Subject: nearest node of Delauny tesselation
Posted by Jeremy Bailin on Thu, 25 Apr 2013 21:35:40 GMT
View Forum Message <> Reply to Message

Under the category of "this must be easy, but I can't seem to figure out
the right function":

If I have created a Delauny tesselation using TRIANGULATE, how can I
easily find which nodes form the triangle that contains an arbitrary
point in the space?

(more specifically, I am using /NATURAL_NEIGHB interpolation in GRIDDATA
and it's going horribly wrong for one point, so I'm trying to figure out
what nodes it's actually using in the interpolation for that point)

Thanks,
-Jeremy.

Subject: Re: nearest node of Delauny tesselation
Posted by ben.bighair on Fri, 26 Apr 2013 20:37:10 GMT
View Forum Message <> Reply to Message

On Thursday, April 25, 2013 5:35:40 PM UTC-4, Jeremy Bailin wrote:
> Under the category of "this must be easy, but I can't seem to figure out
>
> the right function":
>
>
>
> If I have created a Delauny tesselation using TRIANGULATE, how can I
>
> easily find which nodes form the triangle that contains an arbitrary
>
> point in the space?
>
>
>
> (more specifically, I am using /NATURAL_NEIGHB interpolation in GRIDDATA
>
> and it's going horribly wrong for one point, so I'm trying to figure out
>
> what nodes it's actually using in the interpolation for that point)
>

Hi,

COuld you use IDLanROI::ContainsPoint()? You would have to convert the triangulations to ROIs

first.

Cheers,
Ben

Subject: Re: nearest node of Delauny tesselation
Posted by Jeremy Bailin on Fri, 26 Apr 2013 23:46:33 GMT
View Forum Message <> Reply to Message

On 4/26/13 3:37 PM, ben.bighair wrote:
> On Thursday, April 25, 2013 5:35:40 PM UTC-4, Jeremy Bailin wrote:
>> Under the category of "this must be easy, but I can't seem to figure out
>>
>> the right function":
>>
>>
>>
>> If I have created a Delauny tesselation using TRIANGULATE, how can I
>>
>> easily find which nodes form the triangle that contains an arbitrary
>>
>> point in the space?
>>
>>
>>
>> (more specifically, I am using /NATURAL_NEIGHB interpolation in GRIDDATA
>>
>> and it's going horribly wrong for one point, so I'm trying to figure out
>>
>> what nodes it's actually using in the interpolation for that point)
>>
>
> Hi,
>
> COuld you use IDLanROI::ContainsPoint()? You would have to convert the triangulations to ROIs first.
>
> Cheers,
> Ben
>

Yes, that would probably work... although converting each one separately to an ROI sounds like overkill to just figure out where one point lies. :) But that's the best suggestion, so I suspect that's what's going to happen.

-Jeremy.

Subject: Re: nearest node of Delauny tesselation
Posted by ben.bighair on Sat, 27 Apr 2013 02:32:30 GMT
View Forum Message <> Reply to Message

On Friday, April 26, 2013 7:46:33 PM UTC-4, Jeremy Bailin wrote:
> On 4/26/13 3:37 PM, ben.bighair wrote:
>
>> On Thursday, April 25, 2013 5:35:40 PM UTC-4, Jeremy Bailin wrote:
>
>>> Under the category of "this must be easy, but I can't seem to figure out
>
>>>
>
>>> the right function":
>
>>>
>
>>>
>
>>>
>
>>>
>
>>> If I have created a Delauny tesselation using TRIANGULATE, how can I
>
>>>
>
>>> easily find which nodes form the triangle that contains an arbitrary
>
>>>
>
>>> point in the space?
>
>>>
>
>>>
>
>>>
>
>>>
>
>>> (more specifically, I am using /NATURAL_NEIGHB interpolation in GRIDDATA
>
>>>
>
>>> and it's going horribly wrong for one point, so I'm trying to figure out
>
>>>
>
>>> what nodes it's actually using in the interpolation for that point)
>
>>>
>

>>
>
>> Hi,
>
>>
>
>> COuld you use IDLanROI::ContainsPoint()? You would have to convert the triangulations to ROIs first.
>
>>
>
>> Cheers,
>
>> Ben
>
>>
>
>
>
> Yes, that would probably work... although converting each one separately
>
> to an ROI sounds like overkill to just figure out where one point lies.
>
> :) But that's the best suggestion, so I suspect that's what's going to
>
> happen.
>

Well, you could at least narrow it down using the grid coordinates, the input point coordinates and VALUE_LOCATE. Seems like "horribly wrong" would stick out like a sore thumb if you could reduce it to just a handful of candidates.

If you go the IDLanROI route then be sure to stuff them into IDLanROIGroup and let it do the fussing with ContainsPoint().

```
TRIANGULATE, x,  y, tri, bounds
d = SIZE(tri, /dim)
rois = OBJ_NEW("IDLanROIGroup")
for i = 0, d[1]-1 do rois->Add, OBJ_NEW("IDLanROI", x[tri[*,j]], y[tri[j]])
inout = rois->ContainsPoints(badGridX, badGridY)
badTris = which(inout GT 0, nBadTris)
```