MITK_LINUX 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@doctor.com

使用 MITK 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 Anjuta_1.2.4a 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.

Anjuta 安装时所注意的事项:

在新立得软件管理器中搜索 anjuta,并强制其版本为 1.2.4a（最新版本为 2.3.1，但其对 mitk 的支持不太好），安装 anjuta1.2.4a 后，还要安装一些相关的库(如 OpenGL 库等),打开新立得软件包管理器，在其中搜索 automake,autoconf，autogen,build-essential,libgl1-mesa-dev,libglu1-mesa-dev,libglu1t3-dev,freeglut3 并安装,（或者在终端中直接用命令"sudo apt-get install " 来安装）完这些,基本已配置好了相关的编译环境，应该就可以顺利运行 anjuta 了。

MITK 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 基础

MITK 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 对数据的整个处理流程。

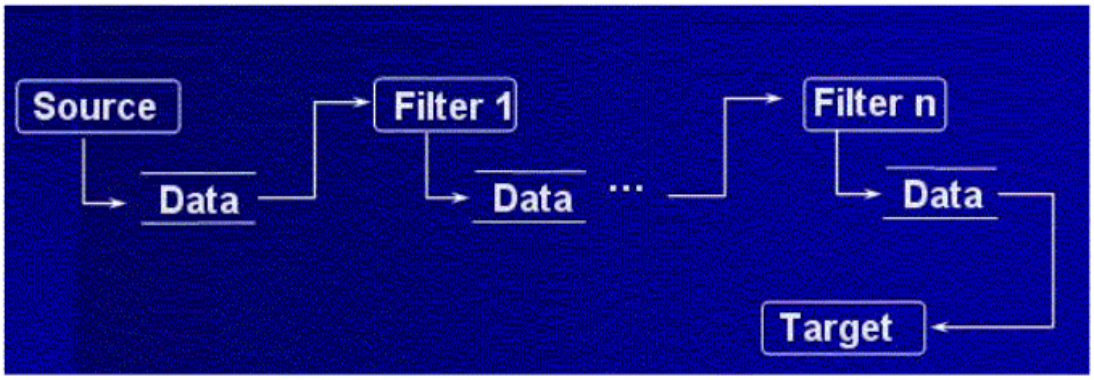


图 1 MITK 的整体框图

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 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 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 中叫做 Writer),或者将得到的结果在屏幕上显示出来(在 MITK 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 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 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。

三维重建需要使用的几个类

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。

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。

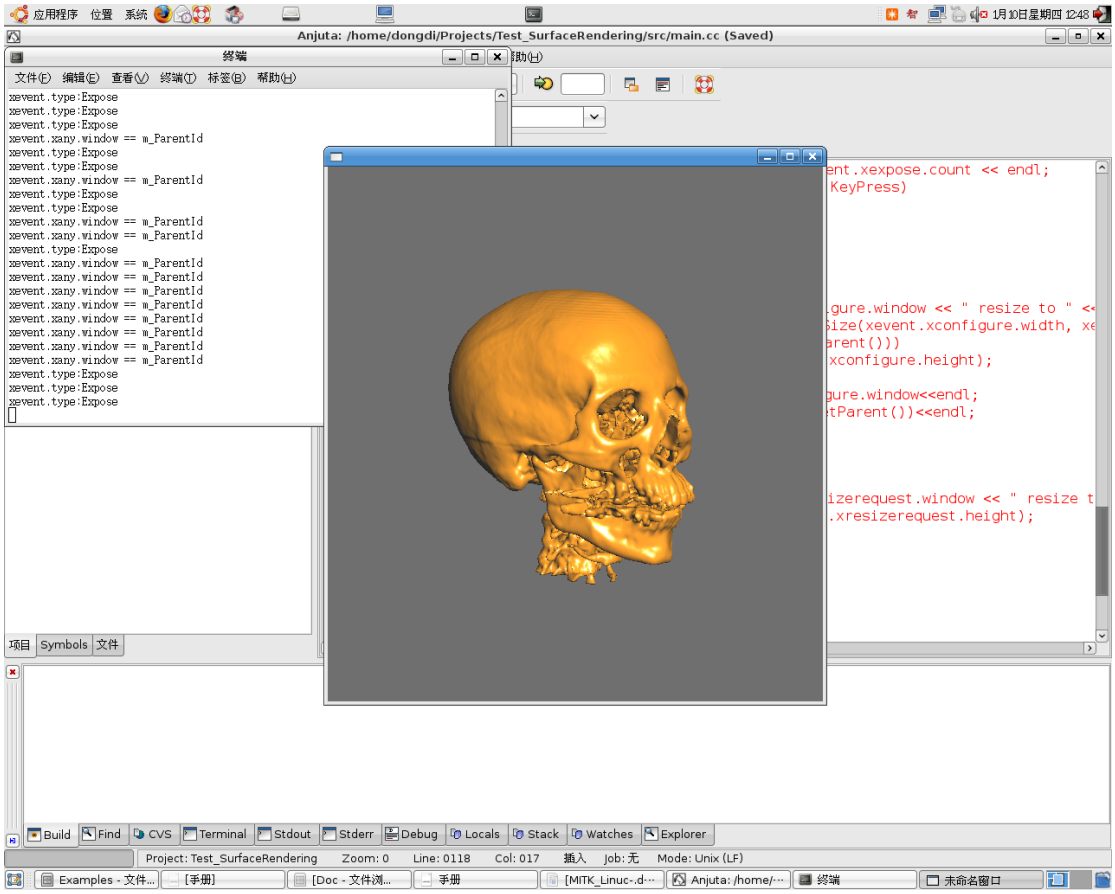
mitkSurfaceModel: mitkSurfaceModel 是一个 Data Model,表示 Data 对象在三维场景中的显示模型。mitkSurfaceModel 即是对应于 mitkMesh 对象的显示模型。

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。

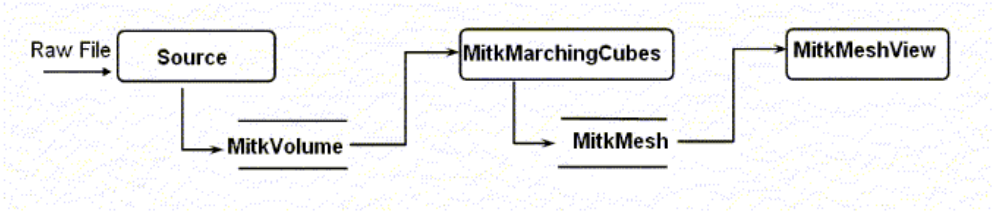
mitkView: mitkView 是一个 View,封装的是屏幕上的一个窗口,它具有能显示多个 Model 对象的能力,并且支持鼠标的交互式操作。要使用它必须先包含 mitkView.h 头文件,关于它的接口函数的详细说明请参看 MITK 的帮助文件。

三维重建实例

首先,我们先看一下这个例子所完成的功能,下面这张图是例子运行时的主界面。您可以用鼠标进行操作：左键旋转、中键平移、右键缩放



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，见下图：

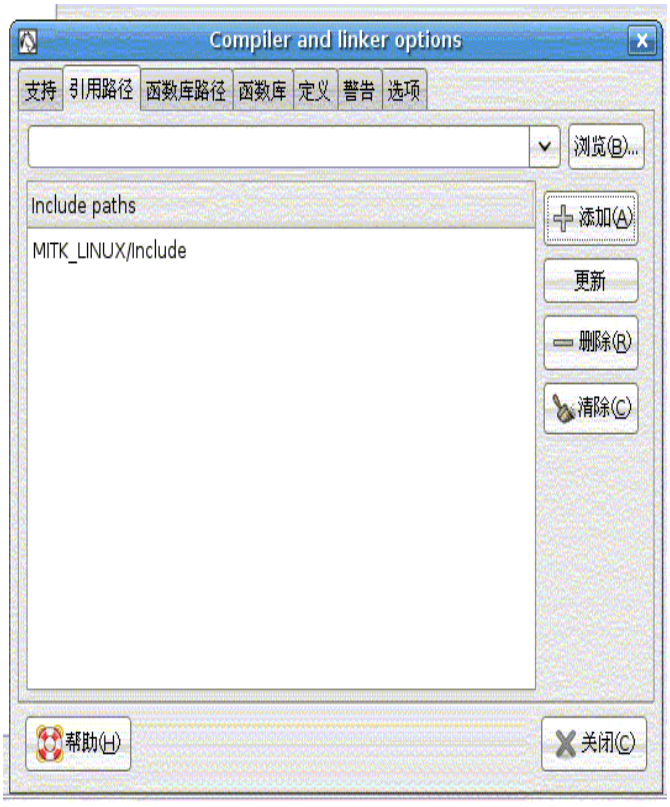


要注意的是，在这里为了演示一下 mitkVolume 的接口函数的使用（因为它是一个非常重要的数据类），在 Source 这一块并没有使用 MITK 中的 mitkRawReader 类，而是直接从文件里面读数据，然后填充至 mitkVolume 中。

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。

第一步,是在 Anjuta 的 IDE 开发环境中新建一个普通/终端机项目,
项目名称：MitkTest
程序名字：Test_SurfaceRendering
程序语言：C 及 C＋＋
目标类型：可执行的程序
项目描述：略
项目选项：把复选框中的对号都去掉
生成项目。

第二步，设置头文件及库的路径
打开 Anjuta 菜单栏中的设定/编译器及连接器设定，
引用路径：MITK_LINUX 解压路径/MITK_LINUX/Include
函数库路径： MITK_LINUX 解压路径/Lib
函数库：mitk,GL,GLU,X11



第三步：书写源代码，

1 在左侧的项目窗口中，在工程自动创建的 main.cc 文件中输入测试代码：

```
#include <iostream>
#include "iomanip.h"
#include "X11/Xlib.h"
#include "X11/Xutil.h"
#include "mitkView.h"
#include "mitkIM0Reader.h"
#include "mitkVolume.h"
#include "mitkImageModel.h"
#include "mitkMesh.h"
#include "mitkMarchingCubes.h"
#include "mitkSurfaceModel.h"
#include "mitkSurfaceRendererUseVA.h"
#include "mitkSurfaceRendererStandard.h"



int main()

{
	//建立和本机之间的连接 dpy
	Display *dpy = NULL;
	dpy = XOpenDisplay(0);

	//检验 dpy 是否有效
	if (dpy == NULL)
	{
		std::cout<<"cannot open display in main()"<<endl;
		exit(1);
	}

	//为 mitkView 窗口建立父窗口，窗口大小 512＊512
	Window parent = XCreateWindow(dpy, XDefaultRootWindow(dpy), 0, 0, 512, 512, 0, CopyFromParent,
	CopyFromParent, CopyFromParent, 0, NULL);
	//选择父窗口 parent 所需要产生的 event 类型
	XSelectInput(dpy, parent, KeyPressMask | StructureNotifyMask);

	// Mapping Window 将父窗口正式影射到显示器画面
	XMapWindow(dpy, parent);

	//输出父窗口 ID 号
	cout <<”ParentId : ” <<parent << endl;
```

```
//建立 mitkView
mitkView *view ;
view= new mitkView;

//设置 View 的属性
view->SetDisplayId(dpy); //设置 displayID
view->SetParent(&parent); //设置父窗口为 parent
view->SetPosition(0,0); //初始位置为 0*0 处
view->SetSize(512, 512); //窗口大小为 512*512
view->SetBackColor(255,255,255); //背景颜色为白色
view->Show();

//创建文件读取对象
mitkIM0Reader *reader = new mitkIM0Reader;

//创建体数据对象
mitkMesh *mesh=NULL;

//读入数据进 m_Volume,数据路径为您的数据所在的路径
reader->AddFileName("MITK_LINUX/TestData/4392.im0");
if (reader->Run())
{
	//建立表面数据对象
	//建立 MarchingCube 算法对象
	mitkMarchingCubes *mc = new mitkMarchingCubes;
	// 为 Marching Cubes 算法设置阈值
	mc->SetThreshold(100, 255);
	// 设置输入数据
	mc->SetInput(reader->GetOutput());
	// 从 mitkMarchingCubes 算法得到输出结果
	if (mc->Run())
	{
		mesh = mc->GetOutput();
	}
	mc->Delete();
}
reader->Delete();

	//建立表面数据模型
	mitkSurfaceModel *model = new mitkSurfaceModel;

	// 设置表面材质属性
	model->GetProperty()->SetAmbientColor(0.75f, 0.75f, 0.75f,1.0f);
	model->GetProperty()->SetDiffuseColor(1.0f, 0.57f, 0.04f,1.0f);
```

```
        model->GetProperty()->SetSpecularColor(1.0f, 1.0f, 1.0f,1.0f);
        model->GetProperty()->SetSpecularPower(100.0f);
        model->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
        mitkSurfaceRenderer *renderer = new mitkSurfaceRendererStandard;
        model->SetRenderer(renderer);
        model->SetData(mesh); mesh=NULL;
        view->AddModel(model);
        view->Update();

        //显示图片
        //输出当前的等待消息数
        cout<<XPending(dpy)<<endl;

        XEvent xevent;

        int res;
        //建立消息处理主循环
        while (1)
        {
            //得到下一个消息
            XNextEvent(dpy, &xevent);
            //输出消息类型
            cout << "Event Type: " << xevent.type << endl;
            //若消息为关闭父窗口，跳出消息循环，释放 mitkView
            if (xevent.type == DestroyNotify)
            {
                cout << "destroy" << endl;
                break;
            }
            //调用 view 中的消息处理函数进行处理
            res =   view->HandleXEvent(&xevent);
        }
    view->Delete();
    XCloseDisplay(dpy);
    return 0;
}
```

第四步 创建项目并执行程序

创建项目：菜单栏 / 创建 / 创建项目（或直接按 Shift+F11）执行程序：菜单栏 / 创建 / 执行程序 (或直接按 F3)，就会出现例子程序的界面，至此整个例子程序结束。