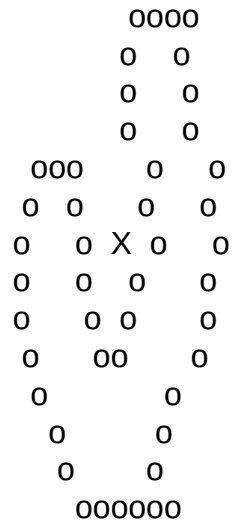
Subject: Surface mesh
Posted by twhaw on Wed, 09 Apr 2008 11:26:43 GMT
View Forum Message <> Reply to Message

Good day.

I have created a 3D model consisting of points sampled on the outline (surface) of a 3D object. The 3D object is initially translated so that the origin is at its center of mass. The points are then sampled in the spherical coordinates and converted back to the rectangular coordinated and stored as a (3, n) array. The outline of the model is sampled at a fixed interval longitudinally (every10 deg). Below is an example of the sample points taken at 0 deg and 180 deg longitude so that we can obtain the sample points around the object (similar to cutting the object into halves through its center of mass).



where o represents the sample points and X represents the origin (center of mass of the 3D object).

This process is repeated at 10 deg & -170 deg, 20 deg & -160 deg, 30 deg & -150 deg etc. Each longitude slice consists of the same number of sample points (k). Hence I have a model consisting of 18 slices, with each slice consisting of k sample points. These sample points are stored as a (3, n) array, where n = k * 18 and each row corresponds to one sample point (1st column gives its x-coord, 2nd column gives y-coord and 3rd column gives z-coord).

Now, I would like to create the surface mesh of the 3D model from the sample points. I have used MESH_OBJ (type 0) but the resulting mesh does not correspond to the surface of the 3D model. I have also triangulated the points first to get the polygons and passed them to IDLgrPolygon, but could not get the desired surface mesh.
Below are the codes that I have used:
; x, y and z are 1D vectors corresponds to the 1st, 2nd and 3rd column

of the (3, n) array defined above.
sph_coord = cv_coord(from_rect=transpose([[x], [y], [z]]), /
to_sphere, /degrees)
longiture = transpose(sph_coord[0, *])
latitute = transpose(sph_coord[1, *])
radius = transpose(sph_coord[2, *])
triangulate, longiture, latitute, tri, FVALUE=radius, /DEGREES,
SPHERE=myS
ntri = SIZE(tri,/DIMENSIONS)
ntri = ntri[1]
connect = LONARR(4,ntri)
connect[0,*] = 3
connect[1:3,*] = tri
oSurf = OBJ_NEW('IDLgrPolygon', TRANSPOSE(myS.XYZ), POLYGON=connect,
STYLE=2, $
SHADING=1, COLOR=[0,20,255])
oGroup = OBJ_NEW('IDLgrModel')
oGroup->ADD, oSurf
XOBJVIEW, oGroup, /block

These codes are adapted from one of the posts in this newsgroup and it works for spherical object. In fact, the resulting mesh resembles a sphere (which it shouldn't be).

If anyone has done this or something similar before, your help in this matter is highly appreciated.

Many thanks.

Subject: Re: Surface mesh
Posted by rtowler on Thu, 10 Apr 2008 16:53:23 GMT
View Forum Message <> Reply to Message

On Apr 9, 4:26 am, twhaw <wooihaw....@gmail.com> wrote:
> Good day.
>
> I have created a 3D model consisting of points sampled on the outline
> (surface) of a 3D object. The 3D object is initially translated so
> that the origin is at its center of mass. The points are then sampled
> in the spherical coordinates and converted back to the rectangular
> coordinated and stored as a (3, n) array. The outline of the model is
> sampled at a fixed interval longitudinally (every10 deg).

<snip>

> Now, I would like to create the surface mesh of the 3D model from the
> sample points. I have used MESH_OBJ (type 0) but the resulting mesh

> does not correspond to the surface of the 3D model. I have also
> triangulated the points first to get the polygons and passed them to
> IDLgrPolygon, but could not get the desired surface mesh.

This problem is similar to an earlier thread titled "how to compute the merged volume of two 3d objects" although you absolutely need to find the equilibrium surface since your object is concave. Unlike the poster in that thread, your task is much easier. Your vertices are ordered by the slice and you know that the verts in one slice will "connect" to the verts in the next. Easier still since the number of samples per slice is fixed. It gets even easier if your verts from each slice are sampled in the same order. Given this, you should be able to generate a polygon mesh comprised of quad strips that create bands connecting one slice to the next.

Since the planes pass thru the center of mass of your object (and I would assume the axis of rotation for your cutting planes is fixed) each slice should share 2 common vertices. Given your example object, the only trick I can see is that these common verts will be transition points where you will have to reverse the order of the polygon winding since the "top" and "bottom" row of verts will flip-flop. This wouldn't affect the wire-frame view, but if you don't change the winding the normals will be pointed inward which would affect lighting/ shading.

You'll have a couple of for loops. The outer loops nSlices-1 (17) times. The inner will loop k times. In the inner you'll fill in your polygon array, starting at the common point on slice n, and connecting it's counterclockwise neighbor, then the neighbor's compliment on slice n+1, then the commons's compliment on n+1. Repeat for each point on that slice, keeping in mind that when you get to the transition point again you'll need to change the order that you connect the verts. The last slice is then meshed to the first after your nested for loops in the same manner. For more complicated objects this gets harder but I think this should work for your example object.

You'll want to read the IDL docs on "About Polygon and Polyline Objects" and "Polygon Optimization" for more info on creating polygon arrays.

Have fun.

-Rick

Subject: Re: Surface mesh
Posted by twhaw on Fri, 11 Apr 2008 06:17:51 GMT
View Forum Message <> Reply to Message

Dear Rick,

Thanks for your very insight reply. It helps a lot.

I have managed to construct the surface mesh now, though I still have problem with the polygon's normals near the transition point. Is there any way to make sure that polygon's vertices are always in the counter-clockwise manner?

Cheers.

On Apr 11, 12:53 am, rtow...@gmail.com wrote:
> On Apr 9, 4:26 am, twhaw <wooihaw....@gmail.com> wrote:
>
>> Good day.
>
>> I have created a 3D model consisting of points sampled on the outline
>> (surface) of a 3D object. The 3D object is initially translated so
>> that the origin is at its center of mass. The points are then sampled
>> in the spherical coordinates and converted back to the rectangular
>> coordinated and stored as a (3, n) array. The outline of the model is
>> sampled at a fixed interval longitudinally (every10 deg).
>
> <snip>
>
>> Now, I would like to create the surface mesh of the 3D model from the
>> sample points. I have used MESH_OBJ (type 0) but the resulting mesh
>> does not correspond to the surface of the 3D model. I have also
>> triangulated the points first to get the polygons and passed them to
>> IDLgrPolygon, but could not get the desired surface mesh.
>
> This problem is similar to an earlier thread titled "how to compute
> the merged volume of two 3d objects" although you absolutely need to
> find the equilibrium surface since your object is concave. Unlike the
> poster in that thread, your task is much easier. Your vertices are
> ordered by the slice and you know that the verts in one slice will
> "connect" to the verts in the next. Easier still since the number of
> samples per slice is fixed. It gets even easier if your verts from
> each slice are sampled in the same order. Given this, you should be
> able to generate a polygon mesh comprised of quad strips that create
> bands connecting one slice to the next.
>
> Since the planes pass thru the center of mass of your object (and I
> would assume the axis of rotation for your cutting planes is fixed)
> each slice should share 2 common vertices. Given your example object,

> the only trick I can see is that these common verts will be transition
> points where you will have to reverse the order of the polygon winding
> since the "top" and "bottom" row of verts will flip-flop. This
> wouldn't affect the wire-frame view, but if you don't change the
> winding the normals will be pointed inward which would affect lighting/
> shading.
>
> You'll have a couple of for loops. The outer loops nSlices-1 (17)
> times. The inner will loop k times. In the inner you'll fill in your
> polygon array, starting at the common point on slice n, and connecting
> it's counterclockwise neighbor, then the neighbor's compliment on
> slice n+1, then the commons's compliment on n+1. Repeat for each
> point on that slice, keeping in mind that when you get to the
> transition point again you'll need to change the order that you
> connect the verts. The last slice is then meshed to the first after
> your nested for loops in the same manner. For more complicated
> objects this gets harder but I think this should work for your example
> object.
>
> You'll want to read the IDL docs on "About Polygon and Polyline
> Objects" and "Polygon Optimization" for more info on creating polygon
> arrays.
>
> Have fun.
>
> -Rick
