Subject: Re: match_2d
Posted by JDS on Wed, 22 Apr 2009 22:39:08 GMT
View Forum Message <> Reply to Message

>
> Aha... I've looked at it in gory detail, and it turns out that the
> routine implicitly assumes that the minimum value of both x2 and y2
> are 0. So you can get it to work if you do the following:

Aha! Thanks for the catch. That's what you get when you evaluate an
algorithm on artificial random coordinates ranging uniformly from
[0,1].

I've updated MATCH_2D at the address mentioned to handle this issue
explicitly, and also catch cases of matching points which fall just
slightly outside the bounding box of the search set. I've also added
a much-needed warning regarding using this Euclidean matching
algorithm for points on the sphere (e.g. star positions, lat/lon,
etc.):

```
; WARNING:
;
;       Distance is evaluated in a strict Euclidean sense.  For
;       points on a sphere, the distance between two given
;       coordinates is *not* the Euclidean distance.  As an extreme
;       example, consider two points very near the N. pole, but on
;       opposite sides (one due E, one due W).  For small patches,
;       this Euclidean assumption is approximately valid, and the
;       method works.  See NOTES above for a tip regarding obtaining
;       a (more) uniform match criterion on the sphere.
;;
```

Give this version a try. By the way, the value of MATCH_DISTANCE for
points which did *not* match is not meaningful.

JD