Subject: Re: position matching
Posted by Edd Edmondson on Tue, 15 May 2007 15:11:34 GMT
View Forum Message <> Reply to Message

cmancone@ufl.edu wrote:
> Hi everyone,
> A common task I have to do is take two lists of stars with x & y
> positions and match up the closest stars within a certain radius (so
> that each star has at most one match, that one being the best match).
> A long time ago I wrote some code to do this that gets the job done,
> but probably not in the fastest way. It just uses a for loop over one
> of the lists and uses a where to search for the closest star to each
> star on the other list. Most of the time this is more than adequate,
> but anytime my star lists get around 10000-20000 stars each (which
> happens on a not-so irregular basis) the program turns into quite a
> beast and takes its sweet time (i.e. a minute or two). Granted, this
> isn't exactly research-stopping time delays, but I'm sure that with a
> well thought-out algorithm, the execution time could be pulled down to
> a handful of seconds. The problem is, I have yet to come up with a
> well thought-out algorithm. I'm sure I'm not the only one who has run
> into this, so I was hoping there might be someone else out there that
> has dealt with the same thing, and knows a better way.

I think you were right in your followup post to the other replier that DTs will at best be tricky to use thanks to your dual lists. I do this fairly often but not in IDL (in Perl in fact, and end up doing some quite ugly things as a result).

Since you need a search within a given radius (from your other post) I'd first sort the longer list and use a binary search to get the subset of stars within that radius in one coordinate. Then search through that much smaller subset using one of the fast but memory-hungry techniques David F has on his pages. By that point you should have narrowed things down enough to not have to consume vast amounts of memory for the job.

--
Edd