Integration of the Vascular Modeling Toolkit in 3D Slicer

Student Research Project

Daniel Hähn
Student of Medical Informatics
University of Heidelberg

04/20/09

Updated 09/27/09

Supervisors:
Ron Kikinis, M.D.
Steve Pieper, Ph.D.

Surgical Planning Laboratory
Brigham and Women's Hospital
Harvard Medical School
Boston, MA, USA

Luca Antiga, Ph.D.

Medical Imaging Unit
Bioengineering Department
Mario Negri Institute
Ranica (BG), Italy

Responsible Professor:
Prof. Dr. Hartmut Dickhaus

Institute for Medical Biometry and Informatics
Department of Medical Informatics
University of Heidelberg
Germany

Abstract

The extraction of vessels in two- and three-dimensional images is part of many clinical analysis tasks. Surgical and radiology procedures often involve the visualization and quantification of vessels in order to perform surgical planning or diagnostics. There is no single segmentation method that can extract vessels from every medical image modality, but different approaches and robust algorithms exist. Various published key algorithms are available within an open-source framework for image-based modeling of blood vessels, referred to as the Vascular Modeling Toolkit (VMTK).

The library of VMTK was made available in 3D Slicer, an application providing a wide range of tools for medical image processing. This was realized using a hidden loadable module approach in order to provide a flexible way of distributing and including the library. To evaluate and verify the integration, a software module offering VMTK level set segmentation methods within 3D Slicer was created.

With the successful connection of the two above mentioned software solutions, processing pipelines between VMTK code and other algorithms can be established. Several techniques for three dimensional reconstruction, geometric analysis, mesh generation and surface data analysis for image-based modeling of blood vessels are now accessible to the 3D Slicer developer. The reference implementation for accessing VMTK, as well as the created library module are available as open-source software.

The research presented in this paper was made possible by a charitable grant of the Thomas-Gessmann Foundation part of the Founder Federation for German Science.

Table of Contents

1 Introduction

Surgical and radiology procedures may involve the visualization and quantification of vessels as clinical analysis tasks. In order to perform surgical planning or diagnostics, vessels in two- or three-dimensional images are often extracted. This task is called vessel or vascular segmentation. [Aylward2002]

Since there is no single segmentation method that can extract vessels or other tubular shaped objects from every medical image modality, a variety of different vessel extraction techniques and algorithms exist [Kirbas2004]. These algorithms follow different approaches which Kirbas and Quek (2004) used to divide into six main categories: (1) pattern recognition techniques, (2) model-based approaches, (3) tracking-based approaches, (4) artificial intelligence-based approaches, (5) neural network-based approaches, and (6) tube-like object detection approaches. One goal of this student research project was to find and choose a suitable method to integrate within 3D Slicer (Version 3)[1], an application which provides a wide range of tools and utilities concerning medical image processing. This application was chosen because it is widely accepted, open-source and a platform independent software solution.

During the analysis of the state-of-the-art methods for vessel and tubular shaped objects segmentation, the Vascular Modeling Toolkit (VMTK)[2] was discovered. VMTK is an open-source framework providing tools for image-based modeling of blood vessels.

Since VMTK offers various segmentation methods, the integration with 3D Slicer appeared to be more promising than just integrating one specific segmentation method. 3D Slicer is a “point-and-click” end user application, therefore the usability enhancement using 3D Slicer modules rather than the console-based VMTK should be an additional option presented in this student research project.

In addition to the back-end encapsulation of the complete VMTK framework, another goal of this project was to create an interactive software module. This module provides specific VMTK functionality in 3D Slicer as a reference implementation to evaluate and verify the integration. The task of vessel segmentation based on level sets was chosen for this purpose. Level set segmentation is a deformable model-based approach where the surface of the segmented object is described by a three-dimensional function [Kirbas2004].

In comparison to previous work on vessel segmentation in 3D Slicer, this student research project differs because it utilizes the powerful VMTK framework as a back-end and the strong end user application 3D Slicer as a front-end. The challenges of connecting these two complex software solutions are described in this paper, as well as the level set segmentation module itself.

The information presented in this project report will be structured as follows. Chapter two describes the software and frameworks used. Chapter three focuses on the methodologies of this work and gives a brief introduction to level-set segmentation in general and the way it is provided by VMTK. Also a way of manually integrating the VMTK libraries in 3D Slicer is shown. It is followed by chapter four which illustrates the major results including the design and implementation of the created software modules. Throughout the development of the modules, feedback was provided from the developers of VMTK. The Discussion and Outlook chapter deals with the available opportunities of the VMTK integration and also with possible further enhancements of the created level set segmentation module.

1 3D Slicer is available at http://www.slicer.org

2 The Vascular Modeling Toolkit is available at http://www.vmtk.org

2 Fundamentals

The following software and frameworks were used during this student research project.

Python

Python is a widely used dynamic programming language [Pilgrim2009].

The software and frameworks used in this student research project take advantage of the possibility that Python is embeddable within applications as a scripting interface.

VTK/ITK

The Visualization Toolkit (VTK) is “an open-source, freely available software system for 3D computer graphics, image processing, and visualization used by thousands of researchers and developers around the world. VTK consists of a C++ class library and several interpreted interface layers including Tcl/Tk, Java, and Python. [...] VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods; and advanced modeling techniques such as implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation.” [VTK2009]

The Insight Segmentation and Registration Toolkit (ITK) is “an open-source, cross-platform system that provides developers with an extensive suite of software tools for image analysis. [...] ITK employs leading-edge algorithms for registering and segmenting multidimensional data.” [ITK2009]

While ITK is a framework for image processing and does not provide visualization methods, the visualization component is often realized using OpenGL[3] or VTK [ITKVTK2009]. The applications used in this student research project heavily rely on the combination of ITK and VTK.

ITK and VTK are part of “the NA-MIC kit”[4] and are available as platform independent software packages[5].

CMake

CMake is a build-system which uses compiler and operating system independent configuration files. Those configuration files are named CMakeLists.txt and are defined using its own language, the CMake configuration language. The actual build process is still performed by native build tools but the advantage of CMake is that the configuration of these tools is much easier. [Cmake2009]

CMake is also part of “the NA-MIC kit” and available as a platform independent software package[6]. VMTK, as well as 3D Slicer, use CMake as the build system which is important in their connection.

---

3 OpenGL (for “Open Graphics Library”) is a standard software interface to graphics hardware and is widely used. [OpenGL2009]

4 The NA-MIC kit is a free open-source platform and consists of several tools and frameworks for visualization, rendering and image manipulation but also for the support of software development in general. The kit is provided by the National Alliance of Medical Imaging Computing (NA-MIC). [NAMICkit2009]

5 VTK and ITK are available at http://www.vtk.org and http://www.itk.org

6 CMake is available at http://www.cmake.org

VMTK

The Vascular Modeling Toolkit (VMTK) is “an open-source framework for image segmentation, geometric characterization, mesh generation and computational hemodynamics specifically developed for the analysis of vascular structures” [Piccinelli2009].

VMTK provides a collection of libraries and tools for image-based modeling of blood vessels.

This includes several scripts which provide high-level functionality and are available as Python classes. These scripts are able to interact with each other using the integrated PypeS (Python pipe-able scripts) framework. The actual algorithms implemented in C++ classes are based on VTK and ITK code and have been published in medical imaging journals. [VMTK2009]

A list of the features available in VMTK can be found in *Appendix A*.

VMTK is actively developed and maintained by two researchers[7]. The toolkit is available as an open-source and platform independent package[8].

KWWidgets

KWWidgets is an open-source library of GUI[9] widgets. The provided widgets are low-level core widgets (like buttons, entries, scales and lists), composite widgets (like toolbars, gauges, histograms, windows and dialogs) and high-level visualization-oriented widgets (like surface material editors, simple animation generators and window/level preset editors) which interface to VTK. Beside providing an object-orientated C++ layer on top of the Tcl/Tk UI toolkit[10], KWWidgets are wrapped automatically into a Tcl[11] package or a Python module and therefore can be directly used from Tcl or Python. [Kwwidgets2009]

This library is also part of “the NA-MIC kit” and is available within a source code repository[12].

3D Slicer

3D Slicer is “a free, open-source software package for visualization and image analysis. 3D Slicer is natively designed to be available on multiple platforms” [Slicer2009]. The application was originally developed by the Surgical Planning Laboratory (SPL) at the Brigham and Women's Hospital and the MIT Artificial Intelligence Laboratory in 1998.

Several past and present projects prove that 3D Slicer is an actively developed platform involved in daily medical imaging research worldwide.[13]

Based on a Model View Controller pattern (see *Appendix B*) and the Medical Reality Modeling

---

7 Developers of VMTK are: Luca Antiga (Medical Imaging Unit, Bioengineering Department, Mario Negri Institute, Bergamo, Italy) and David Steinman (Biomedical Simulation Lab, Mechanical & Industrial Engineering, University of Toronto, Ontario, Canada)

8 Release packages as well as the current development version of VMTK are available at http://www.vmtk.org/Main/Download

9 GUI: graphical user interface

10 The Tk UI toolkit is “the standard GUI for Tcl and other dynamic languages” [Tcl2009].

11 Tcl (Tool Command Language) is a popular dynamic programming language and available as open-source software. [Tcl2009]

12 The KWWidgets toolkit is available at http://www.kwwidgets.org

13 An overview of 3D Slicer Enabled Research is available at http://slicer.org/pages/Slicer_Community

Language (MRML)[14], the functionality of 3D Slicer can be extended using modules.

3D Slicer is also part of “the NA-MIC kit” and is available as a platform independent software package.

14 MRML is a XML based data format to describe 3D scenes including several different types of medical imaging data and their visualization. [Mrml2009]

3 Methodologies

The methodologies of this student research project are outlined in this chapter. A brief information of general level set segmentation is given first. Followed by a documentation of an example segmentation using plain VMTK and an analysis of the VMTK level set segmentation work flow. Also a way of manually integrating the VMTK libraries in 3D Slicer is shown.

Level Set Segmentation

Level set segmentation is a geometric deformable model approach which is based on the use of the Level Set Method developed by Osher and Sethian in 1988. [Kirbas2004]

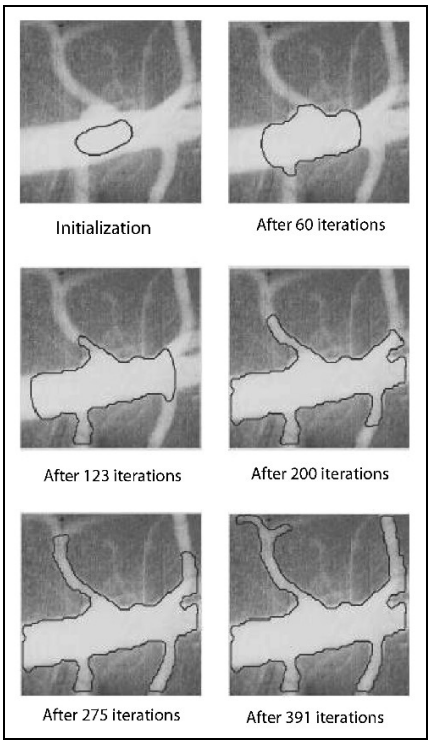


Illustration 3.1: Initialization and Evolution steps [Kirbas2004, Figure 8]

For further investigating the involved mathematical background of the mentioned Level Set Method, see the detailed description in “Level Set Methods and Fast Marching Methods” by J.A. Sethian.

Segmentation using level sets consists of an initialization and an evolution step. The initialization step involves the description of a starting model within a region of interest. In the evolution step this initial deformable model then gets inflated to match the contours of the targeted volume. A good initialization is an important progress towards the segmentation of a vascular branch. [Vmtktut2009]

Illustration 3.1 shows the initialization and evolution principles.

An Example Level Set Segmentation using VMTK

When using VMTK for level set segmentation, the console based Python script *vmtklevelsetsegmentation* must be started. In addition to the required input and output volumes, other optional parameters may be passed as command line arguments. A rendering window showing the input volume is opened. The required interaction takes place in either the rendering window or the console.

An example of the level set segmentation process is displayed in Illustration 3.2. The actual state captured shows a performed evolution after an initialization on a single vascular branch as a translucent model inside the rendering window.

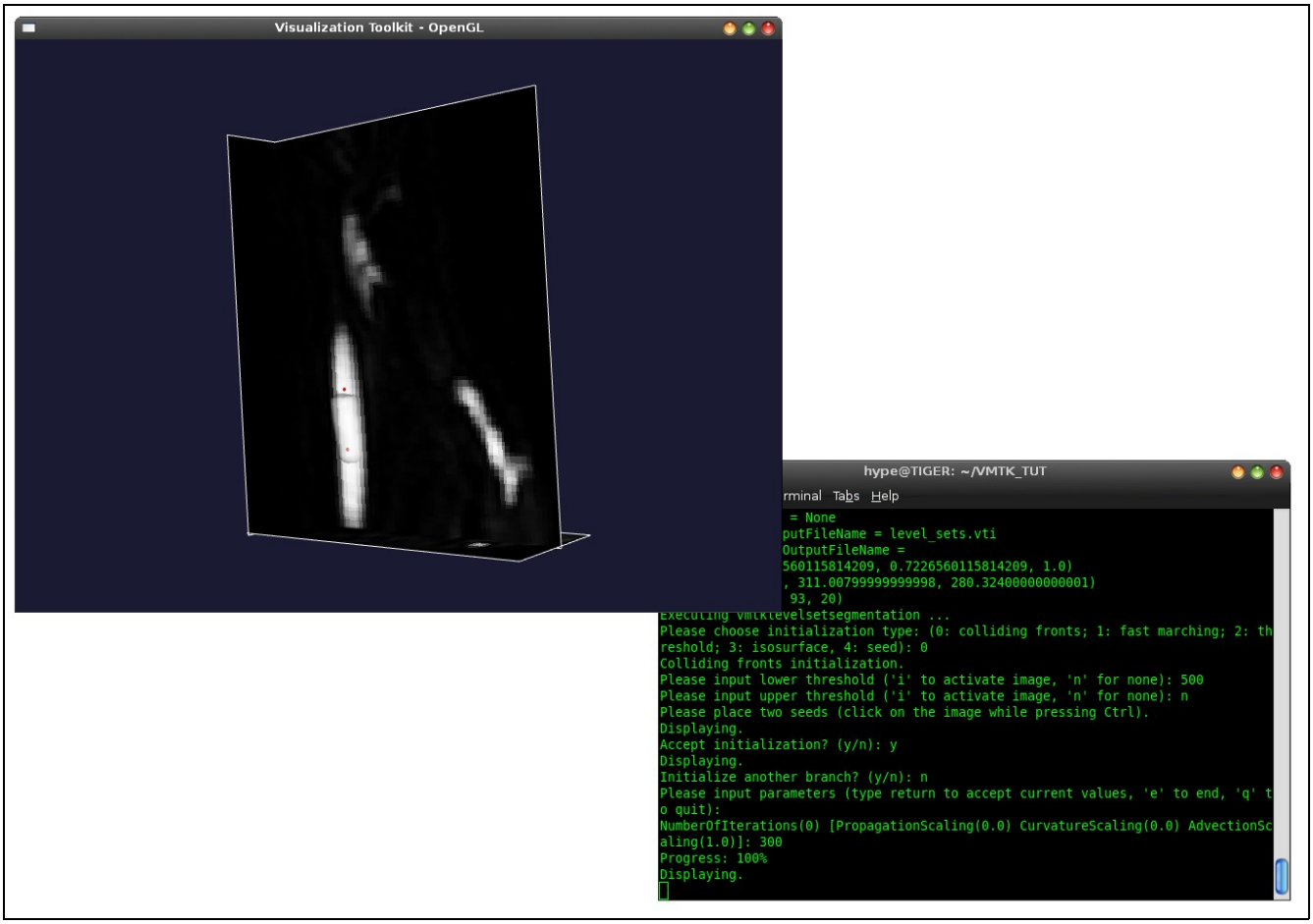


Illustration 3.2: Example level set segmentation using VMTK

The user interaction was performed by keyboard in the console as well as in the rendering window, and by using the mouse in the rendering window for placing seed points.

The VMTK Level Set Segmentation Work flow

In the level set segmentation process of VMTK the user first initializes using a chosen method. After initialization, the segmented volume is generated and displayed as a 3D model. Then the user decides whether to directly accept the results and continue to evolution stage, add another branch or undo the last action. If the initialization is accepted, then the user chooses an evolution method and applies. The

result is an inflated volume which is displayed as a 3D model.

This is followed by the final options which are accepting this evolution, undoing it, or adding another branch by re-initializing. The process is shown as a flow chart in Illustration 3.3.

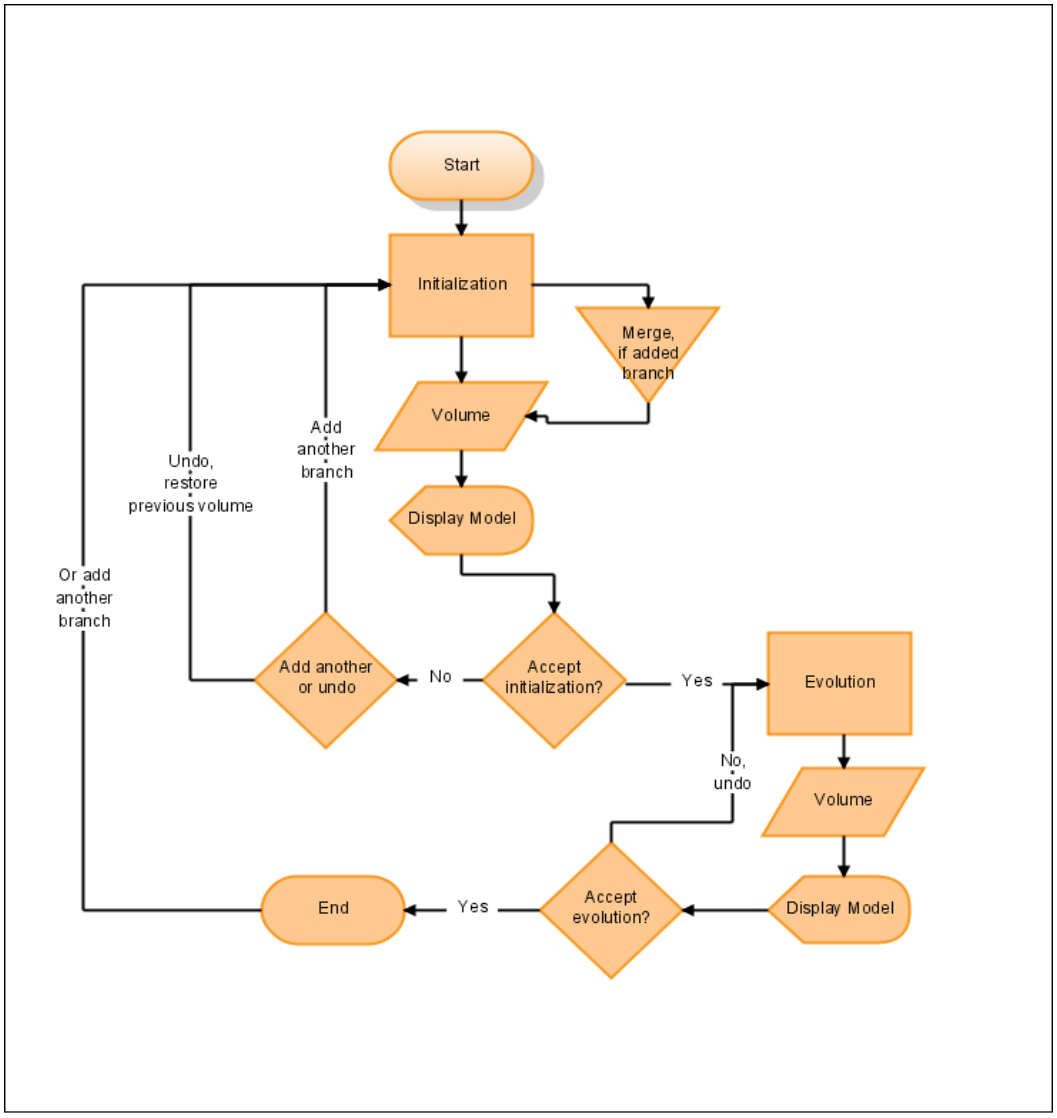


Illustration 3.3: Self created flow chart showing the level set segmentation process (simplified)

Manual Integration of the VMTK Base in 3D Slicer

When the Vascular Modeling Toolkit is downloaded the package shown in Illustration 3.4 is obtained. The algorithms of VMTK are C++ classes inside the folder *vtkVmtk/* and are referred to as the Vascular Modeling Toolkit libraries.

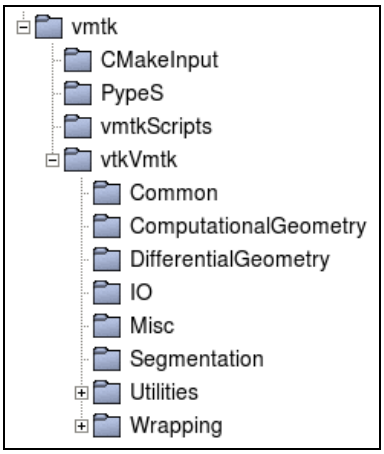


Illustration 3.4: Self created illustration of the VMTK package structure

To provide the desired functionality of the Vascular Modeling Toolkit, the VMTK libraries must be integrated into 3D Slicer.

In the early research stage, this integration was done in a user unfriendly way. Manual changes of the main method in *Slicer3.cxx* had to be performed after copying the *vtkVmtk/* folder into the 3D Slicer library directory *Slicer3/Libs* (see Code 1). These lines connect the VMTK libraries with 3D Slicer by calling the automatically generated[15] Tcl initialization classes.

Additionally, the *CMakeLists.txt* file in the 3D Slicer library directory had to be modified to include the copied VMTK libraries during compilation of 3D Slicer.

```
// [...]

extern "C" int Vtkvmtkcommontcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtkcomputationalgeometrytcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtkdifferentialgeometrytcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtkiotcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtkmisctcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtksegmentationtcl_Init(Tcl_Interp *interp);

// [...]

Vtkvmtkcommontcl_Init(interp);
Vtkvmtkcomputationalgeometrytcl_Init(interp);
Vtkvmtkdifferentialgeometrytcl_Init(interp);
Vtkvmtkiotcl_Init(interp);
Vtkvmtkmisctcl_Init(interp);
Vtkvmtksegmentationtcl_Init(interp);

// [...]
```

Code 1: TCL Initialization of the VMTK library, added manually to Slicer3.cxx

The VMTK libraries consist of several sub-directories holding packages of algorithms. Since VMTK uses CMake as a build system, each sub-folder includes its own CMake configuration file. When compiling 3D Slicer, these *CMakeLists.txt* files get parsed recursively and include the complete folder tree of the VMTK libraries to the build.

Nevertheless, this kind of integration was neither user friendly nor acceptable, therefore a different way had to be found.

15 The wrapped Tcl classes get automatically generated by CMake.

4 Results

The following section presents the results of this student research project. First the integration of the VMTK libraries using a GUI-less module is described, followed by a detailed characterization of the level set segmentation module as a reference implementation of VMTK in 3D Slicer. Then several changes to the 3D Slicer base are documented.

GUI-less Module based Integration of the VMTK Base in 3D Slicer

To omit the required manual changes of the 3D Slicer base, it was decided to add extra functionality to the 3D Slicer infrastructure enabling GUI-less loadable modules[16]. In comparison to the existing loadable module interface, which requires a GUI-class and integrates itself into the 3D Slicer menu, the GUI-less loadable modules should be hidden within the 3D Slicer application. The necessary changes to implement the infrastructure for GUI-less loadable modules were realized [Slicer_SVN, rev. 8770][17]. Using this provided infrastructure, it was possible to create a GUI-less loadable module for VMTK in 3D Slicer named *VmtkSlicerModule*, which consists of the following structure:

vtkVmtk/
The directory which holds all VMTK libraries.

CMakeLists.txt
The CMake build file, which also includes the VMTK library directory for compilation.

VmtkSlicerModule.xml
The required XML description of the module.

vtkVmtkSlicerModuleLogic.cxx
The required logic class of the module to initialize using the TCL mapping which were included manually in *Slicer3.cxx* before (see Code 3).

vtkVmtkSlicerModuleLogic.h
The header file of the logic class to describe the implemented methods.

Other required files exist but can be considered as stubs because no real content was added.

```
[...]

subdirs(
  vtkVmtk

)

[...]

set(libs ${libs} vtkvmtkCommonTCL vtkvmtkComputationalGeometryTCL
    vtkvmtkDifferentialGeometryTCL vtkvmtkIOTCL vtkvmtkMiscTCL vtkvmtkSegmentationTCL)

target_link_libraries(${lib_name}
  ${Slicer3_Libs_LIBRARIES}
  ${Slicer3_Base_LIBRARIES}
  ${KWWidgets_LIBRARIES}
  ${ITK_LIBRARIES}
  ${libs}
  )

[...]
```

Code 2: Extracts of the CMakeLists.txt file of VmtkSlicerModule

16 A loadable module can be dynamically loaded during the 3D Slicer startup process. The user has the option to select which modules are used in 3D Slicer.

17 Acknowledgment: Support for GUI-less loadable modules was integrated into 3D Slicer by Terry G. Lorber II and Steve Pieper, Isomics, Inc., Cambridge, MA, USA

The *CMakeLists.txt* file of the GUI-less loadable module includes the sub-directory holding the VMTK libraries (see Code 2).

```
// [...]

#include "vtkSlicerApplication.h"
#include "vtkTcl.h" // Needed for Tcl_Interp

extern "C" int Vtkvmtkcommontcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtkcomputationalgeometrytcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtkdifferentialgeometrytcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtkiotcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtkmisctcl_Init(Tcl_Interp *interp);
extern "C" int Vtkvmtksegmentationtcl_Init(Tcl_Interp *interp);

// [...]

vtkVmtkSlicerModuleLogic::vtkVmtkSlicerModuleLogic()
{

  Tcl_Interp *interp = NULL;
  vtkSlicerApplication *slicerApp = vtkSlicerApplication::GetInstance ( );
  interp = slicerApp->GetMainInterp();
  if (!interp)
    {
    cout << "Error: InitializeTcl failed" << endl;
    }

  Vtkvmtkcommontcl_Init(interp);
  Vtkvmtkcomputationalgeometrytcl_Init(interp);
  Vtkvmtkdifferentialgeometrytcl_Init(interp);
  Vtkvmtkiotcl_Init(interp);
  Vtkvmtkmisctcl_Init(interp);
  Vtkvmtksegmentationtcl_Init(interp);

}

// [...]
```

Code 3: TCL Initialization of the VMTK library, added to vtkVmtkSlicerModuleLogic.cxx

An error concerning a specific path of the VMTK libraries was fixed by adding the following line to CMakeLists.txt [Slicervmtklvlst_SVN, rev. 34]:

```
set(TETGEN_SOURCE_DIR ${CMAKE_CURRENT_SOURCE_DIR}/vtkVmtk/Utilities/tetgen1.4.2)
```

Nevertheless, this error only appeared on some testing machines and was not reproducible.

The module depends on the VMTK libraries which are actively developed and get extended constantly. To solve the problem of the backwardness of these libraries, a solution was found using SVN property settings. The library folder is integrated using a *svn:external*[18] repository to enable fetching of the latest VMTK version during a checkout of *VmtkSlicerModule*.

After a SVN checkout, the module can be installed in the same way as other 3D Slicer loadable modules by copying it to *Slicer3/Modules/* and adding it to the build process.

Additionally, a **3D Slicer Extension Build System**[19] file (.s3ext-file) was created to describe this GUI-

18 *svn:external* is a SVN property which enables the connection of different SVN repositories (http://svnbook.red-bean.com/en/1.0/ch07s03.html).

19 The 3D Slicer Extension Build System works as a dynamic repository to allow end users to easily download and install modules, but is still in development.

less loadable module and provide end users a comfortable way of installing the latest version of the VMTK library within 3D Slicer in the future.

Using VMTK in 3D Slicer

By installing the GUI-less module *VmtkSlicerModule* all objects and functions of the Vascular Modeling Toolkit libraries are available within 3D Slicer.

Illustration 4.1 shows how to access the Fast Marching [Sethian1999] implementation of VMTK using the 3D Slicer Python console.

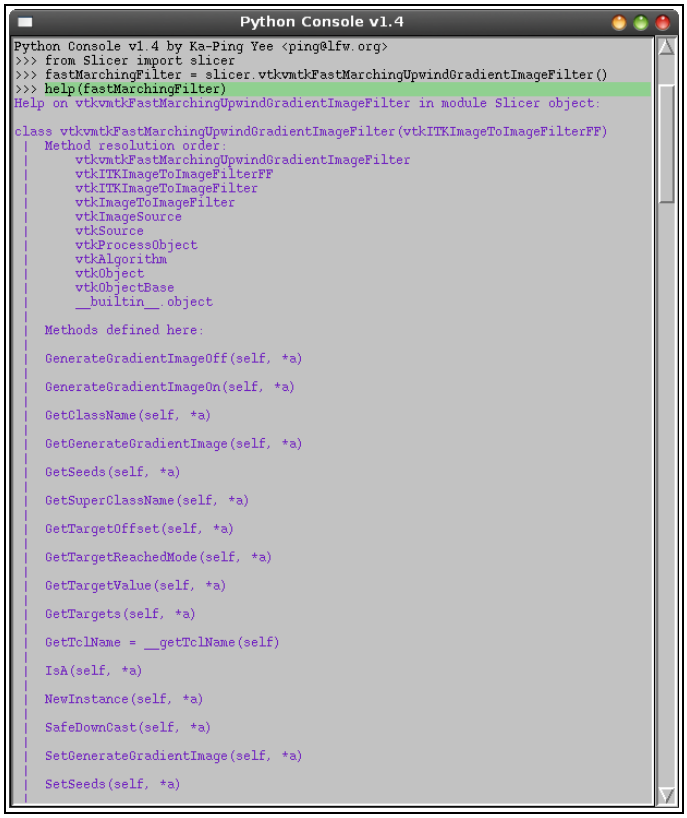


Illustration 4.1: Accessing VMTK using the 3D Slicer Python console

The same filter can be created using the Tcl console as seen in Illustration 4.2.

Beside using the consoles for quick access to the VMTK libraries, all available functions can be called by any 3D Slicer modules.

A reference implementation of a module mapping the level set segmentation process of VMTK to 3D Slicer is presented in the next section.

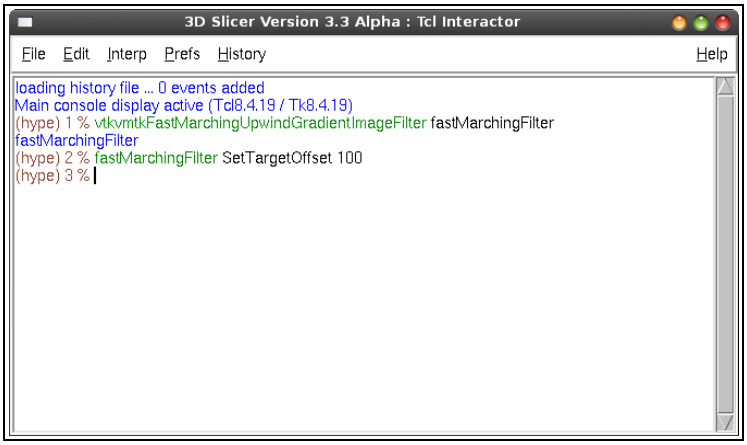


Illustration 4.2: Accessing VMTK using the 3D Slicer Tcl console

Design of the created Level Set Segmentation Module

The following section describes the design and the work flow of the created level set segmentation module which can be seen as a reference implementation of VMTK functionality in 3D Slicer.

When using VMTK for level set segmentation, special user interaction like placing seed points is involved. To realize this functionality within 3D Slicer, a *Scripted Module* approach based on Python was chosen. *Scripted Modules* written in scripting languages like Python or Tcl are able to provide the same functionality as loadable modules. This grants access to 3D Slicer internals, as well as the interactions which are needed to implement the level set segmentation [Antiga2009, pp. 25-27].

Class Design

Since the level set segmentation was realized as a 3D Slicer *Scripted Module*, it follows the conventions of the Model View Controller pattern of slicer modules (see *Appendix B*). This implies the separation of logic and GUI. Beside the advantage that the code is more readable, the isolation of the logic offers an ability to change the invoking GUI (e.g. to a wizard based interface) or write tests for the logic.

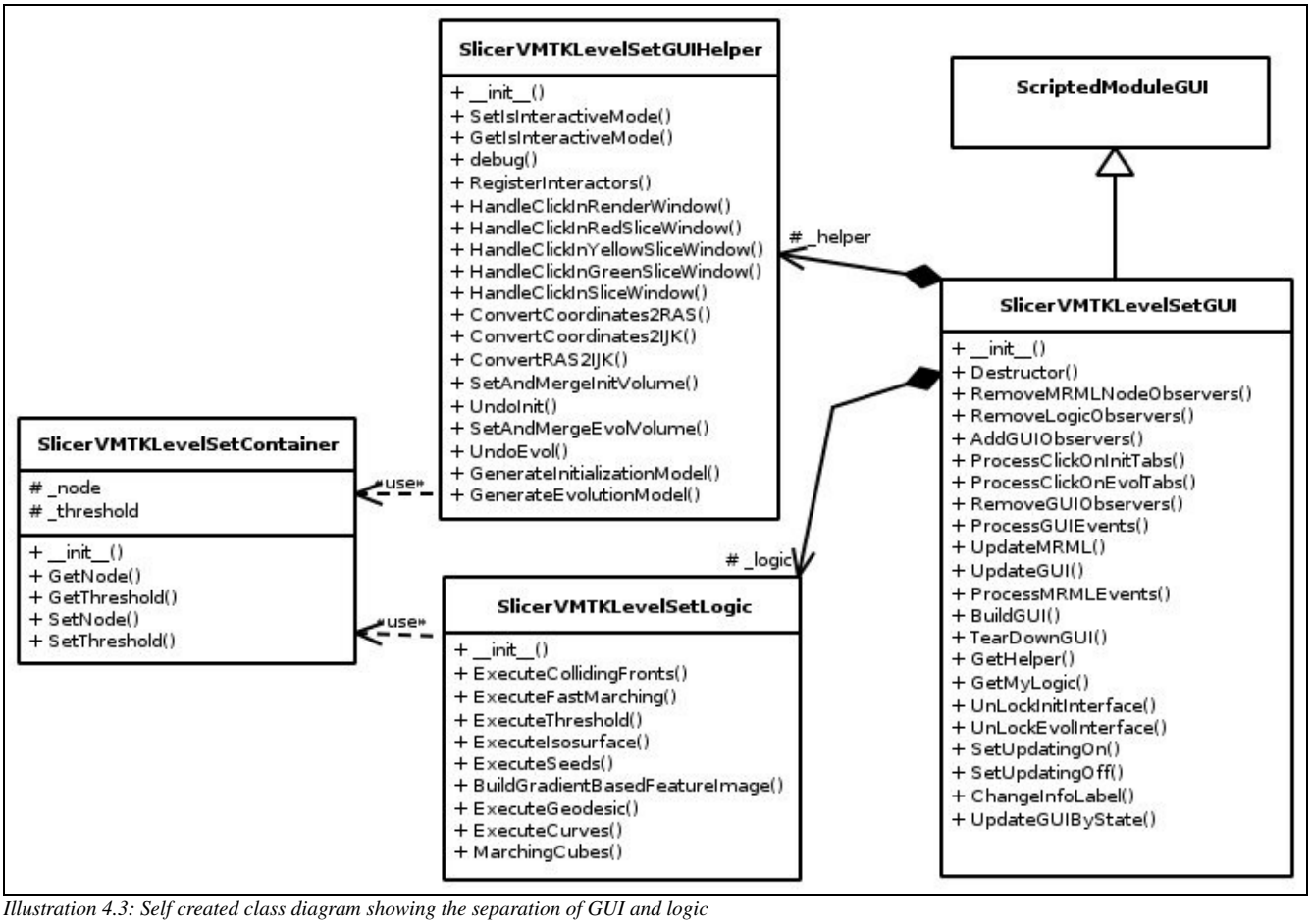


Illustration 4.3: Self created class diagram showing the separation of GUI and logic

SlicerVMTKLevelSetGUI derives from *ScriptedModuleGUI* and saves the current user interface state to its own MRML node. The MRML nodes of the used volume and model data[20] are additionally attached. This allows different instances of the level set segmentation module to be run at the same time. To

20 The used MRML nodes are *vtkMRMLScalarVolumeNode*, *vtkMRMLModelNode* and *vtkMRMLModelDisplayNode*.

ensure this functionality, certain methods that update the GUI and the MRML node have to be implemented[21]. Additional methods must also exist to process events fired from the GUI.

The actual calls to the VMTK libraries are only performed in *SlicerVMTKLevelSetLogic* and are documented on page 21.

Several general supporting functions, like handling interaction within 3D Slicer, converting coordinates, undoing and the merging of volumes are outsourced in the class *SlicerVMTKLevelSetGUIHelper*.

Not only does every initialization and evolution method of the level set segmentation module use its own GUI and widgets but it also provides individual user interaction within 3D Slicer. Seperate classes derived from the interface *SlicerVMTKAdvancedPageSkeleton* were created for each method in order to ensure the possibility of maintenance and extension. The calls to the required methods mentioned above get forwarded to each individual sub-class to support the modular design and enable code encapsulation.

The following illustration shows the link between the initialization methods.

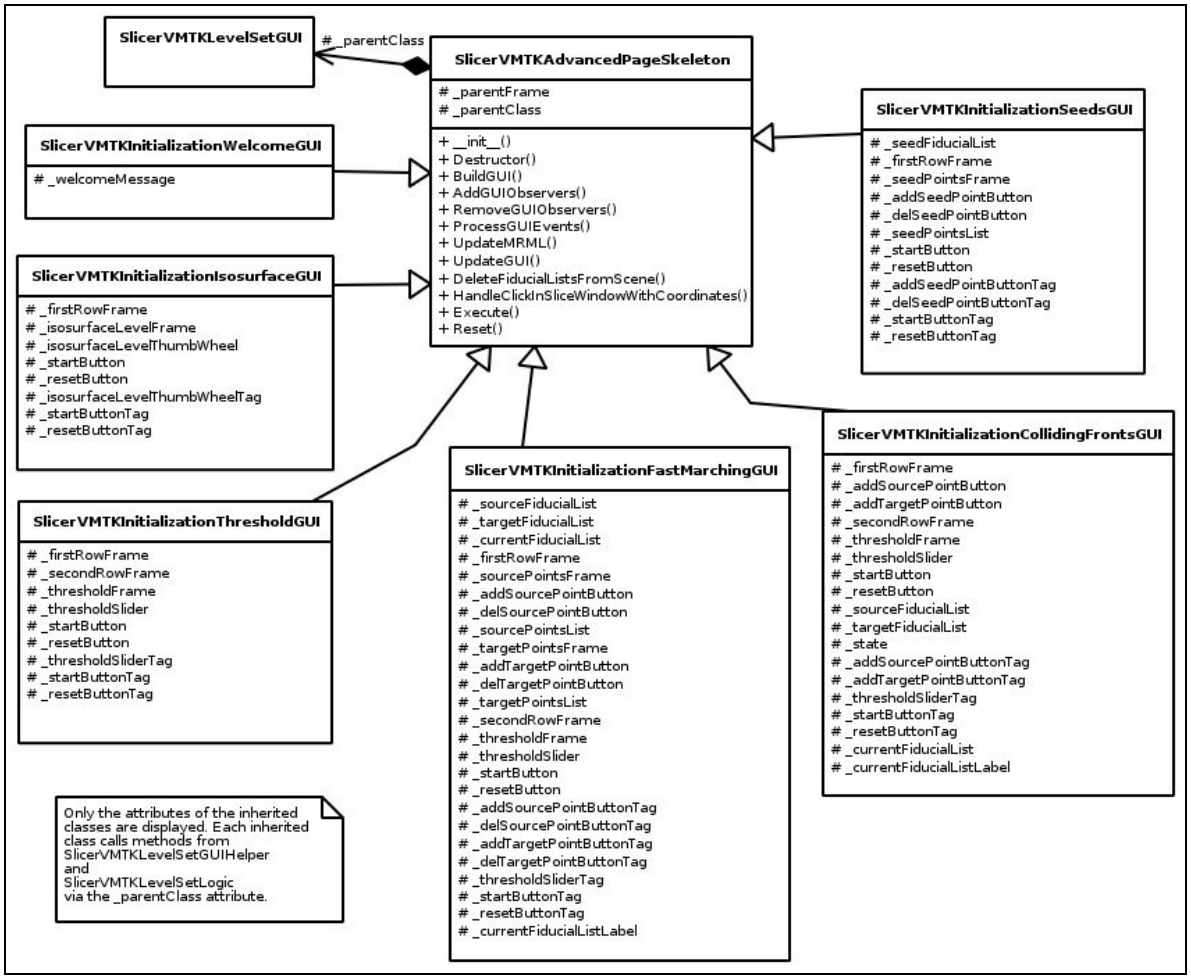


Illustration 4.4: Self created class diagram showing the individual classes of the initialization methods

A comparable diagram of classes used for the evolution methods can be found in *Appendix C*.

21 The required virtual methods must be implemented using overriding [Slicer2009c].

Graphical User Interface

The challenging part during the design of the graphical user interface was to exactly map the VMTK level set segmentation work flow including required parameters and interaction styles to a 3D Slicer module in a suitable way.

Within the top frame of the module two combo-boxes are visible. The first combo-box switches between different instances of the level set segmentation module. This includes loading of connected parameters and the actual state (equals the MRML node of the *Scripted Module*). The second combo-box is used for selecting the input volume on which the segmentation is performed. After changing it automatically gets displayed in the 3D Slicer slice viewers.

Additionally a status label is displayed to inform the user of current input possibilities or the actual state of the module.

The bottom part of the GUI is generally divided by two frames – one for the initialization algorithms and one for the evolution algorithms. Each frame consists of tabbed pages to select the individual methods. For convenience, the first tabbed page shows an overview of the available techniques and provides a short description. Since initialization and evolution methods can be performed several times, action buttons to undo, add another branch or accept the current segmentation are included underneath the frames. The following self created screen shots give a selective overview of the look and feel of the module. Additional screen shots can be found in *Appendix E*.

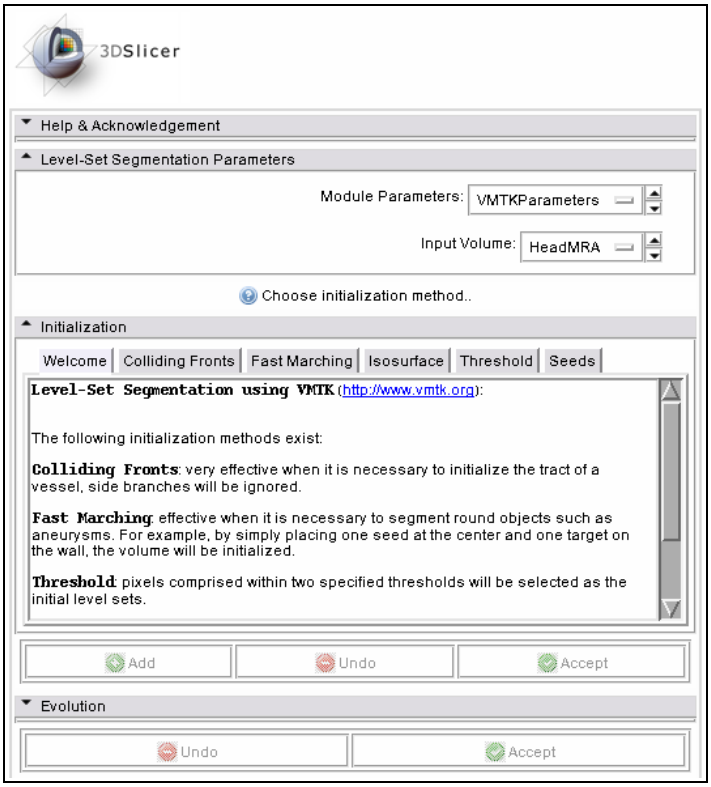


Illustration 4.5: The level set segmentation module, ready to choose an initialization method.

In addition to widgets for individual parameters, *Start* and *Cancel* buttons are always displayed in the

tabbed pages of initialization and evolution methods. These start the execution of the selected algorithm if the required parameters or information are set, or reset the selected widgets to their default values. The following illustrations show the GUI components of algorithms which get further described later in this document (see page 21).

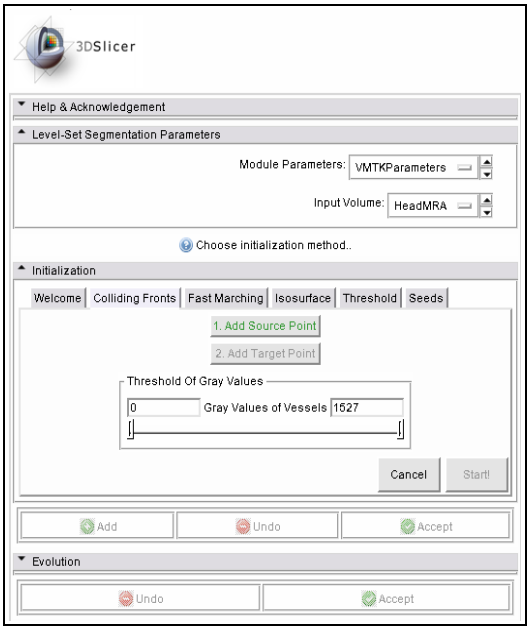


Illustration 4.6: The Colliding Fronts GUI, waiting for user input.

Illustration 4.6 shows the graphical user interface of the Colliding Fronts initialization method, waiting for user input. Since the required parameters (see page 21) are not configured yet, the execution is disabled.

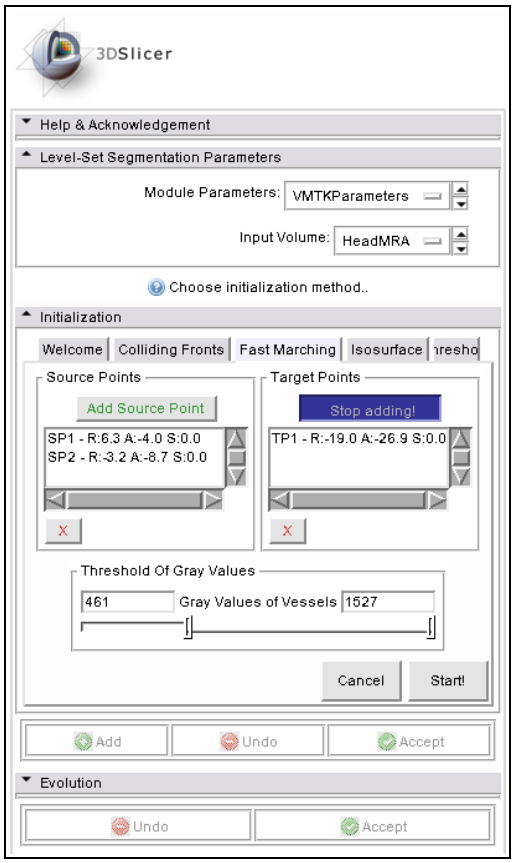


Illustration 4.7: The Fast Marching GUI, ready to run.

The Fast Marching GUI is shown in Illustration 4.7. Because source points, target points and a threshold of gray values are already configured, all needed parameters are set (see page 21) and the algorithm is ready to execute.

Coordinates of the configured fiducial values (source- and target points) are shown within the lists.

The blue "*Stop Adding!*" button implies that the interactive mode of selecting target points is still activated. Thus, the module would also accept another fiducial by clicking into the slice viewers beside starting the execution.

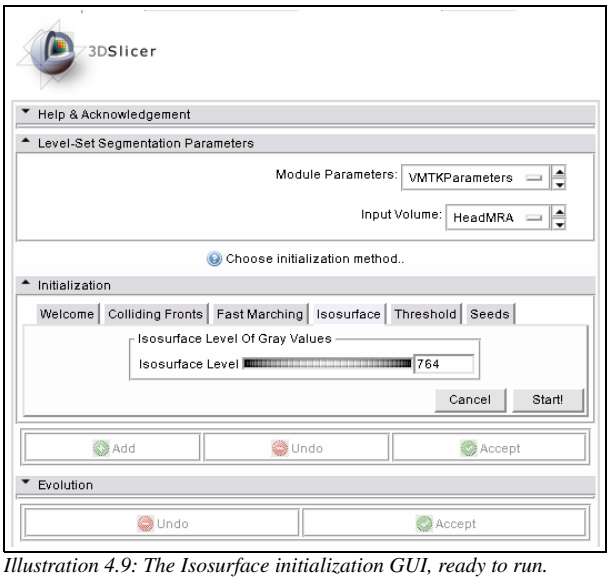


Illustration 4.9: The Isosurface initialization GUI, ready to run.

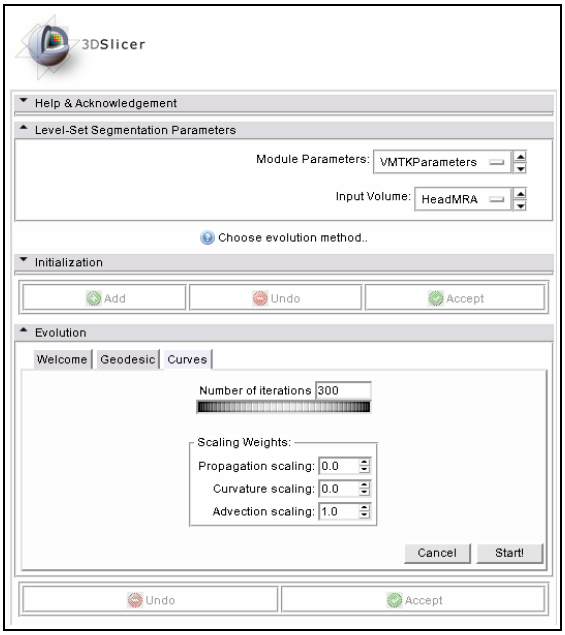


Illustration 4.8: Curves evolution method, ready to run.

Illustration 4.8 and 4.9 show the Isosurface initialization and the Curves evolution methods. Both are ready to run because the required parameters (see page 21) are either already set by the user or are default values.

The challenge of mapping the VMTK level set segmentation work flow to a graphical user interface was already mentioned. To fulfill this purpose, action buttons were added after the initialization and the evolution frames. These provide the ability to trigger segmenting another branch of vessels and undoing or accepting the current segmentation result. The following figure shows the action buttons after one initialization step was successfully performed.

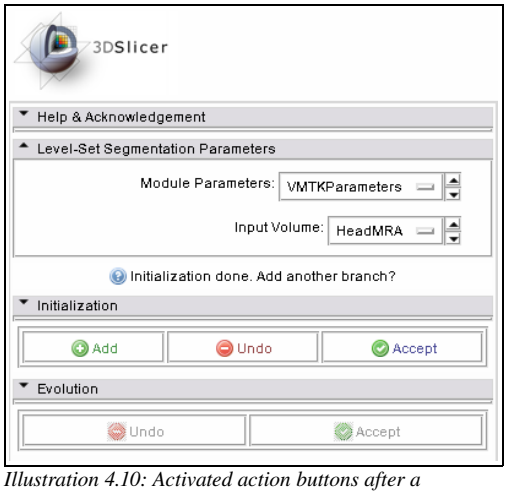


Illustration 4.10: Activated action buttons after a successful initialization.

Beside the described components, the module also uses existent parts of the 3D Slicer GUI to provide the segmentation functionality. For example, the seeding of fiducial values is performed in the slice viewers and after each segmentation step, the generated three dimensional model is shown within the rendering window (see *Appendix D*).

Implementation of the created Level Set Segmentation Module

The created module, named *VMTKLevelSetSegmentation*, enables the segmentation of vascular structures from MR or CT data sets and displays the results as a rendered model within 3D Slicer. As a reference implementation it points out how to access VMTK from 3D Slicer.

The interactive placement of fiducial values (seed points) was realized using *vtkMRMLFiducialListNodes* by observing the slice viewers. Their coordinates have to be converted from the XYZ-system of the slice viewers to the RAS[22]-system and then to the IJK[23]-system to be used within VMTK. Methods for performing these transformations using *vtkMRMLScalarVolumeNode* were included into *SlicerVMTKLevelSetGUIHelper*.

After initialization and evolution steps, a generated model of the currently segmented volume is displayed. In the early stages of development, the model generation was performed by the "Grayscale Model Maker" module, which is part of 3D Slicer. To become more flexible and independent of the model maker, model generation was soon realized using the Marching Cubes[24] implementation of VTK in *SlicerVMTKLevelSetLogic*.

```
SetAndMergeVolume(newVolume):

    if outputVolumeNode does not exist
        create empty outputVolumeNode
        create empty outputVolumeNodeLast

    copy current outputVolumeNode to outputVolumeNodeLast
    set IJKToRASMatrix of outputVolumeNode to outputVolumeNodeLast

    if outputVolume is not empty
        merge newVolume and content of outputVolumeNode using vtkImageMathematics
        newVolume = mergedVolume

    set content of outputVolumeNode to newVolume

    return outputVolumeNode


Undo:

    copy outputVolumeNodeLast to outputVolumeNode
    set IJKToRASMatrix of outputVolumeNodeLast to outputVolumeNode

    return outputVolumeNode
```

Code 4: Pseudo code showing the undo and merge mechanism of the SlicerVMTKLevelSetGUIHelper class

Each step of either initialization or evolution results in a new volume which then gets merged with the last volume by using *vtkImageMathematics*. These steps can be repeated. During this sequence, undoing the last performed step is important. Therefore the Level Set Segmentation module copies the current output volume to a MRML node which can be restored if requested. Code 4 shows the undo and merge mechanism, which is similar for initialization and evolution.

22 Right-Anterior-Superior system, describes coordinates in relation to the patient (world coordinates).

23 IJK are the labels of the grid like XYZ (image coordinates).

24 The Marching Cubes is a geometric rendering technique invented by Lorensen and Cline, patented 1985. [Uspto2009]

VTK/VMTK Pipelines

The Vascular Modeling Toolkit supports several algorithms for segmenting vessels using level sets. Selected methods were chosen to be provided by the level set segmentation module.

Each method is implemented as a VTK/VMTK pipeline in the logic class *SlicerVMTKLevelSetLogic* of the created module. These methods accept MRML nodes as well as primitive types as arguments and return instances of the class *SlicerVMTKLevelSetContainer* to be able to pass more than one object. All techniques were adapted from existing code in the Vascular Modeling Toolkit [Vmtk_SVN, folder vmtkScripts/].

Initialization methods

The following initialization methods exist in VMTK and were integrated into the level set segmentation module. Every method always expects a *vtkMRMLScalarVolumeNode* as an input parameter.

Colliding Fronts

The algorithm starts two fronts – each from every seed point. The region compromised between these two points gets identified because the fronts travel in opposite directions. Side branches are ignored. [Piccinelli2009, p. 2]

Parameters: 2 seed points, lowerThreshold, upperThreshold

Fast Marching

Fronts starting from the source points are propagated until they reach the target points. The region between gets segmented. This is especially effective for segmenting round objects. [Vmtktut2009]

Parameters: n source points, n target points, lowerThreshold, upperThreshold

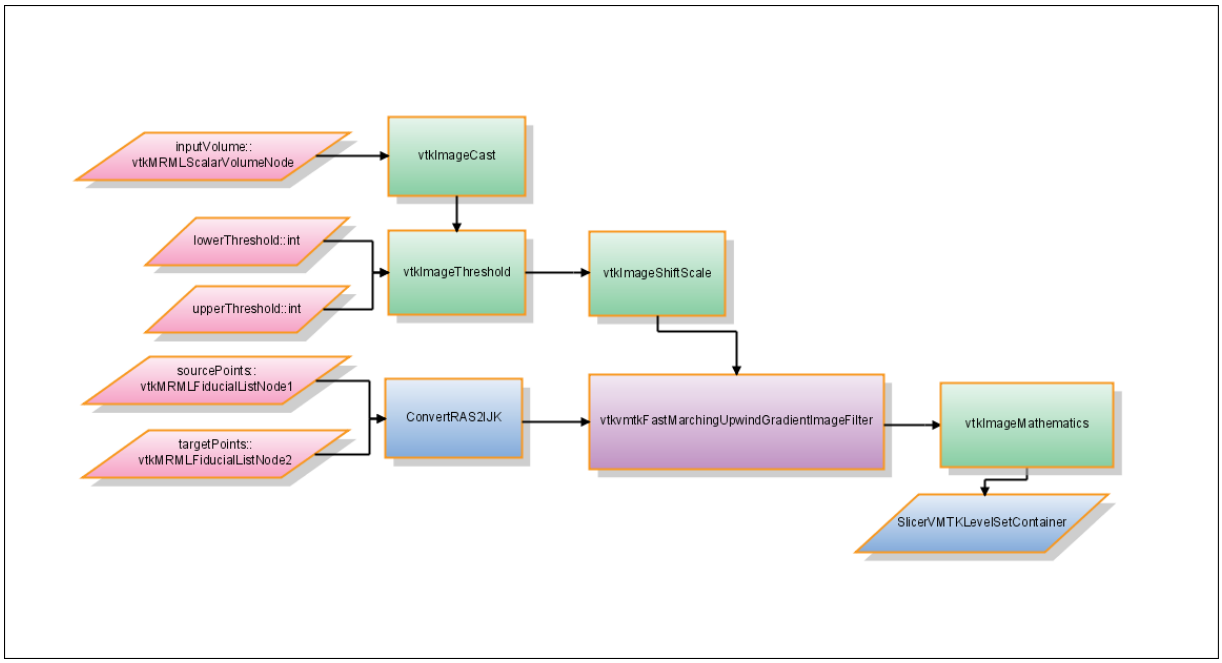


Illustration 4.11: Self created flow chart showing the VTK/VMTK pipeline of Fast Marching initialization

Colored flow chart elements show the following group memberships:

pink: input parameters based on MRML nodes or primitive types
green: VTK filters
blue: VMTKLevelSetSegmentation methods or output classes
purple: VMTK filters

Isosurface

A surface connects all voxels with intensity value greater or equal the isosurface value.

Parameters: isosurfaceValue

Threshold

All voxels between the given threshold get segmented.

Parameters: lowerThreshold, upperThreshold

Seeds

A region of three voxels around the placed seed points get segmented.

Parameters: n seed points

The flow chart diagram in Illustration 4.11 describes the VTK/VMTK pipeline of the Fast Marching method. Additional flow chart diagrams describing initialization methods can be found in *Appendix F*.

Evolution methods

The level set segmentation module provides two evolution methods of VMTK to inflate the initial level set:

1. Curves evolution (*vtkvmtkCurvesLevelSetImageFilter*) which is based on the *ITK Curves filter* [Lorigo2001]
2. Geodesic evolution (*vtkvmtkGeodesicActiveContourLevelSetImageFilter*) [Caselles1997].

Each requiring the following parameters [Antiga2008, p. 7]:

- Number of iterations, equals the number of deformation steps the model performs
- Propagation scaling: a weight for model inflation
- Curvature scaling: a weight for model surface regularization
- Advection scaling: a weight for the attraction of the image gradient modulus ridges

Before the evolution filters are used, a feature image is calculated using the VMTK filter *vtkvmtkGradientMagnitudeImageFilter*.

Changes to the 3D Slicer Base and KWWidgets

During this student research project, the following events were added to the KWWidgets classes and have been committed into the official repository [KWWidgets_CVS].

vtkKWWizardStep.h and *vtkKWWizardStep.cxx*
The events *ShowUserInterfaceEvent*, *HideUserInterfaceEvent* and *ValidateEvent* were added to enable event-driven wizards [KWWidgets_CVS, rev. 1.3 and 1.6].

vtkKWExtent.h and *vtkKWExtent.cxx*
The events *ChangeEvent*, *StartChangeEvent* and *EndChangeEvent* were added to perform event-driven processing of the extent widget [KWWidgets_CVS, rev. 1.42 and 1.55].

Also a bug in *Slicer3/Libs/MRML/vtkMRMLVolumeNode::SetIJKToRASMatrix(vtkMatrix4x4* mat)* was fixed to prevent the modifying of the passed matrix argument while setting the transform matrix of the node [Slicer_SVN, rev. 8937].

Additionally to the realized GUI, a prepackaged simple mode using a wizard (*vtkKWWizard)* was planned but soon discontinued. A wizard-based solution should provide additional preprocessing and post-processing steps which are not part of the original level set segmentation function in VMTK. Nevertheless, the following changes were made to the 3D Slicer base to enable wizard functionality in Python and were checked into the official repository [Slicer_SVN, rev. 8630].

Slicer3/Base/GUI/vtkSlicerApplicationGUI.cxx
The method *PythonCommand(char cmd*)* was fixed to work inside *Scripted Modules* including the passing of the Python dictionary.

Slicer3/Applications/GUI/Slicer3.cxx
The Python initialization code (line 1650) was moved to ensure that the Python dictionary is set during the loading of *Scripted Modules*.

Slicer3/Modules/ScriptedModule/vtkScriptedModuleGUI.h and *vtkScriptedModuleGUI.cxx*
The methods *Invoke (char* method, char* args)* and *Invoke (char* method)* were added to enable *vtkKWWizards* in Python without using events but by passing a specific method and optional arguments.

Slicer3/Base/GUI/Python/SlicerScriptedModule.py
The initialization of the logic and GUI part of Python modules (line 64) were moved to avoid a violation while accessing the module name.

Conclusion

The Vascular Modeling Toolkit libraries are available for 3D Slicer. With the successful connection of these two complex software solutions, processing pipelines between VMTK code and other algorithms can be established. Several techniques for three dimensional reconstruction, geometric analysis, mesh generation and surface data analysis for image-based modeling of blood vessels or tubular structures are then made accessible to the 3D Slicer developer.

Using the GUI-less module approach, a user friendly way of integrating the algorithms of the toolkit is provided without requiring manual changes to the 3D Slicer base. The created module, called *VmtkSlicerModule,* is open-source and available at the "VMTK in Slicer" project website (see Appendix G) within a SVN repository. In order to eliminate the problem of backwardness, the module always includes the newest version of the Vascular Modeling Toolkit libraries using SVN property settings.

Additionally a level set segmentation module, referred to as *VMTKLevelSetSegmentation*, was created as a Python Scripted Module to evaluate and proof the integration of VMTK. This module maps exactly the level set segmentation process of the Vascular Modeling Toolkit into the "point-and-click" end user application 3D Slicer - including undoing and merging of specific segmentation stages.

VMTK provides different level set segmentation methods. A selection of five initialization and two evolution methods are accessible through the 3D Slicer level set segmentation module. These methods are maintainable and extendable without changing the structure of the module because of the class design. Separating the included logic of the module into an isolated class provides possibilities for automatic testing.

Standard techniques like MRML nodes were used as data containers in *VMTKLevelSetSegmentation*. This enables the processing of segmentation results with other functions of 3D Slicer. For example, generated models can be post-processed or fiducial values can be exported. The module itself is bound to a MRML node as well and therefore it is possible to run different instances to perform several segmentation tasks at the same time. The source code of is also available at the "VMTK in Slicer" project website (see *Appendix G*).

During development enhancements and fixes to either the 3D Slicer base or KWWidgets were applied.

Segmentations using the two modules were performed and are selectively documented on the following pages.

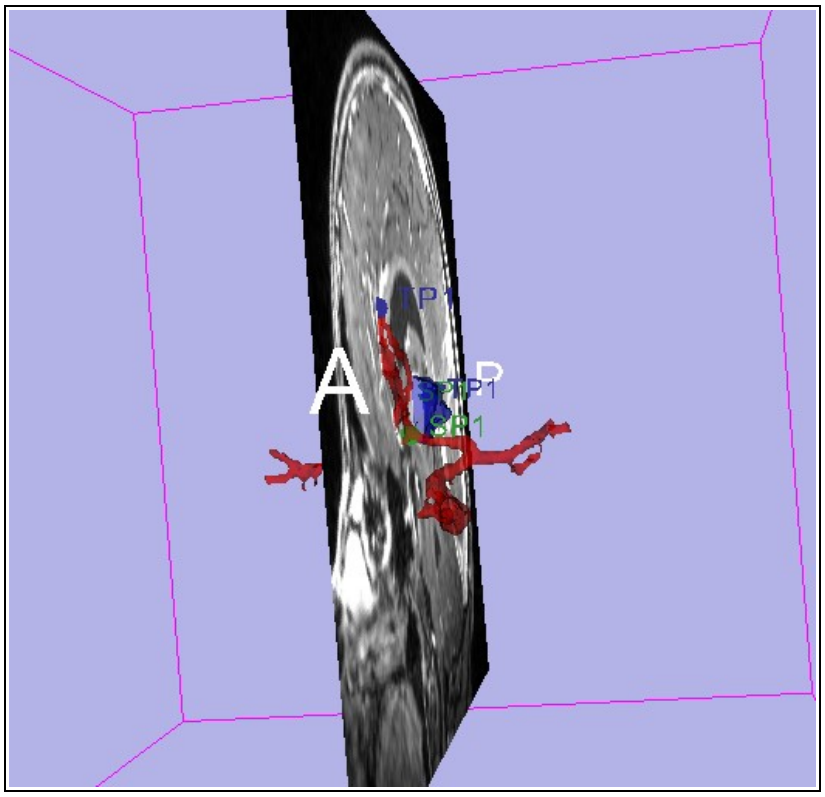


Illustration 4.12: Cerebral vessels and a brain tumor segmented using the created modules (MRA dataset)

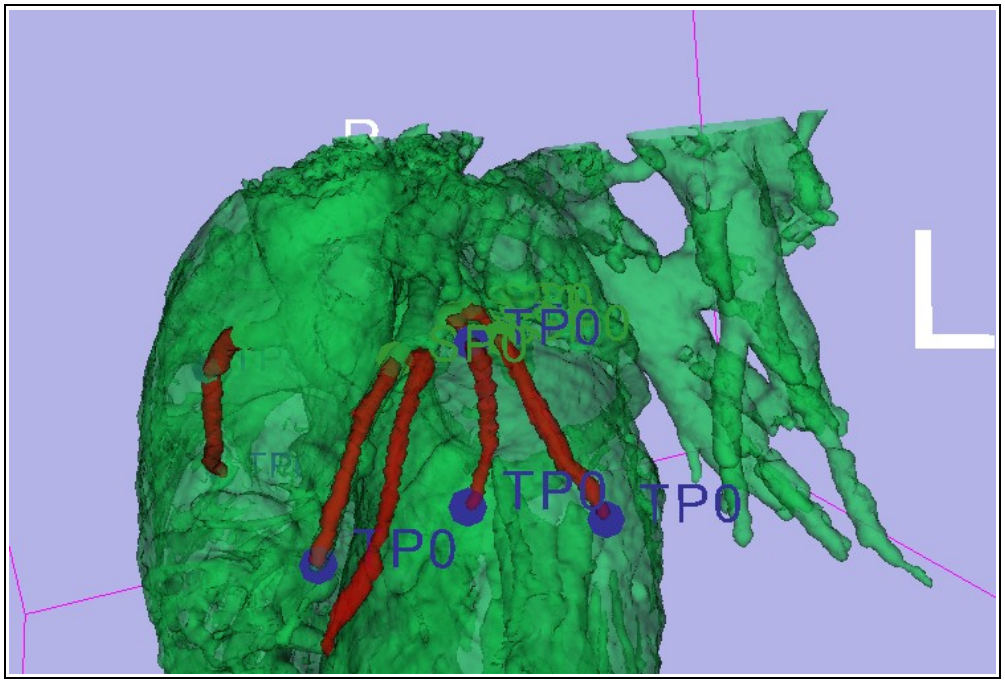


Illustration 4.13: Coronary arteries and vein partially segmented using the created modules (cardiac blood pool MRI dataset)

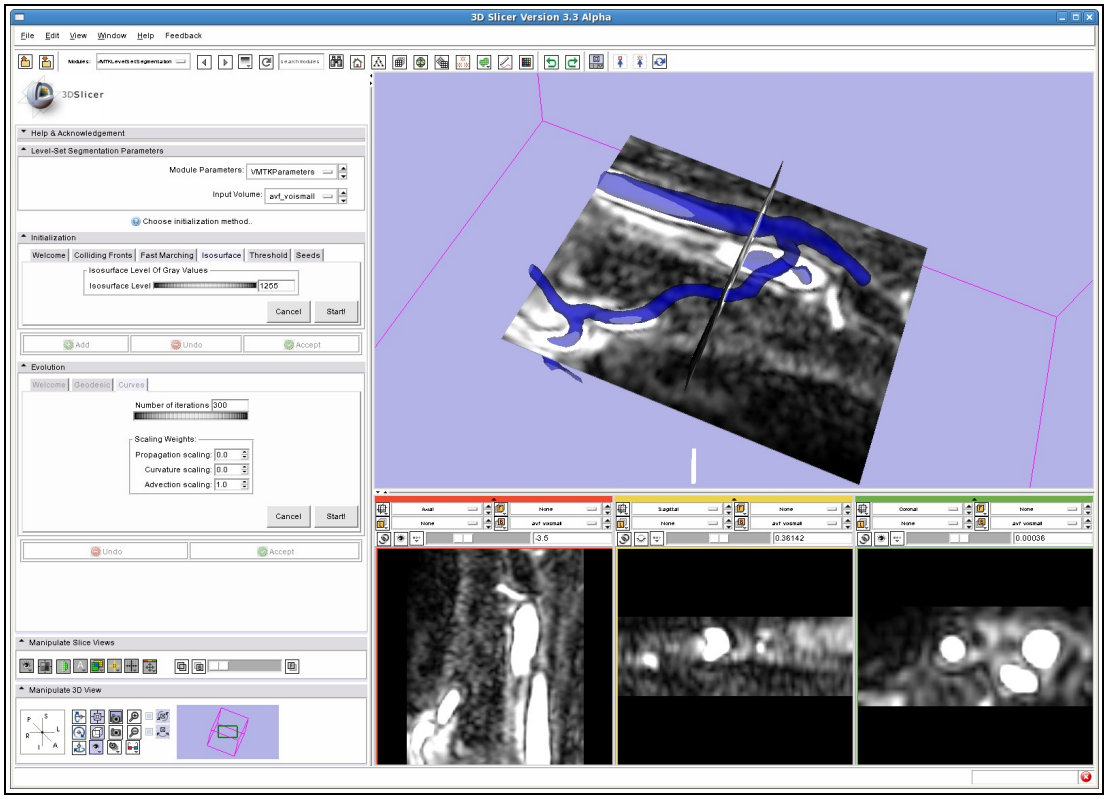


Illustration 4.14: Vessels of an arm segmented using the created modules

5 Discussion and Outlook

When using VMTK within 3D Slicer, one obstacle may be that the needed infrastructure is not directly included in the application base. Thus it requires the downloading and installation of *VmtkSlicerModule* from the project page. Despite this, the integration using the GUI-less module approach still seems to be a suitable way of extending 3D Slicer. It leaves the 3D Slicer base untouched and only requires users interested in VMTK to install the library. Also, the support for the 3D Slicer Extension Build System should simplify the installation procedure.

By providing access to the Vascular Modeling Toolkit within 3D Slicer, the groundwork for further development using VMTK algorithms is laid. Centerline computation or generating meshes are two interesting possible applications for such modules. The created reference implementation supports understanding of accessing the VMTK algorithms. Therefore it can be expected to see more modules offering or using functionalities of VMTK in 3D Slicer.

The VMTK level set segmentation module maps the process of VMTK to a 3D Slicer module. Since this process targets manual segmentation for research purposes, it does not include preprocessing or post-processing steps and requires knowledge of the algorithms. So the module might not be usable as a stand-alone level set segmentation solution for end users. Further enhancements could be:

- A wizard based interface for easier usage
- Volume of interest selection in advance of the segmentation process
- Including vessel enhancement filters as preprocessing steps
- Providing a prepackaged mode based on a robust algorithm
- 3D seeding inside the 3D Slicer rendering window
- Different coloring of individual vessel branches

All of the used and provided solutions are available as actively maintained open-source software targeting large developer and user communities. Consequently, one can be excited about new projects concerning the Vascular Modeling Toolkit in 3D Slicer.

6 Bibliography

The following literature and resources were used during this student research project.

[Antiga2008] Antiga L, Piccinelli M, Botti L, Ene-Iordache B, Remuzzi A, Steinmann DA. (2008) An image-based modeling framework for patient-specific computational hemodynamics. Med Biol Eng Comput 46(11):1097-1112

[Antiga2009] Antiga L (2009) Python and Slicer. NA-MIC AHM 2009, Salt Lake City. http://tinyurl.com/SlicerPython (Last accessed 2009-05-01)

[Aylward2002] Aylward S (2002) Initialization, Noise, Singularities, and Scale in Height Ridge Traversal for Tubular Object Centerline Extraction. IEEE Trans Med Imaging 21(2):61-75

[Caselles1997] Caselles V, Kimmel R, Sapiro G (1997) Geodesic active contours. Int J Comput Vis 22(1):61-79

[Cmake2009] CMake – Cross Platform Make. http://www.cmake.org (Last accessed 2009-05-01)

[ITK2009] ITK – Segmentation & Registration Toolkit. http://www.itk.org (Last accessed 2009-05-01)

[ITKVTK2009] Ibanez L, Schroeder W (2009) Getting Started with ITK+VTK. Insight Software Consortium. http://www.itk.org/CourseWare/Training/GettingStarted-II.pdf (Last accessed 2009-05-01)

[Kirbas2004] Kirbas C, Quek F (2004) A Review of Vessel Extraction Techniques and Algorithms. ACM Comp Surveys 36(2):81-121

[KWWidgets_CVS] KWWidgets, CVS Repository. http://kwwidgets.org/cgi-bin/viewcvs.cgi/?root=KWWidgets (Last accessed 2009-05-01)

[Kwwidgets2009] KWWidgets, Overview. http://www.kwwidgets.org/Wiki/KWWidgets/Overview (Last accessed 2009-05-01)

[Lorigo2001] Lorigo LM, Faugeras OD, Grimson WEL, Keriven R, Kikinis R, Nabavi A, Westin CF (2001) CURVES: curve evolution for vessel segmentation. Med Image Anal 5:195-206

[Mrml2009] 3D Slicer, MRML.
http://www.na-mic.org/Wiki/images/e/e3/Slicer3_MRML.ppt
(Last accessed 2009-05-01)

[Namic2008] 3D Slicer, Architecture and Implementation.
http://www.na-mic.org/Wiki/images/5/5b/Slicer-Architecture-Implementation-London-2008-10-15.ppt (Last accessed 2009-05-01)

[NAMICkit2009] The NA-MIC Kit.
http://wiki.na-mic.org/Wiki/index.php/SoftwareInventory
(Last accessed 2009-05-01)

[OpenGL2009] OpenGL Graphics System, A Specification, Version 3.1.
http://www.opengl.org/registry/doc/glspec31.20090324.pdf
(Last accessed 2009-05-01)

[Piccinelli2009] Piccinelli M, Veneziani A, Steinman DA, Remuzzi A, Antiga L (2009) A framework for geometric analysis of vascular structures: applications to cerebral aneurysms. IEEE Trans Med Imaging. In press.

[Pilgrim2009] M. Pilgrim (2009) Dive into Python. Online edition.
http://www.diveintopython.org/toc/index.html
(Last accessed 2009-05-01)

[Sethian1999] Sethian JA (1999) Level set methods and fast marching methods, 2nd edn. Cambridge University Press, Cambridge

[Slicer_SVN] 3D Slicer, SVN Repository.
http://viewvc.slicer.org/ (Last accessed 2009-05-01)

[Slicer2009] 3D Slicer, Introduction.
http://slicer.org/pages/Introduction (Last accessed 2009-05-01)

[Slicer2009b] 3D Slicer, Integrating with Slicer 3.
http://slicer.spl.harvard.edu/slicerWiki/images/7/75/Integrating_with_Slicer3.ppt
(Last accessed 2009-05-01)

[Slicer2009c] 3D Slicer, Building a Slicer 3 Module GUI.
http://slicer.spl.harvard.edu/pages/Building_a_Slicer_3_Module_GUI
(Last accessed 2009-05-01)

[Slicervmtklvlst_SVN]
VMTK in Slicer, SVN Repository.
http://www.nitrc.org/plugins/scmsvn/viewcvs.php/?root=slicervmtklvlst
(Last accessed 2009-05-01)

[Tcl2009] The Tcl/Tk Toolkit.
http://www.tcl.tk/ (Last accessed 2009-05-01)

[Vmtk_SVN] The Vascular Modeling Toolkit, SVN Repository.
http://vmtk.svn.sourceforge.net/viewvc/vmtk/ (Last accessed 2009-05-01)

[VMTK2009] Antiga L, Steinman DA (2008) The Vascular Modeling Toolkit.
http://www.vmtk.org/ (Last accessed 2009-05-01)

[Vmtktut2009] The Vascular Modeling Toolkit, Image Based Modeling Tutorial.
http://www.vmtk.org/Tutorials/ImageBasedModeling (Last accessed 2009-05-01)

[VTK2009] VTK – The Visualization Toolkit.
http://www.vtk.org/ (Last accessed 2009-05-01)

[Uspto2009] US Patent and Trademark Office, Marching Cubes Algorithm, 1985.
http://tinyurl.com/MarchingCubes (Last accessed 2009-05-01)

[Yim2003] Yim PJ, Vasbinder GBC, Ho VB, Choyke PL (2003) Isosurfaces as deformable models for magnetic resonance angiography . IEEE Trans Med Imaging 22(7):875 - 881

7 Appendix

A. Features available in the Vascular Modeling Toolkit, published at http://www.vmtk.org (Updated 2009-04-20)

Segmentation of vascular segments (or other anatomical structures) from medical images:

- Gradient-based 3D level sets segmentation. A new gradient computation modality based on upwind finite differences allows the segmentation of small (down to 1.2 pixels/diameter) vessels.
- Interactive level sets initialization based on the Fast Marching Method. This includes a brand new method for selecting a vascular segment comprised between two points automatically ignoring side branches, no parameters involved. Segmenting a complex vascular tract comes down to selecting the endpoints of a branch, letting level sets by attracted to gradient peaks with the sole advection term turned on, repeating the operation for all the branches and merging everything in a single model.

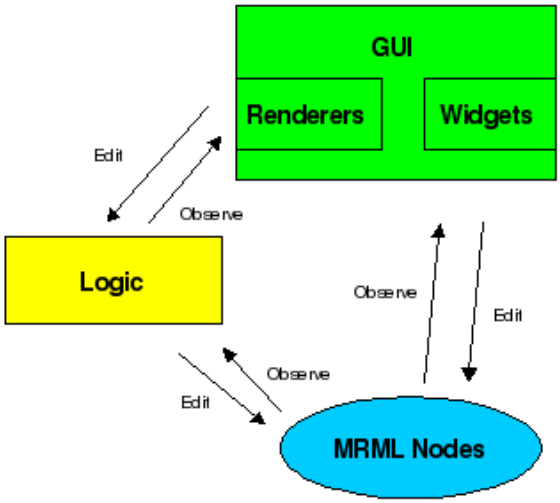
Geometric analysis and surface data processing of 3D models of blood vessels (and tubular objects in general)((The key algorithms have been published on medical imaging journals. You can find a complete reference to publications at David Steinman's and Luca Antiga's homepages)):

- Compute centerlines and maximal inscribed sphere radius of branching tubular structures given their polygonal surface representation
- Split surface models into their constitutive branches based on centerline geometry
- Compute centerline-based geometric quantities (such as bifurcation angles, planarity, symmetry, branch curvature, tortuosity) and surface-based geometric quantities (such as distance to centerlines, surface curvature, deviation from tangency to maximal inscribed spheres)
- Robustly map branches to a rectangular parametric space
- Generate rectangular patches based on the parametric mapping for statistical analysis of geometric and CFD data over populations.

Scripts, I/O tools and simple algorithms to easily work with images and meshes:

- Read and write a number of image, surface and volume mesh formats. Includes a DICOM series reader with auto-flipping capabilities, Netgen mesh format reader, libMesh xda mesh format writer, Tetgen mesh generator wrapper, FIDAP FDNEUT mesh format reader and writer and a Newtetr input file generator
- Display images and meshes
- Incapsulate several VTK classes and make them available as pipeable scripts (e.g. Marching Cubes, surface smoothing, clipping, normal computation, connectivity, subdivision, distance between surfaces, ICP registration)
- Add cylindrical extensions to surface model boundaries as a preprocessing step for CFD simulations.
- Generate boundary layers of prismatic elements with varying thickness for CFD

B. Slicer MVC Pattern (Updated 2008-10-15) [Namic2008]



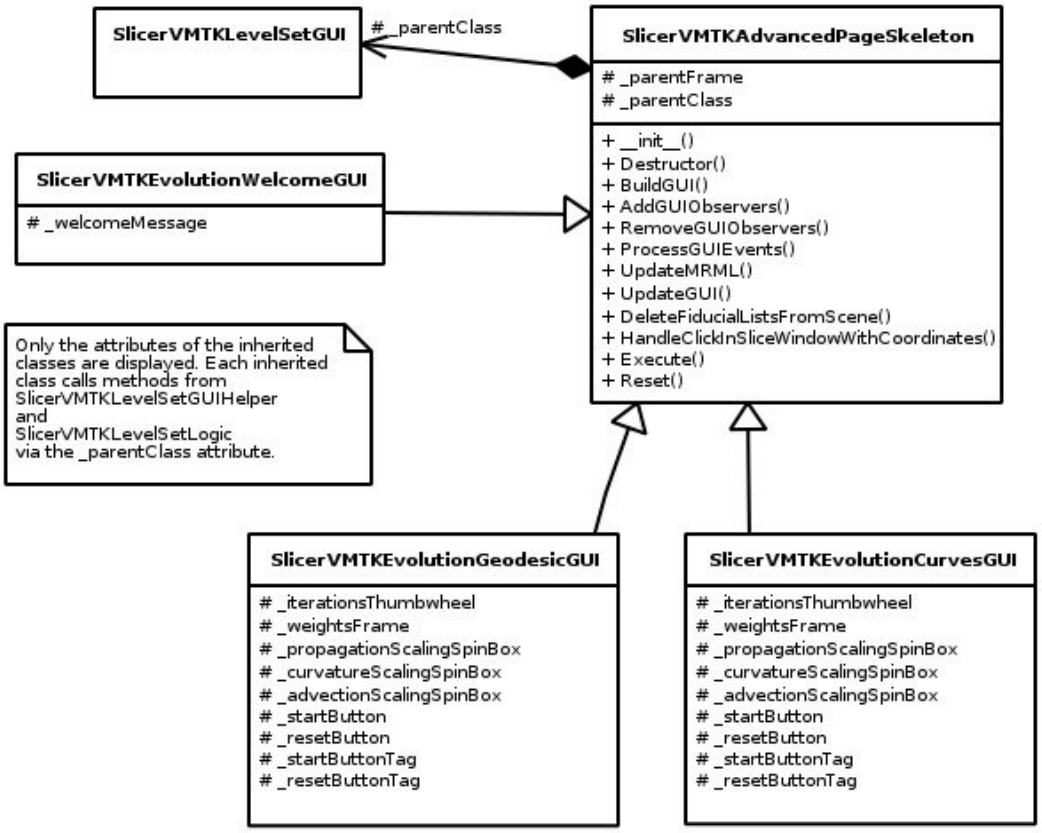
"**Observe**" means generic event mechanisms are used to pass information.
"**Edit**" means code can directly call methods.

Example: GUI can call methods in Logic classes, but Logic *cannot* call GUI methods.
MRML *cannot* call Logic or GUI methods.

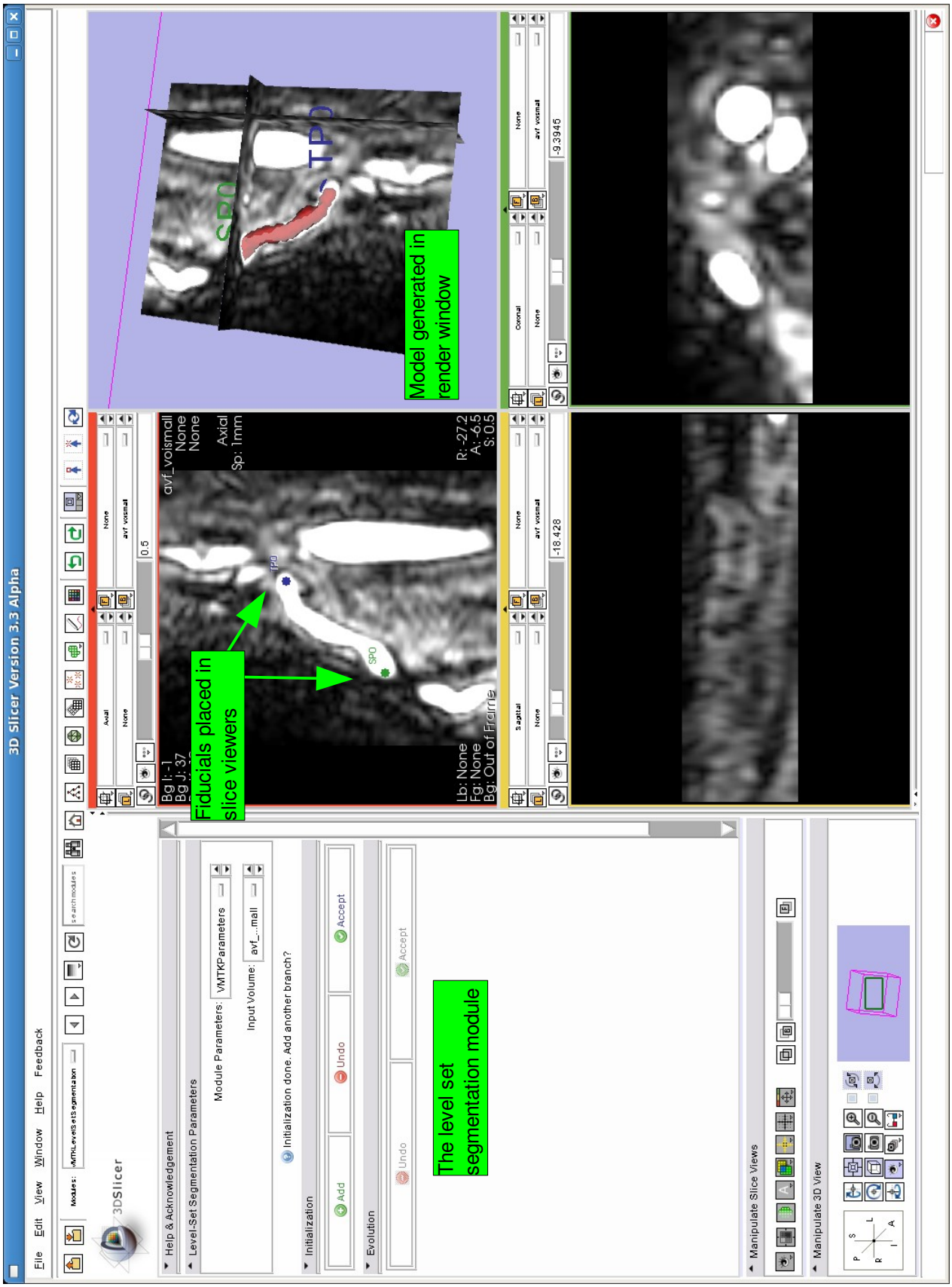
There can be many observers for any event.

- ***MRML*** (Model)
  - For Scene Description and Application State
  - MRML Nodes are Persistent and Undoable
  - Scene and Nodes are Observable
- ***Logic*** Encapsulate VTK and ITK Pipelines (Controller)
  - Observe MRML to Configure Pipelines
  - Help Create/Manage Nodes
  - No UI Components (no Widgets, Actors, Mappers, Renderers or RenderWindows)
- ***GUI*** (View)
  - Observe and Edit MRML
  - Interact with User and Display Hardware
- Modules Should Follow Same Conventions

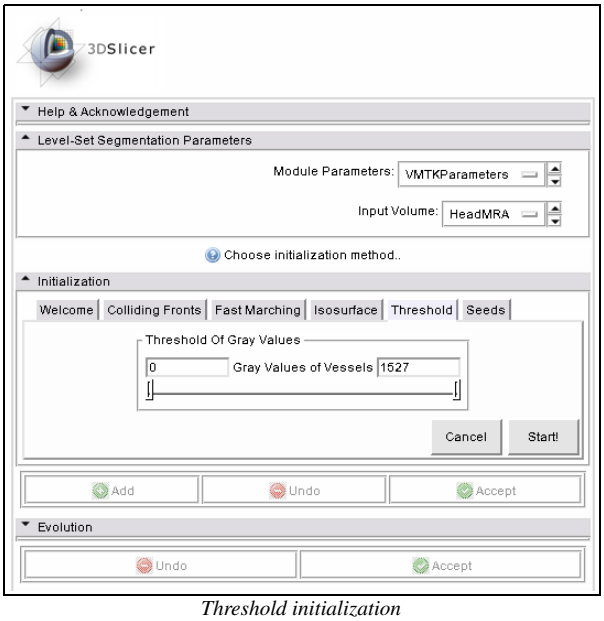
C. Self created class diagram showing the individual classes of the evolution methods



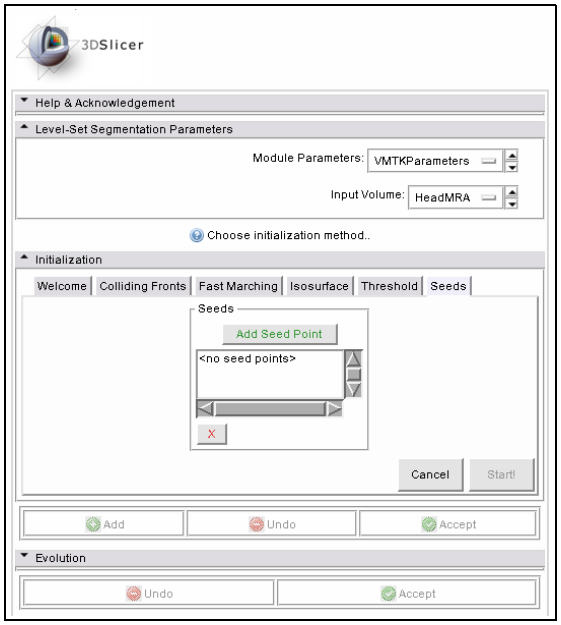
D. Self created screen shot of the level set segmentation module in 3D Slicer



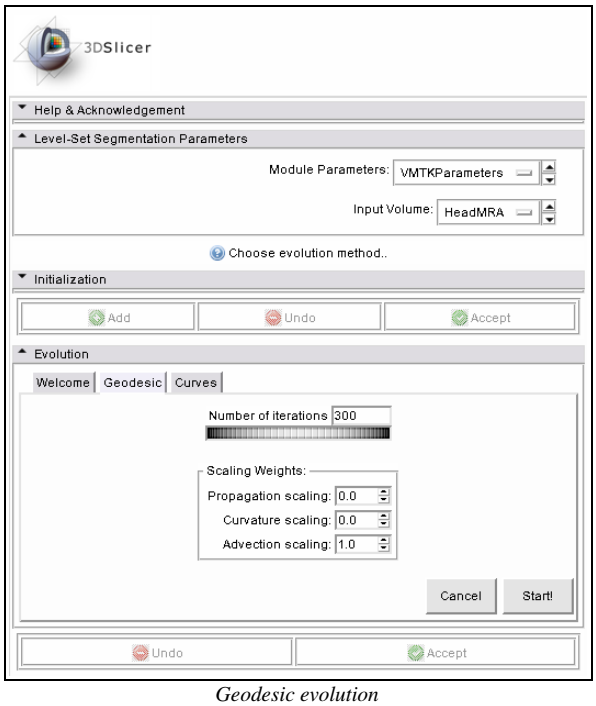
E. Additional screen shots of the Level Set segmentation module GUI



Threshold initialization



Seeds initialization



Geodesic evolution

F. Additional flow chart diagrams, showing the VTK/VMTK pipelines of *VMTKLevelSetSegmentation*

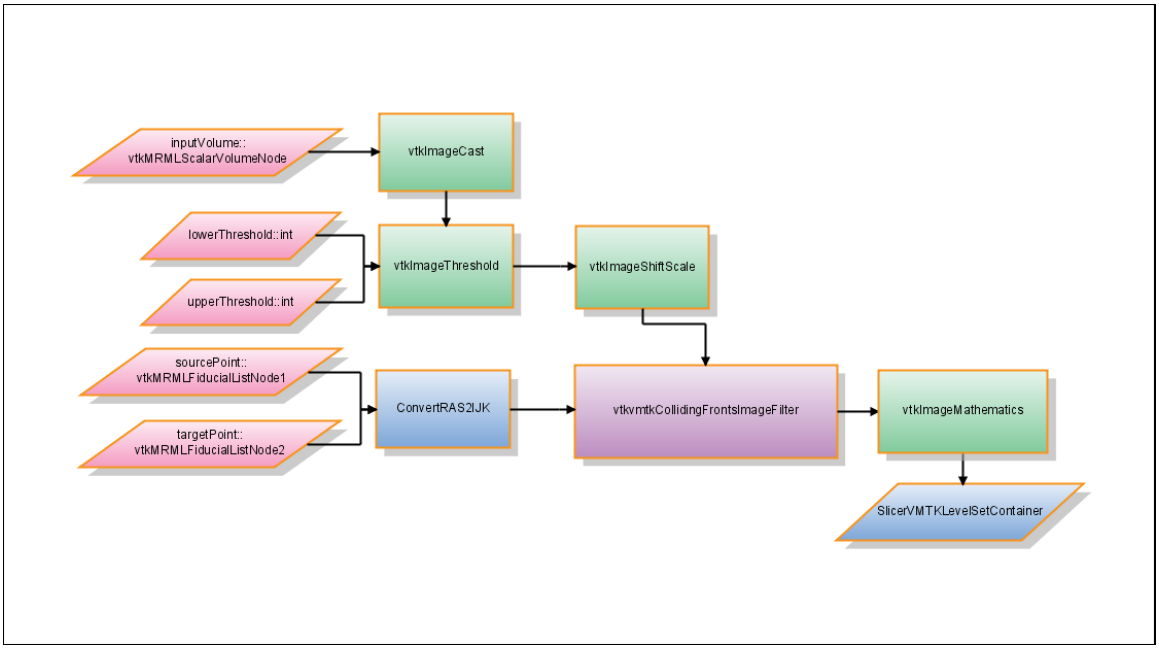


Illustration 7.2: Self created flow chart showing the VTK/VMTK pipeline of Colliding Fronts initialization

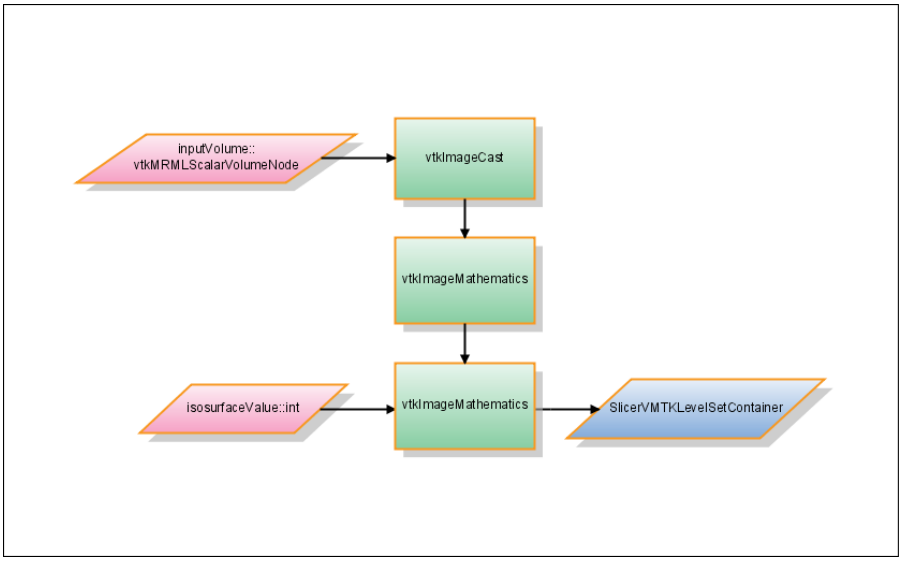


Illustration 7.1: Self created flow chart showing the VTK pipeline of Isosurface initialization

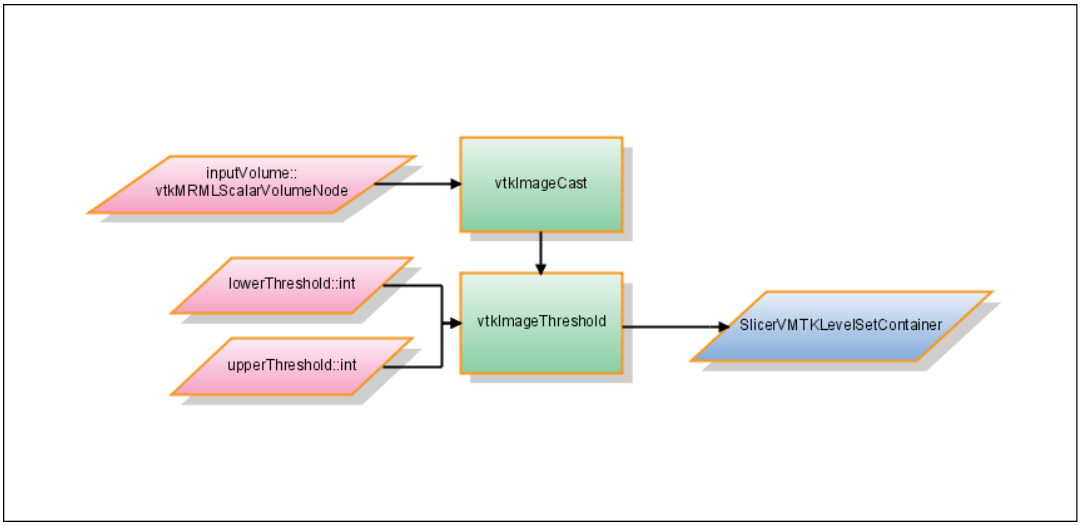


Illustration 7.3: Self created flow chart showing the VTK pipeline of Threshold initialization

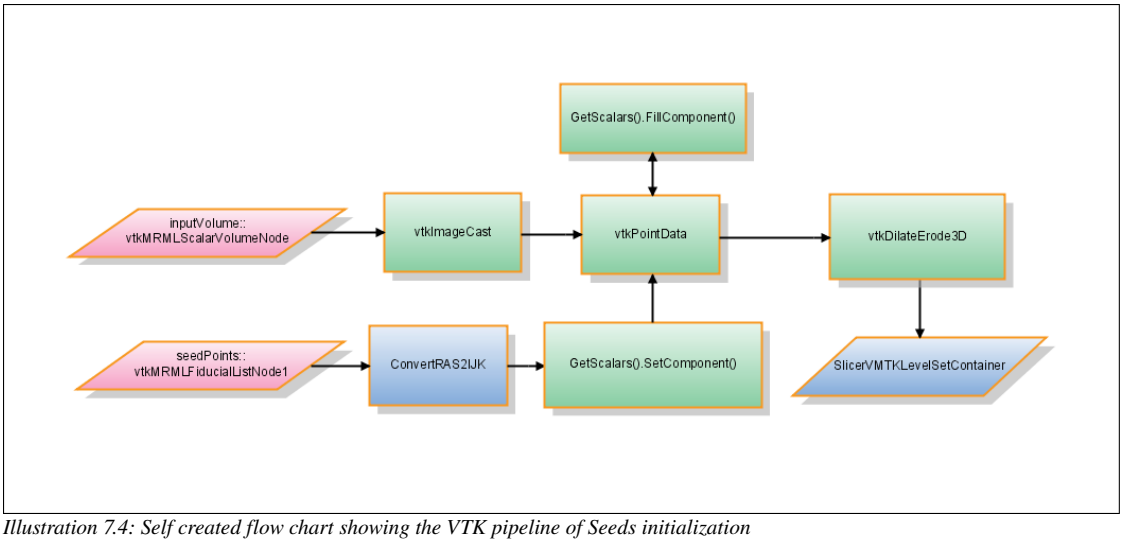


Illustration 7.4: Self created flow chart showing the VTK pipeline of Seeds initialization

G. Screen shot of the NITRC project page of VMTK in Slicer and the Level Set Segmentation Module (http://www.nitrc.org/projects/slicervmtklvlst/, updated 2009-04-27)