Generalizing vesselness with respect to dimensionality and shape

Release 1.00

Luca Antiga[1]

August 3, 2007

[1]Medical Imaging Unit
Mario Negri Institute, Bergamo, Italy
email: antiga at marionegri.it

Abstract

In this paper we present a generalization of Frangi's vesselness measure [2] for the enhacement of M-dimensional shapes in N-dimensional images, together with the implementation of the corresponding Insight Toolkit (ITK) filter [3]. Inspired by the implementation of a multiscale vesselness measure recently presented on the Insight Journal [1], we also propose a unified framework for the evaluation of generic multiscale Hessian-based measures. The manuscript is accompanied by source code and examples.

Contents

1 Introduction

Vessel enhancement techniques are employed for the identification and quantification of vascular structures in 2D and 3D angiographic images. They provide enhancement of visual appearance of vascular structures for clinical evaluation and they have been successfully used for centerline extraction and for pre-processing, initialization and feature definition in segmentation procedures.

The majority of vessel enhancement filters are based on the analysis of eigenvalues of the Hessian matrix of image intensity. The mutual magnitude of eigenvalues is indicative of the shape of the underlying object. Isotropic structures are associated to eigenvalues having a similar non-zero magnitude, while vessels present one negligible and two similar non-zero eigenvalues.

Recalling Frangi's formulation [2] and indicating the eigenvalues of a Hessian matrix as λ_1, λ_2 and λ_3, with $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$, vesselness is defined in 3D as

$$V_\sigma(\lambda) = \begin{cases} (1 - e^{-\frac{R_A^2}{2\alpha^2}}) \cdot e^{-\frac{R_B^2}{2\beta^2}} \cdot (1 - e^{-\frac{S^2}{2\gamma^2}}) & \text{if } \lambda_2 < 0 \text{ and } \lambda_3 < 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where

$$R_A = \frac{|\lambda_2|}{|\lambda_3|} \tag{2}$$

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

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

and α, β and γ are user-defined parameters. The constraint on the sign of eigenvalues assumes that vessels to enhance are bright on a dark background, and has to be reversed in opposite case. In Equation (1), the term containing R_A yields a large value (closer to 1) when the ratio R_A is large, i.e. when the magnitude of the two largest eigenvalues is similar. In contrast, the term containing R_B yields a value closer to 1 when the smallest eigenvalue is much smaller than the other two. Considering the second-order ellipsoid defined by the eigenvalues, we are requiring that the ellipsoid has a high aspect ratio (high anisotropy) along the λ_1 axis and a low aspect ratio (low anisotropy) in the $\lambda_2\lambda_3$ plane. In fact, R_A and R_B can be rewritten as

$$R_A = \frac{|\lambda_2\lambda_3|}{|\lambda_3|^2} \propto \frac{\text{Largest cross sect area}}{\text{Largest cross sect largest axis}} \tag{5}$$

$$R_B = \frac{|\lambda_1\lambda_2\lambda_3|}{|\lambda_2\lambda_3|^{\frac{3}{2}}} \propto \frac{\text{Volume}}{\text{Largest cross sect area}} \tag{6}$$

The term containing S, the Frobenius norm of the Hessian matrix, or second-order structureness, serves to control the sensitivity of V_σ to background noise.

The σ footer in V_σ indicates that vesselness is computed on a smoothed version of the image and is therefore representative of the variations of image intensity at the spatial scale σ. Vesselness is evaluated at a range of spatial scales, and the maximum response is selected at every voxel

$$V(\lambda) = \max_{\sigma \in [\sigma_{\min}, \sigma_{\max}]} V_\sigma(\lambda) \tag{7}$$

Equation (1) can be adapted to extract, for example, isotropic blob-like structures by replacing $e^{-\frac{R_B^2}{2\beta^2}}$ with $(1 - e^{-\frac{R_B^2}{2\beta^2}})$, therefore requesting that all three eigenvalues have a similar magnitude. This is equivalent to requiring that the second-order ellipsoid has a low aspect ratio for all three axes. Indeed, since eigenvalues are sorted, the condition on R_A is redundant, since it is implied by the condition on R_B.

2 Generalization to M-dimensional objects in N-dimensions

From the latter consideration and from Equation (5) we can induce the expression of Equation (1) for the enhancement of M-dimensional structures in N-dimensional images, with $M < N$ ($M = 0$ for blobs, $M = 1$ for vessels, $M = 2$ for plates, $M = 3$ for hyper-plates, etc.). In order to do this, we have to require that the Mth moment of the Nth order ellipsoid is small with respect to higher moments, while the $M+1$th moment of the Nth order ellipsoid is large with respect to higher moments, in the same way as in Equation (5). Since eigenvalues are sorted in magnitude, these two conditions imply analogous conditions on lower and higher moments, respectively.

Let's indicate as $\lambda_1 \dots \lambda_N$ the N eigenvalues of the $N \times N$ Hessian matrix, such that $\lambda_1 \leq \dots \leq \lambda_N$. For the enhancement of M-dimensional structures in a N-dimensional image we define the following ratios

$$R_A = \frac{\prod_{i=M+1}^{N} |\lambda_i|}{\prod_{i=M+2}^{N} |\lambda_i|^{\frac{N-M}{N-M-1}}} \tag{8}$$

$$R_B = \frac{\prod_{i=M}^{N} |\lambda_i|}{\prod_{i=M+1}^{N} |\lambda_i|^{\frac{N-M+1}{N-M}}} \tag{9}$$

which have been expressed as ratio of moments of the Nth order ellipsoid, for ease of comparison to Equation (5). Simple manipulation of Equation (8) leads to the following expressions for R_A and R_B

$$R_A = \frac{|\lambda_{M+1}|}{\prod_{i=M+2}^{N} |\lambda_i|^{\frac{1}{N-M-1}}} \tag{10}$$

$$R_B = \frac{|\lambda_M|}{\prod_{i=M+1}^{N} |\lambda_i|^{\frac{1}{N-M}}} \tag{11}$$

In the above expressions, two special cases can be identified. The first is the case $M = N-1$ (enhancement of hyperplanes), for which the denominator of R_A is not defined. In this case we can allow $R_A \to \infty$ for $M = N-1$. The second is the case $M = 0$ (enhancement of blobs), for which the numerator of R_B is not defined, since λ_0 is not defined. Setting it to zero yields $R_B = 0$ for $M = 0$.

The Frobenius norm of the Hessian matrix follows the usual definition

$$S = \sqrt{\sum_{j=1}^{N} \lambda_j^2} \tag{12}$$

Last, analogously to Equation (1), we associate R_A, R_B and S to the weights α, β and γ.

We finally define the *objectness* measure as

$$O(\lambda)_\sigma = \begin{cases} (1-e^{-\frac{R_A^2}{2\alpha^2}}) \cdot e^{-\frac{R_B^2}{2\beta^2}} \cdot (1-e^{-\frac{S^2}{2\gamma^2}}) & \text{if } \lambda_j < 0 \text{ for } M < j \leq N \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where the constraint on the sign of eigenvalues has been set in the assumption that structures are bright on a dark background, and has to be reversed in case of dark structures on a bright background. As to the special cases identified above, for the case $M = N-1$ the first term of the product (involving R_A) is 1, while for the case $M = 0$ the second term of the product (involving R_B) is 1. This is consistent with the requirements of penalizing all eigenvalues with respect to the largest for the $M = N-1$ case, and not penalizing any eigenvalue for the $M = 0$ case. As expected, Equation (13) reduces to Equation (1) for $N = 3$ and $M = 1$.

Equivalently to vesselness, Equation (13) is evaluated at a range of spatial scales, and the maximum response is selected

$$O(\lambda) = \max_{\sigma \in [\sigma_{\min}, \sigma_{\max}]} O_\sigma(\lambda) \tag{14}$$

3 ITK implementation

The described technique has been implemented in the Insight Toolkit following the design adopted by Enquobahrie et al for multiscale vesselness [1].

The computation of the objectness measure for a given scale is performed in `itk::HessianToObjectnessMeasureImageFilter`, which computes the measure from a input Hessian matrix image. The implementation closely follows Equation (13). N is derived from the dimensionality of the input image, while M is set through the `SetObjectDimensionality` metric. The metric is optionally scaled with the magnitude of the largest eigenvalue (throught the `SetScaleObjectnessMeasure` method), while the case of a bright object over a dark background or vice-versa is specified through the `SetBrightObject` method.

Multiscale computation in `itk::MultiScaleHessianBasedMeasureImageFilter`. This class builds on `itk::MultiScaleHessianSmoothed3DToVesselnessMeasure` presented in [1], adding some general functionality to it. As in [1] the filter takes a scalar image in input and internally computes the Hessian using `itk::HessianRecursiveGaussianImageFIlter` for each of the scales defined by the `SetSigmaMin` `SetSigmaMax` and `SetNumberOfSigmaStep` methods. It then passes the Hessian matrix image to the filter that implements the Hessian-based measure computation and fills the output image with the best response from all the scales.

`itk::MultiScaleHessianBasedMeasureImageFilter` is templated over the Hessian-based measure filter type, so it can be employed with the currently available Hessian-based measure filters namely `itk::HessianToObjectnessMeasureImageFilter`, `itk::Hessian3DToVesselnessMeasureImageFilter` and `itk::HessianSmoothed3DToVesselnessMeasureImageFilter`.

In addition, `itk::MultiScaleHessianBasedMeasureImageFilter` produces two outputs. The first is the image containing the actual measure, returned by the `GetOutput` method. The second is the image containing the value of the scale at which the pixels gave the best response, and it is returned by the

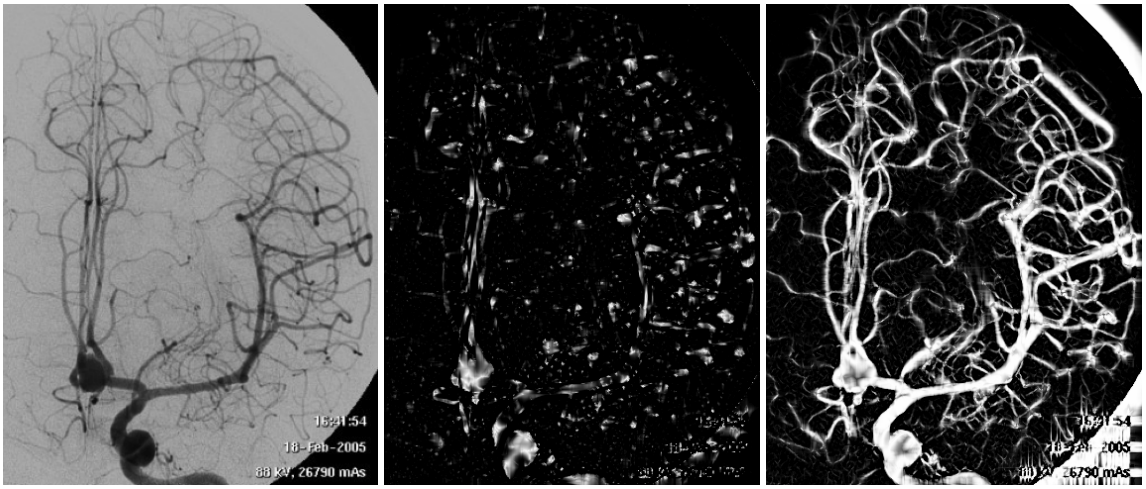

Figure 1: Left: DSA dataset depicting cerebral circulation and two cerebral aneurysms. Center: Multiscale 2D blobness (objectness with $N = 2$ and $M = 0$), $\sigma \in [1,10]$, 10 steps. Right: Multiscale 2D vesselness (objectness with $N = 2$ and $M = 1$), $\sigma \in [1,10]$, 10 steps, $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 5.0$.

`GetScalesOutput` method. This second image can be employed to derive an estimate of the size of the structures being enhanced.

4 Experiments and results

In Figure 1, the multiscale objectness filter has been applied to a 2D digital subtraction angiography (DSA) image of the cerebral circulation depicting two aneurysms. Both blobness and vesselness are computed for $\sigma \in [1,10]$ (pixel spacing is 1).

In Figure 2, blobness ($N = 2$, $M = 0$) with $\sigma \in [5,10]$ is shown, together with the second filter output, depicting the best response spatial scales. Figure 2 shows the same information relative to vesselness ($N = 2$, $M = 0$).

For direct comparison with [1], the cropped lung dataset has been enhanced using the same parameters ($N = 3$, $M = 1$, $\sigma \in [0.5,4]$ in 10 steps, $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 0.5$), as shown in Figure 4, upper. In the lower left image, the same measure has been computed by scaling the measure with the largest eigenvalue. In the lower right, the result of multiscale Sato's vesselness 3D computed using the new generalized multiscale Hessian-based analysis filter is shown (as detailed in Example 3).

5 Conclusions and future directions

In this paper, a general definition for a Hessian-based object enhancement measure was presented, together with the implementation of a generalized multiscale filter for Hessian based measures.

Future work will involve the generalization of the smooth metric presented in [1]. The smoothing term will have to be expressed with an expression of the type $e^{-\frac{2c^2}{R_C}}$, with $R_C = \prod_{i=M+1}^{N} |\lambda_i|^{i-1}$. This will provide a smoother objectness measure and the possibility of employing it for anisotropic diffusion.

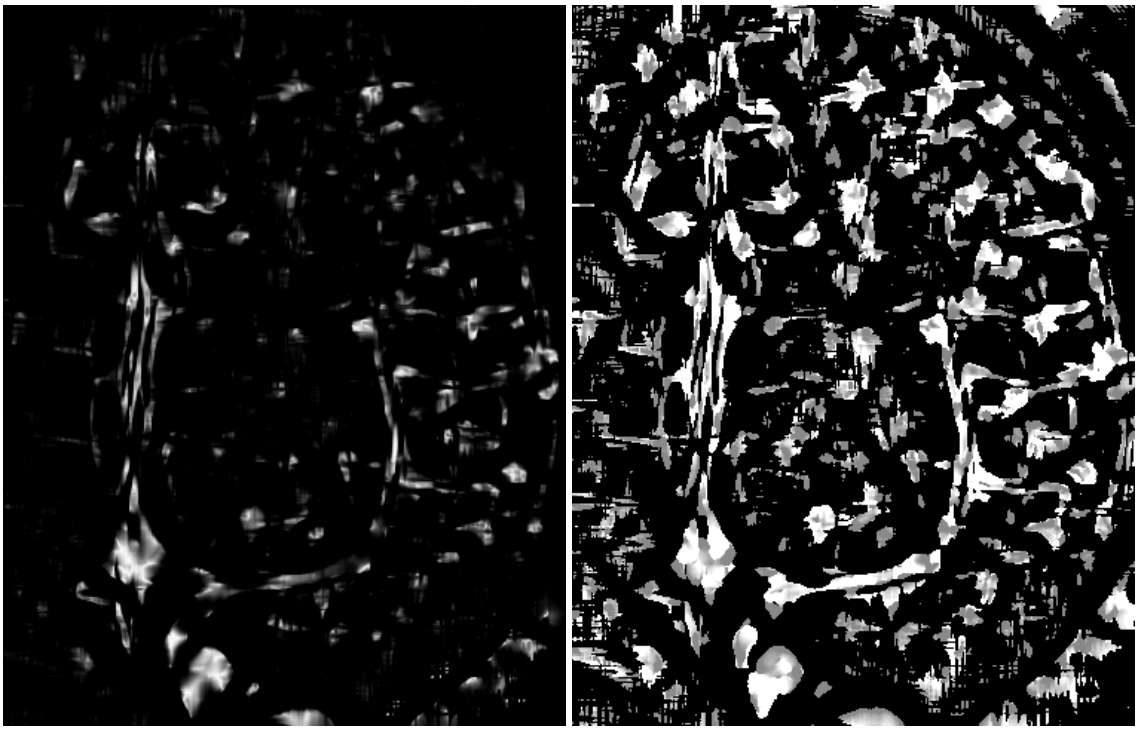

Figure 2: Left: Multiscale 2D blobness (objectness with $N = 2$ and $M = 0$), $\sigma \in [5, 10]$, 10 steps, $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 5.0$. Aneurysms, bifurcations and vessel overlaps are enhanced. Right: best response scales.

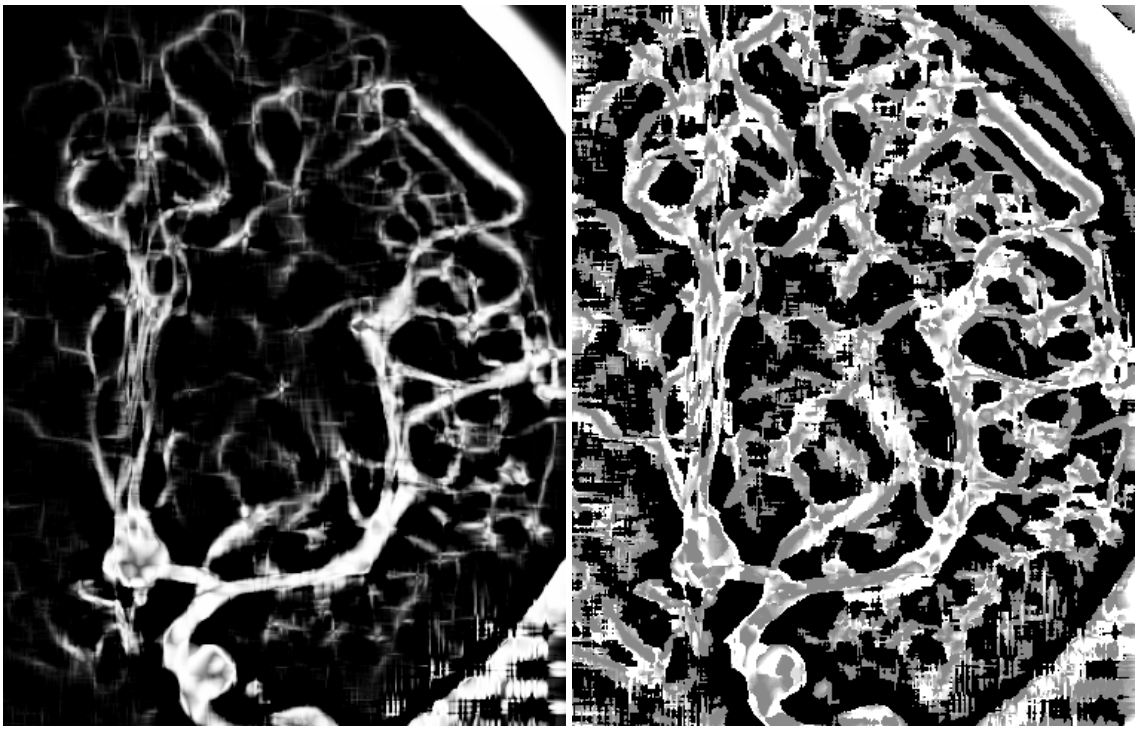

Figure 3: Left: Multiscale 2D vesselness (objectness with $N = 2$ and $M = 1$), $\sigma \in [5, 10]$, 10 steps, $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 5.0$. Right: best response scales.

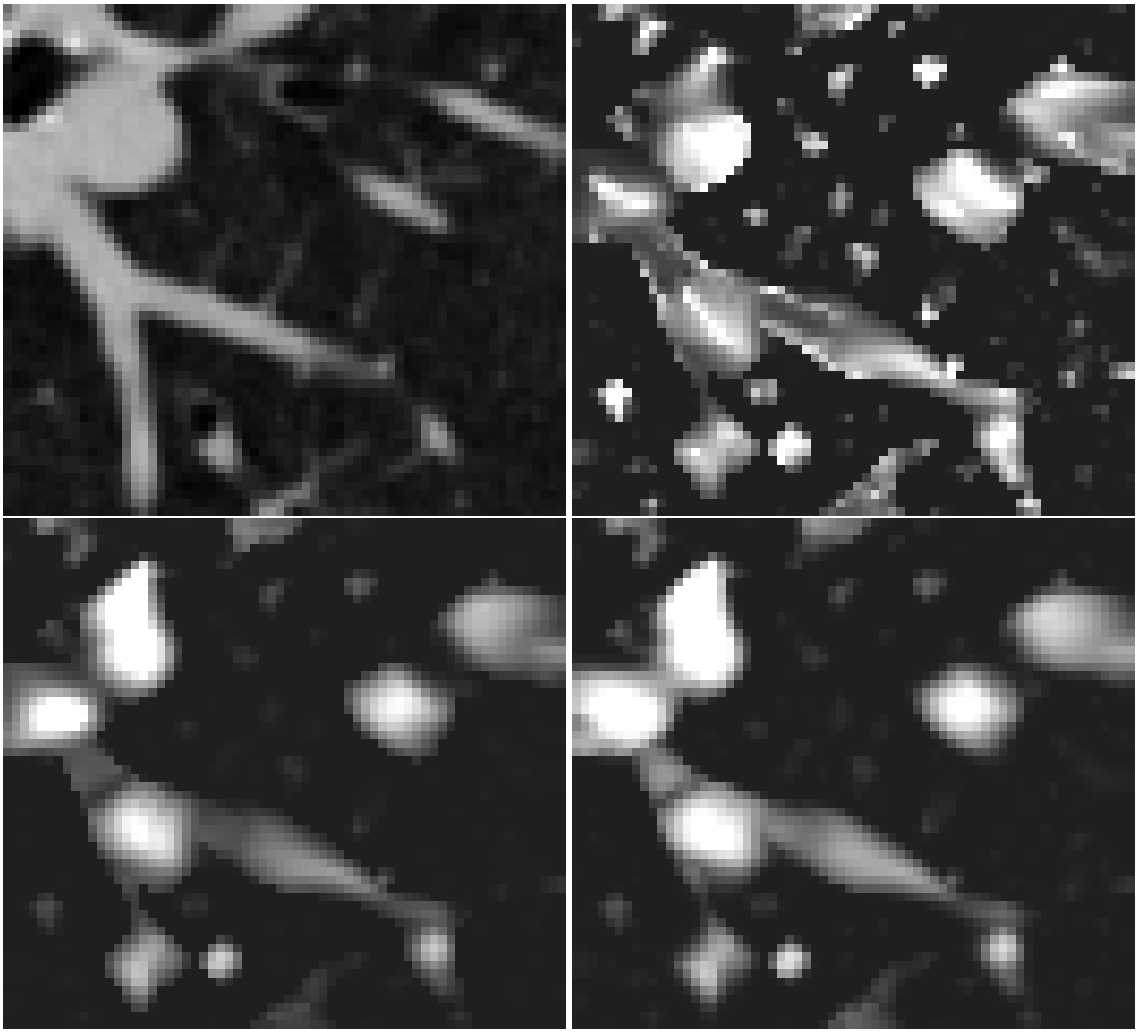

Figure 4: Upper left: Cropped lung dataset employed in [1]. Upper right: multiscale Frangi's vesselness computed with the objectness measure, $N = 3$, $M = 1$, $\sigma \in [0.5, 4]$, 10 steps, $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 5.0$. Lower left: multiscale Frangi's vesselness normalized with the magnitude of the largest eigenvalue computed with the objectness measure (same parameters). Lower right: multiscale Sato's vesselness computed using ITK's 3D vesselness, $\sigma \in [0.5, 4]$, 10 steps, α_1=0.5, $\alpha_2 = 0.5$, within the new multiscale Hessian based measure filter.

Additional work will also include broader testing of the algorithms presented in this paper, especially on 4D (3D+time) datasets, for the evaluation of enhancement performance and optimization of user-defined parameters.

A Example 1 - Multiscale 3D Objectness (M=1)

```
#include "itkHessianToObjectnessMeasureImageFilter.h"
#include "itkMultiScaleHessianBasedMeasureImageFilter.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkRescaleIntensityImageFilter.h"
#include "itkImage.h"

int main( int argc, char *argv[] )
{
  if ( argc < 4 )
    {
    std::cerr << "Missing Parameters: "
              << argv[0]
              << " Input_Image"
              << " Enhanced_Output_Image"
              << " Scales_Output_Image"
              << std::endl;
    return EXIT_FAILURE;
    }

  // Define the dimension of the images
  const unsigned char Dim = 3;

  typedef float PixelType;

  // Declare the types of the images
  typedef itk::Image<PixelType,Dim> ImageType;

  typedef itk::ImageFileReader<ImageType> FileReaderType;
  typedef itk::ImageFileWriter<ImageType> FileWriterType;

  typedef itk::RescaleIntensityImageFilter<ImageType> RescaleFilterType;

  // Declare the type of enhancement filter
  typedef itk::HessianToObjectnessMeasureImageFilter<double,Dim> ObjectnessFilterType;

  // Declare the type of multiscale enhancement filter
  typedef itk::MultiScaleHessianBasedMeasureImageFilter<ImageType,ObjectnessFilterType>
   MultiScaleEnhancementFilterType;

  // Read the image
  FileReaderType::Pointer imageReader = FileReaderType::New();
  imageReader->SetFileName(argv[1]);
  try
  {
    imageReader->Update();
```

```
}
catch (itk::ExceptionObject &ex)
{
  std::cout << ex << std::endl;
  return EXIT_FAILURE;
}

// Instantiate the multiscale filter and set the input image
MultiScaleEnhancementFilterType::Pointer multiScaleEnhancementFilter =
 MultiScaleEnhancementFilterType::New();
multiScaleEnhancementFilter->SetInput(imageReader->GetOutput());
multiScaleEnhancementFilter->SetSigmaMin(0.5);
multiScaleEnhancementFilter->SetSigmaMax(4.0);
multiScaleEnhancementFilter->SetNumberOfSigmaSteps(10);

// Get the objectness filter and set the parameters
ObjectnessFilterType* objectnessFilter =
 multiScaleEnhancementFilter->GetHessianToMeasureFilter();
objectnessFilter->SetScaleObjectnessMeasure(false);
objectnessFilter->SetBrightObject(true);
objectnessFilter->SetAlpha(0.5);
objectnessFilter->SetBeta(0.5);
objectnessFilter->SetGamma(5.0);
objectnessFilter->SetObjectDimension(1);

// The above is equivalent to vesselness

// Now run the multiscale filter
try
{
  multiScaleEnhancementFilter->Update();
}
catch (itk::ExceptionObject &e)
{
  std::cerr << e << std::endl;
}

// Write the enhanced image
FileWriterType::Pointer writer = FileWriterType::New();
writer->SetFileName(argv[2]);
writer->SetInput(multiScaleEnhancementFilter->GetOutput());
try
{
  writer->Update();
}
catch (itk::ExceptionObject &e)
{
  std::cerr << e << std::endl;
}

// Write the image containing the best response scales
FileWriterType::Pointer writer2 = FileWriterType::New();
writer2->SetFileName(argv[3]);
writer2->SetInput(multiScaleEnhancementFilter->GetScalesOutput());
```

```
  try
  {
    writer2->Update();
  }
  catch (itk::ExceptionObject &e)
  {
    std::cerr << e << std::endl;
  }
}
```

B Example 2 - Multiscale 2D Objectness (M=1)

```
#include "itkHessianToObjectnessMeasureImageFilter.h"
#include "itkMultiScaleHessianBasedMeasureImageFilter.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkRescaleIntensityImageFilter.h"
#include "itkImage.h"

int main( int argc, char *argv[] )
{
  if ( argc < 5 )
    {
    std::cerr << "Missing Parameters: "
              << argv[0]
              << " Input_Image"
              << " Enhanced_Output_Image"
              << " Scales_Output_Image"
              << " [SigmaMin SigmaMax NumberOfScales ObjectDimension]" << std::endl;
    return EXIT_FAILURE;
    }

  // Define the dimension of the images
  const unsigned char Dim = 2;

  typedef float PixelType;

  // Declare the types of the images
  typedef itk::Image<PixelType,Dim> ImageType;

  typedef itk::ImageFileReader<ImageType> FileReaderType;
  typedef itk::ImageFileWriter<ImageType> FileWriterType;

  typedef itk::RescaleIntensityImageFilter<ImageType> RescaleFilterType;

  // Declare the type of enhancement filter
  typedef itk::HessianToObjectnessMeasureImageFilter<double,Dim> ObjectnessFilterType;

  // Declare the type of multiscale enhancement filter
  typedef itk::MultiScaleHessianBasedMeasureImageFilter<ImageType,ObjectnessFilterType>
   MultiScaleEnhancementFilterType;
```

```
FileReaderType::Pointer imageReader = FileReaderType::New();
imageReader->SetFileName(argv[1]);
try
{
  imageReader->Update();
}
catch (itk::ExceptionObject &ex)
{
  std::cout << ex << std::endl;
  return EXIT_FAILURE;
}

MultiScaleEnhancementFilterType::Pointer multiScaleEnhancementFilter =
 MultiScaleEnhancementFilterType::New();
multiScaleEnhancementFilter->SetInput(imageReader->GetOutput());

ObjectnessFilterType* objectnessFilter = multiScaleEnhancementFilter->GetHessianToMeasureFilter();
objectnessFilter->SetScaleObjectnessMeasure(false);
objectnessFilter->SetBrightObject(false);
objectnessFilter->SetGamma(5.0);

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

if ( argc >= 6 )
  {
  multiScaleEnhancementFilter->SetSigmaMax( atof(argv[5]) );
  }

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

if ( argc >= 8 )
  {
  objectnessFilter->SetObjectDimension( atoi(argv[7]) );
  }

try
{
  multiScaleEnhancementFilter->Update();
}
catch (itk::ExceptionObject &e)
{
  std::cerr << e << std::endl;
}

RescaleFilterType::Pointer rescale = RescaleFilterType::New();
rescale->SetInput(multiScaleEnhancementFilter->GetOutput());
rescale->SetOutputMinimum(0);
```

```
  rescale->SetOutputMaximum(255);

  FileWriterType::Pointer writer = FileWriterType::New();
  writer->SetFileName(argv[2]);
  writer->SetInput(rescale->GetOutput());
  try
  {
    writer->Update();
  }
  catch (itk::ExceptionObject &e)
  {
    std::cerr << e << std::endl;
  }

  RescaleFilterType::Pointer rescale2 = RescaleFilterType::New();
  rescale2->SetInput(multiScaleEnhancementFilter->GetScalesOutput());
  rescale2->SetOutputMinimum(0);
  rescale2->SetOutputMaximum(255);

  FileWriterType::Pointer writer2 = FileWriterType::New();
  writer2->SetFileName(argv[3]);
  writer2->SetInput(rescale2->GetOutput());
  try
  {
    writer2->Update();
  }
  catch (itk::ExceptionObject &e)
  {
    std::cerr << e << std::endl;
  }

}
```

C Example 3 - Multiscale 3D Vesselness

```
#include "itkHessian3DToVesselnessMeasureImageFilter.h"
#include "itkMultiScaleHessianBasedMeasureImageFilter.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkRescaleIntensityImageFilter.h"
#include "itkImage.h"

int main( int argc, char *argv[] )
{
  if ( argc < 4 )
    {
    std::cerr << "Missing Parameters: "
              << argv[0]
              << " Input_Image"
              << " Enhanced_Output_Image"
              << " Scales_Output_Image"
              << std::endl;
```

```
  return EXIT_FAILURE;
  }

// Define the dimension of the images
const unsigned char Dim = 3;

typedef float PixelType;

// Declare the types of the images
typedef itk::Image<PixelType,Dim> ImageType;

typedef itk::ImageFileReader<ImageType> FileReaderType;
typedef itk::ImageFileWriter<ImageType> FileWriterType;

typedef itk::RescaleIntensityImageFilter<ImageType> RescaleFilterType;

// Declare the type of enhancement filter - use ITK's 3D vesselness (Sato)
typedef itk::Hessian3DToVesselnessMeasureImageFilter<double> VesselnessFilterType;

// Declare the type of multiscale enhancement filter
typedef itk::MultiScaleHessianBasedMeasureImageFilter<ImageType,VesselnessFilterType>
 MultiScaleEnhancementFilterType;

// Read the image
FileReaderType::Pointer imageReader = FileReaderType::New();
imageReader->SetFileName(argv[1]);
try
{
  imageReader->Update();
}
catch (itk::ExceptionObject &ex)
{
  std::cout << ex << std::endl;
  return EXIT_FAILURE;
}

// Instantiate the multiscale filter and set the input image
MultiScaleEnhancementFilterType::Pointer multiScaleEnhancementFilter =
 MultiScaleEnhancementFilterType::New();
multiScaleEnhancementFilter->SetInput(imageReader->GetOutput());
multiScaleEnhancementFilter->SetSigmaMin(0.5);
multiScaleEnhancementFilter->SetSigmaMax(4.0);
multiScaleEnhancementFilter->SetNumberOfSigmaSteps(10);

// Get the vesselness filter and set the parameters
VesselnessFilterType* vesselnessFilter =
 multiScaleEnhancementFilter->GetHessianToMeasureFilter();
vesselnessFilter->SetAlpha1(0.5);
vesselnessFilter->SetAlpha2(0.5);

// Now run the multiscale filter
try
{
  multiScaleEnhancementFilter->Update();
```

```
  }
  catch (itk::ExceptionObject &e)
  {
    std::cerr << e << std::endl;
  }

  // Write the enhanced image
  FileWriterType::Pointer writer = FileWriterType::New();
  writer->SetFileName(argv[2]);
  writer->SetInput(multiScaleEnhancementFilter->GetOutput());
  try
  {
    writer->Update();
  }
  catch (itk::ExceptionObject &e)
  {
    std::cerr << e << std::endl;
  }

  // Write the image containing the best response scales
  FileWriterType::Pointer writer2 = FileWriterType::New();
  writer2->SetFileName(argv[3]);
  writer2->SetInput(multiScaleEnhancementFilter->GetScalesOutput());
  try
  {
    writer2->Update();
  }
  catch (itk::ExceptionObject &e)
  {
    std::cerr << e << std::endl;
  }
}
```

References

[1] A. Enquobahrie, L. Ibanez, E. Bullit, and S. Aylward. Vessel enhancing diffusion filter. *The Insight Journal*, 2007. (document), 3, 4, 5, 4

[2] 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

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