Integrating Segmentation Methods from the Insight Toolkit into a Visualization Application

Ken Martin, Luis Ibáñez, Lisa Avila, Sébastien Barré, Jon Kaspersen

Kitware Inc. Clifton Park, NY, USA
kitware@kitware.com, http://www.kitware.com

SINTEF Unimed Ultrasound, Trondheim, Norway
http://www.sintef.unimed.com

Abstract

The Insight Toolkit initiative from the National Library of Medicine has provided a suite of state-of-the-art segmentation and registration algorithms ideally suited to volume visualization and analysis. A volume visualization application that effectively utilizes these algorithms provides many benefits: it allows access to ITK functionality for non-programmers, it creates a vehicle for sharing and comparing segmentation techniques, and it serves as a visual debugger for algorithm developers. This paper describes the integration of image processing functionalities provided by the Insight Toolkit (ITK) into VolView, a visualization application for high performance volume rendering. A free version of this visualization application is publicly available and is included on the CDROM accompanying this issue. The process for developing ITK plugins for VolView according to the publicly available API is described in detail, and an application of ITK VolView plugins to the segmentation of Abdominal Aortic Aneurysms is presented.

Key words: Volume Rendering, Visualization, Abdominal Aortic Aneurysm Segmentation, Collaborative Work, Remote collaboration

Preprint submitted to Medical Image Analysis 18 September 2003
Fig. 1. Volume rendering of the Visible Woman data set generated with VolView.

1 Introduction

Through the Insight Toolkit initiative from the National Library of Medicine, experts in the field of image processing have created a suite of state-of-the-art segmentation and registration algorithms ideally suited to volume visualization and analysis. Since this functionality is provided as a toolkit rather than an end-user application, access is limited to researchers with fairly advanced programming skills.

Volume visualization and analysis is a key tool in a variety of health care applications. It is widely used in radiation oncology, surgical planning, confocal microscopy, and education. In many of these applications, proper segmentation and registration of medical data is critical to producing the most efficacious results. This task is also generally considered to be the most challenging step in effective volume visualization.

1.1 Goals

The overall goal of this effort is to aid in the development, deployment, and refinement of segmentation and registration algorithms to benefit health care delivery, ed-

* Corresponding author.

Email address: ken.martin@kitware.com (Ken Martin).
ucation, and research. Our method for achieving this is to provide access to the ITK algorithms from VolView, an end-user volume visualization application. It benefits practitioners by allowing them to make use of The Insight Toolkit’s advanced algorithms with a low learning curve. It assists algorithm developers by proving a simple application into which they can insert their algorithms, allowing for quick performance evaluations on a number of different datasets and in comparison to other algorithms already in the application. It benefits the research community by providing a means of disseminating new algorithms to augment the technical descriptions provided in journals and conference proceedings. Finally, it serves as a visual debugger by allowing visual inspection of the results of a technique, which aids developers in the evaluation and verification of new algorithms, and assists practitioners in creating more complex methods from available building blocks.

1.2 Benefits

There are a number of specific benefits to integrating ITK segmentation and registration techniques with VolView as plugins. For the practitioner, educator, and algorithm developer the detailed benefits include:

For the Practitioner and Educator:

- Expand access - Enable practitioners and educators to easily and routinely make use of Insight’s algorithms with little to no learning curve from within a turnkey application. As opposed to direct use of the Insight toolkit, no programming experience would be required.
- Facilitate communication - Most end user tools provide no link between the end-users and the algorithm developers. Ideally a practitioner using an application should be able to contact the author of a particular algorithm she is using. This promotes communication, validation, and integration of algorithms into health care delivery.
- Promote evaluation - Provide a mechanism where a practitioner and researcher can collaborate to evaluate new algorithms. Algorithms should be capable of being emailed to the practitioner (or downloaded from a web site) and inserted into the practitioner’s application.

For the Algorithm Developer:

- Facilitate deployment - The application will be created such that new algorithms can be easily deployed to users without requiring new versions of the application. This removes the limitation of most turnkey systems that provide new functionality at scheduled releases that occur only once or twice a year, and it encourages both early and rapid feedback resulting in shorter development cycles.
- Increase dissemination - Researchers can easily provide a link to a turnkey application that allows readers to evaluate a technique described in a technical pub-
• Improve testing - Many segmentation and registration algorithms are complex and can be difficult to evaluate and test. Often validation can best be performed by visual inspection of the results of the algorithm.

The benefits build on and leverage the powerful features of both ITK and VolView. ITK includes a wide range of segmentation and registration techniques with the ability to process many different data types. Segmentation techniques include both region growing techniques such as confidence connected, isolated connected, RGB confidence connected, as well as level set techniques such as fast marching, shape detection, watershed, and geodesic active contours. ITK’s registration techniques include intensity based registration for a variety of transformations ranging from rigid through fully deformable with a range of optimizers including gradient descent, conjugate gradient, and evolutionary algorithms. Finite-element techniques can also be used as the basis of deformable registration techniques (Ibanez and Schroeder, 2003).

1.3 VolView Features

VolView’s features a wide range of functionality to facilitate viewing and interacting with volumetric data. It was developed as an end user application on top of the Visualization Toolkit VTK (Schroeder et al., 1998). The development of VolView began in early 1999, and the fifth version of the software, VolView 2.0, was released in September 2003. During this time VolView has matured to a full-featured volume visualization and image processing package featuring:

• high performance volume rendering on standard PC hardware
• support for a wide variety of data types from single component short to RGBA volumes
• a powerful editor for material classification
• adjustable lighting parameters to enhance structural understanding
• interactive annotation elements for marking positions, measuring angles and distances, cropping the volume, etc.
• composite and MIP projections in parallel or perspective
• orthogonal, MPR, and lightbox image views
• a rich set of image processing functionality that is extensible
• collaboration between remote sites with transferable control

[LISA:] Volume rendering citation : (Kaufman and Sobierajski, 1995).
Fig. 2. Peer to peer communication in VolView. Two distant sites can alternate control of the visualization and image processing parameters. VolView will present the same display at both sites and transfer data as needed from one site to the other.

1.3.1 Collaborative Mode

The collaborative functionality in VolView was developed as part of a research effort with the University Hospital of Trondheim in Norway. Patients requiring stents for abdominal aortic aneurysms would travel to the main hospital for treatment. These patients require monitoring after the procedure, that also involved the patient traveling, sometimes many hundreds of miles, to the University hospital for a follow-up CT scan. Using VolView, the regional hospitals are able to perform the follow-up scans, with a collaborative VolView session (Fig. 2) between the radiologist at the regional hospital and the expert at the university hospital used to analyze the scan. This reduces the cost of care of the patient, minimizes the impact on the patient’s life, and increases the knowledge in the regional hospitals.

1.3.2 Addition of Plugins

By developing a plug-in using the public API, researchers can access their new image processing algorithms from within a full-featured visualization application. This allows algorithm developers to quickly verify and improve their new processing techniques. Since VolView is able to render a variety of data types, this allows the researcher great freedom in designing the new algorithm. VolView can render any single component data type from 8 bit integers through 64 bit floating point values, and can also render multi-component data such as registered MR/CT studies or processed color (RGBA) volumes.

These plugins are even more useful within a collaborative setting. This allows a segmentation researcher to seek an expert opinion during the algorithm design phase where both the researcher and clinician can interact and evaluate the performance of the segmentation on a variety of datasets.
Fig. 3. Sharing algorithms with VolView. Plugins can be developed for VolView. These plugins implement image processing methods such as segmentation and registration. The plugins can be shared between distant sites, they could be sent by email, or stored in web accessible pages. Once transferred, they can be loaded in VolView at run-time without having to restart the application.

2 Plugin Architecture

This section describes the major issues and requirements involved in the integration of ITK algorithms into VolView. Before discussing the design consider how such plugins may be used.

- A researcher is looking for an algorithm to segment a tumor from within the liver. A search on the Internet yields a web page that discusses a new algorithm within the Insight Toolkit that looks applicable. The researcher can then download a plug-in for this algorithm and run the freely available volume rendering and analysis application. The application will provide additional documentation on how to use the algorithm, what parameters can and should be adjusted, and who to contact with questions and feedback. The researcher uses the algorithm but finds a case for which it fails. He then can contact the algorithm developer to discuss this failure and possibly provide insight into a solution. The algorithm developer modifies the algorithm and produces a new plug-in which is then emailed to the researcher. The researcher can then evaluate the modified algorithm on the data that caused the earlier failure.

- Experiment locally with ITK algorithms for which new plugins are written on-site. The number of plugins currently distributed is limited. However, all the elements required for creating new plugins are publicly available, this makes possible for anyone to create new plugins in their in-house development.

- Create plugins for new image processing methods and send them to other sites for evaluation without having to expose the source code, nor requiring the other site to expose their data. In the cases in which local developments can not be publicly shared, the plugins make still possible to provide portions of executable code that can be evaluated by other sites.

- Provide implementations for methods being published. In order to facilitate the verification of reproducibility for work published in scientific and technical jour-

---

1 A free version of VolView is now available as part of a project sponsored by the National Library of Medicine (NLM). This version is included in the CDROM accompanying this issue.
nals, authors would be able to post plugins implementing their published methods. In this way, their peers in the community would be able to rapidly reproduce their results without being confronted to the discouraging barrier of having to implement an algorithm described in pseudo-code in a paper.

- Facilitate deployment. New releases of algorithms can be deployed without having to wait for full releases of the application. The modular separation between VolView and the plugins facilitate to update the plugins without having to make any changes in the application.
- Distribute specialized plugins as commercial products. Current plugins has been written for very generic image processing tasks. However, very specialized plugins could be written for specific applications and could be distributed in the form of commercial products. The great advantage of being VolView plugins is that all the visualization and user interaction functionalities are obtained for free. The plugin developer can then focus only on the image processing aspects of her work.
- Hosting plugins repositories. Specialized plugins that are in the public domain could be made available in binary form. In combination with adequate querying protocols they could serve as support resources for grid-like computation.

These Use-Cases imply a number of requirements. The following list illustrates some of the fundamental requirements for a successful plugin architecture.

- Plugins must be flexible so that they can be compiled on one computer and then emailed (or downloaded) to another researcher.
- Most segmentation techniques require that the clinician specify a number of parameters or seed points for the segmentation. This dictates that the plugin mechanism must provide for a graphical user interface so that the user can select parameters and adjust values or seed points.
- Plugins must provide built-in documentation describing how to use them and what the different parameters control.
- For computationally expensive plugins there must be some indication of progress and the ability to cancel the computation if it is taking too long.
- For memory intensive plugins VolView must be able to estimate how much memory will be required to run the plugin. Some segmentation techniques require a large amount memory for intermediate results which for a large medical dataset can easily exceed the computer’s memory.
- The plugin must receive a full description of the input data including aspects such as dimensions, spacing, data type, units as well as specifying those same properties for the output it will produce.
- The architecture should support plugins that can process their data in-place or in pieces for lower memory consumption as well as plugins that cannot.
- Some plugins may produce surfaces (or meshes) instead of an output volume.
- A plugin may require more than one volume of input data. This is common for many level set segmentation techniques that require a feature image as well as an initial segmentation to operate.
2.1 Plugin - VolView Interactions

The most interesting aspect of the mechanism used for integrating ITK capabilities into VolView is the decoupling between both systems. ITK methods do not have to be exposed to the internal structure of VolView, nor is VolView exposed at all to the existence of ITK methods. A plain C-language interface has been defined in VolView, that allows very generic methods to interface with the internal data representation. Another interesting aspect of the integration is that it is implemented in the form of run-time plugins. This means that any number of ITK methods and procedures can be added dynamically. The only requirement is that the shared libraries in which the ITK methods are implemented must be placed in a specific directory where VolView will look for them.

The execution of a plugin consists of three key functions: *Initialization*, *UpdateGUI*, and *ProcessData*. Upon loading the plugin the Initialization function is called once to specify the static properties of the plugin. This includes the plugin’s terse and full documentation, name, support of in-place or in-pieces processing, if it only generates a surface, and number of GUI elements. Most of these properties are set using a generic interface than can be easily extended for future capabilities. For example:

```c
info->SetProperty(info, VVP_NAME, "Threshold");
info->SetProperty(info, VVP_GROUP, "Utility");
info->SetProperty(info, VVP_SUPPORTS_IN_PLACE_PROCESSING, "1");
info->SetProperty(info, VVP_SUPPORTS_PROCESSING_PIECES, "1");
info->SetProperty(info, VVP_NUMBER_OF_GUI_ITEMS, "3");
```

Then the UpdateGUI function is called to handle the dynamic elements of the plugin such as creating and updating the user interface and defining the properties of
Fig. 5. GUI entries shown on VolView as defined by the Plugin UpdateGUI() method. At left, parameters for the ITK FastMarching level set segmentation method. At right, parameters for the ITK GeodesicActiveContour level set segmentation method.

the output data. These are dynamic because as the input data changes (or new data is loaded) the properties may change. UpdateGUI may be invoked many times and it is guaranteed to be invoked at least once after any new data set is loaded or when the input data is changed (by a plugin for example). Construction of the GUI elements is also performed through a generic API to facilitate future capabilities. For example

```c++
info->SetGUIProperty(info, 1, VVP_GUI_LABEL, "Threshold Value");
info->SetGUIProperty(info, 1, VVP_GUI_TYPE, VVP_GUI_SCALE);
info->SetGUIProperty(info, 1, VVP_GUI_DEFAULT, "0");
info->SetGUIProperty(info, 1, VVP_GUI_HELP,
    "The value to compare against");
/* set the range of the slider based on the input volume */
sprintf(tmp, "%f %f %f",
    info->InputVolumeScalarRange[0],
    info->InputVolumeScalarRange[1], 1.0);
info->SetGUIProperty(info, 1, VVP_GUI_HINTS, tmp);
```

It is worth noting that in the example above the range of the GUI scale (a slider) is based on the actual scalar range of the current volume. If a new volume is loaded then the range of the scale will change to match the new volume. Figure 5 shows some of the GUI dynamically created by ITK plugins.

The ProcessData function is invoked when the user requests that the plugin be
Fig. 6. Data flow diagram of the volview plugin. Data is exported from VolView in a plain C-Language structure. This structure is imported into an ITK pipeline and processed using parameters transferred from VolView’s GUI. The result is exported into the plain C-Language structure and imported back into VolView.

executed. This function processes the input volume to produce the resulting surface and/or output volume. It receives a pointer to a C structure containing pointers to the input and output buffers. If the plugin supports processing in pieces then the process data function may be invoked many times, once for each piece. In the ProcessData function the values of the GUI elements are typically used to control the processing. For example:

```c
/* obtain the upper threshold from the GUI */
double v1 = atof(info->GetGUIProperty(info, 1, VVP_GUI_VALUE));
```

The ProcessData function fully encapsulates an ITK pipeline for performing the desired image processing and analysis functionalities. The pipeline is created at the beginning of the function and destroyed at the end of the function’s scope. This function should take care of the details involved in transferring data from VolView to the ITK pipeline and from the output of the pipeline back to VolView.

Figure 6 shows the data flow between VolView and the ITK plugin. Within ProcessData an ITK pipeline is instantiated and the data is processed using the parameters provided by the GUI. During the processing, ITK events are translated into VolView refresh calls that display messages and update a progress bar on the GUI. At the end of the processing, the resulting data set is exported into a C-Language structure and passed back to VolView.

The plugin has a couple of mechanisms for displaying information in VolView’s GUI. It can for example, display a message on the status bar, update a progress bar, report errors and report final data on the execution of the plugin (e.g. number of iterations of a filter, final error metric). Plugin execution can also be aborted from the GUI.
2.2 Plugin Life Cycle

The process of developing a VolView plugin is depicted in Figure 7. A single public header file defines the plain C-Language data structures used for exporting and importing data, and for transferring GUI parameters between the VolView GUI and the plugin. A plugin developer must implement a set of C-Language functions that will be invoked by VolView during its interactions with the plugin. The source code of the plugin is compiled and packaged in the form of a shared library. No libraries from VolView are required during this process. The shared library containing the plugin is finally copied into a specific directory of the VolView binary installation. This particular directory is searched by VolView at start-up time. Any shared libraries found in this directory will be dynamically loaded. The name of the library should conform to the name of the plugin initialization function.

The development cycle of a plugin is totally decoupled from VolView’s internal code. The single communication bridge between the plugins and the application is the header file defining the data and GUI structures. A developer of a new plugin, must simply implement the methods defined in the public header file.

3 Writing a Plugin

This section describes the basic steps required for writing a new plugin. The example here illustrates the implementation of a simple filter having only one parameter.
to be set from the GUI.

3.1 Defining the plugin name

The plugin name will determine the name of the shared library used for deployment. It will also determine the name of the initialization function. For example a plugin named vvITKGradientMagnitude will be deployed in a shared library with name libvvITKGradientMagnitude.so in Unix, and vvITKGradientMagnitude.dll on MS-Windows. Its initialization function will be called vvITKGradientMagnitudeInit().

3.2 The initialization function

The initialization function of the plugin must conform to the following API

```c
extern "C" {
    void VV_PLUGIN_EXPORT vvITKGradientMagnitudeInit(vtkVVPluginInfo *info)
    {
    }
}
```

where the symbol VV_PLUGIN_EXPORT and the structure vtkVVPluginInfo are both defined in the public header file vtkVVPluginInfo.h.

This initialization function is invoked by VolView at start-up time, just after the shared library has been dynamically loaded.

The typical content of this function is shown below.

```c
{
    vvPluginVersionCheck();

    // setup information that never changes
    info->ProcessData = ProcessData;
    info->UpdateGUI = UpdateGUI;
    info->SetProperty(info, VVP_NAME, "Gradient Magnitude IIR (ITK)");
    info->SetProperty(info, VVP_GROUP, "Utility");
    info->SetProperty(info, VVP_TERSE_DOCUMENTATION,
                      "Gradient Magnitude Gaussian IIR");
    info->SetProperty(info, VVP_FULL_DOCUMENTATION,
                      "This filter applies IIR filters to compute the equivalent of convolving
                      the input image with the derivatives of a Gaussian kernel and then
                      computing the magnitude of the resulting gradient.");
    info->SetProperty(info, VVP_SUPPORTS_IN_PLACE_PROCESSING, "0");
    info->SetProperty(info, VVP_SUPPORTS_PROCESSINGPieces, "0");
    info->SetProperty(info, VVP_NUMBER_OF_GUI_ITEMS, "1");
    info->SetProperty(info, VVP_REQUIRED_Z_OVERLAP, "0");
    info->SetProperty(info, VVP_PER_VOXEL_MEMORY_REQUIRED, "8");
}
```
First, the macro `vvPluginVersionCheck()` must be called in order to verify that the plugin API conforms to the current version of VolView’s binary distribution. When the versions do not match, the plugin is not executed and an error message is reported to the user at run-time.

Once the version number has been validated, the `info` structure is initialized. The `ProcessData` is set to the pointer of the function that will perform the computation on the input data. Setting the function as a function pointer allows a lot of freedom on the implementation of the function.

The `UpdateGUI` pointer in the `info` structure is also set to a function pointer. The role of this `UpdateGUI` function is to initialize the GUI parameters from the Plugin. In this function we specify all the properties of all the GUI widgets related to the parameters of the plugin.

The function `SetProperty()` is used to define general properties of the plugin. Some of these properties are displayed on the GUI as informative text. For example, the textual name of the plugin, a terse documentation and an extended documentation. The properties are identified by tags. This enforces further the decoupling between the internal representation of information in VolView and the structure of code in the plugin. For example the tag `VVP_NAME` specifies that the string being passed as third argument of the `SetProperty()` method should be used for the text label of the plugin in the GUI.

Other non-GUI properties are also set with this method. For example, we specify whether this filter is capable of performing in-place processing or not, whether it allows to process data in pieces (streaming) or not. We also provide an estimation of the memory consumption that will result from the execution of the filter. This last information is provided in the form of an estimation of number of bytes to be used per pixel of the input data set. Memory consumption estimation is of fundamental importance for the successful coupling between VolView and the plugin. VolView will use this factor for making sure the that system has enough memory for completing the processing of the plugin. Note that this estimation is not based on the size of the final data set produced as output but on the total amount of memory required for intermediate processing. In other words, it should provide the peak of memory consumption during the plugin execution. The initialization function, in this case `vvITKGradientMagnitudeInit()`, will be called only once during the start-up process of VolView.

### 3.3 The `ProcessData` function

The function `ProcessData()` is the one that actually performs the computations on the data. The signature of this function is
static int ProcessData(void *inf, vtkVVProcessDataStruct *pds)

where the first argument is actually a pointer to a vtkVVPluginInfo structure and can be downcasted to it by doing

vtkVVPluginInfo *info = (vtkVVPluginInfo *)inf;

The second argument to ProcessData() is the vtkVVProcessDataStruct. This structures carries the information on the data set to be processed, including the actual buffer of pixel data, the number of pixels along each dimensions in space, the pixel spacing and the pixel type among others. Of particular interest in this structure are the members inData that is a pointer to the data buffer of the input data set, and the outData that is a pointer to the output data set buffer. The ProcessData() function is expected to extract the data from the inData pointer, process it and store the final results in the outData buffer.

The typical starting code of this function involves extracting the meta information about the data set. The following code shows for example, how to extract the dimensions and spacing of the data set from the vtkVVProcessDataStruct and vtkVVPluginInfo structures.

```cpp
SizeType size;
IndexType start;
double origin[3];
double spacing[3];
size[0] = info->InputVolumeDimensions[0];
size[1] = info->InputVolumeDimensions[1];
size[2] = pds->NumberOfSlicesToProcess;
for(unsigned int i=0; i<3; i++)
{
    origin[i] = info->InputVolumeOrigin[i];
    spacing[i] = info->InputVolumeSpacing[i];
    start[i] = 0;
}
```

Using this information, the image data can be imported into an ITK image using the itkImportImageFilter.

```cpp
RegionType region;
region.SetIndex( start );
region.SetSize( size );
m_ImportFilter->SetSpacing( spacing );
m_ImportFilter->SetOrigin( origin );
m_ImportFilter->SetRegion( region );
m_ImportFilter->SetImportPointer( pds->inData, totalNumberOfPixels, false );
```

The output of the import filter is then connected as the input of the ITK data pipeline and the pipeline execution can be triggered by calling Update() on the last filter.

```cpp
m_FilterA->SetInput( m_ImportFilter->GetOutput() );
```
m_FilterB->SetInput( m_FilterA->GetOutput() );
m_FilterC->SetInput( m_FilterB->GetOutput() );
m_FilterD->SetInput( m_FilterC->GetOutput() );
m_FilterD->Update();

Finally the output data can be copied into the pointer provided by VolView. This is typically done using an ITK image iterator that will visit all the pixels.

    outputImage = m_Filter->GetOutput();
    typedef itk::ImageRegionConstIterator< OutputImageType > OutputIteratorType;
    OutputIteratorType ot( outputImage, outputImage->GetBufferedRegion() );
    OutputPixelType * outData = static_cast< OutputPixelType * >( pds->outData );
    ot.GoToBegin();
    while( !ot.IsAtEnd() )
    {
        *outData = ot.Get();
        ++ot;
        ++outData;
    }

When memory consumption is critical, it is more convenient to actually connect the output memory buffer provided by VolView to the output image of the last filter in the ITK pipeline. This can be done by invoking the following lines of code before executing the pipeline.

    m_FilterD->GetOutput()->SetRegions(region);
    m_FilterD->GetOutput()->GetPixelContainer()->SetImportPointer(
        static_cast< OutputPixelType * >( pds->outData ),
        totalNumberOfPixels, false);
    m_Filter->GetOutput()->Allocate();

The current distribution of ITK provides support for not having to write this same code for each new plugin. A templated class is available for providing these basic services. New plugins only need to define their own ITK pipelines and invoke the methods of the base class in the appropriate order.

3.4 Refreshing the GUI

Given that the execution of most image processing algorithms take considerable time on 3D data sets, it is important to provide some feedback to the user as to how the processing is progressing. This also gives a chance to the user for cancelling the operation if the expected total execution time is excessively long.

This notification can be done from the ProcessData() by calling the UpdateProgress() method of the vtkVVPluginInfo structure.

    float progress = 0.5; // 50% progress
This function will update the progress bar on VolView’s GUI and will set the last string as a message in the status bar. There is a balance to be found concerning the frequency with which this function should be invoked. If invoked too often, it will negatively impact the performance of the plugin since a considerable amount of time will be spent in GUI refreshing. If not called often enough, it will produce the impression that the processing is failing and the application is not responding to the user commands anymore.

This concludes our description of the integration between ITK and VolView in the form of plugins. Although a good number of details have been omitted here, this description should provide a good starting point for guiding developers in the process of writing new plugins. You are always encouraged to post any questions to the ITK users-list where we anticipate that the topic of VolView plugins will become a common subject.

4 Plugins Available

The following list details the ITK plugins that are available at the time of submission of this document. The modularity of the plugins facilitates the growth of the number of plugins because anybody can write new plugins and make them available to the public. The entries in this list specify if they are build using components from the Visualization Toolkit (VTK), the Insight Toolkit (ITK) or they are written in plain C. It is entirely possible to combine ITK and VTK components in the same plugin. The grouping of plugins in the list below is also reflected in the GUI by setting them in menus and submenus.

- Noise Suppression
  - Gaussian smooth (VTK)
  - Median Filter (VTK)
  - Median Filter (ITK)
  - Dilate Filter (ITK)
  - Erode Filter (ITK)
  - Curvature Flow Filter (ITK)
  - Curvature Anisotropic Diffusion Filter (ITK)
  - Gradient Anisotropic Diffusion Filter (ITK)
- Segmentation Models
  - Deformable model (ITK)
- Edge Detection

\[2\] We anticipate that this list will grow rapidly in the time following the releases of ITK 1.4 and the free version of VolView.
Canny edge detection (ITK)

- Utility
  - Gradient Magnitude (VTK)
  - Gradient Magnitude (ITK)
  - Gradient Magnitude IIR (ITK)
  - Crop (C)
  - Threshold (C)
  - Merge Volumes (C)
  - Boundary (C)
  - Component Arithmetic (C)

- Level Sets Segmentation
  - Geodesic Active Contour Filter (ITK)
  - Geodesic Active Contour Module (ITK)
  - Watershed Module (ITK)
  - Shape Detection Module (ITK)
  - Fast Marching Filter (ITK)
  - Fast Marching Module (ITK)
  - Canny Segmentation Level Set Module (ITK)

- Region Growing Segmentation
  - Confidence Connected (ITK)
  - Isolated Connected (ITK)
  - Connected Threshold (ITK)
  - RGB Confidence Connected (ITK)

- Intensity Transformations
  - Sigmoid (ITK)
  - Intensity Windowing (ITK)

- Surface Generation
  - Iso-surface (VTK)
  - Anti-Alias (ITK)

The term Module is used in this list to indicate the cases in which pre-processing and post-processing filters were packaged in single plugin with the aim of providing a self-contained specific functionality.

5 Application to Abdominal Aortic Aneurysm Segmentation

The capabilities of ITK plugins for solving real scale medical image problems are demonstrated in this section in an application to the segmentation of Abdominal Aortic Aneurysms (AAA) from CT scans. The data for this experiments was provided by SINTEF, Unimed, Norway. The segmentation procedure was developed in a collaboration between Kitware and SINTEF.

AAAs are a chronic degenerative disease with life-threatening implications. While
AAAs are thought to arise through a localized form of arterial wall injury super-imposed on various predisposing factors, their natural history is one of progressive structural deterioration, gradual expansion, and eventual rupture. Pathologic processes contributing to the changes observed in AAAs include chronic inflammation, destructive remodeling of the extracellular matrix, and depletion of vascular smooth muscle cells. These changes result in progressive aortic dilatation accompanied by alterations in vessel geometry, redistribution of hemodynamic wall stresses, and diminished tensile strength (Steinmetz et al., 2003).

A common treatment for AAA consist on the insertion of a vascular graft that creates a barrier between the blood flow and the weakened vascular wall of the aneurysms. Imaging the patient before and after the graft insertion is fundamental for controlling the efficacy of the procedure. The images considered here correspond to CT scans acquired after the graft has been implanted. Figure 8 shows a CT scan of the abdominal region. The network-like structure of the graft is visible on slices
Fig. 9. Results of the segmentation of AAA data. The upper-right image presents a combination of volume rendering and polygonal rendering. The regions in red correspond to the volume rendering of the original CT data. The polygonal data is the surface of the segmentation obtained with ITK plugins. The three images on the bottom show the superposition of the segmentation boundary on top of the original data. The boundary width has been increased in order to enhance its visibility.

The aorta and the aneurysm were segmented using exclusively a sequence of operations combining ITK plugins and VolView native plugins. We anticipate that the entire procedure will be rewritten as a very specialized plugin in which most of the filter parameters will be pre-defined to values tailored for this particular medical application and only the remaining relevant free parameters will be exposed to the user. This is indeed an example of how the plugins can be used for rapidly prototyping an image analysis application and then proceed to a fully customized implementation.

Figure 10 illustrates the procedure for segmenting the aorta and the aneurysm presented in Figure 9. Details on the operations performed in this procedure are pe-
Fig. 10. Procedure used for segmenting the abdominal aorta and the aneurysm from CT scan dataset using ITK plugins.

Crop region of interest. The original data set covered a large region around the aorta and the aneurysm. In order to speed up computations this original data was cropped. A region of interest was selected using the interactive cropping option in VolView and then executing the Crop plugin in the Utility group. Figure 8 shows a rendering of the data set after the cropping operation.

Smoothing with Gradient Anisotropic Diffusion. The original CT data has a significant amount of texture that can be classified as noise. Since the segmentation methods to be used in the following steps rely on the homogeneity of the image, it is desirable to attenuate the noise present in the image. The Gradient anisotropic diffusion filter implemented in ITK performs edge-preserving smoothing. This type of filtering is described in (Perona and Malik, 1990; Whitaker and Xue, 2001). The advantage of this smoothing is that noise is attenuated in homogeneous regions
without significantly degrading the edges of anatomical structures. The parameters used for this image were: Iterations=5, Conductance=3.0, TimeStep=0.05.

**Manual selection of seed points in the Aorta** Using the *Markers* menu interface defined in *VolView* we selected a number of seed points along the medial line of the Aorta, including the bifurcation on the inferior region. Approximately 15 seed points were selected for this segmentation. Plugins have access to the internal *VolView* data structure used for representing 3D markers. Some segmentation filters use the markers as seed points. This is the case for both of the segmentation filters used here, the region growing method *Confidence Connected* and the level set method *Fast Marching*.

**Confidence Connected Region Growing filter.** This ITK plugin was executed using as input the image smoothed with anisotropic diffusion and a set of seed points (3D markers) placed along the medial line of the aorta. The algorithm uses the seed points to initialize a region growing process in which the inclusion of new pixels on the region is regulated by the statistical characteristics of the gray levels in the region. The following parameters were selected for running this plugin: Number of iterations=2, Variance multiplier=1.5, Initial neighborhood=2.

**Median Filter** The binary image resulting from the region growing segmentation has abundant noise in the boundaries as well as multiple small internal holes. In order to smooth it while still having a binary image, the Median filtering plugin was used. When a median filter is applied on a binary image it behaves like a majority voting cellular automata and its effects are similar to a curvature weighted front propagation (Wolfram, 1994). Thus, it can be used for fusing structures and removing small details.

**Manual selection of seed points in the Aneurysm** Given that in this data set a graft stent has already been implanted in the patient, blood flow is minimum in the region between the aneurysm wall and the graft. Therefore the contrast agent does not penetrate this region and the intensity level under CT is lower than the one of the blood flowing in the aorta. The segmentation of the aneurysm should be done independently of the segmentation of the aorta due to the large difference in they intensity distributions. A level set method is used here to segment the Aneurysm. This level set method requires the definition of an initial set of seed points. Using the *Markers* menu interface defined in *VolView* we selected a number of seed points in the region between the aneurysm wall and the graft. Approximately 10 seed points were selected for this segmentation.

**Fast Marching Level Set.** The fast marching level set method is described in Adalsteinsson and Sethian (1995) and Sethian (1996). It was selected for performing the segmentation of the aneurysm because it has very low computational times. The filter was run with parameters: Maximum value 80, Lower basin=0.1, Lower boundary=20.0, Sigma=1.0. This segmentation method was applied using the *Fast*
Marching Module plugin, which provides the necessary pre-processing and post-processing for this filter. As part of the pre-processing, the image is smoothed with a Gaussian kernel, its gradient magnitude is computed, and its intensity is reversed using a Sigmoid filter. The time-crossing map produced by the fast marching filter is post-processed using simple intensity window in order to produce high intensities inside the segmented object and lower intensities elsewhere. The segmentation of the aneurysm is very underestimated on the regions of high curvature. This defect is corrected after fusing this segmentation with the one of the aorta.

**Maximum Intensity.** The result of the Aorta segmentation and the Aneurysm segmentation were fused together using a \textit{Max} filter. This filter computes the output pixels as the maximum value of the homologous pixels in the two input images. The result in this case is a single-component data set combining the segmentation of the aorta and the aneurysm. In terms of Fuzzy logic, this can be seen as an AND operation.

**Geodesic Active Contours level set.** After both segmentation have been combined, they need to be blended together. This can be done by generating a level set from the segmentation and making it evolve using the geodesic active contour algorithm. This algorithm allows to define a weighting for the curvature term on the computation of the front propagation speed, as well as an advection term based on a vector field (Sethian, 1996). In the ITK implementation, the vector field is computed using the gradient of the gradient magnitude. This vector field points always to the edges of the input image. This module was executed done with a curvature scaling = 4.0, iterations=400 and maximum RMS error = 0.01.

**Thresholding** The resulting Level set is thresholded in order to produce a binary image representing the zero set.

**Median Filter** Three iterations of Median filter are applied with kernel size 5x5x5 in order to smooth and fuse the two structures. By controlling the number of iteration that this filter is run, we can control how much smoothing and fusion is performed in the image.

The binary data set resulting from this medial filter is presented here as the final segmentation of the aorta plus aneurysm system.

Figure 8 presents a visualization of the cropped AAA CT scan used as input to the segmentation process. The region of interest has been reduced to the abdominal section between the kineys and the hip. Figure 9 presents the same data set with the overlay of the segmentation obtained with ITK plugins.

In order to support evaluation of segmentation and registration data \textit{VolView} allows to merge two or more data sets into a single data set of multiple-components. With this mechanism it is possible to simultaneously visualize the results of a segmentation method overlayed on top of the original data, or visualize a registered
image superimposed to the image used as a reference. This multi-component visualization feature is exploited in Figure 9 in order to present the segmentation of the aorta and the aneurysm superimposed to the original CT scan data set.

6 Future Work

Future work on VolView will be focused on the following aspects

Create support for an open community. This will include an open repository for new plugins. They would be made available in the forms of source code and/or compiled shared libraries. A common repository of Data will be created for providing basic examples of commonly used image modalities. A periodic, probably weekly, event will be organized in which users will be able to submit medical image problems and those problems will be discussed by members if the ITK and VTK communities.

Provide tutorials for creating new plugins

Create plugins for image registration. Current plugins are focused on segmentation.

Develop plugins as commercial products

Extend the remote collaboration capabilities of VolView. In particular the option of broadcasting will be added. Currently only peer-to-peer communication is supported. Support for chat exchange will also be added.

7 Acknowledgements

The data used for the segmentations presented in this paper were provided by SINTEF, Unimed. The integration of ITK plugins into Volview was funded by the U.S. National Library of Medicine (NIH) under contract NLM XXX-XXX-XXX. and NLM (Original ITK contract XXX-XXXX-XXX).

References


