Subject: Re: simplify a polyline?
Posted by b_gom on Mon, 26 Apr 2004 18:29:23 GMT
View Forum Message <> Reply to Message

My motivation for doing this is to reduce the size of files generated from plotting to a postscript device, or the contours output by the contour procedure, for example. These files can have 10^5 vertices or more.

The trouble with MESH_DECIMATE is that it is a memory hog. I tried to do this with MESH_DECIMATE but gave up quickly.

Here's what I came up with instead. It is a quick transcription of the Douglas-Peuker (1973) algorithm, as presented by Dan Sunday. Save all the procedures in 'poly_simplify.pro' and compile. 'Test.pro' shows an example. It should work in 2,3, or more dimensions. Its recursive, and not particularly fast, but seems to work.

The trouble is deciding which tolerance value to use. I've tried a simple way to do this automatically in the last lines of test.pro, but the result is still dependant on the relative scale of the x and y (or z) coordinates. There must be a better way..

poly_simplify.pro

```
function dot,x,y  ;dot product
 return, total(x*y)
end

function norm2,x  ;squared length of vector
; return,dot(x,x)
 return,total(x^2)
end

function d2,x,y   ;distance squared of difference between points x and
y
; return, norm2(x-y)
; return, dot(x-y,x-y)
 return,total((x-y)^2)
end

pro simplifyDP,tol,vertices,j,k,mk
; This is the Douglas-Peucker recursive simplification routine
; It just marks vertices that are part of the simplified polyline
; for approximating the polyline subchain vertices[j] to vertices[k].
; Input: tol = approximation tolerance
; vertices[] = polyline array of vertex points
```

```
; j,k = indices for the subchain vertices[j] to vertices[k]
; Output: mk[] = array of markers matching vertex array vertices[]
 j=long(j)
 k=long(k)

 if (k le j+1) then return ; there is nothing to simplify

 ; check for adequate approximation by segment S from vertices[j] to
vertices[k]
 maxi = j ; index of vertex farthest from S
 maxd2 = 0. ; distance squared of farthest vertex
 S = [[vertices[*,j]], [vertices[*,k]]]  ; segment from vertices[j] to
vertices[k]
 u = S[*,1]-S[*,0] ; segment direction vector
 cu = dot(u,u); segment length squared

 ;test each vertex vertices[i] for max distance from S
 ;compute using the Feb 2001 Algorithm's dist_Point_to_Segment()
 ;Note: this works in any dimension (2D, 3D, ...)

 ;Pb = base of perpendicular from vertices[i] to S
 ;dv2 = distance vertices[i] to S squared

 for i=j+1,k-1 do begin
  ;compute distance squared
  w = vertices[*,i] - S[*,0]
  cw = dot(w,u)
  if cw le 0 then begin
   dv2 = d2(vertices[*,i], S[*,0]);
   endif else begin
   if cu le cw then begin
    dv2 = d2(vertices[*,i], S[*,1])
    endif else begin
    b = cw / cu;
    Pb = S[*,0] + b * u;
    dv2 = d2(vertices[*,i], Pb);
    endelse
   endelse
  ;test with current max distance squared
  if dv2 le maxd2 then continue
  ;vertices[i] is a new max vertex
  maxi = i
  maxd2 = dv2
  endfor

 if (maxd2 gt tol^2) then begin ;// error is worse than the tolerance
;  split the polyline at the farthest vertex from S
  mk[maxi] = 1 ; mark vertices[maxi] for the simplified polyline
```

```
;  recursively simplify the two subpolylines at vertices[*,maxi]
  simplifyDP, tol, vertices, j, maxi, mk ; // polyline vertices[j] to
vertices[maxi]
  simplifyDP, tol, vertices, maxi, k, mk ; // polyline vertices[maxi]
to vertices[k]
  endif
;  else the approximation is OK, so ignore intermediate vertices
 return
end


function poly_simplify,vertices,tol
 ;vertices is a 2 or 3 (or more) by n array
 dim=size(vertices,/dimensions)
 n=dim[1] ;number of points
 if dim[0] lt 2 then begin
  message,'Vertices must be at least 2-D!',/cont
  return,vertices*0-1
  endif

 if n lt 2 then begin
  message,'There must be at least 2 Vertices!',/cont
  return,vertices*0-1
  endif

 vt=vertices*0 ;  vertex buffer
 mk=bytarr(n)  ;  marker buffer

 ; Mark vertices that will be in the simplified polyline
 ;Initially Mark V0 and Vn
 ; Recursively simplify by selecting vertex furthest away

; STAGE 1. Vertex Reduction within tolerance of prior vertex cluster
 vt[*,0] = vertices[*,0];  start at the beginning
 k=1L
 pv=0L
 for i=1L,n-1 do begin
  if (d2(vertices[*,i], vertices[*,pv]) lt tol^2) then continue
  vt[*,k++] = vertices[*,i];
  pv = i
  endfor

 if (pv lt n-1) then vt[*,k++] = vertices[*,n-1];  finish at the end

 ; STAGE 2. Douglas-Peucker polyline simplification
 mk[0] = 1
 mk[k-1] = 1  ;  mark the first and last vertices
```

```
 simplifyDP, tol, vt, 0, k-1, mk

 return, vt[*,where(mk)];  return simplified polyline
end


pro test
 RED =  [0, 220, 255, 255, 255, 0,  0,  255, 160, 255]
 GREEN =  [0, 140, 0,  127, 255, 255, 0,  0,  160, 255]
 BLUE =  [0, 127, 0,  0,  0,  0,  255, 255, 160, 255]
 TVLCT,red,green,blue


 x=findgen(500)
 y=sin(x/100*30)/5+sin(x/1000*30)

 vertices=transpose([[x],[y]])

 window,0,xsize=1000,ysize=700
 plot,[min(x),max(x)],[min(y),max(y)],/nodata
 plots,vertices


 ;distance between adjacent points
 d=(sqrt(total((vertices-shift(vertices,0,1))^2,1)))[1:*]

 ;minimum distance in x or y.. between adjacent points
 dmin=min(abs((vertices-shift(vertices,0,1))[*,1:*]))
 ;avg distance in x or y.. between points, whichever is smaller
  davg=(moment(abs((vertices-shift(vertices,0,1))[0,1:*])))[0]  <
(moment(abs((vertices-shift(vertices,0,1))[1,1:*])))[0]

 print,'smallest average distace in x or y:',davg,'   number of output
vertices: ',n_elements(poly_simplify(vertices,davg))/2
 print,'minimum distance in x or y',dmin,'   number of output
vertices: ',n_elements(poly_simplify(vertices,dmin))/2

 plots,poly_simplify(vertices,davg),psym=-4,color=2
 plots,poly_simplify(vertices,dmin),psym=-4,color=4


end



---------------------
```

Ben Tupper <btupper@bigelow.org> wrote in message
news:<c6j2sf$coqvb$1@ID-189398.news.uni-berlin.de>...
> Brad Gom wrote:
>
>> I need a general purpose routine for reducing the complexity of a 2-d
>> polyline. For example, the output of the contour function contains
>> many redundant points, ie. many vertices may be removed as they fall
>> on a straight or nearly straight line. Has anyone implemented a
>> polyline simplification or decimation routine? I don't want to simply
>> smooth the input data.
> Hi,
>
> I haven't tried this, but I believe that you can use MESH_DECIMATE using
> your X, Y vertices coupled with a faked Z value. This is from the
> online description of MESH_DECIMATE
>
>> The MESH_DECIMATE function reduces the density of geometry
>> while preserving as much of the original data as possible.
>
> Ben
