Tubular Geodesics using Oriented Flux: An ITK Implementation

Release 0.00

Fethallah Benmansour, Engin Türetken and Pascal Fua

February 1, 2013

CVLab, École Polytechnique Féderale de Lausanne

Abstract

This document describes an ITK implementation of an interactive method for tracing curvilinear structures. The basic tools provided in this framework are an oriented flux-based tubularity measure and a geodesic path tracer that uses the fast marching algorithm. The framework is efficient and requires minimal user interaction to trace curvilinear structures such as vessels and neurites in 2D images and 3D image stacks.

Contents

1 Introduction

We deal with the problem of finding a complete segmentation of curvilinear structures in 2D images and 3D stacks. The main objective is to compute centerline coordinates and radii of the structures. The proposed method is interactive, requiring only a start and an end point to generate a *geodesic path* between them.

We use a variant of the minimal path method applied in the scale-space domain [4]. As shown in the diagram of Fig. 1, we first compute a *tubularity* value at each image location $\mathbf{x}$ and radius r. This value quantifies the likelihood that there exists a tubular structure of radius r at location $\mathbf{x}$. Given an N-D image, this creates an (N+1)-D scale-space tubularity measure, which, in this work, is taken as the sum of the two dominant eigenvalues of the Optimally Oriented Flux (OOF) matrix [3]. This step is followed by computing a distance map from a given start point using the Fast Marching Algorithm [6]. Finally, the geodesic path is computed by back-propagating from an end point to the start. The method described here is a simplified version of [2] since the considered OOF measure is isotropic.

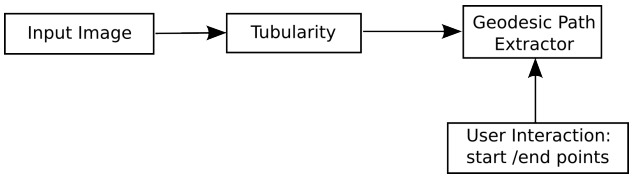

Figure 1: The tubular geodesic computation pipeline.

Compared to the ITK implementation proposed in [5], which provides only centreline locations, our method provides, at the same time, radius estimates. Moreover, our back-propagation step uses the 4^{th} order Runge-Kutta optimizer, and hence, results in fewer oscillations. Finally, in [5] the author claims that "choosing an appropriate speed function is the most difficult part of the entire process". We propose an implementation that provides a well suited speed function to extract tubular structures.

This implementation have been incorporated in the `ImageJ/FIJI` plugin *Simple Neurite Tracer* [1] (SNT) for semi-automatic tracing of curvilinear structures as illustrated in Fig. 2.

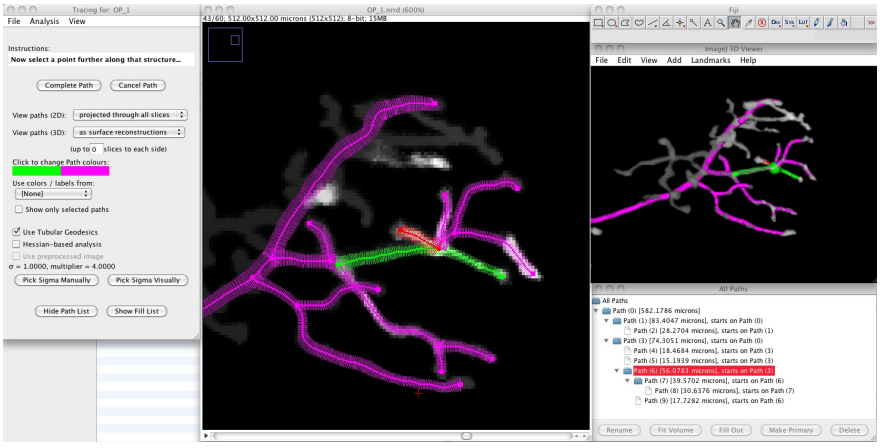

Figure 2: Our extended SNT tracer uses the ITK implementation of the method described here.

[1] http://cvlab.epfl.ch/software/delin/index.php

2 Implementation

In [4], a variant of the classical, purely spatial, minimal path technique is presented. By incorporating an extra *non-spatial* dimension into the search space, the method models a tubular path in a 3D image as a sequence of 4D points (three spatial dimensions for image coordinates and a scale dimension that quantifies curvilinear structure thickness). Thus, each 4D point represents a sphere in 3D space, and the structure is obtained by taking an envelope of these spheres as we move along a curve (see Fig. 3 for a 2D example).

A crucial step of this method is building an informative tubularity measure that drives the propagation. In contrast to the measure used in [4], which requires a few parameters to be tuned, the oriented flux-based measure is parameter free.

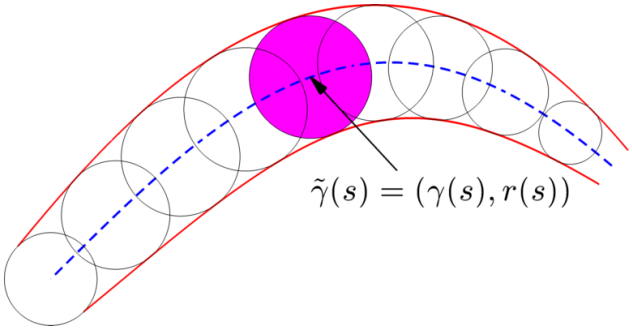

Figure 3: A curvilinear structure is represented as an envelope of a family of cicles with continuously changing center points and radii.

In the following, we describe our implementation of the oriented flux measure and the tubular geodesic method.

2.1 Tubularity Measure

The tubularity measure we use is a function of the OOF matrix [3]. At each image location $\mathbf{x}$, the oriented flux measure is defined as the amount of the image gradient projected along a direction $\mathbf{v}$ flowing out from a 3D local sphere (or a 2D circle) S_r centered at the point $\mathbf{x}$. Let I be the image. The OOF measure is defined as

$$f(\mathbf{x},\mathbf{v};r) = \int_{\partial S_r} ((\nabla(G*I)(\mathbf{x}+\mathbf{h})\cdot\mathbf{v})\mathbf{v})\cdot\mathbf{n}\mathrm{d}a, \tag{1}$$

where G is a Gaussian function with a sufficiently small standard deviation (typically taken as the minimum image spacing or half of it), r is the radius of the sphere (or circle), $\mathbf{h} = r\mathbf{n}$ is the relative position vector along ∂S_r, with $\mathbf{n}$ being the outward unit normal of ∂S_r, and $\mathrm{d}a$ is an infinitesimal area (or length) on ∂S_r. The function f is the flux of the smoothed image gradient $\nabla(G*I)$ projected along direction $\mathbf{v}$ towards the sphere ∂S_r. To detect curvilinear structures having higher intensity values than the background, one would be interested in finding the structure direction at $\mathbf{x}$, which minimizes $f(\mathbf{x},\mathbf{v};r)$, i.e. we are looking for

$$\arg\min_{\mathbf{v}} f(\mathbf{x},\mathbf{v};r). \tag{2}$$

Using the divergence theorem, it can be shown that $f(\mathbf{x},\mathbf{v};r)$ is a quadratic form on $\mathbf{v}$ and its associated matrix can be calculated using a convolution operation,

$$f(\mathbf{x},\mathbf{v};r) = \mathbf{v}^T \left\{I*(\partial_{i,j}G)*\mathbb{1}_{S_r}(\mathbf{x})\right\} \mathbf{v} := \mathbf{v}^T \left\{I*\mathbf{F}_r(\mathbf{x})\right\} \mathbf{v}, \tag{3}$$

where $(\partial_{i,j}G)$ is the Hessian matrix of function G and $\mathbb{1}_{S_r}$ is the indicator function of the sphere (or circle) S_r. $\mathbf{F}_r$ is called the *oriented flux filter*.

By differentiating the above equation with respect to $\mathbf{v}$, minimization of function f is in turn acquired as solving a generalized eigenvalue decomposition problem. Solving the aforementioned generalized eigen decomposition problem gives N eigenvalues for an N-D image, $\lambda_1(\cdot) \leq \cdots \leq \lambda_N(\cdot)$. To handle the structures of varying thickness, we follow the multi-scale approach of Law and Chung [3], which normalizes the eigenvalues of the OOF matrix by the surface area of the sphere ($4\pi r^2$). In the 2D case, the eigenvalues are normalized by the circle perimeter $2\pi r$.

The normalized convolution $I * \mathbf{F}_r / r^{N-1}$ is performed in the Fourier domain for efficiency and implemented in the `itk::OrientedFluxMatrixImageFilter` class. . This filter is templated over the input image type and the output image type for which the default pixel type is `itk::SymmetricSecondRankTensor`, since the OOF provides a symmetric matrix. The filter `itk::MultiScaleTubularityMeasureImageFilter` implements a multi-scale oriented flux measure by summing the $(N-1)$ eigenvalues of the OOF matrix that correspond to the cross-section of the tubular structures. In addition to the (N+1)-D tubularity measure, the filter optionally outputs scale and OOF matrix images. See `itkMultiScaleTubularityMeasureImageFilter.cxx` for a complete example.

2.2 Tubular Geodesic

A *tubular geodesic* is a path, linking two points, that globally minimizes an energy functional of the tubularity measure. Without loss of generality and in order to simplify notations we will assume hereinafter that a path γ is parametrized along its length s ($0 \leq s \leq 1$). The energy minimized by a geodesic is under the form:

$$E(\gamma) = \int \mathcal{P}\big(\gamma(s)\big)\mathrm{d}s. \tag{4}$$

where $\mathcal{P}(\mathbf{x}) > 0$ is the tubularity measure described in the previous section.

For an N-D image, a geodesic path connecting two (N+1)-D points $\mathbf{p}_1$ and $\mathbf{p}_2$, globally minimizes the above energy (4) and is noted as $\mathcal{C}_{\mathbf{p}_1,\mathbf{p}_2}$.

The solution of this minimization problem is obtained through the computation of the *geodesic distance* $\mathcal{U} : \Omega \to \mathbb{R}^+$ associated with $\mathbf{p}_1$ on the domain $\Omega \subset \mathbb{R}^N$. The geodesic distance is the minimal energy integrated along a path between $\mathbf{p}_1$ and any point $\mathbf{x}$ of the domain Ω :

$$\forall\, \mathbf{x} \in \Omega,\ \mathcal{U}(\mathbf{x}) = \min_{\gamma \in \mathcal{A}_{\mathbf{p}_1,\mathbf{x}}} \left\{ \int \mathcal{P}\big(\gamma(s)\big)\mathrm{d}s \right\}, \tag{5}$$

where $\mathcal{A}_{\mathbf{p}_1,\mathbf{x}}$ is the set of paths connecting $\mathbf{x}$ to $\mathbf{p}_1$. The values of $\mathcal{U}$ may be regarded as the arrival times of a front propagating from the source $\mathbf{p}_1$ with velocity $1/\mathcal{P}$. $\mathcal{U}$ satisfies the Eikonal equation

$$\|\nabla \mathcal{U}(\mathbf{x})\| = \mathcal{P}(\mathbf{x}) \qquad \forall\, \mathbf{x} \in \Omega, \text{ and } \mathcal{U}(\mathbf{p}_1) = 0, \tag{6}$$

The map $\mathcal{U}$ has only one global minimum, the source point $\mathbf{p}_1$, and its flow lines satisfy the Euler-Lagrange equation of functional (4). Therefore, the minimal energy path $\mathcal{C}_{\mathbf{p}_1,\mathbf{p}_2}$ can be retrieved with a simple gradient descent on $\mathcal{U}$ from $\mathbf{p}_2$ to $\mathbf{p}_1$ by solving the following ordinary differential equation with standard numerical methods such as Heun's or Runge-Kutta's methods.

$$\frac{\mathrm{d}\mathcal{C}_{\mathbf{p}_1,\mathbf{p}_2}}{\mathrm{d}s}(s) \propto -\nabla \mathcal{U}\big(\mathcal{C}_{\mathbf{p}_1,\mathbf{p}_2}(s)\big), \text{ with } \mathcal{C}_{\mathbf{p}_1,\mathbf{p}_2}(0) = \mathbf{p}_1 \text{ and } \mathcal{C}_{\mathbf{p}_1,\mathbf{p}_2}(1) = \mathbf{p}_2. \tag{7}$$

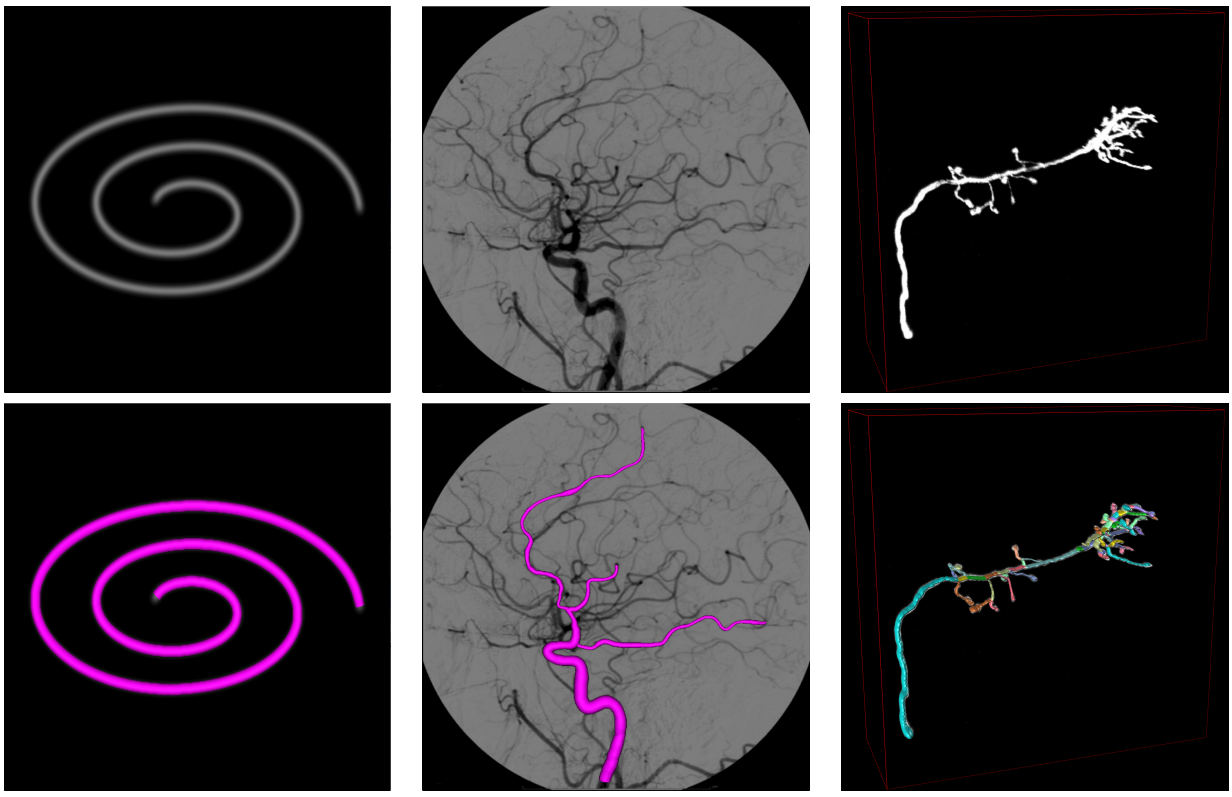

Figure 4: First row: from left to right, a synthetically generated image containing a spiral shape, a 2D cerebral angiogram, a 3D stack of olfactory projection fibers (OP) from the DIADEM competition [1]. Second row: tracing results overlaid on top of the images. Each branch in the structures is traced by manually providing a start and an end point. The reconstruction for the OP stack is color coded for easier visibility.

The filter class that computes a geodesic path from a given tubularity image is `itk::TubularMetricToPathFilter`, which is a subclass of `itk::ImageToPathFilter`. The filter expects one start point, a set of end points, and an (N+1)-D tubularity image to drive the front propagation. The tubularity image must be a scalar one with real and strictly positive pixel values.

The front propagation step is carried out using the ITK filter class `itk::FastMarchingUpwindGradientImageFilter`. The back-propagation can be performed using a discrete neighborhood iterator, `itk::IterateNeighborhoodCharacteristicDirectionsToPathFilter` for discrete centreline location and radius estimates, or a fixed step gradient descent optimizer, `itk::FixedStepDescentCharacteristicDirectionsToPathFilter` for sub-pixel accuracy, similar to the ones proposed in [5]. However, both of these approaches result in oscillations along path centerlines especially when the curvilinear structures are faint and the image is noisy. To achieve robustness against these factors, we implemented the 4^{th} order Runge-Kutta back propagation filter `itk::RK4CharacteristicDirectionsToPathFilter`.

3 Examples

We've evaluated our pipeline on a large variety of noisy 2D images and 3D images stacks. Figure 4 shows three sample results on a synthetically generated image, a 2D cerebral angiogram and a 3D stack of neurite fibers. Maximum intensity projections of the tubularity stacks for each scale level are shown in Fig. 5.

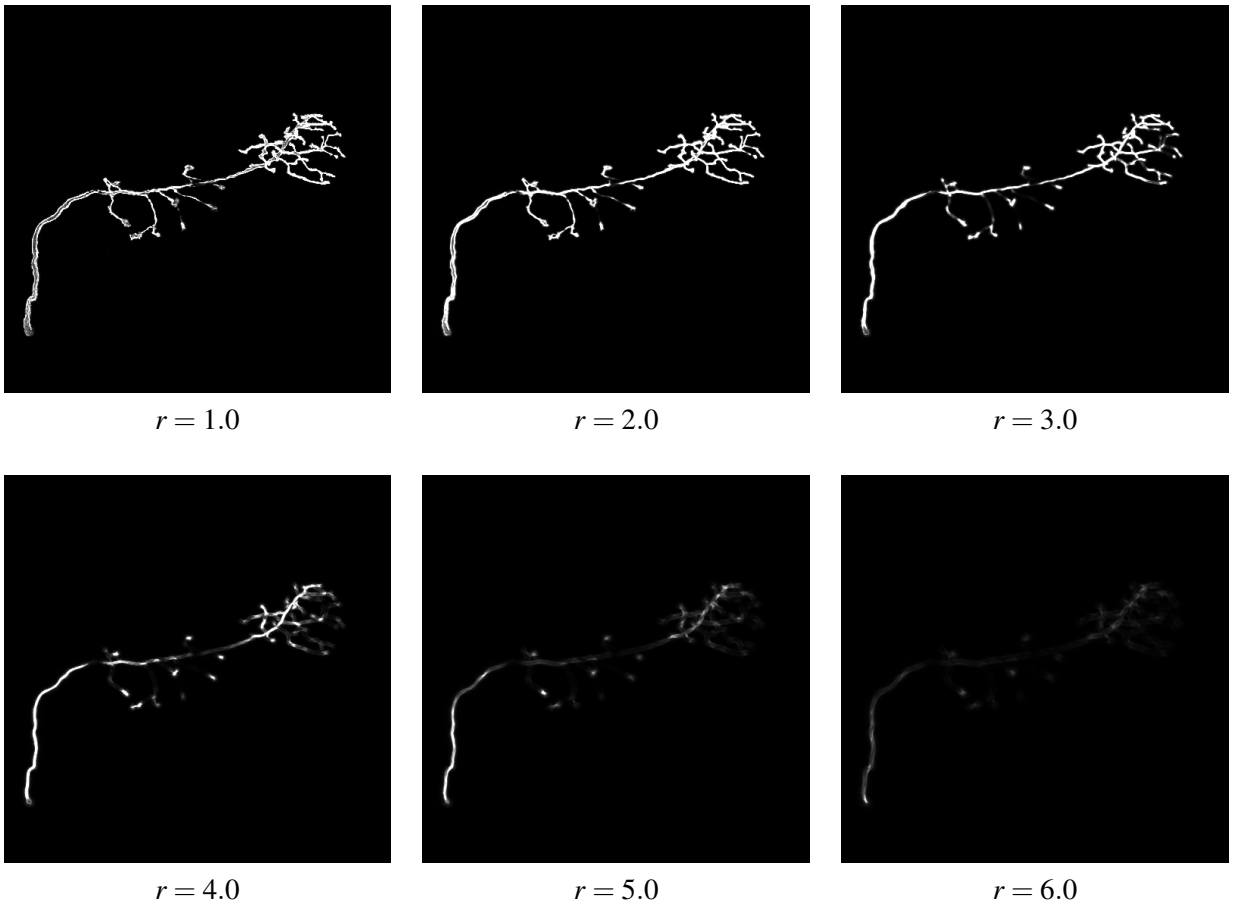

Figure 5: Maximum intensity projections of the computed tubularity volumes for the OP stack shown in the last column of Fig. 4. Each image corresponds to a different radius level r, which is given below it.

In the following, we give an example usage of the tubularity and geodesic computation steps for the 3D stack shown in Fig. 4.

```
// ================================
// TUBULARITY IMAGE COMPUTATION
// ================================

// Constants and typedefs.
const unsigned int Dimension = 3;

typedef itk::Image<unsigned char, Dimension>              InputImageType;
typedef itk::SymmetricSecondRankTensor<float, Dimension>  OrientedFluxPixelType;
typedef itk::Image<float,Dimension>                       OutputImageType;
typedef itk::Image<float,Dimension+1>                     OutputScaleSpaceImageType;
typedef itk::Image< OrientedFluxPixelType, Dimension >    OrientedFluxImageType;
typedef itk::Image<float, Dimension>                      ScaleImageType;
typedef itk::OrientedFluxCrossSectionTraceMeasureFilter
<OFImageType, OutputImageType >                           OFObjectnessFilterType;
typedef itk::MultiScaleOrientedFluxBasedMeasureImageFilter
<InputImageType,
OFImageType,
ScaleImageType,
OFObjectnessFilterType,
OutputImageType >                                         OFMultiScaleFilterType;

// Read the input image.
InputImageType::Pointer inputImage = ...
```

```
25
26 // Declare and allocate the multi-scale tubularity measure filter.
27 OFMultiScaleFilterType::Pointer ofMultiScaleFilter =
28 OFMultiScaleFilterType::New();
29
30 // Set the input and the parameters of the filter.
31 ofMultiScaleFilter->SetInput( inputImage );
32 ofMultiScaleFilter->SetBrightObject( true );
33 ofMultiScaleFilter->SetSigmaMinimum( 1.0 );
34 ofMultiScaleFilter->SetSigmaMaximum( 6.0 );
35 ofMultiScaleFilter->SetNumberOfSigmaSteps( 11 );
36 ofMultiScaleFilter->SetFixedSigmaForOrientedFluxImage( 0.5 );
37 ofMultiScaleFilter->SetGenerateScaleOutput( false );
38 ofMultiScaleFilter->SetGenerateOrientedFluxOutput( false );
39 ofMultiScaleFilter->SetGenerateNPlus1DOrientedFluxOutput( false );
40 ofMultiScaleFilter->SetGenerateNPlus1DOrientedFluxMeasureOutput( true );
41
42 // Run the filter.
43 ofMultiScaleFilter->Update();
44
45 // Get the (N+1)-D tubularity measure image output.
46 OutputScaleSpaceImageType::Pointer scaleSpaceTubul =
47 ofMultiScaleFilter->GetNPlus1DImageOutput(); // (N+1)-D scale-space tubularity image.
48
49
50 // ================================
51 // TUBULAR GEODESIC COMPUTATION
52 // ================================
53
54 // Constants and typedefs.
55 typedef itk::TubularMetricToPathFilter< OutputScaleSpaceImageType > MetricToPathFilterType;
56 typedef MetricToPathFilterType::PathType                            ScaleSpacePathType;
57 typedef OutputScaleSpaceImageType::RegionType                       ScaleSpaceRegionType;
58 typedef itk::PolyLineParametricTubularPath< Dimension >             PathType;
59
60 // Read/Set the start and end points.
61 itk::Index<Dimension+1> startPoint = ...
62 itk::Index<Dimension+1> endPoint = ...
63
64 // Declare and allocate the tubular geodesic filter.
65 MetricToPathFilterType::Pointer pathFilter = MetricToPathFilterType::New();
66
67 // Set the scale-space tubularity measure image and the start & end points.
68 pathFilter->SetInput( scaleSpaceTubul );
69 pathFilter->SetStartPoint( startPoint );
70 pathFilter->AddPathEndPoint( endPoint );
71
72 // Optionally, for efficiency, provide an image sub-region to process.
73 ScaleSpaceRegionType subRegion = ...
74 pathFilter->SetRegionToProcess( subRegion );
75
76 // Run the filter.
77 pathFilter->Update();
78
79 // Get the resulting path, which is a sequence of (Dimension+1) dimensional
80 // points in image coordinates. Therefore, it is a (Dimension+1) dimensional curve.
81 ScaleSpacePathType::Pointer outputPath = pathFilter->GetPath(0);
82
83 // Convert the curve to a tubular path, which contains a sequence
84 // of (Dimension) dimensional points with radius values attached.
85 double radiusOrigin = scaleSpaceTubul->GetOrigin()[Dimension];
86 double radiusSpacing = scaleSpaceTubul->GetSpacing()[Dimension];
87 PathType::Pointer tubularPath =
88 outputPath->ConvertToNMinus1DPath(radiusOrigin, radiusSpacing);
89
90 // Optionally, resample the path with sub-pixel steps and smooth it slightly.
91 tubularPath->Resample(0.5, inputImage.GetPointer());
92 tubularPath->SmoothVertexLocationsAndRadii(1.0, inputImage.GetPointer());
```

```
93
94 // Write the path to the specified swc file in world coordinates.
95 std::string outputSWCFile = ...
96 tubularPath->WriteSwcFile(outputSWCFile, inputImage.GetPointer(), true);
```

4 Conclusion

In this work, we've presented an ITK framework for computing tubular geodesics from point pairs. The framework allows a user to first generate a scale-space tubularity volume with the curvilinear structures enhanced and the background noise suppressed. This volume is then used as a speed function to compute a tubular geodesic for each start and end point pair, which is done using the fast marching algorithm and the 4^{th} order Runge-Kutta gradient descent procedure. This results in a generic and efficient method to trace curvilinear structures in medical imagery.

References

[1] G. A. Ascoli, K. Svoboda, and Y. Liu. Digital Reconstruction of Axonal and Dendritic Morphology Diadem Challenge, 2010. http://diademchallenge.org/. 4

[2] Fethallah Benmansour and Laurent D. Cohen. Tubular structure segmentation based on minimal path method and anisotropic enhancement. 92(2):192–210, 2011. 1

[3] M.W. Law and A.C. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. pages 368–382, 2008. 1, 2.1, 2.1

[4] H. Li and A. Yezzi. Vessels as 4-D curves: Global minimal 4-D paths to extract 3-D tubular surfaces and centerlines. 26(9):1213–1223, 2007. 1, 2

[5] D. Mueller. Fast marching minimal path extraction in itk. 06 2008. 1, 2.2

[6] J.A. Sethian. *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999. 1