
Subject: Best methods for finding a threshold edge, like a coastline?

Posted by [kagoldberg](#) on Sat, 03 Aug 2013 18:13:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

In a 2D image, I'm looking for the best way (accurate, fast) to find the contours of an edge, defined by a threshold along a rough slope. I've come up with three ways to do it, but they give different results.

Imagine a 2D image with a slope in the x direction, and some roughness, like a north-south coastline. Define a threshold value, and use interpolation to find the contours of the edge at every vertical position. Ideally, I'd like to have a continuous value in the x direction, and evenly spaced values in y (i.e. one x position per row).

****Method 1, use INTERPOL on every line. Like this.**

```
for i=0,Ny-1 do $  
  edge[i] = interpol(x, img[:,i], threshold)
```

****Method 2, REBIN the array, stretching in x and then binarize the thresholding (1=above, 0=below) for quick counting. Like this. The result is still discrete, but the steps are as fine as you want to achieve with the stretching.**

```
img1 = rebin(img, 1000, N) GT threshold ;--- stretch  
Edge = total(img1, 1)/stretch_factor ;--- this is a 1D array that counts the points above  
threshold
```

****Method 3, use the CONTOUR procedure to calculate the 'path' of the edge.**

```
contour, img, levels=threshold, /PATH_DATA_COORDS, PATH_XY=xy, CLOSED=0  
;--- now xy is a 2D array, and xy[0,*] contains the edge positions, one per line.
```

Discussion

Method 1 is a for-loop solution, so we know it's not optimized.

Method 2 can be fast, but we have to choose the degree of precision/time.

Method 3 is lightning fast (4-6x faster!) but the results are inconsistent with 1 and 2. Could this be because contour is applying a non-linear interpolation?

Does anyone have any insight to share on this?
