Vessel Enhancing Diffusion Filter

Release 2.00

Andinet Enquobahrie [1], Luis Ibanez[1], Elizabeth Bullitt[2] and Stephen Aylward[1]

September 13, 2007

[1]Kitware Inc.
[2]CASILab, The University of North Carolina.

Abstract

This paper describes vessel enhancing diffusion (VED) filters implemented using the Insight Toolkit (ITK) [2]. The filters are implementation of the VED algorithm developed by Manniesing et al [4]. The VED algorithm follows a multiscale approach to enhance vessels using an anisotropic diffusion scheme guided by a vesselness measure at the pixel level. Vesselness is determined by geometrical analysis of the Eigen system of the Hessian matrix. For this purpose, a smoothed version of the Frangi's vesselness function [1] is formulated. Experiments were conducted to evaluate the performance of the VED filters in enhancing vessels in lung CT scans.

Contents

1 Introduction

Accurate quantification and visualization of vascular structures is critical in diagnosis and treatment of vascular diseases. Successful interventional clinical procedures such as bypass surgery and coronary artery stenting require the accurate vascular structure visualization during the planning stages. Similarly, effective diagnostic procedures such as stenosis grading depend on the accurate vascular structure quantification.

Various vascular imaging techniques are deployed in clinical practices. Among two dimensional techniques, Digital Subtraction Angiography (DSA) is one of the most commonly used technique for the visualization of blood vessels. Three dimensional imaging techniques such as CTA (Computed Tomography Angiography) and MRA (Magnetic Resonance Angiography) are also common in the clinical setting.

Vessel segmentation algorithms can be applied to 2D and 3D vascular images. Several segmentation techniques have been developed. A review of several techniques is given in [3].

To increase the effectiveness of segmentation algorithms, vessel enhancement procedures are often first applied as a preprocessing step [1]. The performance of the enhancement algorithm has been shown to tremendously impact the results of the segmentation algorithm.

Vessel enhancement algorithms are also useful for the visual interpretation of 3D vascular images. For example, clinicians often generate maximum intensity projects (MIP) images for the visual analysis of the massive amount of data produced by 3D imaging techniques. However, the occurrence of overlapping non-vascular anatomical structures greatly affects vascular visualization in MIP images. Additionally, small blood vessels with low contrast edges are often not clearly visible in MIP images. To alleviate these problems, enhancement algorithms can be first applied tn the vascular images to suppress non-vascular structures and improve the delineation of small blood vessels.

This paper presents an open-source implementation of a vessel enhancement algorithm called VED [4].

1.1 Overview of VED algorithm

The VED algorithm is based on anisotropic diffusion scheme guided by vessel-likelihood at pixel level. It is basically a smoothing filter with the strength and direction of diffusion is determined by a ”‘vesselness”’ measure. Vesselness is measured by analyzing the eigen system of the Hessian matrix. Several vesselness functions have been proposed. Manniesing’s VED algorithm ([4])is based on Frangi’s vesselness function. For increasing-magnitude eigen values of a Hessian matrix

$$|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$$

Frangi’s vesselness function is composed of three components formulated to discriminate tubular structures from blob-like and/or plate-like structure as shown in Equation 1

$$V_F(\lambda) = \begin{cases} 0 & \text{if } \lambda_2 > 0 \text{ or } \lambda_3 > 0 \\ (1 - e^{-\frac{R_A^2}{2\alpha^2}}) \,.\, e^{-\frac{R_B^2}{2\beta^2}} \,.\, (1 - e^{-\frac{S^2}{2\gamma^2}}) & \text{otherwise} \end{cases} \quad (1)$$

With

$$R_A = \frac{|\lambda_2|}{|\lambda_3|} \quad (2)$$

$$R_B = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}} \tag{3}$$

$$S = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2} \tag{4}$$

α, β, γ are thresholds which control the sensitivity of the vesselness measure.

However, Frangi's vesselness function is not continuous and can't be used in the diffusion process. Hence, Manniesing et al proposed a smoothed version of Frangi's vesselness function as shown below.

$$V_S(\lambda) = \begin{cases} 0 & \text{if } \lambda_2 \geq 0 \text{ or } \lambda_3 \geq 0 \\ (1 - e^{-\frac{R_A^2}{2\alpha^2}}) \, . \, e^{-\frac{R_B^2}{2\beta^2}} \, . \, (1 - e^{-\frac{S^2}{2\gamma^2}}) \, . \, e^{-\frac{2c^2}{|\lambda_2|\lambda_3^2}} & \text{otherwise} \end{cases} \tag{5}$$

For a multiscale analysis, the vesselness function is computed for a range of scales and the maximum response is selected.

$$V = max_{\alpha_{min} \leq \alpha \leq \alpha_{max}} \; V_s(\lambda) \tag{6}$$

Next, a diffusion tensor is defined in such a way that diffusion is promoted along the vessel but prohibited perpendicular to the vessel.

$$D = Q\lambda' Q^T \tag{7}$$

Where Q is a matrix containing eigen vectors of the Hessian matrix and λ' is a diagonal matrix containing the following elements

$$\lambda_1' = 1 + (w-1)\, V^{\frac{1}{S}}$$
$$\lambda_2' = \lambda_3' = 1 + (\varepsilon - 1) \, . \, V^{\frac{1}{S}}$$

Where ε, w and S are algorithm parameters.

Using this tensor definition, a diffusion equation is formulated as follows

$$L_t = \nabla \, . \, (D \nabla L) \tag{8}$$

Vascular structures will be enhanced by evolving the image according to the above diffusion equation.

2 VED Filters Design in ITK

VED algorithm implementation consists of two main parts. The first part involves implementation of filters to compute smoothed Frangi's vesselness measure for a given image. The second part involves implementation of the anisotropic diffusion filters for vessel enhancement.

2.1 Smoothed Frangi's Vesselness Measure Computing Filters

Two filters were implemented to evaluate Frangi's vesselness measure in a multiscale framework. The first filter, HessianSmoothed3DToVesselnessMeasureImageFilter computes vesselness measure at a single scale. The second filter, MultiScaleHessianSmoothed3DToVesselnessMeasureImageFilter computes the maximum vesselness response from a range of scales. To use the filter for a multiscale analysis, a user has to specify minimum, maximum sigma values and number of scales. The filter computes and selects the maximum vesselness response at exponentially distributed number of scales between the specified minimum and maximum sigma values.

These filters are derived from the `itk::ImageToImageFilter`. For each pixel, Hessian matrix is computed using the `itk::HessianRecursiveGaussianImageFilter` filter. This filter computes Hessian matrix by convolving the input image with second and cross derivatives of the Gaussian function.

The multiscale filter has the following main public methods.

1 To set minimum sigma value

```
SetSigmaMin  ( double );
```

2 To set maximum sigma value

```
SetSigmaMax ( double );
```

3 To set the number of sigma steps

```
SetNumberOfSigmaSteps( int );
```

Appendix A shows an example on how to use this filter.

2.2 Anisotropic Diffusion Filters for Vessel Enhancement

The implementation of the anisotropic filters follow the finite difference solver framework. The implementation consists of two components: solver object (itkAnisotropicDiffusionVesselEnhancementImageFilter) and a specialized finite difference function object (itkAnisotropicDiffusionVesselEnhancementFunction). The solver object establishes the infrastructure for accepting input image and producing output image. In addition, the solver object uses the specialized finite difference function object to perform the diffusion equation computation at each pixel for several iterations. The solver object and the the function objects are derived from `itk::FiniteDifferenceImageFilter` and `itk::FiniteDifferenceFunction` respectively. Figure 1 shows the flowchart for the vessel enhancement diffusion algorithm. An example program that demonstrates how to use this filter is provided in appendix B.

3 Experiments and results

Experiments were conducted to test the effectiveness of the VED algorithm in enhancing vessels in a whole lung CT scan. Figure 2 shows lung CT scan used to test the algorithm.

The testing dataset is distributed as part of the source code submission to the Insight Journal. Readers are encouraged to build the source code and run the algorithm on the testing dataset.

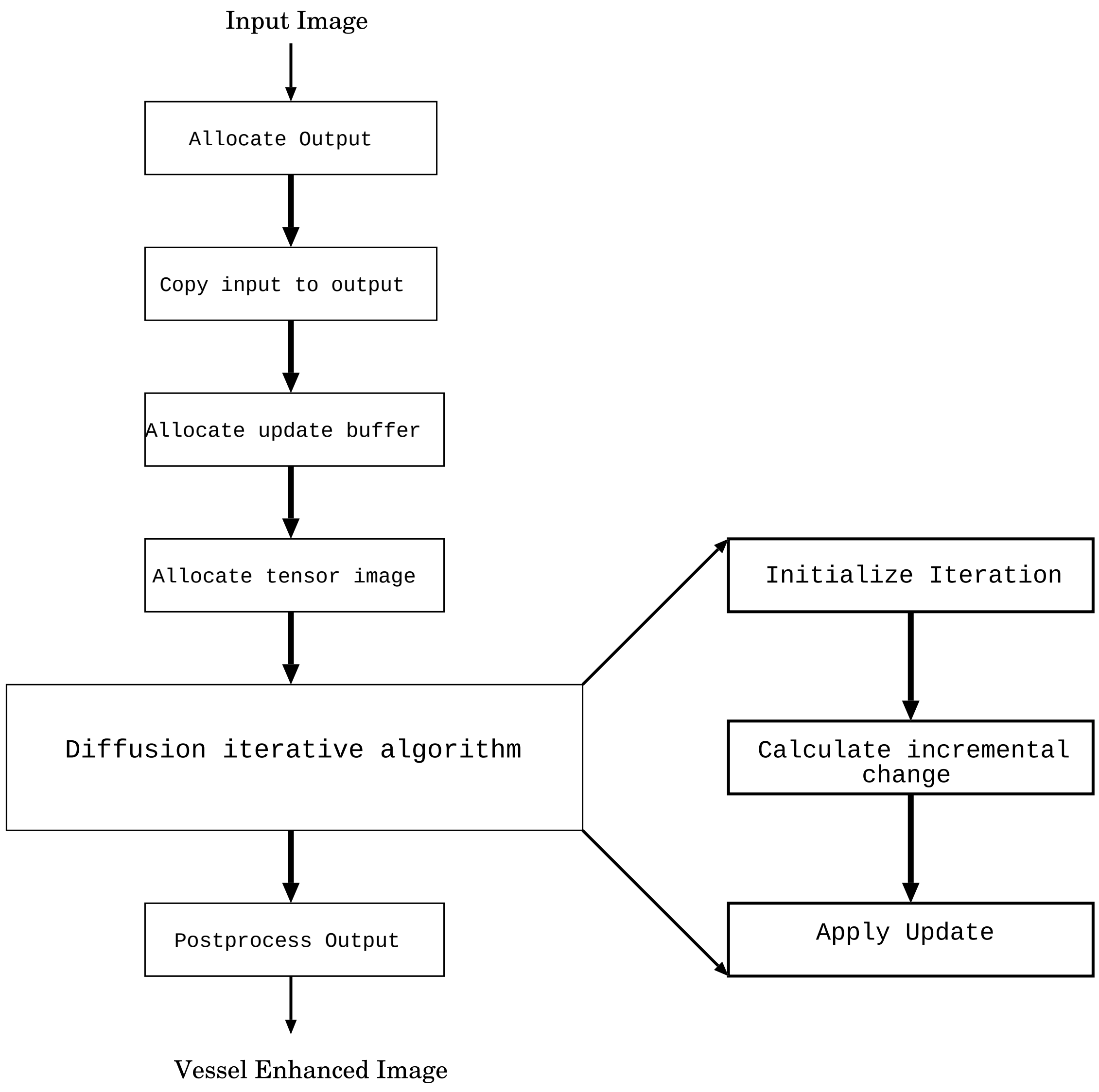

Figure 1: VED ITK filter flowchart

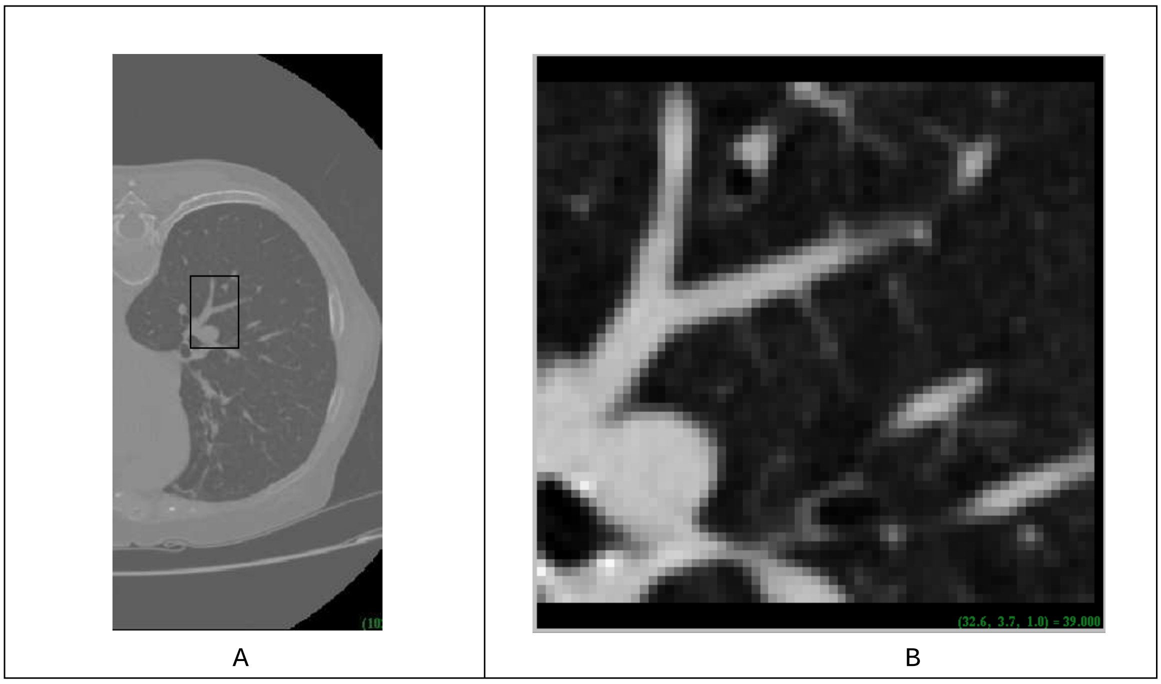

Figure 2: Testing dataset: A) Lung CT scan B) Cropped ROI

In the first experiment, Frangi's vesselness filters were run on the testing dataset to evaluate the performance of the filter on a single scale and a range of scales. The results are shown in Figure 3. With a sigma value of 0.5, small size vessels are enhanced as shown in Figure 3B. If the scale is increased to a sigma value of 4.0, large size vessels are enhanced (Figure 3C). To enhance vessels with varying size, the image was run through Frangi's multiscale filter with a sigma range of [0.5 4.0]. The result is shown in Figure 3D. As clearly evident from the result, multi scale analysis is useful in enhancing various size vessels available in the scan. This is very useful for a complete vascular structure enhancement and reconstruction. Although the vascular structures are generally enhanced, the results show that the vesselness measure is highly sensitive to noise pixels.

In the second experiment, the performance of the VED filters was evaluated. VED filter was run on the same Lung CT scan. The results are shown in Figure 4. The VED filter was operated in a multiscale framework with computations at ten sigma values exponentially distributed in sigma value range [0.5 4.0]. The diffusion process was analyzed for different number of iterations. The results for 10, 25 and 50 number of iterations is shown in Figure 4B, 4C and 4D respectively. Overall, the results show that VED filters enhance the vascular structures better than Frangi's vesselness measure. In addition, VED filters are less affected by noise pixels. Furthermore, with the increase in the number of iterations, an increased smoothing effect was observed.

4 Conclusions

In this paper, we have described VED filters implemented using ITK. The implementation is based on the algorithm by Manniesing et al [4]. VED algorithm uses vesselness measure based on Hessian Eigen system to guide a diffusion process. Experiments were conducted to evaluate the performance of the filters in enhancing vessels in Lung CT scans. Visualizing the enhanced images showed that VED algorithm performs better than Frangi's vesseleness measure.

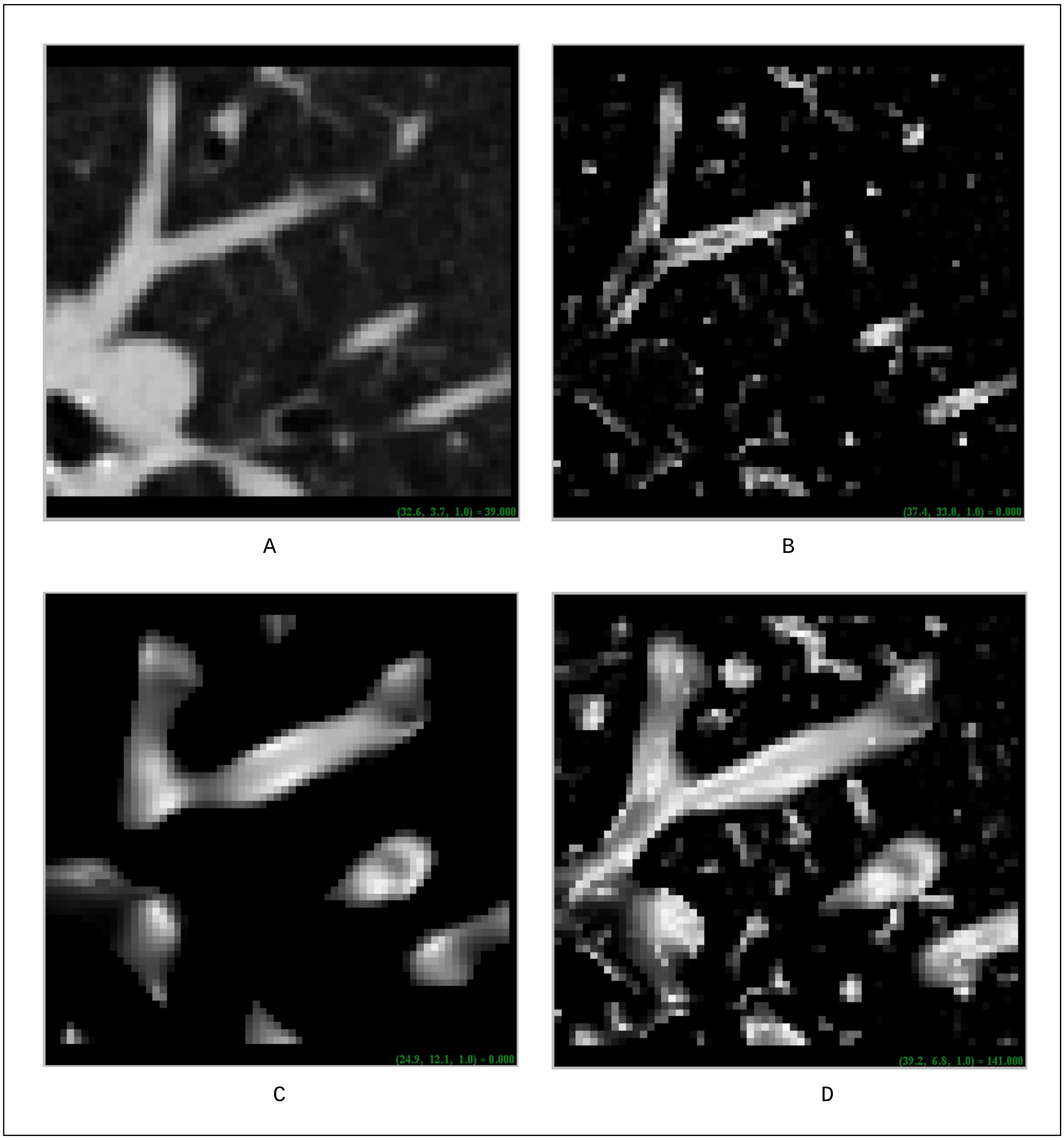

Figure 3: Frangi vesselness results: A) Original image B) sigma=0.5 C) sigma=4.0 D) Multiscale analysis sigma=[0.5 4.0]

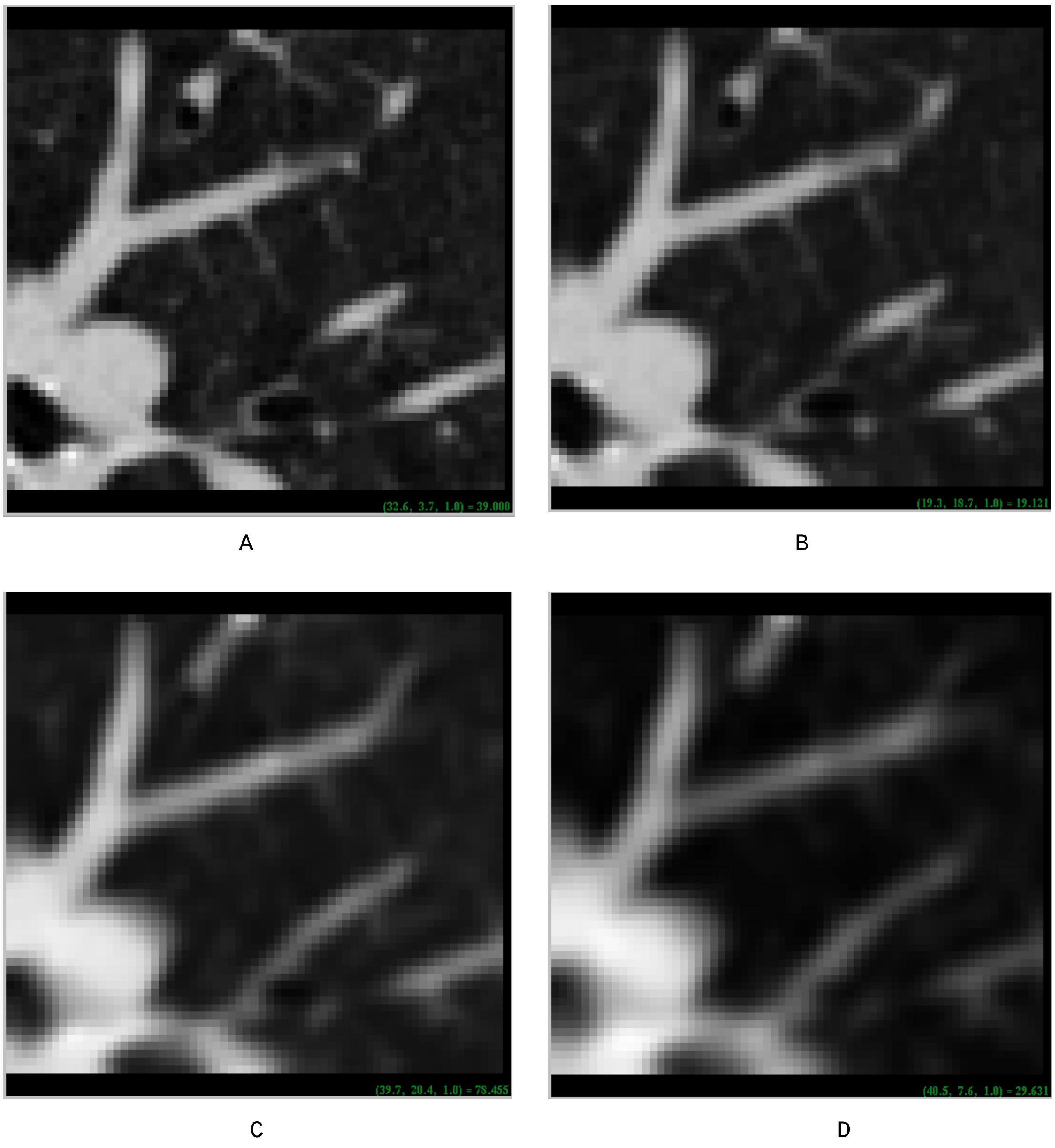

Figure 4: VED algorithm results: A) Original image B) Number of iterations = 10 C) Number of iterations = 25 D) Number of iterations = 50

5 Acknowledgment

This work was supported by NIH R01 EB000219 (PI: Bullitt, UNC) and NIH R01 HL69808 (PI: Bullitt, UNC).

A Example 1 - Computing vesselness measure

```
#include "itkMultiScaleHessianSmoothed3DToVesselnessMeasureImageFilter.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"

#include "itkRescaleIntensityImageFilter.h"

int main(int argc, char* argv [] )
{
  if ( argc < 3 )
    {
    std::cerr << "Missing Parameters: "
              << argv[0]
              << " Input_Image"
              << " Vessel_Enhanced_Output_Image"
              << [SigmaMin SigmaMax NumberOfScales]" << std::endl;
    return EXIT_FAILURE;
    }

  // Define the dimension of the images
  const unsigned int Dimension = 3;
  typedef short      InputPixelType;
  typedef double     OutputVesselnessPixelType;

  // Declare the types of the images
  typedef itk::Image< InputPixelType, Dimension>            InputImageType;

  typedef itk::Image< OutputVesselnessPixelType, Dimension> VesselnessOutputImageType;

  typedef itk::ImageFileReader< InputImageType  >      ImageReaderType;

  ImageReaderType::Pointer   reader = ImageReaderType::New();
  reader->SetFileName ( argv[1] );

  std::cout << "Reading input image : " << argv[1] << std::endl;
  try
    {
    reader->Update();
    }
  catch ( itk::ExceptionObject &err )
```

```
    {
    std::cerr << "Exception thrown: " << err << std::endl;
    return EXIT_FAILURE;
    }


  // Declare the type of multiscale vesselness filter
  typedef itk::MultiScaleHessianSmoothed3DToVesselnessMeasureImageFilter<
                                          InputImageType,
                                          VesselnessOutputImageType>
                                          MultiScaleVesselnessFilterType;

  // Create a vesselness Filter
  MultiScaleVesselnessFilterType::Pointer MultiScaleVesselnessFilter =
                                    MultiScaleVesselnessFilterType::New();

  MultiScaleVesselnessFilter->SetInput( reader->GetOutput() );

  if ( argc >= 4 )
    {
    MultiScaleVesselnessFilter->SetSigmaMin( atof(argv[3])  );
    }

  if ( argc >= 5 )
    {
    MultiScaleVesselnessFilter->SetSigmaMax( atof(argv[4]) );
    }

  if ( argc >= 6 )
    {
    MultiScaleVesselnessFilter->SetNumberOfSigmaSteps( atoi(argv[5]) );
    }

  try
    {
    MultiScaleVesselnessFilter->Update();
    }
  catch( itk::ExceptionObject & err )
    {
    std::cerr << "Exception caught: " << err << std::endl;
    return EXIT_FAILURE;
    }

  std::cout << "Writing out the enhanced image to " <<  argv[2] << std::endl;

  //Rescale the output of the vesslness image
  typedef itk::Image<unsigned char, 3>              OutputImageType;
  typedef itk::RescaleIntensityImageFilter< VesselnessOutputImageType,
                                            OutputImageType>
                                            RescaleFilterType;

  RescaleFilterType::Pointer rescale = RescaleFilterType::New();
  rescale->SetInput( MultiScaleVesselnessFilter->GetOutput() );
  rescale->SetOutputMinimum(   0 );
  rescale->SetOutputMaximum( 255 );
  rescale->Update();

  typedef itk::ImageFileWriter< OutputImageType  >      ImageWriterType;
  ImageWriterType::Pointer writer = ImageWriterType::New();

  writer->SetFileName( argv[2] );
  writer->SetInput ( rescale->GetOutput() );

  try
    {
    writer->Update();
    }
  catch( itk::ExceptionObject & err )
    {
    std::cerr << "Exception caught: " << err << std::endl;
    return EXIT_FAILURE;
    }

  return EXIT_SUCCESS;

}
```

B Example 2 - Vessel enhancement using VED filter

```
#include "itkAnisotropicDiffusionVesselEnhancementImageFilter.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"

int main(int argc, char* argv [] )
{
  if ( argc < 3 )
    {
    std::cerr << "Missing Parameters: "
              << argv[0]
              << " Input_Image"
              << " Vessel_Enhanced_Output_Image [SigmaMin SigmaMax NumberOfScales NumberOfI
    return EXIT_FAILURE;
    }
```

```
// Define the dimension of the images
const unsigned int Dimension = 3;
typedef double      InputPixelType;
typedef double      OutputPixelType;

// Declare the types of the images
typedef itk::Image< InputPixelType, Dimension>           InputImageType;
typedef itk::Image< InputPixelType, Dimension>           OutputImageType;

typedef itk::ImageFileReader< InputImageType  >      ImageReaderType;

ImageReaderType::Pointer   reader = ImageReaderType::New();
reader->SetFileName ( argv[1] );

std::cout << "Reading input image : " << argv[1] << std::endl;
try
  {
  reader->Update();
  }
catch ( itk::ExceptionObject &err )
  {
  std::cerr << "Exception thrown: " << err << std::endl;
  return EXIT_FAILURE;
  }

// Declare the anisotropic diffusion vesselness filter
typedef itk::AnisotropicDiffusionVesselEnhancementImageFilter< InputImageType,
                                          OutputImageType>  VesselnessFilterType;

// Create a vesselness Filter
VesselnessFilterType::Pointer VesselnessFilter =
                                    VesselnessFilterType::New();

VesselnessFilter->SetInput( reader->GetOutput() );

if ( argc >= 4 )
  {
  VesselnessFilter->SetSigmaMin( atof(argv[3])  );
  }

if ( argc >= 5 )
  {
  VesselnessFilter->SetSigmaMax( atof(argv[4]) );
  }

if ( argc >= 6 )
  {
    VesselnessFilter->SetNumberOfSigmaSteps( atoi(argv[5]) );
    }

  if ( argc >= 7 )
    {
    VesselnessFilter->SetNumberOfIterations( atoi(argv[6]) );
    }

  VesselnessFilter->SetSensitivity( 5.0 );
  VesselnessFilter->SetWStrength( 25.0 );
  VesselnessFilter->SetEpsilon( 10e-2 );

  std::cout << "Enhancing vessels.........: " << argv[1] << std::endl;

  try
    {
    VesselnessFilter->Update();
    }
  catch( itk::ExceptionObject & err )
    {
    std::cerr << "Exception caught: " << err << std::endl;
    return EXIT_FAILURE;
    }

  std::cout << "Writing out the enhanced image to " <<  argv[2] << std::endl;

  typedef itk::ImageFileWriter< OutputImageType  >      ImageWriterType;
  ImageWriterType::Pointer writer = ImageWriterType::New();

  writer->SetFileName( argv[2] );
  writer->SetInput ( VesselnessFilter->GetOutput() );

  try
    {
    writer->Update();
    }
  catch( itk::ExceptionObject & err )
    {
    std::cerr << "Exception caught: " << err << std::endl;
    return EXIT_FAILURE;
    }

  return EXIT_SUCCESS;

}
    std::cerr << "Exception caught: " << err << std::endl;
    return EXIT_FAILURE;
    }
  return EXIT_SUCCESS;

}
```

References

[1] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. In W. M. Wells, A. Colchester, and S. Delp, editors, *MICCAI'98 Medical Image Computing and Computer-Assisted Intervention*, Lecture Notes in Computer Science, pages 130–137. Springer Verlag, 1998. (document), 1

[2] L. Ibanez and W. Schroeder. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-10-6, http://www.itk.org/ItkSoftwareGuide.pdf, 2003. (document)

[3] C. Kirbas and F. Quek. A review of vessel extraction techniques and algorithms. *ACM Computing Surveys*, 36(2):81–121, 2004. 1

[4] R. Mannieshing, M.A. Viergever, and W. J . Niessen. Vessel enhancing diffusion: A scale space representation of vessel structures. *Medical Image Analysis*, 2006. (document), 1, 1.1, 4