Subject: Line index
Posted by Hermann Mannstein on Fri, 15 Mar 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Hello,
is there a robust function which returns the indices within an image
(like the where function)
belonging to a line defined by two or more points. A short and dirty, but slow
solution, which is not applicable in batch procedures would be:

```
function LINE_INDEX1,ix,iy,points,count
o_device = !D.NAME
set_plot,'x'
window,1,/pixmap,xsize=ix,ysize=iy,colors=256
plots,points(*,0),points(*,1),/dev,/noclip
im = tvrd()
i = where(im,count)
set_plot,o_device
return,i
end
```

but I need something faster and I'm shure that somebody has already written it.
--
Regards,

```
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~
 Hermann Mannstein            Tel.: +49 8153 28-2503
 Institut fuer Physik der Atmosphaere    or -2558
 DLR - Oberpfaffenhofen             Fax.: +49 8153 28-1841
 Postfach 1116         \        mailto:H.Mannstein@dlr.de
 D-82230 Wessling         \ 0    http://www.op.dlr.de/~pa64
 Germany    ________\/|________
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~`--------\--------'~ ~~~~~~~~~~~~~~~~~~~~~~
                                          \
```

Subject: Re: Line index
Posted by Robert Moss on Mon, 25 Mar 1996 08:00:00 GMT
View Forum Message <> Reply to Message

I may have missed your initial question, but if your looking for the
array indices along a line in an image, the following snippet of
code seems to work fine. The two endpoints are (x,y) and (x1,y1), and
the image is in the variable pimage.

```
dx = float(x1-x)  ;delta x
dy = float(y1-y)  ;delta y
```

```
n = sqrt( dx^2 + dy^2 )  ;length
sy =(y1-y)/n
sx =(x1-x)/n
;
xx = lindgen(n+1)*sx+x  ;X indices,  make into longwords.
yy = lindgen(n+1)*sy+y  ;Y indices
sz = size( pimage )  ; image size
if !order ne 0 then yy=sz(2)-1-yy ;reverse y indices?
ans=pimage(xx,yy)  ; image data along the line
```

You may want to go some fiddling with the (n+1) bit depending on how you want to handle fractional pixels.

By the way, this is what I use when I want a "profile" of an image, i.e I want the image data values along the line connecting two points. The length reported here is equal to the length of the line (within pixel resolution, your milage may vary).

Note that this is different than the IDL supplied library funtction called "profile" which actually returns a list of values that are the length of the x or y projection of the line connecting two points, depending on which is longer. I did not find the builtin "profile" results acceptable for my purposes.

Robet M. Moss, Ph.D.
Texaco Inc.
mossrm@texaco.com

This may not reflect the views of Texaco Inc.

David Foster wrote:
>
> pit@asterix.kis.uni-freiburg.de (Peter Suetterlin) wrote:
>>
>> In article <31499FB9.2F6D@dlr.de>,
>>   Hermann Mannstein <H.Mannstein@dlr.de> writes:
>>
>>> is there a robust function which returns the indices within an image
>>> (like the where function) belonging to a line defined by two or more
>>> points.
>>
>> for shure there is, even built-in IDL:
>> Index=polyfillv(xx,yy,xsize,ysize)

>> Read the manpage for further reference.
>>
>
> According to the "manpage", POLLYFILLV() expects at least 3
> points so it would never work for a simple line. Also, it
> connects each successive line and the first with the last,
> to make a polygon, and then returns the indices of the
> points within this polygon.
>
> You wouldn't be getting the indices of the points on the line(s),
> but of the points within the polygon defined by the points.
>
> Dave Foster
> foster@bial1.ucsd.edu

Subject: Re: Line index
Posted by thompson on Wed, 27 Mar 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Hermann Mannstein <H.Mannstein@dlr.de> wrote:
>
> Hello,
> is there a robust function which returns the indices within an image
> (like the where function)
> belonging to a line defined by two or more points.

The following routine works for me. I've edited out the reference to
GET_IM_KEYWORD, and included INTERP2 below, so you should be able to use it
just as it is.

Bill Thompson

```
 ==============================================================
===================
 FUNCTION PROF,ARRAY,XVAL,YVAL,MISSING=MISSING
;+
; Project     : SOHO - CDS
;
; Name        :
; PROF()
; Purpose     :
; Returns profiles from arrays along the path XVAL, YVAL.
; Explanation :
; After the arrays XVAL and YVAL are converted, the routine INTERP2 is
; called to do the interpolation.
; Use         :
; P = PROF( ARRAY, XVAL, YVAL )
; Inputs      :
; ARRAY   = Image to take profile from.
; XVAL,YVAL = The X,Y coordinates of points defining the path.
; Opt. Inputs :
; None.
; Outputs     :
; Function value = Values of ARRAY along profile.
; XVAL,YVAL = The X,Y coordinates of the resulting path.  The
;     original points are converted to a set with points
;     set one pixel apart along the path.
; Opt. Outputs:
; None.
; Keywords    :
; MISSING  = Value flagging missing pixels.
; Calls       :
; GET_IM_KEYWORD, INTERP2
; Common      :
; None.
; Restrictions:
; ARRAY must be two-dimensional.
;
; In general, the SERTS image display routines use several non-standard
; system variables.  These system variables are defined in the procedure
; IMAGELIB.  It is suggested that the command IMAGELIB be placed in the
; user's IDL_STARTUP file.
;
; Some routines also require the SERTS graphics devices software,
; generally found in a parallel directory at the site where this software
; was obtained.  Those routines have their own special system variables.
;
; Side effects:
; The arrays XVAL and YVAL are changed.
; Category    :
; Utilities, Image_display.
; Prev. Hist. :
; W.T.T., Oct. 1987.
; W.T.T., Jan. 1991.  Changed FLAG to keyword BADPIXEL.
; William Thompson, August 1992, renamed BADPIXEL to MISSING.
; Written     :
; William Thompson, GSFC, October 1987.
; Modified    :
; Version 1, William Thompson, GSFC, 12 May 1993.
;  Incorporated into CDS library.
; Version     :
; Version 1, 12 May 1993.
;-
;
; GET_IM_KEYWORD, MISSING, !IMAGE.MISSING
```

```
;
;  Check the number of parameters.
;
 IF N_PARAMS(0) NE 3 THEN BEGIN
  PRINT,'*** PROF must be called with three parameters:'
  PRINT,'             ARRAY, XVAL, YVAL'
  RETURN,0
 ENDIF
;
;  Check the input variable ARRAY.
;
 S = SIZE(ARRAY)
 IF S(0) NE 2 THEN BEGIN
  PRINT,'*** Variable must be two-dimensional, name= ARRAY, routine PROF.'
  RETURN,0
 ENDIF
;
;  Save the input parameters XVAL and YVAL in temporary arrays X and Y.
;
 X = XVAL
 Y = YVAL
;
;  Find the total length of the path.
;
 LENGTH = 0.
 FOR IP = 1,N_ELEMENTS(X) - 1 DO BEGIN
  D_LENGTH = SQRT( (X(IP) - X(IP-1))^2  +  (Y(IP) - Y(IP-1))^2 )
  LENGTH = LENGTH + D_LENGTH
 ENDFOR
;
;  Find the first and subsequent interpolation points.
;
 NI = FIX(LENGTH)
 PROFILE = FLTARR(NI>1)
 XVAL = FLTARR(NI>1)
 YVAL = FLTARR(NI>1)
 XVAL(0) = X(0)
 YVAL(0) = Y(0)
 IB = 0
 S_LENGTH = 0
 IF NI GT 1 THEN FOR INT = 1,NI-1 DO BEGIN
;
;  Find the proper range for each point.  First try to fit the next point
;  into the range IB,IB+1.  The variable S_LENGTH is the length represented
;  by the segments 0,1 through IB-1,IB (if IB = 0, then S_LENGTH = 0).  Then
;  the length DIST is the distance from the point IB, and FRACTION is the
;  relative position within the range IB,IB+1.  If FRACTION is greater than
;  one, increase IB by one and try again.
```

```
;
  TRY: DIST = INT - S_LENGTH
   D_LENGTH = SQRT( (X(IB+1) - X(IB))^2  +  $
    (Y(IB+1) - Y(IB))^2 )
   IF D_LENGTH EQ 0 THEN FRACTION = 1000. ELSE $
    FRACTION = DIST / D_LENGTH
   IF FRACTION GT 1 THEN BEGIN
    IB = IB + 1
    S_LENGTH = S_LENGTH + D_LENGTH
    GOTO,TRY
   ENDIF
  XVAL(INT) = (1 - FRACTION)*X(IB) + FRACTION*X(IB+1)
  YVAL(INT) = (1 - FRACTION)*Y(IB) + FRACTION*Y(IB+1)
 ENDFOR
;
 RETURN,INTERP2(ARRAY,XVAL,YVAL,MISSING=MISSING)
 END
 ============================================================
===================
 FUNCTION INTERP2,IMAGE,X,Y,MISSING=MISSING
;+
; Project     : SOHO - CDS
;
; Name        :
; INTERP2()
; Purpose     :
; Performs a two-dimensional interpolation on IMAGE.
; Explanation :
; An average is made between the four nearest neighbors of the point to
; be interpolated to.
; Use         :
; OUTPUT = INTERP2( IMAGE, X, Y )
; Inputs      :
; IMAGE = Image to be interpolated.
; X = X coordinate position(s) of the interpolated point(s).
; Y = Y coordinate position(s) of the interpolated point(s).
; Opt. Inputs :
; None.
; Outputs     :
; The function returns a one-dimensional array of the interpolated
; points.
; Opt. Outputs:
; None.
; Keywords    :
; MISSING  = Value flagging missing pixels.  Any such pixels are not
;     included in the interpolation.  If any interpolation point
;     is surrounded only by missing pixels, then the output value
;     for that point is set to MISSING.
; Calls       :
; GET_IM_KEYWORD
; Common      :
; None.
; Restrictions:
; IMAGE must be two-dimensional.
;
; In general, the SERTS image display routines use several non-standard
; system variables.  These system variables are defined in the procedure
; IMAGELIB.  It is suggested that the command IMAGELIB be placed in the
; user's IDL_STARTUP file.
;
; Some routines also require the SERTS graphics devices software,
; generally found in a parallel directory at the site where this software
; was obtained.  Those routines have their own special system variables.
;
; Side effects:
; None.
; Category    :
; Utilities, Image_display.
; Prev. Hist. :
; W.T.T., Oct. 1987.
; W.T.T., Jan. 1991.  Changed FLAG to keyword BADPIXEL.
; William Thompson, August 1992, renamed BADPIXEL to MISSING.
; William Thompson, 5 May 1993, fixed bug when Y > first dim. of IMAGE.
; Written     :
; William Thompson, October 1987.
; Modified    :
; Version 1, William Thompson, GSFC, 13 May 1993.
;  Incorporated into CDS library.
; Version     :
; Version 1, 13 May 1993.
;-
;
; GET_IM_KEYWORD, MISSING, !IMAGE.MISSING
;
;  Check the number of parameters.
;
 IF N_PARAMS(0) NE 3 THEN BEGIN
  PRINT,'*** INTERP2 must be called with three parameters:'
  PRINT,'                  IMAGE, X, Y'
  RETURN,0
 ENDIF
;
;  Check the size of the array IMAGE.
;
 S = SIZE(IMAGE)
 IF S(0) NE 2 THEN BEGIN
```

```
  PRINT,'*** Variable must be two-dimensional, name= IMAGE, routine INTERP2.'
  RETURN,0
 ENDIF
;
;  Find the boundaries of the square containing the point X,Y to interpolate
;  to.
;
 NX = S(1) - 1
 NY = S(2) - 1
 IX1 = 0 > FIX(X) < NX
 IY1 = 0 > FIX(Y) < NY
 IX2 = IX1 + 1 < NX
 IY2 = IY1 + 1 < NY
 DX = 0 > (X - IX1) < 1
 DY = 0 > (Y - IY1) < 1
;
;  Initialize the arrays (or scalers) INT and W_TOTAL.
;
 INT = 0. * (X + Y)
 W_TOTAL = 0. * (X + Y)
;
;  Start adding together the contributions from each corner of the box
;  containing the point X,Y.  Ignore any corners that have the value MISSING.
;
 POS = IX1 + S(1)*IY1
 WEIGHT = (1. - DX) * (1. - DY)
 IF N_ELEMENTS(MISSING) EQ 1 THEN $
  WEIGHT = WEIGHT * (IMAGE(POS) NE MISSING)
 INT = INT + IMAGE(POS)*WEIGHT
 W_TOTAL = W_TOTAL + WEIGHT
;
 POS = IX1 + S(1)*IY2
 WEIGHT = (1. - DX) * DY
 IF N_ELEMENTS(MISSING) EQ 1 THEN $
  WEIGHT = WEIGHT * (IMAGE(POS) NE MISSING)
 INT = INT + IMAGE(POS)*WEIGHT
 W_TOTAL = W_TOTAL + WEIGHT
;
 POS = IX2 + S(1)*IY1
 WEIGHT = DX * (1. - DY)
 IF N_ELEMENTS(MISSING) EQ 1 THEN $
  WEIGHT = WEIGHT * (IMAGE(POS) NE MISSING)
 INT = INT + IMAGE(POS)*WEIGHT
 W_TOTAL = W_TOTAL + WEIGHT
;
 POS = IX2 + S(1)*IY2
 WEIGHT = DX * DY
 IF N_ELEMENTS(MISSING) EQ 1 THEN $
```

```
  WEIGHT = WEIGHT * (IMAGE(POS) NE MISSING)
 INT = INT + IMAGE(POS)*WEIGHT
 W_TOTAL = W_TOTAL + WEIGHT
;
;  Check the size of W_TOTAL.
;
 W_SIZE = SIZE(W_TOTAL)
;
;  If W_TOTAL is an array, then use the following procedure.  Set any points
;  that cannot be interpolated to the value MISSING.
;
 IF W_SIZE(0) NE 0 THEN BEGIN
  IF N_ELEMENTS(MISSING) NE 1 THEN BEGIN
   POS = WHERE(W_TOTAL NE 0,N_FOUND)
   IF N_FOUND GT 0 THEN INT(POS) = INT(POS) / W_TOTAL(POS)
   POS = WHERE(W_TOTAL EQ 0,N_FOUND)
   IF N_FOUND GT 0 THEN INT(POS) = MISSING
  END ELSE BEGIN
   INT = INT / W_TOTAL
  ENDELSE
;
;  If W_TOTAL is a scaler, then use the following procedure.  Again, if the
;  point cannot be interpolated, return the value MISSING.
;
 END ELSE IF (W_TOTAL NE 0) THEN BEGIN
  INT = INT / W_TOTAL
 END ELSE BEGIN
  INT = MISSING
 ENDELSE
;
 RETURN,INT
 END
```