

Hi,

JD Smith wrote:

- > Here's a good reference:
- >
- > <http://astronomy.swin.edu.au/~pbourke/geometry/insidepoly/>

Thanks, that helped, but David's pointer to IDL's own IDLanROI object is what I use now.

- > I believe Mark Hadfield has coded up one or more of these in his
- > Motley library, which at least vectorizes over the points to test.
- > Since all your polygons are the same length (4), it is possible to
- > simultaneously vectorize over both points and polygons (this would not
- > be true for arbitrary-sized polygons).

[snip]

- > However, similar to the "points inside a sphere" problem I just
- > commented on, you'll run out of memory fast using such a brute force
- > method.

I decided to apply some prior knowledge to the problem and limit the number of points I have to search, reducing the memory requirements significantly. The trick? The points are geolocation points from MODIS,

and are neatly arranged. What is more important: I have a set of 1x1 km

geolocations and another set of 5x5 km geolocations. Now I search the coarse array first, take those indices and make a generous estimate of the

indices I can expect in the fine array. I then create a subset from the fine

arrays and search those instead, searching on the order of 800 points (selecting up to 500), instead of searching 2.7 million. In the end the

speedup is a factor of ~ 22, and that is without vectorizing the code.

- > For a smallish number of points and polygons, this brute force method
- > will definitely be the fastest, thanks to the overhead of loops.
- > However, for larger sized problems, just like for the spherical search
- > case, you would do well to pre-reject points which definitely cannot

> be inside a given polygon.

I could probably speed things up by further pre-selection, but for now
I
think I have enough other things to do in each loop to get away with
what
I have now.

Thanks,

Maarten
