FFT shift

Gaëtan Lehmann

November 6, 2006

INRA, UMR 1198; ENVA; CNRS, FRE 2857, Biologie du Développement et Reproduction, Jouy en Josas, F-78350, France

Abstract

A common usage when working with Fourier transform is to shift the the image to put the zero-frequency component in the center of the image. This contribution comes with a filter to perform this transform.

1 Description

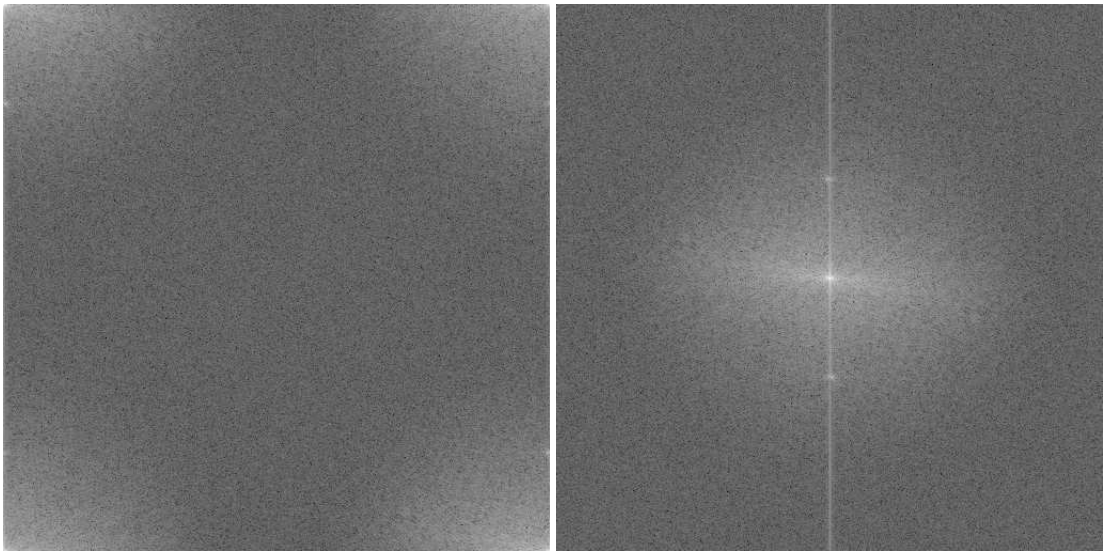
This filter is multitreaded and works with any dimensions or size of image, and with any pixel type. It is very similar to the fftshift and ifftshift function of matlab.

When the size of the image is odd on one dimension or more, performing the transform twice will not produce the same image than the input, as shown in the Figure 2. To get it right, the option `SetInverse(true)` has to be used.

2 Code sample

The filter is very simple, and there shouldn't be any problem to use it. Please look at check.cxx in the tar ball for an example.

```
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkCommand.h"
#include "itkSimpleFilterWatcher.h"
#include "itkRGBPixel.h"
#include "itkFFTShiftImageFilter.h"


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

  if( argc != 4 )
```

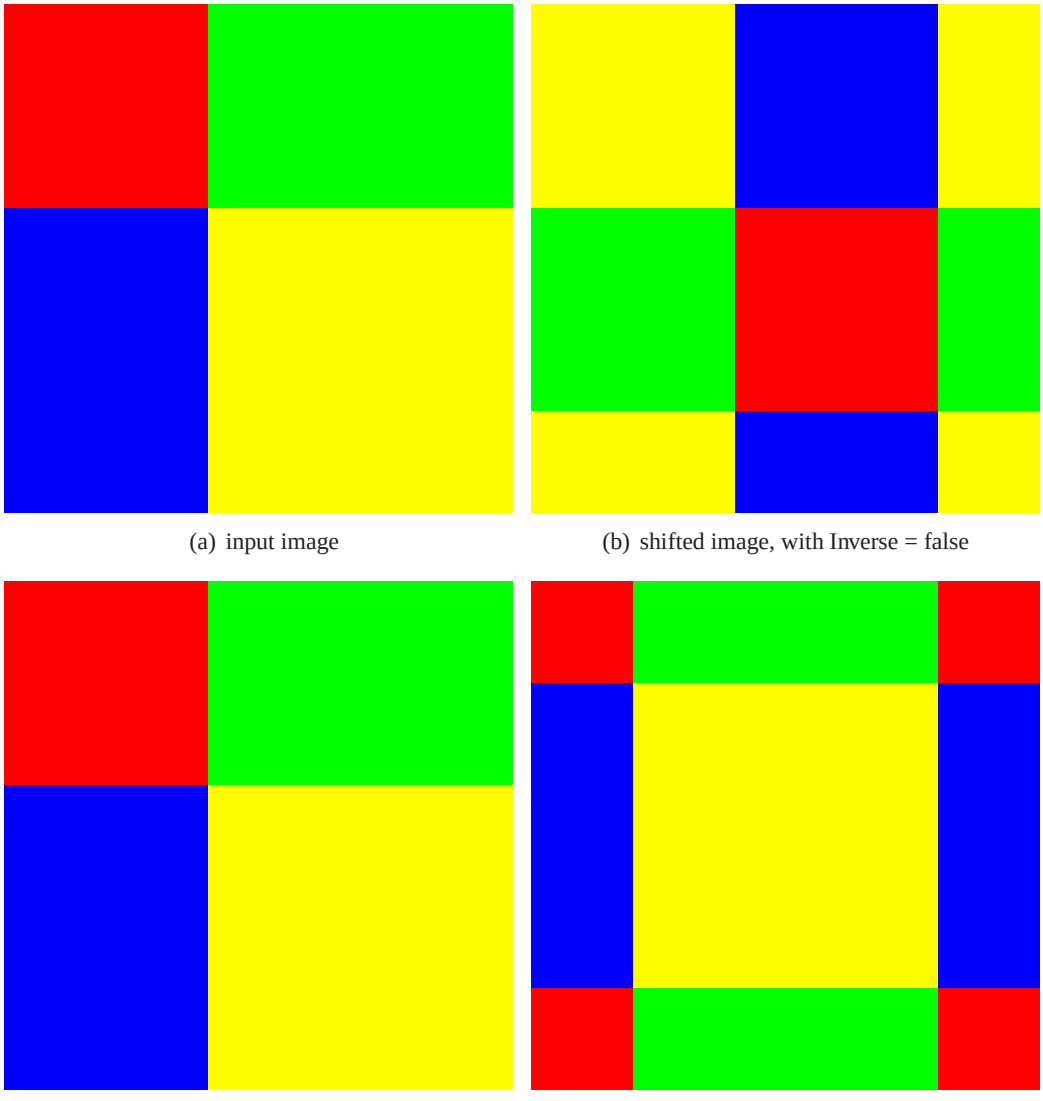


(a) input image

(b) shifted image. The zero-frequency components are now centered.

Figure 1: Result of the transform on the log of the modulus of the fft transform of a real image.

```
  {
  std::cerr << "usage: " << argv[0] << " inputImage outputImage inverse" << std::endl;
  std::cerr << "  inputImage: The input image." << std::endl;
  std::cerr << "  outputImage: The output image." << std::endl;
  std::cerr << "  inverse: 0, to perform a forward transform, or 1 to perform" << std::endl;
  std::cerr << "           an inverse transform." << std::endl;
  exit(1);
  }

const int dim = 3;

typedef itk::RGBPixel< unsigned char > PType;
typedef itk::Image< PType, dim > IType;

typedef itk::ImageFileReader< IType > ReaderType;
ReaderType::Pointer reader = ReaderType::New();
reader->SetFileName( argv[1] );

typedef itk::FFTShiftImageFilter< IType, IType > FilterType;
FilterType::Pointer filter = FilterType::New();
filter->SetInput( reader->GetOutput() );
filter->SetInverse( atoi( argv[3] ) );

itk::SimpleFilterWatcher watcher(filter, "filter");

typedef itk::ImageFileWriter< IType > WriterType;
WriterType::Pointer writer = WriterType::New();
writer->SetInput( filter->GetOutput() );
writer->SetFileName( argv[2] );
writer->Update();
```



(a) input image

(b) shifted image, with Inverse = false

(c) the image shifted twice, with Inverse = false for the first run, and Inverse = true for the second

(d) the imageshifted twice, with Inverse = false for both the first and the second run. The input image is not restored.

Figure 2: Result of the transforms on an image with an odd size (5x5).

```
  return 0;
}
```

References

[1] L. Ibanez and W. Schroeder. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-10-6, http://www.itk.org/ItkSoftwareGuide.pdf, 2003.