Subject: Re: ENVI's codes
Posted by Michael Galloy on Sat, 21 Nov 2009 23:23:31 GMT
View Forum Message <> Reply to Message

Hassan wrote:
> I wonder if we can have an access to source code of ENVI functions?
> I need to export some images from ENVI to IDL, I can use the export-
> varaible-to-idl function in ENVI but I want to know how I can do this
> without using ENVI.

No, ENVI is not open source.

Do you just want to be able to read some ENVI files in IDL? That is
fairly straight-forward; the data file is just a binary file which the
associated header file describes.

Mike
--
www.michaelgalloy.com
Research Mathematician
Tech-X Corporation

Subject: Re: ENVI's codes
Posted by rogass on Sat, 21 Nov 2009 23:26:55 GMT
View Forum Message <> Reply to Message

On 22 Nov., 00:23, Michael Galloy <mgal...@gmail.com> wrote:
> Hassan wrote:
>> I wonder if we can have an access to source code of ENVI functions?
>> I need to export some images from ENVI to IDL, I can use the export-
>> varaible-to-idl function in ENVI but I want to know how I can do this
>> without using ENVI.
>
> No, ENVI is not open source.
>
> Do you just want to be able to read some ENVI files in IDL? That is
> fairly straight-forward; the data file is just a binary file which the
> associated header file describes.
>
> Mike
> --www.michaelgalloy.com
> Research Mathematician
> Tech-X Corporation

You can also try to export your envi-data as geotiff which can be
imported to idl by read_tiff with the geotiff flag. Give it a try. If
this won't work for you, just think about what Michael said...

Regards

CR

Subject: Re: ENVI's codes
Posted by Hassan on Sun, 22 Nov 2009 18:41:05 GMT
View Forum Message <> Reply to Message

I used the following code to read an ENVI file into IDL:

```
imagesize=[488,290,62]
refImage = READ_BINARY(file, DATA_DIMS = imageSize)
DEVICE, DECOMPOSED = 0
LOADCT, 38
WINDOW, 0, XSIZE = imageSize[0], YSIZE = imageSize[1]
TV, refImage, 0
```

but the resulting display was strange. I don't know how to put an image here since it's easier to explain. anyway, the resulting display seems to show a superimposed of the features. for example if in the original image there are 2 features which are a bit far from each other but when I want to display it in the IDL those two features are overlaid.

Subject: Re: ENVI's codes
Posted by Hassan on Sun, 22 Nov 2009 21:35:07 GMT
View Forum Message <> Reply to Message

I used the following code to display the image:

```
image=read_tiff(file)
imagesize=[62,488,290]
DEVICE, DECOMPOSED = 0
LOADCT, 38
WINDOW, 0, XSIZE = imageSize[1], YSIZE = imageSize[2]
TV, Image(0,*,*), 0
```

but there are two problems: first it's displayed upside-down and second the way it shows the image is quite different with other softwares like ENVI, it's more like the image is shown in 256-color table or something like that.

Subject: Re: ENVI's codes
Posted by Jiek on Mon, 23 Nov 2009 00:18:14 GMT
View Forum Message <> Reply to Message

On Nov 23, 5:35 am, Hassan <hkhav...@gmail.com> wrote:
> I used the following code to display the image:
> image=read_tiff(file)
> imagesize=[62,488,290]
> DEVICE, DECOMPOSED = 0
> LOADCT, 38
> WINDOW, 0, XSIZE = imageSize[1], YSIZE = imageSize[2]
> TV, Image(0,*,*), 0
>
> but there are two problems: first it's displayed upside-down and
> second the way it shows the image is quite different with other
> softwares like ENVI, it's more like the image is shown in 256-color
> table or something like that.

I found a function to read ENVI image and it work well.As follows:
pro read_envi_image, infile, img, xs, ys, type, offset, mapinfo

;
; Copyright (c) 2003,Institute of Photogrammetry and Remote Sensing, ;
(IPF),
; Technical University of Vienna. Unauthorised reproduction
prohibited.
;
;+
; NAME:
; read_envi_file
;
; PURPOSE:
; IDL program, which reads standard ENVI image files (*.img).
;
;
; CATEGORY:
; Input_Output
;
; CALLING SEQUENCE:
; read_envi_file, infile, img, xs, ys, type,offset
;
; INPUTS:
; infile - input file name
;
; OPTIONAL INPUTS:
; None
;
; KEYWORD PARAMETERS:
; None

```
;
; OUTPUTS:
; img - ENVI image file, 2D array
; xs - number of samples
; ys - number of lines
; type - image data type
; offset - headeroffset
; mapinfo - information on spatial resolution (spacing) and
coordinates
; of upper left corner (ulx, uly)
;
;
; OPTIONAL OUTPUTS:
; None
;
; COMMON BLOCKS:
; none
;
; SIDE EFFECTS:
;
; RESTRICTIONS:
; None
;
; PROCEDURE:
;
; EXAMPLE:
;
; REMARKS
; None
;
; MODIFICATION HISTORY:
; Written by: Carsten Pathe, cp@ipf.tuwien.ac.at
; Date: 25.08.2003
;
;-

image = infile

header = strsplit(infile,'.',/extract)
header = header(n_elements(header)-2)+'.hdr'

openr, unit, header, /get_lun

header_line = ''

while not eof(unit) do begin

readf, unit, header_line
```

```
tmp = strsplit(header_line(0), '=', /extract)
header_keyword = strsplit(tmp(0), ' ', /extract)

print, header_keyword

if header_keyword(0) eq 'samples' then xs = long(tmp(1))
if header_keyword(0) eq 'lines' then ys = long(tmp(1))
if header_keyword(0) eq 'header' then offset = long(tmp(1))
if header_keyword(0) eq 'data' then type = long(tmp(1))

if header_keyword(0) eq 'map' then begin

mapinfo_tmp=strsplit(tmp(1),'{',/extract)
mapinfo_tmp=strsplit(mapinfo_tmp(1),',',/extract)

mapinfo={ulx:0.,uly:0.,spacing:0.}
mapinfo.ulx=mapinfo_tmp(3)
mapinfo.uly=mapinfo_tmp(4)
mapinfo.spacing=mapinfo_tmp(5)

endif

endwhile

close,unit & free_lun, unit
print, xs, ys

if type eq 1 then img=bytarr(xs, ys)
if type eq 2 then img=intarr(xs, ys)
if type eq 4 then img=fltarr(xs, ys)
if type eq 12 then img=uintarr(xs, ys)

openr, unit,image, /get_lun
point_lun, unit, offset
readu, unit, img
close, unit & free_lun, unit

end
```

Subject: Re: ENVI's codes
Posted by jeffnettles4870 on Mon, 23 Nov 2009 00:27:19 GMT
View Forum Message <> Reply to Message

On Nov 22, 4:35 pm, Hassan <hkhav...@gmail.com> wrote:
> I used the following code to display the image:
> image=read_tiff(file)
> imagesize=[62,488,290]

> DEVICE, DECOMPOSED = 0
> LOADCT, 38
> WINDOW, 0, XSIZE = imageSize[1], YSIZE = imageSize[2]
> TV, Image(0,*,*), 0
>
> but there are two problems: first it's displayed upside-down and
> second the way it shows the image is quite different with other
> softwares like ENVI, it's more like the image is shown in 256-color
> table or something like that.

The upside-down part I can help you with at least. With ENVI the pixel origin is in the upper left corner of the image. With IDL the origin is the lower left corner. So add a reverse and you should be ok:

image = reverse(image,2)

put that aright after you call read_tiff.

As for how its displaying, i'm not sure i understand your description of the problem, so i'll leave that to smarter folks than me :)

Subject: Re: ENVI's codes
Posted by Carsten Lechte on Mon, 23 Nov 2009 11:27:00 GMT
View Forum Message <> Reply to Message

Hassan wrote:
> but there are two problems: first it's displayed upside-down and
> second the way it shows the image is quite different with other
> softwares like ENVI, it's more like the image is shown in 256-color
> table or something like that.

The first can be fixed by using the /ORDER keyword to TV.

The second is how TV is supposed to work. Maybe the tvscale procedure from http://www.dfanning.com/documents/programs.html can help (I do not use it myself).

chl

Subject: Re: ENVI's codes
Posted by Hassan on Mon, 23 Nov 2009 13:20:55 GMT
View Forum Message <> Reply to Message

On Nov 23, 12:18 am, Jiek <676215...@qq.com> wrote:
> On Nov 23, 5:35 am, Hassan <hkhav...@gmail.com> wrote:
>
>> I used the following code to display the image:
>> image=read_tiff(file)
>> imagesize=[62,488,290]
>> DEVICE, DECOMPOSED = 0
>> LOADCT, 38
>> WINDOW, 0, XSIZE = imageSize[1], YSIZE = imageSize[2]
>> TV, Image(0,*,*), 0
>
>> but there are two problems: first it's displayed upside-down and
>> second the way it shows the image is quite different with other
>> softwares like ENVI, it's more like the image is shown in 256-color
>> table or something like that.
>
> I found a function to read ENVI image and it work well.As follows:
> pro read_envi_image, infile, img, xs, ys, type, offset, mapinfo
>
> ;
> ; Copyright (c) 2003,Institute of Photogrammetry and Remote Sensing, ;
> (IPF),
> ; Technical University of Vienna. Unauthorised reproduction
> prohibited.
> ;
> ;+
> ; NAME:
> ; read_envi_file
> ;
> ; PURPOSE:
> ; IDL program, which reads standard ENVI image files (*.img).
> ;
> ;
> ; CATEGORY:
> ; Input_Output
> ;
> ; CALLING SEQUENCE:
> ; read_envi_file, infile, img, xs, ys, type,offset
> ;
> ; INPUTS:
> ; infile - input file name
> ;
> ; OPTIONAL INPUTS:
> ; None
> ;
> ; KEYWORD PARAMETERS:
> ; None
> ;

```
>  ; OUTPUTS:
>  ; img - ENVI image file, 2D array
>  ; xs - number of samples
>  ; ys - number of lines
>  ; type - image data type
>  ; offset - headeroffset
>  ; mapinfo - information on spatial resolution (spacing) and
>  coordinates
>  ; of upper left corner (ulx, uly)
>  ;
>  ;
>  ; OPTIONAL OUTPUTS:
>  ; None
>  ;
>  ; COMMON BLOCKS:
>  ; none
>  ;
>  ; SIDE EFFECTS:
>  ;
>  ; RESTRICTIONS:
>  ; None
>  ;
>  ; PROCEDURE:
>  ;
>  ; EXAMPLE:
>  ;
>  ; REMARKS
>  ; None
>  ;
>  ; MODIFICATION HISTORY:
>  ; Written by: Carsten Pathe, c...@ipf.tuwien.ac.at
>  ; Date: 25.08.2003
>  ;
>  ;-
>
>  image = infile
>
>  header = strsplit(infile,'.',/extract)
>  header = header(n_elements(header)-2)+'.hdr'
>
>  openr, unit, header, /get_lun
>
>  header_line = ''
>
>  while not eof(unit) do begin
>
>  readf, unit, header_line
>  tmp = strsplit(header_line(0), '=', /extract)
```

```
>  header_keyword = strsplit(tmp(0), ' ', /extract)
>
>  print, header_keyword
>
>  if header_keyword(0) eq 'samples' then xs = long(tmp(1))
>  if header_keyword(0) eq 'lines' then ys = long(tmp(1))
>  if header_keyword(0) eq 'header' then offset = long(tmp(1))
>  if header_keyword(0) eq 'data' then type = long(tmp(1))
>
>  if header_keyword(0) eq 'map' then begin
>
>  mapinfo_tmp=strsplit(tmp(1),'{',/extract)
>  mapinfo_tmp=strsplit(mapinfo_tmp(1),',',/extract)
>
>  mapinfo={ulx:0.,uly:0.,spacing:0.}
>  mapinfo.ulx=mapinfo_tmp(3)
>  mapinfo.uly=mapinfo_tmp(4)
>  mapinfo.spacing=mapinfo_tmp(5)
>
>  endif
>
>  endwhile
>
>  close,unit & free_lun, unit
>  print, xs, ys
>
>  if type eq 1 then img=bytarr(xs, ys)
>  if type eq 2 then img=intarr(xs, ys)
>  if type eq 4 then img=fltarr(xs, ys)
>  if type eq 12 then img=uintarr(xs, ys)
>
>  openr, unit,image, /get_lun
>  point_lun, unit, offset
>  readu, unit, img
>  close, unit & free_lun, unit
>
>  end
```

I run it but it seems it imports just one band, right? is the output variable named 'img'?

Subject: Re: ENVI's codes
Posted by jeanh on Mon, 23 Nov 2009 15:20:36 GMT
View Forum Message <> Reply to Message

Hassan wrote:
> I used the following code to read an ENVI file into IDL:

>
> refImage = READ_BINARY(file, DATA_DIMS = imageSize)
>
> [...]anyway, the resulting display
> seems to show a superimposed of the features.

Hi,

this is a typical data type problem.
By default, read_binary reads byte data. If you have floats or else,
this superimposition would occur (though I always witness it the other
way around, when saving a float declared as byte in IDL and opening the
image in Envi). See the DATA_TYPE keywork for read_binary

Jean

Subject: Re: ENVI's codes
Posted by Hassan on Wed, 25 Nov 2009 11:55:04 GMT
View Forum Message <> Reply to Message

On Nov 23, 12:18 am, Jiek <676215...@qq.com> wrote:
> On Nov 23, 5:35 am, Hassan <hkhav...@gmail.com> wrote:
>
>> I used the following code to display the image:
>> image=read_tiff(file)
>> imagesize=[62,488,290]
>> DEVICE, DECOMPOSED = 0
>> LOADCT, 38
>> WINDOW, 0, XSIZE = imageSize[1], YSIZE = imageSize[2]
>> TV, Image(0,*,*), 0
>
>> but there are two problems: first it's displayed upside-down and
>> second the way it shows the image is quite different with other
>> softwares like ENVI, it's more like the image is shown in 256-color
>> table or something like that.
>
> I found a function to read ENVI image and it work well.As follows:
> pro read_envi_image, infile, img, xs, ys, type, offset, mapinfo
>
> ;
> ; Copyright (c) 2003,Institute of Photogrammetry and Remote Sensing, ;
> (IPF),
> ; Technical University of Vienna. Unauthorised reproduction
> prohibited.
> ;
> ;+
> ; NAME:

```
>  ; read_envi_file
>  ;
>  ; PURPOSE:
>  ; IDL program, which reads standard ENVI image files (*.img).
>  ;
>  ;
>  ; CATEGORY:
>  ; Input_Output
>  ;
>  ; CALLING SEQUENCE:
>  ; read_envi_file, infile, img, xs, ys, type,offset
>  ;
>  ; INPUTS:
>  ; infile - input file name
>  ;
>  ; OPTIONAL INPUTS:
>  ; None
>  ;
>  ; KEYWORD PARAMETERS:
>  ; None
>  ;
>  ; OUTPUTS:
>  ; img - ENVI image file, 2D array
>  ; xs - number of samples
>  ; ys - number of lines
>  ; type - image data type
>  ; offset - headeroffset
>  ; mapinfo - information on spatial resolution (spacing) and
>  coordinates
>  ; of upper left corner (ulx, uly)
>  ;
>  ;
>  ; OPTIONAL OUTPUTS:
>  ; None
>  ;
>  ; COMMON BLOCKS:
>  ; none
>  ;
>  ; SIDE EFFECTS:
>  ;
>  ; RESTRICTIONS:
>  ; None
>  ;
>  ; PROCEDURE:
>  ;
>  ; EXAMPLE:
>  ;
>  ; REMARKS
```

```
>  ; None
>  ;
>  ; MODIFICATION HISTORY:
>  ; Written by: Carsten Pathe, c...@ipf.tuwien.ac.at
>  ; Date: 25.08.2003
>  ;
>  ;-
>
>  image = infile
>
>  header = strsplit(infile,'.',/extract)
>  header = header(n_elements(header)-2)+'.hdr'
>
>  openr, unit, header, /get_lun
>
>  header_line = ''
>
>  while not eof(unit) do begin
>
>  readf, unit, header_line
>  tmp = strsplit(header_line(0), '=', /extract)
>  header_keyword = strsplit(tmp(0), ' ', /extract)
>
>  print, header_keyword
>
>  if header_keyword(0) eq 'samples' then xs = long(tmp(1))
>  if header_keyword(0) eq 'lines' then ys = long(tmp(1))
>  if header_keyword(0) eq 'header' then offset = long(tmp(1))
>  if header_keyword(0) eq 'data' then type = long(tmp(1))
>
>  if header_keyword(0) eq 'map' then begin
>
>  mapinfo_tmp=strsplit(tmp(1),'{',/extract)
>  mapinfo_tmp=strsplit(mapinfo_tmp(1),',',/extract)
>
>  mapinfo={ulx:0.,uly:0.,spacing:0.}
>  mapinfo.ulx=mapinfo_tmp(3)
>  mapinfo.uly=mapinfo_tmp(4)
>  mapinfo.spacing=mapinfo_tmp(5)
>
>  endif
>
>  endwhile
>
>  close,unit & free_lun, unit
>  print, xs, ys
>
>  if type eq 1 then img=bytarr(xs, ys)
```

> if type eq 2 then img=intarr(xs, ys)
> if type eq 4 then img=fltarr(xs, ys)
> if type eq 12 then img=uintarr(xs, ys)
>
> openr, unit,image, /get_lun
> point_lun, unit, offset
> readu, unit, img
> close, unit & free_lun, unit
>
> end

it's really useful code, thanks for that. although after running the code, it doesnt ask me for input file and it should be defined into the code but it should be straightforward to do that.
