
Subject: Making MPEG Movies

Posted by [kkk](#) on Mon, 07 Jun 1999 07:00:00 GMT

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

Hi everybody.

It would be nice if somebody could help me on the following issue: I'm

the MPEG_ functions, but once I've added all the frames and closed the file,

Could anybody help? A detailed list of the individual steps to follow when creating an MPEG file would be very useful.

Thanks in advance.

Subject: Re: Making MPEG Movies

Posted by [rivers](#) on Wed, 09 Jun 1999 07:00:00 GMT

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

In article <7jg9nd\$gr3\$1@almendralejo.unex.es>, "kkk" <kkk@kkk.kk> writes:

> Hi everybody.

> It would be nice if somebody could help me on the following issue: I'm

> the MPEG_ functions, but once I've added all the frames and closed the file,

> Could anybody help? A detailed list of the individual steps to follow when

> creating an MPEG file would be very useful.

Here is my MAKE_MOVIE.PRO. It will either play a 3-D array as a series of images either on your screen (the default) or to an MPEG file.

```
pro make_movie, vol, min=min, max=max, wait=wait, scale=scale, index=index, $
    start=start, stop=stop, step=step, mpeg_file=mpeg_file
```

```
;+
; NAME:
;   MAKE_MOVIE
;
; PURPOSE:
;   This procedure plays a 3-D array as a movie either on the screen or to an
;   MPEG file on disk.
;
; CATEGORY:
;   Image display.
;
; CALLING SEQUENCE:
;   MAKE_MOVIE, Volume
;
; INPUTS:
```

```

; Volume: A 3-D array of any numeric data type
;
;
; KEYWORD PARAMETERS:
; MIN:
;     The minimum value of the data display range. This sets the low end
;     of the grey scale range. All values less than this will be appear
;     black when using a linear grey scale. The default is the minimum value
;     of the data in the entire Volume array.
;
; MAX:
;     The maximum value of the data display range. This sets the high end
;     of the grey scale range. All values greater than this will be appear
;     white when using a linear grey scale. The default is the maximum value
;     of the data in the entire Volume array.
;
; SCALE:
;     A scale factor to increase or decrease the size of the displayed images
;     relative to the size of the Volume array. SCALE=2 will double the
;     size of the displayed images, SCALE=3 will display the images at 3
;     times their normal size, etc. SCALE=-2 will display the images at
;     half their normal size, SCALE=-3 one-third their normal size, etc.
;     The default is 1, i.e. no scaling.
;
; INDEX:
;     This keyword controls which dimension of the Volume array is animated.
;     Allowed values are 1, 2 and 3. The following table shows the result
;     of setting this keyword:
;
;     INDEX  frame[i]
;     1  Volume[i,*,*]
;     2  Volume[*,i,*]
;     3  Volume[*,*,i]
;     The default is 3, i.e. the last dimension of the Volume array is the
;     one which is animated.
;
; START:
;     The first frame to display. The default is 0.
;
; STOP:
;     The last frame to display. The default is the highest value of the
;     selected index.
;
; STEP:
;     The array increment from one frame to the next. The default is 1.
;
; WAIT:
;     The delay time from one frame to the next in floating point seconds.
;     The default is 0., i.e. frames are displayed as fast as possible.

```

```

; This keyword has no effect if the MPEG_FILE keyword is used.
;
; MPEG_FILE:
; The name of an MPEG file to which the output should be written. If
; this keyword is used then this procedure does not display its output on
; the screen but rather creates an MPEG file.
;
; OUTPUTS:
; This procedure does not return any output to IDL. It displays images on
; the screen using the IDL TV procedure, or writes output to a disk file
; if MPEG_FILE is used.
;
; SIDE EFFECTS:
; Displays images on the screen using the IDL TV procedure, or writes output
; to a disk file if MPEG_FILE is used.
;
; PROCEDURE:
; This procedure converts each frame to an 8-bit array using the IDL BYTSCL
; function and then either displays it on the screen using the TV procedure
; or writes it to an MPEG file.
;
; EXAMPLE:
; ; Display an array as a movie scanning through the first dimension of the
; ; array, forcing the black level to be 1000, and waiting 0.2 seconds
; ; between frames
; MAKE_MOVIE, Volume, index=1, wait=.2, min=1000
;
; ; Create an MPEG movie of an array scanning through the last dimension
; ; (default) of the array, forcing the white level to be 10000.
; MAKE_MOVIE, Volume, max=10000, mpeg_file='movie1.mpeg'
;
; MODIFICATION HISTORY:
; Written by: Mark Rivers, April 26, 1999. Merged previous routines
; TOMO_MOVIE and WRITE_MPEG
;-

```

```

if (n_elements(min) eq 0) then min=min(vol)
if (n_elements(max) eq 0) then max=max(vol)
if (n_elements(wait) eq 0) then wait=0
if (n_elements(scale) eq 0) then scale=1
if (n_elements(index) eq 0) then index=3

```

```

nx = n_elements(vol[*,0,0])
ny = n_elements(vol[0,*,0])
nz = n_elements(vol[0,0,*])
case index of
  1: begin
    ncols=ny

```

```

    nrows=nz
    nframes=nx
end
2: begin
    ncols=nx
    nrows=nz
    nframes=ny
end
3: begin
    ncols=nx
    nrows=ny
    nframes=nz
end
else: message, 'INDEX must be in the range 1-3'
endcase

if (scale gt 1) then begin
    last_col = ncols - 1
    last_row = nrows - 1
    ncols = ncols * fix(scale)
    nrows = nrows * fix(scale)
endif
if (scale lt -1) then begin
    iscale = fix(abs(scale))
    last_col = (ncols/iscale)*iscale - 1
    last_row = (nrows/iscale)*iscale - 1
    ncols = ncols / iscale
    nrows = nrows / iscale
endif

if (n_elements(start) eq 0) then start=0
if (n_elements(stop) eq 0) then stop=nframes
if (n_elements(step) eq 0) then step=1

if (n_elements(mpeg_file) ne 0) then begin
    mpegid = mpeg_open([ncols, nrows], file=mpeg_file)
    mpeg_mode = 1
endif else begin
    mpeg_mode = 0
endelse

for i=start, stop-1, step do begin
    case index of
        1: temp=vol[i,*,*]
        2: temp=vol[* ,i,*]
        3: temp=vol[* ,*,i]
    endcase
    temp = reform(temp, /overwrite)

```

```
temp = bytscl(temp, min=min, max=max)
if (scale ne 1) then temp = rebin(temp[0:last_col, 0:last_row], $
                                ncols, nrows)
if (mpeg_mode) then begin
    print, 'Frame = ', i
    mpeg_put, mpegid, frame=i, image=temp, order=!order
endif else begin
    tv, temp
    wait, wait
endelse
endfor

if (mpeg_mode) then begin
    mpeg_save, mpegid, file=file
    mpeg_close, mpegid
endif

end
```
