
Subject: Re: Determining circularity

Posted by [Vince Hradil](#) on Fri, 01 Jun 2001 16:22:16 GMT

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

May be simpler to use this definition of circularity: the variance of the boundary pixels distance to the centroid.

- 1- compute centroid, (xc,yc): $xc = \text{sum}(x)/\text{area}$
- 2- find the boundary pixels
- 3- calculate the mean distance from the centroid to the the boundary pixels and the variance
- 4- circularity = variance/mean (circles are 0)

"Craig Markwardt" <craigmnet@cow.physics.wisc.edu> wrote in message news:on8zjkeuhj.fsf@cow.physics.wisc.edu...

>

> Todd Clements <mole6e23@hotmail.com> writes:

>

>> Hi all...

>>

>> I was wondering if anyone knew of routines to determine the circularity of an image. We have images from our detector and for optimal calibration, they need to be as circular as possible, and it's difficult to tell by eye when it's close but not quite there.

>>

>> Generally these images have fairly well defined edges, although there is definitely some gray area as to what consistutes the edge, which makes it difficult to determine by eye sometimes.

>>

>> Basically, I guess it comes down to fitting the edge of the image to an ellipse, but there has to be some determination of what the edge is as well.

>

> Possible solution:

>

- > 1. Edge filter with ROBERTSON (or perhaps SOBEL)
- > 2. Extract X- and Y-position of edge points (use a threshold)
- > 3. Fit to an ellipse using MPFITELLIPSE

>

> The last one is from my web page, and can be used for fitting an ellipse to a set of scattered points. It's not theoretically perfect but is good for rough calcs. Of course you need MPFIT as well. Here is a sample go-round. First I construct some fake data and then perform the steps above.

>

> ;; Construct a filled circle with a radius of 6

> x = findgen(101)*0.2 - 10. & y = x

> xx = x # (y *0 + 1) ;; Construct X, Y, and R (radius) values

```

> yy = (x*0+1) # y
> rr = sqrt(xx^2 + yy^2)
> wh = where(rr LT 6.) ;; Fill the circle
> im = xx*0
> im(wh) = 1
>
> ;; Step 1. Extract edge-filtered image
> edge = roberts(im)
>
> ;; Step 2. Extract edge points using threshold value (value of 2 here)
> wh = where(edge EQ 2)
> xim = xx(wh) & yim = yy(wh)
>
> ;; Step 3. Fit the image
> print, mpfitellipse(xim, yim)
> ... results are ...
>      5.96942    5.96942   -0.100046   -0.100019    0.00000
>      XSEMI      YSEMI    XCENTER    YCENTER    ROTATION = 0
>
> I make it look easy. The real trick is to find the right threshold to
> select the points from the data, and filtering out any other noise
> which will surely screw up the edge enhancement.
>
> Craig
>
> -----
> Craig B. Markwardt, Ph.D.      EMAIL:  craigmnet@cow.physics.wisc.edu
> Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
> -----

```
