Implementation of local structure tensor and enhancement anisotropic diffusion filters in ITK

Release 1.00

Andinet Enquobahrie, Hua Yang and Stephen Aylward

August 10, 2010

Abstract

This paper describes implementation of local structure tensor and anisotropic enhancement diffusion filters using the Insight Toolkit. The anisotropic diffusion filters are implemented using ITK's finite difference solver framework. The filters are used to implement the 3D edge-enhancing diffusion (EED), coherence-enhancing diffusion (CED) and hybrid diffusion with continuous switch noise filtering algorithms described in Mendrick et al [2]. Example programs are provided to demonstrate the use of these filters.

Contents

1 Introduction **1**

2 Filter Designs in ITK **2**

2.1 Structure Local Tensor 3

2.2 Enhancement Anisotropic Diffusion Filters 3

3 Experiments and results **3**

3.1 Local Structure Tensor 3

3.2 Noise filtering using the enhancement filters 4

4 Conclusions **5**

5 Acknowledgment **5**

6 Appendix **5**

1 Introduction

The hybrid diffusion filter with continuous switch (HDCS) is a noise filtering algorithm based on anisotropic non-linear diffusion process[2]. The technique combines edge-preserving noise reduction while enhancing local structures. This algorithm uses a hybrid approach that combines the advantages of edge enhancing diffusion(EED) and Coherence enhancing diffusion(CED).

EED focuses on edge preservation and enhancement. In EED, strong smoothing is applied along the direction of the edge while the strength of the smoothing along the other perpendicular directions depends on the gradient. The higher the gradient the lower the smoothing strength would be. Applying EED to a medical image would enhance boundaries of larger organs but would blurvessels and smaller structures.

On the other hand, CED is designed to to connect lines and improve flow-like structures. Running CED on medical images would preserve smaller structures and filter vessels but would not filter noise and plate-like structures. Therefore, a hybrid technique(HDCS) was proposed to combine intelligently the benefits of the two techniques.

The main underlying equation in this algorithm is the anisotropic diffusion equation

$$\frac{\delta u}{\delta t} = \nabla.(D.\nabla u) \tag{1}$$

Where
∇ is the divergence operator
∇u is the gradient of the image u
D is the diffusion tensor

The diffusion tensor D allows to tune the smoothing(both the strength and direction) across the image. D is defined as a function of the structure tensor.

$$J(\nabla u_\sigma) = K_\rho * (\nabla u_\sigma \nabla u_\sigma^T) \tag{2}$$

Where K is the Gaussian Kernel with standard deviation ρ scale and ∇u_σ is the gradient of the image u at scale σ. Expanding the gradients, one gets

$$J(I) = K * \begin{bmatrix} I_x^2 & I_xI_y & I_xI_z \\ I_xI_y & I_y^2 & I_yI_z \\ I_xI_z & I_yI_z & I_z^2 \end{bmatrix} \tag{3}$$

$$D = [V_1V_2V_3] . \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} . [V_1V_2V_3]^T \tag{4}$$

V_1, V_2, V_3 denote the eigen vectors of the structure tensor. The eigenvalues λ_i define the strength of the smoothing along the direction of the corresponding eigen vector V_i. EED, CED and HDCS differ in the way they define λ_i.

2 Filter Designs in ITK

The anistotropic diffusing filters implementation consists of two steps. The first step involves developing a filter that computes the local structure tensor that is not currently available in ITk. The second part involves integrating the structure tensor into a diffusion tensor for anisotropic diffusion filtering.

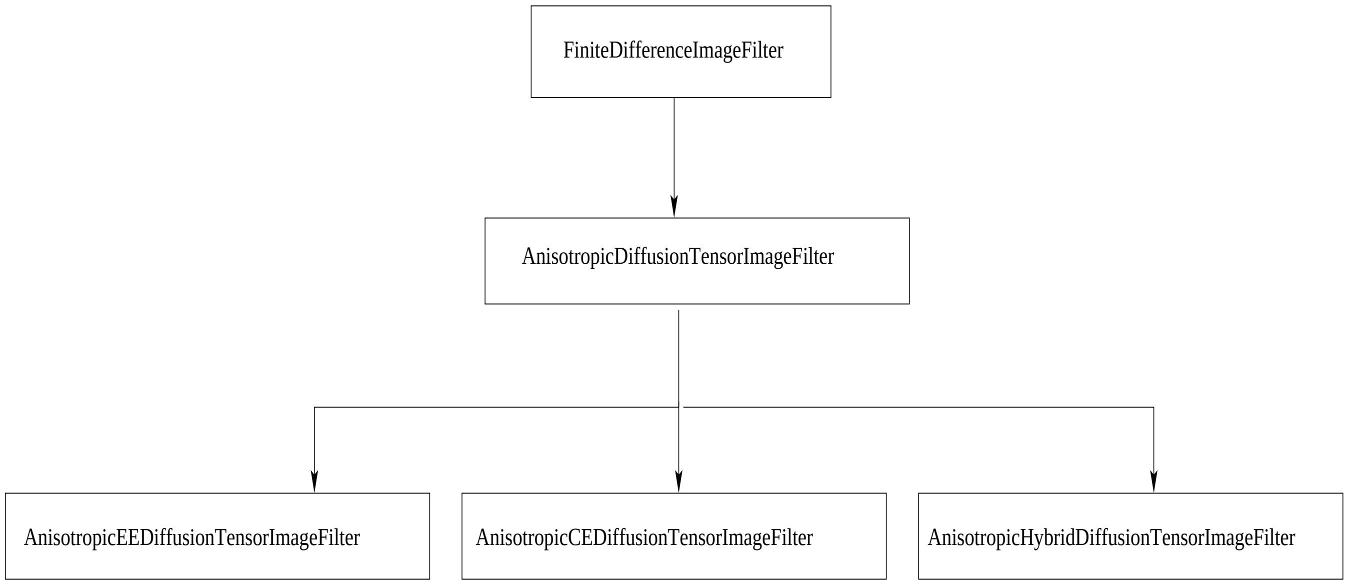
FiniteDifferenceImageFilter
AnisotropicDiffusionTensorImageFilter
AnisotropicEEDiffusionTensorImageFilter
AnisotropicCEDiffusionTensorImageFilter
AnisotropicHybridDiffusionTensorImageFilter

Figure 1: Anisotropic diffusion filters class hierarchy

2.1 Structure Local Tensor

The principle directions of diffusion/smoothing are based on the local structure. For this purpose, local structure tensor generator is needed. We implemented such type of filter using ITK's recursive Gaussian filter. This filter is implemented using the recursive gaussian filters in ITK.

2.2 Enhancement Anisotropic Diffusion Filters

The implementation of the enhancement anisotropic diffusion filters follows ITK's finite difference solver (FDS) framework. The framework has two components: Function and solver objects. The solver object establishes the infrastructure for accepting input image and producing output image. The function object computes a single scalar value from a neighborhood of values and computes the incremental change at a pixel in the solution image from one iteration of the solver to the next. The solver object and the the function objects are derived from itk::FiniteDifferenceImageFilter and itk::FiniteDifferenceFunction respectively. Figure 1 shows the class hierarchy of the diffusion filters. An example program that demonstrates how to use this filter is provided in appendix.

3 Experiments and results

We tested the local structure tensor and diffusion filters on synthetic and real data.

3.1 Local Structure Tensor

For validation of the local structure tensor, we generated a 3D synthetic image with a sinusoid pattern $sin(x)$ (see Figure 2(A)). Note that $I_x = cos(x)$ and $I_y = I_z = 0$. Therefore the 3x3 tensor consists of only one

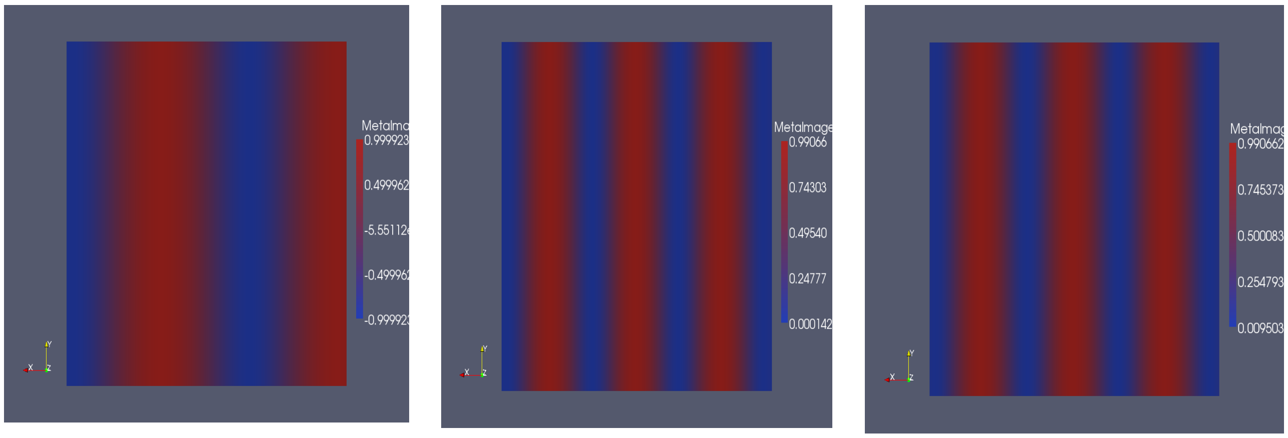

Figure 2: Validation of structure tensor using a synthetic sinusoid image. The three sub-figures respectively show 2D $(X-Y)$ cross sections of three 3D images: (A) the original sinusoid image $sin(x)$, (B) the reference image $cos(x)*cos(x)$, and (C) a 3D scalar image showing the first element $({I_x}^2)$ of the result structure tensor at each pixel.

non-zero element $I_x * I_x = cos(x) * cos(x)$. We validated the structure tensor filter by comparing its output (Figure 2(B)) with the expected analytical function(Figure 2(C)).

In addition, for visual validation of the results of the local structure tensor filter, we generated a cylinderical spatial object and applied the structure local tensor filter. One good way of visuall inspecting the results would be to overlay the primary eigen vectors on the input image. For this, we used Paraview, an open-source, multi-platform data analysis and visualization application, to generate the visualization shown in Figure 3 as follows

1. Load the synthetic cylinder image.
2. Apply a contour filter to generate surface rendered view of the cylinder.
3. Load the primary eigen vector image.
4. Generate a glyph visualization overlay on the cylinder surface. The glyphs are oriented according to the eigen vector of the local structure tensor.

3.2 Noise filtering using the enhancement filters

Experiments were conducted to test the effectiveness of the anisostropic filters in removing noise from a lung CT scan. Figure 4a) shows CT scan used to test the algorithm. The testing dataset is distributed as part of the source code submission to the Insight Journal. We run the filters using the default parameter values suggested in [2] and got the results shown in Figure **??** B, C, and D running the CED, EED and HDCS filters respectively on the cropped lung CT test image.

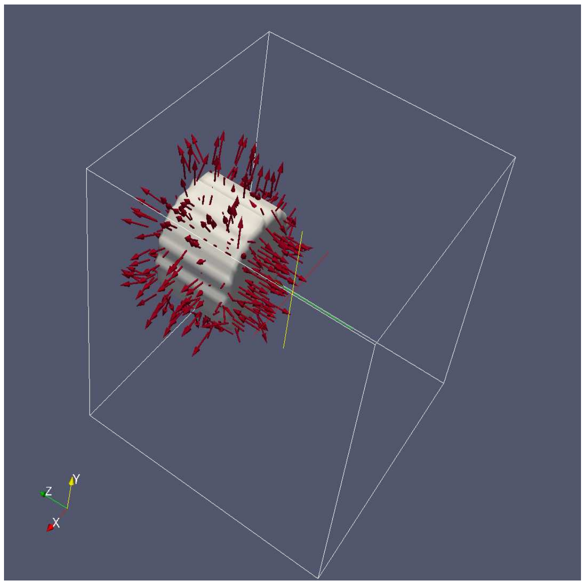

Figure 3: Visualization of the structure tensor primary eigen vectors overlaid as glyphs on the synthetic cylinder surface

4 Conclusions

In this paper, we have described local structure tensor and diffusion anistotropic filters implemented using ITK. The filters are used to implement the 3D edge-enhancing diffusion (EED), coherence-enhancing diffusion (CED) and hybrid diffusion with continuous switch (HDCS) noise filtering algorithms developed by Mendrick et al [2]. We tested the filters on synthetic and lung CT scans.

5 Acknowledgment

This work was supported by NIH/NCI sponsored ”Image Registration for Ultrasound-Based Neurosurgical Navigation” project, 1R01CA138419-01 (Neuralnav, PI:Aylward, Wells)

6 Appendix

This example demonstrates how to use the itk::AnisotropicHybridDiffusionImageFilter to filter the input image using the hybrid diffusion filter using a continous switch.

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

int itkAnisotropicHybridDiffusionImageFilterTest(int argc, char* argv [] )
{
  if ( argc < 3 )
```

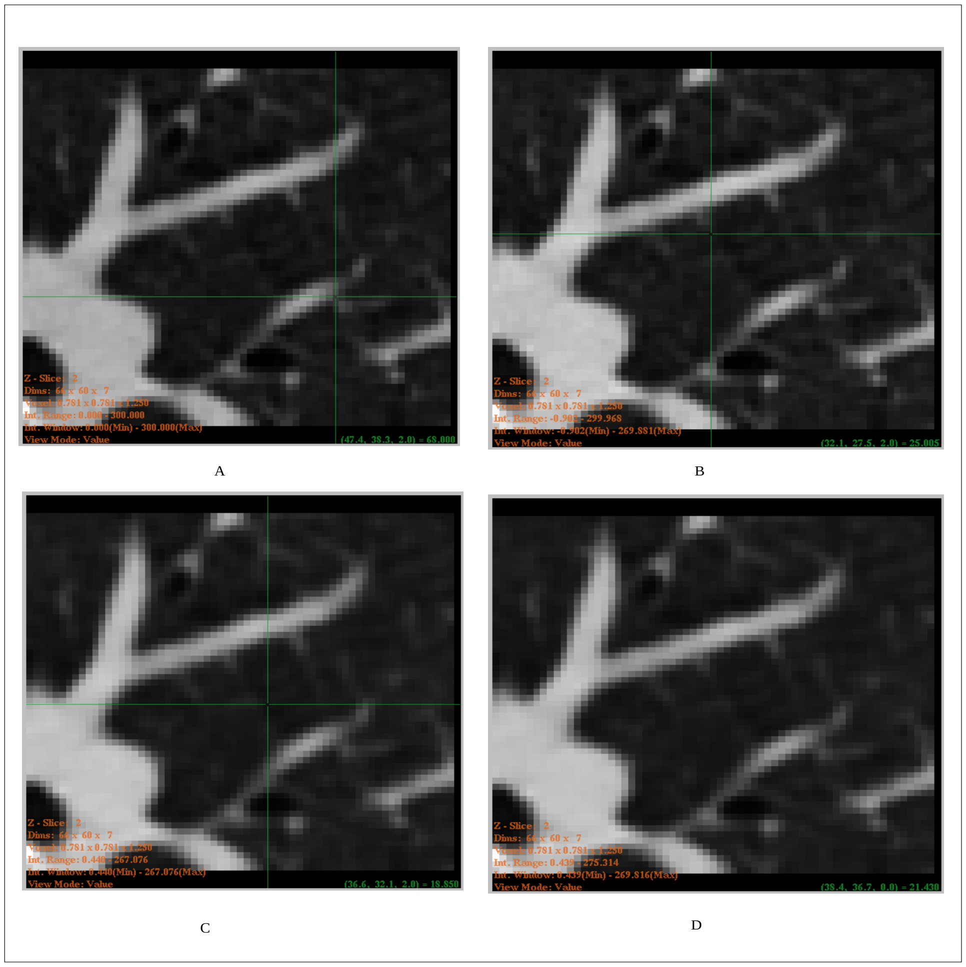

Figure 4: Anisotropic filters applied on lung CT scan a) Original lung CT scan slice b) CED C) EED D) HDCS filtered results.

```
    {
    std::cerr << "Missing Parameters: "
              << argv[0]
              << " Input_Image"
              << " Edge_Enhanced_Output_Image "
              << std::endl;
    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 edge enhancement filter
  typedef itk::AnisotropicHybridDiffusionImageFilter< InputImageType,
                                          OutputImageType>  HybridFilterType;

  // Create a edge enhancement Filter
  HybridFilterType::Pointer HybridFilter =
                                    HybridFilterType::New();

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

  try
    {
```

```
    HybridFilter->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 ( HybridFilter->GetOutput() );

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

  return EXIT_SUCCESS;

}
```

References

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

[2] A.M. Mendrik, E.J Vonken, A. Rutten, M.A Viergever, and B. Van Ginneken. Noise reduction in computed tomography scans using 3-d anisotropic hybrid diffusion with continuous switch. *IEEE Transactions on Medical Imaging*, 28(10):1585–94, 2009. (document), 1, 3.2, 4