Go-Go Gabor Gadgetry

Nicholas J. Tustison and James C. Gee

April 20, 2007

Penn Image Computing and Science Laboratory
University of Pennsylvania

Abstract

Although Gabor filtering is quite prevalent in the computer vision community for such tasks as texture segmentation and motion analysis, such capabilities are conspicuously absent from the Insight Toolkit. The contribution described in this paper attempts to remedy this deficiency by introducing two new classes: `itkGaborKernelFunction` and `itkGaborImageSource`.

Keywords: *filtering, Gabor, image source, kernel*

1 Introduction

We propose two Gabor filter classes for inclusion within the ITK library. The `itkGaborKernelFunction` class calculates the standard 1-D complex Gabor kernel given by

$$h(x) = e^{-x^2/2\sigma^2} e^{j(2\pi f x + \phi)} \tag{1}$$

$$= e^{-x^2/2\sigma^2} \cos(2\pi f x + \phi) + j e^{-x^2/2\sigma^2} \sin(2\pi f x + \phi) \tag{2}$$

which is essentially a complex sinusoid enveloped by a Gaussian function. This kernel simply adds to the existing library of kernel functions (e.g. B-spline, Gaussian).

The second class we introduce takes advantage of the previously mentioned kernel function to formulate the`itkGaborImageSource` class. This class produces an N-D Gabor image where the sense of directionality intrinsic to the Gabor filter is aligned with the x-axis (alternate alignments can be achieved through the various Transform classes). The complex sinusoid in the x direction is enveloped within an N-D Gaussian. This is described mathematically as

$$h(x_1, x_2, \ldots, x_N) = e^{j(2\pi f x_1 + \phi)} e^{-\sum_{i=1}^{N} x_i^2/2\sigma_i^2} \tag{3}$$

2 Class Overview

2.1 Gabor Kernel

The kernel function simply calculates Equation (1). The parameters that are required are as follows:

- σ, the standard deviation of the Gaussian. The variable name is `m_Sigma`.
- f, the modulation frequency of the complex sinusoid. The variable name is `m_Frequency`.
- ϕ, the phase offset. The variable name is `m_PhaseOffset`.

Note that access to these variables is provided via the conventional ITK `Get/Set` mechanisms. In addition, the user must select which part (i.e. real or imaginary) is actually calculated. This is selected by setting the boolean variable `m_CalculateImaginaryPart`.

2.2 Gabor Image Source

There was much borrowing from the itkGaussianImageSource class while creating the itkGaborImageSource class. This seems natural given the similarities between an N-D Gaussian image and the way we have defined an N-D Gabor image in Equation (3). The common variables governing the Gaussian include:

- `m_Sigma` — an N-D array of standard deviation values, and
- `m_Mean` — an N-D array of mean values.

The standard image specifications are also provided.

- `m_Size` — size of output image.
- `m_Spacing` — spacing of output image.
- `m_Origin` — origin of output image.

Finally, all the variables used to specify the Gabor kernel outlined in the previous section are required for specifying the Gabor image.

3 Usage

In the test code included with this submission (`itkGaborImageSourceTest`), we calculate a 2-D and a 3-D Gabor image. Instantiating and executing the 2-D example is given by the following code snippet.

```
typedef itk::GaborImageSource<ImageType> GaborSourceType;
GaborSourceType::Pointer gaborImage = GaborSourceType::New();

GaborSourceType::ArrayType sigma;
sigma[0] = 2.0;
sigma[1] = 5.0;

ImageType::SpacingType spacing;
spacing.Fill( 0.25 );
ImageType::SizeType size;
size.Fill( 64*4 );

gaborImage->SetSpacing( spacing );
gaborImage->SetSize( size );

gaborImage->SetSigma( sigma );
gaborImage->SetFrequency( 0.1 );
gaborImage->SetCalculateImaginaryPart( false );

try
  {
  gaborImage->Update();
  }
catch (itk::ExceptionObject & err)
  {
  std::cout << "ExceptionObject caught !" << std::endl;
  std::cout << err << std::endl;
  return EXIT_FAILURE;
  }
```

This result is given in Figure 1(a). Both the 3-D code snippet and corresponding results (Figure 1(b)) are given below.

```
// Instantiate the filter
typedef itk::GaborImageSource<ImageType> GaborSourceType;
GaborSourceType::Pointer gaborImage = GaborSourceType::New();

GaborSourceType::ArrayType sigma;
sigma[0] = 2.0;
sigma[1] = 10.0;
sigma[2] = 10.0;

ImageType::SpacingType spacing;
spacing.Fill( 0.5 );
ImageType::SizeType size;
size.Fill( 64*2 );

gaborImage->SetSpacing( spacing );
gaborImage->SetSize( size );

gaborImage->SetSigma( sigma );
gaborImage->SetFrequency( 0.1 );
gaborImage->SetCalculateImaginaryPart( true );

try
  {
  gaborImage->Update();
  }
catch (itk::ExceptionObject & err)
  {
  std::cout << "ExceptionObject caught !" << std::endl;
  std::cout << err << std::endl;
  return EXIT_FAILURE;
  }
```

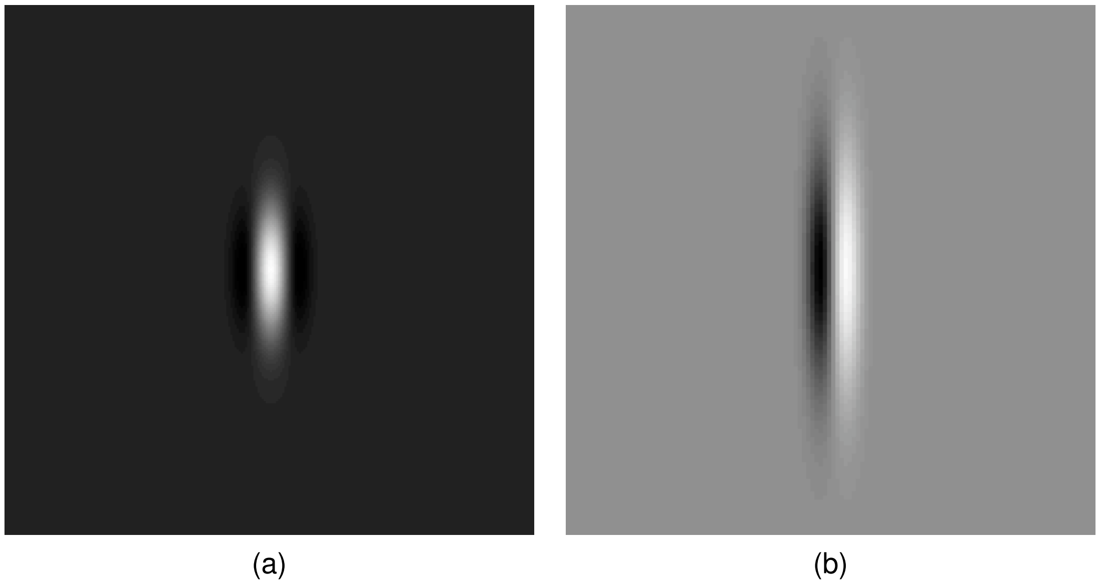

Figure 1: (a) 2-D Gabor image and (b) a 2-D slice from the 3-D Gabor image.