Subject: Re: find the maximum diameter of an object in an image
Posted by Karl Schultz on Tue, 11 Oct 2005 00:58:09 GMT
View Forum Message <> Reply to Message

On Mon, 10 Oct 2005 13:03:16 -0700, andrew.cool wrote:

```
>
> Jeff N. wrote:
>> Hi folks,
>>
>> I have some CT data (of meteorites, not people) that I'm extracting
>> slices of.  The actual images I'm taking slices of are images where
>> grains in the image have been set to 1, and everything else set to
>> 0....so these are binary images.  What I want to do is take a slice out
>> of the CT volume, find the grains in the slice (which I'm doing with
>> David Fanning's FIND_BOUNDARY() function), and then find the longest
>> straight line that you can draw through the object (that goes from one
>> boundary, through the center point, and then to the opposite boundary),
>> which is its maximum diameter.  FIND_BOUNDARY() gives me both the
>> outline of the object and the center point, so I have that to work
>> with.  It also gives me perimeter and area, so I'm wondering if there
>> isn't some kind of geometry trick that I haven't thought of that will
>> get me what I need.  If I do a principal components analysis on the
>> boundary coordinates, wouldn't that be the maximum diameter?
>>
>> If anyone can point me in the right direction, or has some code they'd
>> like to share, I'll be very grateful.
>>
>> Thanks,
>> Jeff
>
> Hi Jeff,
>
> It's 0500 here, I've been up since 0330 with a bad sinus headache, and
> I'm no math-head, but I'll have a go anyway...
>
> I assume that you have the coordinates for every point on the
> perimeter?
>
> for every point do
>   calculate dist to centre
>   calculate angle to centre (0..359 deg say)
>   store dist in an array data[angle]
> end
>
> The diameter at say 45deg is then data[44] + data[44+180]
> (avoid wrapping beyond 359 of course)
>
```

> I guess you only need to check from 0..179 anyway, half way around
> the perimeter?
>
> Should be easy to pull out the maximum in a one liner from there.
>
> Not flash, not hires, but should work?
>
> Cheers,
>
> Andrew (who's on his third coffee...)

Not sure that the points on the perimeter are that evenly distributed.

I think that the brute force way would be to

```
For each point on the perimeter
  Compute line equation from that point to the center
  Find intersections of this line with the perimeter (you have one
  already)
  Hopefully, there are just two.  If so, compute their distance
  from each other.  If there are more than two, find the two that have the
  largest distance between them.  (You'd have to decide how to handle
  this case)
  Keep track of the two pts with the largest distance so far.
endfor
```

You might also try posting the general problem in comp.graphics.algorithms.
Those guys do this kind of thing in their sleep. Maybe they can offer an
approach that can be implemented in IDL nicely. But I can't think of one
at the moment.

I did think about some bitmap-level approaches, but the requirement that
the longest line go through the center, makes those ideas unusable.

Karl