The Design and Implementation of a C++ Toolkit for Integrated Medical Image Processing and Analyzing

Mingchang Zhao, Jie Tian[1], Xun Zhu, Jian Xue, Zhanglin Cheng, Hua Zhao
Medical Image Processing Group, Institute of Automation, The Chinese Academy of Sciences, Beijing 100080, China

ABSTRACT

With the success of VTK and ITK, there are more attentions to the toolkit development issue in medical imaging community. This paper introduces MITK, an integrated medical image processing and analyzing toolkit. Its main purpose is to provide a consistent framework to combine the function of medical image segmentation, registration and visualization. The design goals, overall framework and implementation of some key technologies are provided in details, and some application examples are also given to demonstrate the ability of MITK. We hope that MITK will become another available choice for the medical imaging community.

Keywords: Visualization, Segmentation, Registration, C++ Toolkit

1. INTRODUCTION

With the application of modern medical imaging device, such as CT, MRI, Electronic endoscopy, the traditional diagnosis methods has got a revolutionary change. The techniques which use computer to assist doctors to process and analyze the medical images are called medical image processing and analyzing. They can assist doctors to make better and more accurate diagnosis. Recently people take attention to their health care more than ever, which makes researchers attach more importance to medical image processing and analyzing. Because 21st century is a century in which scientific technology related human self will get quiet great progress, we can believe that after the subject formation and development process in 20st century, medical image processing and analyzing will get into a widely application process.

Medical image processing and analyzing involve many subject, including image processing, computer graphics, pattern recognition, virtual reality and medical knowledge. Now the three main research fields are medical image segmentation, medical image registration and 3D Visualization. In the past two decades, researchers have developed many algorithms in each field, and new algorithms will emerge continuously. In addition to the effort on algorithm research, some research organization developed many software development toolkits to encapsulate existed algorithms and to avoid repeated low level coding. Among them, the most notable examples are VTK (Visualization Toolkit) [1] and ITK (Insight Segmentation and Registration Toolkit) [2]. These toolkits not only bring great convenience to the researchers in the medical imaging community, but also form a new research trend, i.e. software development in medical image processing and analyzing. In the conference of SPIE Medical Imaging 2004, there is a session called Visualization Toolkits to devote to the toolkits development; In the conference of MICCAI (Medical Image Computing & Computer Assisted Intervention) 2003, there was a workshop called Software Development Issues for Medical Imaging Computing & Computer Assisted Interventions to devote to the similar topic.

Inspired by the VTK and ITK, we have been developing an integrated medical image processing and analyzing toolkit. The name of this toolkit is MITK, which stands for Medical Imaging ToolKit. This paper will introduce the design goal, the framework and some implementation details of MITK. In Section 2, we review related works on toolkit development. In Section 3, we describe the overall design idea of MITK, including design goal and computational framework. In Section 4, we introduce the implementation of some key technology in MITK. Application examples are given in Section 5. In Section 6, we draw the conclusions and give the plans of future work.

[1] E-mail: tian@doctor.com, Web site: http://www.3dmed.net

2. RELATED WORKS

There are many software development toolkits in the medical imaging community. Because of the limit of space, we only introduce two most well known, VTK and ITK. They both affected the design of MITK.

2.1. Introduction of VTK

VTK stands for Visualization ToolKit. It is a toolkit for general 3D graphics and visualization [3] [4] [5]. It was released by Will Schroeder and Ken Martin at GE Corporate Research & Development in 1993. Initially it was the company software of the book "The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics". After 5 years practical application and when the 2nd edition of this book published in 1998, VTK had got many users and got a very big success. Now VTK is maintained by Kitware Inc. and is distributed in Open Source form, thus the researchers and developers around the world can contribute to the development of VTK. Currently the stable version of VTK is 4.2, and it provides more than 300 C++ classes and various visualization algorithms.

2.2. Introduction of ITK

ITK stands for Insight Segmentation and Registration ToolKit. It is a toolkit for medical image segmentation and registration [6]. ITK is sponsored by National Library of Medicine as a tool to support the Visible Human Project. The project began in 1999, and released ITK 1.0 in 2002. Now ITK is also maintained by Kitware Inc. and is distributed in Open Source form. Although ITK doesn't have long history as VTK, it has got many applications in the medical imaging community. For example, Medical Image Analysis will devote a Special Issue in 2004 to the topic of Segmentation and Registration using ITK. Currently the stable version of ITK is 1.4, and it provides almost all mainstream medical image segmentation and registration algorithms.

VTK and ITK have become very famous software development toolkits and they provide many conveniences to the related researchers. But they have some disadvantages which prevent them from getting more applications. Firstly, ITK doesn't provide the function of visualization. To make a medical image processing and analyzing system, we must use both VTK and ITK. VTK used the classical Object-Oriented design method, and ITK used Generic Programming. The difference in design methods cause the coding styles of VTK and ITK are quite different. For a user, to master two sets of large scale and different styles toolkits is not very easy. Secondly, VTK is not specially designed for medical image processing and analyzing, and it increase the whole size and the complexity. Finally, ITK make use of many modern C++ language features, especially the template. On the one hand, ITK can take full advantages of the Generic Programming. On the other hand, the intensive use of template and Generic Programming challenge the users of ITK and C++ compilers. Many researchers are not very good at template programming and Generic Programming, and some C++ compilers don't support template completely. All these factors restrict the application range of VTK and ITK.

3. THE OVERALL DESIGN OF MITK

The original intention of MITK is to provide medical imaging community a consistent toolkit which includes the function of segmentation, registration and visualization. The style of MITK is very similar to the style of VTK, but it's more traditional. In addition to the consistent style, MITK also brings some new features. The purpose of developing MITK is to enrich the available toolkits and provide another choice for the related researchers and developers.

3.1. Design goals of MITK

For the software design, especially the design of complex software for specific domain, a clear design goal must be set down in advance. Since the initial design, MITK always pursues the following high level design goals.

3.1.1. Consistent design style

Because the rapid evolution of C++ language in recent years, there are many different design style available by using C++ language. VTK began to develop before the release of ANSI C++ specification in 1998, so it used the classical Object-Oriented method to design the whole framework. However, ITK began to develop in 1999, and at that time, the ANSI C++ specification had been released for a period of time. So ITK used Generic Programming method to design

the framework. Generic Programming is an advanced design method, and it has many advantages over the traditional Object-Oriented method, such as elegant algorithm framework, better run-time performance, etc. But the Generic Programming requires very verbose syntax, and the program is more difficult to understand. In the decision of which design method MITK should use, we insist to use the traditional Object-Oriented method, i.e. inheritance tree and virtual function, to form the main design style. The run-time performance is improved by the optimization of key algorithms. In addition, we use Design Patterns [7] intensively to get a consistent, flexible, reusable overall framework.

3.1.2. Limited goals

MITK only focuses on the specific domain of medical image processing and analyzing. It doesn't pursue to become a general and complete toolkit, but only a focused and specific toolkit. For example, MITK only supports the visualization of regular dataset, which is the data type obtained by medical imaging device. This rule can simplify the design of MITK, and maintain MITK at a moderate scale.

3.1.3. Portability

For a toolkit to get the most widely application, the portability is a very important factor. All the code in MITK are written by using ANSI C++, and the platform dependent code are separated and minimized as possible as we can. Furthermore, MITK doesn't use the advanced template-based C++ features, so the compiler requirement is very low. Currently MITK can be compiled on most mainstream C++ compilers and can run on Microsoft Windows, Unix and Linux Operating Systems.

3.1.4. Algorithm optimization

Many algorithms in the medical image processing and analyzing, especially 3D visualization algorithms, require intensive computation and quick response of user interaction. Because the inheritance and virtual function in the Object-Oriented design method may cause additional overhead, the optimization for key algorithms is very important. Keeping MITK a moderate scale toolkit makes us to have opportunity to specially optimize some algorithms. MITK supports the modern CPU SIMD instructions set, such as SSE provided by Intel on Pentium III and later processors. Considering the portability design goal, we didn't use assembly language to code the SSE instructions set. Instead we made use of the SSE intrinsics provided by compiler. Currently we implemented SSE-accelerated matrix and vector computation, bi-linear and tri-linear interpolation computation. In addition, MITK also supports the modern programmable GPU provided by video card. For example, the recent texture mapping-based volume rendering algorithms [8] [9] have been integrated into MITK to get the real-time rendering performance.

3.2. The computational framework of MITK

Same as VTK and ITK, MITK also use the data flow model to design the computational framework. But MITK use a simplified and more traditional model to get a small and consistent overall framework.

3.2.1. Data flow model based framework

In the data flow model, the center is data processing and data and algorithm are modeled separately. It is well suitable for the application field of medical image processing and analyzing, in which there are many kind of different algorithms and different data to deal with.

Data flow model adopted by MITK is a simplified version of the model in VTK. As is shown in fig. 1, each data and algorithm is represented by an object. A data is abstracted to a Data class, while an algorithm is abstracted to a Filter class, which receives an input data object and generates an output data object. A series of algorithms can be connected into a pipeline and form a consistent computational framework. This is same to the visualization model implemented in VTK, but the difference here is that MITK doesn't provide the support of network topology, network feedback and network execution. MITK use the more traditional model and every Filter execute immediately after it's Run() member function is called. In the overall sense, MITK is a classical software development toolkit, and not an application framework.

In fig. 1, Data is the abstract class represented the medical data, and Source, Filter and Target are three types of abstract class represented different purpose algorithms. Their meanings are explained as follow.

Data: Data abstracts the attributes and methods of medical image data. For different type of medical image data, we can subclass Data to get the concrete data class. In section 3.2.2, we will give some details of the data model in MITK.

Source: Source is one type of algorithm. It has only output data, but doesn't have input data, so it represents the source of one algorithm pipeline. The purpose of Source is to generate the initial Data object to start the execution of the whole pipeline. The examples include reading files from disk to generate data, or generating data by using a algorithm.

Filter: Filter is one type of algorithm. It has a input data and a output data and represents the data processing algorithm. The most algorithms in medical image processing and analyzing can be expressed as one Filter. For the simplicity on the concept, MITK doesn't support multi-input, multi-output Filter. These type algorithms are implemented by some special concrete subclasses of Filter. In section 3.2.3, we will introduce the algorithm model in detail.

Target: Target is one type of algorithm. It has only input data, but doesn't have output data, so it represents the end of one algorithm pipeline. The purpose of Target is to put the final Data object to appropriate location and finish the execution of the whole pipeline. The examples include writing final result to disk files, or displaying final result into the screen.

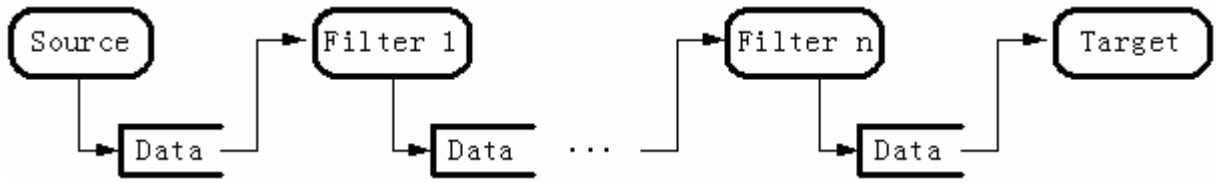

Figure 1. The pipeline of data and algorithms.

3.2.2. Data model

Considering the characteristic of data processed by medical image processing and analyzing algorithms, we get the data model of MITK, as is shown in fig. 2. Volume and Mesh are two concrete subclasses of Data and represents two different kinds of data.

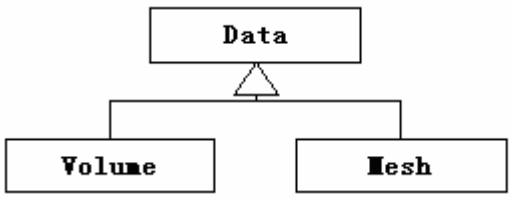

Figure 2. Data model of MITK.

Volume is a concrete data class to represent a medical image data obtained by medical imaging device. It provides an abstract for the multi-dimensional (1, 2, 3), multi-modal (CT, MRI) and regular dataset. The internal data and attributes are exposed to the algorithm object through the interface of Volume. Volume is one of the kernel objects in MITK.

Mesh is a concrete data class to represent a geometrical data. It provides an abstract for the one dimensional lines, two dimensional vector graphics and three dimensional triangular meshes. Mesh is not directly corresponding to the medical image data, but is only an intermediate result data generated by some algorithms. For the efficiency of Mesh processing algorithms, half-edge is used in MITK as internal data structure of Mesh. The internal data and attributes are exposed to the algorithm object through the interface of Mesh. Mesh is one of the kernel objects in MITK.

3.2.3. Algorithm model

Source and Target are two special purpose algorithm abstract, while Filter are the main data processing algorithm abstract. According to the data type of input and output, we get the algorithm model and show it in fig. 3. VolumeToVolumeFilter, VolumeToMeshFilter, MeshToMeshFilter and MeshToVolumeFilter are four abstract subclasses of Filter and represent four different kinds of algorithms.

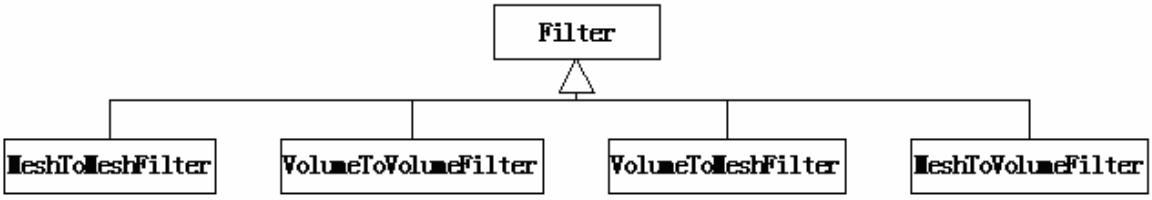

Figure 3. Algorithm model of MITK.

VolumeToVolumeFilter is an abstract algorithm class to represent an algorithm which input data and output data are all Volume. These algorithms include image processing algorithms, image registration algorithms and some image segmentation algorithms; VolumeToMeshFilter is an abstract algorithm class to represent an algorithm which input data is Volume and output data is Mesh. These algorithms include surface reconstruction algorithms and some image segmentation algorithms; MeshToMeshFilter is an abstract algorithm class to represent an algorithm which input data and output data are all Mesh. These algorithms include mesh simplification algorithms, mesh fairing algorithms, mesh subdivision algorithms, etc.; MeshToVolumeFilter is an abstract algorithm class to represent a algorithm which input data is Mesh and output data is Volume. These algorithms include distance field-based visualization algorithms and implicit surface algorithms. Each of these four abstract filters specifies the interface which the subclass must implement.

4. THE IMPLEMENTATION OF SOME KEY TECHNOLOGIES

The design of MITK involved many technologies, including the framework design and the algorithm implementation. Here we only introduce some key technologies at the high level.

4.1. Implementation of volume rendering algorithm framework

Volume rendering algorithm is the most important algorithm in scientific visualization. Because of the complexity and flexibility, implementing an efficient and flexible algorithm framework is very difficult. The volume rendering framework of MITK is based on the framework of VTK, and is enhanced to integrate the transfer function generation algorithm and multi-dimensional transfer function support into the framework.

Before the introduction of volume rendering framework, we first introduce the general rendering framework in MITK, as is shown in fig. 4. View is a subclass of Target and it is responsible for displaying the result images or 3D graphics into screen. View maintains an array of Model. It provides the interface to add a Model into the array. In the member function OnDraw() which is called when the view is updated, each model in the array is visited and its virtual function Render() is called to display itself to the screen. Model is an abstract class and represents a drawable object in a scene, which can be an image, a 3D geometrical data, or a volume. The concrete subclasses must implement the Render function to display its content.

VolumeModel is a concrete subclass of Model and it is the main component of volume rendering framework. In the implementation of its Render function, a Volume is visualized by the volume rendering algorithm. Because there exists many kinds of volume rendering algorithms, and many parameters are adjustable, the VolumeModel plays a very important role in the whole volume rendering framework. Its structure is shown in fig. 5.

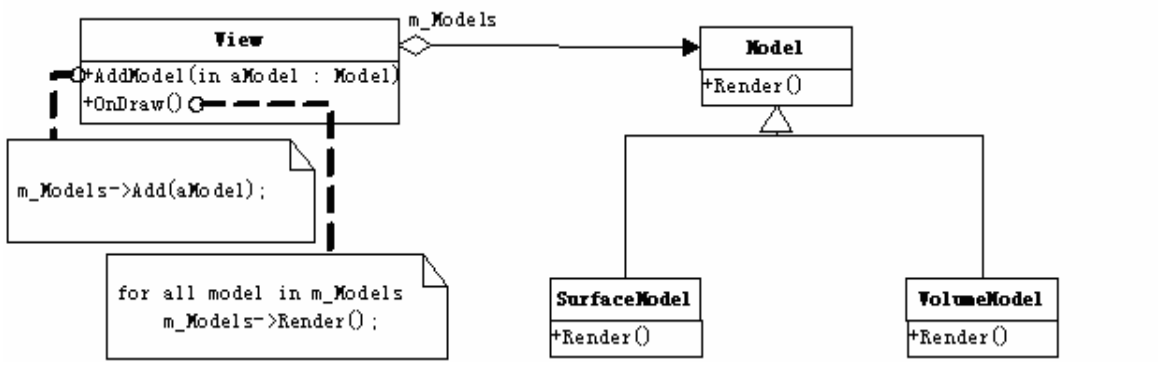

Figure 4. Rendering model of MITK.

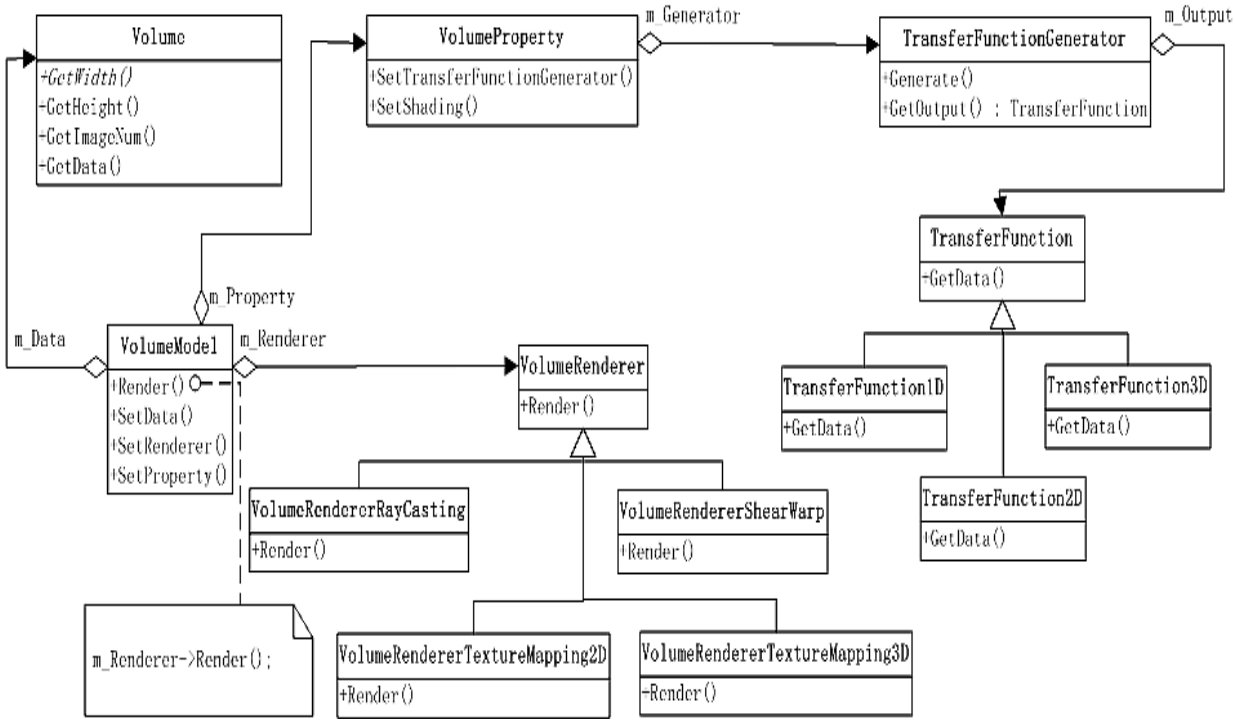

Figure 5. Volume rendering model of MITK.

VolumeModel has three class members, i.e. Volume, VolumeProperty and VolumeRenderer. Volume provides the access of medical image data. VolumeProperty provides the parameters required by volume rendering algorithms, especially the opacity transfer functions. VolumeRenderer provides the actual rendering algorithms. In fact, in the implementation of Render function of VolumeModel, the rendering is delegated to the Render function of VolumeRenderer.

VolumeRenderer is an abstract class and its concrete subclasses implement different volume rendering algorithms by overriding the virtual function Render defined in the VolumeRenderer. Currently MITK implemented the classical Ray Casting algorithm (VolumeRendererRayCasting), the fastest pure software implementation, i.e. Shear Warp algorithm (VolumeRendererShearWarp) [10], the modern GPU and texture mapping based algorithms (VolumeRendererTextureMapping2D, VolumeRendererTextureMapping3D) [8] [9]. The new algorithm can be added easily by subclassing VolumeRenderer and override the Render virtual function.

In addition to the shading related parameters, the main purpose of VolumeProperty is to provide a flexible framework to give VolumeRenderer the parameters of opacity transfer function. To get this aim, VolumeProperty has a class member of TransferFunctionGenerator, which is an abstract class and defines a set of interface to let its subclass to implement to generate different transfer functions. The output of TransferFunctionGenerator is an object of TransferFunction. TransferFunction1D, TransferFunction2D and TransferFunction3D are three concrete subclasses of TransferFunction and are used to support the multi-dimensional transfer function [11]. To support the multi-dimensional transfer function, VolumeRenderer must be aware of the dimensionality of transfer function and deal with them specially.

4.2. Implementation of cross platform

All the code in MITK are written by using the ANSI C++ syntax, and this can ensure the portability of MITK. But for some platform specific code, such as windows management, event processing, virtual memory management, we must write one set of code for every supported platform. In order to get an elegant solutions to encapsulate these platform specific code, MITK use one Design Pattern, named Bridge, to deal with this problem. In the following parts, we use View as an example to explain this solution.

In MITK, View is the only class which is dependent on the graphical user interface. It provides a screen window to display images or 3D graphics, which definitely causes View is dependent on specific operating system. The platform specific code should be separated and adding the support of a new operating system should not affect the client code. To get these purposes, the structure of View is designed as is shown in fig. 6.

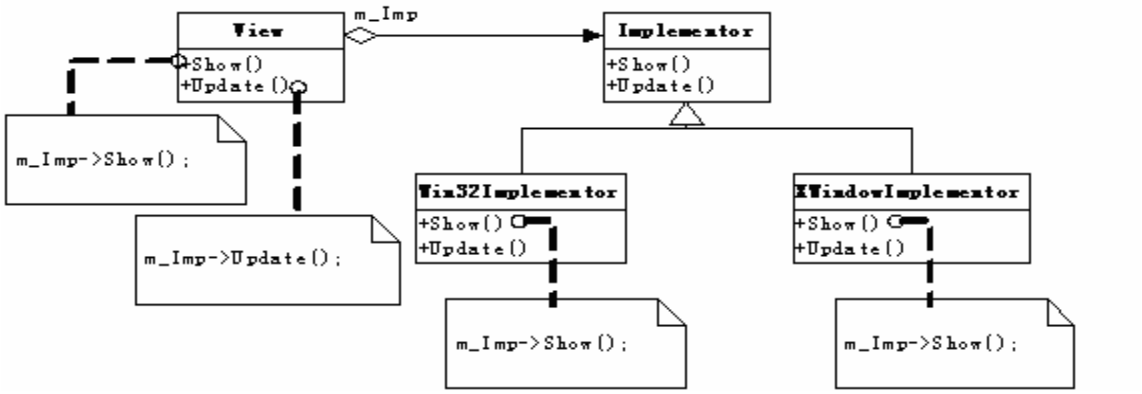

Figure 6. The structure of View.

View maintains a pointer of Implementor and delegates all the platform specific code to the Implementor. Implementor is an abstract class and its concrete subclasses, such as Win32Implementor, XWindowImplementor, override the virtual functions defined in Implementor to implement the platform specific parts by calling the functions provided by specific operating system. In this structure, clients only know the existence of View, and they don't know the existence of Implementor. When we add a support of new operating system, we only require to derive a new subclass from Implementor and the client code doesn't need to modify.

4.3. Implementation of SSE acceleration

Intel added the support of SSE extended instructions set in the Pentium III and later processors to improve the performance of 3D applications and multi media applications. To maximize the performance of MITK, we support SSE instructions set internally and implement some key algorithms using SSE acceleration. To support SSE instructions set, there are some problems to deal with. Firstly, SSE instructions set is used at the assembly language level traditionally. This violates the portability design goal of MITK. Secondly, not all the CPU support SSE instructions set. Thus we must detect the CPU-related information in the run time and dynamically decide whether the SSE-accelerated version should be used. For the first problem, many mainstream C++ compilers (Visual C++ 6.0 plus Processor Pack, Visual Studio .Net, gcc, Intel C++ Compiler) support the SSE intrinsics recently. In MITK, we use these high level intrinsics to coding SSE-accelerated algorithms and this can make the code porting easily. For the second problem, we provide two

sets of shared library. One is SSE-accelerated version, and the other is ordinary version. The client is responsible for the decision of which version is loaded in run time.

4.4. Memory management

In the design of MITK, we must implement an efficient memory management scheme as an infrastructure to maintain the stability and robustness of the toolkit. For the data object, both Volume and Mesh support the loading and the access of out-of-core dataset by using the operating system provided memory mapped file and operating the virtual memory directly. Another level of memory management is to ensure that memory leaking doesn't occur in run time. We used two Design Patterns, named smart pointers and reference counting, to ensure the memory of an MITK object is deleted when it doesn't needed by any other MITK object. In addition this, we also implement a simple garbage collection mechanism to ensure all the MITK objects are deconstructed at the end of one application.

5. APPLICATION EXAMPLES

Fig. 7-9 show some application examples using the MITK, including image segmentation, surface rendering, volume rendering and mesh smoothing.

6. CONCLUSIONS AND FUTURE WORK

This paper introduces MITK, an integrated medical image processing and analyzing toolkit. The design goals, overall framework and implementation of some key technologies are discussed in details. Currently we have finished an early version and it can be download freely at http://www.3dmed.net/mitk to test and evaluate. We have developed the new version of 3DMed [12] using MITK successfully. And this proves that MITK can be used to develop complex system software.

In the future work, we will maintain the stability of the framework of MITK and add various algorithms to enrich the algorithms collections. The out-of-core algorithms and more SSE-accelerated algorithms will get a higher priority to show the new features of MITK. When the MITK is enough stable and mature, we will distribute it as open source software as VTK and ITK.

ACKNOWLEDGMENTS

This paper is supported by the Project for National Science Fund for Distinguished Young Scholars of China under Grant No. 60225008, the Special Project of National Grand Fundamental Research 973 Program of China under Grant No. 2002CCA03900, the National High Technology Development Program of China under Grant No. 2002AA234051, the National Natural Science Foundation of China under Grant Nos. 30370418, 90209008, 60302016, 60172057, 30270403.

The authors would like to thank the developers of VIK and ITK to make these excellent toolkits available and open source. The design of MITK has got a lot of inspirations from them.

REFERENCES

1. Visualization Toolkit, http://www.vtk.org.
2. Insight Segmentation and Registration Toolkit, http://www.itk.org.
3. Will Schroeder, Ken Martin, Bill Lorensen Schroeder, *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics 3rd Edition*, Kitware, Inc. Publisher, 2003.
4. William J. Schroeder, Lisa S. Avila, William Hoffman, "Visualizing with VTK: A Tutorial", *IEEE Transaction on Computer Graphics and Applications*, **Vol. 20**, No. 5, pp. 20-27, 2000.
5. William J. Schroeder, Kenneth M. Martin, William E. Lorensen, "The Design and Implementation Of An Object-Oriented Toolkit For 3D Graphics And Visualization", *Proc. Of IEEE Visualization '96*.
6. Luis Ibanez, Will Schroeder, Lydia Ng, Josh Cates, *The ITK Software Guide: The Insight Segmentation and Registration Toolkit (version 1.4)*, Kitware, Inc. Publisher, 2003.

7. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software*, Pearson Education Publisher, 1994.
8. C. Resk-Salama, K. Engel, M. Bauer, G. Greiner, T. Ertl，"Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage-Rasterization"，*Proc. Eurographics/SIGGRAPH Workshop on Graphics Hardware 2000.*
9. K. Engel, M. Kraus, and T. Ertl, "High-quality pre-integrated volume rendering using hardware accelerated pixel shading", *Proc. Eurographics/SIGGRAPH Workshop on Graphics Hardware 2001*.
10. P. Lacroute and M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transformation," *Proc. SIGGRAPH '94*, pp. 451- 458, 1994.
11. Joe Kniss，Gordon Kindlmann，Charles Hansen，"Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets", *Proc. of IEEE Visualization 2001*.
12. 3DMed (3-Dimensional Medical Image Processing and Analyzing System), http://www.3dmed.net

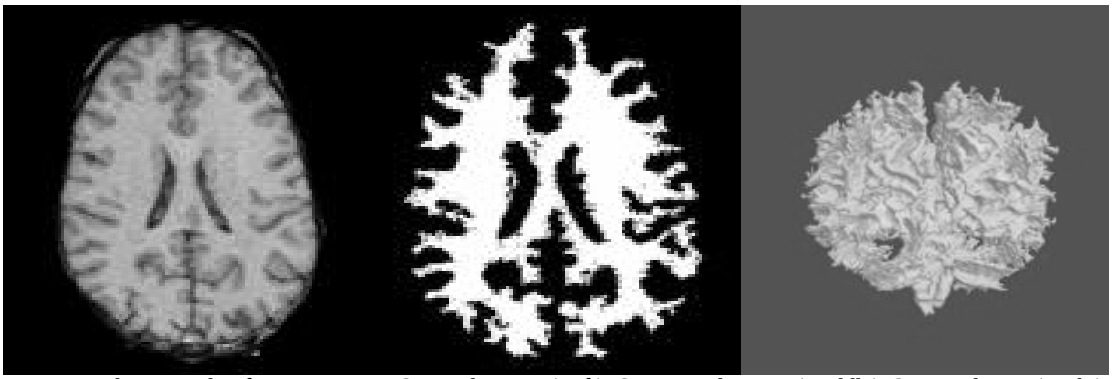

Figure 7. The example of segmentation. Original Image (Left), Segmented Image (Middle), 3D Rendering (Right)

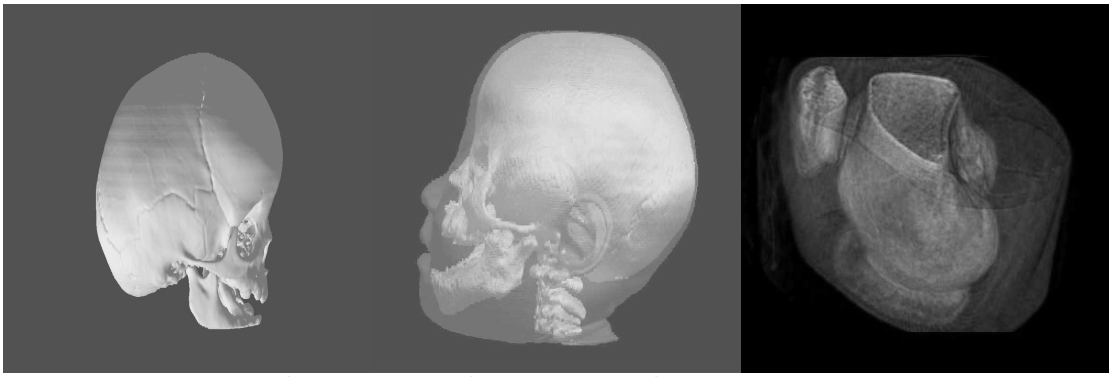

Figure 8. The example of visualization. Surface Rendering (Left and Middle), Volume Rendering (Right).

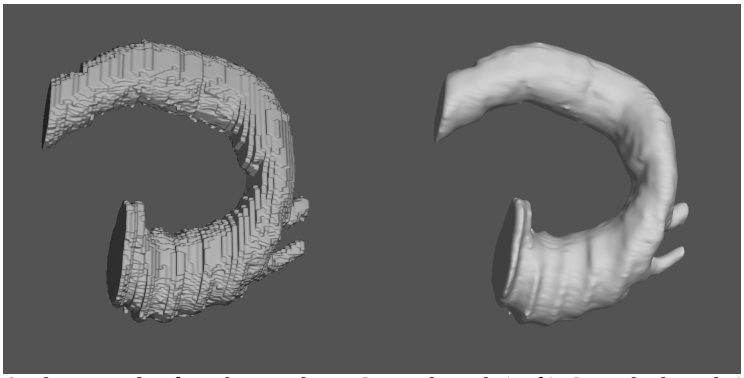

Figure 9. The example of mesh smoothing. Original Mesh (Left), Smoothed Mesh (Right).