Subject: Re: Polygon Clipping Algo in IDL
Posted by Mark Hadfield on Tue, 13 Dec 2005 01:38:42 GMT
View Forum Message <> Reply to Message

raval.chintan@gmail.com wrote:
> Dear All,
>
> Does any one has written code for the polygon clipping (Cohen
> Sutherland or Liang Baskey) in IDL? or is there any direct function
> available in IDL?
>
> Regards
> Chintan
>
Attached, based on code from JD Smith. These are part of my Motley
library and may well require other routines in same. See

http://www.dfanning.com/hadfield/idl/README.html

--
Mark Hadfield "Kei puwaha te tai nei, Hoea tahi tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

```
;+
; NAME:
;   MGH_POLYCLIP
;
; PURPOSE:
;   Clip an arbitrary polygon on the X-Y plane to a line parallel
;   to the X or Y axis using the Sutherland-Hodgman algorithm.
;
; CATEGORY:
;   Graphics, Region of Interest, Geometry
;
; CALLING SEQUENCE:
;   result = MGH_POLYCLIP(clip, dir, neg, polin, COUNT=count)
;
; RETURN VALUE
;   The function returns the clipped polygon as a [2,n] array. The
;   second dimension will equal the value of the COUNT argument, except
;   where COUNT is 0 in which case the return value is -1.
;
; POSITIONAL PARAMETERS
;   cval (input, numeric sclar)
;     The value of X or Y at which clipping is to occur
;
```

```
;   dir (input, integer scalar)
;     Specifies whether clipping value is an X (dir = 0) or Y (dir =
;     1) value.
;
;   neg (input, integer scalar)
;     Set this argument to 1 to clip to the negtive side, 0 to clip to
;     the positive side.
;
;   polin (input, floating array)
;     A [2,m] vector defining the polygon to be clipped.
;
; KEYWORD PARAMETERS
;   COUNT (output, integer)
;     The number of vertices in the clipped polygon.
;
; PROCEDURE:
;   The polygon is clipped using the Sutherland-Hodgman algorithm.
;
;   This function is based on JD Smith's implementation of the
;   Sutherland-Hodgman algorithm in his POLYCLIP function. He can
;   take all of the credit and none of the blame.
;
 ;############################################################## ##################
;
; This software is provided subject to the following conditions:
;
; 1.  NIWA makes no representations or warranties regarding the
;     accuracy of the software, the use to which the software may
;     be put or the results to be obtained from the use of the
;     software.  Accordingly NIWA accepts no liability for any loss
;     or damage (whether direct of indirect) incurred by any person
;     through the use of or reliance on the software.
;
; 2.  NIWA is to be acknowledged as the original author of the
;     software where the software is used or presented in any form.
;
 ;############################################################## ##################
;
; MODIFICATION HISTORY:
;   Mark Hadfield, 2001-10:
;     Written, based on JD Smith's POLYClIP.
;-

function mgh_polyclip, cval, dir, neg, poly, COUNT=count

  compile_opt DEFINT32
  compile_opt STRICTARR
  compile_opt STRICTARRSUBS
```

```
compile_opt LOGICAL_PREDICATE

if n_elements(poly) eq 0 then $
     message, BLOCK='mgh_mblk_motley', NAME='mgh_m_undefvar', 'poly'

;; If the polygon argument is a scalar then return a scalar to
;; to indicate that the polygon has no vertices.

count = 0

if size(poly, /N_DIMENSIONS) eq 0 then return, -1

;; Vector "in" specifies whether each vertex is inside
;; the clipped half-plane

case dir of
  0B: in = neg ? reform(poly[0,*] lt cval) : reform(poly[0,*] gt cval)
  else: in = neg ? reform(poly[1,*] lt cval) : reform(poly[1,*] gt cval)
endcase

;; Calculate number of points in polygon--it is a little
;; more efficient to get it from the size of "in" than
;; from the dimensions of "poly"

np = n_elements(in)

;; Vector "inx" specifies whether an intersection with the clipping line
;; is made by the segment joining each vertex with the one before.

inx = in xor shift(in, 1)

;; Precalculate an array of shifted vertices, used in calculating
;; intersection points in the loop.

pols = shift(poly, 0, 1)

;; Loop thru vertices

for k=0,np-1 do begin

  ;; If this segment crosses the clipping line, add the intersection
  ;; to the output list. I tried calculating the intersection points
  ;; outside the loop in an array operation but it turned out slower.

  if inx[k] then begin
    s0 = pols[0,k]
    s1 = pols[1,k]
    p0 = poly[0,k]
```

```
        p1 = poly[1,k]
        case dir of
          0B: ci = [cval,s1+(p1-s1)/(p0-s0)*(cval-s0)]
          else: ci = [s0+(p0-s0)/(p1-s1)*(cval-s1),cval]
        endcase
        polout = count eq 0 ? [ci] : [[polout],[ci]]
        count ++
      endif

      ;; If this vertex is inside the clipped half-plane add it to the list

      if in[k] then begin
        polout = count eq 0 ? [poly[*,k]] : [[polout],[poly[*,k]]]
        count ++
      endif

    endfor

    return, count gt 0 ? polout : -1

end

;+
; NAME:
;   MGH_POLYCLIP2
;
; PURPOSE:
;   Clip an arbitrary polygon on the X-Y plane to a line of arbitrary
;   orientation using the Sutherland-Hodgman algorithm.
;
; CATEGORY:
;   Graphics, Region of Interest, Geometry
;
; CALLING SEQUENCE:
;   result = MGH_POLYCLIP2(clip, poly, COUNT=count)
;
; POSITIONAL PARAMETERS
;   poly (input, floating array)
;     A [2,m] vector defining the polygon to be clipped.
;
;   clip (input, 3-element numeric vector)
;     This parameter describes the line to be clipped to. The polygon is
;     clipped to the half-plane (clip[0] x + clip[1] y + clip[2]) < 0.
;
; KEYWORD PARAMETERS
;   COUNT (output, integer)
;     The number of vertices in the clipped polygon.
;
```

```
; RETURN VALUE
;   The function returns the clipped polygon as a [2,n] array. The
;   second dimension will equal the value of the COUNT argument, except
;   where COUNT is 0 in which case the return value is -1.
;
; PROCEDURE:
;   The polygon is clipped using the Sutherland-Hodgman algorithm.
;
;   This function is similar to MGH_POLYCLIP, which was written first
;   & clips to vertical or horizontal lines only. It turns out that
;   MGH_POLYCLIP2 is competitive with MGH_POLYCLIP in terms of speed
;   so the former may supersede the latter.
;
;   Both polygon-clipping functions are based on JD Smith's
;   implementation of the Sutherland-Hodgman algorithm in his POLYCLIP
;   function. He can take most of the credit and none of the blame.
;
 ;############################################################## ################
;
; This software is provided subject to the following conditions:
;
; 1.  NIWA makes no representations or warranties regarding the
;     accuracy of the software, the use to which the software may
;     be put or the results to be obtained from the use of the
;     software.  Accordingly NIWA accepts no liability for any loss
;     or damage (whether direct of indirect) incurred by any person
;     through the use of or reliance on the software.
;
; 2.  NIWA is to be acknowledged as the original author of the
;     software where the software is used or presented in any form.
;
 ;############################################################## ################
;
; MODIFICATION HISTORY:
;   Mark Hadfield, 2002-08:
;     Written.
;-

function mgh_polyclip2, poly, clip, COUNT=count

   compile_opt DEFINT32
   compile_opt STRICTARR
   compile_opt STRICTARRSUBS
   compile_opt LOGICAL_PREDICATE

   if size(poly, /N_ELEMENTS) eq 0 then $
        message, BLOCK='mgh_mblk_motley', NAME='mgh_m_undefvar', 'poly'
```

```
if size(clip, /N_ELEMENTS) ne 3 then $
     message, BLOCK='mgh_mblk_motley', NAME='mgh_m_wrgnumelem', 'clip'

;; If the polygon argument is a scalar then return a scalar to
;; to indicate that the polygon has no vertices.

count = 0

if size(poly, /N_DIMENSIONS) eq 0 then return, -1

;; Precalculate an array of shifted vertices, used in calculating
;; intersection points in the loop.

pols = shift(poly, 0, 1)

;; Vector "pp" is the position of each vertex along the
;; perpendicular to the clipping line. Vector "ps" is the same for
;; the shifted vertices

pp = reform(clip[0]*poly[0,*]+clip[1]*poly[1,*]+clip[2])
ps = shift(pp, 1)

;; Vector "in" specifies whether each vertex is inside the clipped
;; half-plane.  Vector "inx" specifies whether an intersection with
;; the clipping line is made by the segment joining each vertex
;; with the one before.

in = pp lt 0
inx = in xor (ps lt 0)

;; Loop thru vertices

np = n_elements(in)

for k=0,np-1 do begin

   ;; If this segment crosses the clipping line, add the
   ;; intersection to the output list.

   if inx[k] then begin
      ap = ps[k]/(ps[k]-pp[k])
      ci = ap*poly[*,k] + (1-ap)*pols[*,k]
      polout = count eq 0 ? [ci] : [[polout],[ci]]
      count ++
   endif

   ;; If this vertex is inside the clipped half-plane add it to the
   ;; list
```

```
    if in[k] then begin
      polout = count eq 0 ? [poly[*,k]] : [[polout],[poly[*,k]]]
      count ++
    endif

  endfor

  return, count gt 0 ? polout : -1

end
```

File Attachments

1) mgh_polyclip.pro, downloaded 148 times
2) mgh_polyclip2.pro, downloaded 147 times