Subject: Re: clip polyhedron mesh
Posted by Dick Jackson on Thu, 21 May 2015 07:09:22 GMT
View Forum Message <> Reply to Message

guni wrote on 2015-05-20 2:48pm:
> On Wednesday, 20 May 2015 20:39:19 UTC+2, Dick Jackson wrote:
>> Hi Guni,
>>
>> On Tuesday, 19 May 2015 10:51:48 UTC-7, guni wrote:
>>> Dear all, I have a 3-dimensional polyhedron mesh where two polyhedrons
>>> are overlapped. I want to clip the polyhedron to make new polyhedrons
>>> where one portion belong the overlapping region and other non-overlapping
>>> region. If somebody knows how to do this, please let me know.
>>
>> First, it's a much simpler problem if you know you're working with *convex*
>> polyhedra.
>>
>> A Google search on [intersection of convex polyhedra algorithm] shows that
>> at least *somebody* knows how to do this. :-) For example: "Finding the
>> intersection of two convex polyhedra" from 1977:
>> http://www.sciencedirect.com/science/article/pii/03043975789 00518
>>
>> There are lengthy algorithms that might take a lot of work to implement.
>> Some even give solutions for intersecting convex and non-convex polyhedra.
> Hi Dick
>
> On Wednesday, 20 May 2015 20:39:19 UTC+2, Dick Jackson wrote:
>> Hi Guni,
>>
>> On Tuesday, 19 May 2015 10:51:48 UTC-7, guni wrote:
>>> Dear all, I have a 3-dimensional polyhedron mesh where two polyhedrons
>>> are overlapped. I want to clip the polyhedron to make new polyhedrons
>>> where one portion belong the overlapping region and other non-overlapping
>>> region. If somebody knows how to do this, please let me know.
>>
>> First, it's a much simpler problem if you know you're working with *convex*
>> polyhedra.
>>
>> [...]
>>
>>> 2nd option:I saw IDL's 'mesh_clip' but it is a clip using a planar
>>> surface. I dont prefer to clip using a plane, but in case if I have to
>>> use it, how I can get the coordinates of the overlapping portion?
>>
>> Something like this came up some time ago, and it may be the easiest way to
>> go (assuming convex polyhedra). If you use each polygon from mesh 1 as a
>> clipping plane into mesh 2 (and keep the correct piece each time!), when
>> you're done, you'll be left with the intersection. This link includes

>> another link to a useful example:
>> https://groups.google.com/forum/#!searchin/comp.lang.idl-pvwave/intersection$20polyhedron/comp.lang.idl-pvwave/qAvnBjaws oY/JaiOeUS3KpoJ
>>
>> [...]
>>
>> I hope this helps!
>>
>> Cheers, -Dick
>>
>> Dick Jackson Software Consulting Inc. Victoria, BC, Canada ---
>> http://www.d-jackson.com
>>
>> P.S.: This was a nice example you gave:
>>
>>> Anyway my script/polyhedron is something like this. Dick helped me to
>>> create polyhedrons, but here I used iplot, and ipolygon.
>>>
>>> ;;1st polyhedron x=randomu(seed,4) y=randomu(seed,4) z=randomu(seed,4)
>>> xyz=[transpose(x),transpose(y),transpose(z)]
>>> iPLOT,xyz,LINESTYLE=6,AXIS_STYLE=2,identifier='1' QHULL,xyz,Vert
>>> conn=[REPLICATE(3,[1,N_ElEMENTS(Vert)/3]),Vert]
>>> iPOLYGON,xyz,/DATA,CONNECTIVITY=conn,visualization='1',transparency=50,/FILL_BACKGROUND,FILL_COLOR='SKY
>>> BLUE'
>>>
>>> ;;2nd polyhedron x=randomu(seed,12) y=randomu(seed,12)
>>> z=randomu(seed,12) xyz=[transpose(x),transpose(y),transpose(z)]
>>> iPLOT,xyz,LINESTYLE=6,/OVERPLOT,identifier='2' QHULL,xyz,Vert
>>> conn=[REPLICATE(3,[1,N_ElEMENTS(Vert)/3]),Vert]
>>> iPOLYGON,xyz,/DATA,CONNECTIVITY=conn,visualization='2',transparency=50,/FILL_BACKGROUND,FILL_COLOR='red'
>>>
>>>
>>>
Thanks,
>>> Guni
>
> Hi Dick, Thanks a lot for your help. Well, I would like to see how MESH_CLIP
> works in my convex polyhedrons. I looked the link, and also the example. But,
> I dont know how to derive the plane coefficients in my polyhedron mesh. In
> the example it is defined as [1., 1., 1., 0.]. How can I derive these plane
> coefficients? "Plane--Input four element array describing the equation of the
> plane to be clipped to. The elements are the coefficients (a,b,c,d) of the
> equation ax+by+cz+d=0." When I placed the mouse pointer in the plot (mesh),
> it shows x,y,z co-ordinates, Is it something related to the coefficients I am
> looking? Thanks Guni
>

Right, that's not trivial. I had found the magic at
  http://paulbourke.net/geometry/pointlineplane/
... and a function implementing this (and more) is below.

For each triangle in the mesh, you can use this routine to get the coefficients:

```
  abcd = PlaneCoeffs([[x0,y0,z0],[x1,y1,z1],[x2,y2,z2]])
```

You should double-check that the resulting value results in the correct side of the plane being used. If it's wrong, then do one of these:
- send points in reordered as [[x0,y0,z0],[x2,y2,z2],[x1,y1,z1]], or
- use -(abcd) instead of abcd

Let me know if that works out!

Cheers,
-Dick

Dick Jackson Software Consulting Inc. -- www.d-jackson.com

```
;-----
; PlaneCoeffs
;+
; :Description:
;    From the input data provided in one of several forms, return the
;    coefficients that define the plane.
;
; :Returns:
;    Floating-point vector [a,b,c,d] defining the plane *ax + by + cz + d = 0*
;
; :Keywords:
;    Points : in, optional, type=numeric 1-D or 2-D array
;       Either:
;         - One point [x,y,z] on the desired plane, requiring one
;           of XYangle, XZangle, YZangle or NormalVector to be provided, or:
;         - Three points [[x0,y0,z0],[x1,y1,z1],[x2,y2,z2]] that
;           fully define the plane.
;    XYangle : in, optional, type=numeric scalar
;       For a plane parallel to the Z axis, the angle between the Y=0 line and
;       the desired plane (in degrees, counter-clockwise)
;    XZangle : in, optional, type=numeric scalar
;       For a plane parallel to the Y axis, the angle between the Z=0 line and
;       the desired plane (in degrees, counter-clockwise)
;    YZangle : in, optional, type=numeric scalar
;       For a plane parallel to the X axis, the angle between the Y=0 line and
;       the desired plane (in degrees, counter-clockwise)
;    NormalVector : in, optional, type=numeric 1-D array
;       A 3-element vector (x,y,z) describing the normal to the desired plane
;
; :Examples:
;    Find the equation of the plane that passes through the points
;       [1,1,1], [-1,1,0], [2,0,3]
;    IDL> Print, PlaneCoeffs(Points=[[1,1,1], [-1,1,0], [2,0,3]])
;          -1       3       2      -4
;          (or, -x + 3y - 2z - 4 = 0)
;    IDL> Print, PlaneCoeffs(Points=[0,0,0], YZAngle=30)
;         0.000000     0.500000    -0.866025    -0.000000
;    IDL> Print, PlaneCoeffs(Points=[1,2,3], NormalVector=[4,5,6])
;           4       5       6     -32
;
; :Author:
;    Dick Jackson Software Consulting Inc. -- www.d-jackson.com
;
; :History:
; 2009-10-02 djackson
;    First revision, partly from former PlaneFrom3Points.pro
; 2009-10-05 djackson
;    New: Keyword NormalVector
;    Doc: Improved comments, changed to RST format
; 2015-05-20 djackson
;    Doc: Improved docs
;-
```

```
FUNCTION PlaneCoeffs, Points=points, XYangle=xyAngle, XZangle=xzAngle, $
   YZangle=yzAngle, NormalVector=normalVector

COMPILE_OPT IDL2 ; Integers default to 32-bit and indexing requires use of []

CASE 1B OF

   ;;    Three points

   Array_Equal(Size(points, /Dimensions), [3, 3]) AND $
      (N_Elements(xyAngle)+N_Elements(xzAngle)+N_Elements(yzAngle) ) EQ 0: $
     BEGIN

     ;;    Method from http://paulbourke.net/geometry/pointlineplane/

     p1 = points[*, 0] & p2 = points[*, 1] & p3 = points[*, 2]
     x1 = p1[0] & y1 = p1[1] & z1 = p1[2]
     x2 = p2[0] & y2 = p2[1] & z2 = p2[2]
     x3 = p3[0] & y3 = p3[1] & z3 = p3[2]

     a = y1*(z2 - z3) + y2*(z3 - z1) + y3*(z1 - z2)
     b = z1*(x2 - x3) + z2*(x3 - x1) + z3*(x1 - x2)
     c = x1*(y2 - y3) + x2*(y3 - y1) + x3*(y1 - y2)
     d = -( x1*(y2*z3 - y3*z2) + x2*(y3*z1 - y1*z3) + x3*(y1*z2 - y2*z1) )

   END ;; Three points case

   ;;    One point and an angle (one of XYangle, XZangle or YZangle)

   N_Elements(points) EQ 3 AND N_Elements(xyAngle) EQ 1: BEGIN
     a = Cos((xyAngle-90) * !DtoR)
     b = Sin((xyAngle-90) * !DtoR)
     c = 0
     d = -(a*points[0]+b*points[1])
```

```
  END ;; One point and XYangle

  N_Elements(points) EQ 3 AND N_Elements(xzAngle) EQ 1: BEGIN
    a = Cos((xzAngle-90) * !DtoR)
    c = Sin((xzAngle-90) * !DtoR)
    b = 0
    d = -(a*points[0]+c*points[2])
  END ;; One point and XZangle

  N_Elements(points) EQ 3 AND N_Elements(yzAngle) EQ 1: BEGIN
    b = Cos((yzAngle-90) * !DtoR)
    c = Sin((yzAngle-90) * !DtoR)
    a = 0
    d = -(b*points[1]+c*points[2])
  END ;; One point and YZangle

  N_Elements(points) EQ 3 AND N_Elements(normalVector) EQ 3: BEGIN
    a = normalVector[0]
    b = normalVector[1]
    c = normalVector[2]
    d = -(a*points[0]+b*points[1]+c*points[2])
  END ;; One point and NormalVector

ENDCASE ;; of different input options

Return, [a, b, c, d]

END

;-----
```

--

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada
www.d-jackson.com