Multi-scale Steerable Phase-Symmetry Filters for ITK

Release 0.00

Charles Hatt

December 7, 2011

University of Wisconsin – Madison, hatt@wisc.edu

Abstract

We have implemented filters for computing phase symmetry (PS) in ITK using steerable filters computed and applied in the image Fourier domain. In some recent publications, phase symmetry has been shown to be a useful pre-processing measure for performing image registration and segmentation. Matlab implementations of the code are currently available, but may be slower than C++ versions and most importantly, do not work with volumetric (3D) data. In this paper we briefly provide background information and list some applications of PS filters from the literature. We then provide instructions and examples of how to use the classes and executables.

Contents

1 Local-phase measures and applications Error! Bookmark not defined.

2 Implementation Details 3

3 Example 4

4 Software Requirements 6

1 Local-phase measures and their applications

Phase-Symmetry (PS) is a dimensionless measure of symmetry defined at each point in a n-dimensional signal. It belongs to a broader class of local signal measures often referred to in the literature as 'local-phase' (LP), which also includes phase-congruency (PC). LP measures were first introduced in [1], and then further developed in [2],[3],[4],and[5].

The concepts behind the PS measure are illustrated in Figure 1. The Fourier series components of a square and triangle are shown. Notice that at the axis of symmetry, the Fourier components are all at local optima, while at the axis of asymmetry, they are all at local inflection points.

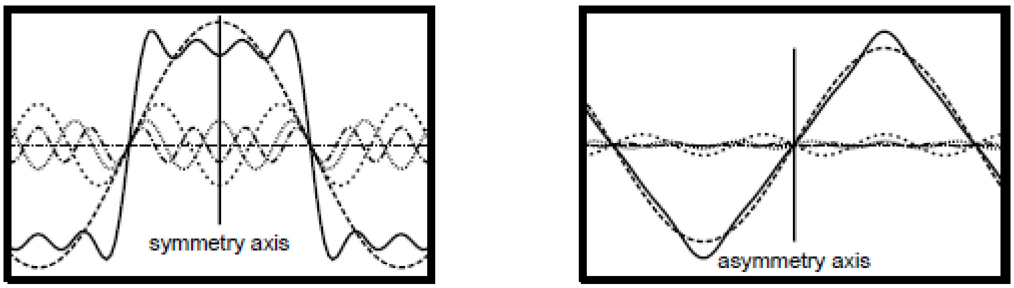


Figure 1 Fourier components of square and triangle functions. The axis of symmetry for the square function contains optima of the Fourier components, and the axis of asymmetry for the triangle function contains inflection points of the Fourier components. The figure is taken from [2].

To derive local phase measures used to compute PS, quadrature pair wavelets are used for performing local frequency analysis. Specifically, directional log-gabor wavelets are used, as specified in [2] and suggested in [6]. Log-gabor wavelets are useful because they allow the creation of large-bandwidth filters that maintain a zero-DC response.

Filtering is accomplished by pixel-wise image multiplication in the Fourier domain. In order to facilitate the easy creation of these filters, two image source classes were created: `itk::LogGaborFreqImageSource` and `itk::SteerableFilterFreqImageSource`. `itk::LogGaborFreqImageSource` is used to create log-gabor filter kernels in the frequency domain, and `itk::SteerableFilterFreqImageSource` is used to created functions that, when multiplied with filters in the frequency domain, 'steer' them so that they have directionality. Another class, `itk::ButterworthFilterFreqImageSource`, is used to create frequency domain low-pass butterworth filter kernels.

Multiplication with these filters in the Fourier domain produces even and odd representations of the image. The PS measure is defined as the even part of the signal minus the odd part of the signal summed at each scale, normalized by the sum of the magnitude of the filter response at each scale:

$$PS = \frac{\sum^{O}\sum^{N}\left(\lfloor Energy(n,o) - T\rfloor\right)}{\sum^{O}\sum^{N}A(n,o)}$$

$$Energy(n,o) = \left|real\left(i\left(x\right) * g\left(x\right)\right)\right| - \left|imag\left(i\left(x\right) * g\left(x\right)\right)\right|$$

$$A(n,o) = \left|i\left(x\right) * g\left(x\right)\right|$$

$$T = Noise\,Threshold$$

$$i\left(x\right) = image$$

$$i\left(x\right) = gaborfilter$$

Latest version available at the Insight Journal link http://hdl.handle.net/10380/xxxx
Distributed under Creative Commons Attribution License

Below is a brief, highly under-sampled selection of research that utilizes local phase measures for medical imaging applications:

- In [7], the authors propose mutual information between LP images for improved medical image registration.
- In [8], the authors register real-time, 3D ultrasound from multiple views using LP filters applied to the images
- In [9], the authors localize the bone surface in 2D ultrasound images using PS features
- In [10], the authors register 3D ultrasound bone images to CT in near real-time using PS filters applied to the ultrasound images.

2 Implementation Details

The main class for performing PS filtering is `itk::PhaseSymmetryImageFilter`, derived from `itk::ImageToImageFilter`. The code for this filter was adapted from matlab code found at [11]. The filter has six parameters: wavelengths, orientations, sigma, angular bandwidth, polarity, and noise threshold. Wavelengths and orientations are `itk:Array2D` objects, where the number of rows are equal to the number of spatial scales (wavelengths) or filter orientations, and the number of columns are equal to the image dimension. Orientations are represented as vectors. For example, in a 2D image filtered at 0 and 90 degrees, the orientation matrix should be specified as [1 0; 0 1]. Sigma controls the frequency bandwidth of the log-gabor filters, and the angular bandwidth controls the width of the angular spreading function of the log-gabor filters, where the bandwidth is equal to the width of the function at half maximum. Polarity controls where the filter responds to dark areas (-1), light areas (1), or both (0). The noise threshold is used for multiscale noise reductions, where the noise should be the filter response to noise at the lowest possible filter wavelength. This value should be determined empirically for the type of image used. A more detailed explanation is found in [4].

Because the PS filter uses a bank of wavelet filters that should not need to be created every time the filter is updated, there is a member function `Initialize` that needs to be called before `Update` is called. The `Initialize` routine creates the bank of wavelet filters that is used by the `GenerateData` method. The advantage of doing things this way is that, for real-time applications, there is a large time savings by only have to create the filter bank once.

The classes are also bundled with executable for applying the filters to an image and saving the output. PhaseSymmetryImageFilter3D.exe can be run from the command line with the following usage:

```
Usage: PhaseSymmetryFilter3D.exe infile outfile wavelengths orientations sigma
angular_bandwidth polarity noise_threshhold
```


Distributed under Creative Commons Attribution License

```
Example: PhaseSymmetryFilter3D.exe i.mhd o.mhd 3,3,3,6,6,6,12,12,12 1,0,0,0,1,0,0,0,1
0.55 3.14 0 10.0
```

PhaseSymmetryImageFilter2D.exe has the same usage but the wavelengths and orientations strings should contain an amount of numbers that is divisible by 2, not 3.

An important consideration when using this filter is that it will be replicated in memory greater than N x O times, where N=wavelengths and O =orientations. It therefore can potentially consume a large amount of memory when lots of scales and orientations are used. When tested with and image volume that was sized 256x256x256 pixels, over 2 scales and 3 orientations, on a machine with 8GB of RAM the application ran out of memory.

Also, since the filter uses the Vnl FFT, images must have dimensions that are powers of 2. The filter will be modified to use FFTW in the future, once the FFTW FFT implementations in ITK produce output that is of equal size to the input. This will increase the speed and allow arbitrary image sizes.

PS has values between 0 and 1, so the data will need to be scaled if it is saved as integer data.

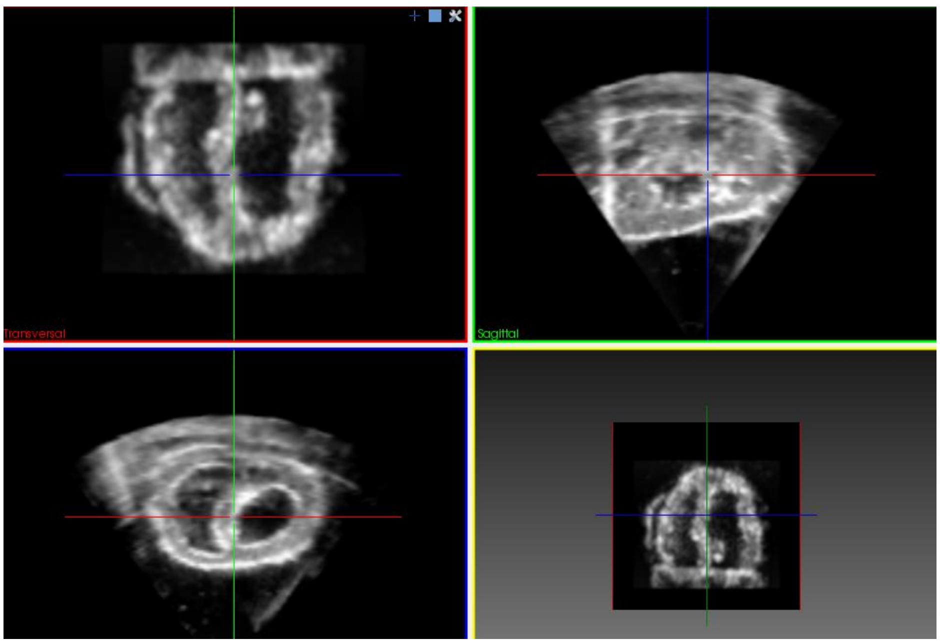
3 Example

The following code shows an example of how to use the filter at isotropic wavelengths of 10.0, 20.0 and 40.0 pixels, at three directions (x,y,z), with a Sigma value of .40, angular bandwidth value of pi,enhancing light spots, with a noise threshold of 10.0. The input data is a 128x128x128 3D ultrasound image of a cardiac phantom. The input and output is shown in Figure 3.

```
typedef itk::Array2D<double> DoubleMatrix;
typedef itk::PhaseSymmetryImageFilter< ImageType, ImageType > PSFilterType;
PSFilterType::Pointer psfilter = PSFilterType::New();
typedef itk::ImageFileWriter< ImageType > WriterType;
WriterType::Pointer writer= WriterType::New();
DoubleMatrix wavelengths(3,3);
DoubleMatrix orientations(3,3);
wavelengths(0,0)=10.0;
wavelengths(0,1)=10.0;
wavelengths(0,2)=10.0;
wavelengths(1,0)=20.0;
wavelengths(1,1)=20.0;
wavelengths(1,2)=20.0;
wavelengths(2,0)=40.0;
wavelengths(2,1)=40.0;
wavelengths(2,2)=40.0;
orientations(0,0)=1.0;
orientations(0,1)=0.0;
orientations(0,2)=0.0;
orientations(0,0)=0.0;
orientations(0,1)=1.0;
orientations(0,2)=0.0;
orientations(0,0)=0.0;
orientations(0,1)=0.0;
orientations(0,2)=1.0;
```


Distributed under Creative Commons Attribution License

```
double sigma=.40;
double anglebandwidth=pi;
double polarity=1;
double noiseT=10.0;
psfilter->SetInput(inImg);
psfilter->SetWavelengths(wavelengths);
psfilter->SetOrientations(orientations);
psfilter->SetSigma(sigma);
psfilter->SetAngleBandwidth(anglebandwidth);
psfilter->SetPolarity(polarity);
psfilter->SetT(noiseT);
psfilter->Initialize();
writer->SetFileName(outfile.c_str());
writer->SetInput(psfilter->GetOutput());
writer->Update();
```



Distributed under Creative Commons Attribution License

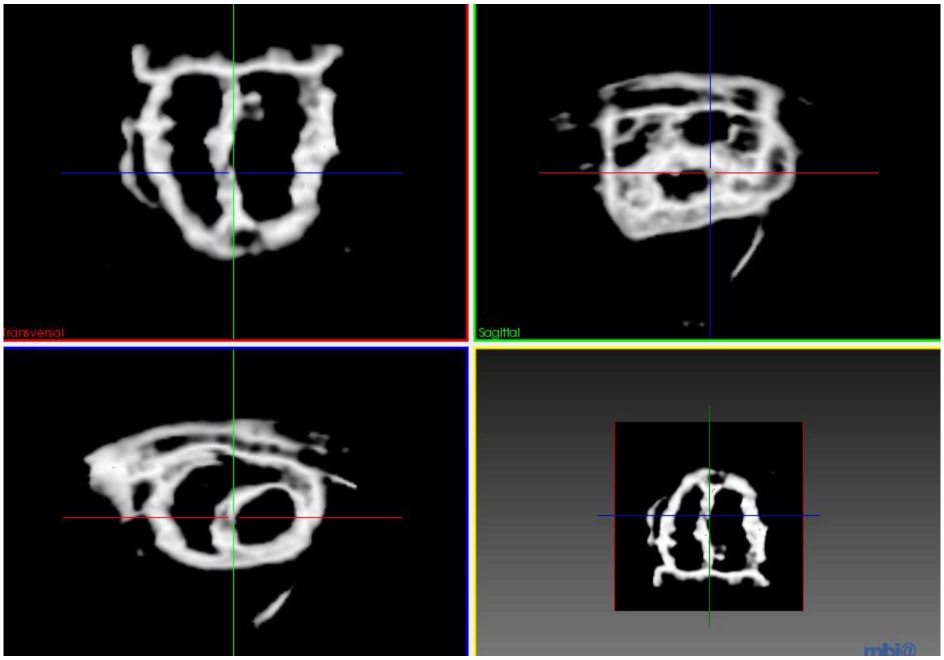


Figure 2 Input and output images for the PS filter. The input is a 3D ultrasound of a cardiac phantom. Notice the reduction of noise and enhancement of structure.

4 Software Requirements

You need to have the following software installed:

- Insight Toolkit 3.20.
- CMake 2.8.4
- Future versions may require FFTW 3.3

Note that other versions of the Insight Toolkit are also available in the testing framework of the Insight Journal. Please refere to the following page for details

http://www.insightsoftwareconsortium.org/wiki/index.php/IJ-Testing-Environment

Reference

[1] M.C. Marrone, R.A. Owens. *Feature Detection from Local Energy*. Pattern Recognition Letters, 1987.

Distributed under Creative Commons Attribution License

[2] P. Kovesi. *Symmetry and Asymmetry From Local Phase*. AI'97, Tenth Australian Joint Conference on Artificial Intelligence. 2 - 4 December 1997. Proceedings - Poster Papers. pp 185-190.

[3] P. Kovesi. *Image Features From Phase Congruency.* Videre: A Journal of Computer Vision Research. MIT Press. Volume 1, Number 3, Summer 1999.

[4] P. Kovesi. *Edges Are Not Just Steps.* Proceedings of *ACCV2002 The Fifth Asian Conference on Computer Vision*, Melbourne Jan 22-25, 2002. pp 822-827.

[5] P. Kovesi. *Phase Congruency Detects Corners and Edges.* The Australian Pattern Recognition Society Conference: DICTA 2003. December 2003. Sydney. pp 309-318.

[6] D.J. Field. *Relations Between the Statistics of Natural Images and the Response Properties of Cortical Cells.* Journal of the Optical Society of America: Volume 4. Issue 12: 2379-2394. December 1987.

[7] M. Mellor. M. Brady, *Phase mutual information as a similarity measure for registration.* Medical Image Analysis. Volume 9, Issue 4. 330-343. August 2005.

[8] V. Grau. H. Becher, J.A. Noble. *Registration of Multiview Real-time Echo Cardiographic Sequences.* IEEE Transactions in Medical Imaging. Volume 26, Issue 9. 1154-1165. September 2007.

[9] I. Hacihaliloglu, R. Abugharbieh, A.J. Hodgson, R.N. Rohling. *Bone Surface Localization in Ultrasound Using Image Phase-Based Features.* Ultrasound in Medicine and Biology. Volume 35, Issue 9. 1475-1487. September 2009.

[10] A. Brounstein, I. Hacihaliloglu, P. Guy, A. Hodgson, R. Abugharbieh. *Towards Real-Time 3D US to CT Bone Image Registration Using Phase and Curvature Feature Based GMM Matching*. Medical Image Computing and Computer Aided Interventions. Volume 6891. 235-242. September 2011.

[11] P. Kovesi. *MATLAB and Octave Functions for Computer Vision and Image Processing*. Centre for Exploration Targeting School of Earth and Environment. The University of Western Australia. Available from:<http://www.csse.uwa.edu.au/~pk/research/matlabfns/>

Distributed under Creative Commons Attribution License

Latest version available at the Insight Journal link http://hdl.handle.net/10380/xxxx
Distributed under Creative Commons Attribution License