Subject: Re: Chain-Link Algorithm for Perimeter
Posted by dmarshall on Fri, 19 Apr 2002 20:22:29 GMT
View Forum Message <> Reply to Message

If you know the pixel co-ords and your blob has no "serious" cavities (ie, openings that cause the perimeter to fold back on itself) you can convert your cartesian co-ords to polar (centered about the blob center), sort on the angle and convert back.

But then I guess, if you could assume the constraints of this method, you wouldn't be asking about a Chain-Link method.....

....never mind.

Dave

In article <MPG.172a01b613e13adb98989e@news.frii.com>, David Fanning <david@dfanning.com> writes:
> Folks,
>
> I have a blob (naturally) and I want to know the
> pixels, in order, that describe the perimeter of
> the blob. The output of the Contour command, etc.
> is inappropriate, in this case.
>
> I think what I want to use is the Chain-Link Coding
> algorithm of R.L.T. Cederberg as described in The
> Image Processing Handbook by John Russ. Has anyone
> coded this up in IDL by any chance?
>
> I've been known to pay for code that saves me a ton
> of time. :-)
>
> Cheers,
>
> David
>
> --
> David W. Fanning, Ph.D.
> Fanning Software Consulting
> Phone: 970-221-0438, E-mail: david@dfanning.com
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Chain-Link Algorithm for Perimeter
Posted by David Fanning on Fri, 19 Apr 2002 20:41:09 GMT

View Forum Message <> Reply to Message

dmarshall@ivory.trentu.ca (dmarshall@ivory.trentu.ca) writes:

> If you know the pixel co-ords and your blob has no "serious" cavities (ie,
> openings that cause the perimeter to fold back on itself) you can convert
> your cartesian co-ords to polar (centered about the blob center), sort on
> the angle and convert back.

Alas, those damn blobs are *full* of "serious" cavities,
which is exactly what is causing my current algorithm
to go cross-eyed. :-(

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Chain-Link Algorithm for Perimeter
Posted by Ted Cary on Sun, 21 Apr 2002 00:01:05 GMT

View Forum Message <> Reply to Message

David Fanning wrote

>
> I think what I want to use is the Chain-Link Coding
> algorithm of R.L.T. Cederberg as described in The
> Image Processing Handbook by John Russ. Has anyone
> coded this up in IDL by any chance?
>
> I've been known to pay for code that saves me a ton
> of time. :-)

Hi David,

I coded up a routine that should order the boundary points of your blobs
for you. I don't know if it uses the chain-link algorithm. It's based
on C++ code by Dave Eberly of Magic Software. It will handle
concavities and folds but not all bow ties, at least in my tests.

It took me a good part of the day, but Eberly and IDL did most of the
work. I've been meaning to code something like this for myself anyway

and your request just got me started. It's still not the UNMASK routine
I was looking for in an earlier post, but it gets the job done.

The routine is called simply "BOUNDARY" and is included in this post as
both an attachment and as text below.   See the header for instructions.

Enjoy,

TC

```
;+
; NAME:
;   BOUNDARY
;
; PURPOSE:
;
;    Returns ordered lists of x and y coordinates of ALL points on the
boundary of a region.
;   Will work on regions with concavities and regions whose boundaries
have folds.
;
; AUTHOR:
;
;  Ted Cary
;    adapted from C++ code by David Eberly
;    (see www.magic_software.com/ImageAnalysis.html)
;
; CATEGORY:
;
;   Image Analysis/Contouring.
;
; CALLING SEQUENCE:
;
;   BOUNDARY, mask, X, Y
;
;   MASK : A (binary) 2D array of the type put out by
IDLanROI::ComputeMask.
;          The interior  points of the region to be contoured should be
set
;          to 1, while background points should be set to 0.
;
;   X : An output vector that will contain the x coordinates of the
boundary points.
;
;   Y : An output vector that will contain the y coordinates of the
boundary points.
;
```

```
; EXAMPLE:
;
; Find boundary of blob made of overlapping disks::
;
;  ; Create disk of radius r
;
; r = 50
; disk = Shift(Dist(2*r+1), r, r) LT r
;
;  ; Create "blob" of overlapping disks
;
; mask1 = BytArr(275, 200)
; mask2 = BytArr(275, 200)
; mask1[49, 49] = disk
; mask2[124, 49] = disk
; mask = mask1 OR mask2 ; overlap two disks
;
;  ; Show blob
;
; TVSCL, mask
;
;  ; Get boundary
;
; Boundary, mask, x, y
;
;  ; Plot Boundary in red (using 24-bit color)
;
; PlotS, x, y, Color=255, /Device
;
; LICENSE:
;
;   Please give me and Dave Eberly of Magic Software credit for the code
and
;   note any changes you make to it. Don't remove this notice, etc...
;
;   "AS IS", no warranty, express or implied.  Authors are not liable.
;
;   Enjoy.
;
 ;################################################################# ################



PRO Boundary, mask, X, Y

 ; Create distance map of mask, padded by two on each side.

dims = Size(mask, /Dimensions)
```

```
distMap = BytArr(dims[0] + 4 , dims[1] + 4)
distMap[2,2] =  Morph_Distance(mask, Neighbor_Sampling=0)

 ; Set initial coordinate of starting boundary point.

startIndex = Max(Where(distMap EQ 1))
x0 = startIndex MOD (dims[0] + 4)
y0 = startIndex/(dims[0] + 4)

 ; Set up x, y directional tables.

dX = [-1,  0, +1, +1, +1,  0, -1, -1]
dY = [-1, -1, -1,  0, +1, +1, +1,  0]

 ; Determine direction from background to start point.

iCx = x0
iCy = y0
FOR idir = 0, 7 DO BEGIN
   iNx = iCx + dX[idir]
   iNy = iCy + dY[idir]
   IF distMap[iNx, iNy] NE 0 THEN  BEGIN
  idir = (idir + 1) MOD 8
  BREAK
 ENDIF
ENDFOR

 ; Initialize arrays to hold found boundary vertex coordinates.

xList = [x0 - 2]
yList = [y0 - 2]

 ; Traverse boundary in CW order.
 ; Mark traversed points as -1 in distance map.

distMap[x0, y0] = -1 ; Already visited first point.

WHILE(1) DO BEGIN
 iNbr = iDir
 FOR i=0, 7 DO BEGIN
  iNx = iCx + dx[iNbr]
  iNy = iCy + dy[iNbr]
  iNbr = (iNbr + 1) MOD 8
  IF distMap[iNx, iNy] EQ 1 THEN BREAK
 ENDFOR

 IF i EQ 8 THEN BREAK
 IF iNx EQ x0 AND iNy EQ y0 THEN BREAK
```

```
  ; Update vertex coords lists.

 xList = [xList, iNx-2]
 yList = [yList, iNy-2]

  ; Mark visited pixels.

 distMap[iNx, iNy] = -1

  ; Set new point coords, new direction.

 iCx = iNx
 iCy = iNy
 iDir = (i + 5 + iDir) MOD 8
ENDWHILE

 ; Output vectors.

x = xList
y = yList
 END;-----------------------------------------------------



;+
; NAME:
;   BOUNDARY
;
; PURPOSE:
;
;  Returns *ordered* lists of x and y coordinates of ALL points on the boundary a region.
;   Will work on regions with concavities and regions whose boundaries have folds.
;
; AUTHOR:
;
;  Ted Cary
;    adapted from C++ code by David Eberly
;    (see www.magic_software.com/ImageAnalysis.html)
;
; CATEGORY:
;
;   Image Analysis/Contouring.
;
; CALLING SEQUENCE:
;
;   BOUNDARY, mask, X, Y
;
```

```
;   MASK : A (binary) 2D array of the type put out by IDLanROI::ComputeMask.
;          The interior  points of the region to be contoured should be set
;          to 1, while background points should be set to 0.
;
;  X : An output vector that will contain the x coordinates of the boundary points.
;
;  Y : An output vector that will contain the y coordinates of the boundary points.
;
; EXAMPLE:
;
; Find boundary of blob made of overlapping disks::
;
;  ; Create disk of radius r
;
; r = 50
; disk = Shift(Dist(2*r+1), r, r) LT r
;
;  ; Create "blob" of overlapping disks
;
; mask1 = BytArr(275, 200)
; mask2 = BytArr(275, 200)
; mask1[49, 49] = disk
; mask2[124, 49] = disk
; mask = mask1 OR mask2 ; overlap two disks
;
;  ; Show blob
;
; TVSCL, mask
;
;  ; Get boundary
;
; Boundary, mask, x, y
;
;  ; Plot Boundary in red (using 24-bit color)
;
; PlotS, x, y, Color=255, /Device
;
; LICENSE:
;
;   Please give me and Dave Eberly of Magic Software credit for the code and
;   note any changes you make to it. Don't remove this notice, etc...
;
;   "AS IS", no warranty, express or implied.  Authors are not liable.
;
;   Enjoy.
;
 ;################################################################ #################
```

```
PRO Boundary, mask, X, Y

 ; Create distance map of mask, padded by two on each side.

dims = Size(mask, /Dimensions)
distMap = BytArr(dims[0] + 4 , dims[1] + 4)
distMap[2,2] =  Morph_Distance(mask, Neighbor_Sampling=0)

 ; Set initial coordinate of starting boundary point.

startIndex = Max(Where(distMap EQ 1))
x0 = startIndex MOD (dims[0] + 4)
y0 = startIndex/(dims[0] + 4)

 ; Set up x, y directional tables.

dX = [-1,  0, +1, +1, +1,  0, -1, -1]
dY = [-1, -1, -1,  0, +1, +1, +1,  0]

 ; Determine direction from background to start point.

iCx = x0
iCy = y0
FOR idir = 0, 7 DO BEGIN
   iNx = iCx + dX[idir]
   iNy = iCy + dY[idir]
   IF distMap[iNx, iNy] NE 0 THEN  BEGIN
  idir = (idir + 1) MOD 8
  BREAK
 ENDIF
ENDFOR

 ; Initialize arrays to hold found boundary vertex coordinates.

xList = [x0 - 2]
yList = [y0 - 2]

 ; Traverse boundary in CW order.
 ; Mark traversed points as -1 in distance map.

distMap[x0, y0] = -1 ; Already visited first point.

WHILE(1) DO BEGIN
 iNbr = iDir
 FOR i=0, 7 DO BEGIN
  iNx = iCx + dx[iNbr]
  iNy = iCy + dy[iNbr]
  iNbr = (iNbr + 1) MOD 8
  IF distMap[iNx, iNy] EQ 1 THEN BREAK
 ENDFOR

 IF i EQ 8 THEN BREAK
 IF iNx EQ x0 AND iNy EQ y0 THEN BREAK

  ; Update vertex coords lists.

 xList = [xList, iNx-2]
 yList = [yList, iNy-2]

  ; Mark visited pixels.

 distMap[iNx, iNy] = -1

  ; Set new point coords, new direction.

 iCx = iNx
 iCy = iNy
 iDir = (i + 5 + iDir) MOD 8
ENDWHILE

 ; Output vectors.

x = xList
y = yList
 END;--------------------------------------------------------
```

File Attachments

1) boundary.pro, downloaded 101 times

Subject: Re: Chain-Link Algorithm for Perimeter
Posted by gutmann on Fri, 26 Apr 2002 20:58:04 GMT
View Forum Message <> Reply to Message

heh, I wrote something to remove the boundary pixels following a
contour when I was first learning IDL. You're welcome to it though it
comes with all the standard caveats about my not being responsible for
it eating your machine and kicking you out of the house.

(warning lots of for loops ahead)
http://aster.colorado.edu/tmp/edgeRemove.pro

I think it follows a very similar algorithm, I don't think I've ever
come across a shape it died on (as long as two shapes aren't
touching). I originally wrote c-code to trace sand grain boundaries

so it saw some pretty ugly shapes, this is modified from the c-code
but it should be equally robust to odd boundaries.

Ethan