nD morphological contour interpolation

Release 1.0

Dženan Zukić[1], Jared Vicory[2], Matthew McCormick[3] Laura E.M. Wisse[4], Guido Gerig[5], Paul Yushkevich[6] and Stephen Aylward[7]

August 19, 2016

[1]Kitware Inc., dzenan.zukic@kitware.com
[2]Kitware Inc., jared.vicory@kitware.com
[3]Kitware Inc., matt.mccormick@kitware.com
[4]Penn Image Computing & Science Laboratory, Department of Radiology, University of Pennsylvania, laura.wisse@uphs.upenn.edu
[5]NYU Tandon School of Engineering, Department of Computer Science and Engineering, gerig@nyu.edu
[6]Penn Image Computing & Science Laboratory, Department of Radiology, University of Pennsylvania, pauly2@mail.med.upenn.edu
[7]Kitware Inc., stephen.aylward@kitware.com

Abstract

This document describes a new class, `itk::MorphologicalContourInterpolator`, which implements a method proposed by Albu et al. in 2008. Interpolation is done by first determining correspondence between shapes on adjacent segmented slices by detecting overlaps, then aligning the corresponding shapes, generating transition sequence of one-pixel dilations and taking the median as result. Recursion is employed if the original segmented slices are separated by more than one empty slice.

This class is n-dimensional, and supports inputs of 3 or more dimensions. 'Slices' are $n-1$-dimensional, and can be both automatically detected and manually set. The class is efficient in both memory used and execution time. It requires little memory in addition to allocation of input and output images. The implementation is multi-threaded, and processing one of the test inputs takes around 1-2 seconds on a quad-core processor.

The class is tested to operate on both `itk::Image` and `itk::RLEImage`. Since all the processing is done on extracted slices, usage of `itk::RLEImage` for input and/or output affects performance to a limited degree.

This class is implemented to ease manual segmentation in ITK-SNAP (`www.itksnap.org`). The class, along with test data and automated regression tests is packaged as an ITK remote module `https://github.com/KitwareMedical/ITKMorphologicalContourInterpolation`.

Latest version available at the Insight Journal [`http://hdl.handle.net/10380/3563`]
Distributed under Creative Commons Attribution License

Contents

1 Introduction

Proliferation of automatic segmentation algorithms is on the rise, but their performance is measured against a manually created 'gold standard' segmentation [3]. Additionally, manual segmentation is required to train automatic segmentation algorithms, as well as in problems where automatic segmentation is not sufficiently reliable.

As anyone who has ever done manual image segmentation can attest, it becomes monotonous very quickly and thus also may become less reliable. This is especially true for high-resolution 3D images with many slices. One strategy to reduce the need for repetitive tracing is inter-slice interpolation (Fig. 1). Namely, in regions where the anatomy is slowly changing and adjacent slices are quite similar, some slices can be skipped by a human segmenter and then interpolated by an algorithm.

Albu's method [1] is centered around interpolating a single shape on slice i into a single shape on slice j. Shapes are contiguous groups of pixels with the same label value. Correspondence between shapes is determined by examining overlaps of shapes from from slices i and j. If a shape on one slice overlaps more than one shape on the other slice, the one shape is split into disjoint regions based on dilations starting from overlaps (intersections), and those disjoint regions are interpolated as if they were separate shapes.

Before interpolating i shape into j shape, they are aligned to achieve maximum shape overlap with minimum shape displacement. When writing out the interpolated slices, the opposite translation is applied. To create gradual transition, the overlap shape o is created by intersection of shapes i and j. Then two transition sequences are generated: $o \rightarrow i$ and $o \rightarrow j$. Each step in these sequences is a one-pixel dilation of the previous step, constrained by the final shape (to not leak outside). Then $o \rightarrow j$ sequence is reversed (to form $j \rightarrow o$) and merged with $o \rightarrow i$ by forming union at each step. The resulting sequence is a smooth transition from shape i into j. The final interpolating shape is the median in the sequence, such that its pixel-count difference to both i and j shapes is as equal as possible.

This approach retains some distinctive features, unlike some surface interpolation approaches which tend to smooth the shapes which are being interpolated. If there is large overlap and no shifting of structures between contours being interpolated, this method works well. If the genus of the object changes, this approach might also be better than some traditional shape interpolation methods. But if a structure appears to be moving along the boundary (an elliptical shape structure appears to rotate between slices), if the center-line of the object is not orthogonal to the slices so that the object appears to translate between slices, or if the overlap is low, then a shape-based method (that incorporates constraints to preserve boundary curvature or the total area of the interpolated structure) might be preferred. Also, this method does not involve changing representation into a surface, so it can be implemented efficiently which is of significance in practice.

Distributed under Creative Commons Attribution License

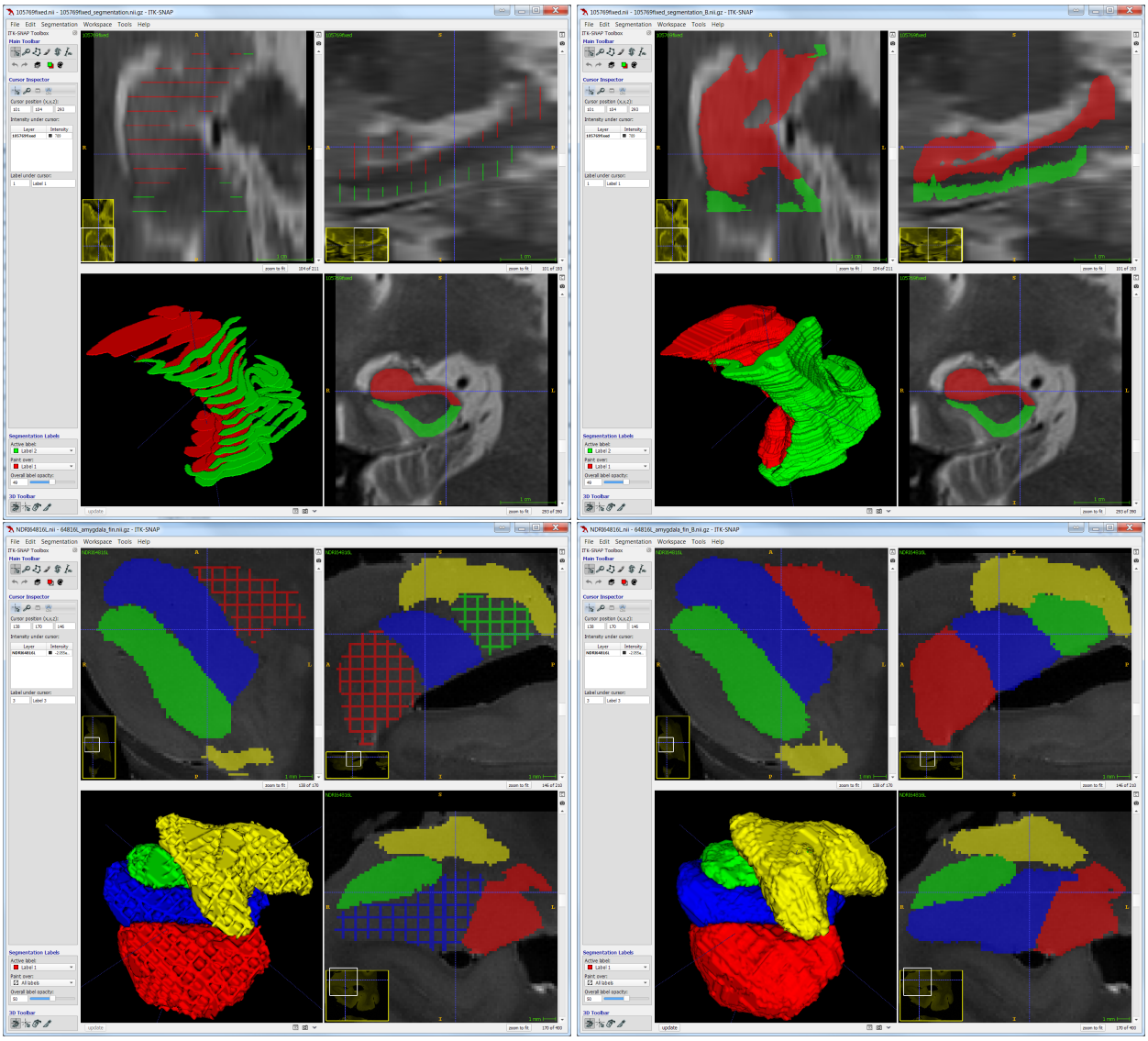

Figure 1: Examples of inputs (left) and outputs (right). First row: hippocampal sub-fields in in-vivo T_2-weighted MRI, two labels, manual segmentation on coronal slices only. Second row: amygdalar nuclei in ex-vivo 9.4 Tesla MRI, four labels, manual segmentation along all three principal axes of the body.

Manual corrections are possible after invoking this filter through ITK-SNAP, as well as undo, redo and other standard editing features. This is important for cases of imperfect interpolation, and correcting an incorrect segmentation is usually quicker than doing it manually in each slice.

2 Class description

`itk::MorphologicalContourInterpolation` is a class derived from `itk::ImageToImageFilter`. Input is n-dimensional, usually 3D. 'Slices' are cross-sections along one axis, usually 2D. Locations of manually segmented slices can be set using `SetUseCustomSlicePositionsOn` and `SetLabeledSliceIndices`. The default is auto-detection of manually segmented slices using a heuristic.

Distributed under Creative Commons Attribution License

The filter can be restricted to interpolate only along one axis and/or to interpolate just one label. The default is to interpolate all labels along all axes.

The shapes in 'source slices' need to be aligned for interpolation. Albu's method [1] calls for alignment which requires inefficient exhaustive search, so `HeuristicAlignment` is implemented and enabled by default. `HeuristicAlignment` uses breadth-first-search strategy with 'no translation' and 'centroid alignment' as first two seeds. 'No translation' seed is required to prevent exceptional cases in which 'centroid alignment' produces no overlap for some concave shapes. The search is terminated when further expansion of search space (relative translations) leads to pixel overlap counts less than 90% of the current maximum.

The most compute intensive part of the algorithm is the generation of two dilation sequences which gradually transform the shapes. Those dilations can be done with either `itk::BinaryBallStructuringElement` or `itk::BinaryCrossStructuringElement`. The option `UseBallStructuringElement` controls which one is used.

As generation of the dilation sequences is computationally intensive, we also experimented with usage of distance field. A histogram of pixel counts of shapes 'A' and 'B' for different distances can be used to calculate a threshold which directly produces median shape. However, this approach loses the geodesic aspect of shape growth and is not much faster (which was our hope) for our test examples. We decided to retain it, and it is activated by `UseDistanceTransform`.

To minimize the usage of memory, the output of interpolation is written directly into the output image. The interpolation is done on each axis successively, in the order of increasing indices (0, 1, 2, 3 ... or X, Y, Z, t ... or sagittal, coronal, axial, temporal ...), and a union is implicitly made. In case of conflicts between labels, the label with bigger number has precedence (gets written into output). This is in contrast to 'keep results of earlier processed axes and labels' (not writing pixels which have already been written once) and 'overwrite the result by latest processed axis or label'. The chosen convention ensures that the output is not dependent on the variability of execution introduced by multi-threading. This elimination of variability also improves reliability of automated regression tests.

3 Evaluation

To test how well the interpolation works, we tested it on a couple of images with full manual segmentations. Starting from a fully manually segmented image leave every n-th slice (to simulate sparse manual segmentation) and clear rest of the image, then use that as an input to the filter and compare output with the original image (Fig. 2). The images are courtesy of Aleš Neubert [2]. Table 1 is taken from [4].

Dataset	**Sx,y**	**Sz**	**xyR**	**zR**	**AF**	**N_{sVB}**	**MVox**	**A_{iso}**
case_2	0.34	1.2	641	296	3.49	9	121.62	0.52
case_10	0.34	1	636	299	2.91	8	120.94	0.49

Table 1: Imaging information about the datasets used for the quantitative evaluation. $S_{x,y}$ – voxel spacing in $x-$ and $y-$directions (millimeters), S_z – spacing along $z-$axis. xyR – resolution of image along $x-$ and $y-$axes, zR - $z-$resolution. AF – anisotropy factor $\frac{S_z}{S_{xy}}$. N_{sVB} – number of segmented vertebral bodies in a dataset. MVox – millions of voxels. A_{iso} – spacing of isotropic image with equivalent voxel volume.

The tests were done on a PC which was bought in 2015. It has Windows 7, Xeon E3-1220 quad core processor, 32 GB of RAM and a solid state drive.

Figure 3 presents 2D and 3D visualization of quantitative validation of the proposed method. Figure 4

Distributed under Creative Commons Attribution License

provides summary results for a testing sparsity on both test images.

Results demonstrate the expected loss of precision with increasing sparsity of manual segmentations. Figure 5 gives detailed results for a single image with exploration of loss of precision in the three spatial directions.

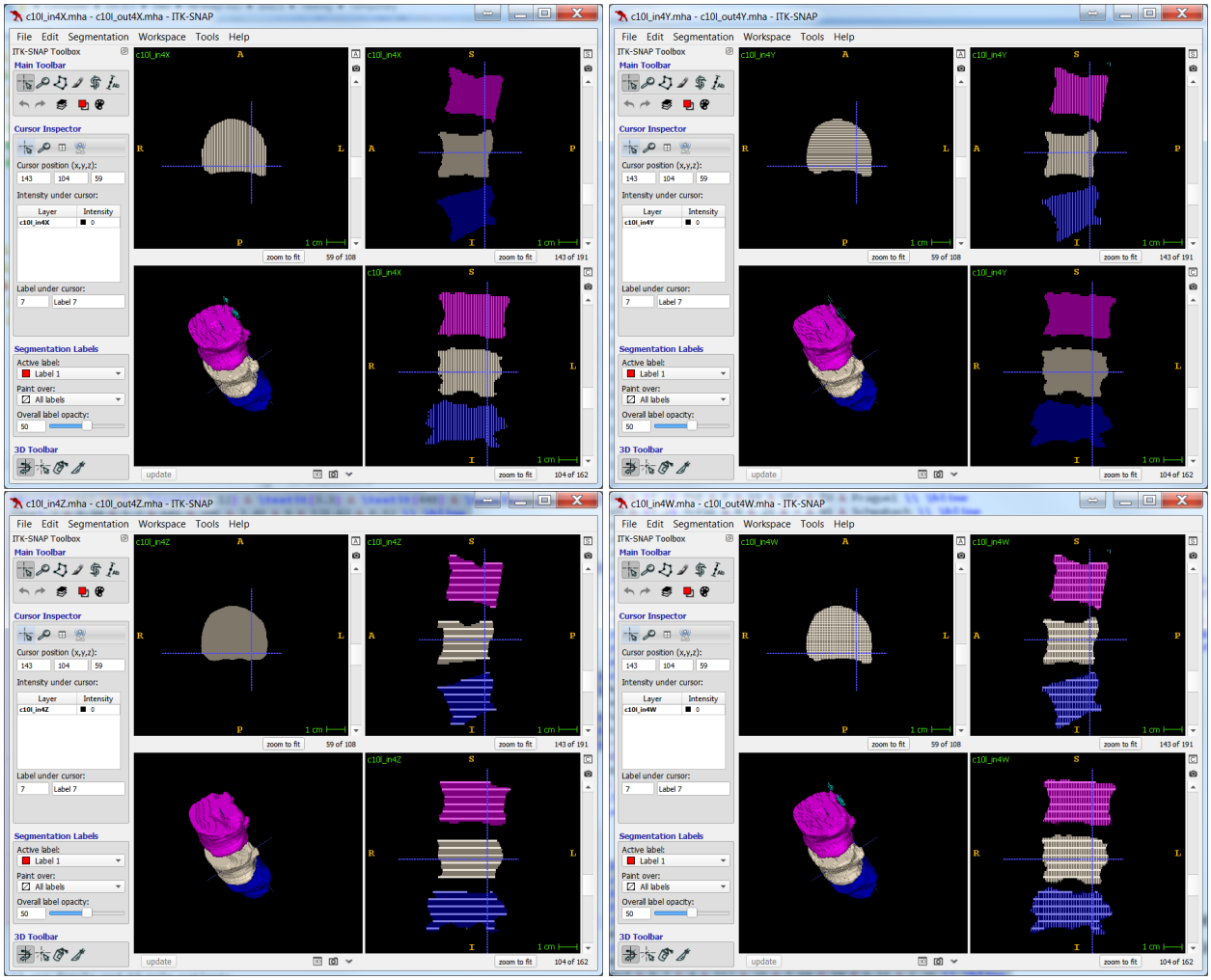

Figure 2: Interpolated output overlaid over a synthetic input with sparsity 4, i.e. every 4th slice from the original manually segmented image is kept. This is a cropped version of case_10 dataset, containing L3, L4 and L5 vertebrae. Top left: synthetic slicing perpendicular to sagittal axis. Top right: coronal. Bottom left: axial. Bottom right: all three.

Distributed under Creative Commons Attribution License

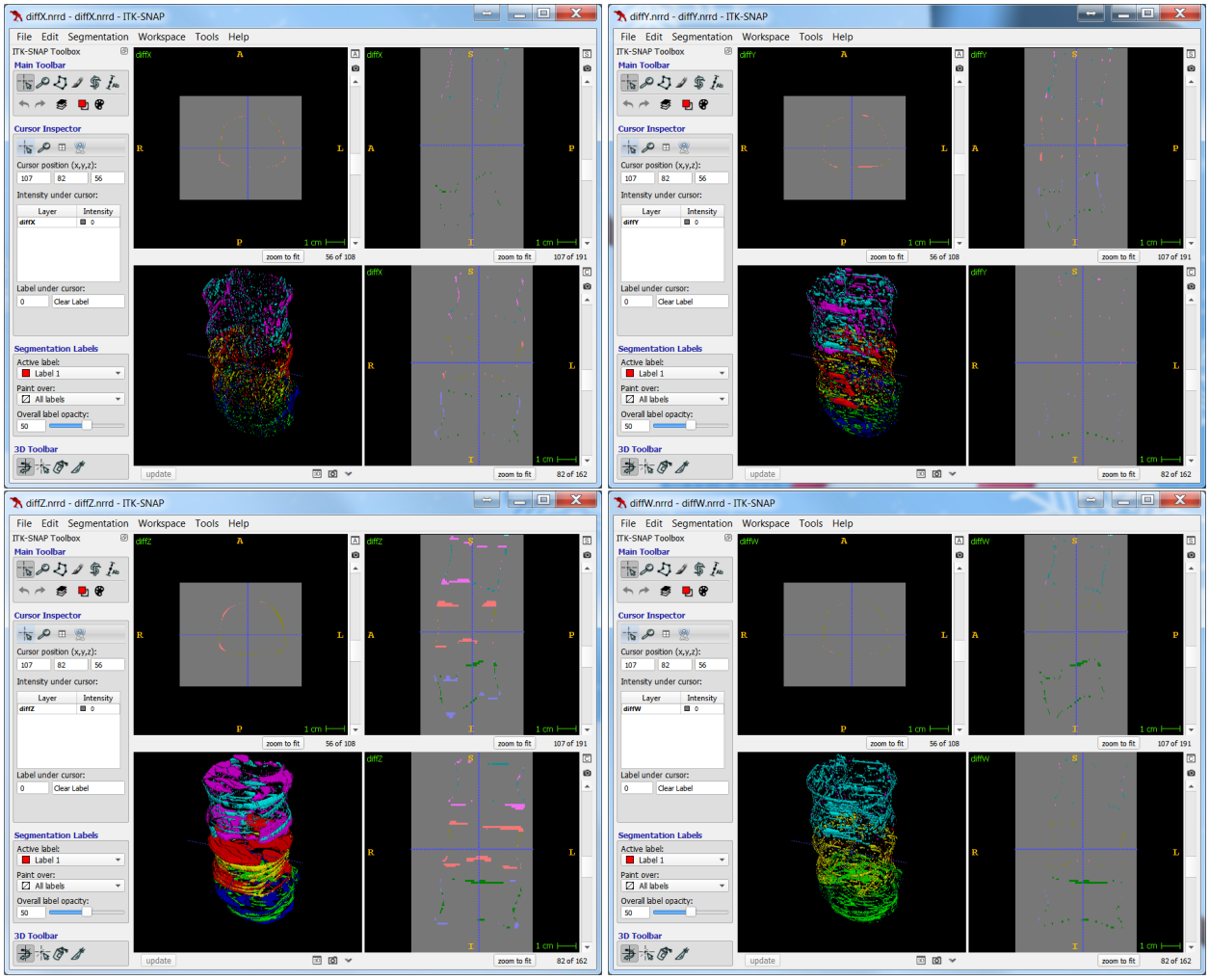

Figure 3: Visualization of the interpolation errors. Same data and order as in Fig. 2. Cyan, yellow and green pixels are false positives (present in interpolated image, absent in ground truth). Violet, red and blue are false negatives (present in ground truth, absent in interpolated image). It can be easily seen that the structure of errors follows the axis of 'manual' segmentation: the segmented slices at regular intervals cause the edges of object perpendicular to that axis to be missing. This is most pronounced for Z axis, because of its thickest spacing. When all three axes are used, false positives dominate.

Distributed under Creative Commons Attribution License

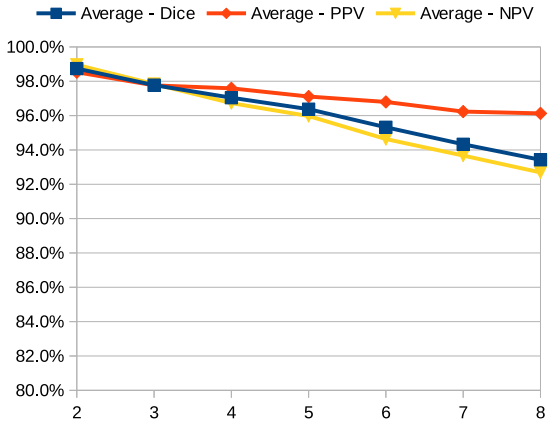

Figure 4: Loss of precision with increasingly sparse 'manual' segmentations. Dice similarity coefficient, positive predictive value and negative predictive value. To calculate PPV and NPV, a background equal in size to the objects is assumed. Averaged over both images and four axes (X, Y, Z, X+Y+Z).

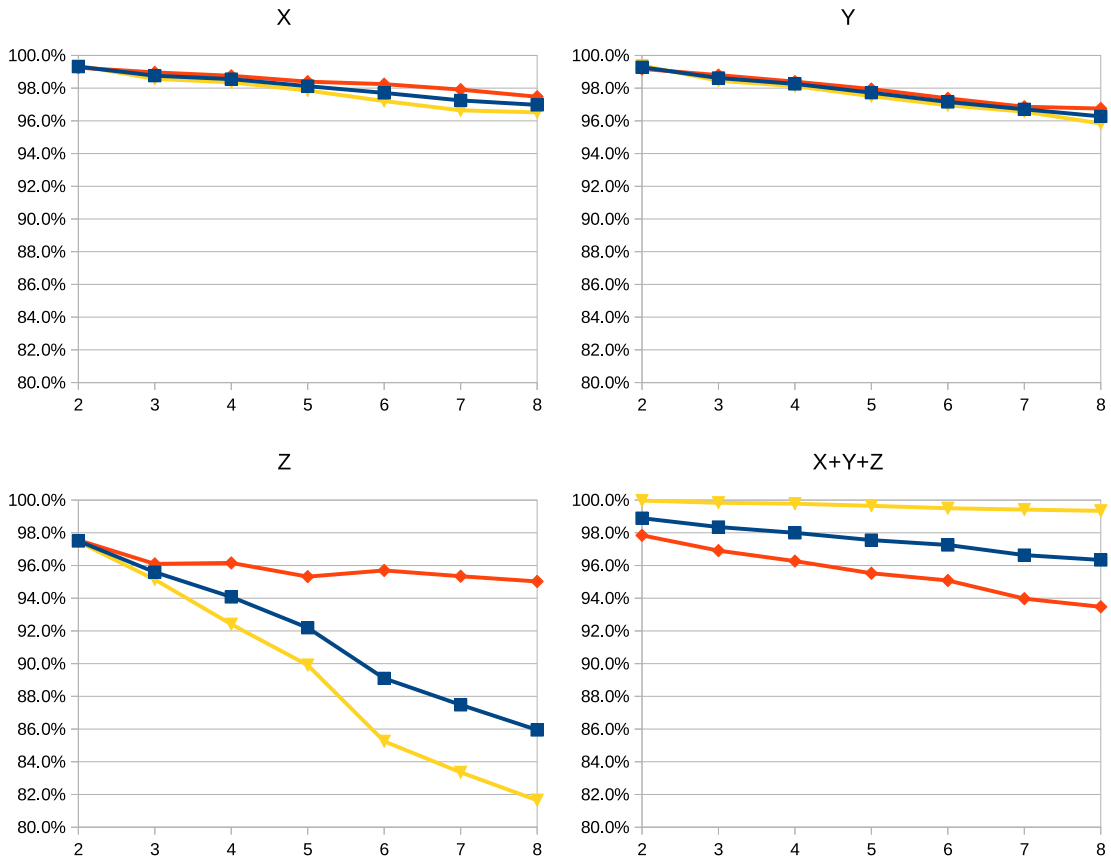

Figure 5: Loss of precision for Case_10. Legend the same as in Fig. 4. Top left: synthetic slicing perpendicular to sagittal axis. Top right: coronal. Bottom left: axial. Bottom right: all three. Because the resolution is already 3× lower along Z axis, adding sparsity exacerbates the problem – most errors are false negatives (pixels from ground truth are absent in the interpolated image).

Distributed under Creative Commons Attribution License

4 Acknowledgments

This work is supported by NIH grant R01 EB014346, 'Continued development and maintenance of the ITK-SNAP 3D image segmentation software'.

References

[1] Alexandra Branzan Albu, Trevor Beugeling, and Denis Laurendeau. A morphology-based approach for interslice interpolation of anatomical slices from volumetric images. *IEEE Transactions on Biomedical Engineering*, 55(8):2022–2038, Aug 2008. 1, 2

[2] Aleš Neubert, Jurgen Fripp, Craig Engstrom, Raphael Schwarz, Lars Lauer, Olivier Salvado, and Stuart Crozier. Automated detection, 3D segmentation and analysis of high resolution spine MR images using statistical shape models. *Physics in Medicine and Biology*, 9:8357–8376, 2012. 3

[3] 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, 2006. 1

[4] Dženan Zukić, Aleš Vlasák, Jan Egger, Daniel Hořínek, Christopher Nimsky, and Andreas Kolb. Robust detection and segmentation for diagnosis of vertebral diseases using routine mr images. *Computer Graphics Forum*, 33(6):190–204, 2014. 3

Distributed under Creative Commons Attribution License