doi: 10.6046/gtzyyg.2014.03.30
引用格式: 陈超,姚国清. 可扩展遥感图像处理软件框架的研究与设计[J]. 国土资源遥感,2014,26(3):182 - 186. (Chen C, Yao G Q. Study and design of a scalable software framework for remote sensing image processing[J]. Remote Sensing for Land and Resources,2014,26(3):182 - 186.)

可扩展遥感图像处理软件框架的研究与设计

陈 超, 姚国清
(中国地质大学(北京)信息工程学院,北京 100083)

摘要: 为了适应不同用户的需求及遥感技术的发展,提出一种可扩展遥感图像处理软件框架的设计方法。通过构建软件平台,规定统一的遥感图像格式及访问方式,提供基础功能的应用程序接口(application programming interface,API); 制定扩展模块的调用接口,将可执行程序(EXE)和动态链接库(dynamic link library,DLL)以扩展模块的形式添加到软件平台中,并完成相应菜单的动态添加及响应,实现平台与扩展模块、扩展模块与扩展模块之间的数据交换。科研人员可依托该软件平台现有功能进行二次开发,避免重复劳动; 也可仅针对新的研究成果快速地编写程序,并方便地添加到软件平台中。

关键词: 遥感; 应用程序接口(API); 动态链接库(DLL)
中图法分类号: TP 751.1　**文献标志码**: A　**文章编号**: 1001 - 070X(2014)03 - 0182 - 05

0 引言

随着科技的发展,遥感技术已经形成了一个从地面到空中乃至空间,从数据获取、处理到解译分析和应用,对全球进行探测和监测的多层次、多视角、多领域的观测体系,成为获取地球资源与环境信息的重要手段[1]。随着对如此庞大数据的处理速度的提升,遥感数字图像处理技术也得以迅速发展。

任何一套遥感图像处理软件都不能完全满足所有用户的需求; 并且随着数字图像处理技术的发展,图像处理软件还需要不断地添加新的功能,如果对软件进行修改,然后重新编译、发布,不仅费时费力,而且还可能出现版本兼容性方面的问题[2]。

本文提出一种可扩展遥感图像处理软件框架的设计方法,可将执行程序(EXE)和动态链接库(dynamic link library,DLL)程序以扩展模块的形式添加到软件平台中; 并动态添加菜单项,以约定方式对其调用,利用文件映射技术进行数据交换,以便科研人员能够快速地将研究成果编写成新的功能模块,并方便地添加到遥感图像处理软件系统中。

1 系统框架设计

可扩展遥感图像处理系统主要实现框架功能,构成一个遥感图像处理软件平台。该系统中需要规定统一的遥感图像格式及其访问模式; 通过应用程序接口(application programming interface,API)以平台功能方式提供基础图像处理功能,定义对外开放的内部程序接口和外部程序调用接口; 其他功能均可由扩展模块来实现。系统的整体结构如图 1 所示。

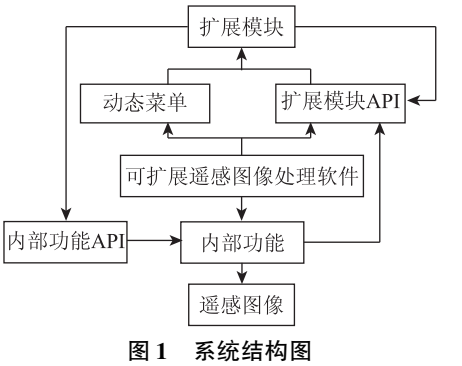

图 1　系统结构图

Fig. 1　Diagram of system structure

1.1 数据接口

遥感图像的格式多种多样[3],为了方便图像处理功能的开发,在遥感图像处理软件平台中规定一种统一的图像格式,并且提供格式转换功能,在处理图像之前将图像转换成统一格式。主程序将实现读写遥感图像的功能(比如获取头文件信息,获取某波段、某区块的图像数据),扩展模块时不需要了解遥感图像的具体格式,只要通过接口直接调用主程序中的功能就能获取到图像信息。

收稿日期: 2013 - 07 - 05; **修订日期**: 2013 - 08 - 15

1.2 基础平台功能

主程序可以提供图像文件访问、图像显示等基础功能,还可以包括以下功能: ①图像信息查看,包括图像直方图查看和图像波谱查看等; ②图像增强功能,包括图像拉伸、图像平滑、图像锐化等功能; ③显示外部图像源与窗口控制,可以让扩展模块新建窗口或指定现有窗口并显示扩展模块组织的图像; ④图像裁剪、旋转与缩放功能。

1.3 API 的设计

软件平台应提供应用程序接口(API),主要包括遥感图像处理的一些基础功能,如上文提到的数据接口,基本的遥感图像增强、图像显示等。将这些功能封装到一个动态链接库(DLL)中,并开放 API 供扩展模块调用。专业人员可以软件平台为基础,进行二次开发,不必在基础功能上重复开发、浪费精力。平台和扩展模块之间、扩展模块和扩展模块之间都可以互相调用,完全开放、透明,使系统具有良好的可扩展性[4]。

1.4 扩展模块与动态菜单

扩展模块使用 C\C ++ 进行开发,可以以 EXE, DLL(或其他形式,只要定义相应 API 即可)的形式实现。在图像界面中通过动态菜单调用扩展模块, EXE 模块可以被直接调用,DLL 模块可以开放程序接口或者定义一个导出变量(包含接口信息)。科研人员可以随时将新功能添加到系统平台中(有新的研究成果时,可以依托平台现有功能,仅针对新的研究成果快速地编写程序,并方便地添加到系统平台中),新的功能模块可以实时地反映到系统平台的动态菜单中。整个系统平台的关键是扩展模块的添加与调用方式。

2 菜单的动态添加与响应

在图形界面中使用扩展模块功能,主要是通过动态添加的菜单来调用的。可以把菜单项和扩展模块的对应关系保存到一个配置文件中,每次程序启动时,根据这个配置文件来实现动态菜单初始化,这种方法需要开发人员自行开发一个编辑配置文件的工具; 还可以把所有扩展模块存储到同一个文件夹中,每次程序启动时扫描该文件夹,为每一个扩展模块动态添加一个菜单项。第一种方法需要额外开发一个工具,但是配置文件中可以保存多种信息(比如菜单的上下级关系),这样在添加菜单时可以更好地进行扩展模块的分类和显示; 而第二种方法虽然使用起来简单方便,但是在菜单中所有的扩展模块只能罗列在一起,不美观也不方便使用。

2.1 菜单的动态添加

动态添加菜单的操作可在应用程序类 CWinApp 的 InitInstance 函数中进行。添加菜单项的操作主要使用 AppendMenu 函数,使用该函数时要根据设计的菜单级别获取相应的父级菜单对象,设置该函数的第一个参数为 MF_POPUP 和 MF_STRING,可以分别添加下拉菜单和菜单项。在所有的菜单项添加完成之后,进行 DrawMenuBar 操作,这样动态添加的菜单就可以显示出来[5]。添加 2 级菜单的动态添加过程可以通过以下代码实现:

```
CMenu * pTopMenu = AfxGetMainWnd( ) - > GetMenu( ); //获取系统菜单指针
int nMenu = pTopMenu - > GetMenuItemCount( ); //获取现有菜单项个数
while (i < nMenuCount)
{
    szTitle. Format( _T( "% s" ) ,menuItem[ i ]. title) ;
    if( menuItem[ i ]. type = = TOP_MENU)
    {
        CMenu newMenu;
        newMenu. CreatePopupMenu( ) ;
        pTopMenu - > AppendMenuW( MF_POPUP, ( UINT) newMenu. m_hMenu, szTitle) ;
        newMenu. Detach( ) ; //菜单项与局部变量分离
        + + nMenu; //递增菜单数,以便让子菜单获取正确的父级菜单
    }
    else if ( menuItem[ i ]. type = = SUB_MENU)
    {
        //获取当前子菜单的父级菜单
        pParentMenu = pTopMenu - > GetSubMenu( nMenu - 1) ;
        //子菜单需要有相应的命令响应,需要一个全局唯一的 ID
        pParentMenu - > AppendMenuW( MF_STRING, newMenuID, szTitle) ;
```

```
            ++newMenuID;
        }
        ++i;
    }
    DrawMenuBar(AfxGetMainWnd()->m_hWnd)。
```

如果只进行添加菜单的操作,那么界面中的这些动态菜单项都是“灰色的”(不可用状态),而不像其他系统自带菜单项那样显示为可用状态;这是因为其他系统的程序对自带的菜单项进行统一管理,而通过上述方法添加的菜单项需要自行管理。对菜单状态进行自定义操作需要重写 CView 类中的OnCmdMsg函数,其中 nCode 参数表明本次命令的消息类型,当 nCode 值为 CN_UPDATE_COMMAND_UI 时,表示更新用户界面。开发人员可以根据具体需求有选择性地设置各个菜单项的状态。将所有动态添加的菜单“激活”的过程可以通过以下代码实现:

```
BOOL CView::OnCmdMsg(UINT nID,int nCode,void* pExtra,AFX_CMDHANDLERINFO* pHandlerInfo)
{
    if(nCode == CN_UPDATE_COMMAND_UI
        && nID >= IDM_DYNAMENU
        && nID <(IDM_DYNAMENU + cntNewMenu)
        && pHandlerInfo == NULL)
    {
        CCmdUI* cmdUI = (CCmdUI*)pExtra;
        cmdUI->Enable(TRUE);
        return TRUE;
    }
    return CView::OnCmdMsg(nID,nCode,pExtra,pHandlerInfo);
}
```

2.2 菜单的动态响应

通过相应设置,动态添加的菜单项即可正常显示;但是点击菜单项之后是没有反应的,因为还没有设置每个菜单项对应的命令,对菜单命令进行自定义操作也需要重写 CView 类的 OnCmdMsg 函数,当 nCode 参数为 0 时,表示对菜单进行响应。动态响应菜单命令需要维护一个全局变量,这个变量记录了每个菜单项与相应的扩展模块的对应关系。该变量的初始化可以在添加动态菜单的同时进行。假设扩展模块都是 EXE 形式,设置菜单命令响应的过程可以通过以下代码实现:

```
BOOL CView::OnCmdMsg(UINT nID,int nCode,void* pExtra,AFX_CMDHANDLERINFO* pHandlerInfo)
{
    if(nCode == NULL
        && nID >= IDM_DYNAMENU
        && nID < (IDM_DYNAMENU + cntNewMenu)
        && pHandlerInfo == NULL)
    {
    for(i = 0; i < cntNewMenu; ++i)
      {
        if(menuCmdList[i].ID == nID)
          {
            WinExec(menuCmdList[i].path,SW_SHOW);
            return TRUE;
          }
      }
    }
    return CView::OnCmdMsg(nID,nCode,pExtra,pHandlerInfo);
}
```

3 扩展模块的接口设计及数据交换

扩展模块可以以 EXE 或 DLL 的形式(当然也可以是其他形式)实现。对 EXE 程序的调用很简单,直接使用文件路径和 WinExec,ShellExecute 或者 CreateProcess 等函数即可; 而调用 DLL 程序则略微复杂,除了文件路径,还要知道程序接口,然后利用 LoadLibrary,GetProcAddress 和 FreeLibrary 函数调用。对其他形式扩展模块的调用只要定义相应的调用方式即可。

主程序中的部分功能(包括图像文件访问功能、图像显示功能、窗口控制等),需要开放给扩展模块。可以把这些功能封装到一个 DLL 中,并对扩展模块开放程序接口。

3.1 扩展模块的接口设计

调用扩展模块的接口稍显复杂。上述内部程序的接口由于已经做好规定,容易调用; 而扩展模块是随时添加的,对其他程序来说其接口是未知的,这就需要制定相应的调用规则(比如把所有扩展模块的接口都命名为“exFunc( )”),再根据扩展模块的路径进行调用; 或者在每个 DLL 中都构造一个结构统一的导出变量(这个变量中包含了接口的信息),再根据这个变量中的接口信息对接口进行调用。

3.2 数据交换

接口规则制定好之后,就要处理数据交换(或称“进程通信”)的问题。

主程序和扩展模块都是独立的进程。在 Windows 系统中,每个进程都有自己独立的内存空间,该独立的内存空间包含了所有的 EXE 模块或 DLL 模块的代码和数据以及动态内存分配的空间。每个进程的内存空间只能被该进程访问,其他进程则不能访问。如果要在进程间共享数据(也就是创建一块不同进程都能访问的内存),可以使用内核对象(因为内核对象由 Windows 系统内核所拥有,而不是由进程所拥有[6])。考虑到需要交换的数据量可能会很大,可以使用文件映射方式(当然也可以使用其他方式)进行数据交换。在程序初始化时使用 CreateFileMapping 函数开辟文件映射空间,然后使用 MapViewOfFile 函数创建文件视图,可以根据需求创建多个文件视图。在主程序运行过程中,根据需要更新文件视图的内容(也就是更新共享数据)、在主程序退出时,使用 UnmapViewOfFile 函数和 CloseHandle 函数,分别关闭文件视图和文件映射,防止内存泄漏。在读写文件视图时,可以使用互斥(Mutex)来控制对公共资源的访问。

值得注意的是,每个进程的虚拟地址空间都是有限的。在 Windows 系统中,32 位进程的虚拟地址空间通常为 2 GB,64 位进程的虚拟地址空间为 8 TB[7]。如果使用 32 位的程序,而遥感图像文件又大于 2 GB,则进行文件映射时就不能将整个图像文件装入内存。解决这个问题可以有 2 种方法: ①每次使用 MapViewOfFile 函数时,只把图像的一部分映射到内存空间即可; 程序使用图像的其他部分时,先使用 UnmapViewOfFile 函数解除上一个映射,然后再映射新的图像,这种方法需要一个高效灵活的图像分块读取算法; ②开发 64 位的程序。但考虑到目前 PC 机的内存容量一般不超过 16 GB,如果图像过大,映射到内存空间就会导致大量使用虚拟内存; 而大量的图像文件存储在硬盘空间中,会导致程序运行速度变慢。

4 结论

1)开发具有基本功能的软件平台,为可执行程序(EXE)和动态链接库(DLL)等扩展模块预留调用接口,在需要时开发扩展模块对平台进行扩充,可大大降低软件开发的复杂度,避免重复劳动。

2)统一遥感图像格式,制定图像读写方法,可降低图像处理的复杂度。

3)制定合理的应用程序接口(API)及其调用规则,使软件平台具有良好的可扩展性,并使各个模块间的独立性也得到很大提高,有利于模块的扩展与更新。

随着遥感技术的发展,遥感图像包含的信息量越来越大,处理海量遥感图像(快速格式转换、海量图像快速显示等)需要更好的算法,这方面还有待进一步的研究。

参考文献(References):

[1] 陈 红. 现代遥感技术的特点及应用[J]. 科学时代,2013(3):118.
Chen H. Features and application of modern remote sensing technology[J]. Science Times,2013(3):118.

[2] 王宏勇,张 杰,张永生. 遥感图像处理软件设计中插件技术的应用[J]. 海洋测绘,2005,25(2):34 – 36.
Wang H Y,Zhang J,Zhang Y S. The application of plug – in technique in the remote sensing image processing software[J]. Hydrographic Surveying and Charting,2005,25(2):34 – 36.

[3] 郜风国,冯 峥,唐 亮,等. 基于 GDAL 框架的多源遥感数据的解析[J]. 计算机工程与设计,2012,33(2):760 – 765.
Gao F G,Feng Z,Tang L,et al. Multisource remote sensing data analysis based on GDAL frame[J]. Computer Engineering and Design,2012,33(2):760 – 765.

[4] 方 晓,董 辉,孙士新,等. 软件开发本体构建与模块化的应用研究[J]. 湖南工业大学学报,2013,27(1):71 – 76.
Fang X, Dong H, Sun S X, et al. The applied research of software development ontology construction and modularity[J]. Journal of Hunan University of Technology, 2013, 27(1): 71 – 76.

[5] 郝丽萍. VC 中动态菜单及命令响应的真正实现[J]. 福建电脑,2010(4):104 – 106.
Hao L P. Implement of dynamic menu and command response with visual C ++ [J]. Fujian Computer, 2010(4): 104 – 106.

[6] 李朝中. 进程间通信中内存映射文件[J]. 电脑编程技巧与维护,2010(3):47 – 51.
Li C Z. File mapping in process communication[J]. Computer Programming Skills and Maintenance, 2010(3): 47 – 51.

[7] Microsoft Corporation. Virtual address spaces[EB/OL]. [2013 – 06 – 11]. http://msdn. microsoft. com/en – us/library/hh439648(v = vs. 85). aspx.

Study and design of a scalable software framework for remote sensing image processing

CHEN Chao, YAO Guoqing
(*College of Information Engineering*, *China University of Geosciences*, *Beijing* 100083, *China*)

Abstract: In order to adapt the remote sensing technology to different users' needs and remote sensing technical development, the authors proposed a scalable software framework design method for remote sensing image processing. A software platform was developed, the unified remote sensing image format and access methods were formulated, the application program interfaces (API) of basic functions and expansion modules were provided, and the menus and corresponding responses were dynamically achieved when the expansion modules such as the executable program (EXE) and dynamic link library (DLL) were added to the platform. The data could be exchanged between the platform and expansion modules or between the extension modules. Researchers can develop new expansion modules which can be easily added to the platform based on the available functions in the platform without duplication of effort.

Key words: remote sensing; application programming interface(API); dynamic link library(DLL)

第一作者简介: 陈 超(1988 –),男,硕士研究生,主要从事计算机应用技术研究。Email: imagpc2hs@ gmail. com。

(责任编辑: 刘心季)