Shape-based Interpolation of a Set of 2D Slices

Release 1.00

Christine Boydev[1,2], David Pasquier[3,4], Foued Derraz[1,5], Laurent Peyrodie[6], Abdelmalik Taleb-Ahmed[1] and Jean-Philippe Thiran[2,7]

October 8, 2012

[1]Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH), Université de Valenciennes et du Hainaut-Cambrésis, France
[2]Signal Processing Laboratory (LTS5), École Polytechnique Fédérale de Lausanne, Switzerland
[3]Centre de Radiothérapie et d'Oncologie Galilée de Lille, France
[4]Département Universitaire de Radiothérapie, Centre Oscar Lambret, Lille, France
[5]Unité de Traitements de Signaux Biomédicaux (UTSB), Faculté Libre de Médecine, Lille, France
[6]Unité de Traitements de Signaux Biomédicaux (UTSB), Hautes Études d'Ingénieur, Lille, France
[7]Department of Radiology, University Hospital Center (CHUV) and University of Lausanne (UNIL), Switzerland
{christine.boydev, jp.thiran}@epfl.ch, davidpasquier@free.fr, {foued.derraz,taleb}@univ-valenciennes.fr, Laurent.PEYRODIE@hei.fr

Abstract

We implemented the shape-based interpolation method described by Raya and Udupa in 1990 for three-dimensional images, and created two standalone filters using the Insight Toolkit ITK www.itk.org. The image to be interpolated must be a 3D binary image which represents an object as a series of 2D slices with pixel values at 1 inside the object and 0 outside by convention. The first filter takes as input a 3D binary image wherein the object of interest is represented in full, that is, on all contiguous slices that comprise it. Such filter yields an upsampled 3D binary image. The second filter takes as input a 3D binary image wherein the object of interest is represented only on certain slices that are regularly spaced. It yields a 3D binary image of the same size as that of the input and wherein the object is represented on all contiguous slices that comprise it. This paper is provided with the source code as well as the data used for validation.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3389]

Contents

1 Introduction

In this work, we examine two issues that may arise after the segmentation of a three-dimensional image:

- Let us first consider that all the slices of the original image are subject to (manual or automatic) segmentation. This results in a set of 2D contiguously delineated slices with the same resolution as that of the original image. Nevertheless, one may want to obtain a segmented image with a higher resolution in the direction orthogonal to the slices, that is, with a thinner slice spacing. Indeed our code allows to interpolate between the delineated slices and obtain new intermediate delineated slices, which originally were nonexistent.
- Suppose instead that to save time the expert that is in charge of the manual segmentation of the 3D image decides to only delineate a few slices. Our code allows to estimate the contours that have not been drawn.

Using filters from the Insight Toolkit, we developed two standalone filters based on Raya and Udupa's shape-based interpolation method for 3D images [5], where the authors report the superior performance of their method with respect to the classical gray-value interpolation followed by thresholding.

2 Implementation

The piece of data passed as input is a 3D binary image (e.g. a meta-image with an mhd extension).

In our code, we consider separately the two above-described cases. In the first case (cf. §2.1), the input binary image is a set of contiguously delineated slices. The object of interest has been segmented on all possible slices. We aim at interpolating between all slices in the original dataset and coming up with an ensemble of new thinner estimated contours. In the second case (cf. §2.2), the input binary image is a set of sparsely and equally spaced delineated slices (e.g. one slice over two or three delineated). We now aim at ending up with an image wherein the missing contours are estimated.

Let us define n as the ratio of the number of the *delineated* slices in the output image to the number of the *delineated* slices in the input image. The user can specify the value of n using the `SetDelineationRatio()` function.

In both cases, we created a filter which derives from `itk::ImageToImageFilter`. The output image is a 3D binary image, either with a number of slices *n* times higher (cf. §2.1) or of the same size as that of the input (cf. §2.2). The input and output image types being always `itk::Image< unsigned char, 3 >`, we considered there was no need for the filters to be templated.

2.1 Interpolation of a set of contiguously delineated slices

There are three steps to be considered. First, a distance transform is applied to the data on a slice-by-slice basis using the `itk::SliceBySliceImageFilter` associated with the internal filter `itk::DanielssonDistanceMapImageFilter`. The `itk::DanielssonDistanceMapImageFilter` is based on the algorithm developed by PE Danielsson [1]. This results in the creation of a gray-value image wherein the value of each pixel represents the Euclidean distance from that point to the cross-sectional boundary of the object (i.e. within the slice). By convention, points are assigned negative values inside the object, and positive values outside. Secondly, the distance-representing gray-value slices are interpolated using linear or higher-order interpolation to obtain new estimated less spaced gray-value slices (there are *n* times more slices than in the input). Raya and Udupa report that cubic spline interpolation gives superior results than linear interpolation. In our implementation, we let the user set the kind of interpolation he wants (linear or spline) by means of the `SetInterpolator()` function. Finally, the set of slices is thresholded at zero to produce the interpolated binary dataset.

We call this filter `IntraBinaryShapeBasedInterpolationImageFilter`. As the spacing and the largest possible region of the output are different from that of the input (a change of the meta-information occurs during filtering), it was necessary to override the `ProcessObject::GenerateOutputInformation()` and `ProcessObject::GenerateInputRequestedRegion()`. We refer to [4] for more details on pipeline execution.

2.2 Interpolation of a set of sparsely and equally spaced delineated slices

Here the object of interest is not delineated on every slice. One slice over *n* is delineated. *The user should be warned that our implementation requires the delineated slices to be regularly spaced.* We first produce an intermediate, subsampled image made of contiguously delineated slices. Indeed only the delineated slices are copied into that intermediate image whose slice spacing is *n* times bigger than that of the original image. The undelineated slices in the original image are left out. We now can go back to §2.1 for the methodology. Each slice of that intermediate image is passed to the `itk::DanielssonDistanceMapImageFilter` which calculates the Euclidean distance map with respect to the cross-sectional boundary. The distance map image is then upsampled back to the original sampling rate using linear or higher-order interpolation.

We call this filter `InterBinaryShapeBasedInterpolationImageFilter`. We note that the implementation of this filter differs from the previous one in section 2.1 notably because of the `ProcessObject::GenerateOutputInformation()` and `ProcessObject::GenerateInputRequestedRegion()` that are not overriden here.

3 Usage

The usage of these two filters is similar to other ITK filters. Their interface is very straightforward. The following snippet of code illustrates the usage of the filter we describe in §2.2:

```
typedef unsigned char InputPixelType;
const unsigned int Dimension = 3;

// Declare the input and output types
typedef itk::Image< InputPixelType,  Dimension >       InputImageType;
typedef itk::Image< InputPixelType,  Dimension >       OutputImageType;

// Declare the filter
typedef itk::InterBinaryShapeBasedInterpolationImageFilter  FilterType;
FilterType::Pointer filter = FilterType::New();

... Read an image and pass it to the filter
filter->SetInput( reader->GetOutput() );

// Setup the default parameters
filter->SetDelineationRatio( 3 );   // n is the ratio of the number of the delineated
                                    // slices in the output to that in the input
... Setup an interpolator
filter->SetInterpolator( interpolator );   // by default, linear interpolator


// Write output
... Setup a writer
writer->SetInput( filter->GetOutput );
writer->Update();
```

4 Software requirements

This code was developed on a Linux computer with distribution openSUSE 11.4 x86_64. You need to have the following software installed:

- Insight Toolkit 3.20.1
- CMake 2.8.3
- gcc 4.5.1

5 Example

In this section, we show two use cases of the shape-based interpolation algorithm as described in §2.2.

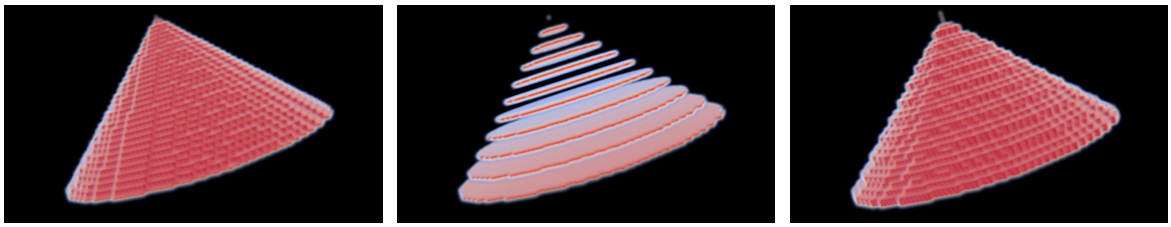
5.1 Interpolation of a cone

We manually drew a right circular cone with aperture 2π using ITK-SNAP (http://www.itksnap.org), a free cross-platform open-source software application for manual segmentations [7]. Given that the image is pixelated, the drawn object is not stricly a cone. The resolution of this image is 1mm $\times$ 1mm $\times$ 1mm. We then purposely replaced the values of the pixels that belong to every two and three slices by the background value in order to illustrate the case where only one slice over three has been segmented. After that, we

run the algorithm of section 2.2 using linear interpolation to estimate the missing contours. Figure 1 shows 3D views of each of these images. Figure 2 compares the same 2D view of an axis plane in the original cone, the cone before interpolation and the cone after interpolation. Dice's similarity coefficient between the interpolated volume, A, and the volume initially drawn (original volume), B, was also calculated as follows [2] :

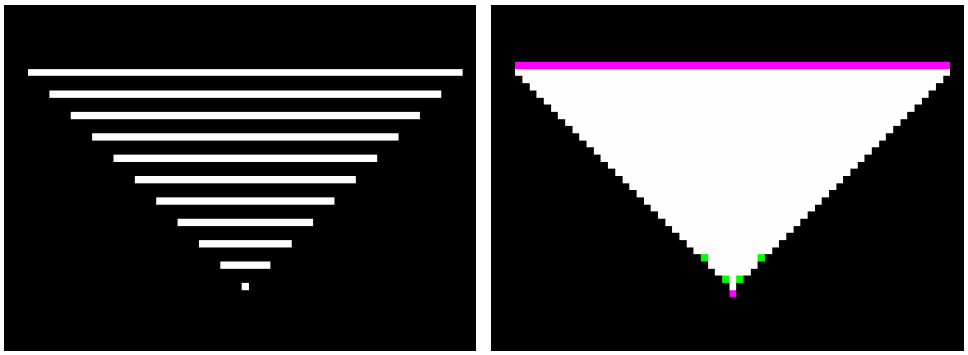
$$\textit{Dice's similarity coefficient} = \frac{2 \cdot (A \cap B)}{A + B} \tag{1}$$

It was found to be 0.94.



(a) *Original cone* ***(b)*** *Cone before interpolation* ***(c)*** *Cone after interpolation*

Fig. 1: *3D views of (a) the cone manually drawn, (b) the same cone but with missing contours, and (c) the interpolated cone. The display software used is ParaView (*`http://www.kitware.com/products/paraview.html`*) [3].*



(a) *Cone before interpolation in a plane passing through the axis*

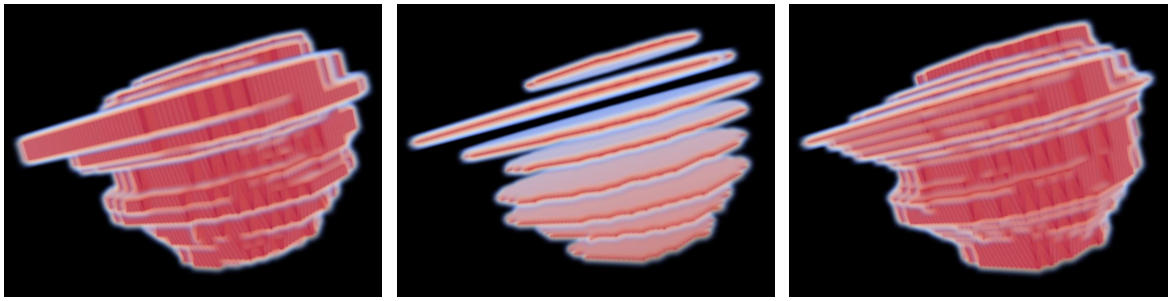
(b) *Overlay of the original cone and the interpolated cone in the same plane as in (a). Pixels in white are in perfect correspondance; pixels in green belong to the original cone only; pixels in purple belong to the interpolated cone only.*

Fig. 2: *(a) 2D central view of the cone before interpolation. (b) Overlay of 2D central views of the original and interpolated cones. The display software used is VV (*`http://vv.creatis.insa-lyon.fr/`*) [6]. The display interpolation is purposely desactivated here.*

5.2 Interpolation of a clinical image

Here is a more realistic example, a clinical case. An expert has manually delineated a prostate every slice in a cone-beam computed tomography (CBCT) image. The resolution of the original dataset is 1mm $\times$ 1mm $\times$ 1mm. As previously, we purposely replaced the values of the pixels that belong to every two and three slices by the background value in order to illustrate the case where only one slice over

three has been delineated. We run the algorithm on the latter binary delineated image using linear interpolation to estimate the missing contours. The result is shown in Figure 3. The interpolated contours have been visually validated by the expert after examining the overlay of the binary interpolated image onto the gray-value CBCT image. Dice's similarity coefficient between the interpolated volume and the volume initially delineated by the expert was also calculated and found to be 0.94.



(a) *Original prostate* ***(b)*** *Before interpolation* ***(c)*** *After interpolation*

Fig. 3: *Interpolation of a prostate binary image derived from the manual segmentation of a clinical gray-value CBCT image. 3D views of the prostate (a) before interpolation and (b) after interpolation. The display software used is ParaView (*`http://www.kitware.com/products/paraview.html`*) [3].*

6 Conclusion

We implemented two filters that perform a shape-based interpolation of 3D binary images. Two different situations are handled: the image to be interpolated has been delineated on all possible slices and the interpolation process results in an upsampled image with thinner delineated contours, or the image to be interpolated has not been delineated on every slice but the contours are regularly spaced and the missing contours are estimated.

7 Acknowledgments

This work is funded by Elekta SAS, Boulogne Billancourt, France.

References

[1] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227 – 248, 1980. 2.1

[2] Lee Raymond Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945. 5.1

[3] A. Henderson. *ParaView Guide, A Parallel Visualization Application*. Kitware Inc., 2007. 1, 3

[4] L. Ibanez, W. Schroeder, L. Ng, J. Cates, and The Insight Software Consortium. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, second edition, 2005. 2.1

[5] S. P. Raya and J. K. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Trans Med Imaging*, 9(1):32–42, 1990. 1

[6] S. Rit, R. Pinho, V. Delmon, M. Pech, G Bouilhol, J. Schaerer, B. Navalpakkam, J. Vandemeulebroucke, P. Seroul, and D. Sarrut. VV, a 4D slicer. In *Medical Imaging and Computer-Assisted Intervention MICCAI, Fourth International Workshop on Pulmonary Image Analysis*, pages 171–175, 2011. 2

[7] Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-guided 3d active contour segmentation of anatomical structures: significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, Jul 2006. 5.1