Subject: fast search
Posted by m.goullant@gmail.com on Tue, 17 Oct 2006 12:09:34 GMT
View Forum Message <> Reply to Message

Hi there,

I have the following problem:

```
;data structure of an irregular point cloud
x = points.x
y = points.y
z = points.z

search radio
radio = 8

FOR i=0L,N_ELEMENTS(z)-1 DO BEGIN

      square = WHERE(x LE x[i] + radio AND x GE x[i] - radio AND y LE
y[i] + radio AND y GE y[i] - radio)
     ;(...)

   ENDFOR
```

I realize that WHERE will do the job, but at very low efficiency.
WHERE
makes no assumptions about the list being ordered. It seems to me it
has
to check every element of the array, requiring N steps for an N-element
array

There is a faster way to do this?

thanks,
Marie

Subject: Re: fast search
Posted by m.goullant@gmail.com on Thu, 19 Oct 2006 10:51:57 GMT
View Forum Message <> Reply to Message

Just a little correction in the code: the dist is the mask :):

> This is more and less What I want to do:
>
> have this geographic data (could be 100000, 2 million, 4 million
> depends):
>

```
>
>  PRO example
>
>
>  points = myData()
>
>  ;data structure of an irregular point cloud
>  x = points.x ;X coord
>  y = points.y ;Y coord
>  z = points.z ; Elevation
>
>
>  n = 5 ; number of iterations
>  mask = 4.0 ; diameter. in meters. Like your dist
>  maskType = 0 ; 0 a circle, 1 square
>
>  FOR i=0L,n-1 DO BEGIN
>          newZ = erosion(x,y,z,mask,maskType)
>          ;(...)
>          mask = mask + 2
>  ENDFOR
>
>  END
>
>  FUNCTION erosion,x,y,z,mask,maskType ; Apply erosion to the data
>
>    newZ=z
>    radio = mask /2
>    FOR i=0L,N_ELEMENTS(z)-1 DO BEGIN
>                    kernel = applyKernel(x,y,z,i,radio,maskType)
>  ;center the kernel in the data(i) and get neighbours there are inside
>  of the mask
>                    newZ[i]=MAX(z[kernel])
>    ENDFOR
>    RETURN,newZ
>
>  END
>
>  FUNCTION applyKernel,x,y,z,i,dist,maskType
>
>    square = WHERE(x LE x[i] + dist AND x GE x[i] - dist AND y LE y[i]
>  + dist AND y GE y[i] - dist)
>
>    IF (maskType EQ 0) THEN BEGIN
>          sqDistance = sqrt((x[square] - x[i])^2 + (y[square] -
>  y[i])^2)
>          neighbors = WHERE(sqDistance LE dist)
>          circle = square[neighbors]
```

> RETURN,circle
> ENDIF
>
> RETURN,square
> END
>
> Because this is an iterative process with a "dist" variable, it's
> posible implement your code?
>
>
> Thank's in advance,
> Marie

Subject: Re: fast search
Posted by greg michael on Thu, 19 Oct 2006 14:18:12 GMT
View Forum Message <> Reply to Message

Hi Marie,

Here's an annotated version.

I understand you're looking for the maximum z within the vicinity of every point, with a gradually increasing radius of the vicinity. I can't see what you're doing with this value, though - does it feed back into the set of points somehow? What's the result you're trying to get? Where do these points come from?

It would be simple to reduce my search code to 2-D - just remove the z-lines, change the distance calculation, and the binning line to b=bx+by*n_split. But I'm not sure if this is right way - it depends what your z-values mean.

I've just realised that my later versions don't handle the case where the pair lie across a subdivision boundary - only the slower first version does that. Something to fix...

many greetings,
Greg

Subject: Re: fast search
Posted by greg michael on Thu, 19 Oct 2006 14:24:21 GMT
View Forum Message <> Reply to Message

erm... I mean here...

Greg

```
pro splitsearch3,p,dist
 ;recursively splits the search volume into n_split^3 subvolumes. When
 ;there are fewer than 'threshold' points
 ;in a subvolume, checks for matches the brute force way - every point
 ;against every other.

 n_split=4        ;1-D cutting factor (for 3, cube is cut into 3x3x3=27
subvolumes)
 threshold=75     ;no. of points to start pairwise comparison
 n=n_elements(p)  ;no. of points in cloud

 if n gt threshold then begin ;if more than threshold points, further
divide the volume
  mxx=max(p.x,min=mnx) ;get range of x
  mxy=max(p.y,min=mny) ;get range of y
  mxz=max(p.z,min=mnz) ;get range of z

  ;determine which subrange of x,y,z each point belongs to:
  bx=fix((p.x-mnx)/(mxx-mnx)*n_split)<(n_split-1) ;< to ensure max
element not in new bin
  by=fix((p.y-mny)/(mxy-mny)*n_split)<(n_split-1)
  bz=fix((p.z-mnz)/(mxz-mnz)*n_split)<(n_split-1)

  ;determine which subvolume each point belongs to (for n_split=3 there
will be 3x3x3=27)
  b=bx+by*n_split+bz*n_split^2

  ;get reverse_indices - a ordered list telling which points lie in
which subvolume (bin)
  ;                    Note that how this works is obscure - check
here to understand:
  ;
http://www.dfanning.com/tips/histogram_tutorial.html
  h=histogram(b,min=0,reverse_indices=ri)

  for i=0,n_elements(h)-1 do begin
   if ri[i] ne ri[i+1] then begin ;see again histogram tutorial
    q=[ri[ri[i]:ri[i+1]-1]]       ;see again histogram tutorial

    ;if there are more than two points in the subvolume, call this
routine with just those points
    ;this is what makes the routine recursive. Eventually there will be
fewer than threshold points,
    ;and then the second half of the routine gets executed, with the
brute force comparison.
    if n_elements(q) ge 2 then splitsearch3,p[q],dist ;splitsearch
```

```
again, if enough to compare
   endif
  endfor

 endif else begin
  ;This is the brute force comparison. Using q1 and q2 as indices to p
lets you compare every element
  ;against every other. d is a matrix of these pairwise distances
  q1=rebin(indgen(n),n,n) ;set up indices for pairwise matching
  q2=transpose(q1)
   d=sqrt((p[q1].x-p[q2].x)^2+(p[q1].y-p[q2].y)^2+(p[q1].z-p[q2 ].z)^2)
;calculate pair distances
  i=where((d le dist) and (q1 gt q2)) ;select close neigbours (q1>q2 to
avoid duplicate pairs)

  ;this is where the results come out:
  if i[0] ne -1 then print,p[q1[i]],p[q2[i]] ;print the neighbours, if
found
 endelse
end
```

Subject: Re: fast search
Posted by m.goullant@gmail.com on Sat, 21 Oct 2006 00:37:49 GMT
View Forum Message <> Reply to Message

greg michael wrote:
> Hi Marie,

Hello once again Greg! :-)

> Here's an annotated version.
>
> I understand you're looking for the maximum z within the vicinity of
> every point, with a gradually increasing radius of the vicinity. I
> can't see what you're doing with this value, though - does it feed back
> into the set of points somehow? What's the result you're trying to get?

Z is a elevation value (sea level)

I need the calculate de altimetric difference (difZ) after for example
a morphological operation of erosion to all data:

difZ = z-newZ

with this result I can make some comparisons and point exclusions...The
main problem I have is when I apply this filtering and iterative
process to a great volume of data.

Imagine if I want to calculate a morphological Opening operation
(erosion following dilation)?

> Where do these points come from?

These points come from an ASCII file (.dat;.txt;.xyz), here a little
sample:

"X","Y","Z"
645107.178,4808512.652,382.900
645106.228,4808512.642,381.940
645104.798,4808512.492,378.220
645103.678,4808512.502,377.500
645100.329,4808511.973,366.170
645089.639,4808512.643,367.570
645097.298,4808512.602,375.630
(...)

> It would be simple to reduce my search code to 2-D - just remove the
> z-lines, change the distance calculation, and the binning line to
> b=bx+by*n_split. But I'm not sure if this is right way - it depends
> what your z-values mean.
>
> I've just realised that my later versions don't handle the case where
> the pair lie across a subdivision boundary - only the slower first
> version does that. Something to fix...
>

I will try to see in what way I can adapt the cut method to this
situation: where when I center the kernel in a point(i) I only
calculate euclidean distances to that point in the subvolume or
subvolumes that are inside of the kernel
Any idea will be welcome :-)
Thanks!

> many greetings,
> Greg
Thanks! Same
Marie

Subject: Re: fast search
Posted by greg michael on Sat, 21 Oct 2006 21:42:28 GMT
View Forum Message <> Reply to Message

Hi Marie,

So this is a set of independent point measurements? The instrument is some kind of single-shot altimeter? And you want to reduce the set of points to something which represents a maximum surface?

Is it not adequate to simply bin the data over an x,y-grid and take the max value for each bin? Have you looked at the IDL KRIG2D function? This would handle the interpolation of empty cells (I've never tried it myself, but I think that's what it does).

Otherwise, perhaps you could give a more detailed description of what you're trying to achieve, and what the correct result would be...

good luck!
Greg
