Journal of Digital Imaging

Rapid Development of Medical Imaging Tools with Open-Source Libraries

Jesus J. Caban,[1,3] Alark Joshi,[1] and Paul Nagy[2]

Received: 18 June 2007; Revised: 12 July 2007; Accepted: 15 July 2007

Rapid prototyping is an important element in researching new imaging analysis techniques and developing custom medical applications. In the last ten years, the open source community and the number of open source libraries and freely available frameworks for biomedical research have grown significantly. What they offer are now considered standards in medical image analysis, computer-aided diagnosis, and medical visualization. A cursory review of the peer-reviewed literature in imaging informatics (indeed, in almost any information technology-dependent scientific discipline) indicates the current reliance on open source libraries to accelerate development and validation of processes and techniques. In this survey paper, we review and compare a few of the most successful open source libraries and frameworks for medical application development. Our dual intentions are to provide evidence that these approaches already constitute a vital and essential part of medical image analysis, diagnosis, and visualization and to motivate the reader to use open source libraries and software for rapid prototyping of medical applications and tools.

KEY WORDS: Open source, image processing, programming language

INTRODUCTION

Rapid development of software programs, tools, and applications is essential to the advancement of research and innovation in medical imaging. Open source libraries and freely available frameworks play a crucial role in many biomedical and radiologic advances. Open source programs are usually developed as a public collaboration and made available for use, modification, and redistribution. Currently, even the largest commercial companies in medical imaging contribute to and rely on open source software to develop flexible and robust systems.[1,2]

A growing amount of interest is focused on developing new techniques, algorithms, and applications that can be used with medical images, and the burden of proof for these innovations falls on the researchers. The development of robust and stable applications to load different medical image modalities, manipulate 2-dimensional/3-dimensional images, convert images, and effectively visualize them can take a significant amount of time. A clear need exists for tools and libraries that can push forward both the development and certainty of medical imaging by sparing researchers the time and effort required to revisit already-solved problems or to redevelop existing programs.

Several open source libraries and frameworks play particularly important roles in the rapid development of medical imaging tools. Moreover, extensive support is now provided by universities, federal agencies, and companies to the development of open tools, libraries, and software programs for biomedical research and innovation.[3] Such commitment has made a number of flexible

[1]*From the Department of Computer Science, University of Maryland, Baltimore County, Baltimore, Maryland, USA.*

[2]*From the Associate Professor of Radiology, University of Maryland School of Medicine, Baltimore, Maryland, USA.*

[3]*From the Computer Science and Electrical Engineering, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD, 21250, USA.*

Correspondence to: Jesus J. Caban, Computer Science and Electrical Engineering, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD, 21250, USA; e-mail: caban1@cs.umbc.edu

Online publication 7 August 2007

doi: 10.1007/s10278-007-9062-3

and robust application program interfaces (APIs) and libraries freely available for medical image analysis and processing.

To motivate the reader to investigate and use these freely available libraries and software programs for custom-designed medical imaging tools, we survey and compare several open source libraries. In doing so, we explain the importance of these libraries for medical imaging and weigh the advantages and limitations of each. The overall intent of this paper is to show the benefits of open source libraries for rapid development of biomedical applications and, in the process, decrease the frequency that developers and researchers find themselves reinventing the imaging informatics wheel.

OPEN SOURCE LIBRARIES

A number of open source libraries are available for medical imaging processing and analysis. Many smaller open source libraries are limited by excess specificity, inflexibility, and lack of cross-platform access. The four related cross-platform libraries profiled here, VTK, ITK, KWWidgets, and IGSTK, are widely used and well tested, guaranteeing their robustness, flexibility, and extensibility—characteristics that establish them as standard setters.

Visualization Toolkit (VTK)

The Visualization Toolkit (VTK), a widely used library for visualization, is a primary resource for achieving rapid development of medical imaging tools in a cost-effective way.[4] VTK contains a number of functionalities for 2-D/3D image processing, isosurface generation, and 3D volumetric visualization. Developed in C++, the toolkit provides a number of high-level classes, extensive documentation, and examples, thereby making it easy for practitioners and developers to use its codebase. To facilitate rapid development, VTK provides binding for popular scripting languages including support for Tcl (Tool Command Language) and Python. Kitware, Inc. (Clifton Park, NJ) provides commercial and expert support for VTK and custom-designed software for companies and customers.[5]

VTK is capable of handling different types of data, such as image data (vtkImageData), rectilinear grid (vtkRectilinearGrid), structured grid (vtkStructuredGrid), unstructured grid (vtkUnstructuredGrid), and unstructured points (vtkPolyData). A structured grid or a rectilinear grid (based on the spacing between subsequent sizes) is suitable for visualizing computed tomography (CT) and magnetic resonance (MR) imaging data, whereas an unstructured grid is more suitable for ultrasound data.

The toolkit uses a data flow approach to obtain a visual representation of underlying data. Once one understands the pipeline and architecture to VTK, it is easy to see how medical imaging software can be developed rapidly. Figure 1 shows a conceptual overview of the VTK pipeline. Data are read in the source module and then filtered by one or many of the vast number of filters available in VTK. A mapper is then used to create a visual representation that can be interacted with and transformed through the use of the actor.

Image Processing in VTK

Computed tomography (CT), magnetic resonance (MR) imaging, and ultrasound scans are among the types of medical imaging supported by VTK. A special data type (vtkImageData) performs basic image processing and handling on image data. Figure 2 shows simple source code used to render digital imaging and communications in medicine (DICOM) images with VTK. Figure 3 shows two MRI images rendered with the

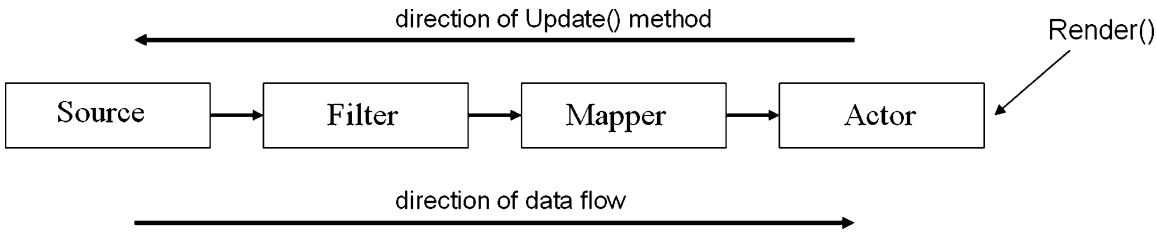

Fig 1. Conceptual overview of the VTK pipeline.

Example: Display a DICOM image using VTK and TCL

```
## FILE: Image_reader.tcl
package require vtk

# Read the image
vtkDICOMImageReader reader
reader SetFileName "$VTK_DATA_ROOT/Data/mr.001"

# Shift and scale the image
vtkImageShiftScale shiftScale
  shiftScale SetInputConnection [reader GetOutputPort]
  shiftScale SetShift 0
  shiftScale SetScale 0.07
  shiftScale SetOutputScalarTypeToUnsignedChar

# Create the RenderWindow, Renderer and Actors
vtkImageActor ia
  ia SetInput [shiftScale GetOutput]

vtkRenderer ren1
vtkRenderWindow renWin
   renWin AddRenderer ren1

# Add the actors to the renderer, set the background and size
ren1 AddActor ia
ren1 SetBackground 0.1 0.2 0.4
renWin SetSize 400 400

# render the image
renWin Render

# prevent the tk window from showing up then start the event loop
wm withdraw .
```

Fig 2. Source code for a custom-designed DICOM image viewer using VTK and TCL.

VTK program presented in Figure 2. These examples show the simplicity and rapid development possible with VTK and its Tcl binding to load, manipulate, and visualize medical images.

Other useful image-processing utilities built into VTK permit are: coloring images based on a prespecified colormap (vtkImageMapToColors), producing and visualizing histograms (vtkImageAccumulate), Gaussian smoothing (vtkImageGaussianSmooth), image reslicing/resampling along an arbitrary axis from volumes (vtkImageReslice), appending images to create a volume (vtkImageAppend), and extracting and visualizing a region of interest (vtkExtractVOI).

Volume Rendering Using VTK

Volume rendering allows visualization of 3D data such as that captured by CT and MR imaging. In volume rendering, a color and opacity are assigned to each 3D point (voxel) to allow simultaneous visualization of external and internal structure. To represent and interact with the data in the scene, the VTK pipeline uses a mapper in conjunction with vtkVolume, which in this instance replaces vtkActor. To guarantee flexibility, particularly in terms of speed and quality, VTK provides two primary volume mappers. vtkVolumeRayCastMapper is the mapper for obtaining an image using raycasting, and vtkVolumeTextureMapper2D is the mapper for texture mapping based on volume rendering.

Like vtkActor, vtkVolume contains information about the position, orientation, and scaling of data within a scene. In addition, its attribute vtkVolumeProperty represents parameters such as color and opacity that affect the appearance of the data. A transfer function is often attached to this attribute

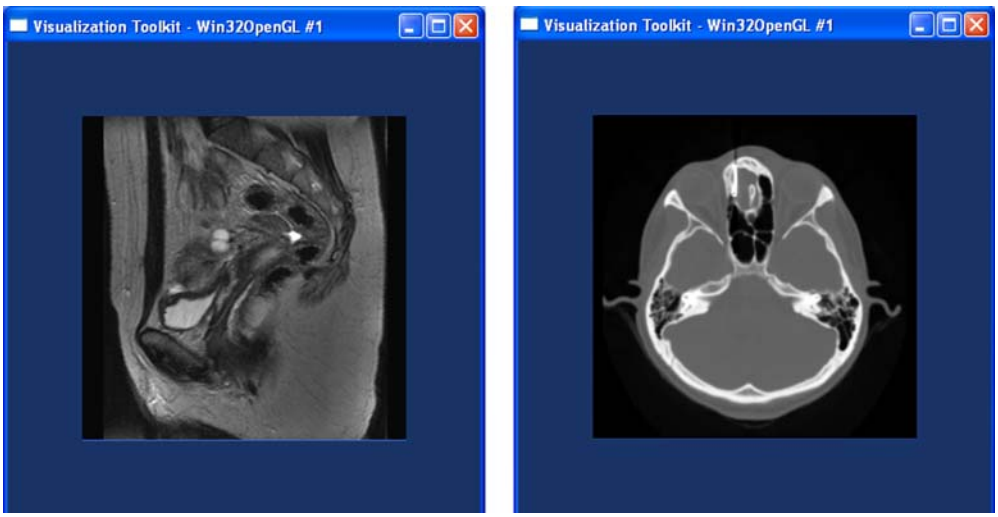

Fig 3. MR images rendered with VTK and its Tcl binding using the source code presented in Figure 2.

and used to more specifically define the appearance of the volume properties, thus making possible translucent skin, opaque skulls, and red vessels.

Figure 4 shows a schematic of the VTK volume-rendering pipeline. The volume rendering pipeline starts with the data reader. A transfer function is used to map intensity values to color and opacity. The composite function defines the order of interpolation and classification used by the raycast function. Then, a mapper assembles all the information from the compositing function and reader and sends it to vtkVolume, which renders the resulting volume onto the screen. Figure 5 includes the TCL source code to create a VTK volume rendering. Figure 6 shows two volumes rendered with VTK.

Insight Registration and Segmentation Toolkit (ITK)

The open-source Insight Registration and Segmentation Toolkit (ITK) expands the possibilities of medical image processing.[6] Developed and implemented in C++, ITK guarantees cross-platform support by relying on CMake for the compilation and configuration process. To enable and support flexibility and rapid development, ITK has wrappers for Java, Tcl, and Python. ITK provides extensive segmentation, registration, and image-filtering techniques, but does not provide graphical interface or methods for visualizing data. There is, however, a well-defined and established process available to integrate the power of ITK with the robustness of VTK for visualization.

ITK was developed from the concept of generic programming and efficient memory management techniques. Generic programming allows the effective reuse of software components by abstracting core classes and permitting the same software modules to be used with different data types. For memory efficiency, ITK uses smart pointers with reference counting. Each object, such as 2D and 3D DICOM images, has a counter with the number of references to that specific instance. When the reference goes to zero, the object destroys itself. This memory management technique gives ITK the flexibility, robustness, and efficiency to handle large and time-variant data sets.[7]

ITK follows a simple data flow and processing pipeline. The general idea of the pipeline is that the user defines process objects that operate on specific data objects. Images, polynomial meshes, and point clouds are represented as data objects, whereas image filters, processing algorithms, and registration techniques are represented as process objects. Figure 7 shows the general data flow and pipeline followed by ITK.

The generic programming style, flexibility, and robustness of ITK can be seen in the data objects. In ITK, the data object itk::Image represents an *n*-dimensional sample of data, and the same function (method) can be used to handle 2D joint photographic experts group (JPEG) images with 8-bit pixels as well as 4D functional MR imaging data sets with 12 bits per voxel.

The second class of objects within the ITK pipeline is the process objects. An ITK process object is a class that operates on data objects such as itk:Image to transform the data, analyze the data, or produce new data objects. ITK divides the process object class into three groups: sources, filters, and mappers.

Data sources are divided into image readers and writers. To enable rapid development and prototyping of medical image applications, ITK supports a number of file formats and image modalities, including DICOM, PNG, VTK, BMP, JPEG, Siemens, Tiff, RAW, GE4x, and many others. The user frequently calls on itk::ImageFileReader and itk::ImageFileWriter to read and write images, and, behind the scenes, the itk::

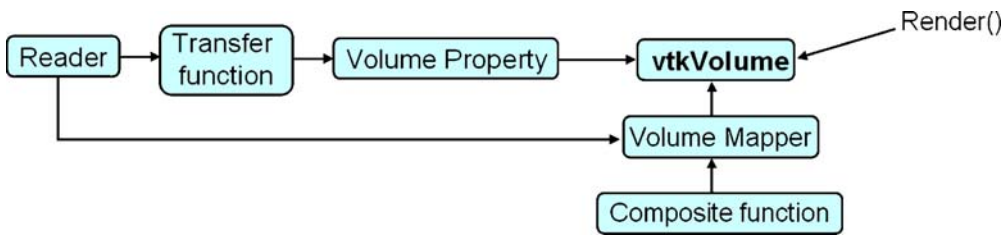

Fig 4. Overview of the volume-rendering pipeline in VTK.

Example: Volume Rendering using VTK and TCL

```
# This is a simple volume rendering example that uses a
# vtkVolumeRayCast mapper

package require vtk
package require vtkinteraction

# Create the standard renderer, render window
# and interactor
vtkRenderer ren1
vtkRenderWindow renWin
    renWin AddRenderer ren1
vtkRenderWindowInteractor iren
    iren SetRenderWindow renWin

# Create the reader for the data
vtkStructuredPointsReader reader
    reader SetFileName "$VTK_DATA_ROOT/Data/foot.vtk"

# Create transfer mapping scalar value to opacity
vtkPiecewiseFunction opacityTransferFunction
    opacityTransferFunction AddPoint  50   0.0
    opacityTransferFunction AddPoint  100   0.3
    opacityTransferFunction AddPoint  255  1.0

# Create transfer mapping scalar value to color
vtkColorTransferFunction colorTransferFunction
    colorTransferFunction AddRGBPoint      0.0 0.0 0.0 0.0
    colorTransferFunction AddRGBPoint     50.0 .0 0.0 0.0
    colorTransferFunction AddRGBPoint    100.0 .5 0.25 0.15
    colorTransferFunction AddRGBPoint    255.0 1.0 0.5 0.25

vtkPiecewiseFunction gradientTransferFunction
      gradientTransferFunction AddPoint  10   0.0
      gradientTransferFunction AddPoint  55  1.

# The property describes how the data will look
vtkVolumeProperty volumeProperty
    volumeProperty SetColor colorTransferFunction
    volumeProperty SetScalarOpacity opacityTransferFunction
    #volumeProperty SetGradientOpacity
gradientTransferFunction
    volumeProperty ShadeOn
    volumeProperty SetInterpolationTypeToLinear

# The mapper / ray cast function know how to render the data
vtkVolumeRayCastCompositeFunction  compositeFunction
compositeFunction SetCompositeMethodToClassifyFirst
vtkVolumeRayCastMapper volumeMapper
    volumeMapper SetVolumeRayCastFunction compositeFunction
    volumeMapper SetInputConnection [reader GetOutputPort]

# The volume holds the mapper and the property and
# can be used to position/orient the volume
vtkVolume volume
    volume SetMapper volumeMapper
    volume SetProperty volumeProperty

ren1 AddVolume volume
ren1 SetBackground 1 1 1
renWin SetSize 600 600
renWin Render

proc TkCheckAbort {} {
  set foo [renWin GetEventPending]
  if {$foo != 0} {renWin SetAbortRender 1}
}
renWin AddObserver AbortCheckEvent {TkCheckAbort}

iren AddObserver UserEvent {wm deiconify .vtkInteract}
iren Initialize

wm withdraw .
```

Fig 5. Volume rendering using VTK and Tcl.

ImageIO class picks the corresponding file format, compression, and low-level details required to load or write the corresponding image.

Once a data object has been loaded, filters and image processing algorithms are used to manipulate the data. ITK provides a number of filters, registration, and segmentation algorithms to enhance and process medical images. Examples of image filters implemented in ITK include image thresholding, edge detection, gradient estimation, smoothing algorithms, and frequency transformations. Examples of registration techniques implemented in ITK include rigid registration, multimodal registration, multiresolution registration, and deformable registration. Some of the segmentation techniques implemented in ITK are region-growing, watersheds, level sets, and hybrid methods.

Mappers are the third classification of the ITK's process objects. Mappers terminate the data processing pipeline by outputting the data. A mapper usually has one or more data outputs; for example, a mapper writes the image data to a file and sends it to a graphical interface. Figure 7 shows the ITK pipeline followed to filter an image and save it to a file while displaying it.

Figure 8 shows the ITK C++ source code needed to apply a Gaussian smoothing algorithm to 3D volumes. The program loads an image, applies the smoothing algorithm in each direction, and then creates a new file with the smooth image.

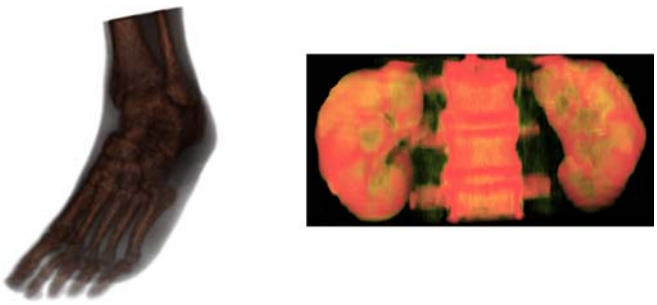

Fig 6. (*Left*) Volumetric foot rendered with the VTK/TCL source-code presented in Figure 5. (*Right*) By changing the raycast function it is possible to highlight specific anatomy.

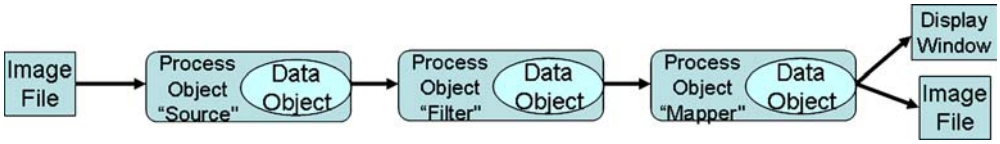

Fig 7. General data pipeline followed by ITK.

Figure 9 shows the original and resulting images after applying the code presented in Figure 8.

KWWidgets

KWWidget is a cross-platform open-source graphical interface toolkit mainly developed to support rapid development of graphical applications that use VTK and ITK.[8,9] KWWidgets was developed by Kitware, Inc. and has been used in multiple software applications, including ParaView, VolView, and 3D Slicer.[13–17]

Developed in Tcl/TK, the primary advantage of KWWidgets is that, in addition to the standard features provided by most graphical user interfaces (menus, buttons, and tabs), it also provides composite widgets to facilitate the rapid development of medical image analysis tools. New widgets provided by KWWidgets are surface material editors, video generation, transfer function

Example: Gaussian Smoothing using ITK and C++

```
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
#include "itkRescaleIntensityImageFilter.h"
#include "itkRecursiveGaussianImageFilter.h"

int main( int argc, char * argv[] ){
 if( argc < 4 ){
   printf("Usage: \n");
   printf("%s inputImageFile outputImageFile sigma\n",
argv[0]);
   return 1;
 }

 /*Define pixel type and create images*/
 const int dimension = 3;
 typedef    float    InputPType;
 typedef    float    OutputPType;

 typedef itk::Image<InputPType, dimension>
InputImageType;
 typedef itk::Image<OutputPType, dimension>
OutputImageType;

 /*Define image reader */
 typedef itk::ImageFileReader< InputImageType >
ReaderType;
 ReaderType::Pointer reader = ReaderType::New();
 reader->SetFileName( argv[1] );

 /*Define filter in each direction*/
 typedef itk::RecursiveGaussianImageFilter<
            InputImageType, OutputImageType >
FilterType;

 FilterType::Pointer filterX = FilterType::New();
 FilterType::Pointer filterY = FilterType::New();
 FilterType::Pointer filterZ = FilterType::New();

 filterX->SetDirection( 0 );   // X
 filterY->SetDirection( 1 );   // Y
 filterZ->SetDirection( 2 );   // Z

 filterX->SetOrder( FilterType::ZeroOrder );
 filterY->SetOrder( FilterType::ZeroOrder );
 filterZ->SetOrder( FilterType::ZeroOrder );
 filterZ->SetNormalizeAcrossScale( false );
 filterX->SetNormalizeAcrossScale( false );
 filterY->SetNormalizeAcrossScale( false );
 /*Data pipeline. Apply filter to X, Y, and Z*/
 filterX->SetInput( reader->GetOutput() );
 filterY->SetInput( filterX->GetOutput() );
 filterZ->SetInput( filterY->GetOutput() );

 /*Define filter's window size */
 const double sigma = atof( argv[3] );
 filterX->SetSigma( sigma );
 filterY->SetSigma( sigma );
 filterZ->SetSigma( sigma );

 /*Execute the pipeline*/
 filterZ->Update();

 /*Create output image*/
 typedef unsigned char WritePixType;
 typedef itk::Image<WritePixType, 3> WriteImgT;
 typedef itk::RescaleIntensityImageFilter
   <OutputImageType,WriteImgT> RescaleFilterType;

 RescaleFilterType::Pointer rescaler =
   RescaleFilterType::New();

 /*Rescale values to 0-255*/
 rescaler->SetOutputMinimum( 0 );
 rescaler->SetOutputMaximum(255);

 typedef itk::ImageFileWriter<WriteImgT>
       WriterType;
 WriterType::Pointer writer = WriterType::New();
 writer->SetFileName( argv[2] );

 /*Write filtered*/
 rescaler->SetInput( filterZ->GetOutput() );
 writer->SetInput( rescaler->GetOutput() );
 writer->Update();

 return 0;
}
```

Fig 8. ITK/C++ source code to apply a Gaussian filter to a given 2D/3D image.

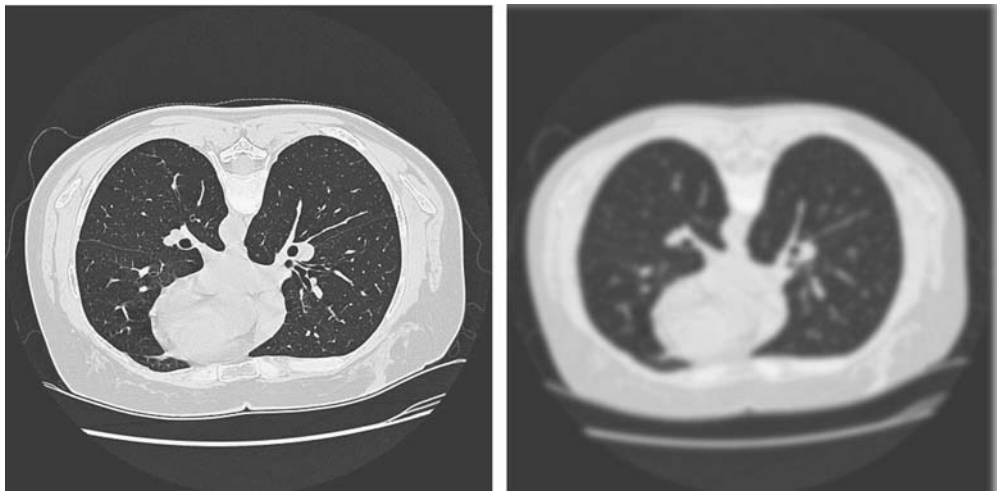

Fig 9. Resulting image (*left*) after applying the Gaussian smoothing filter presented in Figure 8 to the original image (*right*).

editor, image annotators, and others. Figure 10 shows three composite widgets included in KWWidgets to enable rapid development of medical image analysis applications.

The Image-Guided Surgery Toolkit (IGSTK)

The open source libraries we have reviewed so far mainly target effective ways to accomplish visualization, filtering, registration, and transformation of medical images. State-of-the-art interventional radiology suites are increasingly becoming integral parts of radiology centers and hospitals and require their own specialized computational and IT resources. Research, software development, and rapid prototyping for such facilities are currently possible with open source libraries.

The Image-Guided Surgery Toolkit (IGSTK) is a cross-platform open source C++ software library that provides the basic components required to develop applications for image-guided surgery and interventional radiology procedures.[10,11] IGSTK is built on top of several open source software packages, including ITK, VTK, and the Fast Light Toolkit (FLTK). In addition, cross-platform support is accomplished by relying on CMake to configure and compile the different components of the library.

Systems that integrate medical images, visualization, and external input devices have proven to be highly beneficial in medical image analysis and minimal interventional radiology. IGSTK has been designed as a collection of modules to facilitate such integration. Four core components make IGSTK an open source solution that can be used to integrate medical image analysis with external tracking systems. These components are tracker, spatial object, spatial object representation, and the view module.

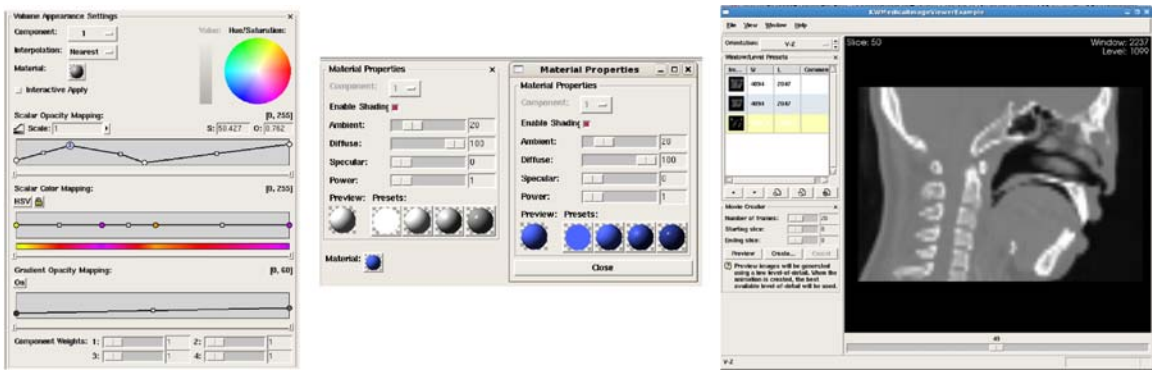

Fig 10. Three of the additional widgets provided by KWWidgets to enable rapid development of medical imaging tools.

One of the most important requirements within applications intended for interventional radiology suites is the capability to reliably correlate specific features and objects within a medical image with the same features on a patient's body. To accomplish this, it is necessary to assign a coordinate system to the images, accurately correlate the patient's anatomy to that coordinate system, compensate for deformable anatomy, and meaningfully render and show the results.

Interventional radiological tools are commonly tracked with devices that can determine the relative position of the instruments. Those devices usually provide six degrees of freedom, outputting the relative position of the instrument within the calibrated volume. The first core component of IGSTK is the tracker module. The IGSTK tracker module supports several widely used optical and magnetic trackers to enable the rapid development, integration, and testing of new techniques and algorithms. The main role of the tracker is to acquire and make the data available to other IGSTK components, such as spatial object or view. For example, by using IGSTK and a simple calibrated instrument, we can rapidly develop an application that updates the DICOM slice number according to the specific anatomical area being analyzed.

The second core component of IGSTK includes the spatial objects. The spatial object component mainly provides manipulation and interconnection between objects in a given space. The general concept behind spatial object is that by describing different sections of the visual space as a spatial object, a number of different image analysis, transformations, and studies (such image registration, atlas formation, model approximation, and simple image annotation) are now possible. The characteristics, visual representation, and rendering aspects of each spatial object are defined with the third core component of IGSTK: spatial object representations. A spatial object defines the geometry of a given object, whereas a spatial object representation describes how the object should be displayed on the screen.

The fourth core module of IGSTK is the view component. The main purpose of medical image analysis and applications for interventional radiology suites is to provide radiologists and physicians with more information to assist them during the procedure. The way in which images, tracking information, and different views are displayed plays a critical role in the overall benefit of the application. IGSTK encapsulates VTK classes into their API to robustly display, show, and illustrate medical images.

OPEN SOURCE PROGRAMS

Access to a means of avoiding the reinvention of existing tools and facilitating the identification and reuse of existing software modules are two key elements in the rapid development of applications used to process medical images. To enable those elements, medical imaging frameworks can be used. Various types of software, applications, and tools for medical image analysis are freely available. Most of the freely available software programs are limited to a specific application or type of image analysis.[12] In this section, we briefly present five open source and freely available frameworks that are robust, cross-platform, and extendable to custom-designed medical applications. The five open source frameworks we describe are Volview, Paraview, MeVisLab, SciRun, and Slicer.

Volview

Volview is a graphical interface for volume rendering and data visualization.[13] Volview was developed by Kitware and designed primarily to enable easy and effective exploration of 3D medical data sets and scientific volumetric data. The software allows the data to be visualized through synchronized multiple-view layout. The data can be annotated, and opacity, gradient, and color-transfer functions are supported. No programming skills are required to use Volview; however, the framework offers a plug-in interface that programmers can use to create, incorporate, and test their own image filters. Thirty-seven ITK and VTK filters are currently incorporated with the regular Volview framework, and these filters can be used in connection with user-defined image filters. Kitware has free and commercial versions of Volview. Figure 11a shows a CT image of the heart visualized with Volview.

Paraview

ParaView is an open source application built on top of VTK and ITK.[14] It uses the underlying functionality of VTK and adds other desirable

features, such as support for visualization using parallel processing and large data handling capacity. ParaView provides support for advanced rendering, such as tiled displays, as well as the ability to automatically switch to using parallel composite rendering when data become huge. Paraview has a number of built-in filters and image analysis techniques that can be extended through a plug-in interface to include user-defined filters. One advantage of Paraview is that it can handle and annotate vector images. That is, given a flow volumetric data set, Paraview has built-in techniques to visualize motion and annotate with arrows the direction and magnitude of the motion between timesteps. Figure 11b shows the isosurface of a CT of the head visualized with Paraview.

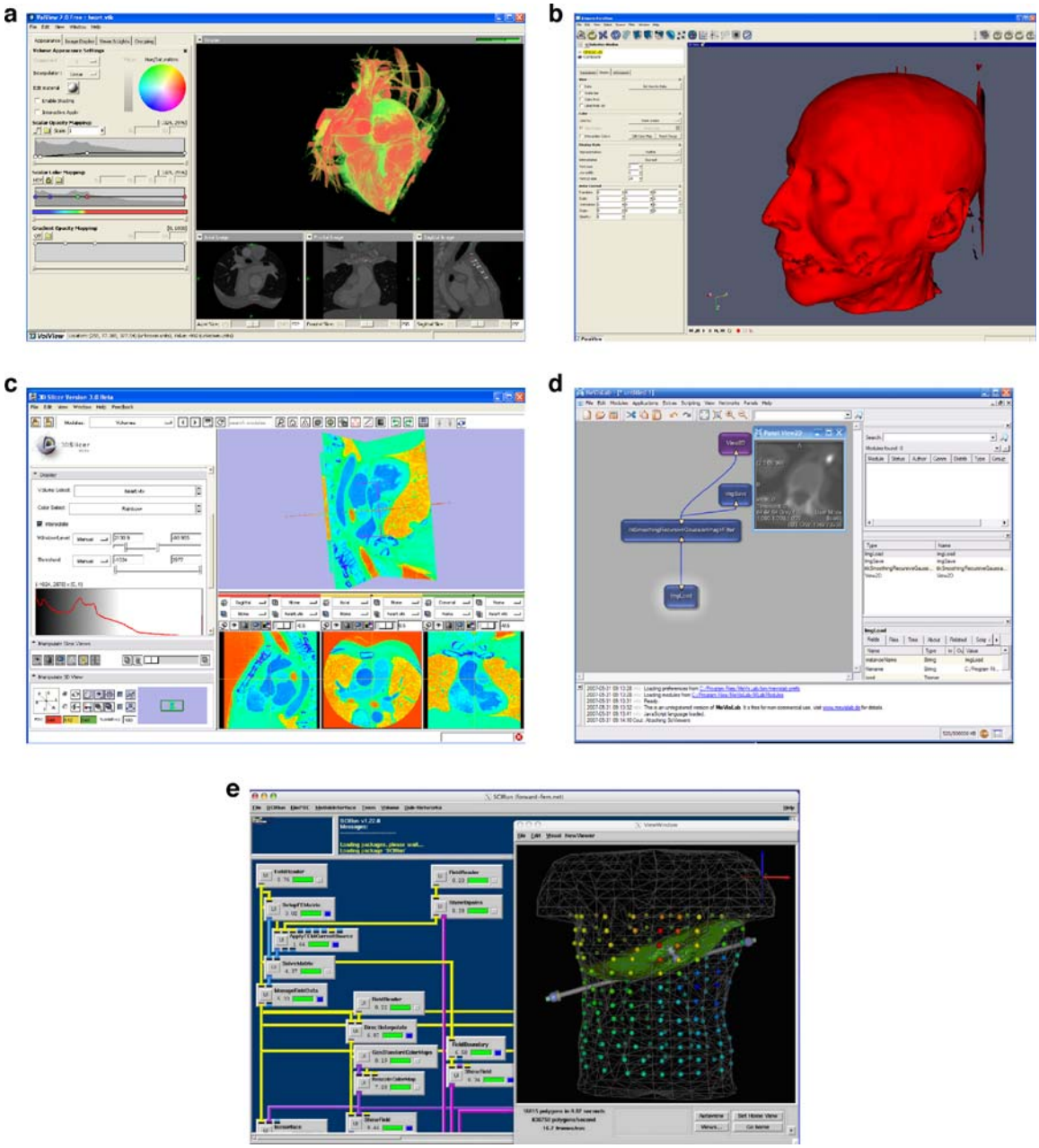

Fig 11. Five flexible and extendable open source frameworks with VTK and ITK modules.

Slicer3

Slicer (also called 3D Slicer) is an open source software developed to enable flexible radiological and biomedical medical imaging research.[15,18] Developed with KWWidgets, TCL, VTK, ITK, and IGSTK, Slicer inherits exceptional robustness, flexibility, and functionality. Slicer3 is still in beta testing and under development. However, because it has been the result of a productive collaboration between engineers and physicians, Slicer3 provides a number of modules, filtering, and components essential for medical imaging analysis. Figure 11c shows a CT of the heart visualized with Slicer3.

MeVisLab

MeVisLab is a graphical interface that uses visual dataflow programming to create custom applications and visualization tools.[15] With more than 500 modules, MeVisLab provides an interface in which the user visually connects loading, filtering, registration, and visualization modules to create a pipeline. Once the pipeline is created, the user can run the pipeline and analyze the resulting image and data. MeVisLab supports 2D, 3D, and 3D+time data and 2D/3D visualization with Open Inventor, OpenGL fragment shaders, or VTK. Figure 11d shows a four-step pipeline that loads a DICOM image, applies a filter to it, and sends the resulting image to a window and to a file.

3.5 SciRun

SCIRun is a problem-solving environment that can be used for a wide variety of applications, ranging from bioelectric field simulation and cognitive neuroscience to image processing and 3D volume rendering of medical data.[16]

Image processing in the SCIRun framework can be performed by using the native SCIRun capabilities for interpolation, gradient finding, and so on. ITK integration facilitates segmentation (threshold, confidence-connected, level sets) and registration. At the same time, MATLAB integration facilitates other customizable image processing that a user may wish to perform.

SCIRun provides extensive support for volume rendering. It features slice-based volume rendering, maximum-intensity projection (MIP)-based volume rendering, and direct volume rendering. Advanced volume rendering features, such as multidimensional transfer functions, are built into SCIRun.

The framework contains PowerApps, which are specialized programs built onto SCIRun. One such PowerApp is BioImage, a tool for visualizing regular, 3D scalar volumes such as CT and MR data. BioImage also provides a number of dynamic filters for resampling and cropping. These filters help the user accentuate important features.

BioImage also offers 2D visualization of axial, sagittal, and coronal planes. Radiologists and other biomedical practitioners can use these 2D visualizations to investigate a volume slice by slice or as MIPs and interact with them using window level. Figure 11e shows a screenshot of SciRun and a specific pipeline and its resulting image.

CONCLUSION

Open source software, libraries and APIs play a critical role in medical imaging and analysis. In this paper we described 4cross-platform, flexible, and robust open source libraries that can be used for rapid development of medical imaging tools and applications. Furthermore, we have commented on five open source frameworks that can be used to develop custom medical imaging applications and require little or no programming skills.

ACKNOWLEDGEMENTS

This work was supported by Telemedicine and Advanced Technology Research Center (TATRC) through protocol #06151004. We deeply appreciate Nancy Knight, PhD for her guidance and help in preparing and writing this manuscript.

REFERENCES

1. GE Healthcare: MicroCT Sofware: MicroView. Available at: http://www.gehealthcare.com/usen/fun_img/pcimaging/products/microview.html . Accessed on June 16, 2007

2. GE Healthcare: Preclinical Imaging. Available at: http://www.gehealthcare.com/usen/fun_img/pcimaging/products/microctscanner.html . Accessed on June 16, 2007

3. Kikinis R: The National Alliance for Medical Imaging Computing (NA-MIC). In: Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference, Aug. 8–11, 2005, p 8

4. Kitware, Inc.: VTK User's Guide Version 5. Clifton Park, NJ: Kitware, Inc., September 19, 2006

5. Kitware, Inc.: About Kitware. Available at: http://kitware.com/profile/about.html . Accessed on June 15, 2007

6. National Library of Medicine: National Library of Medicine Insight Segmentation and Registration Toolkit (ITK). Available at: http://www.itk.org . Accessed on June 15, 2007

7. Ibanez L, Schroeder W: The ITK Software Guide 2.4. Clifton Park, NJ: Kitware, Inc., November 21, 2005

8. KWWidgets Web site. Available at: http://www.kwwidgets.org . Accessed on June 15, 2007

9. Barre S: An Introduction to KWWidgets. Clifton Park, NJ: Kitware Source; July 2006

10. The Image-Guided Surgery Toolkit (IGSTK) Web site. Available at: http://www.igstk.org . Accessed on June 15, 2007

11. Gary K, Ibanez L, Aylward S, Gobbi D, Blake MB, Cleary K: GSTK: An open source software toolkit for image-guided surgery. Computer 39(4):46–53, 2006

12. Bitter I, van Uitert R, Wolf I, Ibanez L, Kuhnigk JM: Comparison of four freely available frameworks for image processing and visualization that use ITK. IEEE Trans Vis Comput Graph 13:483–493, 2007

13. Kitware, Inc. Interactive and intuitive volume visualization. Available at: http://www.kitware.com/products/volview.html . Accessed on June 15, 2007

14. ParaView: Parallel Visualization Application Web page. Available at: http://www.paraview.org . Accessed on June 15, 2007

15. Me*VisLab*: Medical Image Processing and Visualization Web page. Available at: http://www.mevislab.de . Accessed on June 15, 2007.

16. Parker SG, Johnson C: SCIRun: A scientific programming environment for computational steering. Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, 1995

17. 3D Slicer Web page. Available at: http://www.slicer.org . Accessed on June 15, 2007

18. Pieper S, Lorensen W, Schroeder W, Kikinis R: The NA-MIC Kit: ITK, VTK, pipelines, grids and 3D Slicer as an open platform for the medical image computing community. IEEE Symp Biomed Imaging 698–701, 2006