Subject: Re: IDLy approach to splatting points on a grid?
Posted by JD Smith on Mon, 27 Nov 2006 18:05:38 GMT
View Forum Message <> Reply to Message

On Thu, 23 Nov 2006 03:46:43 -0800, Jonathan wrote:

> I've been reading fairly carefully this group recently and many of the
> resources suggested, but I'm unable to figure out a loopless (or fast) way
> to do the following:
>
> I have a collection of points in space, each contributing some value
> locally over some region (how big a region depends on the point), and I'm
> trying to produce a 1/2/3d grid of the sum of their contributions.
>   In general, the points contribute over a small number of grid cells
> compared to the whole grid. (For those who are interested, the
> application here is analysis of Smoothed Particle Hydrodynamics
> simulations.)   So for some field rho I want to calculate rho[i,j] =
> \sum_k {W_k(i,j) m_k} where the summation is over particles and there's a
> particle-dependant kernel W which for the time being is Gaussian, although
> later it'd likely be something tabulated.
>
<snip>
> Is this just hopeless? Do I do this in a low-level language and spit out
> the data in some format IDL can read for vis/analysis?

Shouldn't be hopeless. This is highly akin to the "Drizzle" method
often used for linear restoration of dithered astronomical images,
except each of your points has a varying region of influence which
doesn't depend on geometry (well, normal non-SPH geometry anyway).
Search the archive for information on the method. The reigning champ
(and one I still use all the time) is a dual histogram method. The
first histogram finds all contributing "input" pixels in a given
output pixel. The second one finds all output pixels with 1, 2, 3, 4,
.... up to n contributing input pixels. Then you can loop efficiently
over bin count and operate on large chunks of data at a time (to avoid
feeling the loop penalty).

See also:

 http://www.dfanning.com/code_tips/drizzling.html

For you (actually for everyone doing this type of thing in IDL), the
problem will be efficiently calculating the "region of influence" of
all your input particles. Since it is highly variable depending on
particle position and size, it's somewhat difficult to vectorize
(though perhaps you have "bins" of influence radius and can attack it
that way). If you do write a DLM, the clipping operation is the best area
to concentrate effort.

JD