Subject: Re: Need Some Good Ideas
Posted by btupper on Thu, 21 Feb 2002 18:43:21 GMT
View Forum Message <> Reply to Message

> I have a bunch of blobs. (Think spots on the
> Gateway cow.) I would like to analyze the curvature
> and bends in the perimeter of the blobs. I have
> the indices of the points that make up the blob, and
> I have obtained the "perimeter" points by contouring
> the blob. Unfortunately, these perimeter points are
> not evenly distributed. (Think of a blob that has a
> long, straight side. The contour command will put a
> point at either end of the straight bit, so the points
> on that side of the blob are sparse, while the points
> along a tight bend on the other side of the blob
> are dense.)

Hi,

I have stumbled over this ground before. I use the FillIn_Contour function pasted in below. It's pretty typical of my brute force technique for doing geometery with IDL.

While we are on the subject of contouring, I recall that a call to CONTOUR with the Path_XY keyword set to retrieve the vertex pairs will change the direct graphics plot scaling. A call to CONTOUR with the OverPlot keyword set does not change the plot scaling. I *always* forget that Contour does this when someone signing my paycheck is looking over my shoulder. I show an example of this in the FunnyContour procedure below.

Ben

```
;*******START EXAMPLE HERE

PRO FunnyContour

TVLCT, R, G, B, /get

Img = BytArr(100,100)
Img[25:75,25:75] = 1
Img = Rot(Img, 30)
Img[0:50, *] = 0

LoadCT, 0, bottom = 32
```

```
Tek_Color
ImDisp, Img, /no_Scale, /Axis, /Erase
Print, 'XS after plot ',  !X.S

 ;save these for later
XS = !X.S
YS = !Y.S


;contour directly to the display - shown in red
Contour, Img, levels = [1], c_color = 2, /over
Print, 'XS after contour to display ',  !X.S

;contour directly to the output keywords show in green
Contour, Img, levels = [1], Path_XY = xy, /Path_Data_Coord
Print,   'XS after contour to path_xy ',  !X.S
oPlot, XY[0,*], XY[1,*], color = 3, psym = 1

;restore the plotting parameters
;and show the results in blue
!X.S = XS
!Y.S = YS
oPlot, XY[0,*], XY[1,*], color = 4, psym = 2


 ;pad in the missing pixel locations
 ;show these in orange
newXY = FillIn_Contour(XY)
oPlot, newXY[0,*], newXY[1,*], color = 8, psym = 5

TVLCT, R, G, B

END

;*******END EXAMPLE HERE


;**********ANOTHER START HERE
;+
; NAME:
; FillIn_CONTOUR
;
; PURPOSE:
; Given the XY data path coordinates returned
; from a call to CONTOUR using
; an image, this routine returns ALL of the pixels
; coordinates that lie along the contour.
;
```

```
; CATEGORY:
; Image analysis.
;
; ARGUMENTS:
; XY  A 2xn element array of contour vertices returned
;  by the CONTOUR routine keyword PATH_XY.
; DELTA  A one or two element vector of the 'pixel'
; size in the x and y directions. By default [1,1] is used.
;
; RETURNED:
; A 2xm element array of the coordinates of all
; the pixels/cells that lie along the path.
; The first point and last are the not the same.
;
; EXAMPLE:
; ;make an image
; img = BytArr(250,250)
; Img[50:200, 50:200] = 1
; Tek_color
; TV, Image
;  CONTOUR, img, Path_XY = XY,
;  /Path_Data_Coord, Levels = [1]
;  ;only 4 vertices (since the 'island' is a rectangle
; plots, XY[0,*], XY[1,*], psym = 6, color = 2, /dev
;  ;get all the pixels
;  newxy = filin_contour(xy)
;  ;show the new points
; plots,newXY[0,*], newXY[1,*], psym = 3, /dev, color = 3
;
; MODIFICATION HISTORY
; 18 JUNE 2001 Written by Ben Tupper.
;-

FUNCTION FillIn_Contour, XY, delta

 ;establish the step size
Case n_elements(Delta) of
 0: d = [1.0,1.0]
 1: d = [float(delta),delta]
 Else: d = Float(Delta[0:1])
EndCase

d = ABS(d)
neg_d = 0.0 - d

;convert original array into 'wrapped' vectors
oX = [Reform(XY[0,*]), XY[0,0]]
oY = [Reform(XY[1,*]), XY[1,0]]
```

```
 ;seed the new xy paring
X = Ptr_new(oX[0])
Y = Ptr_New(oY[0])

 ;step through the original verticies, padding the
 ;'new' vertices as needed

i = 1L & newCount = 1L
While i lt n_elements(oX) do begin

  ;get the x and y displacements
 dx = oX[i] - (*X)[newCount-1]
 dy = oY[i] - (*Y)[newCount-1]

 Case 1 of

   ;in this case the next vertex in origianl set
is more than
   ;one pixel away
  ABS(dx) GT d[0] OR ABS(dy) GT d[1]: Begin

  *X = [*X,(*X)[newCount-1] + (neg_d[0] > dx < d[0])]
  *Y = [*Y,(*Y)[newCount-1] + (neg_d[1] > dy < d[1])]
    ;the newCount will be incremented, but
not the
    ;counter for the old set
   newCount = NewCount+1

  End ;big gap to be filled in

  ;in this case we have arrived at a vertex... so simply

  ;move on to the next vertex in the original data set
 dx EQ 0 AND dy EQ 0: i = i+1

  ;in this case the next vertex is the one form the
original data set
 Else:  Begin

  *X = [*X, oX[i]]
  *Y = [*Y, oY[i]]
   ;move on to the next vertex
  i = i + 1
  NewCount = NewCount + 1
  End

 EndCase
```

```
EndWhile


NewXY = $
 transpose( [ [ (*X)[0: newCount-2] ],[ (*Y)[0: newCount-2] ]
])
ptr_free, X,Y

Return, NewXY
END
;*********ANOTHER END HERE
```