Implementation of graph-based interactive 3D vessel segmentation filter

Release 1.00

David Pellow[1], Moti Freiman[1] and Leo Joskowicz[1]

July 15, 2010

[1]School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel

Abstract

This paper describes an ITK implementation of a 3D vascular segmentation filter using a graph-based energy minimization algorithm. The method first computes the shortest path between two user provided points in the vessel and then performs a graph min-cut based segmentation of the vessel based on the intensity information coupled with the computed path as a spatial constraint. The shortest path computation is adapted from the algorithm in [1] to use a vesselness based weighting function. The min-cut algorithm uses the Boost graph library [2] to calculate the minimal cut. Several examples of applications are provided, along with images of the resulting segmentations.

Keywords: min-cut, vesselness, vessel segmentation, ITK, boost

Contents

1 Introduction

Minimum cut graph algorithms are useful in accurately performing vascular segmentation. In this method the image voxels are represented as nodes of a graph, and every node has edges to its neighboring voxels and to a “source” and “sink” node. The weight of edges between voxel nodes is determined based on the similarity in voxel intensity. The source and sink weights depend on the probability that the voxel belongs

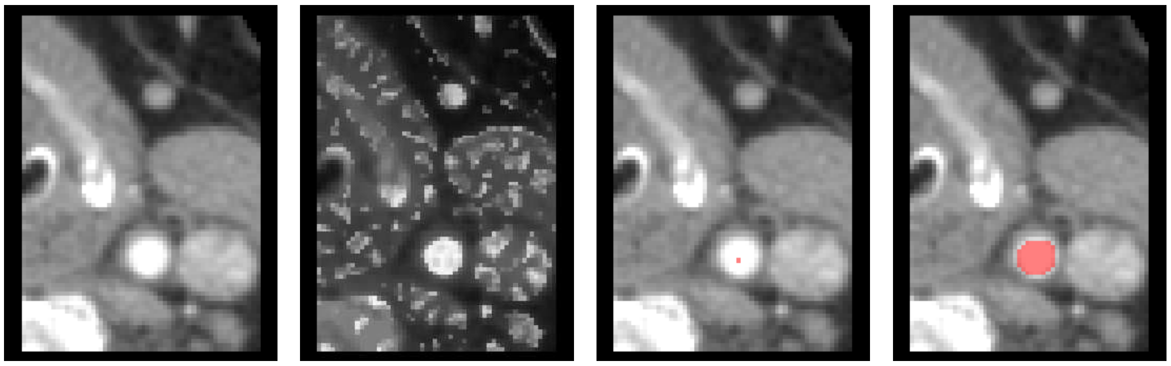
(a) Original image (b) Vesselness filter response (c) Shortest path (d) Final result

Figure 1: An outline of the algorithm: (a) The original image (b) The vesselness image overlayed over the original (c) The shortest path centerline (d) The segmentation cut.

to either the foreground or the background class. The minimum cut classifies each voxel as belonging to the foreground or background class. The original graph min-cut implementation [3] requires extensive user interaction to compute the prior intensity model. In addition, the segmentation is biased to blob-like structures. Therefore it is less suitable for 3D vascular segmentation.

Our implementation uses a two-phase approach. The first phase is the computation of an approximate centerline inside the vessel based only on two user provided seeds. In the second phase the vessel segmentation is computed with a graph min-cut based technique based on both intensity and spatial information. Both intensity and spatial prior information are computed using the results of the first phase.

The algorithm is outlined as follows:

- Input: Start and end points of the vessel
  1. Compute vesselness image
  2. Assign edge weights to voxels based on their vesselness
  3. Compute the shortest path
  4. Use mean and standard deviation along the path to estimate prior values of the image classes
  5. Compute class means and standard deviations and use them to assign voxel sink and source weights
  6. Calculate the minimum graph cut
- Output: Segmentation of the vessel

This algorithm is an improvement of our previous algorithm [4, 5]. The main differences are the use of the vesselness based weighting function to compute the path, the Gaussian mixture model estimation used to determine source and sink weights, and the use of the open-source Boost graph library [2].

Figure 1 shows images created during the different stages of this algorithm. The images are axial slices from the 3D volume.

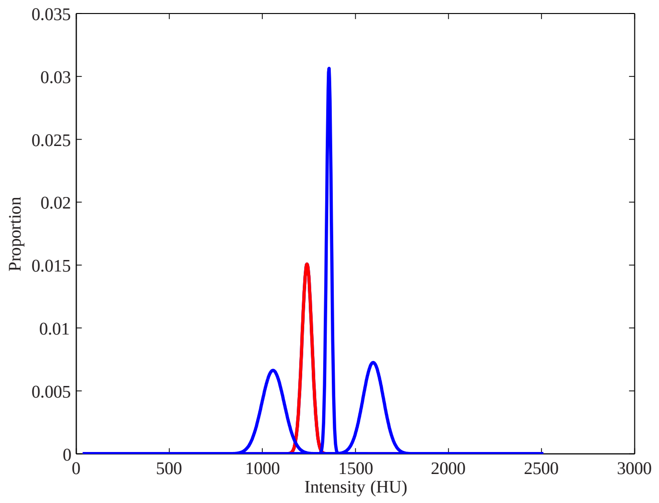

Figure 2: An example of the intensity model that is used as a prior. The vessel class is in red and background classes in other colors.

Shortest-path computation

The shortest-path computation inputs two user seeds, one from the beginning of the vessel, and another one from the end of the vessel. Next, it computes the vesselness response using Frangi's method [6]. The shortest-path is computed using the implementation proposed in [1], in which the vesselness map is used as input. The vesselness image v is rescaled to intensity values between 0 and 500. The edge weight between two voxels i, j is assigned as:

$$w(i,j) = 500 - v(j) \tag{1}$$

The shortest-path filter thus finds a path along which the vesselness is maximized. This step computes a rough estimation of the vessel centerline by maximizing the vesselness response along the path.

Prior intensity model

Given the path along the vessel, we model the intensity information as a mixture of Gaussians. One class is used to represent the vessel intensity model and an additional four classes are used to represent the background intensity. Only a narrow band along the path is used to estimate the intensity model, to limit information that is irrelevant to the separation of the vessel from its background. An ITK based Expectation-Maximization (EM) is used to estimate the model parameters.

The initial prior of the foreground class is based on the previously calculated path through the vessel, and the additional background classes are defined relative to this initial information.

Fig. 2 shows an example of the estimated intensity model.

Spatially constrained graph min-cut segmentation

Let $G = (V,E)$ be the graph corresponding to the image, where $V = \{v_{x_1}, \ldots v_{x_n}, v_s, v_t\}$ are the graph nodes such that node v_x corresponds to voxel x and terminal nodes v_s and v_t correspond to the object and background classes. The graph edges $E = \{(v_x, v_s), (v_x, v_t), (v_x, v_y)\}$ consist of three groups: 1) edges (v_x, v_s) from voxels to the object terminal node; 2) edges (v_x, v_t) from voxels to the background terminal node, and; 3) edges (v_x, v_y) between adjacent voxels (4 or 8 neighbors for 2D images, 6 or 26 neighbors for 3D images). The cost of a cut C that divides the graph into the object class (source vertex) and the background class (terminal vertex) is defined as the sum of the weights of the cut edges $e \in C$.

Edge weights w_e are assigned as follows. The voxel to source edges weights are defined as:

$$w(v_x, v_s) = exp\left(-\frac{(i-\mu_v)^2}{2\sigma_v^2}\right) \tag{2}$$

where μ_v is the mean and σ_v the standard deviation of the vessel class.

The voxel to sink edges weights are defined as:

$$w(v_x, v_t) = \sum_{k=1}^{n} exp\left(-\frac{(i-\mu_k)^2}{2\sigma_k^2}\right) \tag{3}$$

where μ_k and σ_k are the class means and standard deviations of the background classes. To increase the effect of the source weight on voxels close to the path and of the sink weight on voxels farther from the path, source weights are divided by a sigmoid function s and sink weights are multiplied by it. The sigmoid function s is defined as:

$$s = \left|\frac{2}{1+e^{-\alpha x}} - 1\right| \tag{4}$$

where x is the Euclidean distance from the path. In this implementation α has a value of 0.8.

The minimal cut is then computed using Boykov's algorithm [3] as implemented in the Boost [2] open-source library.

In principle, the algorithm can be implemented in n-dimensions, but since the vesselness filter is not templated over the dimension, our implementation performs only 3-dimensional segmentation.

2 Proposed class and implementation

2.1 Overview

This project consists of two filters, the `itk::ActiveContourMinCutImageFilter`, and the `itk::TwoPointsVesselSegmenterImageFilter` which utilizes it. The additional utility class `itk::GMMEstimatorImageFilter` is used to wrap the classes that are required by ITK to compute the prior intensity model into one class. The min-cut filter calculates the minimum cut using Boykov's algorithm [3] as implemented in the Boost graph library [2]. To reduce the memory requirements and to improve the accuracy, the cut is computed only on a user-defined narrow band along the path.

The `itk::TwoPointsVesselSegmenterImageFilter` performs a vessel segmentation by first computing a path through the vessel and then passing this path to the min-cut filter as the region of interest. The path is computed using the shortest path algorithm from [1] with a vesselness based weighting function.

2.2 Implementation

In this section we will give examples of how to call the different classses used in our algorithm.

Shortest-path computation

The shortest-path filter is given two user provided seeds

```
typename itkShortestPathImageFilterType::Pointer
                dijkstra = itkShortestPathImageFilterType::New();

//set the input image
dijkstra->SetInput (this->GetInput());

//set the start and end points
dijkstra->SetStartPoint (m_StartPoint);
dijkstra->SetEndPoint (m_EndPoint);

//set the neighbors mode
dijkstra->SetFullNeighborsMode (m_FullNeighborsMode);
dijkstra->Update();
```

Prior intensity model

The `itk::ActiveContourMinCutImageFilter` filter uses the path to compute the class intensity information using the `itk::GMMEstimatorImageFilter`:

```
typename itkGMMEstimatorImageFilterType::Pointer
                meansEstimator = itkGMMEstimatorImageFilterType::New();

//set the input image
meansEstimator->SetInput(inputImage);

meansEstimator->SetNumberOfClasses(numberOfClasses);

//arrays of initial class means, standard deviations, and proportions
//are computed from the output of the shortest path filter
meansEstimator->SetInitialParams
                (initialMeans, initialStd, initialProps);

meansEstimator->Update();

//the final value of each class mean and standard deviation is
//assigned to class_mean and class_std
meansEstimator->GetClassParams(i, class_mean, class_std);
```

Min-cut filter

Both the original image and the shortest-path are passed on to the `itk::ActiveContourMinCutImageFilter`:

```
typename itkActiveContourMinCutImageFilterType::Pointer
                minCut = itkActiveContourMinCutImageFilterType::New();

//Set the input image
minCut->SetInput (this->GetInput());

//Set the output of the shortest path filter as the region of interest
minCut->SetObjImage (outputImage);

//Set the radius
minCut->SetUncertaintyRadius (UncertaintyRadius);

//Set the number of background classes within the region of interest
minCut->SetNumberOfBackgroundClasses(number_of_classes);

//Set initial means of the background classes
minCut->SetInitialMeans(initial_means_vector);

//Set initial proportions for all classes - the first value is for the
//vessel class, and the last value for is for the black masked out area
minCut->SetInitialProportions(initial_props_vector);

minCut->Update();
```

Using the `itk::TwoPointsVesselSegmenterImageFilter`

The `itk::TwoPointsVesselSegmenterImageFilter` instantiates both the shortest-path image filter and min-cut image filter. It passes the output of the shortest-path filter to the min-cut filter. The user provides the seed points and can set the neighbors and the radius of the region of interest for the min-cut filter. In this example the filter's default parameters are used. The defaults are: full neighbors mode - on, and min-cut radius - 5.0, number of background classes - 4, initial background means - $[-200, 100, 200, 300]$, initial proportions - $[0.01, 0.04, 0.005, 0.003, 0.002, 0.94]$.

```
//instantiate the seed points
PointType p1, p2;

//get the seed points from the user

//instantiate the initial class vectors
std::vector<double> initialMeans;
std::vector<double> initialProps;

//get the initial class data from the user

FilterType::Pointer filter = FilterType::New();

//set the input image
```

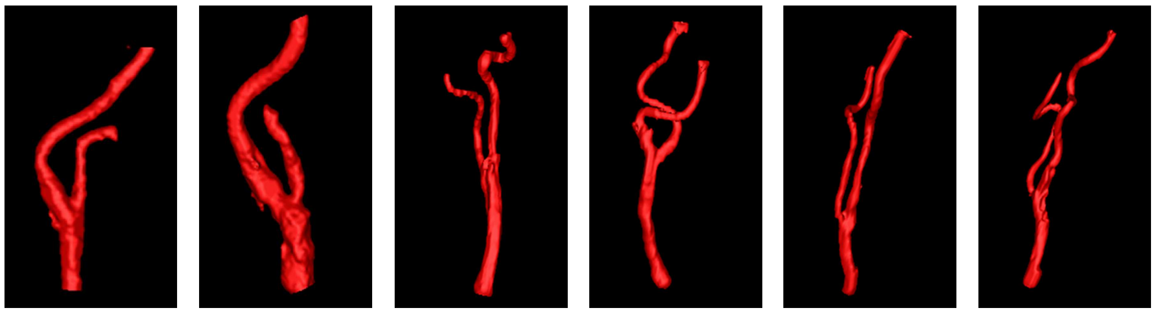
(a) (b) (c) (d) (e) (f)

Figure 3: Carotid artery bifurcation segmentation examples. To produce these images two segmentations were combined, one with start point at the base of the carotid and end point at the end of the left side of the bifurcation, and the other with the same start point and end point at the end of the right side of the bifurcation.

```
filter->SetInput(reader->GetOutput());

//set the seed points
filter->SetStartPoint (p1);
filter->SetEndPoint (p2);

filter->Update();
```

3 Examples

This section provides several examples of the results of the `itk::TwoPointsVesselSegmenterImageFilter` running on representative exmaples from the CLS2009 database [7]. The resulting carotid artery segmentations can be seen in Figure 3.

To run the example inside the provided code, You may type the following command in your commandline shell:

```
$> ./3D_test ./Data/3D_test.vtk ./Results/output.vtk -12.32 138.21 5.50
-15.31 119.09 70.00
```

4 Software Requirements

You need to have the following software installed:

- Insight Toolkit 2.4 [8].
- CMake 2.2 [9].
- Boost C++ libraries 3.7 [2].

References

[1] L. J. L. Weizman, M. Freiman, “Implementation of weighted Dijkstra’s shortest-path algorithm for n-D images,” *Insight Journal*, 2009. (document), 1, 2.1

[2] http://www.boost.org. (document), 1, 1, 2.1, 4

[3] Y. Boykov and G. Funka-Lea, “Graph Cuts and Efficient N-D Image Segmentation.,” *Int. J. of Comp. Vision*, vol. 70, no. 2, pp. 109–131, 2006. 1, 1, 2.1

[4] M. Freiman, N. Broide, M. Natanzon, L. Weizman, E. Nammer, O. Shilon, J. Frank, L. Joskowicz, and J. Sosna, “Vessels-Cut: A Graph Based Approach to Patient-Specific Carotid Arteries Modeling,” in *Proc. of the 2nd 3D Physiological Human workshop, 3DPH’09*, vol. 5903 of *LNCS*, pp. 1–12, 2009. 1

[5] M. Freiman, J. Frank, L. Weizman, E. Nammer, O. Shilon, L. Joskowicz, and J. Sosna, “Nearly automatic vessels segmentation using graph-based energy minimization,” in *3D Segmentation in the Clinic: Carotid Lumen Segmentation and Stenosis Grading Challenge (http://cls2009.bigr.nl)*, 2009. http://www.insight-journal.org/browse/publication/661. 1

[6] A. Frangi, W. Niessen, K. Vincken, and M. Viergever, “Multiscale Vessel Enhancement Filtering,” in *Proc. of the 1st Int. Conf. on Med. Image Comp. and Comp. Aided Interventions, MICCAI’98*, vol. 1496 of *LNCS*, pp. 130–137, 1998. 1

[7] R. Hameeteman, M. Zuluaga, L. Joskowicz, M. Freiman, and T. van Walsum, “3D segmentation in the clinic: Carotid lumen segmentation and stenosis grading challenge,” in *3D Segmentation in the Clinic: Carotid Lumen Segmentation and Stenosis Grading Challenge*, 2009. http://cls2009.bigr.nl. 3

[8] http://www.itk.org. 4

[9] http://www.cmake.org. 4