
Subject: Converting an ENVI image into a file that can be read by IDL

Posted by [procomp9](#) on Fri, 26 Jan 2007 21:58:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi group,

Has anyone ever heard of conerting an ENVI image file into a file that can be read by IDL? The file extention that I have is *.dat and *.dat.hdr. Can ENVI do this? If not, is there a third-party program that will do this? If none of these options are available, I will write a program that parses the *.dat and *.dat.hdr contents and returns a file that can be read by IDL. To the experts of ENVI and IDL out there: Do you think parsing the data will be possible? What problems can you foresee? I am aware that the semantics of ENVI and IDL are slightly different but the fact that ENVI uses the IDL platform leads me to believe that this can be done. Thanks in advance...

Erik

Subject: Re: Converting an ENVI image into a file that can be read by IDL

Posted by [Carsten Pathe](#) on Tue, 30 Jan 2007 11:26:58 GMT

[View Forum Message](#) <> [Reply to Message](#)

procomp9 wrote:

> Hi group,
>
> Has anyone ever heard of conerting an ENVI image file into a file that
> can be read by IDL? The file extention that I have is *.dat and
> *.dat.hdr. Can ENVI do this? If not, is there a third-party program
> that will do this? If none of these options are available, I will
> write a program that parses the *.dat and *.dat.hdr contents and
> returns a file that can be read by IDL. To the experts of ENVI and IDL
> out there: Do you think parsing the data will be possible? What
> problems can you foresee? I am aware that the semantics of ENVI and IDL
> are slightly different but the fact that ENVI uses the IDL platform
> leads me to believe that this can be done. Thanks in advance...
>
>
> Erik
>

Hi Erik,

here is a little program we are using for reading standard ENVI image files (somename.img + somename.hdr). The program first analyses the image header (somename.hdr) to create a 2d array. Then the image data is

read in the 2d array (img), which you then can use within IDL.

```
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 - header offset
; 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
```
