Subject: Re: Point Cloud Isosurface
Posted by Karl[1] on Thu, 12 Aug 2010 21:27:02 GMT
View Forum Message <> Reply to Message

On Aug 12, 2:42 pm, tegus <tegusbillhar...@gmail.com> wrote:
> Hello,
> I'm working with noisy 3D point cloud data approximated by:
> xyz_0=randomn(seed, 3, 1000) + 5.0
> xyz_1=randomn(seed, 3, 1000) + 10.0
> xyz_2=randomu(seed, 3, 10000) * 20.0
> xyz=[[xyz_0], [xyz_1], [xyz_2]]
>
> although the actual data sets are larger and far more complex ...
>
> My current method of reducing and rendering the data:
> - Create a 3D histogram (bin size = 1) using hist_nd from JDHU
> library:
> vol=hist_nd(xyz, 1.0)
> - Create isosurface (density threshold=10)
> isosurface, vol, 10, verts, conn
> - Create polygon object and display
> oPoly=obj_new('IDLgrPolygon', verts, polygons=conn)
> xobjview, oPoly
>
> This gives me the desired result which in this example is a polygon
> object which depicts two blobs approximating the measured positions.
>
> However, rendering and analysis of a more complex scene as a single,
> complex polygon becomes unwieldy (e.g., no dynamic culling, z
> clipping ...)
> My question is, how do I separate these two solid objects, represented
> by a single polygon (verts and conn), into two seperate polygon
> objects (verts1, conn1 and verts2, conn2)?
>
> Thanks,
> Bill

MESH_CLIP() ?

Subject: Re: Point Cloud Isosurface
Posted by tegus on Fri, 13 Aug 2010 01:55:31 GMT
View Forum Message <> Reply to Message

On Aug 12, 5:27 pm, Karl <karl.w.schu...@gmail.com> wrote:
> On Aug 12, 2:42 pm, tegus <tegusbillhar...@gmail.com> wrote:
>
>

>
>
>
>> Hello,
>> I'm working with noisy 3D point cloud data approximated by:
>> xyz_0=randomn(seed, 3, 1000) + 5.0
>> xyz_1=randomn(seed, 3, 1000) + 10.0
>> xyz_2=randomu(seed, 3, 10000) * 20.0
>> xyz=[[xyz_0], [xyz_1], [xyz_2]]
>
>> although the actual data sets are larger and far more complex ...
>
>> My current method of reducing and rendering the data:
>> - Create a 3D histogram (bin size = 1) using hist_nd from JDHU
>> library:
>> vol=hist_nd(xyz, 1.0)
>> - Create isosurface (density threshold=10)
>> isosurface, vol, 10, verts, conn
>> - Create polygon object and display
>> oPoly=obj_new('IDLgrPolygon', verts, polygons=conn)
>> xobjview, oPoly
>
>> This gives me the desired result which in this example is a polygon
>> object which depicts two blobs approximating the measured positions.
>
>> However, rendering and analysis of a more complex scene as a single,
>> complex polygon becomes unwieldy (e.g., no dynamic culling, z
>> clipping ...)
>> My question is, how do I separate these two solid objects, represented
>> by a single polygon (verts and conn), into two seperate polygon
>> objects (verts1, conn1 and verts2, conn2)?
>
>> Thanks,
>> Bill
>
> MESH_CLIP() ?

Karl,

To use MESH_CLIP I need to specify a clipping plane between the blobs,
which becomes intractable for any real data set (on the order of
millions of points and tens of thousands of blobs).

Subject: Re: Point Cloud Isosurface
Posted by Karl[1] on Fri, 13 Aug 2010 14:58:11 GMT
View Forum Message <> Reply to Message

On Aug 12, 7:55 pm, tegus <tegusbillhar...@gmail.com> wrote:
> On Aug 12, 5:27 pm, Karl <karl.w.schu...@gmail.com> wrote:
>
>
>
>> On Aug 12, 2:42 pm, tegus <tegusbillhar...@gmail.com> wrote:
>
>>> Hello,
>>> I'm working with noisy 3D point cloud data approximated by:
>>> xyz_0=randomn(seed, 3, 1000) + 5.0
>>> xyz_1=randomn(seed, 3, 1000) + 10.0
>>> xyz_2=randomu(seed, 3, 10000) * 20.0
>>> xyz=[[xyz_0], [xyz_1], [xyz_2]]
>
>>> although the actual data sets are larger and far more complex ...
>
>>> My current method of reducing and rendering the data:
>>> - Create a 3D histogram (bin size = 1) using hist_nd from JDHU
>>> library:
>>> vol=hist_nd(xyz, 1.0)
>>> - Create isosurface (density threshold=10)
>>> isosurface, vol, 10, verts, conn
>>> - Create polygon object and display
>>> oPoly=obj_new('IDLgrPolygon', verts, polygons=conn)
>>> xobjview, oPoly
>
>>> This gives me the desired result which in this example is a polygon
>>> object which depicts two blobs approximating the measured positions.
>
>>> However, rendering and analysis of a more complex scene as a single,
>>> complex polygon becomes unwieldy (e.g., no dynamic culling, z
>>> clipping ...)
>>> My question is, how do I separate these two solid objects, represented
>>> by a single polygon (verts and conn), into two seperate polygon
>>> objects (verts1, conn1 and verts2, conn2)?
>
>>> Thanks,
>>> Bill
>
>> MESH_CLIP() ?
>
> Karl,
>
> To use MESH_CLIP I need to specify a clipping plane between the blobs,
> which becomes intractable for any real data set (on the order of
> millions of points and tens of thousands of blobs).

oh ok. The conn list is a vector where each polygon is represented by

"n", followed by n vertex indices. So you could extract the first polygon (blob) by:

blob_conn1 = conn[0:conn[0]]

and then generate additional conn lists for each blob by looping through the original list.

These lists refer to the original vertex list. You can leave it that way and just reference the original vertex list from the grPolygon objects. So you would use verts (the original vert list) and blob_conn1 in a grPolygon object to represent the first blob.

If putting the entire vertex list in each polygon object is bad, as I suspect it is, you can do additional work to pull the vertices referenced by each conn list out of the original vert list and make a new, smaller, vert list for each blob. Then you can delete the original vertex list.

This will likely be a good path since there are likely no shared verts between blobs.

You'll have to be careful when pulling the verts out since you'll have to remap the conn indicies and pay attention to verts used more than once, etc. But it can all be done pretty easily.

And all this assumes that each blob is represented by one polygon in the original polygon object. If that is not the case, you'll have to do some more work to find all the polys that belong to a blob.

Subject: Re: Point Cloud Isosurface
Posted by tegus on Fri, 13 Aug 2010 17:15:31 GMT
View Forum Message <> Reply to Message

On Aug 13, 10:58 am, Karl <karl.w.schu...@gmail.com> wrote:
> On Aug 12, 7:55 pm, tegus <tegusbillhar...@gmail.com> wrote:
>
>
>
>
>
>> On Aug 12, 5:27 pm, Karl <karl.w.schu...@gmail.com> wrote:
>
>>> On Aug 12, 2:42 pm, tegus <tegusbillhar...@gmail.com> wrote:
>
>>>> Hello,
>>>> I'm working with noisy 3D point cloud data approximated by:

>>>> xyz_0=randomn(seed, 3, 1000) + 5.0
>>>> xyz_1=randomn(seed, 3, 1000) + 10.0
>>>> xyz_2=randomu(seed, 3, 10000) * 20.0
>>>> xyz=[[xyz_0], [xyz_1], [xyz_2]]
>
>>>> although the actual data sets are larger and far more complex ...
>
>>>> My current method of reducing and rendering the data:
>>>> - Create a 3D histogram (bin size = 1) using hist_nd from JDHU
>>>> library:
>>>> vol=hist_nd(xyz, 1.0)
>>>> - Create isosurface (density threshold=10)
>>>> isosurface, vol, 10, verts, conn
>>>> - Create polygon object and display
>>>> oPoly=obj_new('IDLgrPolygon', verts, polygons=conn)
>>>> xobjview, oPoly
>
>>>> This gives me the desired result which in this example is a polygon
>>>> object which depicts two blobs approximating the measured positions.
>
>>>> However, rendering and analysis of a more complex scene as a single,
>>>> complex polygon becomes unwieldy (e.g., no dynamic culling, z
>>>> clipping ...)
>>>> My question is, how do I separate these two solid objects, represented
>>>> by a single polygon (verts and conn), into two seperate polygon
>>>> objects (verts1, conn1 and verts2, conn2)?
>
>>>> Thanks,
>>>> Bill
>
>>> MESH_CLIP() ?
>
>> Karl,
>
>> To use MESH_CLIP I need to specify a clipping plane between the blobs,
>> which becomes intractable for any real data set (on the order of
>> millions of points and tens of thousands of blobs).
>
> oh ok. The conn list is a vector where each polygon is represented by
> "n", followed by n vertex indices. So you could extract the first
> polygon (blob) by:
>
> blob_conn1 = conn[0:conn[0]]
>
> and then generate additional conn lists for each blob by looping
> through the original list.
>
> These lists refer to the original vertex list. You can leave it that

> way and just reference the original vertex list from the grPolygon
> objects. So you would use verts (the original vert list) and
> blob_conn1 in a grPolygon object to represent the first blob.
>
> If putting the entire vertex list in each polygon object is bad, as I
> suspect it is, you can do additional work to pull the vertices
> referenced by each conn list out of the original vert list and make a
> new, smaller, vert list for each blob. Then you can delete the
> original vertex list.
>
> This will likely be a good path since there are likely no shared verts
> between blobs.
>
> You'll have to be careful when pulling the verts out since you'll have
> to remap the conn indicies and pay attention to verts used more than
> once, etc. But it can all be done pretty easily.
>
> And all this assumes that each blob is represented by one polygon in
> the original polygon object. If that is not the case, you'll have to
> do some more work to find all the polys that belong to a blob.

Karl,

Thank you for your insight. This helps a great deal!!

Subject: Re: Point Cloud Isosurface
Posted by Karl[1] on Fri, 13 Aug 2010 17:56:11 GMT
View Forum Message <> Reply to Message

On Aug 13, 11:15 am, tegus <tegusbillhar...@gmail.com> wrote:
> On Aug 13, 10:58 am, Karl <karl.w.schu...@gmail.com> wrote:
>
>
>
>> On Aug 12, 7:55 pm, tegus <tegusbillhar...@gmail.com> wrote:
>
>>> On Aug 12, 5:27 pm, Karl <karl.w.schu...@gmail.com> wrote:
>
>>>> On Aug 12, 2:42 pm, tegus <tegusbillhar...@gmail.com> wrote:
>
>>>> > Hello,
>>>> > I'm working with noisy 3D point cloud data approximated by:
>>>> > xyz_0=randomn(seed, 3, 1000) + 5.0
>>>> > xyz_1=randomn(seed, 3, 1000) + 10.0
>>>> > xyz_2=randomu(seed, 3, 10000) * 20.0
>>>> > xyz=[[xyz_0], [xyz_1], [xyz_2]]
>

>>>> > although the actual data sets are larger and far more complex ...
>
>>>> > My current method of reducing and rendering the data:
>>>> > - Create a 3D histogram (bin size = 1) using hist_nd from JDHU
>>>> > library:
>>>> >   vol=hist_nd(xyz, 1.0)
>>>> > - Create isosurface (density threshold=10)
>>>> >   isosurface, vol, 10, verts, conn
>>>> > - Create polygon object and display
>>>> >   oPoly=obj_new('IDLgrPolygon', verts, polygons=conn)
>>>> >   xobjview, oPoly
>
>>>> > This gives me the desired result which in this example is a polygon
>>>> > object which depicts two blobs approximating the measured positions.
>
>>>> > However, rendering and analysis of a more complex scene as a single,
>>>> > complex polygon becomes unwieldy (e.g., no dynamic culling, z
>>>> > clipping ...)
>>>> > My question is, how do I separate these two solid objects, represented
>>>> > by a single polygon (verts and conn), into two seperate polygon
>>>> > objects (verts1, conn1 and verts2, conn2)?
>
>>>> > Thanks,
>>>> > Bill
>
>>>> MESH_CLIP() ?
>
>>> Karl,
>
>>> To use MESH_CLIP I need to specify a clipping plane between the blobs,
>>> which becomes intractable for any real data set (on the order of
>>> millions of points and tens of thousands of blobs).
>
>> oh ok. The conn list is a vector where each polygon is represented by
>> "n", followed by n vertex indices. So you could extract the first
>> polygon (blob) by:
>
>> blob_conn1 = conn[0:conn[0]]
>
>> and then generate additional conn lists for each blob by looping
>> through the original list.
>
>> These lists refer to the original vertex list. You can leave it that
>> way and just reference the original vertex list from the grPolygon
>> objects. So you would use verts (the original vert list) and
>> blob_conn1 in a grPolygon object to represent the first blob.
>
>> If putting the entire vertex list in each polygon object is bad, as I

>> suspect it is, you can do additional work to pull the vertices
>> referenced by each conn list out of the original vert list and make a
>> new, smaller, vert list for each blob. Then you can delete the
>> original vertex list.
>
>> This will likely be a good path since there are likely no shared verts
>> between blobs.
>
>> You'll have to be careful when pulling the verts out since you'll have
>> to remap the conn indicies and pay attention to verts used more than
>> once, etc. But it can all be done pretty easily.
>
>> And all this assumes that each blob is represented by one polygon in
>> the original polygon object. If that is not the case, you'll have to
>> do some more work to find all the polys that belong to a blob.
>
> Karl,
>
> Thank you for your insight. This helps a great deal!!

Actually, I don't think it will help much. I just recalled that ISOSURFACE probably emits independent triangles. So all the "n"'s will be 3, and that isn't good to do what you want.

Here is one really slow approach:

All the triangles in a blob should be vertex-connected to other tris in the blob. So, you could pick and remove the first triangle from the list. Then find another triangle in the original list that shares a vert with any triangle in your current blob tri list. Remove this tri from the main list and add it to the blob list. Repeat until no more tris in the main list match any vert in the blob list. Then start a new blob by taking the first tri in the remaining main list and start the process again, repeating until the main list is empty.

The polygon returned by ISOSURFACE does not repeat verts in the vertex list, so you can just compare connectivity indices when comparing verts.

There may be some clever ways to implement this in IDL to make it run fast enough, but there will probably be a lot of looping which will be pretty slow if there are a lot of triangles.