



ENVI/IDL 二次开发

航天星图科技（北京）有限公司

support@imagnetekinfo.com



主要内容



ENVI /IDL二次开发介绍

.....●



ENVI /IDL二次开发基础

.....●



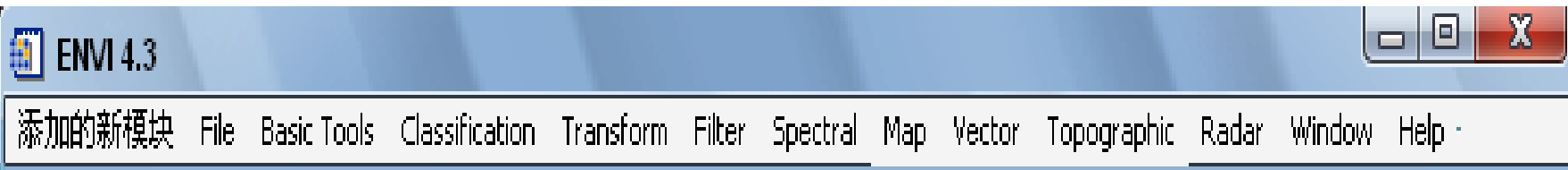
如何进行ENVI二次开发

.....●



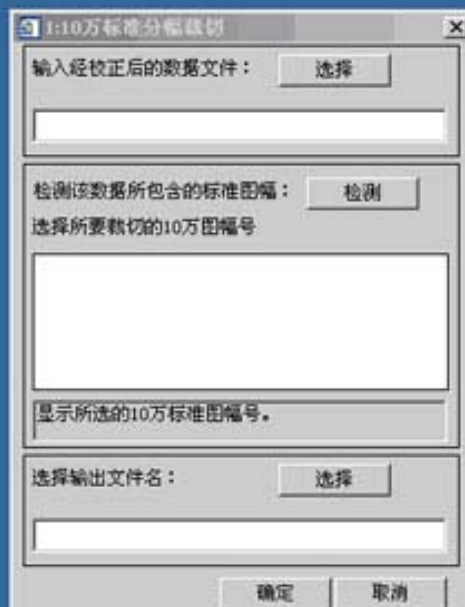
ENVI /IDL二次开发介绍

- 通过使用IDL来扩展ENVI的功能
- 两种模式，三种方法
 - 保留ENVI的界面
 - 波段和波谱运算：利用ENVI波段和波谱运算功能调用我们编写的函数
 - 用户函数：在ENVI下自己编写菜单来调用我们自己编写的函数。要处理I/O接口、事件控制和修改菜单。



ENVI /IDL二次开发介绍

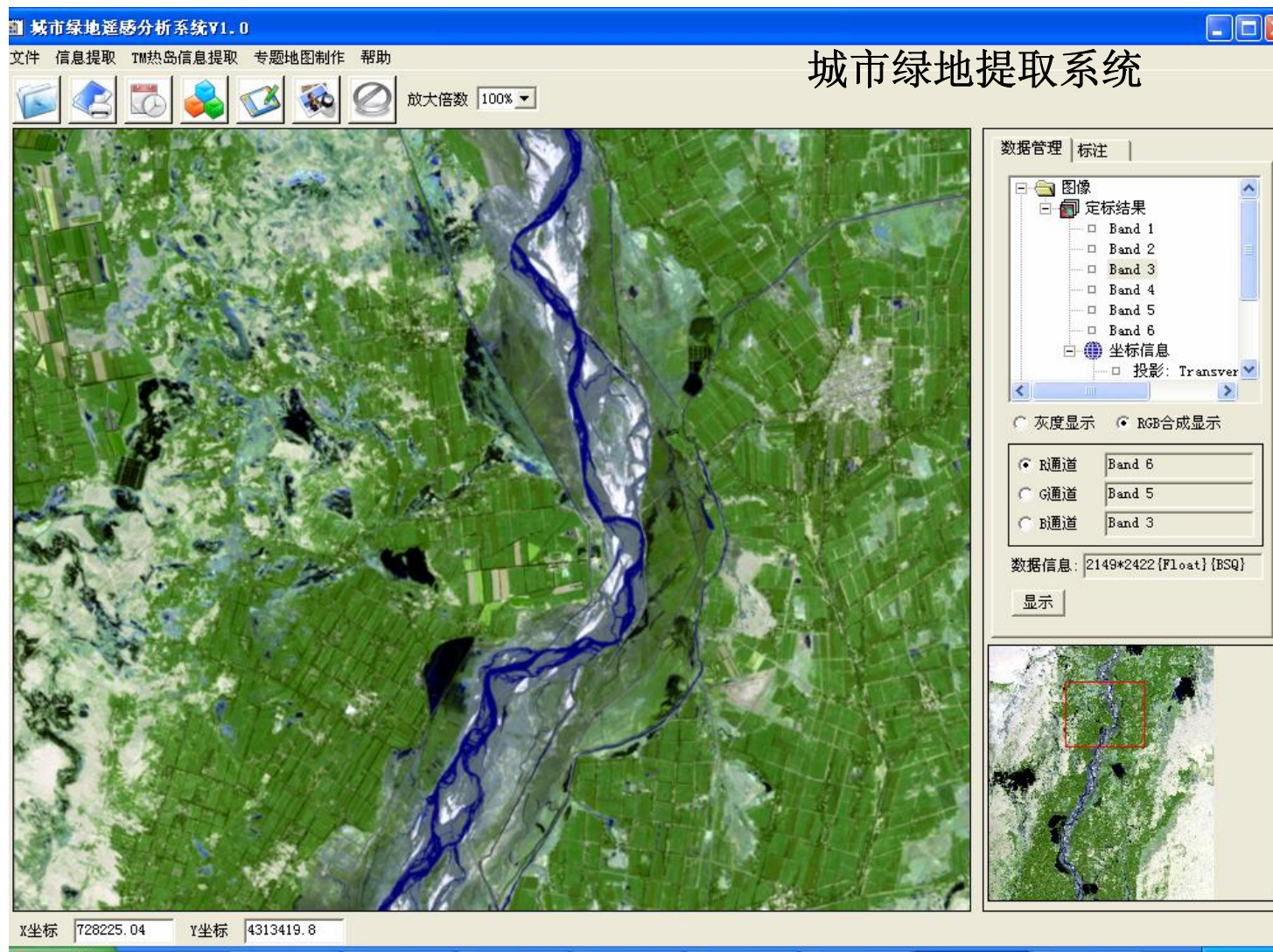
- 看不见ENVI的界面（用ENVI定制属于自己的系统）
 - 批处理方法



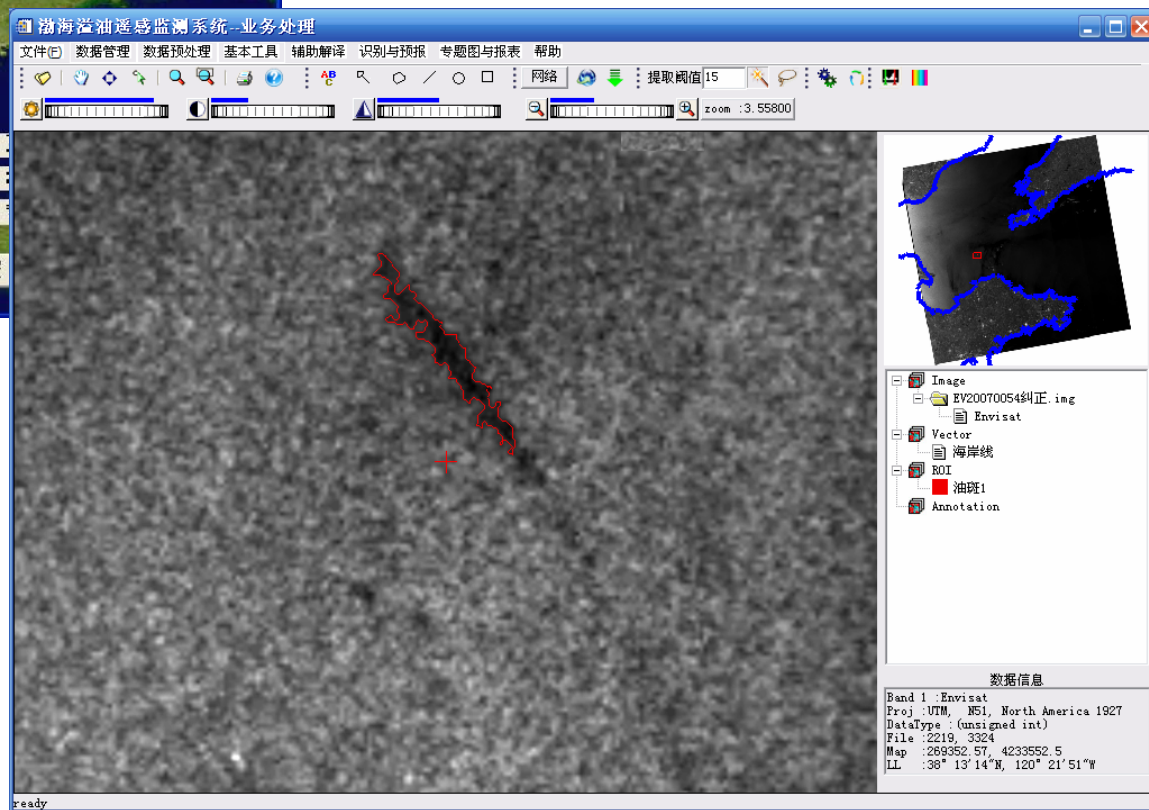
生态环境监测系统



ENVI /IDL二次开发介绍



ENVI /IDL二次开发介绍



ENVI /IDL二次开发基础

■ ENVI程序中的通用关键字

■ FID

FID是一个长整型的标量。FID可以用于打开或选择文件。所有对该文件进行操作的ENVI程序都是通过FID完成。如果文件打开失败，则FID返回为-1。

```
ENVI_OPEN_FILE, file, r_fid=fid
```

```
IF (fid EQ -1) THEN RETURN
```



ENVI /IDL二次开发基础

■ R_FID和M_FID

ENVI处理程序产生的影像结果也包括一个R_FID，或者称为返回FID关键字。如果结果是存在内存中的，R_FID关键字是访问数据的唯一方法。

进行掩模的处理时还包括一个M_FID，或者称为掩模关键字，用于确定用于掩模波段的文件。



ENVI /IDL二次开发基础

■ DIMS

DIMS关键字是一个5个元素长整型数组。它定义了处理数据的空间子集。当确定一个文件的时候，必须同时确定该文件的空间范围。

DIMS[0] 存储一个打开的ROI区域的指针，仅在ROI被定义的时候使用，其它时候设为-1L

DIMS[1] 采样的起始位置 Sample start

DIMS[2] 采样的终止位置 Sample end

DIMS[3] 行的起始位置 Line start

DIMS[4] 行的结束位置 Line end

如DIMS=[-1, 0, ns-1, 0, n1-1]

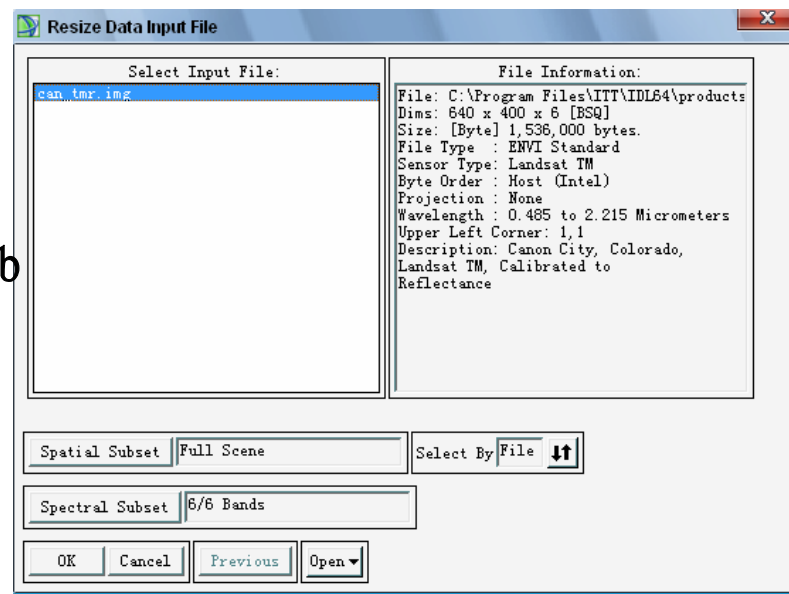


ENVI /IDL二次开发基础

■ POS

POS关键字定义了用于处理的波段位置，是一个变长的长整型数组。波段从0开始，例如，要处理第三波段和第四波段，POS=[2, 3]。

```
file=dialog_pickfile(title='',/read)
ENVI_OPEN_FILE, file, r_fid=fid
IF (fid EQ -1) THEN RETURN
ENVI_FILE_QUERY, fid, ns=ns, n1=n1, nb=nb
dims=[-1, 0, ns-1, 0, n1-1]
pos=lindgen(nb)
```



ENVI /IDL二次开发基础

■ 常用ENVI功能函数介绍

■ 文件管理

ENVI_PICKFILE: 产生一个提示用户选择文件的对话框，并返回用户所选择的文件名

ENVI_SELECT: 产生对话框提示用户从ENVI中已经打开的文件中选择一个文件，并返回用户所选择文件的FID，该函数还可以返回DIMS和POS的值

ENVI_OPEN_FILE: 该函数返回一个文件的FID，并将文件信息添加到可用波段列表中

ENVI_FILE_MNG: 该函数可以打开、关闭或者删除硬盘上的文件。无需用户干预

ENVI_GET_FILE_IDS: 该函数返回所有当前打开的文件的FID



ENVI /IDL二次开发基础

■ 打开外部文件格式

ENVI_OPEN_DATA_FILE: 该函数打开ENVI所支持的外部文件（无ENVI头文件）并返回FID

■ 获取数据

ENVI_GET_DATA: 该函数从一个打开的文件中获取影像数据。它每次只返回某一波段的数据，数据的范围由DIMS关键字控制。

ENVI_GET_SLICE: 该函数从一个打开的文件中获取波谱影像数据，它返回影像某一行所有波段的数据值。结果以BIP或BIL的格式返回



ENVI /IDL二次开发介绍基础

■ 生成ENVI格式的文件

■ 将数据保存到内存

ENVI_ENTER_DATA: 该函数将IDL数组中的数据输入到内存中，并通过可用波段列表进行管理。

■ 将数据存入硬盘

使用IDL的WRITEU函数写入数据

ENVI_SETUP_HEAD: 使用该函数写某个影像数据的头文件



如何进行ENVI的二次开发

■ 二次开发的三种方法

■ 波谱、波段函数

扩展ENVI功能的最简单方法，波段和波谱运算函数，不需要处理文件I/O，不需要进行事件控制，不需修改菜单，用户只需编写运算函数部分内容，其它由ENVI进行管理。

■ 批处理模式

不出现ENVI的菜单界面，通过调用ENVI提供的非函数来实现ENVI所提供的功能。



ENVI /IDL二次开发介绍

■ 用户函数

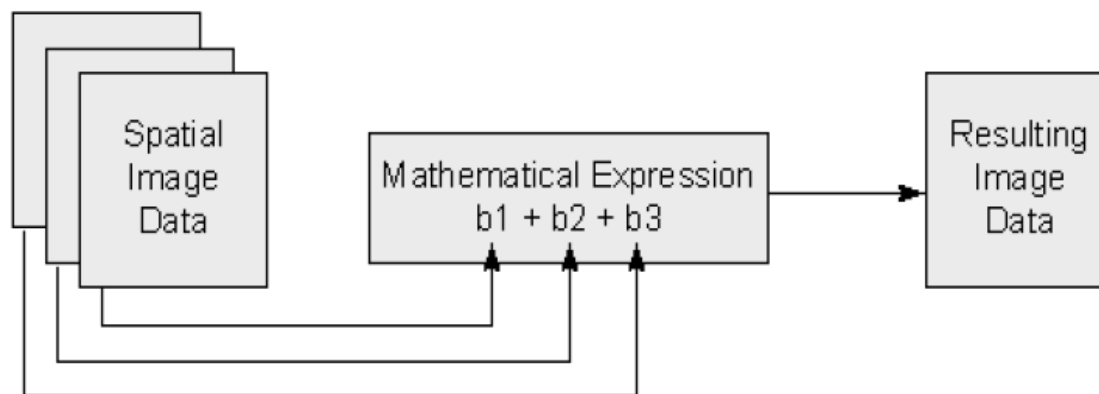
用户函数可以用IDL、C、Fortran或者其它的高级语言编写，并集成到ENVI软件中，通过ENVI的菜单来执行。用户函数可以通过ENVI获得输入数据，并将结果直接输入到ENVI中。

用户函数包括了部件的定义，事件的处理，以及处理程序。用户函数和ENVI菜单的一个按钮联系起来，并像ENVI的其它函数一样执行。



波段、波谱运算

■ 波段运算



```
Function bm_func,b1,[b2,...,bn, parameters and keywords]  
    processing steps  
    return,result  
end
```

ndvi.pro



波段、波谱运算

■ 波谱运算

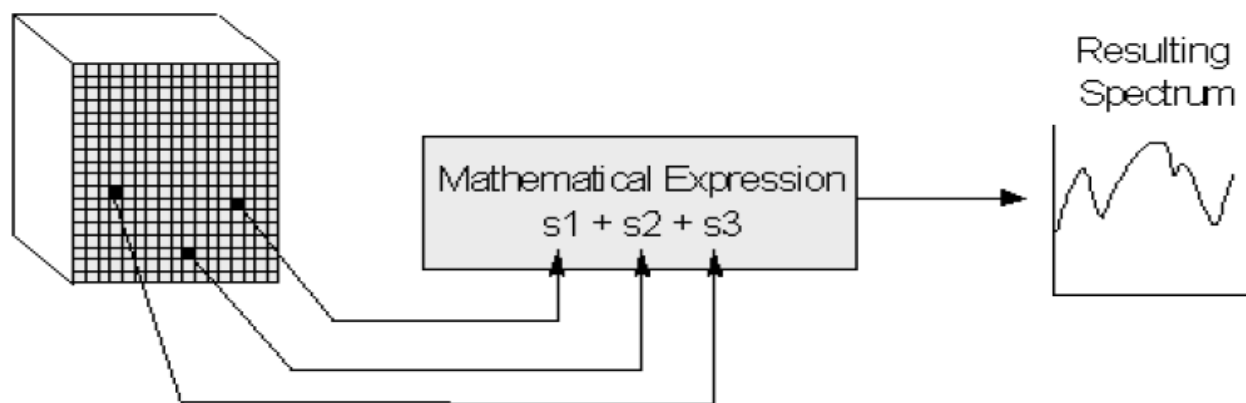


Figure 2-2: Spectral Math Processing — Addition of Three Spectra

```
function sm_func, s1, [s2,..., sn, parameters and keywords]  
    processing steps  
    return, result  
end
```

sm_ration.pro



ENVI批处理模式

■ 批处理模式简介

- 批处理模式的ENVI和正常模式下没有什么区别，只是通过一系列的特定的函数库来执行ENVI的功能。为了使用这些函数，必须首先将它们恢复到IDL内存中。因此为了正确获取这些函数，有必要了解一下ENVI程序的结构。
- ENVI功能文件由大约50多个小的IDL save文件组成，这些文件是包括数据和编译后的程序的二进制文件。这些save文件存放在ENVI安装目录下的Save目录下。ENVI的核心save文件包括ENVI的基本功能函数，动态运行函数以及ENVI运行所需的内部变量。



ENVI批处理模式

■ 如何开始批处理模式

恢复ENVI save核心文件

`envi, /restore-base-save-files`

开始批处理模式

`envi_batch_init, log_file='batch.txt'`

退出批处理模式

`envi_batch_exit`

■ 批处理的例子

`bt_init.pro`

`batch_stats1.pro`

`satstrch.pro`



用户函数

■ 用户函数简介

- 用户函数允许用户为ENVI添加新的功能并通过ENVI的菜单进行访问。用户能够添加任意数量的用户函数，并且每个函数都可以获得它自己的菜单选项。
- 用户函数是事件处理程序。因此，所有的ENVI用户函数必须遵循事件处理的基本规则，即用户函数定义时必须加上一个附加的变量来接受事件结构。



用户函数

■ ENVI菜单修改

使用任何文本编辑器就可以打开envi.men文件。文件的结构如下所示：

```
0 {File}
  1 {Open Image File} {open envi file} {envi-menu-event}
  1 {Open Vector File} {open vector
file} {envi-menu-event}
  1 {Open External File}
    2 {Landsat}
      3 {Fast} {open fast tm} {envi-menu-event}
      3 {GeoTIFF} {open tiff} {envi-menu-event}
      3 {HDF} {open envi file} {envi-menu-event}
      3 {NLAPS} {open nlaps} {envi-menu-event}
```

每一行开始的数据定义了菜单项的层次。0表示最顶层，1表示一级子菜单，2表示二级子菜单，依次类推。



用户函数

- {Open External File} 第一个大括号括起来的部分定义了显示在菜单上的内容。
- {open envi file} 第二个大括号括起来的部分定义了为菜单项所赋给的用户值。用户值在同一用户函数处理多个菜单项时非常有用，可以区别那个菜单项被选择。
- {envi_menu_event} 第三个定义了菜单项事件处理程序的名称，即编写的用户函数名。此处使用的是用户函数名，而不是用户函数所在的文件名，所以没有后缀。
- 需要注意的部分：用户值在大多数ENVI的程序中是需要的，要保持用户值的唯一性。但当编写用户函数时，大多数情况下，用户值部分是没有用的，这时候，可以将用户值设为和用户函数名一致，也可以将它设置为 {not used} 等醒目的标示。
- 修改ENVI菜单
添加如下部分：
0 {MyFunctions}
1 {Basic File Info} {not used} {file_info}

file_info.pro



用户函数

■ 为用户函数添加小部件

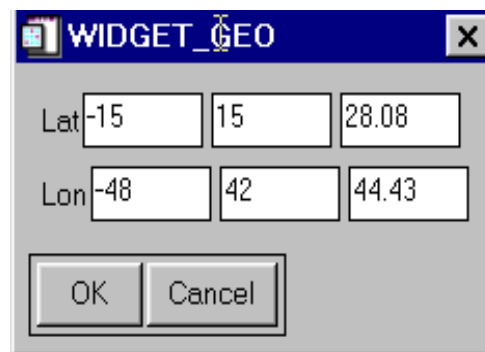
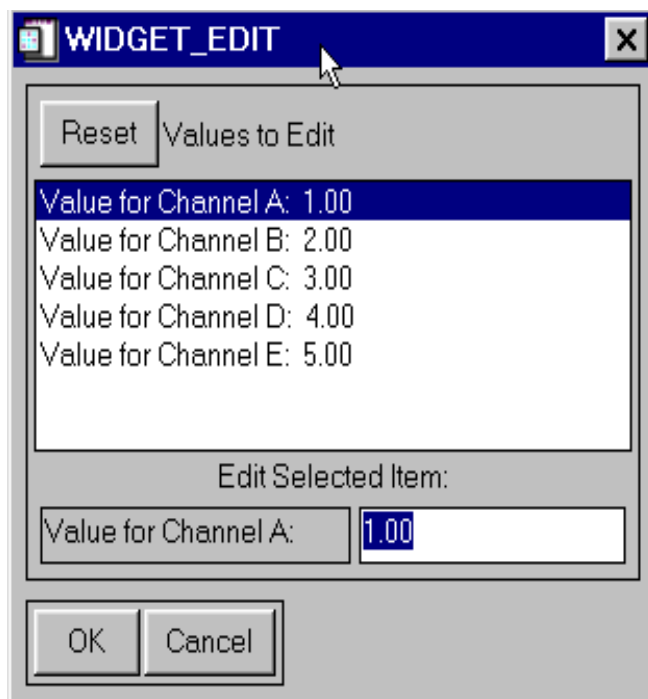
- ENVI包括了20多种的小部件，可以为用户函数所用。大多数的函数以WIDGET-开头。
- ENVI_PICKFILE: 用于从硬盘上选择一个文件。可以用来收集任意类型的文件名。
- ENVI_SELECT: ENVI标准的文件选择对话框，用来选择一个打开的文件，确定空间和光谱子区，以及掩模波段。它也包括了一个打开按钮，能够允许用户从硬盘上打开一个新的文件。

pro 用户函数名, event



用户函数

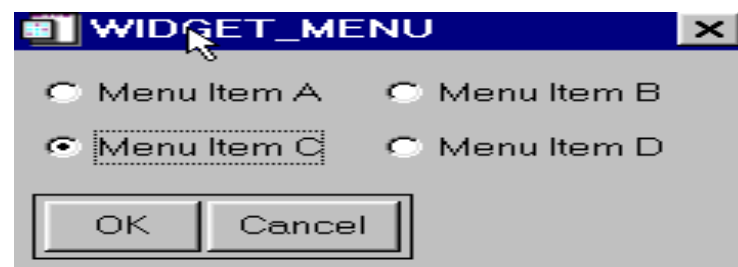
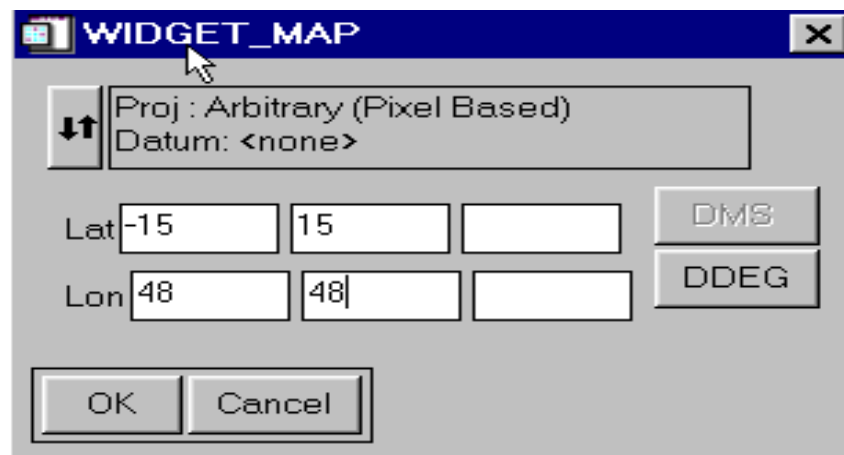
- WIDGET_EDIT: 提供了一个部件从列表中选择项目



- WIDGET_GEO: 用于提示用户选择经纬度值。

用户函数

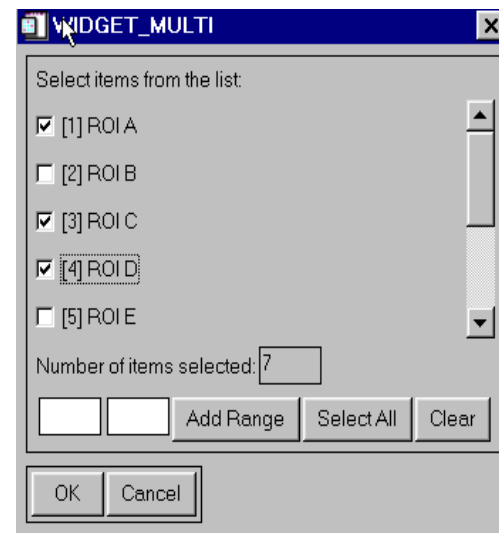
- WIDGET_MAP: 用于编辑地图坐标和投影



- WIDGET_MENU: 一个复合部件，用来生成一些复选或者非复选的按钮

用户函数

- WIDGET_MULTI: 用于多项选择，分类的ROI选择中常用

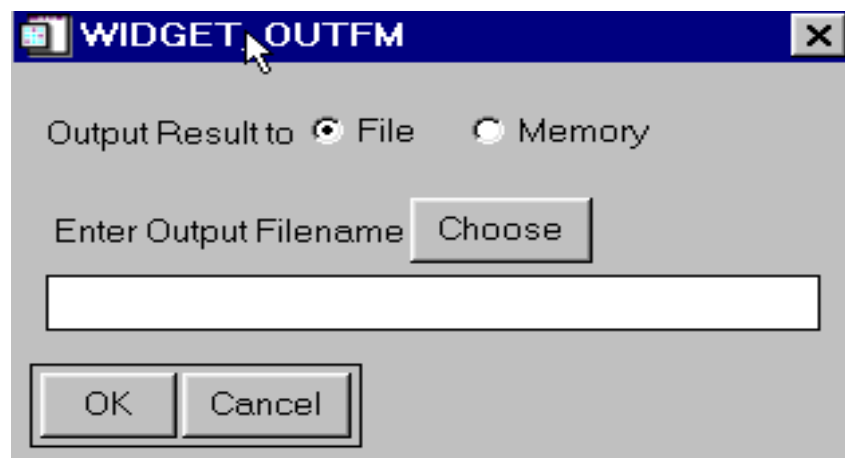


- WIDGET_OUTF
用于选择一个输出文件名

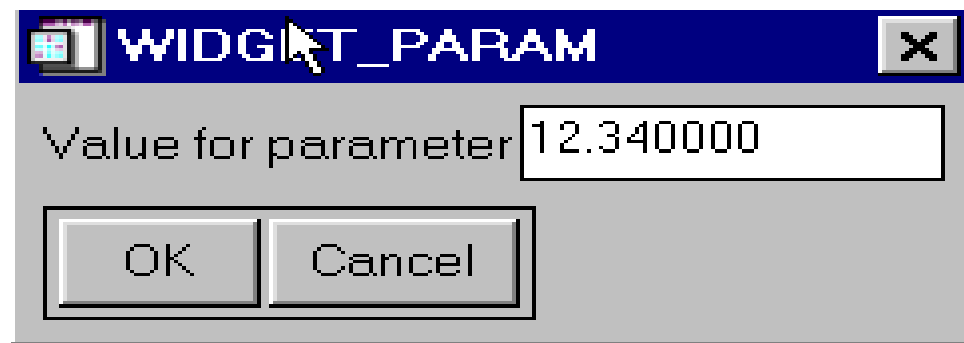


用户函数

- WIDGET_OUTFM: 用于选择一个输出文件名或是输入到内存

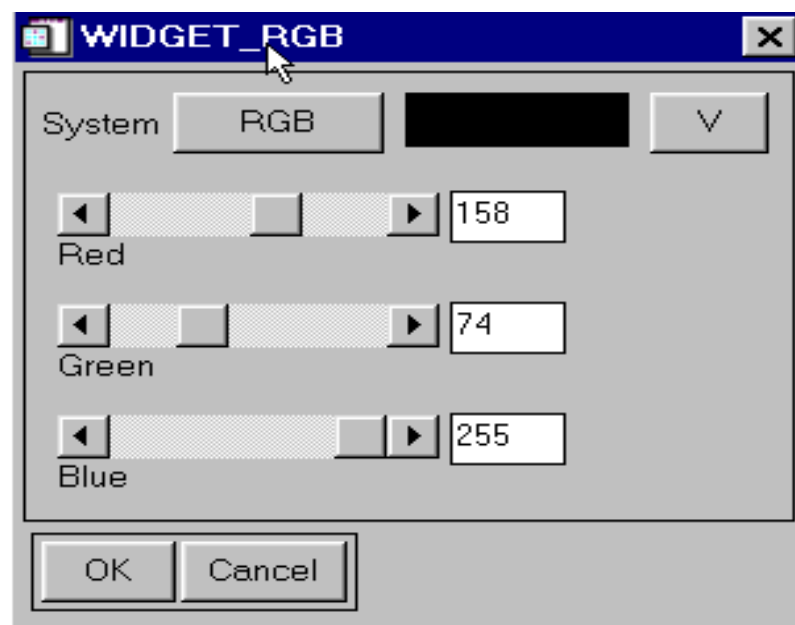
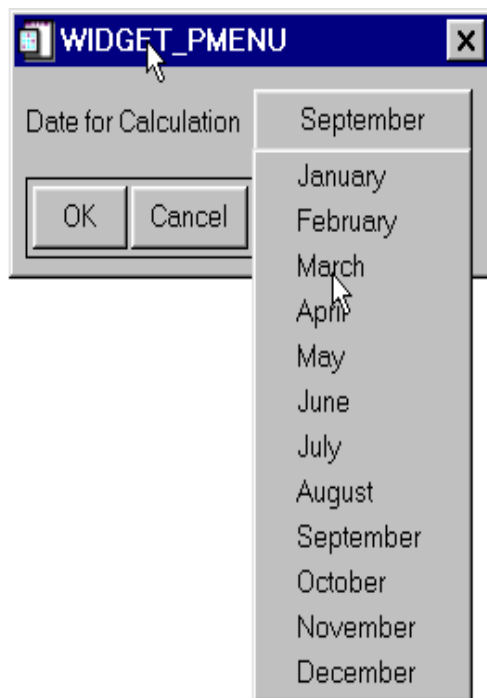


- WIDGET_PARAM: 用于输入数字参数



用户函数

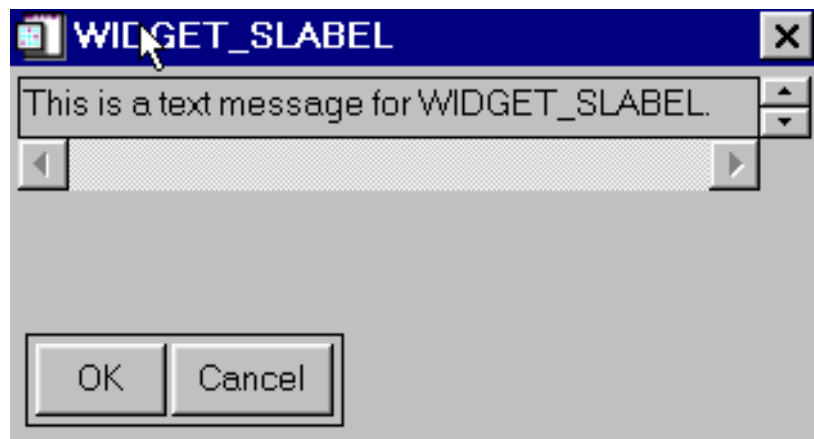
■ WIDGET_PMENU: 提供下拉菜单



■ WIDGET_RGB: 用于修改RGB颜色值

用户函数

- WIDGET_SLABEL: 用于显示文本信息，并具有滚动条



- WIDGET_STRING: 用于输入字符串的复合部件



用户函数

■ Widget部件自动事件管理程序

■ WIDGET_AUTO_BASE

如果要创建事件自动管理的部件构架，必须通过WIDGET_AUTO_BASE创建顶级BASE，在构建GUI的过程中使用的其它BASE使用原来的WIDGET_BASE函数创建。

■ AUTO_WID_MNG

在通常的IDL部件中，一旦GUI被定义，XMANAGER程序就会被调用进行部件的注册并进行部件事件的检测，这种情况下程序员要对不同的事件进行管理。

而在自动事件管理的ENVI程序中，通过调用AUTO_WID_MNG函数进行部件的注册，检测事件，并以结构的形式返回用户输入的值

test_widgets.pro



用户函数

■ 用户函数中错误的捕获

对用户函数运行时对可能发生的错误进行捕获，就是错误时弹出错误提示。

避免常见错误：

- (1) Cancel按钮是否被选择
- (2) 尽量首先设置好变量默认值
- (3) 使用where时，不要忘记使用count参数确保合法的结果被返回。



使用Catch函数进行非 I /O错误的捕获

系统变量! error_state里包含了最新的错误信息, 错误发生时error被更新为错误代码(初始为0), 程序跳转到catch语句并从那里开始执行

```
Catch, error
```

```
IF (error NE 0) THEN BEGIN
```

```
; 提示系统变量里的错误信息
```

```
ok = DIALOG_MESSAGE(!error_state.msg, /cancel)
```

```
IF (STRUPCASE(ok) EQ 'CANCEL') THEN return
```

```
ENDIF
```



用户函数

■ 与显示窗口进行交互

ENVI中每一个三窗口的显示组都能够通过一个唯一数字标识DN进行区别。一旦获得某一显示组的DN值，就可以使用ENVI提供的函数，获取显示组的信息。

- ENVI_DISP_QUERY: 该函数能获取当前显示影像的基本信息，包括影像文件的FID，空间分辨率，影像的显示类型（RGB，灰度或分类），显示的波段位置，以及三个窗口的大小



用户函数

- ENVI_GET_IMAGE: 该函数类似于ENVI_GET_DATA函数，但它用于从显示窗口中返回数据。给定波段位置，维度，以及DN值，能够返回拉伸后的灰度值。（保存图像）
- DISP_GET_LOCATION, 该函数返回当前选定的像素的位置
- DISP_GOTO, 该函数移动Zoom窗口到一个指定的位置，并在必要的情况下更新Image和Scroll窗口

example_disp_operation.pro



用户函数

- 在用户函数中使用影像分块技术
 - 为确保用户函数也能够处理任意大小的数据文件。所有的ENVI用户函数能够通过ENVI内建的分块函数获取数据。
 - ENVI的分块来自于三种格式：BSQ格式，BIL格式以及BIP格式。ENVI还提供了进度条部件来显示分块的处理情况。



用户函数

- ENVI分块处理将输入数据分成很多同样大小的单元
- 分块方式可以是空间方式也可以是波谱方式。一个空间分块的大小是 $\text{lines} \times \text{sample}$ ，而波谱分块的大小是 $\text{Sample} \times \text{band}$ 。通常BSQ采用空间分块技术，其他的采用波谱分块技术

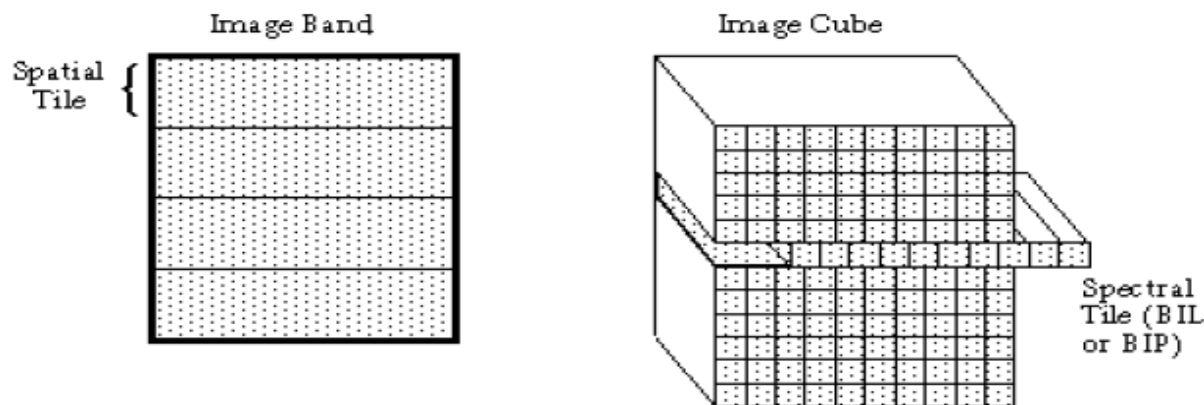


Figure 4-19: Illustration of Spatial and Spectral Tiles

用户函数

- 空间分块近似等于按输入波段对影像进行分块，不用考虑文件的存储方式而进行空间处理。访问单一文件的多个波段，所有波段将拥有同样数目的空间分块。
- 当进行邻域处理时，空间分块也可以设定重叠的行数。重叠行仅加在每个分块的顶部，在整个波段作为一个分块时，没有重叠行。例如：进行 3×3 卷积时，需要一行重叠来处理上一个分块的最后一行。



用户函数

- 分块处理的步骤如下:

获取分块需求: ENVI_INIT_TILE

获取分块输入数据, ENVI_GET_TILE

当所有的分块数据都处理完毕, 释放分块需求, ENVI_TILE_DONE

spat_tile.pro



用户函数

■ 保存结果

输出文件通过使用IDL程序OPENW写入，在调用OPENW程序前，需要通过GET_LUN函数获得文件单元号。通过IDL程序WRITEU函数将处理后的分块数据写入文件。在所有分块数据都写入后，文件被关闭，文件单元号通过IDL程序FREE_LUN释放。

一旦文件被写入硬盘，可以使用ENVI函数ENVI_SETUP_HEAD进行ENVI头文件的写入。下列文件信息必须写入头文件：文件名，采样数，行数，波段数，偏移，存储方式，以及数据类型。此外还有一些可选的关键字。如X、Y的起始位置，文本描述，波段名称等等。

spat_disk.pro



用户函数

- 对于内存输出，结果存储在内存中分配的数组中。处理后的数据块插入合适的存储位置。内存数组的大小为 $NS*NL*NB$ ，IDL函数BYTARR，INTARR，LONARR，FLTARR，DBLARR，以及MAKE_ARRAY用来创建相对应的比特类型、整型、长整型、浮点、双精度浮点以及任意类型的内存数组。
- 当处理结果完成后，包含处理结果的内存数组可以使用ENVI_ENTER_DATA传递给ENVI。在最简单的情况下，仅仅内存数组是必须的。同样有一些额外的信息可以提供，如XY的起始位置以及文字描述和波段名称。



用户函数

■ 非分块处理程序

■ ENVI_GET_DATA

该函数从文件中获取数据，一次只能对单一波段操作，范围由DIMS关键字指定。该函数提供了xfactor和yfactor两个参数能够产生放大和缩小的影像。

■ ENVI_GET_SLICE

该函数从文件中获取波谱数据，以BIP或BIL的格式返回。



用户函数

■ 使用处理进度报告

处理进度报告显示了当前处理的完成程度。使用ENVI提供的处理进度报告，开发人员只需控制增量大小和更新频率。可选的关键字Cancel用来在下次增量更新时终止处理进程。处理进度报告由三个程序控制，分别为初始化、设置增量、更新状态。这些函数列在下面：

ENVI_REPORT_INC

设置报告的增量

ENVI_REPORT_INIT

初始化报告对话框

ENVI_REPORT_STAT

更新完成的百分数并检查用

户是否执行了Cancel

注：只有在处理进程小于100%时，用户才可以取消处理，如果处理进度已达到100%，Cancel将被忽略。



用户函数

■ 用户函数编译

- 由于IDL编译器不能识别ENVI库函数，因此用户程序在编译的时候通常会报错。
- 同时为了向下兼容，IDL编译器将 () 作为数组的定义，当IDL编译器不能识别函数时，它会将 () 当作是数组定义，从而导致编译错误。
- FORWARD_FUNCTION可以告诉编译器，哪些变量是函数，而非数组定义。
- COMPILE_OPT id12 强制编译器以 [] 作为数组的定义。



用户函数

RESOLVE_ALL

- 在IDL程序中，用到许多IDL内置的函数，都是以源码的形式提供的。在IDL编译器中，它们被自动编译。但是在ENVI中，ENVI不能编译这些函数，因此要想将用户函数打包，必须要找到所有依赖的函数，而IDL提供了一个工具函数就是RESOLVE_ALL，该函数可以自动寻找和编译用户程序所依赖的所有函数。
- 在使用RESOLVE_ALL函数时要注意，它也不能识别ENVI库函数，在遇到ENVI库函数时会报错，因此在使用时，必须加上CONTINUE_ON_ERROR关键字。

