The Image-based Vascular Analysis Toolkit (IvanTk)

Release 0.00

Ivan Macia[1]

September 13, 2013

[1]Vicomtech-IK4 Foundation, Donostia-San Sebastian (Spain)
imacia@vicomtech.org

Abstract

This document describes the *Image-based Vascular Analysis Toolkit (IvanTk)*, a set of cross-platform C++ open source software libraries for the detection, extraction and modeling of vessels in 2D/3D medical images based on the *Insight Toolkit*. We provide an overview of the libraries and their purpose and describe their functional modules and future directions of the toolkit.

Contents

1 Introduction

The *Image-based Vascular Analysis Toolkit* (*IvanTk*) is a set of C++ cross-platform software libraries for 2D/3D vascular analysis in medical imaging developed using concepts of generic programming. It can be considered as an extension of the Insight Toolkit (ITK) (www.itk.org) [?] for vascular image analysis and implements many of the vascular detection, extraction and modeling methods described and used for the experiments in the PhD thesis *"Medical Image Analysis for the Detection, Extraction and Modelling of Vascular Structures"* (urlhttp://bit.ly/18fIIoV) [?]. The libraries are distributed under the BSD License.

The great number of existing approaches to vascular analysis, its increasing complexity and the absence of a unified framework makes difficult to compare the performance of the different methods in a meaningful, consistent and reproducible manner. The implementation of such methods in an object-oriented design allows reusability of its constituent building-blocks, provides a better understanding of its algorithmic mechanisms, reveals important implementation details and improves reproducibility of experiments [?].

2 Motivation

The main motivation behind the development of *IvanTk* was the lack of dedicated libraries for image-based analysis of vascular structures. Some of the existing open source medical image analysis libraries, such as the Insight Toolkit (ITK) (www.itk.org), implement a few popular vascular detection methods. *IvanTk* may be considered an extension of ITK dedicated specifically to vascular analysis, since it reuses and extends many of the concepts and classes of the libraries. The Vascular Modeling Toolkit (VMTK) (www.vmtk.org) is an open-source library dedicated to vascular analysis, but it is more oriented to computational geometry of vessels, generating detailed patient-specific models that can be used for structural or computational fluid dynamics studies. At the same time of the development of our software, a similar toolkit called *TubeTk* (www.tubetk.org) was developed, for the segmentation, registration and analysis of tubes and surfaces of images. *IvanTk* does not provide any visualization or GUI functionalities, although it is compatible with libraries such as the Visualization Toolkit (www.vtk.org) (see related example on the toolkit) and with different C++ GUI frameworks. This software pretends to be complementary and if possible compatible to the aforementioned softwares.

Currently, the main focus of *IvanTk* is on vascular detection, extraction and modeling, trying to establish a framework in which different algorithm implementations can be compared consistently. It also incorporates object-oriented paradigms modeling extraction schemes that allow to replace components in the extraction chain. Currently this is being developed specifically for vessel tracking algorithms. Finally, it implements a set of tools to develop synthetic tubular volumetric models that may be useful in different experiments. Most of the current implementations of vascular detectors in *IvanTk* are local, point-based approaches, many of which kernel-based, that may not be effective computationally for exploring large volumes. Instead, they are designed to be used in local extraction schemes (such as vessel tracking) and for testing a sparse number of candidate points, with the purpose of comparing accuracy or robustness of different approaches.

In the future, it is expected to incorporate also global approaches, such as those operating in the frequency domain, that require computing the vesselness value densely for the whole volume. Another important pending enhancement is to provide faster implementations, specially for kernel-based operations such as Gaussian filtering which are used extensively in the toolkit. For this, the novel GPU acceleration framework in ITK v4 could be used in some algorithms.

3 Scope of the Libraries

The *IvanTk* libraries currently provide the following features:

- A repository of (3D) vascular detection algorithms: currently most of the methods are based on centerline-based detection approaches, with its main focus on medialness functions.
- A component-based unified framework for (3D) vascular tracking called *Generalized Vascular Tracking* (GVT): it models vascular tracking procedures in a set of different individual stages and components which accommodate existing methods in a flexible manner. Its object-oriented architecture allows testing the influence of the individual components. Moreover, complex extraction schemes can be generated by creating new components or extending existing ones, and by combining different parts of existing methods in innovative ways.
- An implementation of the Vessel Information Model (VIM)[1]: it allows the representation of vascular structures in a way that can be easily used to store the results of the vascular extraction algorithms, so that this information can be readily used by the clinical applications (see [**?**] Ch. 3).
- A means of generating synthetic models of ideal shapes as volumetric images, such as the cylinder, toroid or helix models (see (see [**?**] Ch. 3) with different cross-sections, shape, image parameters and noise levels, which are representative of different aspects of vessels such as elongated shape, presence of curvature and/or torsion. These models may be used for testing or validation purposes.
- A means of reproducing the experiments of the thesis, since they are incorporated as unit tests and examples with configurable parameters.

4 Functional Modules

4.1 Modeling Module

The purpose of the Modeling Module is to represent and structure information that is relevant both to vascular image analysis and related clinical applications. This representation is called *Vessel Information Model* (VIM) and is described (with the VKR name) in [**?**]. The model should accommodate the vascular information obtained from the image analysis stages in such a way that can be used efficiently in the clinical applications. Such model permits easily reusing of software pieces through appropriate abstractions, facilitating the development of novel methods, procedures and applications of vascular image analysis.

The data structures of the model are designed in order to describe qualitative (i.e. section shape, anatomical location, diagnostic annotations) and quantitative (section area, curvature, etc.) information in such a way that it may be handled efficiently both by the extraction schemes and by the final applications while keeping a high degree of versatility. The data structures represent vessel information at different levels, ranging from a complete vascular network to a vessel center or boundary point.

Two main structures can be identified within this model:

[1]Originally this model was named Vessel Knowledge Representation (VKR) model but is renamed to avoid possible confusion with semantic knowledge representation models in artifical intelligence and because the term "‘information model" better fits its purpose.

- *Vessel Graph* (class `VesselGraph`)[2]: it is the highest level element representing a vascular network. It comprises a set of nodes representing vessel branches and bifurcations, as well as vessel "features" that describe the presence of an abnormality, accident or annotation. The Vessel Graph is also useful to perform standard graph algorithms on nodes, in order to perform tasks such as pruning spurious branches or minimum cost paths[3].
- *Vessel Centerline* (class `VesselCenterline`): the centerline is an important component of the VIM model for two main reasons: first, it is a good descriptor of the vessel shape and second, it allows to refer spatially other information to the corresponding centerline points. A centerline is the main component of `VesselBranchNode` objects in the Vessel Graph. In fact, the `VesselGraph` class itself is templated over the Centerline model. A centerline in VIM can be thought of as a container of sections (class `VesselSection`). The most simple section type is a single point, but the model allows for storing almost any type of section (circular, radial profile etc.). Stored discretized centerline points are independent from a possible underlying mathematical model describing the centerline (such as a B-spline) but are necessary to localize quantitative properties along the centerline. Both, the centerline and section models are heavily used within *IvanTk*.

4.2 Detection Module

The Detection Module is one of the most important parts of the toolkit. It mainly consists of vesselness functions for vascular detection, most of them corresponding to medialness functions enhancing vessel centerlines (see [?] Ch 5). It implements both central (derivative-based) and offset (integral-based) medialness functions, and provides support for multi-scale analysis.

Figure 1 shows the inheritance diagram for the implemented vesselness functions. They take the form of an `itk::ImageFunction`, meaning that they provide a value for every discrete or continuous point in physical space. Many of the existing implementations required computing vesselness functions densely accross the whole volumetric space. In many cases, this is prohibitive in terms of computational resources, for example, when it is required to calculate the six components of the 3D Hessian matrix with single or double floating-point precision for every image pixel. The implementation in *IvanTk* assumes that these calculations are to be performed sparsely, due to the local nature of the extraction scheme, such as those corresponding to vessel tracking or minimum cost paths, or due to a preselection or filtering of points which greatly reduces the number of vessel candidate points. Thus, *IvanTk*, allows to calculate most vesselness values locally, only at the desired specific points.

A few important base classes are the following:

- `HessianBasedVesselnessImageFunction`: corresponds to vesselness functions that require the calculation of the local Hessian matrix, and possibly its eigenvalues and vectors. The main subclasses are:
  - `HessianOnlyBasedVesselnessImageFunction`: corresponds to the vessselness functions that only require information derived from the Hessian matrix for obtaining the function value.

[2]Although the graph-based representation is conceptually correct, its implementation is currently incomplete and subject to changes, which include the use of the Boost Graph Library (BGL) and some improvements in the model. For this reason, we discourage a heavy usage of the higher level structures beyond what is shown in the examples.

[3]Work in progress which has been partially implemented but is not yet in the toolkit.

Hence, it corresponds to central medialness approaches which include, among others, the functions of Sato *et al.* [?] and Frangi *et al.* [?]. The Hessian eigenvalues and eigenvectors may also be obtained directly, and may be used, for example, for vessel section estimation.

  - `OffsetMedialnessImageFunction` : corresponds to the offset medialness of Krissian *et al.* [?].
  - `FluxBasedVesselnessImageFunction` : corresponds to flux-based approaches, and includes the non-linear steerable filter of Koller *et al.* [?] or the approach of Lesage *et al.* in [?].

- `SphereGridBasedImageFunction` : corresponds to vesselness functions that use a sphere grid for calculations. It includes, among others, the Optimally Oriented Flux of Law and Chung [?], the polar profile vesselness measure of Qian *et al.* [?], and a means of directly calculating the eigenvalues and eigenvectors of the oriented flux matrix, as in the case of the Hessian matrix.

- `MultiScaleImageFunction` : implements a generic means of converting single-scale image functions into multi-scale image functions, as long as that some conditions about the scale representation are met for the target functions. It also provides the ability to specify the way in which scales are integrated.

4.3 Extraction Module

The Extraction Module contains the algorithms for vessel extraction, which produce a set of vascular descriptors as output. The vesselness functions from the Detection Module may be used in order to determine the vessel (centerline) locations. It implements the *Generalized Vessel Tracking* (GVT) framework described in [?] Ch. 6 as a generic process model for vascular tracking.

In this article, the main components and stages of the GVT process model are briefly described. The vessel tracking procedure can be thought of as an iterative procedure with different sequential *stages*. The different interchangeable models, metrics and algorithms implemented following an object-oriented programming paradigm are called *components*.

The main components in GVT are the following:

- *Section Estimator*: calculates locally, for every point, an estimate of the section normal and in some cases the local radius based on image content, previous estimations and section models. The Hessian Matrix and the Oriented Flux Matrix (OFM) are examples of section estimators from the eigenvalue analysis of these matrices.

- *Vesselness Metric*: is a measure of the likelihood of a point of being part of a vessel. The metric may be used in several parts of the extraction process, for example, to estimate the correct scale maximizing this metric, or to search for the centerline point.

- *Centerline Model*: defines which output extraction information is stored. It may be a set of centerline points or complete description of sections associated to these points.

- *End Condition*: provides the stopping criterion for the algorithm which, for example, may be a maximum number of iterations or may be based on the vesselness metric.

- *Scale Estimator* (optional): used in multi-scale approaches. It estimates the optimal scale for performing calculations on the current point. In general, the scale is proportional to the relative size of the vascular structures. The estimation of the scale is usually based on a (vesselness) metric.

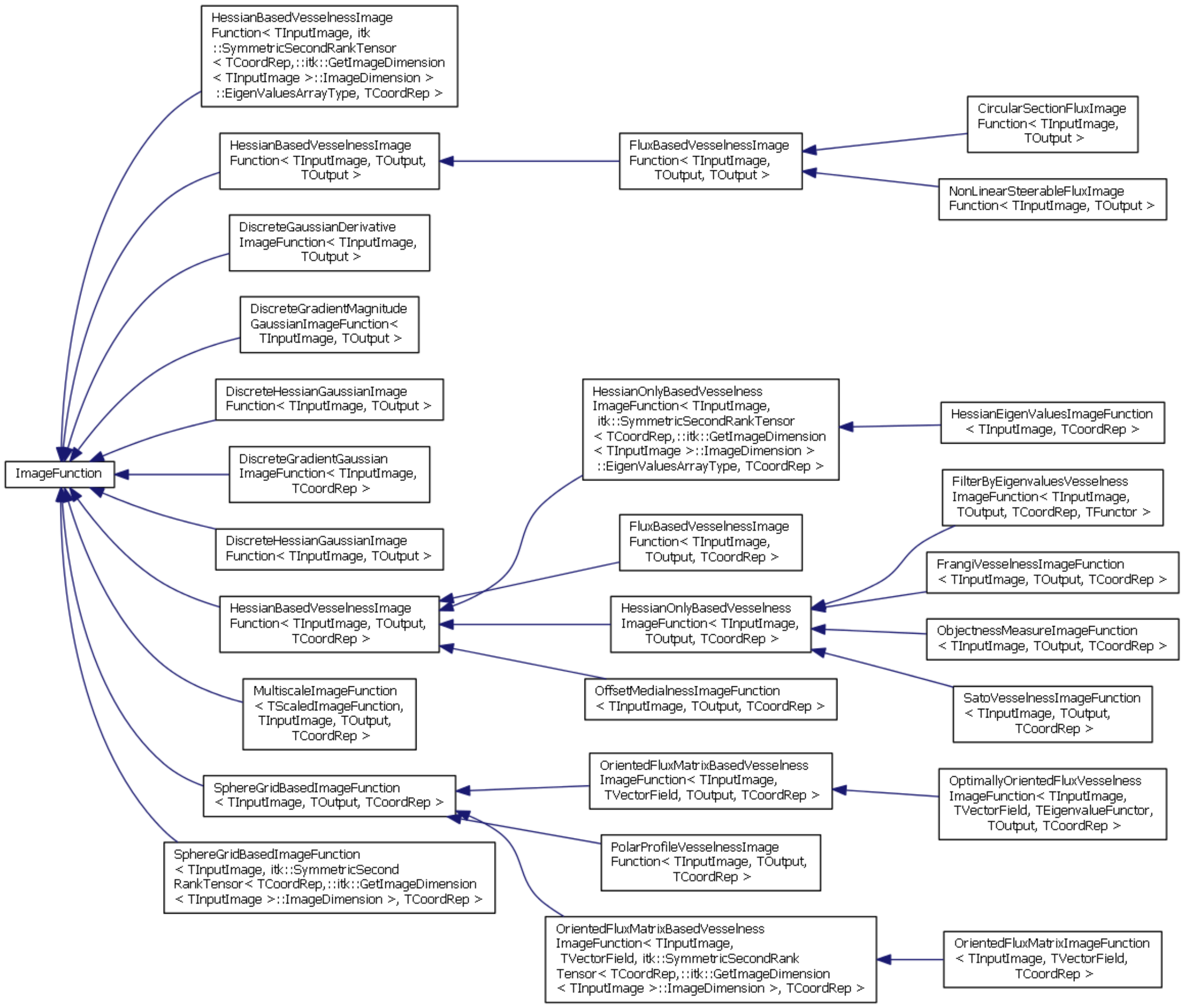

ImageFunction
HessianBasedVesselnessImage Function< TInputImage, itk ::SymmetricSecondRankTensor < TCoordRep,::itk::GetImageDimension < TInputImage >::ImageDimension > ::EigenValuesArrayType, TCoordRep >
HessianBasedVesselnessImage Function< TInputImage, TOutput, TOutput >
FluxBasedVesselnessImage Function< TInputImage, TOutput, TOutput >
CircularSectionFluxImage Function< TInputImage, TOutput >
NonLinearSteerableFluxImage Function< TInputImage, TOutput >
DiscreteGaussianDerivative ImageFunction< TInputImage, TOutput >
DiscreteGradientMagnitude GaussianImageFunction< TInputImage, TOutput >
DiscreteHessianGaussianImage Function< TInputImage, TOutput >
HessianOnlyBasedVesselness ImageFunction< TInputImage, itk::SymmetricSecondRankTensor < TCoordRep,::itk::GetImageDimension < TInputImage >::ImageDimension > ::EigenValuesArrayType, TCoordRep >
HessianEigenValuesImageFunction < TInputImage, TCoordRep >
DiscreteGradientGaussian ImageFunction< TInputImage, TCoordRep >
FilterByEigenvaluesVesselness ImageFunction< TInputImage, TOutput, TCoordRep, TFunctor >
DiscreteHessianGaussianImage Function< TInputImage, TOutput >
FluxBasedVesselnessImage Function< TInputImage, TOutput, TCoordRep >
FrangiVesselnessImageFunction < TInputImage, TOutput, TCoordRep >
HessianBasedVesselnessImage Function< TInputImage, TOutput, TCoordRep >
HessianOnlyBasedVesselness ImageFunction< TInputImage, TOutput, TCoordRep >
ObjectnessMeasureImageFunction < TInputImage, TOutput, TCoordRep >
OffsetMedialnessImageFunction < TInputImage, TOutput, TCoordRep >
SatoVesselnessImageFunction < TInputImage, TOutput, TCoordRep >
MultiscaleImageFunction < TScaledImageFunction, TInputImage, TOutput, TCoordRep >
OrientedFluxMatrixBasedVesselness ImageFunction< TInputImage, TVectorField, TOutput, TCoordRep >
OptimallyOrientedFluxVesselness ImageFunction< TInputImage, TVectorField, TEigenvalueFunctor, TOutput, TCoordRep >
SphereGridBasedImageFunction < TInputImage, TOutput, TCoordRep >
PolarProfileVesselnessImage Function< TInputImage, TOutput, TCoordRep >
SphereGridBasedImageFunction < TInputImage, itk::SymmetricSecond RankTensor< TCoordRep,::itk::GetImageDimension < TInputImage >::ImageDimension >, TCoordRep >
OrientedFluxMatrixBasedVesselness ImageFunction< TInputImage, TVectorField, itk::SymmetricSecondRank Tensor< TCoordRep,::itk::GetImageDimension < TInputImage >::ImageDimension >, TCoordRep >
OrientedFluxMatrixImageFunction < TInputImage, TVectorField, TCoordRep >

Figure 1: Inheritance diagram of vesselness functions in IvanTk

A process diagram describing the GVT process is shown in 2. Most of the stages are performed at each iteration. Others, such as the *Initialize* and *Post-process* stages are performed only once. The most important stages are:

- *Initialize*: initializes the tracking procedure. The simplest initialization consists of providing a seed point in the vessel centerline.
- *Turn*: finds the correct orientation and section of the vessel when a new (centerline) point is obtained, after advancing one step (*Step* stage), or after searching for the correct centerline point (*Search* stage). The process involves using a *Section Estimator* and possibly, in multi-scale approaches, a *Scale Estimator* and a *Vesselness Function*.
- *Search*: the purpose of this stage is to correct the trajectory in the vicinity of the new vessel candidate point. For example, it might be the maximum value of the vesselness (medialness) in the previously estimated plane or the closest ridge point of a medialness function. A *Turn* stage estimates the orientation and section of the new centerline point. This may be done once or in an iterative loop as shown in 2.
- *Measure*: once a new vessel (centerline) point is found, local or cumulative quantitative measurements may be performed. This may involve, for example, a detailed section shape or profiles, curvatures or cumulative lengths.
- *Step*: this stage estimates the new direction of advance and performs a step following the current direction. The simplest direction estimation is the direction of the current section normal. More advanced strategies include filtering (i.e. averaging) the trajectory, extrapolation, etc. The process also involves setting the size of the current step, which may be fixed or adaptive.

The main object of the GVT implementation is the `VesselTrackerFilter` class, which inherits `ImageToVesselDataObjectFilter` (see Figure 3). It is the base class for the algorithms that take an image as input, and produce a vessel network descriptor as output, usually in the form of a `VesselGraph` (see [**?**] Ch. 3).

The result is stored as a `VesselCenterline` in the current `VesselBranchNode` of the `VesselGraph`. The `VesselTrackerEndCondition`, is the base class that tests the end condition of the algorithms, for example, a maximum number of iterations or a minimum value of the medialness.

The collaboration diagram for the `VesselTrackerFilter` class is shown in Figure 4. The most important component used by this object is the `VesselSectionEstimator`, which computes vessel sections during the *Turn* and *Search* stages of the GVT and ultimately determines the centerline path. Figure 5 shows the class diagram for the *Section Estimator* classes. The following main subclasses are considered:

- `FixedScaleHessianBasedVesselSectionEstimator` / `FixedScaleOOFBasedVessel-SectionEstimator`: implements the section estimation the eigenvectors of the Hessian and Oriented Flux matrices respectively at a single, user-selected scale.
- `MultiScaleVesselSectionEstimator`: estimates vascular sections at given locations based on image content and at multiple scales. It includes support for estimating the section by eigenvalue analysis of different types of local orientation matrices via the subclass `MultiscaleTensorBasedVesselSectionEstimator`. Two main approaches are implemented, for the Hessian matrix (`MultiScaleHessianBasedVesselSectionEstimator`) and for the Oriented

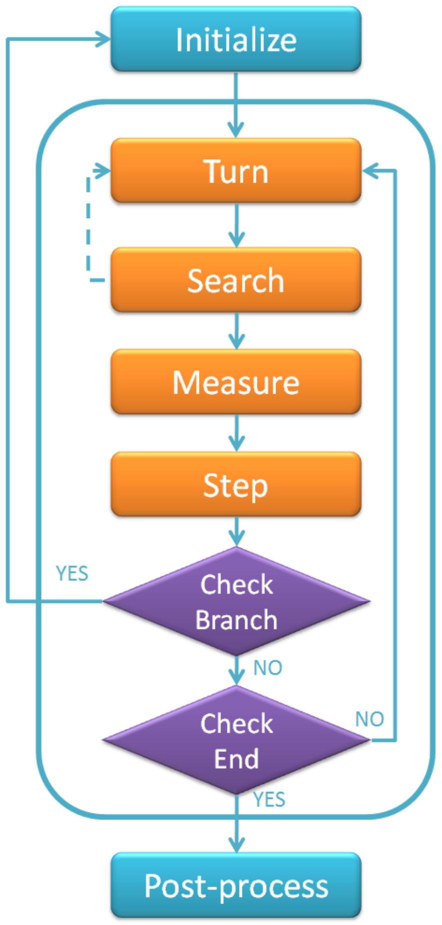


Figure 2: *Generalized Vascular Tracking* (GVT) process model in *IvanTk*

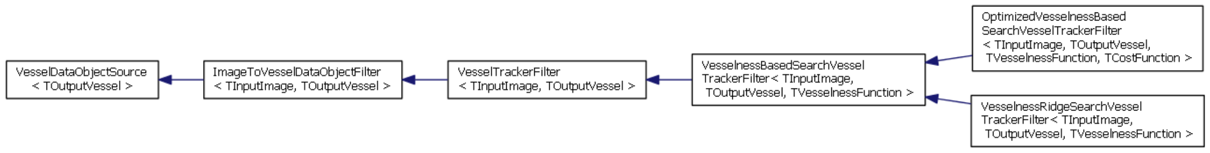

VesselDataObjectSource < TOutputVessel >
ImageToVesselDataObjectFilter < TInputImage, TOutputVessel >
VesselTrackerFilter < TInputImage, TOutputVessel >
VesselnessBasedSearchVessel TrackerFilter< TInputImage, TOutputVessel, TVesselnessFunction >
OptimizedVesselnessBased SearchVesselTrackerFilter < TInputImage, TOutputVessel, TVesselnessFunction, TCostFunction >
VesselnessRidgeSearchVessel TrackerFilter< TInputImage, TOutputVessel, TVesselnessFunction >

Figure 3: Inheritance diagram for vessel tracking classes in IvanTk

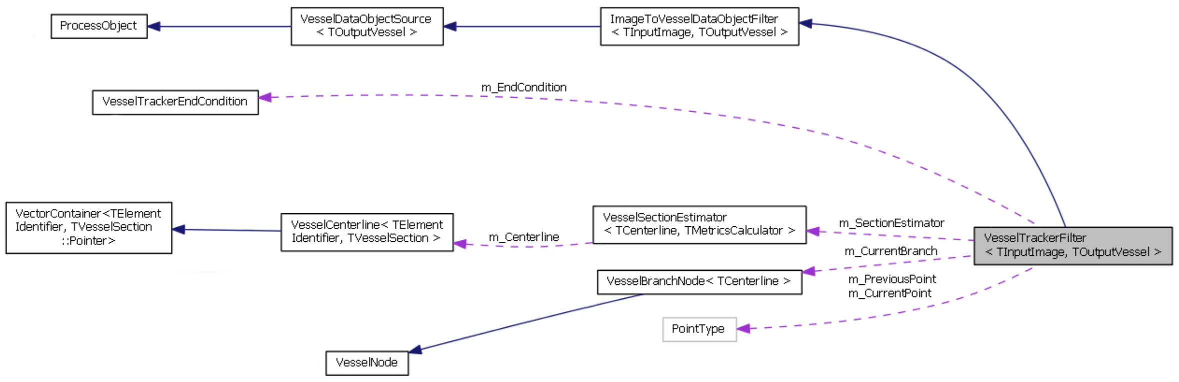

ProcessObject
VesselDataObjectSource < TOutputVessel >
ImageToVesselDataObjectFilter < TInputImage, TOutputVessel >
VesselTrackerEndCondition
m_EndCondition
VectorContainer<TElement Identifier, TVesselSection ::Pointer>
VesselCenterline< TElement Identifier, TVesselSection >
m_Centerline
VesselSectionEstimator < TCenterline, TMetricsCalculator >
m_SectionEstimator
m_CurrentBranch
VesselTrackerFilter < TInputImage, TOutputVessel >
VesselBranchNode< TCenterline >
m_PreviousPoint
m_CurrentPoint
PointType
VesselNode

Figure 4: Collaboration diagram for vessel tracking classes in IvanTk

Flux matrix (`MultiScaleOOFBasedVesselSectionEstimator`). Adaptive subclasses of these two estimators automatically select the best scale for estimation based on a medialness metric function.

- `OptimizedVesselSectionEstimator`: implements the novel section estimation with optimization described in [**?**] Ch 6. It optimizes a `VesselSectionFitCostFunction` with different possible types of Optimizer objects.

Figure 6 shows the multiscale tracking in *IvanTk* operating on real CT datasets. Trajectories and section normals are not filtered and still, with and adequate step size, the algorithm is able to track the vessel. Note that in Figure 6 (c) the centerline tracking tries to adhere to one of the vessel walls and then continues through one of the branches at the bifurcation. This is because a small maximum scale was selected. Mechanisms for trajectory and normal filtering, bifurcation detection and more intelligent scale selection are still to be implemented. However, these can be accommodated easily into the framework.

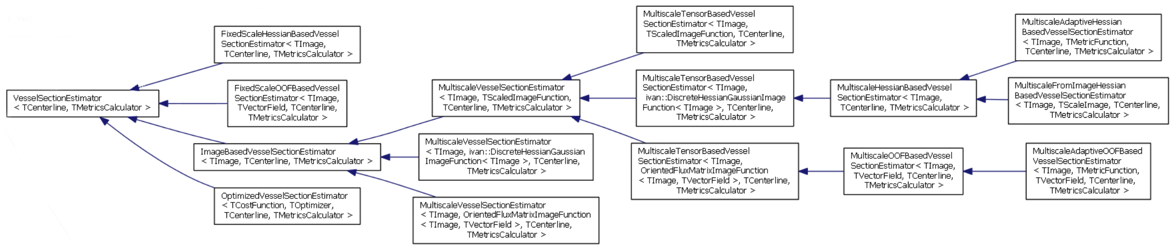

VesselSectionEstimator < TCenterline, TMetricsCalculator >
FixedScaleHessianBasedVessel SectionEstimator< TImage, TCenterline, TMetricsCalculator >
FixedScaleOOFBasedVessel SectionEstimator< TImage, TVectorField, TCenterline, TMetricsCalculator >
ImageBasedVesselSectionEstimator < TImage, TCenterline, TMetricsCalculator >
OptimizedVesselSectionEstimator < TCostFunction, TOptimizer, TCenterline, TMetricsCalculator >
MultiscaleVesselSectionEstimator < TImage, TScaledImageFunction, TCenterline, TMetricsCalculator >
MultiscaleVesselSectionEstimator < TImage, ivan::DiscreteHessianGaussian ImageFunction< TImage >, TCenterline, TMetricsCalculator >
MultiscaleVesselSectionEstimator < TImage, OrientedFluxMatrixImageFunction < TImage, TVectorField >, TCenterline, TMetricsCalculator >
MultiscaleTensorBasedVessel SectionEstimator< TImage, TScaledImageFunction, TCenterline, TMetricsCalculator >
MultiscaleTensorBasedVessel SectionEstimator< TImage, ivan::DiscreteHessianGaussianImage Function< TImage >, TCenterline, TMetricsCalculator >
MultiscaleTensorBasedVessel SectionEstimator< TImage, OrientedFluxMatrixImageFunction < TImage, TVectorField >, TCenterline, TMetricsCalculator >
MultiscaleHessianBasedVessel SectionEstimator< TImage, TCenterline, TMetricsCalculator >
MultiscaleOOFBasedVessel SectionEstimator< TImage, TVectorField, TCenterline, TMetricsCalculator >
MultiscaleAdaptiveHessian BasedVesselSectionEstimator < TImage, TMetricFunction, TCenterline, TMetricsCalculator >
MultiscaleFromImageHessian BasedVesselSectionEstimator < TImage, TScaleImage, TCenterline, TMetricsCalculator >
MultiscaleAdaptiveOOFBased VesselSectionEstimator < TImage, TMetricFunction, TVectorField, TCenterline, TMetricsCalculator >

Figure 5: Inheritance diagram for section estimator classes in IvanTk

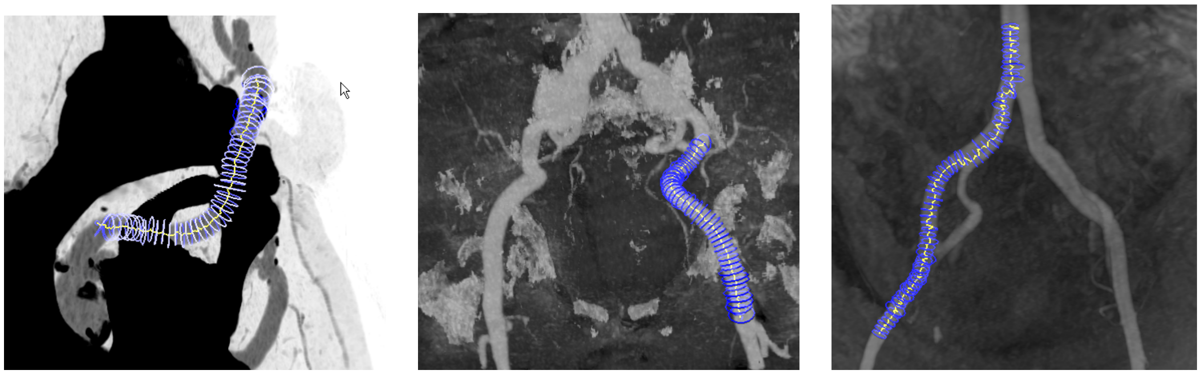


Figure 6: Volume renders of multiscale vascular tracking in real CT datasets. Centerline trajectories are shown in yellow and corresponding sections in blue, with their radius corresponding to estimated scale.

4.4 Synthetic Module

The Synthetic Module implements ideal image models for vascular analysis and other auxiliary objects such as platonic solids as 3D meshes.

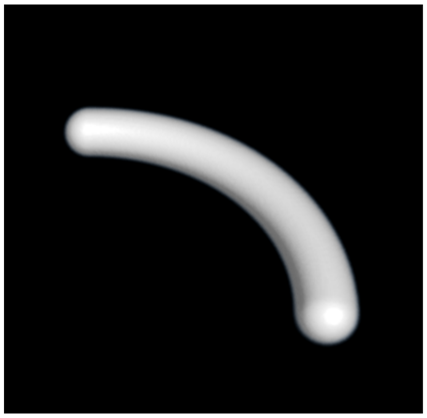
We can distinguish three main types of objects used for creating the ideal models:

- *Section Generators*: objects which create images of ideal sections with specific shape and intensity profiles. For example, a `CircularBarSectionGenerator` obtains an image of a circular section with constant/bar intensity profile whereas a `CircularGaussianSectionGenerator` creates an image of a circular section with Gaussian profile of the desired aperture. Several shape or image parameters can be configured, such as the circle radius, or image size, spacing or maximum value.
- *Centerline Generators*: objects which create centerline curves of different shapes. These may be used, among other things, to generate complex volumetric models from these centerlines. For example, a `CircularCenterlineGenerator`, creates a centerline corresponding to a full circumference or a circular arc, whereas a `CircularHelixCenterlineGenerator` creates an helix centerline with the desired radius, pitch and height.
- *Volumetric Generators*: objects that create volumetric ideal shape models. These include the cylinder, toroid and helix shape models. Section Generators and Centerline Generators may be used to create such models. For example a `CircularBarTubeGenerator` creates a straight cylinder whose section is a `CircularBarSectionGenerator`. `GaussianCircularHelixGenerator` creates an helix with Gaussian profile whose centerline is defined by a `CircularHelixCenterlineGenerator`.

Figure 7 shows two examples of volumetric synthetic datasets, a toroid and an helix with Gaussian cross-sections, generated with the module. The implementation generates these shapes by sweeping a Gaussian ball along the desired trajectory. The module provides several parameters that ultimately determine the final shape. Since the centerline trajectory is known, it can be used as a ground truth in experiments.

5 Installation and First Steps

To install, first download the latest snapshot from the Git repository


(a) Gaussian Toroid ($\sigma = 10$, $R = 250$)

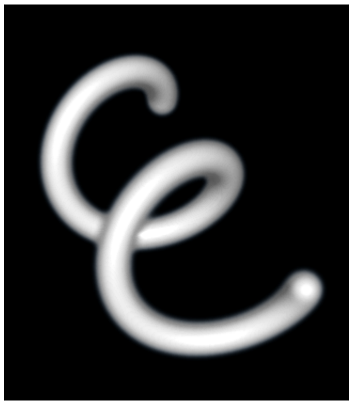

(b) Gaussian Helix ($\sigma = 3$, $R = 30$, $H = 10$, $\theta = 4\pi$)

Figure 7: Volume renders of synthetic datasets generated with the *IvanTk* Synthetic Module

https://github.com/imacia/ivantk

An initial project page with future releases and intermediate snapshots can be found at

https://sourceforge.net/projects/ivantk/

The build process requires CMake 2.8 or higher and ITK 4.0 or higher. The use of VTK (5.0 or higher) is optional. The installation is straightforward using CMake.

Doxygen (www.doxygen.org) documentation can be generated automatically from CMake. See the README file for details. A mailing list has also been created at

https://lists.sourceforge.net/lists/listinfo/ivantk-users

6 Development Status

The development of *IvanTk* is at an initial stage requiring more extensive validation and testing. Still, most of the implementations have been tested successfully.

The libraries have been designed with the aim of comparing detection and extraction algorithms in similar conditions, with more emphasis on accuracy and versatility. There is room for improvements in speed, since some of the implementations, specially those involving 3D Hessian matrices, are slow.

Most of the vesselness algorithms are oriented to provide results for a sparse number of points. Hence, they are implemented as `itk::ImageFunction` subclasses. However, vesselness functions calculated densely for the whole volume, such as those already existing in ITK as filters, can be used by the extraction algorithms, in the Vesselness Metric or in the Section Estimator among others.

Currently, the *Extraction Module* only considers tracking with some basic components. Some modifications are required to filter trajectories and to incorporate bifurcation detectors.

To sum up, the libraries are in a experimental state, but we think that the concepts behind may be useful for the medical imaging community, which may provide the necessary feedback in order to make the toolkit as useful as possible. Thus, it is expected that interfaces and names may be subject to substantial change at these initial steps.

7 Conclusions

We have developed the *IvanTk* open source libraries for vascular image analysis. It implements algorithms for vascular detection and extraction in an object-oriented manner, providing a means of comparing their perfomance, offering versatility for testing different components and parameters of the methods, and providing a ground for future extensions and implementation of new algorithms. The library pretends to be compatible and complementary to other related existing initiatives.

Future work will involve improving the robustness, speed and versatility of the library, incorporating new detection algorithms and implementing additional vascular extraction schemes.

References

[1] A.F. Frangi, W.J. Niessen, K.L. Vincken, and M.A. Viergever. Multiscale vessel enhancement filtering. In *Medical Image Computing and Computer-Assisted Interventation (MICCAI)*, pages 130–137. Springer-Verlag, 1998.

[2] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide 2nd Ed.* Kitware, Inc. ISBN 1-930934-15-7, 2005.

[3] T. M. Koller, G. Gerig, G. Székely, and D. Dettwiler. Multiscale detection of curvilinear structures in 2-d and 3-d image data. In *In Proc. Fifth International Conference on Computer Vision (ICCV'95)*, pages 864–9. IEEE Computer Society Press, 1995.

[4] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, and Y. Trousset. Model based detection of tubular structures in 3d images. *Computer Vision and Image Understanding*, 80(2):130–171, 2000.

[5] M.W.K. Law and A.C.S. Chung. Three dimensional curvilinear structure detection using optimally oriented flux. In P. Torr D. Forsyth and A. Zisserman, editors, *In Proc. European Conference of Computer Vision (ECCV'08)*, volume LNCS 5305, pages 368–82, 2008.

[6] D. Lesage, E.D. Angelini, I. Bloch, and G. Funka-Lea. Design and study of flux-based features for 3d vascular tracking. In *In Proc. of Int. Symp. Biomedical Imaging (ISBI)*, pages 286–9, 2009.

[7] D. Lesage, E.D. Angelini, I. Bloch, and G. Funka-Lea. A review of 3d vessel lumen segmentation techniques: Models, features and extraction schemes. *Medical Image Analysis*, 13(6):819–845, December 2009.

[8] I. Macia, M. Graña, and C. Paloc. Knowledge management in image-based analysis of blood vessel structures. *Knowledge and Information Systems*, 30(2):457–491, 2012.

[9] Ivan Macia. *Medical Image Analysis for the Detection, Extraction and Modelling of Vascular Structures*. PhD thesis, University of the Basque Country (UPV/EHU), 2012.

[10] Xiaoning Qian, Matthew P. Brennan, Donald P. Dione, Wawrzyniec L. Dobrucki, Marcel Jackowski, Christopher K. Breuer, Albert J. Sinusas, and Xenophon Papademetris. A non-parametric vessel detection method for complex vascular structures. *Medical Image Analysis*, 13(1):49–61, 2009.

[11] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, and R. Kikinis. 3d multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical Image Analysis*, 2(2):143–168, June 1998.