Phase Correlation Method for ITK

Release 1.1

Jakub Bican[1]

December 14, 2006

[1]jakub.bican@matfyz.cz, Department of Image Processing, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic

Abstract

Phase Correlation Method (PCM or SPOMF - Symmetric Phase-Only Matched Filter) is a well known image registration method, that exploits Fourier Shift Theorem property of Fourier Transform. Even though it is limited to estimate only shifts between two images, it is very useful as it is robust to frequency-dependent noise and initial image overlap area. Furthermore, it evaluates very fast by way of two forward FFTs, one complex image division and one inverse FFT.

This document describes the implementation of Phase Correlation Method for the Insight Toolkit ITK www.itk.org. The implementation splits the algorithm in several classes to enable further improvements to the method to be implemented (such as different sub-pixel algorithms). The paper is accompanied with the source code, tesing and example code and input data for purposes of testing of this implementation.

1 Theoretical background

Phase correlation method [16] takes advantage of *Fourier shift theorem* that relates the phase information of an image and its shifted copy. If f_M is shifted copy of f_F:

$$f_M(\vec{x}) = f_F(\vec{x}+\Delta\vec{x}),$$

then according to the shift theorem, the Fourier transforms of the images are related as follows:

$$F_M(\vec{\omega}) = F_F(\vec{\omega})e^{i(\omega_x\Delta x_x+\omega_y\Delta x_y+\omega_z\Delta x_z)},$$

and

$$\frac{F_M(\vec{\omega})}{F_F(\vec{\omega})} = e^{i(\omega_x\Delta x_x+\omega_y\Delta x_y+\omega_z\Delta x_z)}. \tag{1}$$

Inverse Fourier transform of equation 1 gives us a *correlation surface* $Corr(\vec{x})$ which has a form of Dirac delta function located at $\Delta\vec{x}$:

$$Corr(\vec{x}) = \mathfrak{F}^{-1}\left(\frac{F_M(\vec{\omega})}{F_F(\vec{\omega})}\right) = \mathfrak{F}^{-1}\left(e^{i(\omega_x\Delta x_x+\omega_y\Delta x_y+\omega_z\Delta x_z)}\right) = \delta(\vec{x}+\Delta\vec{x}). \tag{2}$$

Thus, locating a peak in a correlation surface results in offset $\Delta\vec{x}$ that can be used to align the two images at pixel-level.

Quotient of spectrums F_M and F_F in equations 1 and 2 is in practise (when f_M is not exactly the shifted copy of f_F) usually computed as

$$\widehat{Corr}(\vec{\omega}) = \frac{F_M F_F^*}{|F_M|\,|F_F|},$$

so that PCM computes the correlation of whittened images (images with $|F| := 1$). $\widehat{Corr}(\vec{\omega})$ is a Fourier transform of correlation surface $Corr(\vec{x})$ and is usually called *cross-power spectrum*.

2 Principles of PCM implementation

The implementation is based on basic PCM principle described above, but is dealing with arbitrarily sized and spaced images. It also enables implementation of various enhancements published later.

The basic PCM algorithm has 4 (5) steps:

0. Resample Fixed and Moving image f_F and f_M to same physical size and resolution: images f_F^{rs}, f_M^{rs}.
1. Compute forward FFT of the images: spectra F_F^{rs}, F_M^{rs}.
2. Compute the ratio of the two spectra and get complex (frequency domain) cross-power spectrum $\widehat{Corr}(\vec{\omega})$.
3. Compute inverse FFT of the complex cross-power spectrum and get real correlation surface $Corr(\vec{x})$.
4. Find the maximum peak in the correlation surface and determine the shift $\Delta\vec{x}$.

The 0th step is not necessarily the part of the algorithm, but it is a prerequisite for further operations. The images must have same size and should have same real size (i.e. spacing) to reach meaningful results.

First, the image with smaller real size in certain dimension is padded by zeros to reach the size of the other image in that dimension. This is done for all dimensions in one step. The next step should be resampling of one image to have the same spacing as the second image. This is actually done after forward FFTs by cropping high frequencies in some dimension from an image which has higher resolution in that dimension. This is done implicitly for all dimensions by simply omitting those high frequencies from computation during 2nd step.

The 2nd step is implemented in separate class called *operator* that is plugged in the main class during run time. It is the first point of possible extensions. It enables for example frequency-dependent filtering and other techniques.

The 3rd and 4th step are called *optimization* as they have to determine the shift from the cross-power spectrum. Optimization is carried by *optimizer*, which is another pluggable component of the method and the second point of possible extensions. As some optimizers work on complex cross-power spectrum (for example [7]) and some others on real correlation surface (as the maximum-peak method introduced by original paper [16] and implemented here), the method incorporates the inverse FFT filter according to the input image type of the optimizer.

3 Usage

The method itself is implemented in class `PhaseCorrelationImageRegistrationMethod` and it is in fact a mini-pipeline that consists of two `itk::PaddImageFilter`s

that padd the images and cast data of integral types to real types, two itk::FFTRealToComplexConjugateImageFilters, one PhaseCorrelationOperator (or descendant), possibly one itk::FFTComplexConjugateToRealImageFilter and one descendant of PhaseCorrelationOptimizer (for example MaxPhaseCorrelationOptimizer).

Before execution, it is necessary to set fixed image, moving image, operator and optimizer to the PhaseCorrelationImageRegistrationMethod's instance. The input images may be of any dimension and any scalar data type that is convertible to some real type and may have different size, spacing and real size, since they are automatically resampled to the same proportions as described above.

The method is invoked normally by Update() or by updating some downstream filter. The result can be obtained in form of itk::TranslationTransform as filter output, that is passable down the pipeline, or in form of translation parameters that describe the shift in every dimension. The transformation can be directly used to transform the Moving image to match the Fixed image.

Users familiar with itk::ImageRegistrationMethod will notice similar interface of the main class PhaseCorrelationImageRegistrationMethod.

4 Example

This section describes the example code attached with this contribution in PhaseCorrelationImageRegistrationMethodExample.cxx. With this example, you can use BrainProtonDensitySliceBorder20.png and BrainProtonDensitySliceShifted13x17y.png images from ITK's example data set. These images are attached with this source and testing code as well.

First, make the standard includes amd check the command line arguments.

```
#include "itkPhaseCorrelationImageRegistrationMethod.h"
#include "itkPhaseCorrelationOperator.h"
#include "itkMaxPhaseCorrelationOptimizer.h"

#include "itkImageFileReader.h"
#include "itkResampleImageFilter.h"
#include "itkCastImageFilter.h"
#include "itkImageFileWriter.h"


int main( int argc, char *argv[] )
{

  if (argc<4)
    {
    std::cout << "Usage: " << std::endl;
    std::cout << argv[0] << "  FixedImageFile  MovingImageFile"
                         << "  OutputImageFile" << std::endl;

    return EXIT_FAILURE;
    }
```

The second standard step is to declare the types and read the images.

```
  const unsigned int                          Dimension = 2;
```

```
typedef float                                PixelType;
typedef itk::Image< PixelType, Dimension >  ImageType;

typedef itk::ImageFileReader<ImageType>  ReaderType;

ReaderType::Pointer fixedReader  = ReaderType::New();
ReaderType::Pointer movingReader = ReaderType::New();
fixedReader->SetFileName(  argv[1] );
movingReader->SetFileName( argv[2] );

try
  {
  fixedReader->Update();
  movingReader->Update();
  }
catch (itk::ExceptionObject & e)
  {
  std::cerr << "Unable to read input images!" << std::endl;
  std::cerr << e << std::endl;

  return EXIT_FAILURE;
  }
```

Now init the registration method. Declare the type of the registration method, operator and optimizer and instantiate these components.

```
typedef itk::PhaseCorrelationImageRegistrationMethod<ImageType,ImageType>
                                                          RegistrationType;
typedef itk::PhaseCorrelationOperator<RegistrationType>      OperatorType;
typedef itk::MaxPhaseCorrelationOptimizer<RegistrationType>  OptimizerType;

RegistrationType::Pointer pcmRegistration  = RegistrationType::New();
OperatorType::Pointer     pcmOperator      = OperatorType::New();
OptimizerType::Pointer    pcmOptimizer     = OptimizerType::New();
```

Now connect the optimizer and operator to the registration method and assign the inputs (fixed image and moving image).

```
pcmRegistration->SetOperator(    pcmOperator               );
pcmRegistration->SetOptimizer(   pcmOptimizer              );
pcmRegistration->SetFixedImage(  fixedReader->GetOutput()  );
pcmRegistration->SetMovingImage( movingReader->GetOutput() );
```

When everything is prepared, execute the registration.

```
try
  {
  pcmRegistration->Update();
  }
catch ( itk::ExceptionObject & e )
  {
  std::cout << "Some error during registration:" << std::endl;
  std::cout << e << std::endl;
```

```
    return EXIT_FAILURE;
    }
```

The registration result can be retrieved as translation parameters or as a translation transform from registration method's output.

```
  RegistrationType::ParametersType parameters
      = pcmRegistration->GetTransformParameters();
  RegistrationType::TransformType::ConstPointer transform
      = pcmRegistration->GetOutput()->Get();

  std::cout << "Translation found: " << parameters << std::endl;
```

Finally, resample the moving image by found transformation and write it. First, declare the types for resampler, caster and writer and instantiate these classes.

```
  typedef itk::ResampleImageFilter<ImageType,ImageType>  ResamplerType;
  typedef itk::Image<unsigned char,2>                    CharImageType;
  typedef itk::CastImageFilter<ImageType,CharImageType>  CasterType;
  typedef itk::ImageFileWriter<CharImageType>            WriterType;

  ResamplerType::Pointer resampler  = ResamplerType::New();
  CasterType::Pointer    caster     = CasterType::New();
  WriterType::Pointer    writer     = WriterType::New();
```

Connect the components and set the resultant transform to the resampler.

```
  resampler->SetOutputParametersFromImage( fixedReader->GetOutput() );
  resampler->SetTransform( transform );
  resampler->SetInput( movingReader->GetOutput() );
  caster->SetInput( resampler->GetOutput() );
  writer->SetInput( caster->GetOutput() );
  writer->SetFileName( argv[3] );
```

Execute the pipeline to resample, write the output image and then exit.

```
  try
    {
    writer->Update();
    }
  catch(itk::ExceptionObject & e)
    {
    std::cerr << "Unable to generate or write output image!" << std::endl;
    std::cerr << e << std::endl;

    return EXIT_FAILURE;
    }

  return EXIT_SUCCESS;
}
```

5 Literature survey

This section aims to give a short overview of principal papers focusing phase correlation. The relation of PCM to other image registration methods is described in [23]. This overview should be considered as introductory, just to give some references to the methods and techniques related to PCM.

The method was originally introduced by Kuglin and Hines [16] in 1975. Their approach designated to register only translated images is a basis for implementation in this contribution. Then, the method was used to register translated and rotated images [4] and finally to register translated, rotated and scaled images [20]. The last approach transforms rotation and scaling to translation by log-polar mapping of shift-invariant magnitudes of image spectrums. Such operation can be easily implemented by combination of ITK filters with the classes from this contribution, but the technique is limited to register 2-D images only (as it is not possible to transform 3-D rotations to translations). The idea was further developed in [24]. Authors of papers [11, 14] deal with disadvantages of polar mapping by employing pseudo-polar Fourier transform[2]. Other techniques were proposed in [18, 17].

There is a group of papers that aim to enhance PCM to register images with subpixel precision. The basic approach of interpolating the images or correlation surface (e.g. [1]) is outperformed by estimating the polyphase decomposition of cross power spectrum in [21, 6]. In [22], Stone et al. estimate the subpixel shift by first eliminating the effect of aliasing and then using least-squares fit to 2-D phase difference data. As this is a difficult task, authors of [7, 8, 9] separate the shift estimation in every dimension by SVD or high-order SVD of cross-power spectrum. An improvement of robustness of this approach is given in [12].

Foroosh and Hoge give in [5] a detailed study of some subpixel PCM techniques both in spatial and frequency domain. They also discuss the performance of PCM in case of images acquired by different sensor modalities.

Several methods for registration of 3-D images using frequency-domain techniques were proposed in [3, 19] and [13, 15]. The second group of papers employs 3-D pseudopolar Fourier transform to estimate the rotation from shift-invariant frequency magnitudes.

6 Further improvements

This contribution provides implementation of basic PCM algorithm. It can be further improved or extended for example by:

- filtering of input data to overcome the boundary effect,
- filtering of spectral data to overcome the effect of aliasing,
- implementing various subpixel techniques both in spatial and frequency domain.

Most of the techniques that exist in the literature can be implemented by creating a new operator or optimizer.

7 Software Requirements

You need to have the following software installed:

- Insight Toolkit 3.0
- CMake 2.4

PCM implementation uses a new customization of `New()` method of `itk::FFTRealToComplexConjugateImageFilter` and `itk::FFTComplexConjugateToRealImageFilter` introduced in ITK 3.0. This customization automatically selects the available FFT implementation between Vnl (default) or FFTW (if available for current data type). This behavior can be overriden by using a factory mechanism and so force PCM implementation to use desired FFT implementation.

References

[1] I.E. Abdou. Practical approach to the registration of multiple frames of video images. In K. Aizawa, R. L. Stevenson, and Y.-Q. Zhang, editors, *Proc. SPIE Vol. 3653, p. 371-382, Visual Communications and Image Processing '99, Kiyoharu Aizawa; Robert L. Stevenson; Ya-Qin Zhang; Eds.*, pages 371–382, December 1998. 5

[2] A. Averbuch, R.R. Coifman, D.L. Donoho, M. Elad, and M. Israeli. Fast and accurate polar fourier transform. *Applied and Computational Harmonic Analysis*, 21(2):145–167, September 2006. 5

[3] G.M. Cortelazzo, G. Doretto, and L. Lucchese. Free-form textured surfaces registration by a frequency domain technique. In *International Conference on Image Processing*, pages I: 813–817, 1998. 5

[4] E. de Castro and C. Morandi. Registration of translated and rotated images using finite fourier transform. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(5):700–703, September 1987. 5

[5] H. Foroosh and W.S. Hoge. *Motion Information in the Phase Domain*, chapter 3. Kluwer, May 2003. 5

[6] H. Foroosh, J.B. Zerubia, and M. Berthod. Extension of phase correlation to subpixel registration. *IEEE Trans. Image Processing*, 11(3):188–200, March 2002. 5

[7] W.S. Hoge. A subspace identification extension to the phase correlation method. *IEEE Trans. Medical Imaging*, 22(2):277–280, February 2003. 2, 5

[8] W.S. Hoge, D. Mitsouras, F.J. Rybicki, R.V. Mulkern, and C. Westin. Registration of multi-dimensional image data via sub-pixel resolution phase correlation. In *International Conference on Image Processing*, pages II: 707–710, 2003. 5

[9] W.S. Hoge and C.F. Westin. Identification of translational displacements between n-dimensional data sets using the high-order svd and phase correlation. *IEEE Trans. Image Processing*, 14(7):884–889, July 2005. 5

[10] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, http://www.itk.org/ItkSoftwareGuide.pdf, second edition, 2005.

[11] Y. Keller, A. Averbuch, and M. Israeli. Pseudopolar-based estimation of large translations, rotations, and scalings in images. *IEEE Trans. Image Processing*, 14(1):12–22, January 2005. 5

[12] Y. Keller, A. Averbuch, and O. Miller. Robust phase correlation. In *International Conference on Pattern Recognition*, pages II: 740–743, 2004. 5

[13] Y. Keller, A. Averbuch, and Y. Shkolnisky. Algebraically accurate volume registration using euler's theorem and the 3-d pseudo-polar fft. In *IEEE Computer Vision and Pattern Recognition*, pages II: 795–800, 2005. 5

[14] Y. Keller, Y. Shkolnisky, and A. Averbuch. The angular difference function and its application to image registration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(6):969–976, June 2005. 5

[15] Y. Keller, Y. Shkolnisky, and A. Averbuch. Volume registration using the 3-d pseudopolar fourier transform. *IEEE Trans. Image Processing*, 54(11):4323– 4331, November 2006. 5

[16] C.D. Kuglin and D.C. Hines. The phase correlation image alignment method. In *Proc. Int. Conf. on Cybernetics and Society*, pages 163–165. IEEE, September 1975. 1, 2, 5

[17] L. Lucchese. A hybrid frequency-space domain algorithm for estimating projective transformations of color images. In *International Conference on Image Processing*, pages II: 913–916, 2001. 5

[18] L. Lucchese and G.M. Cortelazzo. A noise-robust frequency domain technique for estimating planar roto-translations. *IEEE Trans. Signal Processing*, 48(6):1769–1786, June 2000. 5

[19] L. Lucchese, G. Doretto, and G.M. Cortelazzo. A frequency domain technique for range data registration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(11):1468–1484, November 2002. 5

[20] B.S. Reddy and B.N. Chatterji. An fft-based technique for translation, rotation, and scale-invariant image registration. *IEEE Trans. Image Processing*, 5(8):1266–1271, 1996. 5

[21] H. Shekarforoush, M. Berthod, and J. Zerubia. Subpixel image registration by estimating the polyphase decomposition of the cross power spectrum. In *IEEE Computer Vision and Pattern Recognition*, pages 532–537, 1996. 5

[22] H.S. Stone, M.T. Orchard, E.C. Chang, and S.A. Martucci. A fast direct fourier-based algorithm for subpixel registration of images. *IEEE Trans. Geoscience and Remote Sensing*, 39(10):2235–2243, October 2001. 5

[23] B. Zitová and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, October 2003. 5

[24] S. Zokai and G. Wolberg. Image registration using log-polar mappings for recovery of large-scale similarity and projective transformations. *IEEE Trans. Image Processing*, 14(10):1422–1434, October 2005. 5