Hanfei Geng

hgeng4

Nov 14th, 2016

Report Item 1:

code:

[sound,fs] = audioread('audioclip.wav');

sizeO = length(sound);

SO = fftshift(fft(sound));

w = fftshift([0:sizeO-1]/sizeO\*2\*pi);

w(1:sizeO/2) = w(1:sizeO/2) - 2\*pi;

freqO = fs\*w/2/pi;

figure(6)

plot(freqO,mag2db(abs(SO)))

U = 2

D = 3

sound\_up = upsample(sound,U);

sizeU = length(sound\_up);

SU = fftshift(fft(sound\_up));

wU = fftshift([0:sizeU-1]/sizeU\*2\*pi);

wU(1:sizeU/2) = wU(1:sizeU/2) - 2\*pi;

fsU = 2\*fs

freqU = fsU\*wU/2/pi;

figure(1)

plot(freqU,mag2db(abs(SU)))

title('Spectrum after upsampling before filtering')

xlabel('frequency(Hz)')

ylabel('Magnitude(dB)')

%filtering

filtered = zeros(length(SU),3);

filtered(:,1) = freqU;

filtered(:,2) = (SU(:,1))';

filtered(:,3) = (SU(:,2))';

filtered(1:337495,2:3) = 0;

filtered(1012495:1350000,2:3) = 0;

figure(2)

filtered(:,2:3) = 2 \* filtered(:,2:3);

plot(filtered(:,1),mag2db(abs(filtered(:,2:3))))

title('Upsampled spectrum after filtering')

xlabel('frequency(Hz)')

ylabel('Magnitude(dB)')

%go back to time domain

sound\_up\_filter = ifft(ifftshift((filtered(:,2:3))))

figure(3)

plot(real(sound\_up\_filter))

hold on

plot(sound\_up)

sound\_down = downsample(real(sound\_up\_filter),D);

figure(4)

plot(sound\_down)

hold on

plot(sound)

sizeD = length(sound\_down);

SD = fftshift(fft(sound\_down));

wD = fftshift([0:sizeD-1]/sizeD\*2\*pi);

wD(1:sizeD/2) = wD(1:sizeD/2) - 2\*pi;

fsD = fsU/D

freqD = fsD\*wD/2/pi;

figure(5)

plot(freqD,mag2db(abs(SD)))

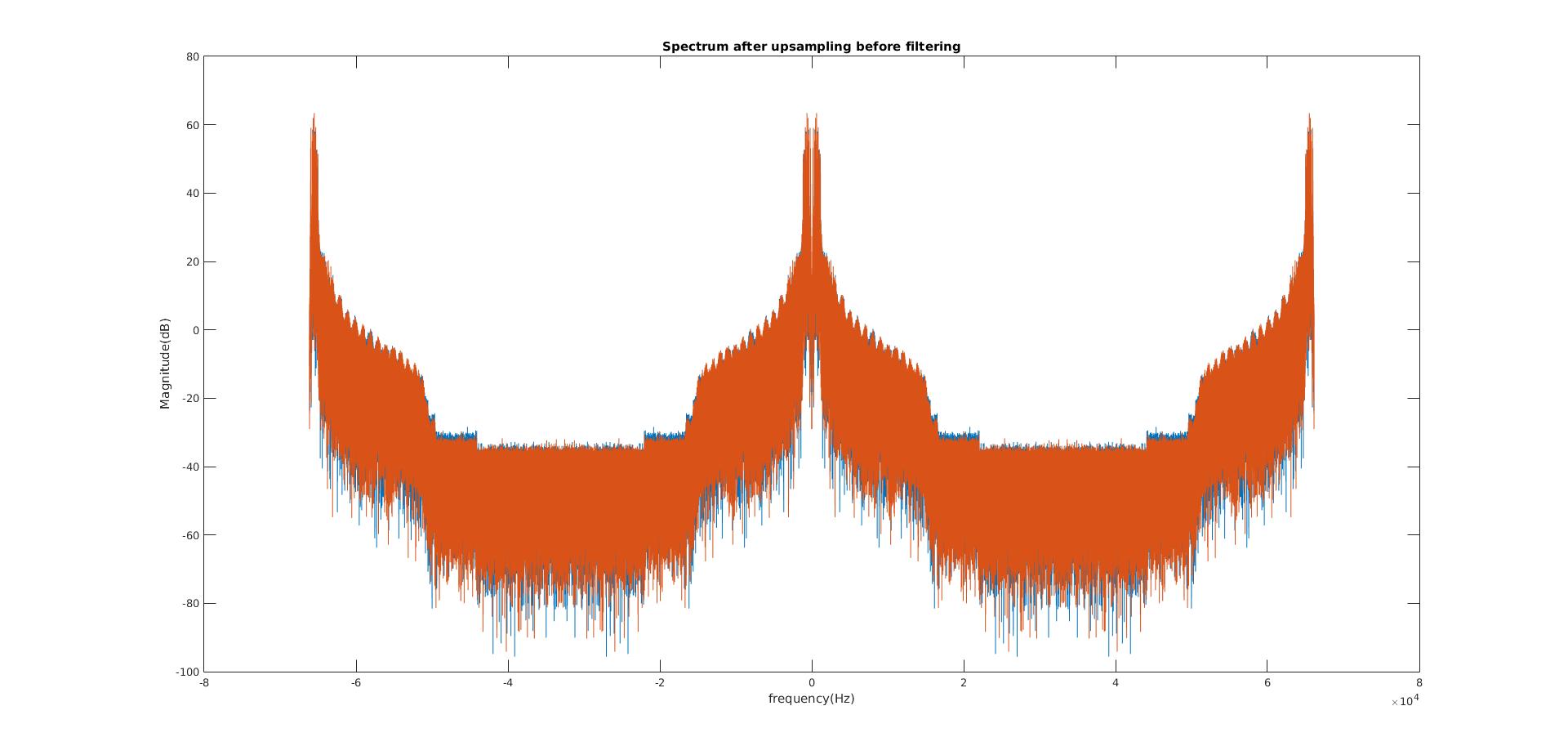
title('Final sample after downsampling')

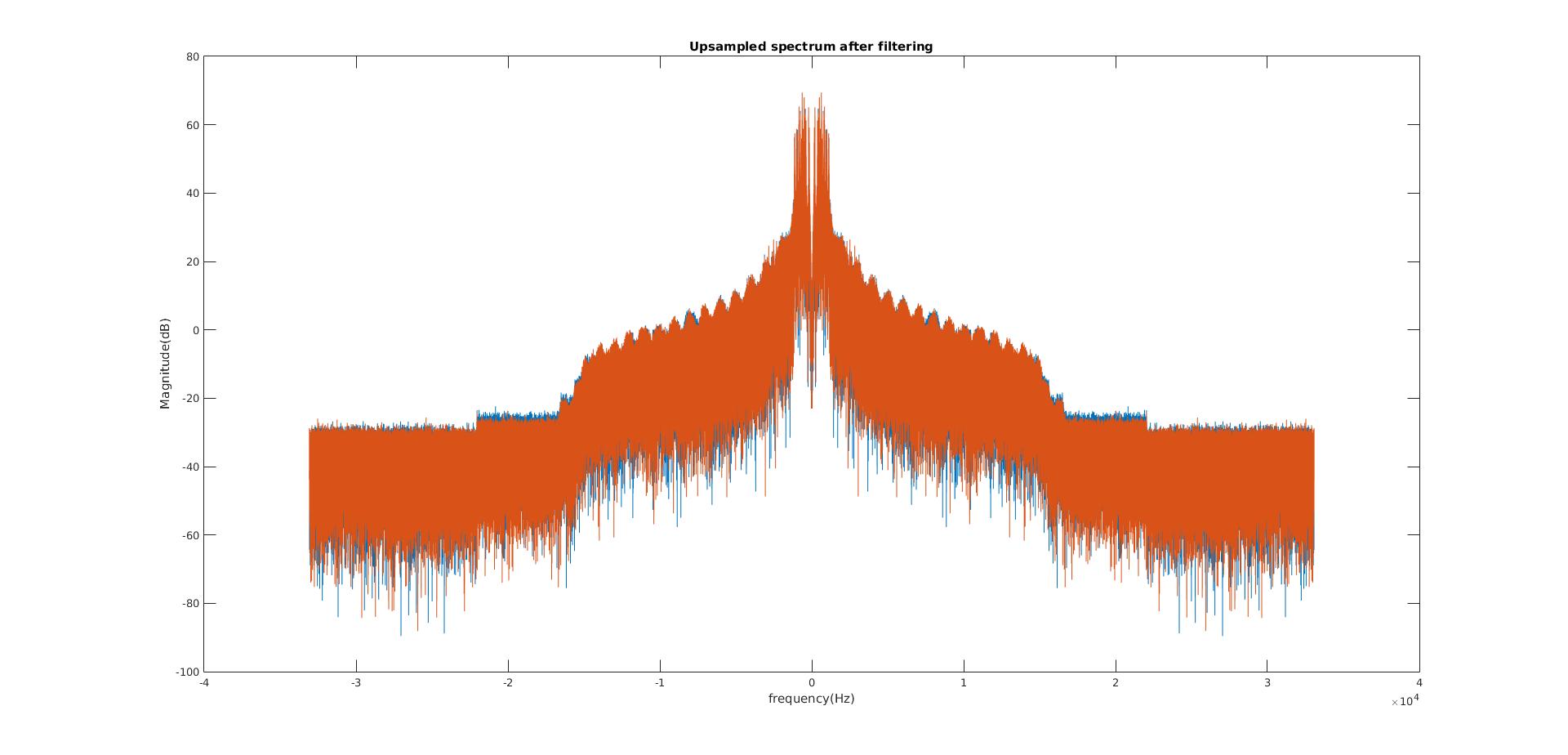
xlabel('frequency(Hz)')

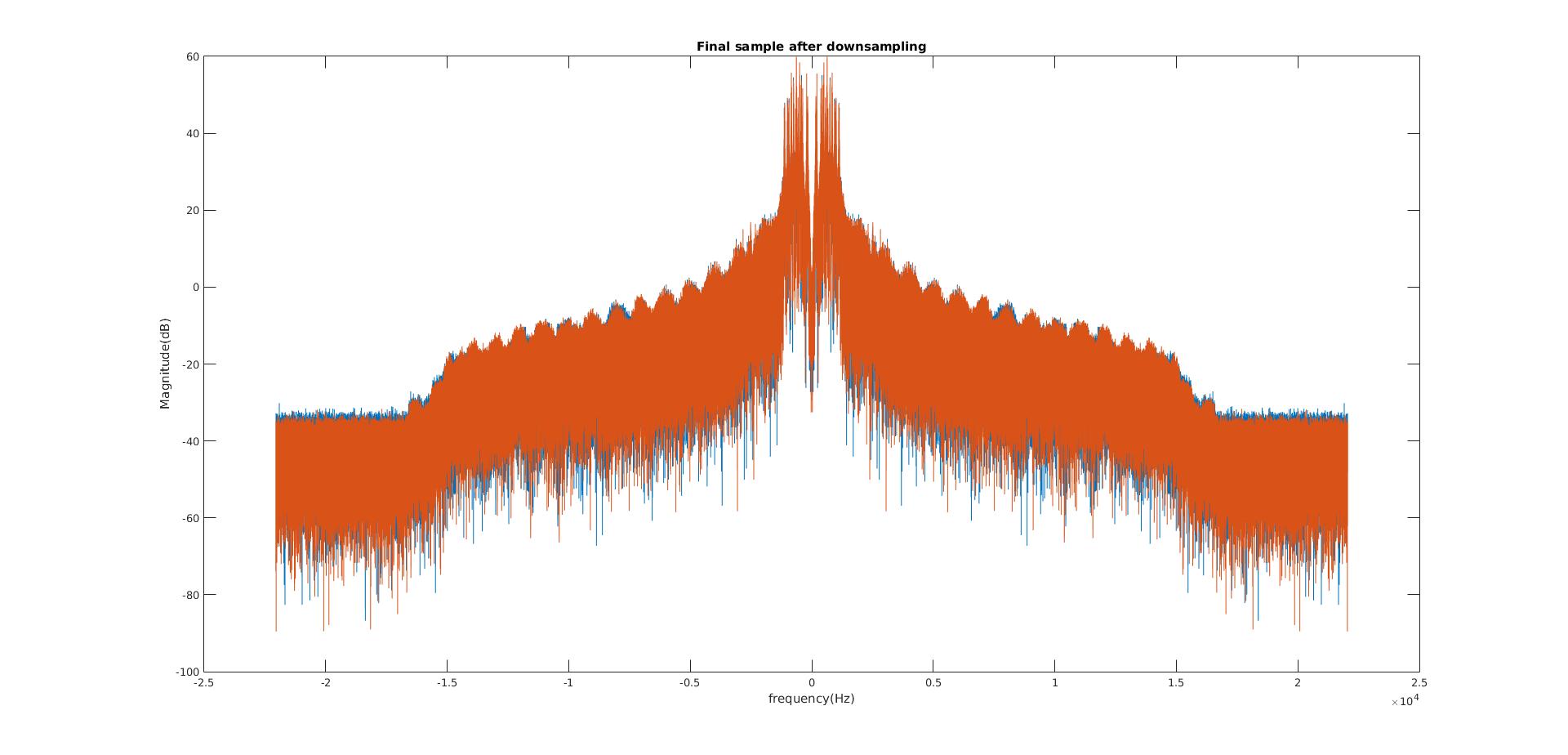
ylabel('Magnitude(dB)')

U = 2

D = 3







They sound the same

Report Item 2:

function [dft2] = myDFT2(image)

[row,col] = size(image);

dft2 = zeros(row,col);

for m = 1:row

dft2(m,:) = fft(image(m,:));

end

for n = 1:col

dft2(:,n) = fft(dft2(:,n)) ;

end

end

M = 1:30;

N = 1:30;

[m,n] = meshgrid(M,N)

kx = pi/4

ky = 0

z = cos(ky\*m + kx\*n);

figure(1)

subplot(121)

imagesc(z)

title('f for kx = pi/4,ky=0')

xlabel('n')

ylabel('m')

Z = myDFT2(z)

Z1 = fft2(z)

subplot(122)

kX = ([0:length(N)-1]/length(N)\*2\*pi)

kY = ([0:length(M)-1]/length(M)\*2\*pi)

imagesc(kX,kY,abs((Z)))

title('DFT for f for kx = pi/4,ky=0')

xlabel('kx')

ylabel('ky')

kx = 0

ky = pi/4

z = cos(ky\*m + kx\*n);

figure(2)

subplot(121)

imagesc(z)

title('f for kx = 0,ky=pi/4')

xlabel('n')

ylabel('m')

Z = myDFT2(z)

Z1 = fft2(z)

subplot(122)

kX = ([0:length(N)-1]/length(N)\*2\*pi)

kY = ([0:length(M)-1]/length(M)\*2\*pi)

imagesc(kX,kY,abs((Z)))

title('DFT for f for kx = 0,ky=pi/4')

xlabel('kx')

ylabel('ky')

kx = pi/4

ky = pi/4

z = cos(ky\*m + kx\*n);

figure(3)

subplot(121)

imagesc(z)

title(' f for kx = pi/4,ky=pi/4')

xlabel('n')

ylabel('m')

Z = myDFT2(z)

Z1 = fft2(z)

subplot(122)

kX = ([0:length(N)-1]/length(N)\*2\*pi)

kY = ([0:length(M)-1]/length(M)\*2\*pi)

imagesc(kX,kY,abs((Z)))

title('DFT for f for kx = pi/4,ky=pi/4')

xlabel('kx')

ylabel('ky')

kx = pi/4

ky = -pi/4

z = cos(ky\*m + kx\*n);

figure(4)

subplot(121)

imagesc(z)

title('f for kx = pi/4,ky=-pi/4')

xlabel('n')

ylabel('m')

Z = myDFT2(z)

Z1 = fft2(z)

subplot(122)

kX = ([0:length(N)-1]/length(N)\*2\*pi)

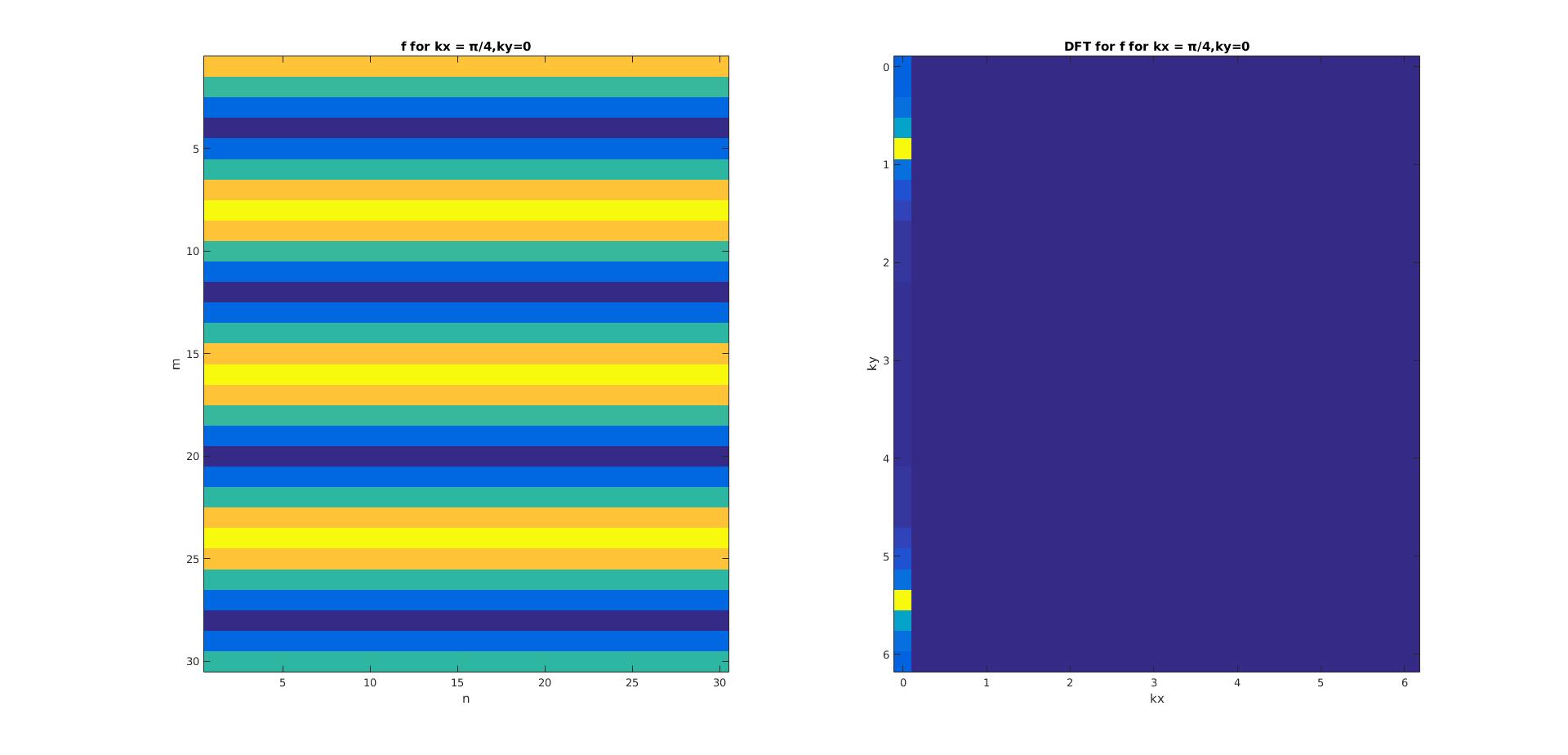
kY = ([0:length(M)-1]/length(M)\*2\*pi)

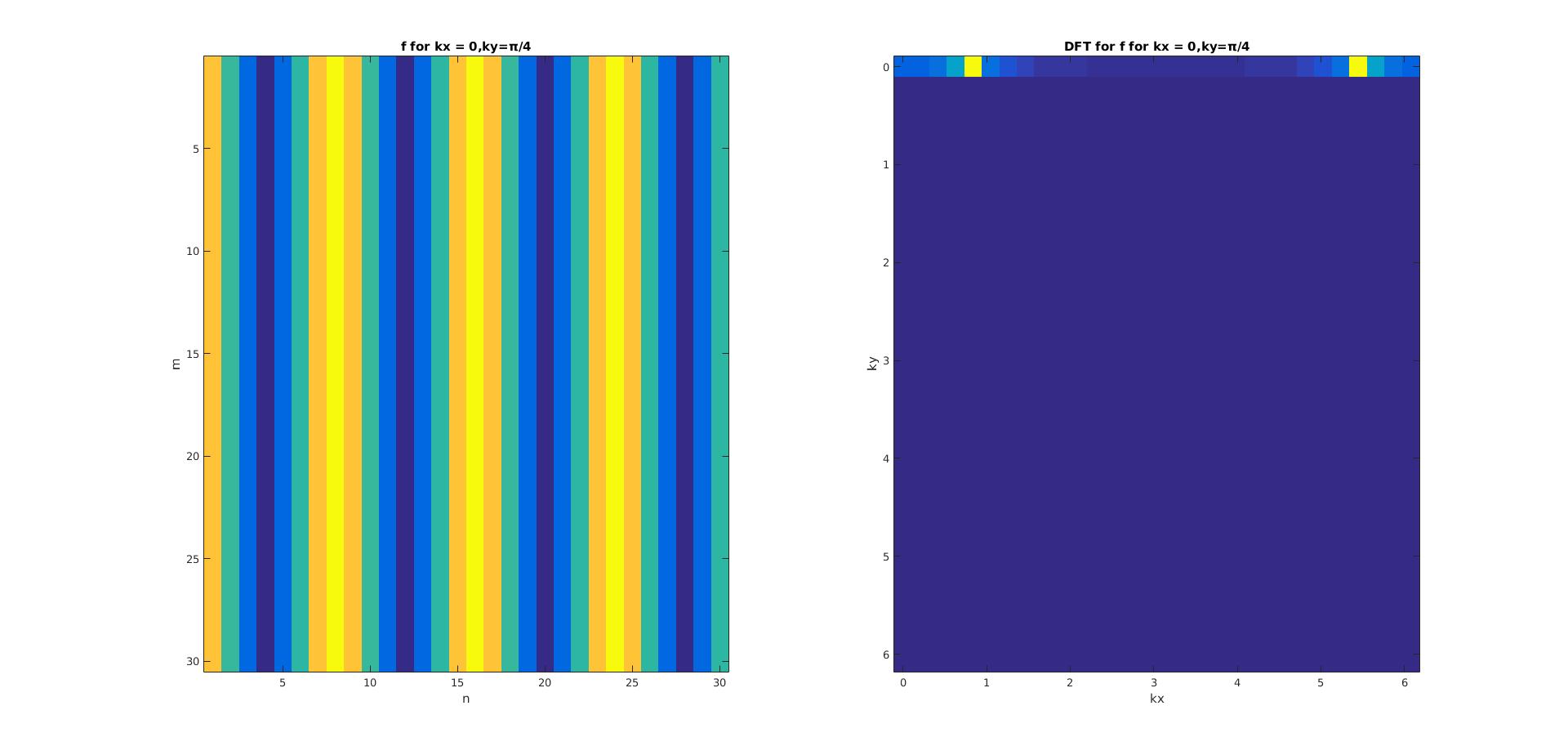
imagesc(kX,kY,abs((Z)))

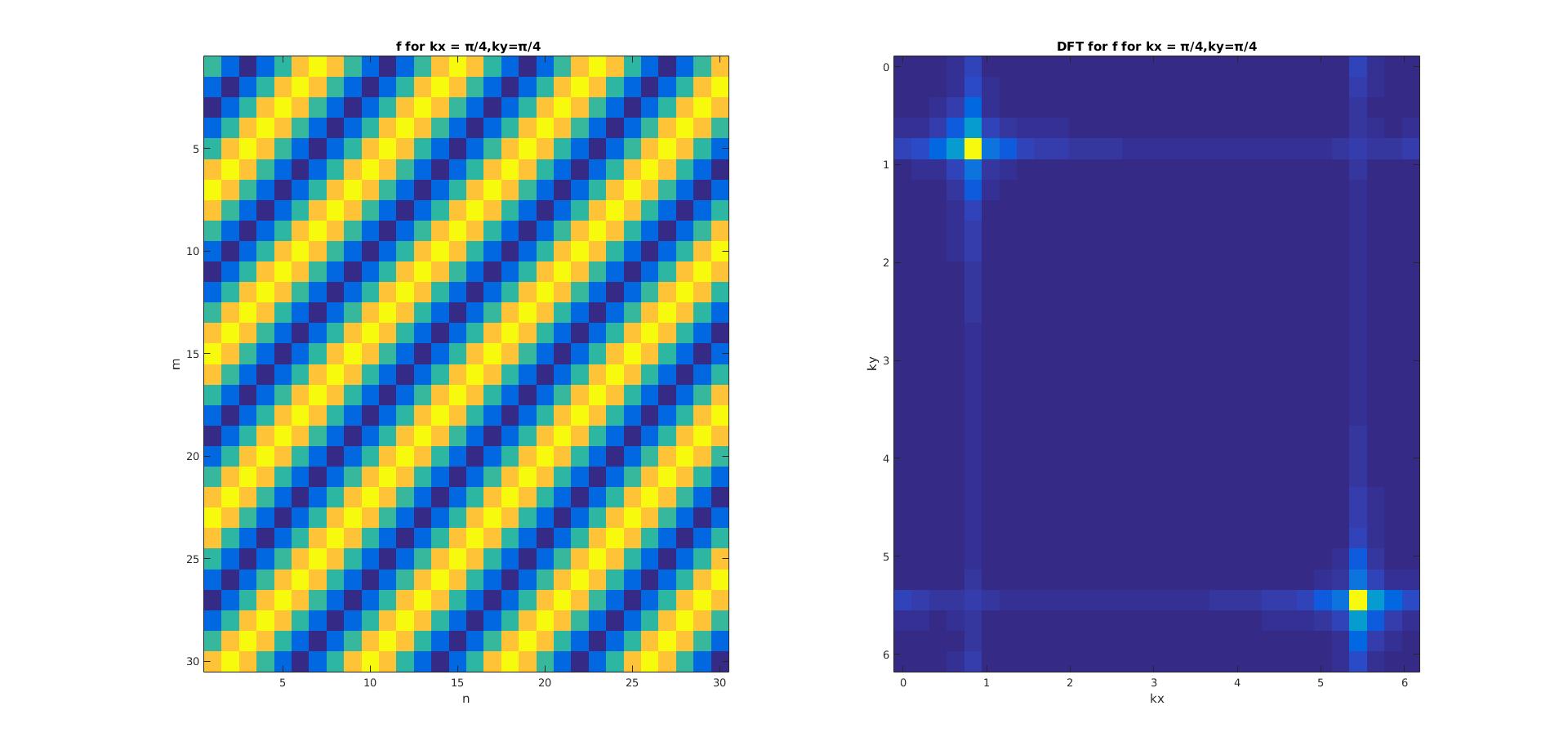
title('DFT for f for kx = pi/4,ky=-pi/4')

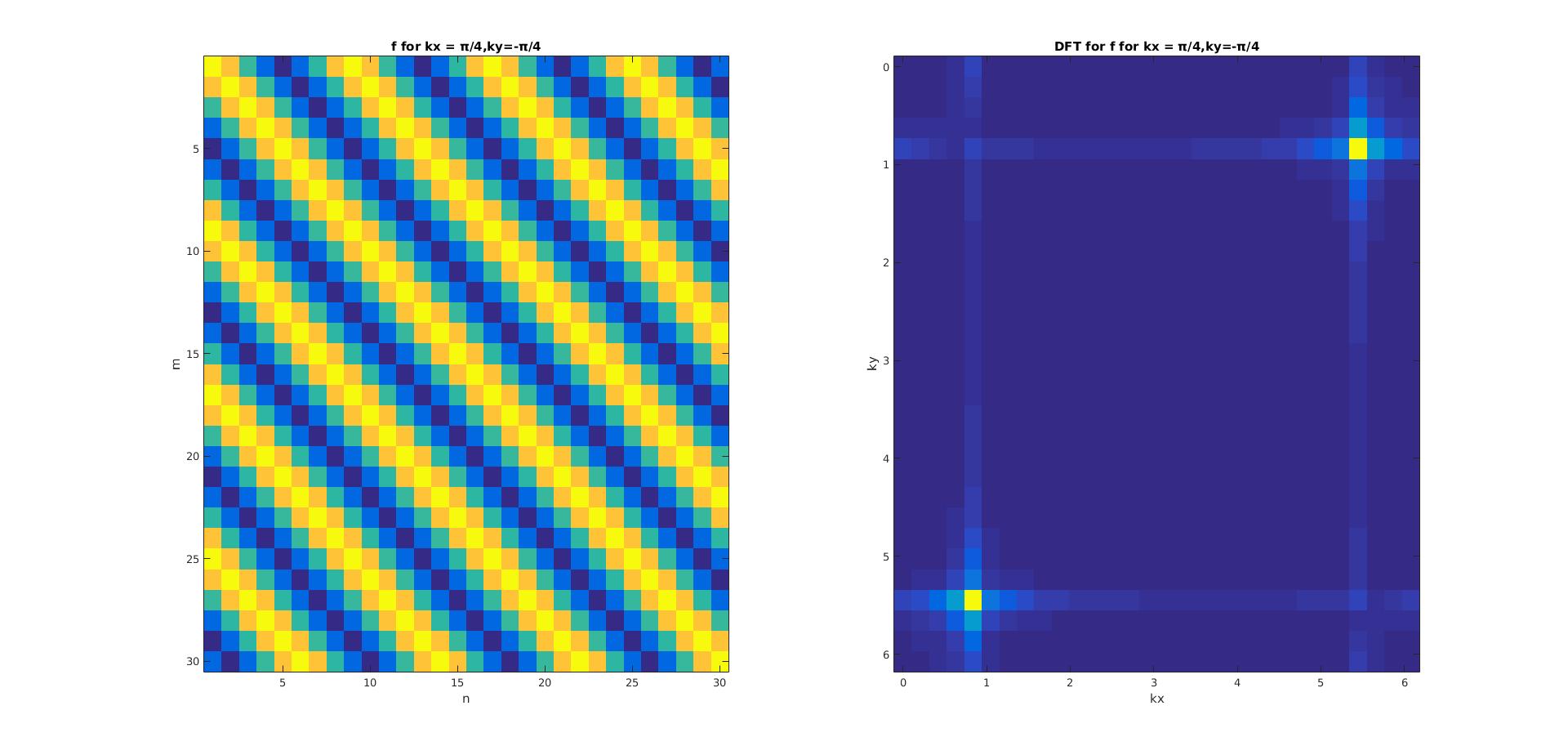
xlabel('kx')

ylabel('ky')









Report Item 3:

image1 = imread('image1.jpg')

image1 = double(image1)

h = [1/8,1/16,1/8;

1/16,1/4,1/16;

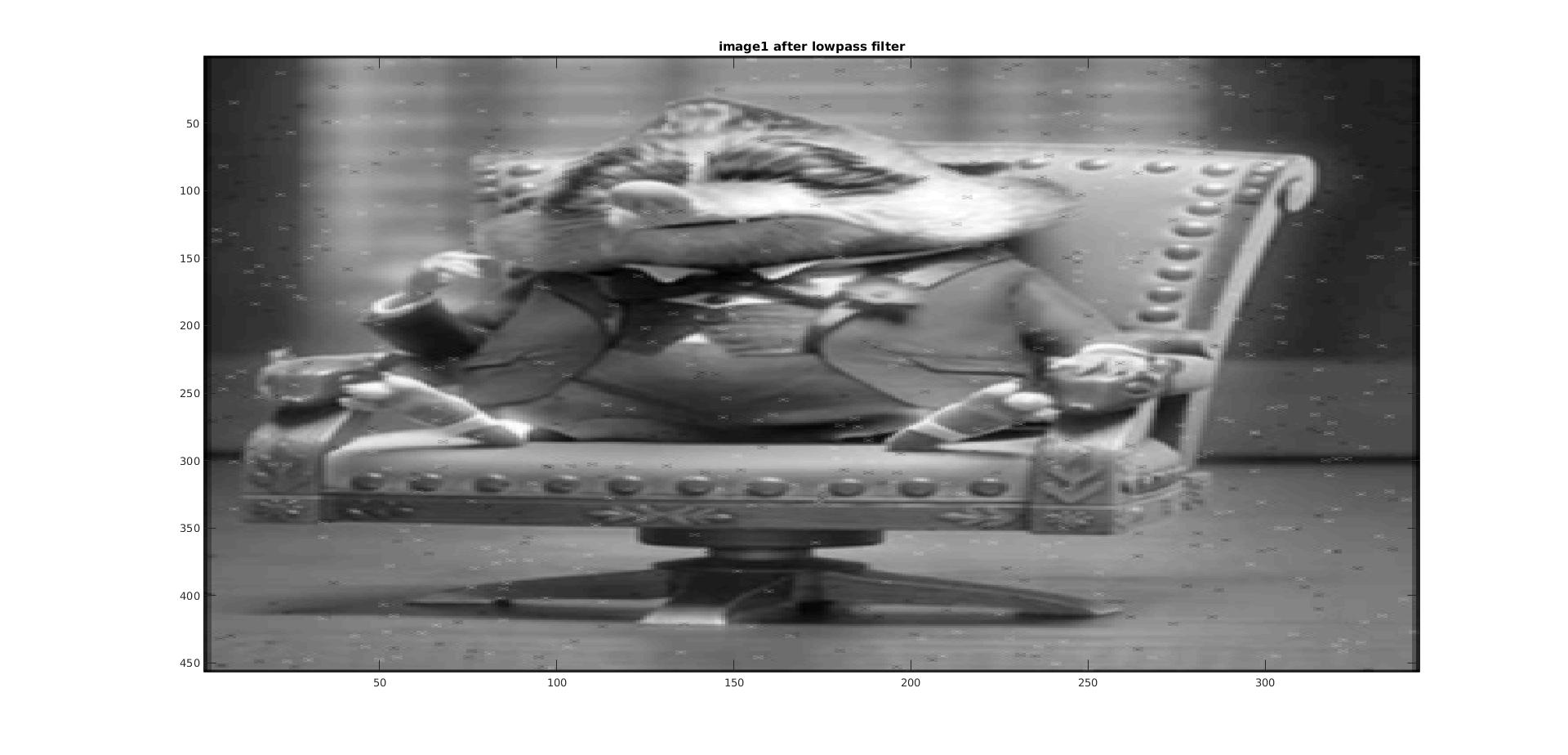
1/8,1/16,1/8]

c = conv2(h,image1)

colormap gray

imagesc(c)

title('image1 after lowpass filter')

Report Item 4:

code:

image2 = imread('image2.jpg')

image2 = double(image2)

h = [-1,-1,-1;

-1,8,-1;

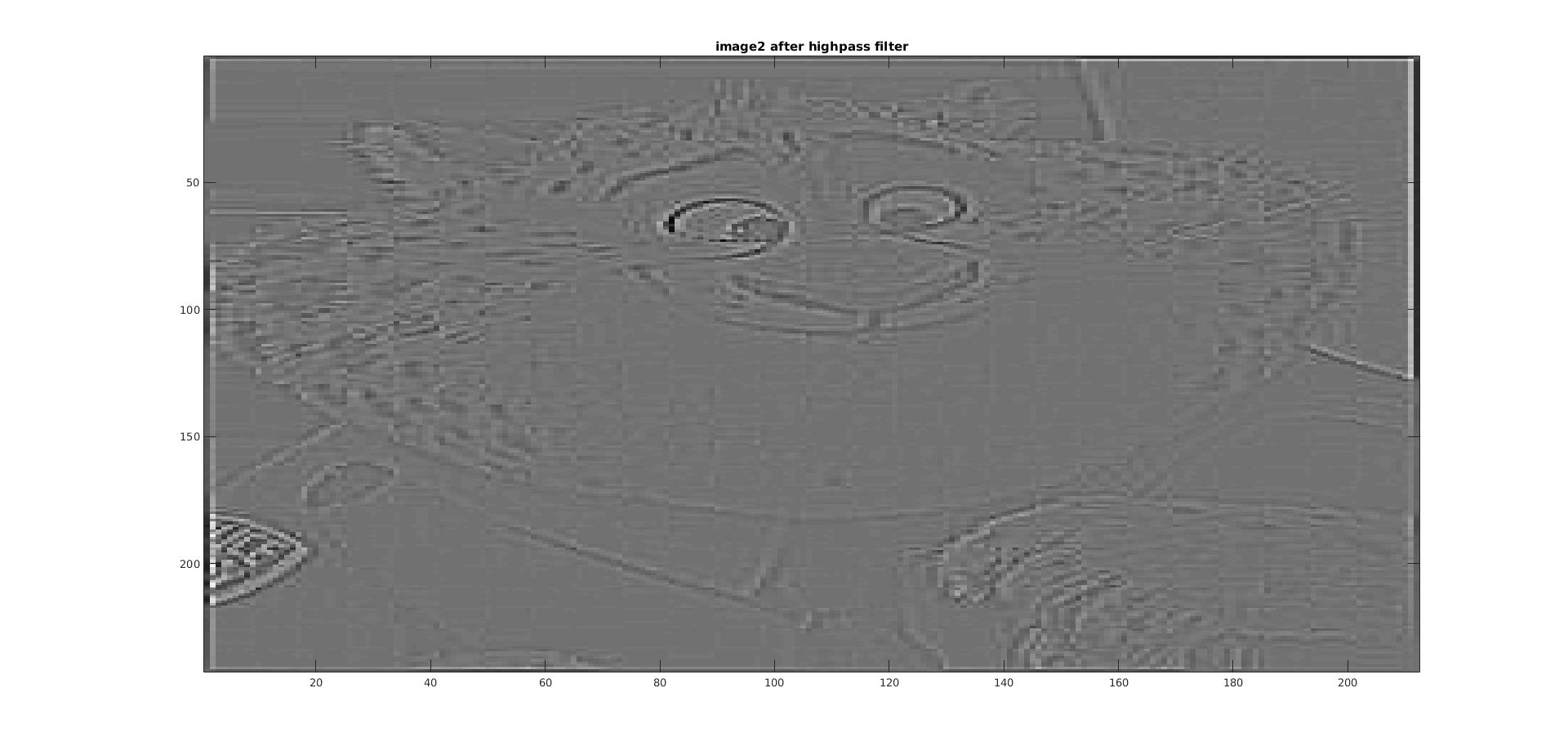
-1,-1,-1]

c = conv2(h,image2)

colormap gray

imagesc(c)

title('image2 after highpass filter')

Report Item 5:

train = imread('train.jpg');

train = double(train);

%imshow(train)

train\_lab = rgb2lab(train);

imshow(train\_lab,[])

title('train in lab color')

figure;

subplot(131)

imshow(train\_lab(:,:,1),[])

title('channel 1')

subplot(132)

imshow(train\_lab(:,:,2),[])

title('channel 2')

subplot(133)

imshow(train\_lab(:,:,3),[])

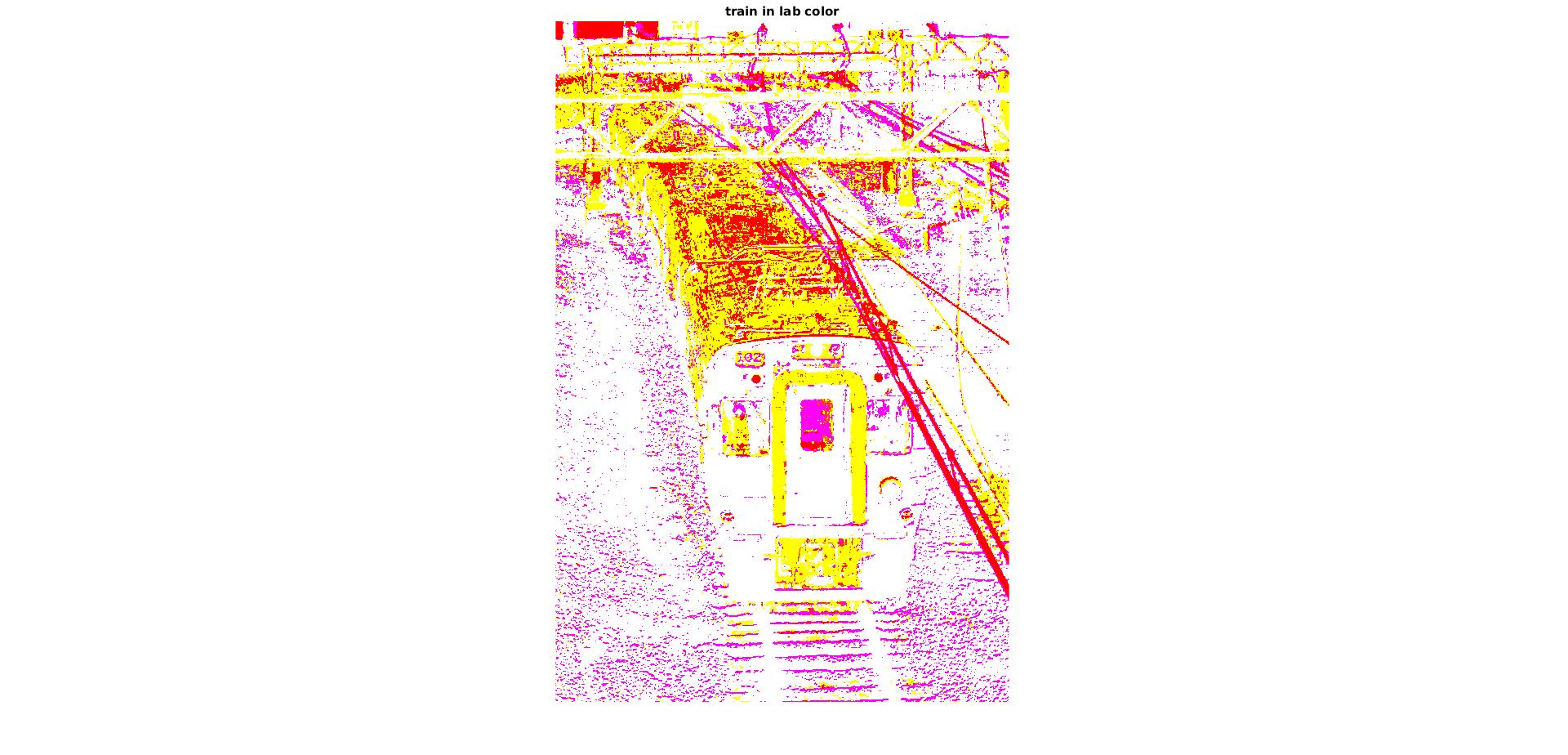
title('channel 3')

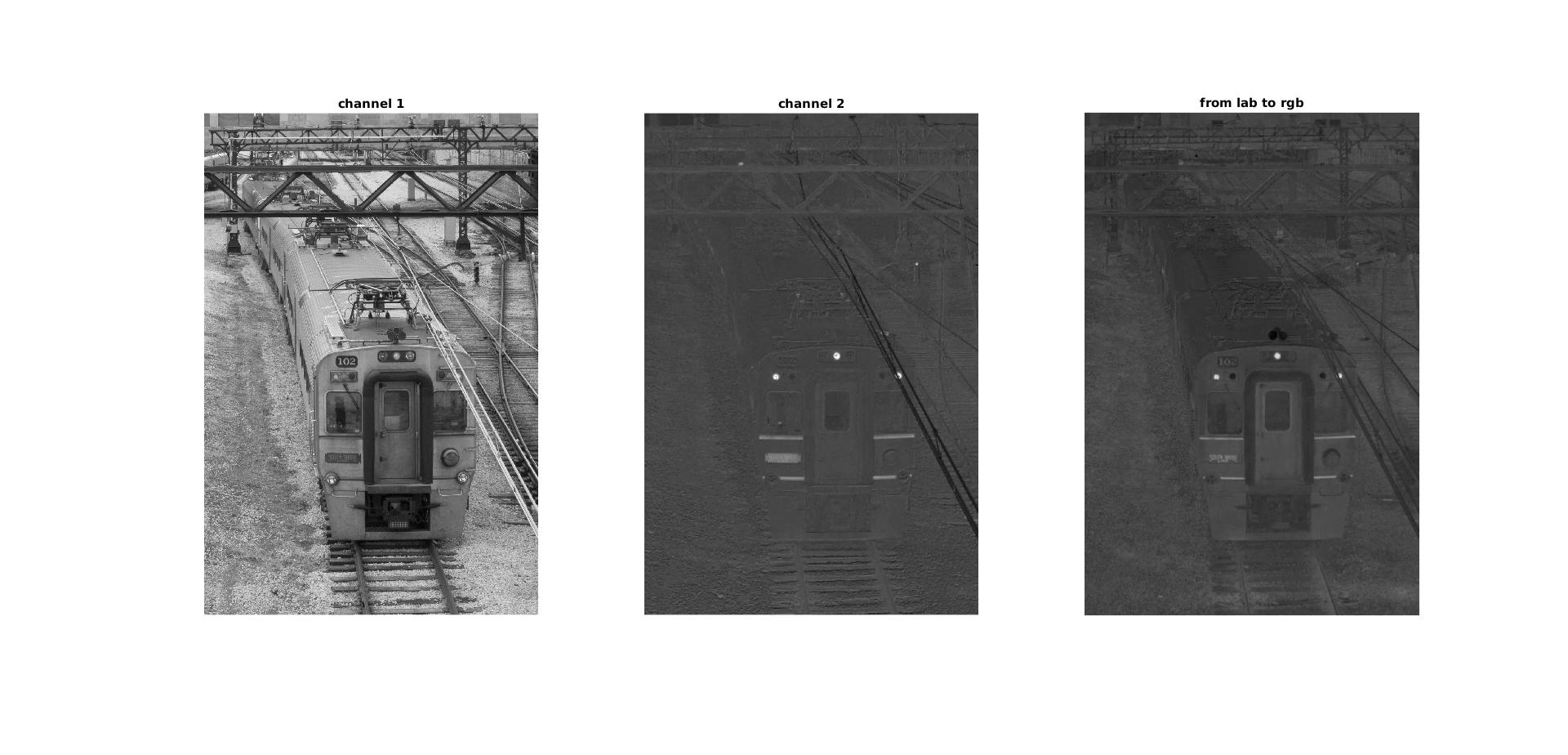
train\_new = lab2rgb(train\_lab)

figure

imshow(uint8(train\_new),[])

title('from lab to rgb')







The converted image is the same as the input!!!

Report Item 6:

function [out\_image] = meanFilter(image,kernelSize)

image\_lab = rgb2lab(image);

kernel = 1/(kernelSize\*kernelSize) \* ones(kernelSize,kernelSize);

first = image\_lab(:,:,1)

con = conv2(first,kernel,'same');

image\_lab(:,:,1) = con;

out\_image = lab2rgb(image\_lab);

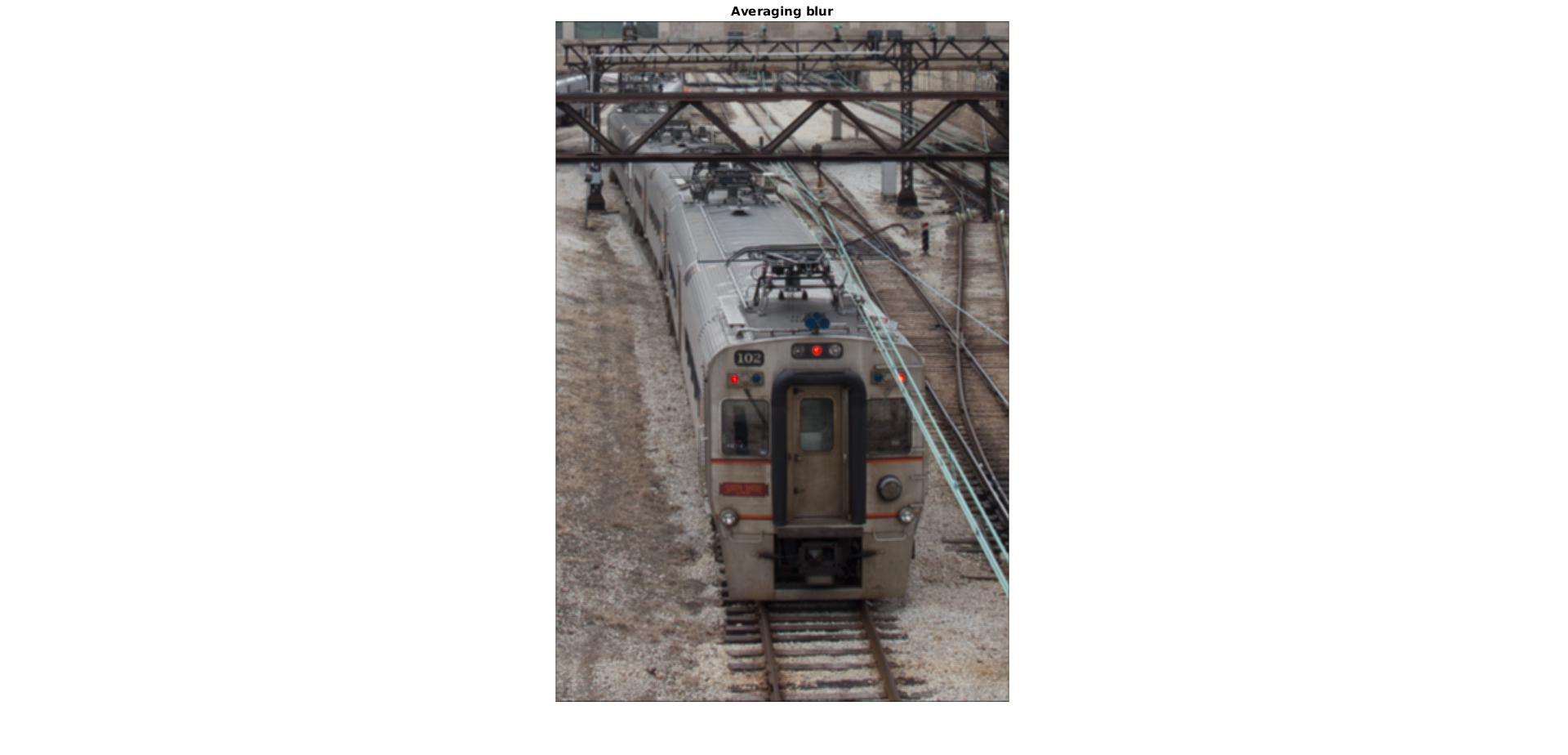
end

train = imread('train.jpg');

out = meanFilter(train,7);

imshow(out)

title('Averaging blur')



This kernel gives the same weight to both the nearer pixels and the further pixels;therefore it's called a meanfilter.

Report Item 7:

hsize = [5,11,5]

sigma = [1.5,3,3]

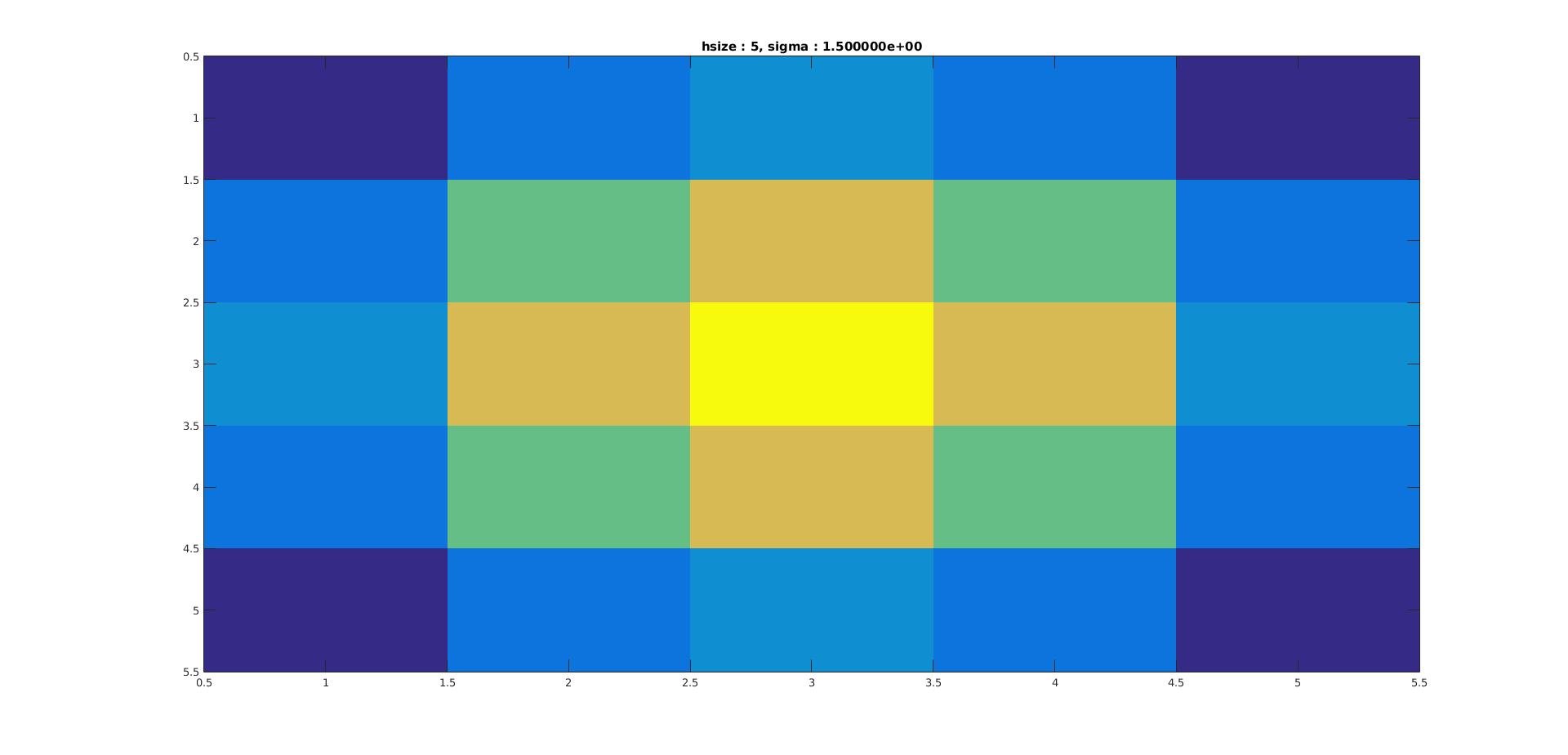
for i = 1:3

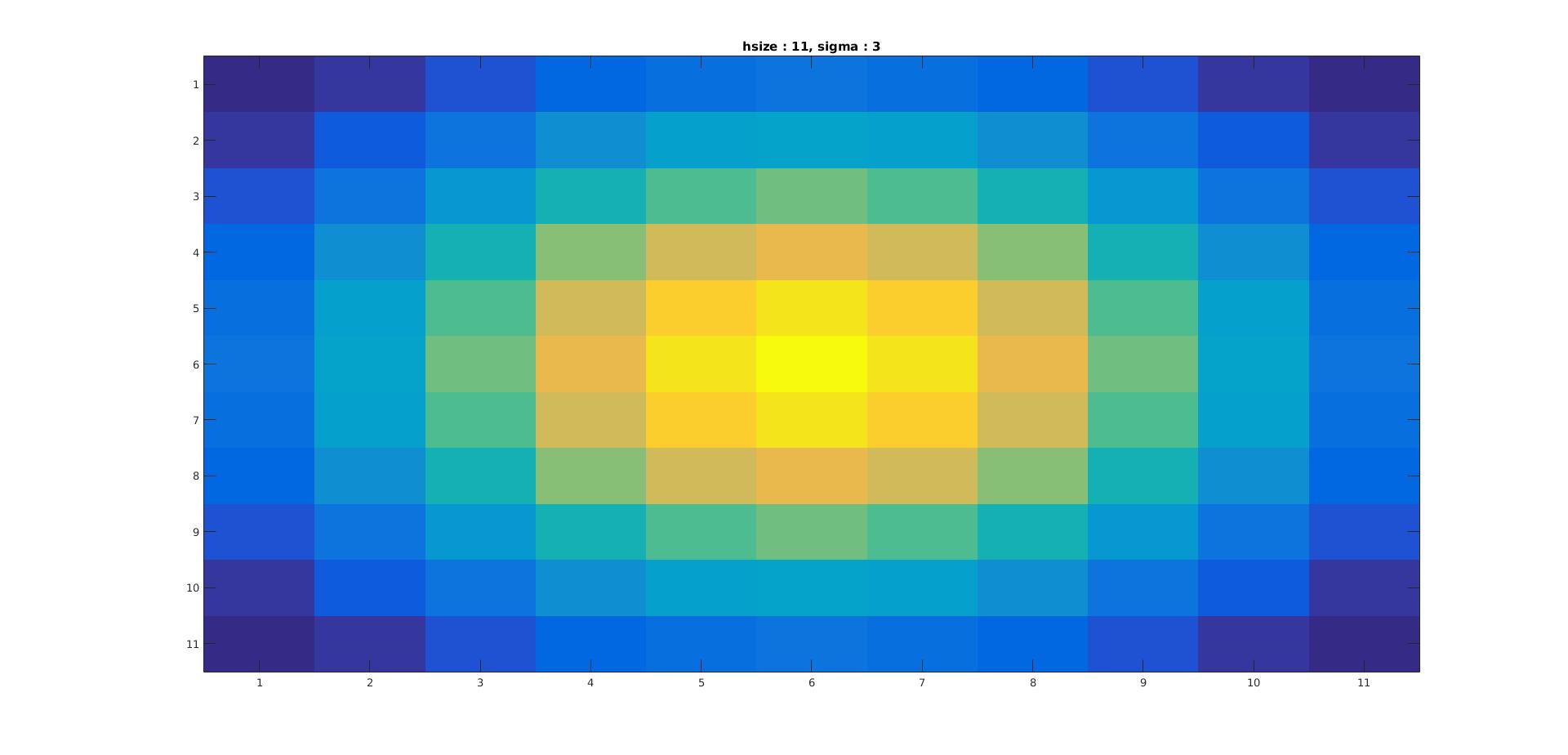
figure(i)

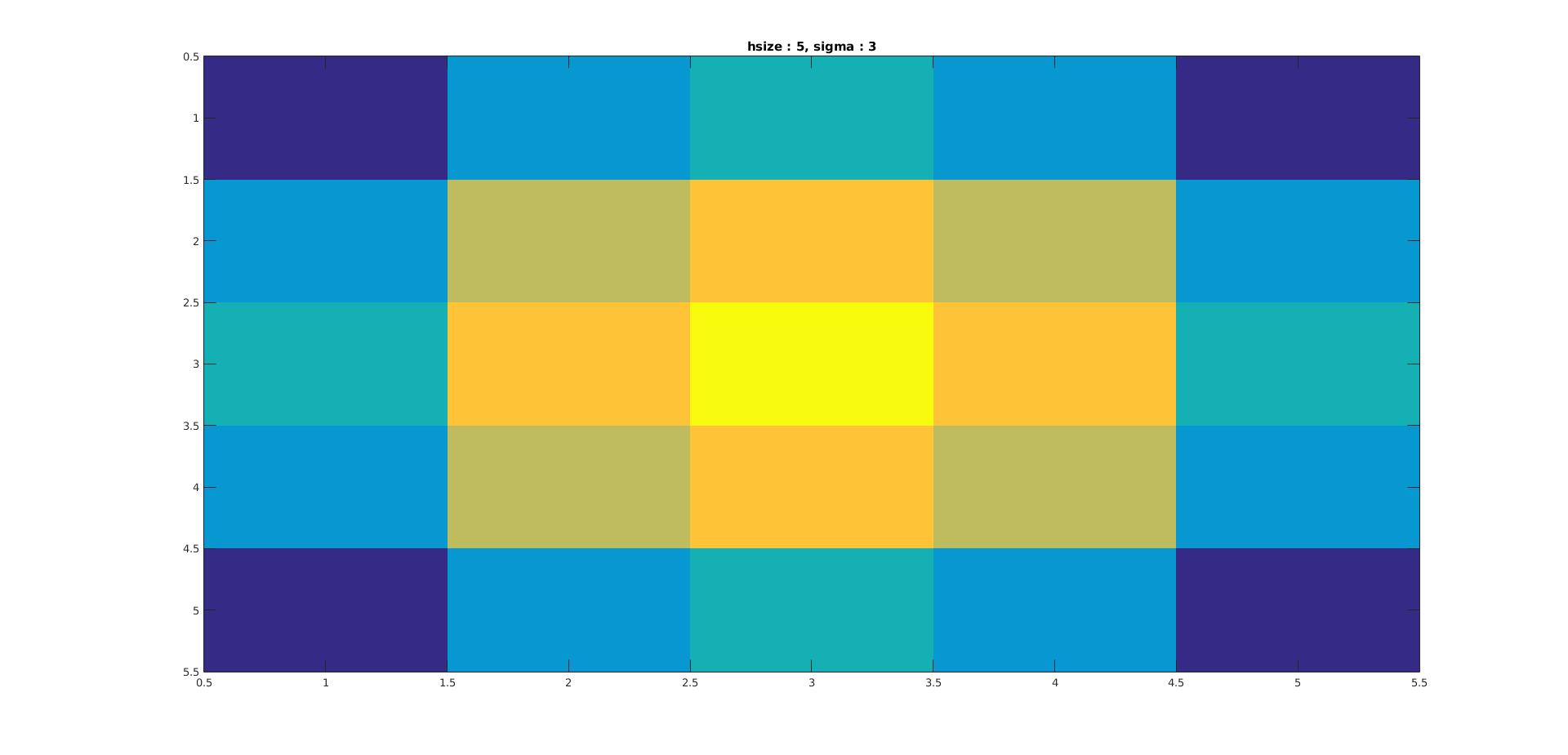
imagesc(fspecial('gaussian',hsize(i),sigma(i)))

title(sprintf('hsize : %d, sigma : %d',hsize(i),sigma(i)))

end







When hsize is small and sigma is really large, all cells on the image will almost have the same color, as the gaussian function changes more slowly with large standard deviation.

Report Item 8:

function [out\_image] = gaussianFilter(image,kernelSize,std)

image\_lab = rgb2lab(image)

kernel = fspecial('gaussian',kernelSize,std)

first = image\_lab(:,:,1)

con = conv2(first,kernel,'same');

image\_lab(:,:,1) = con;

out\_image = lab2rgb(image\_lab);

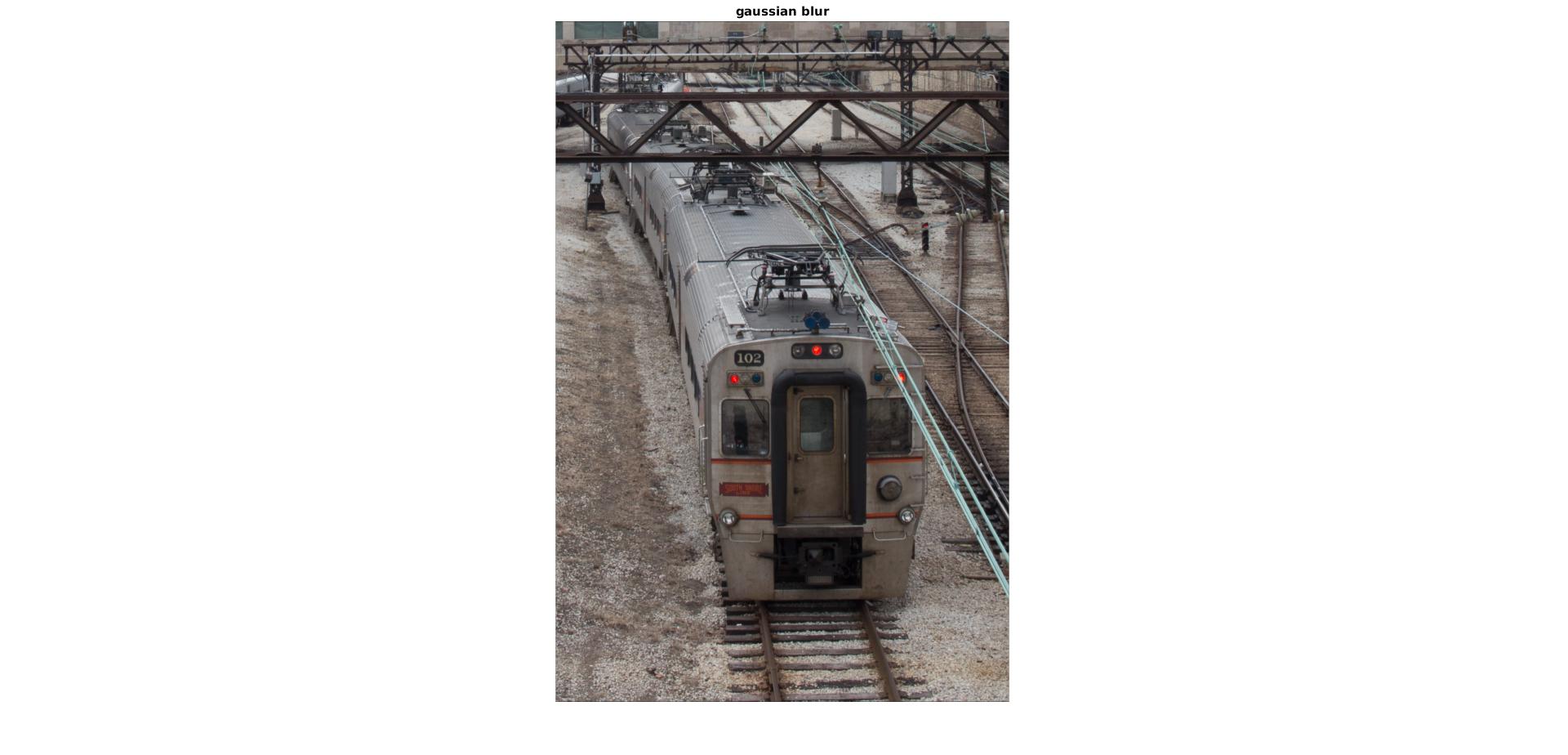
end

train = imread('train.jpg');

out = gaussianFilter(train,7,1);

imshow(out)

title('gaussian blur')



Report Item 9:

function [sharpened] = unsharpMask(original,lowpass,alpha)

ori\_L = rgb2lab(original);

low\_L = rgb2lab(lowpass);

lab\_sharp = ori\_L;

lab\_sharp(:,:,1) = ori\_L(:,:,1) - alpha\*(ori\_L(:,:,1)-low\_L(:,:,1));

sharpened = lab2rgb(lab\_sharp);

end

train = imread('train.jpg');

meanK\_first = unsharpMask(train,meanFilter(train,7),2);

meanK\_second= unsharpMask(train,meanFilter(train,50),2);

gaussian\_first=unsharpMask(train,gaussianFilter(train,7,1),2);

gaussian\_second=unsharpMask(train,gaussianFilter(train,50,1),2);

figure(1)

subplot(121)

imshow(meanK\_first);

title('mean filter, kernelsize= 7');

subplot(122)

imshow(meanK\_second);

title('mean filter,kernelsize = 50');

figure(2)

subplot(121)

imshow(gaussian\_first);

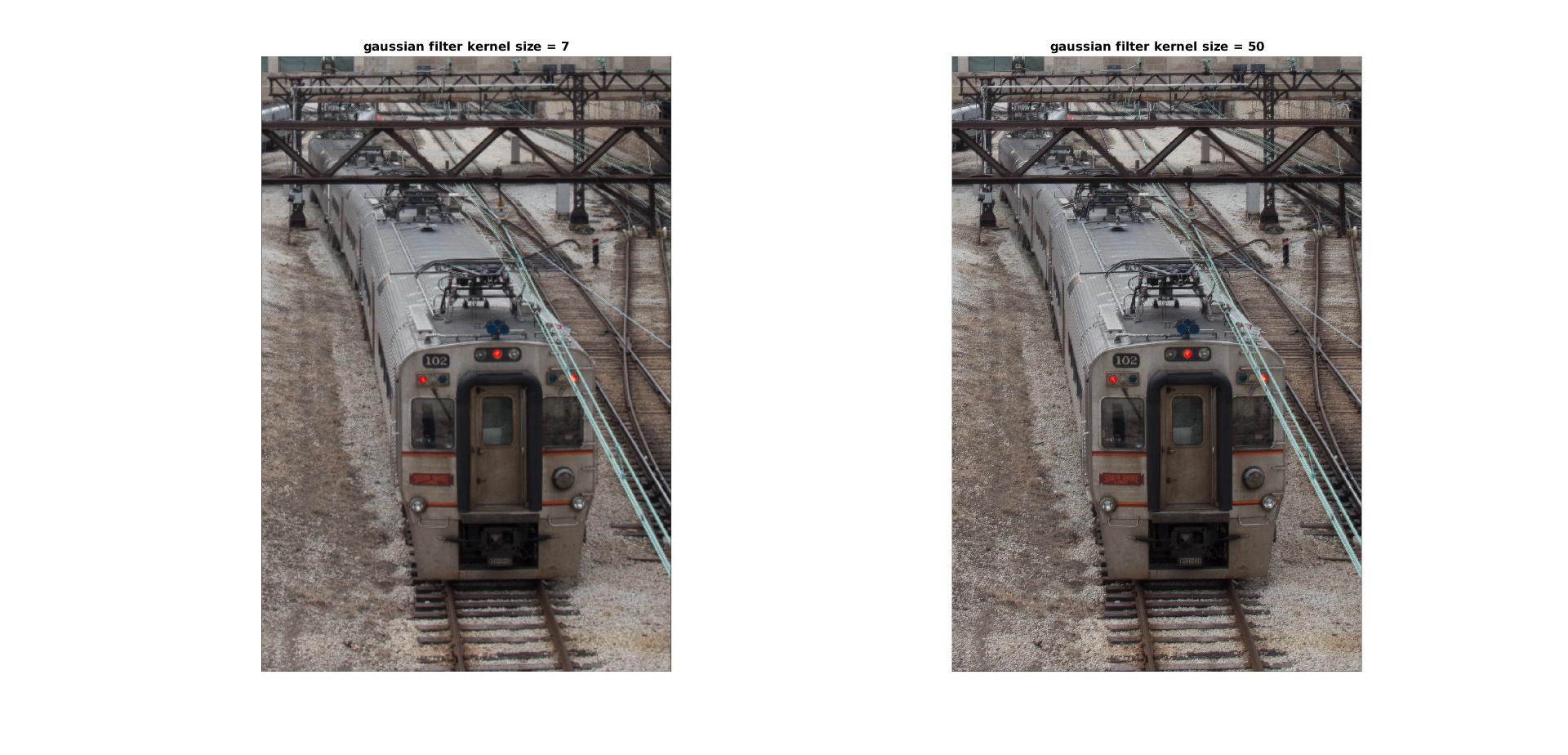
title('gaussian filter kernel size = 7');

subplot(122)

imshow(gaussian\_second);

title('gaussian filter kernel size = 50')





When the filters are of the same size but different type, the result generated by the meanFilter tends to sharpen the image more, as the contour is more emphasized.

When the images are both generated by the same type of filter, there is huge difference the image gets sharpened, as the one with larger kernel size gets sharpened more. When the gaussian filter is used,difference between the results are little harder to see.