Subject: Re: yet another 2d matching question
Posted by JDS on Tue, 03 Aug 2010 20:53:03 GMT
View Forum Message <> Reply to Message

On Jul 31, 7:47 am, Gray <graylikethec...@gmail.com> wrote:
> On Jul 30, 6:23 pm, JD Smith <jdtsmith.nos...@yahoo.com> wrote:
>
>> Paulo spotted the issue. What determines whether a given point in the
>> search list "is not matched to a closer point"? Your 1-to-1 match
>> will be sensitive to the input ordering of the target list. The
>> intention of match_radius is to specify the maximum separation beneath
>> which all matches are "equally good". For example, the statistical
>> uncertainty in the position itself. Multiple matches would then imply
>> either is an equally good match. If you still wanted to do this (for
>> example if you are conducting a match for which sub-match_distance
>> separations are still meaningful), it will have to be a pre- or post-
>> processing step, since all matches are performed in parallel (which is
>> what gives MATCH_2D its speed).
>
>> JD
>
> Hmm... if all matches are equally good within the match_distance, then
> how does match_2d prioritize matches when there is more than one
> source in list b within the match radius of list a? This could happen
> when, for example, the positional accuracy of the sources in each list
> is low, but there is a possible shift (translation+rotation+etc.)
> between the members of the two lists which necessitates a larger match
> radius.

This is only true if your match_distance represents some positional uncertainty; i.e. it's not meaningful to say a given star is 50 milliarseconds closer when the precision with which you know your search list is 2 arcseconds. It does simply return the closest point within match_distance if there are multiple matches. You could certainly alter this to return *all* matches within match_radius, then use post-processing to enforce a 1-to-1 matching. IDL 8's new LIST type would make this much easier than before (when I've used REVERSE_INDEX style arrays for the same purpose).

JD