpose(wR);
reduceIm1 = imfilter(cropIm, wR);
reduceIm1 = imfilter(reduceIm1, wC);
sampledImage1 = reduceIm1 (1:2:end, 1:2:end);
[expandIm1, laplacian1] = interpolation(cropIm, sampleIm1);
imwrite(uint8(reduceIm1), 'reduceIm1.png');
imwrite(uint8(sampleIm1), 'sampleIm1.png');
imwrite(uint8(laplacian1), 'laplacian1.png');
%level 2
reduceIm2 = imfilter(sampleIm1, wR);
reduceIm2 = imfilter(reduceIm2, wC);
sampledImage1 = reduceIm2 (1:2:end, 1:2:end);
[expandIm2, laplacian2] = interpolation(sampleIm1, sampleIm2);
imwrite(uint8(reduceIm2), 'reduceIm2.png');
imwrite(uint8(sampleIm2), 'sampleIm2.png');
imwrite(uint8(laplacian2), 'laplacian2.png');

%level 3
reduceIm3 = imfilter(sampleIm2, wR);
reduceIm3 = imfilter(reduceIm3, wC);
sampledImage2 = reduceIm3 (1:2:end, 1:2:end);
[expandIm3, laplacian3] = interpolation(sampleIm2, sampleIm3);
imwrite(uint8(blurIm3), 'reduceIm3.png');
imwrite(uint8(sampleIm3), 'sampleIm3.png');
imwrite(uint8(laplacian3), 'laplacian3.png');

expandIm3 = interpolation(sampleIm2, sampleIm3);
recon3 = laplacian3 + expandIm3;
expandIm2 = interpolation(sampleIm1, recon3);
recon2 = laplacian2 + expandIm2;
expandIm1 = interpolation(cropIm, recon2);
recon = expandIm1 + laplacian1;

imwrite(uint8(expandIm1), 'expandIm1.png');
imwrite(uint8(expandIm2), 'expandIm2.png');
imwrite(uint8(expandIm3), 'expandIm3.png');
imwrite(uint8(recon), 'Reconstruction.png');
imwrite(uint8(recon2), 'Reconstruction2.png');
imwrite(uint8(recon3), 'Reconstruction3.png');
imagesc(recon);
axis('image');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Problem 2

walkIm = double(imread('walk.bmp'));
bgIm = double(imread('bg000.bmp'));
```

```
T = 10;
B = abs(walkIm - bgIm) > T;
imwrite(B, 'p2_10.png');
T = 20;
B = abs(walkIm - bgIm) > T;
imwrite(B, 'p2_20.png');
T = 30;
B = abs(walkIm - bgIm) > T;
imwrite(B, 'p2_30.png');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Problem 3


for i=1:30
filename = sprintf('bg%03d.bmp', i-1); % if starts with 000
bgIms(:,:,i) = double(imread(filename));
end

std_val = std(bgIms, 0, 3);
inten_mea = mean(bgIms,3);
T = 10;
B1 = ((walkIm - inten_mea).^2 ./ std_val.^2) > T^2;
imwrite(B1, 'p3_10.png');
T = 20;
B2 = ((walkIm - inten_mea).^2 ./ std_val.^2) > T^2;
imwrite(B2, 'p3_20.png');
T = 30;
B3 = ((walkIm - inten_mea).^2 ./ std_val.^2) > T^2;
imwrite(B3, 'p3_30.png');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Problem 4
d_bsIm1 = bwmorph(B1, 'dilate');
imwrite(d_bsIm1, 'p4_10.png');
d_bsIm2 = bwmorph(B2, 'dilate');
imwrite(d_bsIm2, 'p4_20.png');
d_bsIm3 = bwmorph(B3, 'dilate');
imwrite(d_bsIm3, 'p4_30.png');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Problem 5
L1 = bwareafilt(d_bsIm1,1);
imwrite(L1, 'p5_10.png');
L2 = bwareafilt(d_bsIm2,1);
imwrite(L2, 'p5_20.png');
L3 = bwareafilt(d_bsIm3,1);
imwrite(L3, 'p5_30.png');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```