
Subject: Re: match_2d

Posted by [Jeremy Bailin](#) on Sat, 09 May 2009 05:02:40 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Apr 29, 8:00 pm, Jeremy Bailin <astroco...@gmail.com> wrote:

> On Apr 29, 7:30 pm, Jeremy Bailin <astroco...@gmail.com> wrote:

>

>

>

>> On Apr 29, 4:29 pm, JDS <jdtsmith.nos...@yahoo.com> wrote:

>

>>> On Apr 28, 10:44 pm, Jeremy Bailin <astroco...@gmail.com> wrote:

>

>>>> On Apr 27, 3:06 pm, JDS <jdtsmith.nos...@yahoo.com> wrote:

>

>>>> > > I'm pretty sure there's a HIST_ND-based algorithm of doing this

>>>> > > similar toMATCH_2Dbut taking spherical trig into account, but I

>>>> > > don't have the patience to figure it out.

>

>>>> > That would be challenging for the whole sphere, since histogram can

>>>> > only evaluate monotonic coordinate fields. You can always first remap

>>>> > your coordinates using some projection which puts the ill-behaved

>>>> > parts (nominally, the poles) far away, and preserves distance

>>>> > locally. For example, if you have a small field (a degree or so) near

>>>> > the pole, this would be a nice way of solving the converging longitude

>>>> > lines issues. But generally? Sounds tough.

>

>>>> > JD

>

>>>> How about if it was done in 3D? Instead of 2D angular coordinates, use

>>>> the 3D coordinates of the relevant points on the surface of a unit

>>>> sphere, and then use HIST_ND to determine which 3D bin the points are

>>>> in and build the algorithm analogously toMATCH_2D?

>

>>>> The main problem I see is that, for small bin sizes (ie. small desired

>>>> angular separations), there's a lot of wasted memory storing the

>>>> histogram in locations that don't lie on the surface of the sphere and

>>>> therefore are necessarily zero. But maybe there's a way of enumerating

>>>> the bins that do contain part of the surface - if so, then you could

>>>> use that enumeration to map the 3D positions into a simple number that

>>>> you can run HISTOGRAM on.

>

>>> I thought of that and rejected it for the reason you mention. The

>>> vast majority of memory would be devoted to empty volume, and as the

>>> resolution grew, the fraction of wasted memory would grow as well.

>>> The mapping you describe to do away with the empty space is equivalent

>>> to spherical projection, for which there is no unique mapping for the

>>> whole sphere. One possibility would be to project iteratively,

```

>>> forming low distortion projections over the sphere to push the poles
>>> off out of the way, matching against a subset of the data, rotate the
>>> projection, repeat. Some heuristic for deciding which projection, how
>>> large, and where to center it, would be needed.
>
>>> At some point, it would become simpler to use pattern matching via
>>> Delauney triangulation or other patterns formed from the target list.
>
>>> JD
>
>> One idea to at least limit the amount of wasted memory is to use a
>> larger grid spacing than absolutely required - thematch_2dalgorithm
>> needs grid spacings that are no smaller than 2x the desired
>> separation, but I think it should work fine if the spacing is
>> larger... the drawback would be that you need to calculate the correct
>> angular distance for a larger number of particles than strictly
>> necessary, but that would be a worthwhile tradeoff at some point.
>
>> But I think that there should be an enumeration mapping solution.
>> There certainly exists an enumeration for any grid size... I can
>> generate one by placing points randomly on the surface of the sphere,
>> calculating the 3D histogram, and then getting a list of which cells
>> contain points - the enumeration is then simply be the ordinal of the
>> cell within the list. But that's a stupid solution in this case,
>> because the entire point is to avoid calculating the full 3D
>> histogram. Still, the fact that an enumeration is possible makes me
>> think that it should be possible to generate it from first principles
>> rather than empirically. :-)=
>
>> -Jeremy.
>
> Now that I think about it, you can use the random points directly to
> get the grid cells without going via the histogram - just calculate
> the bin each one would fall into, and UNIQ them. You just need to
> generate enough points that you're virtually guaranteed to get a hit
> in each bin. You could guarantee it by also including every neighbour
> of any grid cell that contains a point - that way even a cell that
> only has a sliver of surface pass through it and therefore does not
> contain a point will get into the list.
>
> Hmm... okay, I'm going to code that up and see if it works.
>
> -Jeremy.

```

Here is my best version so far:

http://www.physics.mcmaster.ca/~bailinj/idl/withinsphrad_vec 3d.pro

It borrows heavily from JD's MATCH_2D, but does it in 3D on the surface of the sphere. It sidesteps all of the "wasted space" and "enumerating the surface of the sphere" issues by enumerating cells that contain points in either the input or target coordinate lists - any other cells on the sphere are irrelevant, since all we need to know about them is that they don't contain any points. So I just make a list of all cells that contain a point in either coordinate list and use the index into that list as my mapping. The histograms then become very efficient! And it doesn't waste extra space generating the enumeration, since we need to calculate those cell locations anyway.

In my simple tests, this does a factor of 100 better than my WITHINSPHRAD_VEC, and about a factor of 10 better than a for loop using the scalar WITHINSPHRAD, but ymmv depending on the number of input and target coordinates, as well as the angular matching radius.

-Jeremy.
