Subject: Label_region and Erosion
Posted by lbryanNOSPAM on Tue, 03 Nov 1998 08:00:00 GMT
View Forum Message <> Reply to Message

I'm sending the group some sample code (it hasn't been overly debugged or documented, but should work) I've written with some of David Fanning's and Struan (Gray)'s code as a base. I'm looking to create a multi-surface plot, from a 2-D data array. The procedure is straightforward and works great on the simple data set I've set up. I need, however, something that can handle some ambiguous and noisy surfaces. My first question is how do I find out what algorithm is used in LABEL_REGION? Since I do not see this function in the library, I assume it is written in C somewhere. I'm trying to use it to detect surfaces in my target volume and am having mixed results. How does it decide what is a unique surface and what is only a bump on a surface? Thanks for any info you can pass along.

Also, I've had a suggestion to use morphologic filters, erode and dilate. They look helpful for my goal. From the IDL books, I think I see how they work on binary applications, but the greyscale use is confusing me. Does anyone have an example of how they work on this kind of an application?

Lastly, I wan't to apply a median filter (and possible other filters) to some sections of my surfaces but not others (all irregular shapes). I imaging I'll have to write my own procedure where I pass over my data with a filter and a masking function to exclude certain areas. Has anyone already done this? Am I missing an easy way to do this?

Thanks in advance. Here's my code (messy as it is!).

Lisa Bryan

```
PRO MULTI_SURF_EXAMPLE

plane = fltarr(100,100)
plane(55:85,20:50) = dist(31)+10
plane(5:15,40:70) = (findgen(341))/100+30
plane(30:35,5:75) = (findgen(426))/100+20
plane(75:95,75:95) = (findgen(441))/100+25

shade = dist(100)

MULTI_SURF, PLANE, SHADE, MAXSHADE = 70, b

END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
PRO MULTI_SURF, bottdepth, bottshade, maxshade = maxshade,$
 b
;+
; NAME:
; MULTI_SURF
; PURPOSE:
; To seperate a single surface into multisurfaces and
; plot them all to a single 3-D SHADE_SURF plot.
;
; CALLING SEQUENCE:
; MULTI_SURF, bottdepth, bottshade
;
; INPUTS:
; bottdepth is a 2d array that is the depth of each pixel
; bottshade is a 2d array with the same dimensions as bottdepth
; which shows the relative intensity of each pixel
;
; INPUT KEYWORDS:
; MAXSHADE: This is the maximum value to be included in the
; shades which are to be plotted over the surface.
;
;
; OUTPUTS:
;
;
;
; NOTES:
;
; REVISION HISTORY:
; Written E.L. Bryan Nov. 1998
;

;  check the dimensions

if (total(size(bottdepth)) ne total(size(bottshade))) then begin
 print, 'Depth and Shade arrays must be of the same dimension'
 return
endif

image = bottdepth
image = image - min(image) ;set min(image) to 0

szimage = size(image)

b = LABEL_REGION(image,/eight) ;define surfaces in b
h = HISTOGRAM(b, REVERSE_INDICES=r) ;Get population and members of

      ;each blob.
```

```
;set up Z buffer
thisDevice = !D.Name
Set_Plot, 'Z'
Device, Set_Resolution=[szimage(1) > 500,szimage(2) > 400]

;set up axes with no data
surface,image,zrange = [min(image),max(image)],$
  min_value = min(image),/nodata



FOR i=0, N_ELEMENTS(h)-1 DO BEGIN ;Select each region

 current_surf = image
 current_surf(where(b(*) ne i)) = -10 ;place everything but
    ;current surface below plotted region

 shade_surf,current_surf,zrange = [min(image),max(image)],$
  min_value = min(image),/noerase, $
  shades = bytscl(bottshade,max = maxshade)
 ;count regions
 p = r(r[i]:r[i+1]-1) ;Subscripts of members of region i.
 q = image[p] ;Pixels of region i
 PRINT, 'Region ', i, $
  ', Population = ', h[i]
ENDFOR

snapshot = TVRD()
Set_Plot, thisDevice
window,0,xs = szimage(1) > 500,ys = szimage(2) > 400
TV, snapshot
end
```

Arete Associates
Tucson, Arizona
lbryan@arete-az.com

Subject: Re: Label_region and Erosion
Posted by Struan Gray on Wed, 04 Nov 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Lisa Bryan, lbryanNOSPAM@arete-az.com writes:

> Now if I can just get dilate to work on a
> greyscale image.

I'm not sure you really need any help, but here are some random

thoughts.

As a general rule, IDL is much happier, and programs run much faster, if you use as much memory and as few loops and sub-arrays as possible. Unfortunately, the duty sadists at RSI have seen to it that you have no decent tools within IDL to keep track of your memory usage. However, your images are not too large, so keeping extra masks and working copies of the image in memory is not going to be a problem on most platforms, even if like me you need to keep users with elderly, RAM-poor, PC-class machines in mind.

So you can always make a binary mask from your greyscale image, and use that to define your regions. The most obvious way is to manually set pixel values using the output of a WHERE call, but often it is possible, and faster, to use a array comparison like:

mask = image > threshold_val

where threshold_val distinguishes between the regions (100 or so in your example) and mask becomes a binary image containing 0s where the condition is false and 1s where it is true. Then you can erode and dilate this mask to get rid of single-pixel spikes and feed the result to LABEL_REGION to find the regions themselves.

At this point I find it useful to make a *second* mask, which is a n-times dilated version of the first. By subtracting the first from the second I get a mask containing a n-pixel wide ribbon of boundary pixels for the regions I want to filter. I can then use the ribbon to fill the relevant pixels of a temporary copy of the original image with values appropriate to whatever filter I want to apply (region average, nearest value, NANs, whatever). Having filtered the copy of the image containing the ad-hoc ribbon values I copy the filtered regions back into the final image using the first mask as a guide.

This sounds complex, but is not hard to implement and is particularly useful for things like differentiation filters where the example you gave would produce nasty effects at the edges of the dilated but unaltered regions where the values go from 300+/-10 to 0+/-10. My way puts all the nasties outside the region you are actually going to use. Downsides are that if you want to filter both the 300+/-10 areas and the 0+/-10 ones you have to do the whole thing twice, and that the choice of pixel values to put into the ribbons, while often perfectly logical and scientifically respectable, is a fudge that might lead unsophisticated users astray, though the degree to which that occurs depends on both the nature of your data, and of your users.

Enough waffle. I hope this helps.

Struan

Subject: Re: Label_region and Erosion
Posted by lbryanNOSPAM on Wed, 04 Nov 1998 08:00:00 GMT
View Forum Message <> Reply to Message

On 4 Nov 1998 12:10:34 GMT, Struan Gray <struan.gray@sljus.lu.se>
wrote:

> when playing with masked filtering I have found that it
> is easiest to filter a copy of the whole dataset and then use the mask
> to pick out the parts you actually wanted and insert them into the
> original data. For my sorts of data the time penalty incurred by
> filtering everything is more than compensated for by the generality of
> the procedure (multiple, oddly shaped, reentrant regions are handled
> transparently) and it's programming simplicity. You also have a
> nicely behaved default behaviour for how boundary values are dealt
> with when using filters of various widths.
>
>
> Struan

Ok, I think this may work...

This is my data. Where the 0's represent pixel values which have a
mean of 0 and deviation of about 10. And the +'s have a mean of 300
and a deviation of about 30. And the size of the array is actually
about 400x400. Plus both surfaces have a few nasty noise spikes that
are significantly higher or lower than described.

```
00000000000000000000
000000+000++00000000
0000+++00+++++000000
0000++++++++00000000
00000+0++++++0+00000
00000000++0++++00000
000000000000++++0000
00000000000000+00000
00000000000000000000
00000000000000000000
```

If I can get dilate to work on my greyscale image can expand the size
of the island by the width of my filter, filter the entire image, then
select the original region, I should be able to avoid all edge
effects. (Up to this point filtering causes all my surfaces to turn

into upside down bowls.) Now if I can just get dilate to work on a
greyscale image. Thanks for the ideas!

Lisa B.

Arete Associates
Tucson, Arizona
lbryan@arete-az.com

Subject: Re: Label_region and Erosion
Posted by Struan Gray on Wed, 04 Nov 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Lisa Bryan, lbryanNOSPAM@arete-az.com writes:

> Lastly, I wan't to apply a median filter (and possible other filters)
> to some sections of my surfaces but not others (all irregular shapes).
> I imaging I'll have to write my own procedure where I pass over my
> data with a filter and a masking function to exclude certain areas.
> Has anyone already done this? Am I missing an easy way to do this?

I haven't used the morphological functions much so can't comment on that, but when playing with masked filtering I have found that it is easiest to filter a copy of the whole dataset and then use the mask to pick out the parts you actually wanted and insert them into the original data. For my sorts of data the time penalty incurred by filtering everything is more than compensated for by the generality of the procedure (multiple, oddly shaped, reentrant regions are handled transparently) and it's programming simplicity. You also have a nicely behaved default behaviour for how boundary values are dealt with when using filters of various widths.

Struan

Subject: Re: Label_region and Erosion
Posted by David Foster on Thu, 05 Nov 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Lisa Bryan wrote:
>
> On 4 Nov 1998 12:10:34 GMT, Struan Gray <struan.gray@sljus.lu.se>
> wrote:
>
>> when playing with masked filtering I have found that it

>> is easiest to filter a copy of the whole dataset and then use the mask
>> to pick out the parts you actually wanted and insert them into the
>> original data. For my sorts of data the time penalty incurred by
>> filtering everything is more than compensated for by the generality of
>> the procedure (multiple, oddly shaped, reentrant regions are handled
>> transparently) and it's programming simplicity. You also have a
>> nicely behaved default behaviour for how boundary values are dealt
>> with when using filters of various widths.
>>
>>
>> Struan
>
> Ok, I think this may work...
>
> This is my data. Where the 0's represent pixel values which have a
> mean of 0 and deviation of about 10. And the +'s have a mean of 300
> and a deviation of about 30. And the size of the array is actually
> about 400x400. Plus both surfaces have a few nasty noise spikes that
> are significantly higher or lower than described.
>
> 00000000000000000000
> 000000+000++00000000
> 0000+++00+++++000000
> 0000++++++++00000000
> 00000+0++++++0+00000
> 00000000++0++++00000
> 000000000000++++0000
> 00000000000000+00000
> 00000000000000000000
> 00000000000000000000
>
> If I can get dilate to work on my greyscale image can expand the size
> of the island by the width of my filter, filter the entire image, then
> select the original region, I should be able to avoid all edge
> effects. (Up to this point filtering causes all my surfaces to turn
> into upside down bowls.) Now if I can just get dilate to work on a
> greyscale image. Thanks for the ideas!
>
> Lisa B.
>
> Arete Associates
> Tucson, Arizona
> lbryan@arete-az.com

Lisa -

Be sure you understand what the DILATE function is doing when
used on grayscale images. From the OnLine help:

Used with grayscale images, which are always converted to byte type, the DILATE function is accomplished by taking the maximum of a set of sums. It can be used to conveniently implement the neighborhood maximum operator with the shape of the neighborhood given by the structuring element.

To use DILATE and ERODE to implement convolution, which sounds like what you would like to do, you need to use a binary image mask. You can always manipulate the mask, and then apply it to your original image to get the "new" image. You might want to play around with "openings" and "closings" (combinations of Erodes and Dilates).

Hope this is useful.

Dave
--

~~~~~~ ~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~ ~~~~~~

Subject: Re: Label_region and Erosion
Posted by Alex Schuster on Thu, 05 Nov 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray wrote:

> So you can always make a binary mask from your greyscale image,
> and use that to define your regions. The most obvious way is to
> manually set pixel values using the output of a WHERE call, but often
> it is possible, and faster, to use a array comparison like:
>
> mask = image > threshold_val

Slight correction: mask = image gt threshold_val

Alex
--
Alex Schuster Wonko@weird.cologne.de PGP Key available

alex@pet.mpin-koeln.mpg.de

Subject: Re: Label_region and Erosion
Posted by Struan Gray on Fri, 06 Nov 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Alex Schuster, alex@rosa.mpin-koeln.mpg.de writes:
>
> Struan Gray wrote:
>>
>> mask = image > threshold_val
>
> Slight correction: mask = image gt threshold_val

Sigh. I made this mistake in my very first programming class and have been making it consistently ever since. Thanks for pointing it out.

While I'm here, another thing I like to do when using masks is to display the mask as a colour cast on the original image by creating an RGB image and zeroing one of the channels wherever the mask is active:

```
image = byte(dist(250))
mask = image gt 100
rgb_image = bytarr(3,250,250)
rgb_image[0,*,*] = image*(mask eq 0)
rgb_image[1,*,*] = image
rgb_iamge[2,*,*] = image
tv, image, /true
```

This has the effect of changing every pixel where the mask is active from a greyscale to cyanscale value. If you are running in 8 bit colour you'll have to use COLOR_QUAN to construct a custom colour table before using tv. Zapping the a different channel shades the mask with the appropriate complementary colour.

A final tip: if, like me, you often end up creating masks which cannot easily be created with a simple global selection criteria, it is often easiest to export the image as a TIFF file, use Photoshop's excellent selection tools to create a mask, and then load it back into IDL.

Struan