Gaussian Interpolation

Paul A. Yushkevich, Nicholas J. Tustison, and James C. Gee

November 25, 2009

Penn Image Computing and Science Laboratory
University of Pennsylvania

Abstract

In this submission, we offer the `itk::GaussianInterpolationImageFunction` which adds to the growing collection of existing interpolation algorithms in ITK for resampling scalar images such as the `itk::LinearInterpolateImageFunction`, `itk::BSplineInterpolateImageFunction`, and `itk::WindowedSincInterpolateImageFunction`. We provide a brief discussion of the theory behind the submission and the algorithmic implementation.

1 Introduction

In many ways, the Gaussian is an optimal filter for resampling images. Although the sinc interpolation is theoretically optimal, most images violate the associated periodicity assumption, which leads to ringing. Further discussion of such issues can be found in various image/signal processing textbooks and online reviews. In this submission, we provide the `itk::GaussianInterpolationImageFunction` which is derived from the `itk::InterpolationImageFunction` and functions similarly to the `itk::LinearInterpolateImageFunction`, `itk::BSplineInterpolateImageFunction`, and `itk::WindowedSincInterpolateImageFunction`.

Viewing a given scalar image as a piecewise constant intensity function, one can calculate the interpolated intensity value at a given sample point using Gaussian interpolation. This is best illustrated with the 1-D example given in Figure 1. Suppose we wish to calculate the interpolated value at point s where the image intensities are represented by the piecewise function $F(x)$, the values of which are weighted by the Gaussian function centered at s with a user-specified σ value and cutoff distance. We can calculate this with the following integral:

$$\begin{aligned} I(s) &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{t_1}^{t_6} F(x) \exp\left(-\frac{(x-s)^2}{2\sigma^2}\right) dx \\ &= \sum_{i=1}^{5} \frac{1}{\sqrt{2\pi\sigma^2}} I_i^c \int_{t_i}^{t_{i+1}} \exp\left(-\frac{(x-s)^2}{2\sigma^2}\right) dx \\ &= \sum_{i=1}^{5} I_i^c \left(\operatorname{erf}(t_{i+1}) - \operatorname{erf}(t_i)\right) \end{aligned} \tag{1}$$

where I_i^c is the constant intensity value over the interval $[t_i, t_{i+1})$ and erf is the error function. This is easily extended to handle N-dimensional images.

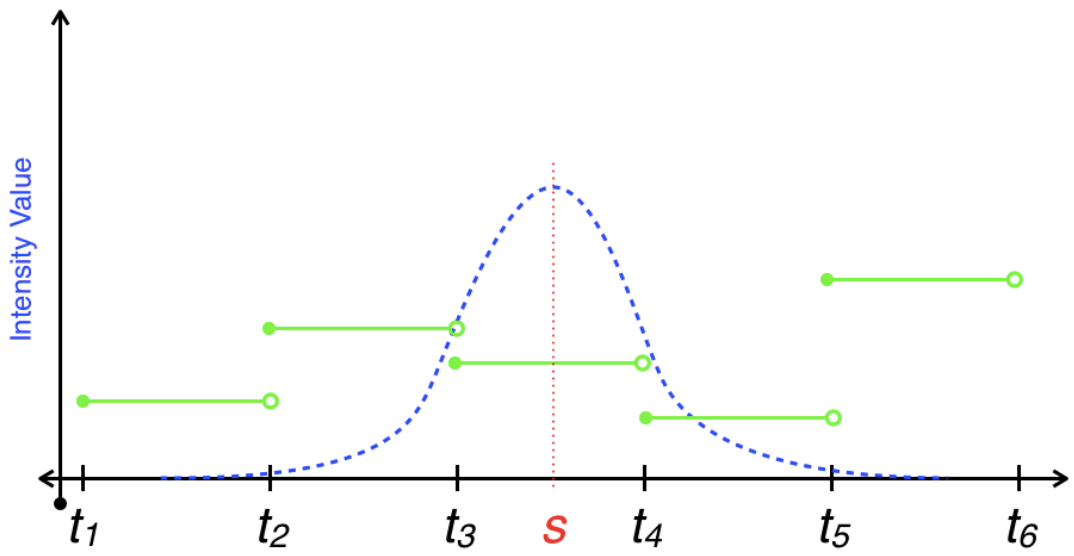


Figure 1: 1-D example of the proposed Gaussian interpolation approach. Image intensity values are represented as a piecewise constant intensity function (in green) with a normalized Gaussian centered at the sample point, s.

2 Implementation

Since the proposed class was derived from the `itk::InterpolateImageFunction`, it is meant to have the same basic interface as the other interpolate image functions. However, there are two parameters that the user can specify, specifically `Sigma` and `Alpha`. `Sigma` defines the shape of the Gaussian in each image dimension whereas `Alpha` determines the cutoff as a multiplicative factor of `Sigma`. These parameters can be set using

```
>gaussianInterpolator->SetParameters( sigma, alpha );
```

where `sigma` is an array of `RealType` and `alpha` is of `RealType`. These parameters can also be set individually through the usual macro methods of

```
>gaussianInterpolator->SetSigma( sigma );
```

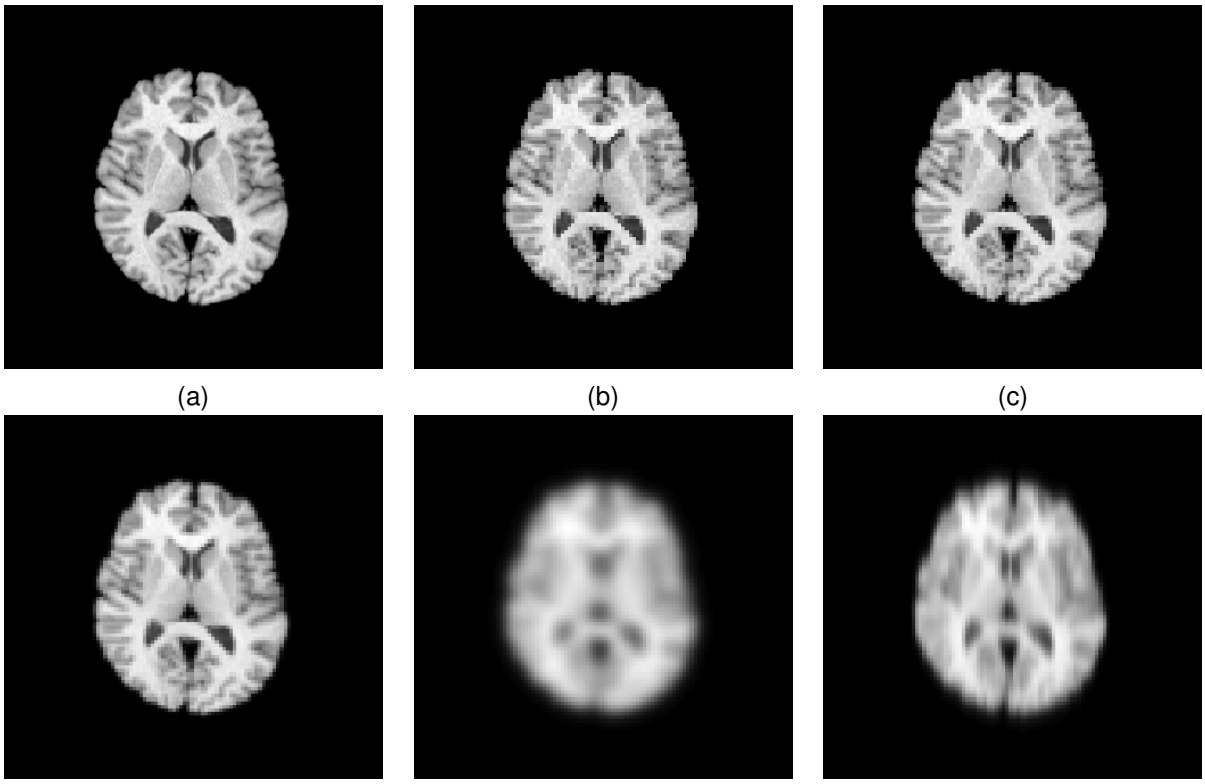
and

```
>gaussianInterpolator->SetAlpha( alpha );
```

We have included the file `itkGaussianInterpolateImageFunctionTest.cxx` where one can explore three sampling routines including the one proposed in this submission. Usage is as follows:

```
Usage: ./itkGaussianInterpolateImageFunctionTest imageDimension inputImage
  outputImage MxNxO [size=1,spacing=0] [interpolation type]
  Interpolation type:
    0. linear (default)
    1. nearest neighbor
    2. gaussian [sigma] [alpha]
```

Tests are provided in `CMakeLists.txt` which were used to produce the examples in Figure 2. In most



(a) (b) (c)

(d) (e) (f)

Figure 2: (a) Original axial brain slice of size 256×256 to be resampled to size 100×100. (b) Nearest neighbor interpolation. (c) Linear interpolation. (d) Gaussian interpolation ($\sigma = 1.0 \times 1.0, \alpha = 3.0$). (e) Gaussian interpolation ($\sigma = 5.0 \times 5.0, \alpha = 3.0$). (f) Gaussian interpolation ($\sigma = 1.0 \times 5.0, \alpha = 3.0$).

applications a very small sigma (0.8 or so) would probably be sufficient. Otherwise, applying the Gaussian filter to the input image and then sampling it with linear interpolation is probably just as good. And also the large sigma makes the filter pretty slow.