Subject: point inside/outside of 3D object.
Posted by Junum on Tue, 14 Jun 2011 19:01:25 GMT
View Forum Message <> Reply to Message

I want to determine whether a given point is inside or outside of 3D object.
There have been some suggestions on this discussion group, so I have tried IDLanROI::ContainsPoints.
It works for 2D case, but not for 3D.
Assuming a tetrahedron,
its coordinates for 4 points are
(1,0,0)
(0,1,0)
(-1,0,0)
(0,0,1)

px = [1., 0., -1., 0.]
py = [0.,1.,0.,0.]
pz = [0.,0.,0.,1.]

object = Obj_New('IDLanROI', px, py, pz)
print,object->containspoints(0.1,0.1,0.1)

It prints 0, Exterior, although (0.1,0.1,0.1) is inside of tetrahedron.

My questions are
1. I guess that a method defining a tetrahedron is wrong (i.e., px, py, pz).
How can I define a 3D object consisting of several plane surfaces (e.g., cube) in IDLanROI?

2. How can I draw a tetrahedron in 3D graphic?

Could you help me?
Thank you.

Sincerely,
Jun

Subject: Re: point inside/outside of 3D object.
Posted by Karl[1] on Mon, 20 Jun 2011 19:57:25 GMT
View Forum Message <> Reply to Message

On Jun 20, 7:49 am, Wox <s...@nomail.com> wrote:
> On Fri, 17 Jun 2011 12:47:36 -0700 (PDT), Karl

>
> <karl.w.schu...@gmail.com> wrote:
>> I'm not sure you'd want to draw a tet with an ROI. A tet can be drawn
>> with a grPolygon. You would supply the 4 verts and then the
>> connectivity list which would be something like:
>
>> [3,0,1,2, 3,1,0,3, 3,2,1,3, 3,0,2,3]
>
>> The order is important to make all the faces facing "out". If any of
>> these are wrong, reverse the order. E.g., if the last tri is facing
>> the wrong way, change 3,0,2,3 to 3,3,2,0.
>
> Could you elaborate on that?
>
> I understand that the normal vector on each trianglular face of the
> polyhedron should point outwards. So if you take the three vertices of
> a triangle, they should be ordered so that when using the "right-hand
> rule", the normal points outwards.
>
> Lets mark the vertices of a tetrahedron Red, Green, Blue and Gray
> (http://tinypic.com/r/513fau/7). So the ordered vertices for each
> triangle should be
> back: Red-Green-Blue (equivalents: Green-Blue-Red, Blue-Red-Green)
> front-right: Red-Gray-Green
> front-left: Red-Blue-Gray
> bottom: Green-Blue-Gray
>
> So how do the quadruples come into play?

The first "3" is the number of indices that follow that contribute to
the next face. I believe that is how the connectivity list is defined
in IDLgrPolygon. Please see the docs for IDLgrPolygon.

If you put your vertices into an array v in this order: Red, Green,
Blue, Gray

Then your connectivity list that you'd pass to IDLgrPolygon along with
the vertex array v would be:

3,0,1,2, 3,0,3,1, 3,0,2,3, 3,1,2,3

Although I think you want Green-Gray-Blue for the bottom. You want
the normal for the bottom to point down, giving:

3,0,1,2, 3,0,3,1, 3,0,2,3, 3,1,3,2

which is equivalent to the first list I gave:

>> [3,0,1,2, 3,1,0,3, 3,2,1,3, 3,0,2,3]

taking different starting positions into account.

Subject: Re: point inside/outside of 3D object.
Posted by Wout De Nolf on Tue, 21 Jun 2011 08:52:07 GMT
View Forum Message <> Reply to Message

On Mon, 20 Jun 2011 12:57:25 -0700 (PDT), Karl
<karl.w.schultz@gmail.com> wrote:

> The first "3" is the number of indices that follow that contribute to
> the next face.

Ah, I learned something new today. Thanks!

Subject: Re: point inside/outside of 3D object.
Posted by Wout De Nolf on Tue, 21 Jun 2011 09:24:01 GMT
View Forum Message <> Reply to Message

On Sat, 18 Jun 2011 11:34:42 -0700 (PDT), Junum <junshikum@gmail.com>
wrote:

> Thanks Karl.
> I wanted know whether IDLanROI::ContainsPoints can be applied to 3D
> case.

I'd guess the answer is no. You should implement this yourself (as
Karl suggested) or you could do something like below. I'm not sure
whether this is the best way, but it seems to work.

```
; Generate vertices
v=TetrahedronVertices(r=10,phideg=-20)

; Connectivity list: [n,i[0],...,i[n-1],n,j[0],...,j[n-1],...]
;   n: number of vertices for each face
;   i[0],...,i[n-1]: vertices for face 1, ordered so that the normal
;                    points outwards (right-hand rule)
;   j[0],...,j[n-1]: vertices for face 2, ordered so that the normal
;                    points outwards (right-hand rule)
conn=[3,0,3,1, 3,0,1,2, 3,0,2,3, 3,1,3,2]

; Remark: if the number of vertices > 4 then you could generate
; the list like this:
```

```
;Qhull, v, tr, /delaunay
;conn=tetra_surface(v, tr)

; Point
p=[0,0,0.]

; Volume of the polyhedron
volume=tetra_volume(v,conn)

; Expanded polyhedron (including your point)
; vertices and connectivity list
v2=[[v],[p]]
Qhull, v2, tr, /delaunay
conn2=tetra_surface(v2, tr)

; Volume of the expanded polyhedron
volumeexp=tetra_volume(v2,conn2)

; If the "expanded volume" is larger, the point lies outside
if volumeexp gt volume then print,'Exterior' else print,'Interior'
```

Subject: Re: point inside/outside of 3D object.
Posted by Karl[1] on Tue, 21 Jun 2011 15:18:02 GMT

View Forum Message <> Reply to Message

On Jun 21, 3:24 am, Wox <s...@nomail.com> wrote:
> On Sat, 18 Jun 2011 11:34:42 -0700 (PDT), Junum <junshi...@gmail.com>
> wrote:
>
>> Thanks Karl.
>> I wanted know whether IDLanROI::ContainsPoints can be applied to 3D
>> case.
>
> I'd guess the answer is no. You should implement this yourself (as
> Karl suggested) or you could do something like below. I'm not sure
> whether this is the best way, but it seems to work.
>
> ; Generate vertices
> v=TetrahedronVertices(r=10,phideg=-20)
>
> ; Connectivity list: [n,i[0],...,i[n-1],n,j[0],...,j[n-1],...]
> ; n: number of vertices for each face
> ; i[0],...,i[n-1]: vertices for face 1, ordered so that the normal
> ;   points outwards (right-hand rule)
> ; j[0],...,j[n-1]: vertices for face 2, ordered so that the normal
> ;   points outwards (right-hand rule)
> conn=[3,0,3,1, 3,0,1,2, 3,0,2,3, 3,1,3,2]

>
> ; Remark: if the number of vertices > 4 then you could generate
> ; the list like this:
> ;Qhull, v, tr, /delaunay
> ;conn=tetra_surface(v, tr)
>
> ; Point
> p=[0,0,0.]
>
> ; Volume of the polyhedron
> volume=tetra_volume(v,conn)
>
> ; Expanded polyhedron (including your point)
> ; vertices and connectivity list
> v2=[[v],[p]]
> Qhull, v2, tr, /delaunay
> conn2=tetra_surface(v2, tr)
>
> ; Volume of the expanded polyhedron
> volumeexp=tetra_volume(v2,conn2)
>
> ; If the "expanded volume" is larger, the point lies outside
> if volumeexp gt volume then print,'Exterior' else print,'Interior'

>> I wanted know whether IDLanROI::ContainsPoints can be applied to 3D
>> case.
>
> I'd guess the answer is no.

Right. Although not explicitly stated, I think that most of the anROI support is for 2D ROI's even though you can specify points in 3D space. The ROI's were intended for image analysis, I think. I understand that ROI's can be 1D and 3D as well, but I think that the algorithms in anROI are intended for 2D. Note that there are methods to compute such things as area and perimeter, but not volume and surface area. I also recall that some of the anROI algorithms will go ahead and try to do things like compute areas for non-planar ROI's even though the result may be questionable. Other algorithms might actually go ahead and do a planarity check and throw an error if the data wasn't planar.

The expanding polyhedron approach is clever! I like it.

Subject: Re: point inside/outside of 3D object.
Posted by Junum on Tue, 21 Jun 2011 17:22:14 GMT
View Forum Message <> Reply to Message

On Jun 21, 10:18 am, Karl <karl.w.schu...@gmail.com> wrote:
> On Jun 21, 3:24 am, Wox <s...@nomail.com> wrote:
>
>
>
>
>
>> On Sat, 18 Jun 2011 11:34:42 -0700 (PDT), Junum <junshi...@gmail.com>
>> wrote:
>
>>> Thanks Karl.
>>> I wanted know whether IDLanROI::ContainsPoints can be applied to 3D
>>> case.
>
>> I'd guess the answer is no. You should implement this yourself (as
>> Karl suggested) or you could do something like below. I'm not sure
>> whether this is the best way, but it seems to work.
>
>> ; Generate vertices
>> v=TetrahedronVertices(r=10,phideg=-20)
>
>> ; Connectivity list: [n,i[0],...,i[n-1],n,j[0],...,j[n-1],...]
>> ; n: number of vertices for each face
>> ; i[0],...,i[n-1]: vertices for face 1, ordered so that the normal
>> ; points outwards (right-hand rule)
>> ; j[0],...,j[n-1]: vertices for face 2, ordered so that the normal
>> ; points outwards (right-hand rule)
>> conn=[3,0,3,1, 3,0,1,2, 3,0,2,3, 3,1,3,2]
>
>> ; Remark: if the number of vertices > 4 then you could generate
>> ; the list like this:
>> ;Qhull, v, tr, /delaunay
>> ;conn=tetra_surface(v, tr)
>
>> ; Point
>> p=[0,0,0.]
>
>> ; Volume of the polyhedron
>> volume=tetra_volume(v,conn)
>
>> ; Expanded polyhedron (including your point)
>> ; vertices and connectivity list
>> v2=[[v],[p]]
>> Qhull, v2, tr, /delaunay
>> conn2=tetra_surface(v2, tr)
>
>> ; Volume of the expanded polyhedron
>> volumeexp=tetra_volume(v2,conn2)

>
>> ; If the "expanded volume" is larger, the point lies outside
>> if volumeexp gt volume then print,'Exterior' else print,'Interior'
>>> I wanted know whether IDLanROI::ContainsPoints can be applied to 3D
>>> case.
>
>> I'd guess the answer is no.
>
> Right. Although not explicitly stated, I think that most of the anROI
> support is for 2D ROI's even though you can specify points in 3D
> space. The ROI's were intended for image analysis, I think. I
> understand that ROI's can be 1D and 3D as well, but I think that the
> algorithms in anROI are intended for 2D. Note that there are methods
> to compute such things as area and perimeter, but not volume and
> surface area. I also recall that some of the anROI algorithms will go
> ahead and try to do things like compute areas for non-planar ROI's
> even though the result may be questionable. Other algorithms might
> actually go ahead and do a planarity check and throw an error if the
> data wasn't planar.
>
> The expanding polyhedron approach is clever! I like it.

Thank you very much, Wox and Karl.

Jun

Subject: Re: point inside/outside of 3D object.
Posted by Guneshwar Thangjam on Sat, 04 Jul 2015 19:55:29 GMT
View Forum Message <> Reply to Message

On Tuesday, 21 June 2011 11:24:01 UTC+2, Wox wrote:
> On Sat, 18 Jun 2011 11:34:42 -0700 (PDT), Junum <junshikum@gmail.com>
> wrote:
>
>> Thanks Karl.
>> I wanted know whether IDLanROI::ContainsPoints can be applied to 3D
>> case.
>
> I'd guess the answer is no. You should implement this yourself (as
> Karl suggested) or you could do something like below. I'm not sure
> whether this is the best way, but it seems to work.
>
>
> ; Generate vertices
> v=TetrahedronVertices(r=10,phideg=-20)
>

```
>  ; Connectivity list: [n,i[0],...,i[n-1],n,j[0],...,j[n-1],...]
>  ;   n: number of vertices for each face
>  ;   i[0],...,i[n-1]: vertices for face 1, ordered so that the normal
>  ;                    points outwards (right-hand rule)
>  ;   j[0],...,j[n-1]: vertices for face 2, ordered so that the normal
>  ;                    points outwards (right-hand rule)
>  conn=[3,0,3,1, 3,0,1,2, 3,0,2,3, 3,1,3,2]
>
>  ; Remark: if the number of vertices > 4 then you could generate
>  ; the list like this:
>  ;Qhull, v, tr, /delaunay
>  ;conn=tetra_surface(v, tr)
>
>  ; Point
>  p=[0,0,0.]
>
>  ; Volume of the polyhedron
>  volume=tetra_volume(v,conn)
>
>  ; Expanded polyhedron (including your point)
>  ; vertices and connectivity list
>  v2=[[v],[p]]
>  Qhull, v2, tr, /delaunay
>  conn2=tetra_surface(v2, tr)
>
>  ; Volume of the expanded polyhedron
>  volumeexp=tetra_volume(v2,conn2)
>
>  ; If the "expanded volume" is larger, the point lies outside
>  if volumeexp gt volume then print,'Exterior' else print,'Interior'
```

Hi Wox,
I am using this approach to find out points inside or outside of a 3d polyhedron. It is working and I used it in some of my analysis. Can you please give me a reference or literature where I can find some more details of this?
Thanks,
Guni
