

Orientation estimation.

1. The test image

Generate a few test images by changing the variable *A* below according to the values $A=1$, $A=0.75$, and $A=0.5$ in the function *fmtest* (included in the exercise) while displaying the corresponding *magn* images (in true size) in gray colors.

The header of the function *fmtest* documents it in further detail.

```
[magn, argument]=fmtest(256,[0.1,0.33]*pi, A); %give A a value!  
imagesc(magn); colormap(gray); truesize; %display the magnitude image
```

Display one of the argument images.

```
imagesc(argument); colormap(gray); truesize;
```

In image magn:

How do neighborhoods around two points, symmetrically located with respect to the origin, compare to each other when A is changed?

Compare the neighborhoods according to their linear symmetry properties.

Generate a constant image *magnone* that equals to 1, having the size of image *magn*.

Display the image *magnone* and the image $2*argument$ jointly by using *lsdisp* (included in the exercise) in figure 17.

Figure 17 shows how local orientations are coded by color. Don't overwrite figure 17, because you need to inspect this figure later on.

```
magnone=ones(256,256); %1-magnitude image  
im1_c=magnone.*exp(i*2*argument); %complex image  
figure(17); lsdisp(im1_c); truesize; %color code for local orientations
```

2. Orientation estimation by Linear Symmetries (LS)

Complete a matlab function that implements *linear symmetry estimation algorithm*, which is the *local orientation of an image*. Do this by filling in the dots (...) in the code *linsymexer.m* (provided).

a) Run the code on image *magn* (with $A=0.75$) and display the result with *lsdisp* by using the values $B=0.5$, $B=1.5$, and $B=3.5$.

```
[magn, argument]=fmtest(256,[0.1,0.33]*pi, 0.75));  
LS=linsymexer(magn); %estimate the linear symmetry (LS) which contains both I20 and I11  
lsdisp(LS, Gamma); truesize; %and display (give Gamma, a value for example 1!)
```

-Inspect the code of the linsymexer and the lsdisp functions and write down how the "linear symmetry parameters" I20 (complex) and I11 (scalar) are stored in the three component real valued image LS (linsymexer).

-Also, write down how these three local image measurements are steering the HSV color model (lsdisp).

Is Gamma part of the measurements in image or it only influences the way we see LS (that is I20 and I11)?

Are the orientations correct? Tips: compare your result with figure(17).

Is there a specific frequency where the function linsymexer is stronger than other frequencies?

Hint: You can increase Gamma=3 to see the orientations of the frequencies that remain visible/strong. You can influence the "strongest" frequency by changing sma1 in linsymexer, e.g.

by calling it with 2 variables.

Can the orientation estimate in noisy part made be more visible by displaying LS image differently?

Hint: what happens to Saturation (=abs(i20)) and Value (=i11), when you decrease Gamma in comparison to each other? Try to explain the differences in what you see when changing Gamma, in the following parts of the image: in the left half-circle (clean signal), in the right half-circle (signal+noise), outside of the half-circle (clean constant). It is easiest to draw conclusions if you assume that the highest value of I11 in the entire image is 1 and that |I20| never exceeds I11.

b) Run the linsymexer code on image *fingerp.tif*:

```
fim=double(imread('fingerp.tif','tif')); %read the image
imagesc(fim);colormap(gray);axis image; %and display it
LS2=linsymexer(fim,[1.5,2]); %estimate LS
lsdisp(LS2,0.5); truesize; %and display
```

What does color represent in the LS2 image? (Hint: check the color against figure 17).

The input parameters [1.5,2] to the linsymexer function are the filter parameters used to compute the LS-variables i20 and i11.

Which parameter is used for the derivative filters respective the smoothing filters?

Make a corner (minutiae in finger-prints) detector by detecting 'remof linear symmetries.

```
mnt=LS2(:,3)-LS2(:,2);
```

```
%Display the result by imagesc and by modulating your response mnt with
%the original image via lsdisp:
```

```
'lracpf ao pv? llo 0 gxr *k 9, o pv=
ndisp(lracpf ao pv); truesize;
```

''

Check what lsdisp is displaying in the hue when the input (hsvim) is complex.

What does the color red represent--does it mean that we have LS or not have LS at red pixels?

Did your detector succeed in coloring most minutiae with another color than red?

Hint: mnt can be negative but never complex..but fp_and_mnt is complex.

c) Run the linsymexer code on image *wood+chain.tif*:

```
fim=double(imread('wood+chain.tif'));
figure(11);
subplot(2,2,1);imagesc(fim); colormap(gray); axis image;
subplot(2,2,2); lsdisp(linsymexer(fim,[0.4 4]),0.2); axis image;
subplot(2,2,3); lsdisp(linsymexer(fim,[6 10]),0.5); axis image
```

Study the two orientation images.

Compare your result with figure 17 (local orientation is coded by color!) and explain which orientations are extracted in respective image.

Also, explain why different orientations are extracted in the two images.

(Hint: the input filter parameters to the linsymexer function are changed).

What effect had when you changed the std parameters of the filters has (on local image frequency)?

Comment: Bandpass filtering can also be done in the Laplacian pyramid. Orientation estimation can then be done for each level in the pyramid with the same (small) std parameters of the filters used in the linsymexer function (compared to changing the std parameters) yielding a more effective implementation from computation point of view.