Subject: Re: Delaunay triangulation search
Posted by Charudatta Phatak on Thu, 11 Oct 2007 16:22:47 GMT
View Forum Message <> Reply to Message

Hi Ben,

Thanks for the help. But i was able to do it simply using the built in
idl function POLYFILLV which in fact does exactly what i wanted. it
finds the indices of all the points enclosed in a given polygon and it
works pretty fast too.

cheers,
-cd

ben.bighair wrote:
> On Oct 5, 6:17 pm, Charudatta Phatak <cpha...@andrew.cmu.edu> wrote:
>> Hello All,
>>
>> I want to know, is there a function in IDL which will give me the
>> corresponding indices of a traingle from a delaunay triangulation for
>> the given point. So suppose i use TRIANGULATE to generate a
>> triangulation. Then i want to go through the image pixel by pixel and
>> want to determine which pixels are enclosed in a particular triangle.
>>
>> In MATLAB, the function tsearch does this. This is the help of tsearch
>> from matlab website:
>>
>> tsearch
>> Search for enclosing Delaunay triangle
>> Syntax
>> T = tsearch(x,y,TRI,xi,yi)
>> Description
>> T = tsearch(x,y,TRI,xi,yi) returns an index into the rows of TRI for
>> each point in xi, yi. The tsearch command returns NaN for all points
>> outside the convex hull. Requires a triangulation TRI of the points x,y
>> obtained from delaunay.
>>
>> I think a function like QHULL in IDL might be able to do the same thing
>> but i'm unable to figure how to use it.
>> Thank you...
>>
>> cheers,
>> -cd
>
> Hi,
>

```
>  I am sure there are much more efficient ways, but I suggest the brute
>  force approach by using IDL's IDLanROI and IDLanROIgroup objects.
>  They each have a ContainsPoints method which comes in handy even if
>  not very fast (or maybe it is fast and I am just impatient!)  In any
>  event, I have modified the TRIANGULATE example from the online docs to
>  show what I mean.  Hope it helps get you started.
>
>  Cheers,
>  Ben
>
>  ***BEGIN
>  PRO TriSearch
>
>  ; Make 50 normal x, y points:
>  x = RANDOMN(seed, 50)
>  y = RANDOMN(seed, 50)
>
>  mmX = [MIN(x), MAX(x)]
>  mmY = [MIN(y), MAX(y)]
>
>  x = (x-mmx[0])/(mmx[1]-mmx[0]) * 99
>  y = (y-mmy[0])/(mmy[1]-mmy[0]) * 99
>
>
>  WINDOW, 0
>  ; Show points:
>  PLOT, x, y, psym=1,/ISO
>
>  ; Obtain triangulation:
>  TRIANGULATE, x, y, tr, b
>
>  roiGroup = OBJ_NEW('IDLanROIgroup')
>  ; Show the triangles:
>  FOR i=0, N_ELEMENTS(tr)/3-1 DO BEGIN
>     ; Subscripts of vertices [0,1,2,0]:
>     t = [tr[*,i], tr[0,i]]
>     ; Connect triangles:
>     PLOTS, x[t], y[t], psym = -3
>     roiGroup->Add, OBJ_NEW('IDLanROI', x[t], y[t])
>  ENDFOR
>
>  ;mock up an image and the indices for each pixel
>  img = REPLICATE(255B, 100,100)
>  xx = FINDGEN(100) # REPLICATE(1.0,100)
>  yy = REPLICATE(1.0,100) # FINDGEN(100)
>
>  ;find out if the pixels are in/out
>  ;you could devise a way of testing each polygon (triangle) -
>  ;it shouldn't be too hard - just arr = roiGroup->Get(/all)
>  ;to retrieve the individual rois, then loop through testing each
>  ;using the array[i]->ContainsPoints(xx,yy).  It is your call.
>  ok = roiGroup->ContainsPoints(xx,yy)
>  A = WHERE(ok, nA, COMP = B, NCOMP = nB)
>  if nA then PLOTS, xx[A],yy[A], psym = 3
>
>  if nB GT 0 then begin
>    WINDOW, 1, XSIZE = 100, YSIZE = 100
>    img[B] = 0
>    TVSCL, img
>  endif
>
>  END
>  ***END
>
```