TestPhaseCorrelation package is a collection of C/C++ routines which validate Phase-Only Matched Filter(POMF), and prove the result usage, in other words, we should apply the result to which input data(vector/image/volume) to matched another input data.

1 Theoretical Background

Due to its ability to deal with the partially overlapped problem between the signals, the cross-correlation technique is widely used to determine translation parameters between signals. However, the value of the standard cross correlation method is heavily dependent on the energy of underlying signals, so it often fails to discriminate the signals which are of different shapes but similar energy. Furthermore, the correlation peaks could be relatively broad depending on the signal structures, which makes it difficult and unreliable to locate the correct displacement between noisy signals. In order to decrease the impact of such detrimental points and achieve more distinct sharp peaks, the POMF algorithm [1] is used to resolve the signal correlation problem in this paper.

The POMF technique has extraordinary significance in transform-based registration methods as regards partially overlapped scans.

The POMF decouples the local signal energy from the signal structures based on the fact that two shifted signals carry the shift information within the phase of their Fourier spectrum. Let $f_1(x, y, z)$ and $f_2(x, y, z)$ be two shifted signals, and $\mathcal{F}_1(u, v, k)$ and $\mathcal{F}_2(u, v, k)$ be their corresponding Fourier spectra. The shift between these two translated signals could be solved by the following equations:

$$S(u,v,k) = \frac{\mathcal{F}_1(u,v,k)}{|\mathcal{F}_1(u,v,k)|} \bullet \frac{\mathcal{F}_2(u,v,k)^*}{|\mathcal{F}_2(u,v,k)|} \quad (1)$$

$$s(x,y,z) = \mathcal{F}^{-1}\{S(u,v,k)\} \quad (2)$$

$$(x_p, y_p, z_p) = \arg\max_{(x,y,z)} s(x,y,z) \quad (3)$$

where $*$ indicates the complex conjugate, and (x_p, y_p, z_p) is the displacement between the two signals. In theory, it could be used in arbitrary dimensional signal registration problems. Ideally, the $s(x, y, z)$ contains a Dirac peak, but the Dirac pulse deteriorates in practical due to the noise.

2 Package Structure

✗ CMakeLists.txt
used by **CMake**, the cross-platform, open-source build system. Makes this package be used under other OS.

✗ FindFFTW3.cmake
.cmake file used by **CMake** to tell compiler the location of **FFTW3** library. Copy it to the 'Modules' folder of CMake, then it is unnecessary to set the FFTW3 library location by hand.

✗ PhaseCorrelation.h
Declaration of three function:
Note: we adopt RIGHT-hand Cartesian Coordinate System

- ***PhaseCorrelation1D***

```
void PhaseCorrelation1D(const Eigen::VectorXf signal,
                        const Eigen::VectorXf pattern,
                        const int size,
                        int &offset);
```

/* Parameters:
* [in] signal the input(signal) vector
* [in] pattern the input(pattern) vector
* [in] size the size of input vectors
* [out] offset the result offset, move pattern right (positive x axis) to match signal *
*/

- ***PhaseCorrelation2D***

```
void PhaseCorrelation2D(const EigenMatrixRowXf signal,
                        const EigenMatrixRowXf pattern,
                        const int height,
                        const int width,
                        int &height_offset,
                        int &width_offset);
```

/* Parameters:
* [in] signal the input(signal) image
* [in] pattern the input(pattern) image
* [in] height the height of input images(how many rows/size of column/range of x)
* [in] width the width of input images (how many columns/size of row/range of y)
* [out] `height_offset` the result offset, move down (positive x axis) pattern `height_offset` to match signal
* [out] `width_offset` the result offset, move right (positive y axis) pattern `width_offset` to match signal
*/

- ***PhaseCorrelation3D***

```
void PhaseCorrelation3D(const float *signal,
                        const float *pattern,
                        const int height,
                        const int width,
                        const int depth,
                        int &height_offset,
                        int &width_offset,
                        int &depth_offset);
```

* Parameters:
* [in] signal the input(signal) volume

* [in] pattern the input(pattern) volume
* [in] height the height of input volumes(range of x)
* [in] width the width of input volumes (range of y)
* [in] depth the depth of input volumes (range of z)
* [out] `height_offset` the result offset, move down (positive x axis) pattern `height_offset` to match signal
* [out] `width_offset` the result offset, move right (positive y axis) pattern `width_offset` to match signal
* [out] `depth_offset` the result offset, move close to viewer (positive z axis) pattern `depth_offset` to match signal
*/

✗ PhaseCorrelation.hpp
Implementation of the three functions declared in ***PhaseCorrelation.h***

✗ TestPhaseCorrelation.cpp
An example for the usage of functions PhaseCorrelation1D, PhaseCorrelation2D and PhaseCorrelation3D

3 Package Dependency

The code was developed and tested in the GNU/Linux environment. But we give the *CMakeLists.txt* which used by CMake, the cross-platform, open-source build system. We believe the code could be used under other OS. Some modifications might be required, but I do not think anything drastic should be necessary.

This package depends on **FFTW3**, which is free available on `http://www.fftw.org`. You could tell the complier the location of **FFTW3** by copying the *FindFFTW3.cmake* to the **Modules** fold of cmake, if you want. Or you could tell the complier the location of **FFTW3** by hand.

4 Application Example

TestPhaseCorrelation.cpp contains five examples using PhaseCorrelation1D, PhaseCorrelation2D and PhaseCorrelation3D.

It does not need input, since we generate the data randomly.

5 Discussion

Actually, it is possible to use the *pointer* instead of Eigen::Vector or Eigen::Matrix to pass input data to PhaseCorrelation1D, PhaseCorrelation2D, as what PhaseCorrelation3D did.

Besides, it is also possible to use the 1D FFTW in PhaseCorrelation2D and PhaseCorrelation3D instead of 2D or 3D FFTW.

References

[1] Joseph L. Horner and Peter D. Gianino. Phase-only matched filtering. *Applied Optics*, 23(6):812–816, 1984.