Color and Orientation.

1. The test image and displaying color in HSV
Generate a test image by running the function *fmtest* (included in the exercise).
Display the *magn* image (in true size) and the *argument* image in gray colors
The header of the function fmtest documents it in further detail (**help fmtest**).

[magn, argument]=fmtest(256,[0.1,0.33]*pi, 1); %generate testimage
figure(1); imagesc(magn); colormap(gray); truesize; %display
figure(2); imagesc(argument); colormap(gray); truesize;

Interpret the magnitude image magn on a line through origin according to local frequency and local orientation.
In the image argument, why is there a discontinuity?
(Hint: how is the argument 0→2π coded?).

The function *lsdisp* (included in the exercise) can display a 3 component *real image* by using HSV (Hue, Saturation and Value) color model. Please note that the computer screen is hard wired to work in RGB model (Red, Green and Blue). A Red, Green and Blue triplet directly steers the Red, Green and Blue electron intensities of the cathode ray tube (or the corresponding color TFT cells if you have a flat screen) so that an exact color pixel is obtained on a point in the screen. In other words the pixels on a color monitor are 3 dimensional vectors that are directly steered by a triplet of RGB.
What the function lsdisp does is to convert the Hue, Saturation and Value triplet to the corresponding RGB values and then send this RGB triplet to the electron guns so that each time you want to get the impression of an HSV triplet at a pixel you also get it without hassle.
Using a similar conversion, the function lsdisp can also display a *complex valued image* i.e. the pixels are complex (the pixels are two dimensional vectors). *In this case the argument of the complex pixel steers the Hue component, the magnitude steers the Value component whereas the Saturation component is put to its maximum, which is 1 at all pixels*. The resulting HSV triplet is converted to an RGB triplet and sent to the electron guns of the screen. The header of the function lsdisp, documents it in greater detail (**help lsdisp**).

Generate a constant image *magnone* that equals to 1, having the size of image *magn*. In a new figure, display *magnone* and *argument* images jointly by using lsdisp.
There are two options in lsdisp:
-display a three component real valued image
-display a complex valued image
try both options.

magnone=ones(256,256); %constant image=1
%display a three component real valued image im_r
im_r(:,:,1)=argument; %steers H,
im_r(:,:,2)=magnone; %steers V,
im_r(:,:,3)=ones(256,256); %steers S (is put to 1)
figure(3); lsdisp(im_r); truesize;

%display a complex image im_c
%angle(im_c) steers H, abs(im_c) steers V,
%and S is put to 1 in the lsdip function
im_c=magnone.*exp(i*argument); %make a complex image;
figure(4); lsdisp(im_c); truesize; %and display it

Do you see any discontinuity in color?
Why not? (Hint: how is the argument 0→2π coded in the HSV color model?).
How are the arguments of two points that are symmetric with respect to origin related to each other?

Display the image *magnone* and the image 2**argument* jointly by using lsdisp in figure(17). Put figure handle immediately back to another figure, e.g. issue figure(5), and do not delete figure(17) or display anything in figure(17) since you will need to inspect it later.
Im2Arg1=magnone.*exp(i*2*argument); %generate "2*argument" image
figure(17); lsdisp(Im2Arg1); truesize;
figure(5);

In figure(17) any two points that are symmetric with respect to origin have the same color!
*Why? (Hint: write down the 2*argument of the symmetric points).*

Modulate now *magn* with the 2*argument via lsdisp.
Im2Arg=magn.*exp(i*2*argument); %generate "2*argument" image
figure(5); lsdisp(Im2Arg) ;truesize;

*What does the color represent in Im2Arg when color is modulated by 2*argument?*
(Hint: how did you interpret the local orientation on a line through origin in the image magn?)

2. Orientation
When estimating a feature in an image (here: the local orientation of the pattern in the image *magn*) it is important to do local averaging to obtain a robust estimation of the feature.

Study the gradient image at different positions of the image magn by running the function *showex_gr* (included in the exercise). Ten positions in one run can be studied.
Present in the report the gradients at a position in the mid frequency band.
gradient=showex_gr(magn); %shows gradient in image magn

Do local averaging in the *gradient* image by using a large gaussian with σ=2.5.
%design 1D gaussian filters
std=2.5;
x=-round(3*std):round(3*std);
gx=exp(-(x.*x)/2/std/std);
gx=gx/sum(gx); %in the x direction
gy=gx'; %and in the y direction

%do the local averaging on the gradient image
avg_grad=filter2(gy,filter2(gx,gradient));

%display the avg_grad image by using lsdisp and option complex input
figure(51);lsdisp(avg_grad,3.5,'scloff'); truesize;

Is it possible to do local averaging in the gradient image to estimate the local orientation of the pattern? Compare your result with figure(17).
Why not?

Use "2*argument" of the gradient image, this new image is called the orientation image.
Do the changes in the function showex_gr to compute the orientation image and run it to study the orientation image at different positions.
Present in the report the orientations at a position in the mid frequency band.
orientation=showex_gr(magn); %shows orientations in image magn

What is the main difference between the gradient and the orientation image?

Do local averaging in the *orientation* image.
Display the orientation image by using lsdisp and the complex input option.
avg_or=filter2(gy,filter2(gx,orientation)); %do local averaging
figure(52); lsdisp(avg_or); truesize; %and display

Is it now possible to do local averaging in the gradient image to estimate the local orientation of the pattern? Compare your result with figure(17).
Why?