MITK_Linux1.4使用教程

中科院自动化所医学影像研究室

网址:http://www.mitk.net/

联系人:田捷

E-mail:tian@ieee.org

目录

1 MITK_Linux1.4 的安装说明 3

1.1下载MITK_Linux1.4 3

1.2 Lib中动态链接库文件的处理 3

1.3 Include中头文件的处理 3

2 MITK_Linux的一些基本概念和功能..... 3

3使用 MITK_Linux1.4 进行三维重建 5

3.1 例子程序需要使用的几个类..... 5

3.2 三维重建实例..... 6

3.2.1 第一步 建立C++工程 7

3.2.2 第二步 布置GUI界面 8

3.2.3 第三步 设置信号和槽的连接..... 12

3.2.4 第四步 编写源代码..... 13

3.2.5 第五步 编译连接生成可执行文件 15

3.2.6 第六步 运行可执行文件 15

附录 1 mainform.ui.h源码 18

附录 2 setthresholddialog.ui.h源码 25

附录 3 main.cpp源码 26

1 MITK_Linux1.4 的安装 说明：

1.1下载MITK_Linux 1.4

请访问http://www.mitk.net/download下载MITK 1.4 alpha for Linux (zip package),解压缩到任意位置。

1.2 Lib中动态链接库文件的处理

方法 1 将Lib文件夹中的libmitk.so ,libmitk.so.0 ,libmitk.so.0.0 三个文件复制到路径“/usr/lib”下，即将动态库放入系统默认的引用路径。

方法 2 终端中运行 sudo gedit /etc/ld.so.conf ,打开系统动态链接库配置文件ld.so.conf，在ld.so.conf最后加入一行Lib文件夹实际所在路径，例如“/home/dongdi/Desktop/mitk_linux1.4/Lib”,保存；然后运行 /sbin/ldconfig 载入配置，Lib路径永久添加进了环境变量。

1.3 Include中头文件的处理

Include文件夹中的头文件原则上可以存放在任何路径下，建议存放在路径“/usr/include/mitk_linux1.4/include”下。

2 MITK_Linux 的一些基本概念和功能.

在讲述具体例子之前,首先理解 MITK 的一些基本概念和一些重要的类是非常必要的,下面将先介绍这些知识。

MITK_Linux 基础

MITK_Linux的设计采用的是数据流的模型,以数据处理为中心,在概念上算法与数据是分开表达的,下面的框图表示了MITK_Linux对数据的整个处理流程。

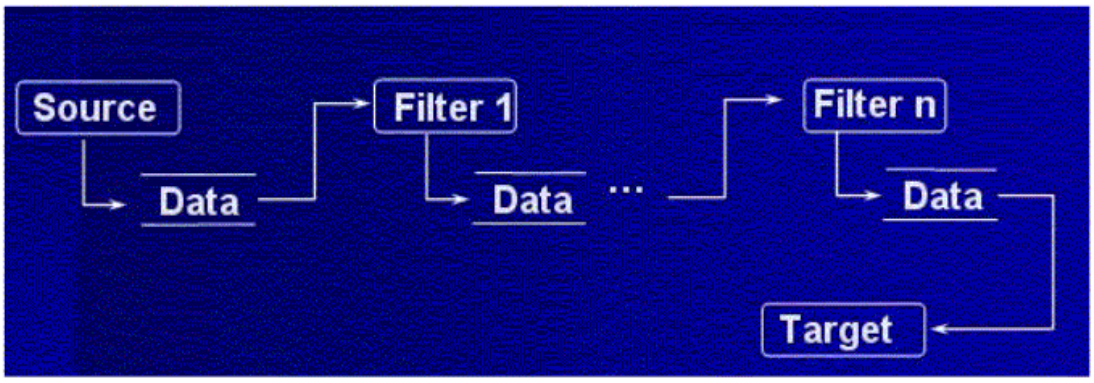

图 2.1 MITK_Linux的整体框图

其中 Source、Filter、Data、Target 等都是一些高层抽象的类,封装的是一些概念,具体工作通过继承由它们的子类来完成。

Source: 负责生成数据,具体的例子包括从磁盘上读文件进来(在MITK_Linux中叫做 Reader),或者用某些算法生成数据,比如随机数产生器(在MITK_Linux中叫做ProcedualSource)。

Filter: 对数据进行处理,也就是各种各样的算法。

Target:顾名思义,就是数据的终点。具体的例子包括:将得到的结果数据保存至磁盘(在 MITK_Linux中叫做 Writer),或者将得到的结果在屏幕上显示出来(在MITK_Linux中叫做View)。

Data: 上面的三种对象实际上都是封装的算法的行为,而 Data 就是对算法要处理的数据所进行的抽象。当然一个系统要能处理多种不同的数据,在MITK_Linux中是通过 Data 的各个具体的子类来描述不同的数据对象的。

清楚了这些高层的概念以后,剩下的就是如何将这些东西联系在一块,组成一个实用的系统。正如上面的图 1 所示的,系统通过这四种对象组成了一条流水线(Pipeline),数据起于 Source,终于 Target,中间经过多个 Filter 的处理。这种抽象关系足以描述我们大部分的以数据处理为中心的应用程序的需求,并且在概念上非常简单。

那么在具体写程序的时候,又如何联系起这个Pipeline呢?答案是通过SetInput和GetOutput。这里举个例子,比如有两个连续的Filter,一个叫Filter1,一个叫Filter2,可以参看图1,那么可以通过 Filter2->SetInput (Filter1->GetOutput())来将两个Filter联系起来.那么对于Filter2来说,现在拿到了自己的输入数据,就可以调用 Filter2->Run()来让 Filter2 干自己的活,干完以后,当然就可以通过Filter2->GetOutput()来将数据传给下一个Filter了。

通过这种所谓的 SetInput / GetOutput 方式,使用 MITK_Linux的客户端程序员的工作将非常轻松,可以通过短短的几行代码来完成本来非常复杂的工作,这些将在下面的例子程序中得到体现.

3使用 MITK_Linux1.4进行三维重建

本章将用一个例子来演示使用 MITK_LINUX进行三维断层图像的表面重建,例子使用 Ubuntu下的Qt3作为开发环境,为了清晰起见,这里使用了比较多的截图来演示.（注：目前Qt在Ubuntu下已经推出4.3.0版本，但是例子程序中使用qt-x11-free-3.3.7进行编写，用户可以根据Qt4的新特性对例子程序进行适当的修改以使其在Qt4上也能正常运行。）

3.1 例子程序需要使用的几个类

mitkVolume:首先,mitkVolume 是一个 Data,也就是说它是代表一个数据对象的。它是整个 MITK 中最重要的数据对象,代表一个三维断层图像数据,所有要处理的断层数据都必须先表达成 mitkVolume 的形式才能得到后续的处理。要使用它必须先包含 mitkVolume.h 头文件,关于它的接口函数的详细说明请参看 MITK 的帮助文件。

mitkImageModel: mitkImageModel 是一个 Data Model,表示 Data 对象在二维场景中的显示模型。mitkImageModel 即是对应于 mitkVolume 对象的显示模型。

mitkImageView: mitkImageView 是mitkView的子类,封装的是屏幕上的一个窗口,它具有能显示多个 mitkImageModel对象的能力,并且支持鼠标的交互式操作。要使用它必须先包含 mitkImageView.h 头文件,关于它的接口函数的详细说明请参看MITK 的帮助文件。

mitkMesh: mitkMesh 也是一个 Data,代表一个三维的几何数据对象,具体说就是三维重建以后物体表面的表达。要使用它必须先包含mitkMesh.h 头文件,关于它的接口函数的详细说明请参看 MITK 的帮助文件。

mitkSurfaceModel: mitkSurfaceModel 是一个 Data Model,表示 Data 对象在三维场景中的显示模型。mitkSurfaceModel 即是对应于 mitkMesh 对象的显示模型。

mitkMarchingCubes: mitkMarchingCubes 是一个 Filter,也就是说它是代表一个算法对象的。它封装了三维重建算法,是我们这里要使用的主要的算法。要使用它必须先包含 mitkMarchingCubes.h 头文件,关于它的接口函数的详细说明请参看 MITK 的帮助文件。

mitkView: mitkView 是一个 View,封装的是屏幕上的一个窗口,它具有能显示多个 Model 对象的能力,并且支持鼠标的交互式操作。要使用它必须先包含 mitkView.h 头文件,关于它的接口函数的详细说明请参看MITK 的帮助文件。

3.2 三维重建实例

首先，我们先看一下这个例子所完成的功能，图3.1是例子运行时的主界面。

左边是原始体数据浏览，您可以用鼠标进行操作：左键旋转、中键亮度、右键缩放；可以通过拖动下面的滑动器，来切换不同的切片数据的显示；点击工具栏上的按钮，进行切片数据的X_Y,Y_Z,Z_X方向的显示。

右边是进行表面重建后的结果，您可以用鼠标进行操作：左键旋转、中键平移、右键缩放。

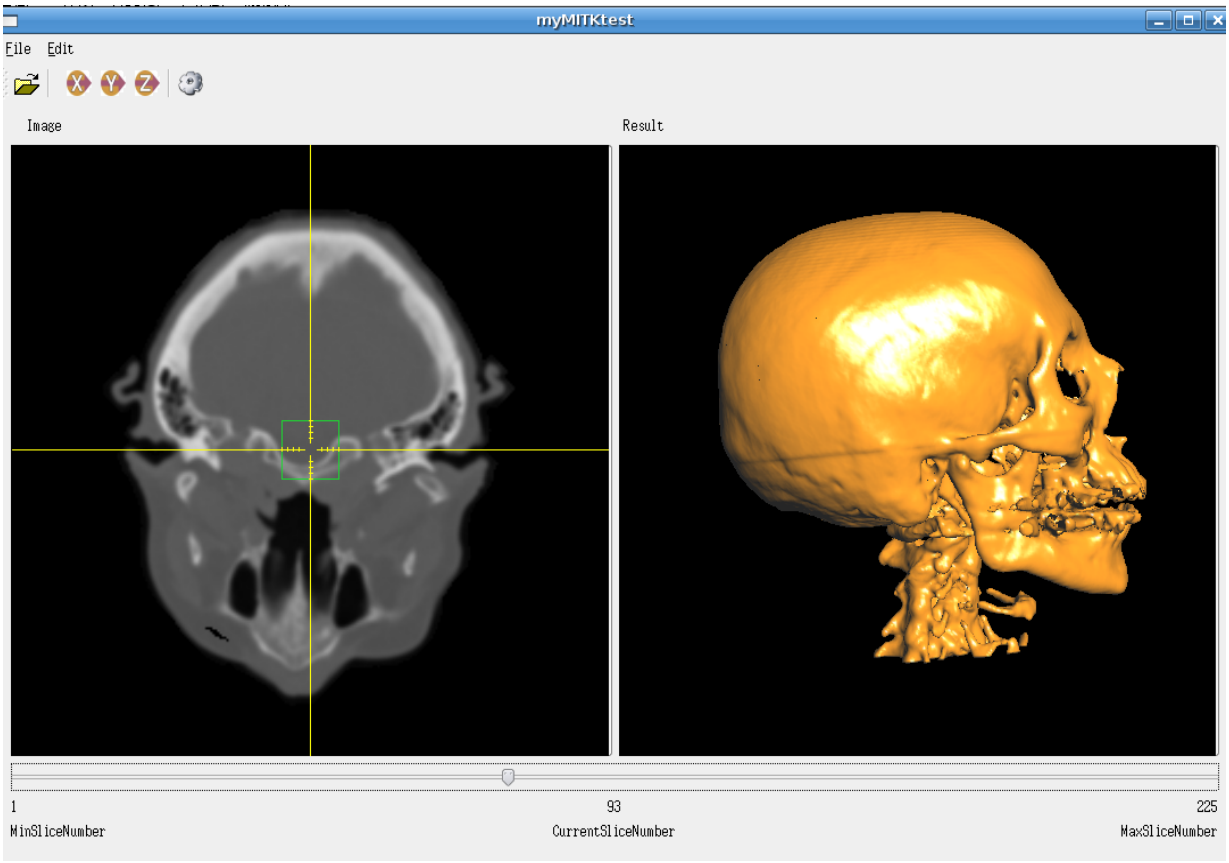

图 3.1

现在我们用上面介绍的Pipeline的概念先来描述下面这个例子程序，见图3.2。

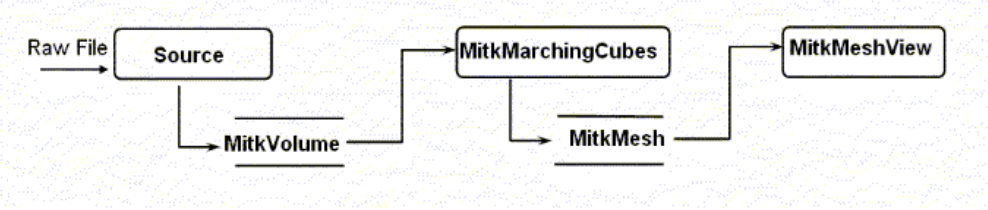

图 3.2

要注意的是，在这里为了演示一下mitkVolume的接口函数的使用（因为它是一个非常重要的数据类），在Source这一块并没有使用MITK_Linux中的mitkRawReader类，而是直接从文件里面读数据，然后填充至mitkVolume中。

经过了前面漫长的准备过程,终于到了该讲实际例子的时候,我相信这时你应该已经对 MITK_LINUX有了一个比较粗浅的认识了,如果你感觉前面非常抽象的话,那么现在就到了实践的时候了。

您可以按下面的步骤来亲自创建一个三维重建的应用程序，或者更简单的，直接打开“Example/SurfaceRender_QT/myMITKtest.pro”，修改工程设置中的 Libs 和 Includepath 两项后，转跳到“第五步 编译连接生成可执行文件”。

3.2.1第一步 建立C++工程

打开Qt Designer，新建C++ Project，设置如下：

Project File：工程路径/ myMITKtest.pro
Language : C++
Database File: 无
Template: app
Config : qt release opengl
Libs : Lib文件夹所在路径/libmitk.so
Defines: 无
Includepath: Include文件夹所在路径

如图 3.3，图 3.4。

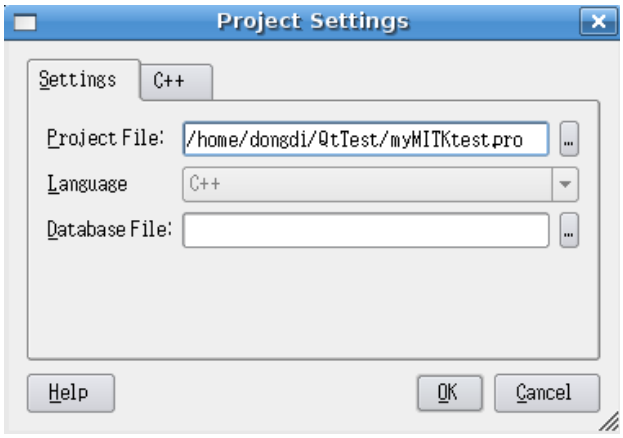

图 3.3

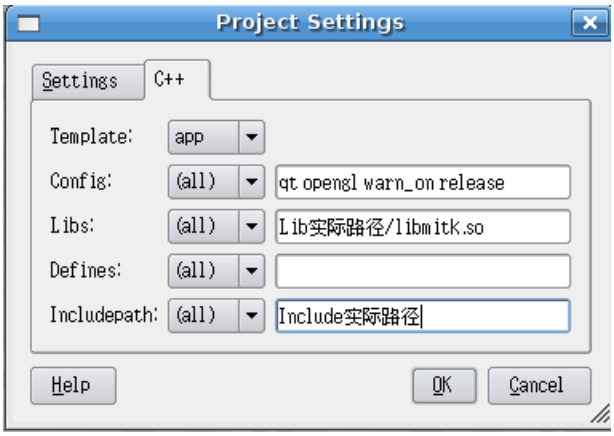

图 3.4

3.2.2第二步 布置GUI界面

1 建立主界面mainform.ui

（1）新建Main Window，设置选项如图3.5，图 3.6 所示。

name：MainForm

Caption：myMITKtest；

保存为mainform.ui

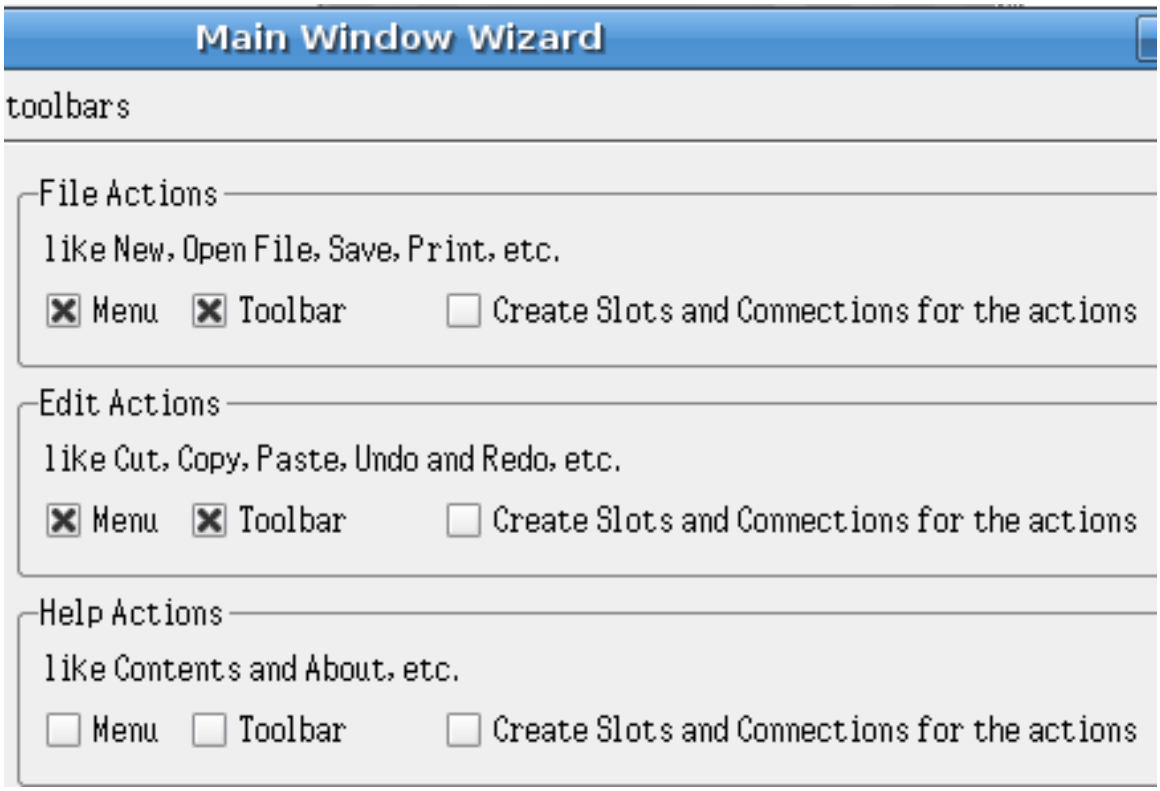

图 3.5

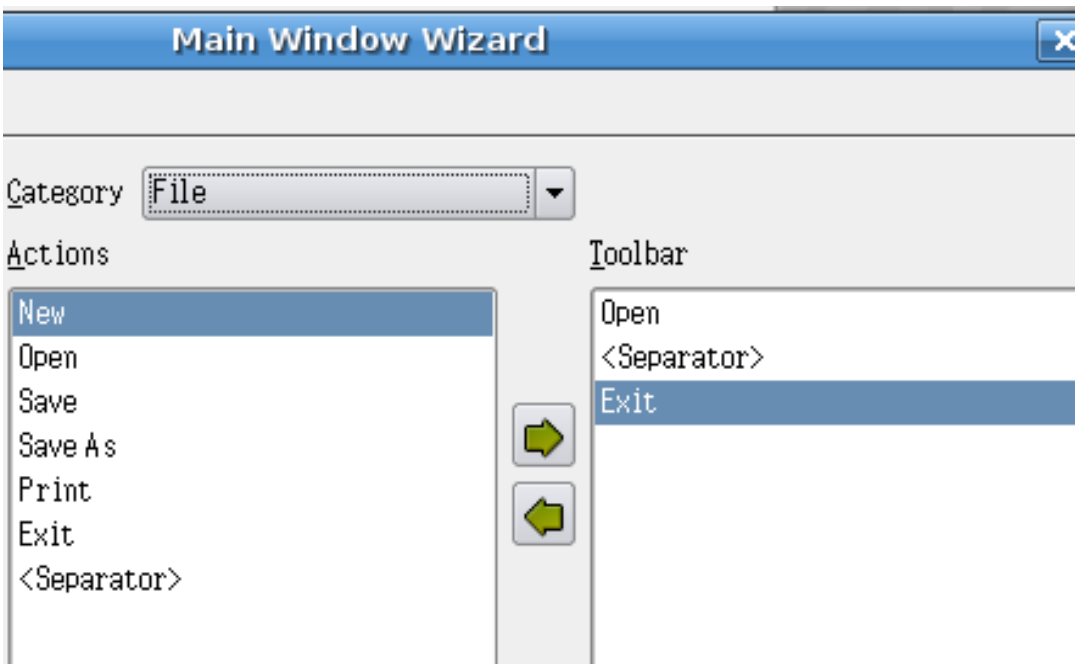

图 3.6

（2）Action Editor中只保留fileOpenAction和fileExitAction,其他Action全删掉，再新建四个Action：

name：editXYAction，editYZActioni，editZXAction，editRenderAction；

iconSet：X.jpg，Y.jpg，Z.jpg ，Render.png ；

text：X_Y_View，Y_Z_View，Z_X_View，Render ；

然后再将这四个Action拖入工具栏和菜单栏的Edit中，如图 3.7

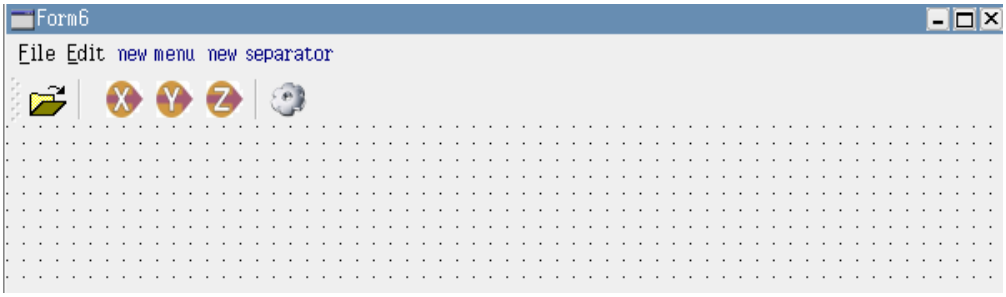

图 3.7

（3）在工具栏下添加两textlabel控件：

text：image ， result

加入两spacer控件用布局管理器使对齐，如图 3.8 示

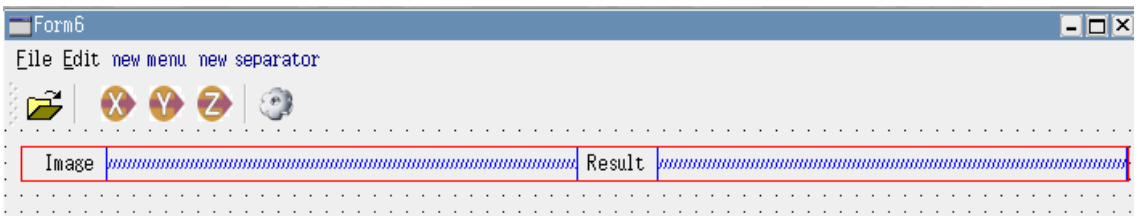

图 3.8

（4）窗口中央添加两textlabel控件：

name：ImageLb， SurfaceRenderingLb ；

text：No Image， No Result ；

功能：这两个textlabel用来做ImageView和SurfaceView显示的容器。

如图 3.9 所示。

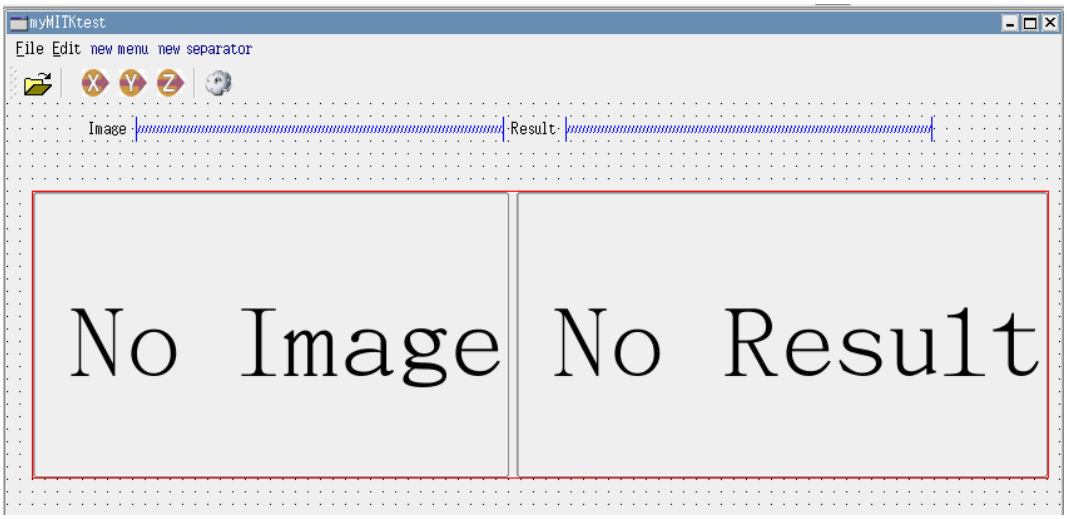

图 3.9

（5）添加slider控件：

name：sliceNumSlb ；

功能：通过对其拖动来切换不同切片数据的显示 ；

如图 3.10 所示。

图 3.10

（6）下面添加三对textlabel控件：

name：MinValLb，MinValNameLb

CurValLb，CurValNameLb

MaxValLb，MaxValNameLb ；
text： 1 ， MinSliceNumber
50 ， CurrentSliceNumber
100 ， MaxSliceNumber ；
功能：MinValLb 切片第一张 ；
CurValLb 当前显示的切片张数 ；
MaxValLb 切片总数 ；
添加两个spacer控件，使用布局管理器对齐，如图 3.11 所示。

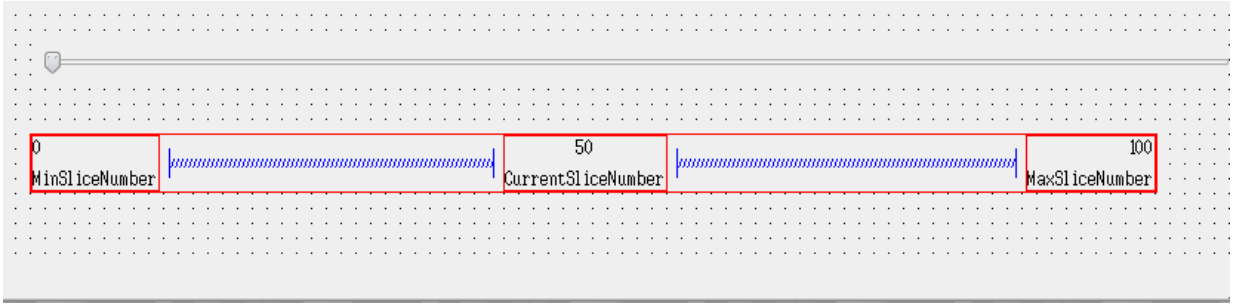

图 3.11

（7）将所有控件用布局管理器对齐，如图 3.12 示。

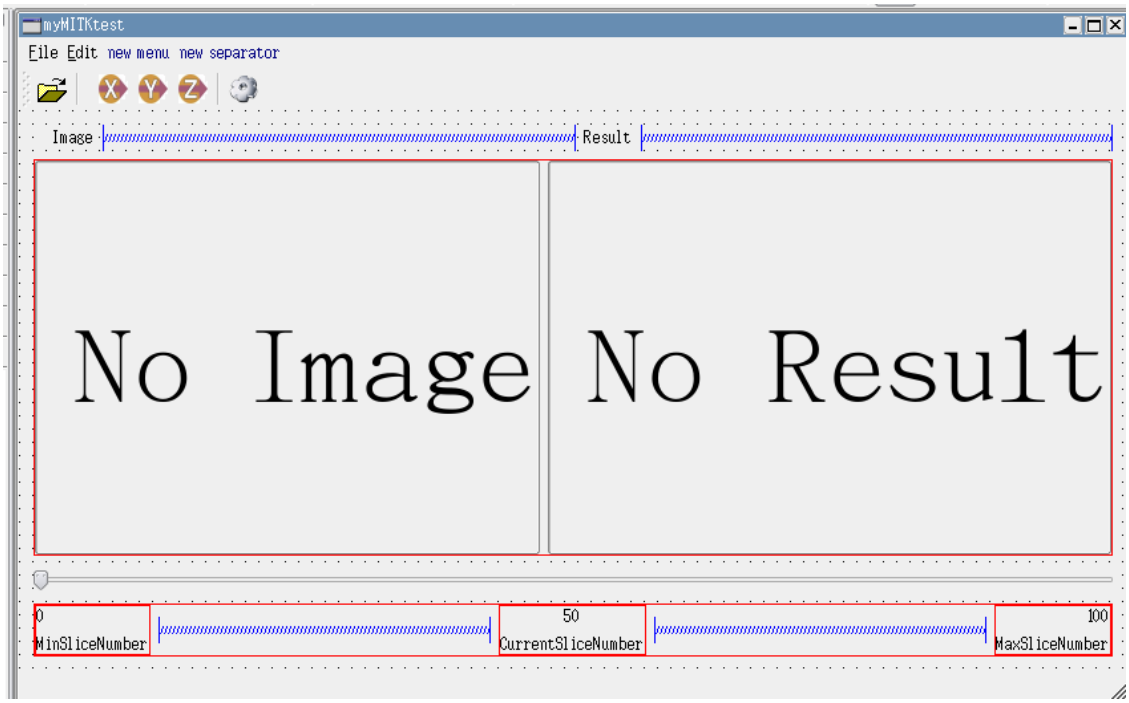

图 3.12

（8）保存。

2 建立阈值设置弹出窗口setthresholddialog.ui

（1）新建Dialog：

name: SetThresholdDialog ;

caption: SetThreshold ;

保存为setthresholddialog.ui 。

（2）添加两个TextLabel控件：

text：HighThreshold，LowThreshold ；

添加两个LineEdit控件

name：highThresholdLineEdit，lowThresholdLineEdit ；

text：150，100 ；

功能：用来输入阈值分割的高低阈值 ；

如图 3.13 所示。

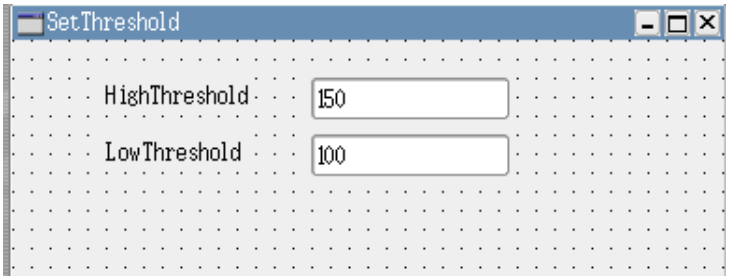

图 3.13

（3）添加两个PushButton控件：

name： okPushButton ， cancelPushButton ；

text： OK ， Cancel ；

再加入一个spacer控件然后利用布局管理器对齐，如图 3.14 所示。

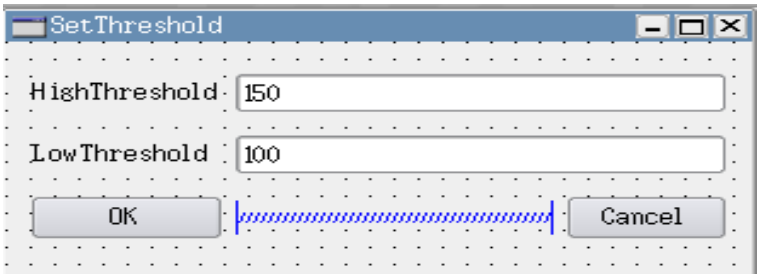

图 3.14

（4）保存。

3.2.3 第三步 设置信号和槽的连接

（1）mainform.ui

1）新建五个的void 类型的SLOT：on_Open(), on_Render(), on_XY(), on_YZ(), on_ZX(), sliceValChanged(int) .

2）设置信号和槽的连接如图3.15所示.

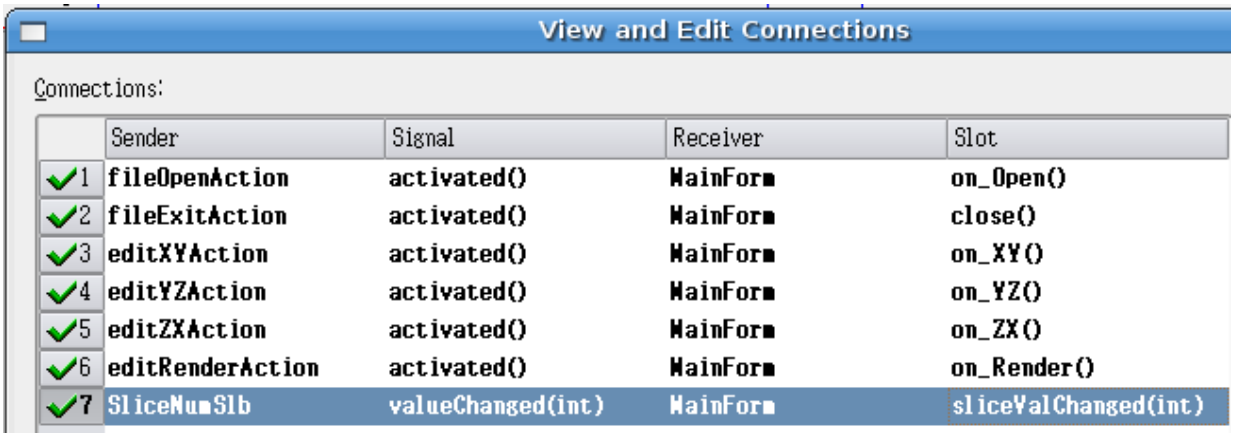

图3.15

（2）setthresholddialog.ui

1）新建两个的void 类型的SLOT：on_OK()，on_Cancel() .

2）设置连接如图3.16 所示。

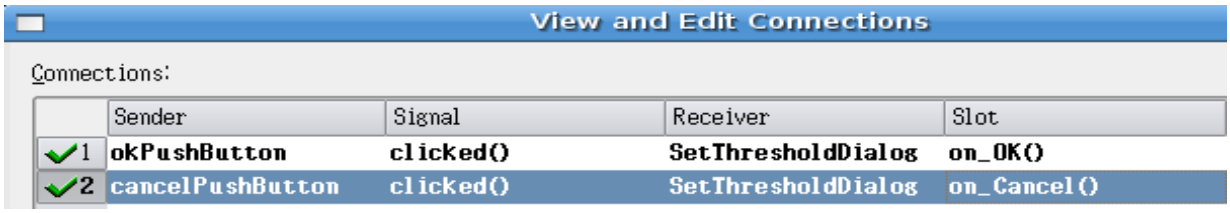

图3.16

3.2.4 第四步 编写源代码

（1）编写 mainform.ui.h:

1）双击mainform.ui创建mainform.ui.h 。

2）在Object Explore添加如下项：

Class Variables：

protected :

```
mitkVolume *m_Volume;
mitkMesh * m_Mesh;
mitkImageModel *m_ImageModel;
mitkImageView *m_ImageView;
mitkSurfaceModel * m_SurfaceModel;
mitkView * m_SurfaceView;
Display * m_dpy;
int m_CurSlice;
mitkSurfaceRendererUseVA * m_Renderer;
```

Forward Declarations:

```
class mitkIMOReader
class mitkImageView
class mitkVolume
class mitkImageModel
```

```
        class mitkMesh
        class mitkView
        class mitkSurfaceModel

    Include(in Declaration):
        "mitkVolume.h"
        "mitkMesh.h"
        "mitkImageModel.h"
        "mitkImageView.h"
        "mitkView.h"
        "mitkIMOReader.h"
        "mitkSurfaceRendererStandard.h"
        "mitkSurfaceModel.h"
        "mitkMarchingCubes.h"
```

结果如图 3.17 所示：

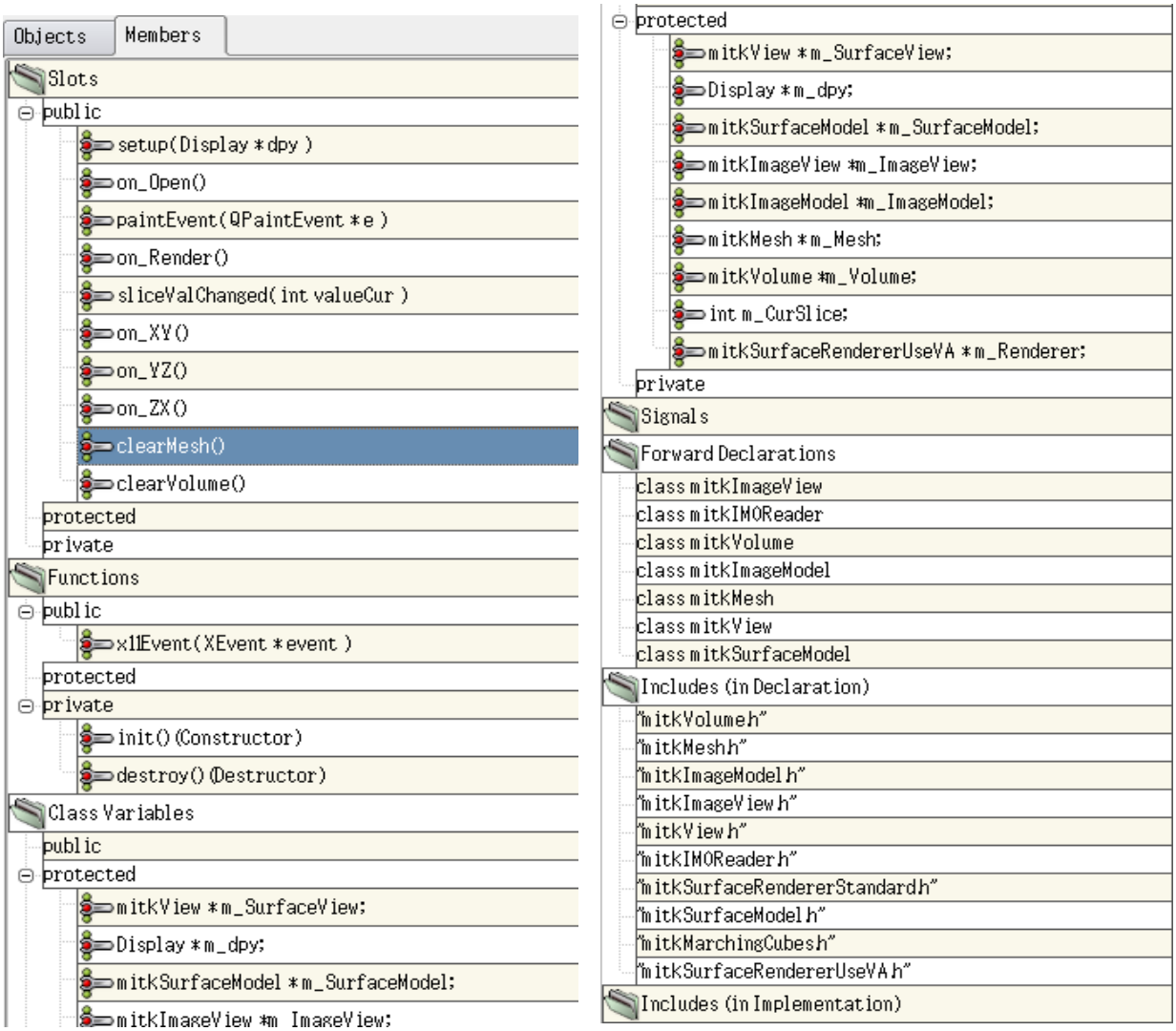

图 3.17

3）将附录 1 mainform.ui.h源码直接覆盖mainform.ui.h中代码。
4）保存。

（2）编写 setthresholddialog.ui.h:

1）双击setthresholddialog.ui生成setthresholddialog.ui.h。
2）将附录 2 setthresholddialog.ui.h源码直接覆盖setthresholddialog.ui.h中代码。
3）保存。

（3）编辑main.cpp：

1）新建 C++ Main-File(main.cpp)。
2）将附录 3 main.cpp源码直接覆盖 main.cpp中代码。
3）保存。

3.2.5第五步 编译连接生成可执行文件

1）打开终端，输入sudo -i,使用超级用户权限。
2）用cd命令进入到工程所在录。
3）使用命令 qmake -o makefile ./myMITKtest.pro生成建程文件makefile。
4）使用命令 make 编译连接生成可执行文件myMITKtest。

3.2.6 第六步 运行可执行文件

（1）在上步的基础上直接在终端输入“./myMITKtest”或在窗口界面下进入myMITKtest所在文件夹，双击myMITKtest以运行，界面面如图 3.18 所示。

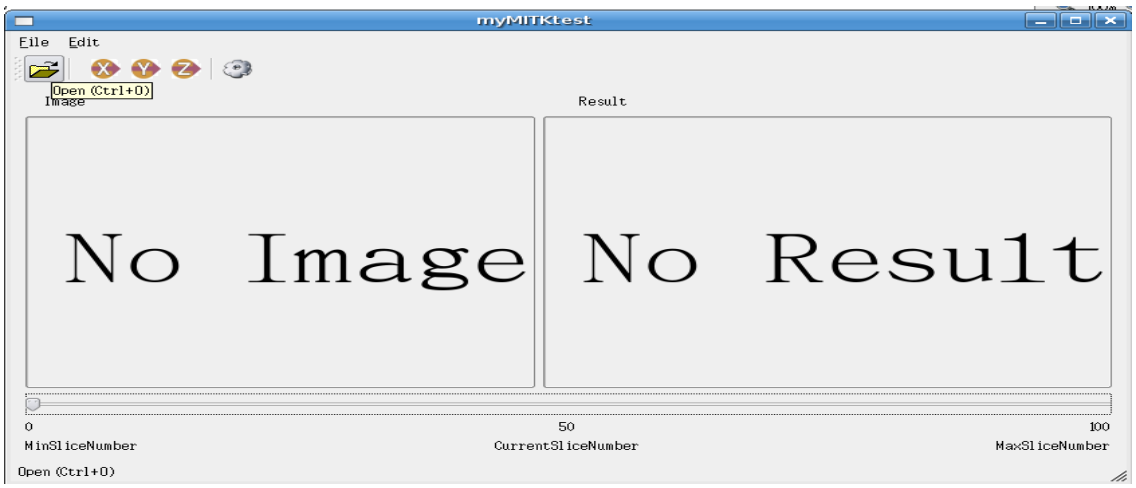

图 3.18

（2）点击打开按钮，弹出 Open 对话框，选择所要打开的 im0 文件，如图 3.19 所示。

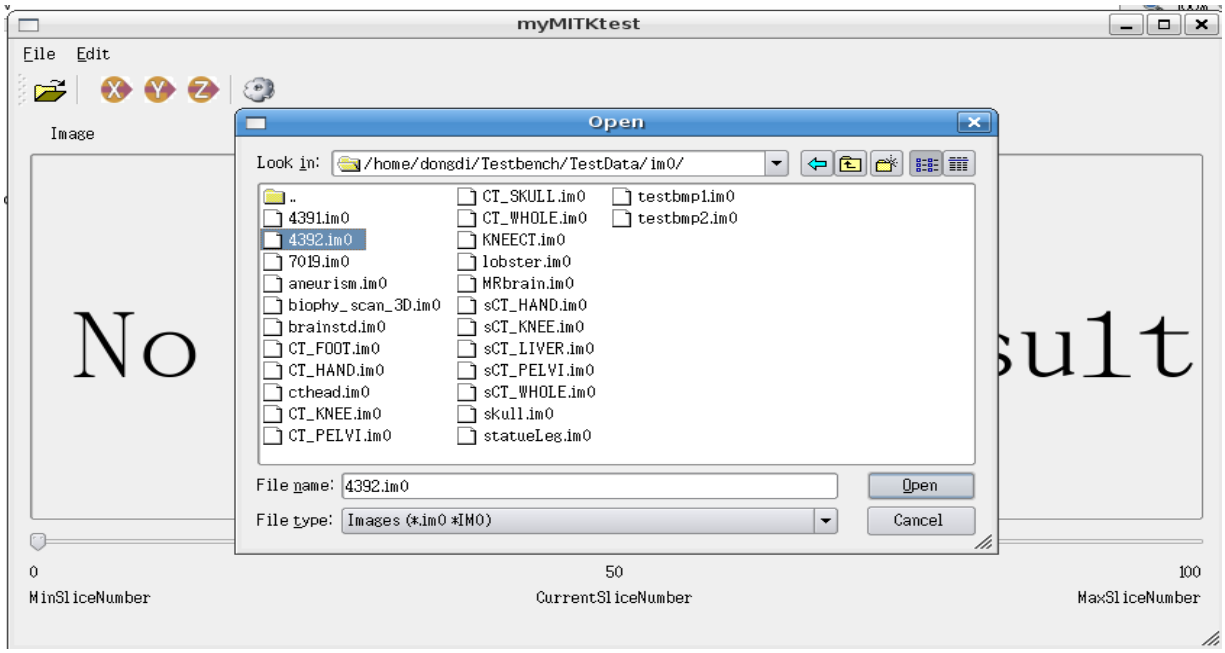

图 3.19

（3）打开文件后界面如图 3.20 示。

图 3.20

（4）点击 Render 按钮，弹出 SetThreshhold 对话框，填入阈值分割的高低阈值如图 3.21,输入后点击 OK 按钮。

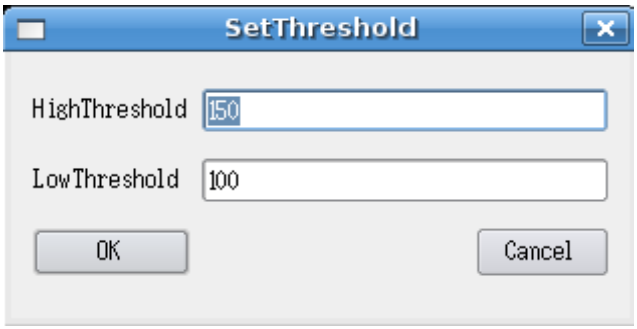

图 3.21

(5) 重建结果如图 3.22 示.

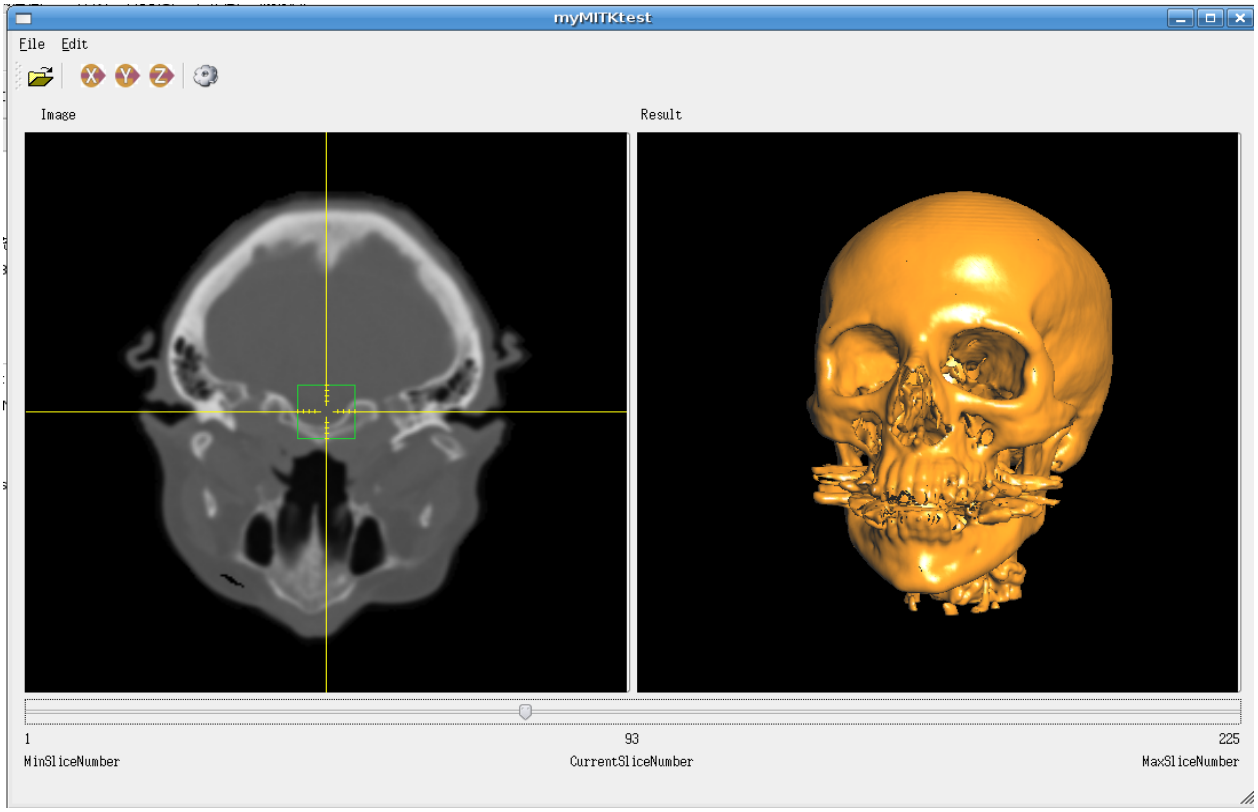

图 3.22

附录 1 mainform.ui.h 源码

```
/*=========================================================================
   Program:    SurfaceRender_QT for MITK_Linux1.4
   Date:          $Date:2008/05/07 14:02 $
   Version:     $Version: 1.0 $
   Copyright: MIPG, Institute of Automation, Chinese Academy of Sciences
   Last Modified by: Dong Di
=========================================================================*/
#include <qgl.h>
#include <stdio.h>
#include <qstring.h>
#include <qstringlist.h>
#include"qfiledialog.h"
#include "qlineedit.h"
#include "X11/Xlib.h"
#include "X11/Xutil.h"
#include "setthresholddialog.h"
//---------------------------------------------------------

//重载了QMainWidow类的x11Event(XEvent * event)函数，用来处理发生在Main Window中的
XEvent
bool MainForm::x11Event (XEvent * event)
{

   if (m_ImageModel&&event->xany.window == *(Window*)(m_ImageView->GetWindowId()))
      {
      bool res;
      res =m_ImageView->HandleXEvent(event);
      return TRUE;
      }
      else if (m_SurfaceModel&&event->xany.window ==
*(Window*)(m_SurfaceView->GetWindowId()))
      {
      bool res;
      res =m_SurfaceView->HandleXEvent(event);
      return TRUE;
      }
      return FALSE;
}
//---------------------------------------------------------
```

```
//建立Main Window时的初始化函数，对各个变量进行初始化
void MainForm::init()
{
	m_dpy=NULL;
	m_ImageView =NULL;
	m_ImageModel=NULL;
	m_CurSlice=1;
	m_SurfaceView = NULL;
	m_SurfaceModel = NULL;
	m_Volume=NULL;
	m_Renderer = NULL;

}
//-------------------------------------------------------------

//设置m_ImageView,m_SurfaceView的属性,DisplyId,ParentId,位置，大小，背景色
void MainForm::setup(Display *dpy)
{
	m_dpy=dpy;
	Window win = ImageLb->winId();
	m_ImageView=new mitkImageView;
	m_ImageView->SetDisplayId(dpy);
	m_ImageView->SetParent(&win);
	m_ImageView->SetPosition(0,0);
	m_ImageView->SetSize(ImageLb->width()-3,ImageLb->height());
	m_ImageView->SetBackColor(0,0,0);

    win = SurfaceRenderingLb->winId();
	m_SurfaceView= new mitkView;
	m_SurfaceView->SetDisplayId(dpy);
	m_SurfaceView->SetParent(&win);
	m_SurfaceView->SetPosition(0,0);
	m_SurfaceView->SetSize(SurfaceRenderingLb->width()-3, SurfaceRenderingLb->height());
	m_SurfaceView->SetBackColor(0, 0, 0);

}
//-------------------------------------------------------------

//打开im0数据，将其转化为mitkVolume并存入m_Volume中
void MainForm::on_Open()
{

	QString file = QFileDialog::getOpenFileName(    "/home","Images (*.im0 *IM0)",    this);
```

```
    if (file==NULL)
    {
    cout<<"Filename is NULL"<<endl;
    return;
    }
    mitkVolumeReader * m_Reader= new mitkIM0Reader;
    m_Reader->AddFileName(file);
    if (m_Reader->Run())
    {
    if(m_Volume)
    {
        m_Volume->RemoveReference();
        m_Volume=NULL;
    }
    m_Volume= m_Reader->GetOutput();
    m_Volume->AddReference();
    }
    if    (m_ImageModel==NULL)
    {
    m_ImageModel = new mitkImageModel;
    m_ImageModel->SetData(m_Volume);
    m_ImageView->AddModel(m_ImageModel);
    m_ImageView->Show();
    }
    else
    {
    m_ImageModel->SetData(m_Volume);
    }
    this->repaint(this);
    m_ImageView->Update();
    int sliceNum = m_Volume->GetImageNum();
    sliceNumSlb->setMaxValue(sliceNum);
    sliceNumSlb->setMinValue(1);
    MinValLb->setNum(1);
    MaxValLb->setNum(sliceNum);
    m_CurSlice=1;
    sliceNumSlb->setValue(m_CurSlice);
}
//------------------------------------------------------------

//重载重绘事件处理函数，设置m_ImageView,m_SurfaceView的大小
void MainForm::paintEvent ( QPaintEvent * e)
{
    m_ImageView->SetSize(ImageLb->width()-3,ImageLb->height());
```

```
    m_SurfaceView->SetSize(SurfaceRenderingLb->width()-3,SurfaceRenderingLb->height());
}
//-------------------------------------------------------------

//对m_Volume进行阈值分割，并对分割结果进行三维重建
void MainForm::on_Render()
{
    if (!m_Volume) return;
    int high(0),low(0);
    SetThresholdDialog * m_Threshold=new SetThresholdDialog
(this,"mySetThreholdDialog",TRUE);
    if (m_Threshold->exec())
    {
    high=int(m_Threshold->highThresholdLineEdit->text().toInt());
    low=int(m_Threshold->lowThresholdLineEdit->text().toInt());
    }
    else return;

    if( m_SurfaceModel)
    {
    this->clearMesh();
    mitkMarchingCubes *mc = new mitkMarchingCubes;
    mc->SetInput(m_Volume);
    if (mc->Run())
    {
        m_Mesh = mc->GetOutput();
        m_Mesh->AddReference();
    }
    m_SurfaceModel->SetData(m_Mesh);
    m_SurfaceView->Update();
    }
    else
    {
   //建立表面数据对象
    m_Mesh=NULL;
    //建立MarchingCube算法对象
    mitkMarchingCubes *mc = new mitkMarchingCubes;
    // 为 Marching Cubes 算法设置阈值
    mc->SetThreshold(low, high);
    // 设置输入数据
    mc->SetInput(m_Volume);
    // 从 mitkMarchingCubes 算法得到输出结果
    if (mc->Run())
    {
```

```
        m_Mesh = mc->GetOutput();
        m_Mesh->AddReference(); //重要
        }
        //建立表面数据模型
        m_SurfaceModel=NULL;
        m_SurfaceModel = new mitkSurfaceModel;
        // 设置表面材质属性
        m_SurfaceModel->GetProperty()->SetAmbientColor(0.75f, 0.75f, 0.75f,1.0f);
        m_SurfaceModel->GetProperty()->SetDiffuseColor(1.0f, 0.57f, 0.04f,1.0f);
        m_SurfaceModel->GetProperty()->SetSpecularColor(1.0f, 1.0f, 1.0f,1.0f);
        m_SurfaceModel->GetProperty()->SetSpecularPower(100.0f);
        m_SurfaceModel->GetProperty()->SetEmissionColor(0.0f, 0.0f, 0.0f,0.0f);
        //为表面数据，模型设置显示算法对象
        m_Renderer=new mitkSurfaceRendererUseVA;
        m_SurfaceModel->SetRenderer(m_Renderer);
        m_SurfaceModel->SetData(m_Mesh);
        m_SurfaceView->AddModel(m_SurfaceModel);
        //显示图片
        m_SurfaceView->Show();
}
}

//对SliceNumSlb被拖动的处理
void MainForm::sliceValChanged( int valueCur )
{
        if (m_ImageModel)
  //如果m_ImageModel已经创建，才会响应
        {
        CurValLb->setNum(valueCur);
        m_ImageModel->SetCurrentSliceNumber(valueCur - 1);
        m_ImageView->Update();
        m_CurSlice = valueCur;
        this->repaint(this);
        }
}
//-------------------------------------------------------------

//m_Volume进行XY方向显示
void MainForm::on_XY()
{
        //如果m_ImageModel已经创建，即已经输入过im0文件才会相应
        if(m_ImageModel)
        {
        m_ImageModel->SetViewMode(mitkImageModel::VIEW_XY);
```

```
        int sliceNum = m_ImageModel->GetTotalSliceNumber();
        sliceNumSlb->setMaxValue(sliceNum);
        sliceNumSlb->setMinValue(1);
        MinValLb->setNum(1);
        MaxValLb->setNum(sliceNum);
        m_CurSlice=1;
        sliceNumSlb->setValue(m_CurSlice);
        m_ImageView->Update();

        }
}
//-------------------------------------------------------------

//m_Volume进行YZ方向显示
void MainForm::on_YZ()
{
        if(m_ImageModel)
        {
        m_ImageModel->SetViewMode(mitkImageModel::VIEW_YZ);
        int sliceNum = m_ImageModel->GetTotalSliceNumber();
        sliceNumSlb->setMaxValue(sliceNum);
        sliceNumSlb->setMinValue(1);
        MinValLb->setNum(1);
        MaxValLb->setNum(sliceNum);
        m_CurSlice=1;
        sliceNumSlb->setValue(m_CurSlice);
        m_ImageView->Update();
              }
}
//-------------------------------------------------------------

//m_Volume进行ZX方向显示
void MainForm::on_ZX()
{
        if(m_ImageModel)
        {
        m_ImageModel->SetViewMode(mitkImageModel::VIEW_ZX);
        int sliceNum = m_ImageModel->GetTotalSliceNumber();
        sliceNumSlb->setMaxValue(sliceNum);
        sliceNumSlb->setMinValue(1);
        MinValLb->setNum(1);
        MaxValLb->setNum(sliceNum);
        m_CurSlice=1;
        sliceNumSlb->setValue(m_CurSlice);
```

```
        m_ImageView->Update();
        }
}
//-------------------------------------------------------------

//释放m_Mesh
void MainForm::clearMesh()
{
  if(m_Mesh)
        {
        m_Mesh-> RemoveReference();
        m_Mesh = NULL;
        }
}
//-------------------------------------------------------------

//释放m_Volume
void MainForm::clearVolume()
{
        if(m_Volume)
        {
        m_Volume-> RemoveReference();
        m_Volume = NULL;
        }
}
//-------------------------------------------------------------

//Main Window被关闭时，释放所占的资源
void MainForm::destroy()
{
        this->clearVolume();
        this->clearMesh();
        if (m_ImageView)
        {
        m_ImageView->Delete();
        m_ImageView = NULL;
        }
           if (m_SurfaceView)
        {
        m_SurfaceView->Delete();
        m_SurfaceView = NULL;
        }
}
```

附录 2 setthresholddialog.ui.h源码

```
/*==========================================================================

   Program:    SurfaceRender_QT for MITK_Linux1.4
   Date:          $Date:2008/05/07 14:02 $
   Version:     $Version: 1.0 $
   Copyright: MIPG, Institute of Automation, Chinese Academy of Sciences
   Last Modified by: Dong Di

==========================================================================*/
#include <iostream.h>

void SetThresholdDialog::on_Ok()
{
  QString    highThresholdValue=highThresholdLineEdit->text();
  QString lowThreshholdValue=lowThresholdLineEdit->text();
  if ( highThresholdValue.isEmpty()){}
  else if ( highThresholdValue.isEmpty()){}
  else accept();
}

void SetThresholdDialog::on_Cancel()
{
reject();
}
```

附录 3 main.cpp 源码

```
/*=========================================================================

   Program:    SurfaceRender_QT for MITK_Linux1.4
   Date:          $Date:2008/05/07 14:02 $
   Version:     $Version: 1.0 $
   Copyright: MIPG, Institute of Automation, Chinese Academy of Sciences
   Last Modified by: Dong Di

=========================================================================*/
#include <iostream.h>
#include "mitkImageView.h"
#include "mitkView.h"
#include <qgl.h>
#include <qapplication.h>
#include "mainform.h"
#include "X11/Xlib.h"
#include "X11/Xutil.h"

//继承 QApplication 类，其中加入 Mian Form 对象的指针 m_MainForm,并重载 x11EventFilter(XEvent
* event)，
//向其中的窗口分发 XEvent;
class mitkApplication : public QApplication
{
public:
    MainForm * m_MainForm;
    //构造函数,生成 Mian Form 对象，并设置其 Display，将对象设置为 MainWidget;
     mitkApplication(Display* dpy): QApplication(dpy)
     {
     m_MainForm= new MainForm;
     m_MainForm->setup(dpy);
     setMainWidget(m_MainForm);
     }
    bool x11EventFilter ( XEvent *event);
    virtual ~mitkApplication()
    {
        delete m_MainForm;
    }
};
//--------------------------------------------------------------------

//将截获的 XEvent 向下发送给 m_MainForm,m_MainForm 处理就返回 true;否则返回 flase;
```

```
bool mitkApplication::x11EventFilter (XEvent *event)
{
return    m_MainForm->x11Event(event);
}
//---------------------------------------------------------------------

//主函数;
int main( int argc, char ** argv )
{
        Display * dpy ;
   //   建立一个 display  至 X Server;
        dpy= XOpenDisplay(0);
 //生成 mitkApplication 对象，并将生成的连接传给构造函数;
        mitkApplication a(dpy);
//显示 Main Form 对象;
        a.m_MainForm->show();
 //进入 mitkApplication 事件处理循环;
        return a.exec();
}
```