Subject: Re: Drizzling Algorithms
Posted by JD Smith on Fri, 26 Sep 2003 17:36:40 GMT
View Forum Message <> Reply to Message

On Fri, 26 Sep 2003 08:26:48 -0700, Wayne Landsman wrote:

>> The second article is another dazzling one anchored by JD about
>> "drizzling" algorithms. No, we are not talking about chocolate
>> sprinkles on ice-cream sundaes. Drizzling algorithms are used in the
>> reconstruction or warping of images from undersampled or dithered data.
>> (I don't know, you'll have to read the article!) It's another can't
>> miss hit for you Histogram fans.
>>
>> http://www.dfanning.com/code_tips/drizzling.html
>>
>>
> Well, we are all thankful for David's late night inspirations (or sleep
> difficulties), and I feel very curmudgeonly to have to bring up a
> caveat. But the above article might lead people to think that there are
> drizzle or other flux-conserving algorithms available in IDL, and I am
> not aware that there are any. David's article is really about array
> decimation, which would be just one step in writing a vectorized drizzle
> code (as described in
> http://www-int.stsci.edu/~fruchter/dither/drizzle.html) My own attempts
> at writing a vectorized drizzle code floundered on determining partial
> pixel weights.
>
> I'm pretty sure now that drizzle code is best written in C and linked to
> IDL. Or perhaps I need to try out the V6.0 IDL-Java bridge with Tom
> McGlynn's software at
> http://skyview.gsfc.nasa.gov/polysamp/PolySamp.java

I tend to agree with Wayne, but I do have a Sutherland-Hodgeman polygon clipper written in pure IDL which, together with any one of the aggregation techniques described in the article, could constitute a full drizzle algorithm. I'm using it for a different, but similar, purpose. If enough people bug me I'll polish it up and put it out for the enjoyment of all.

I did find that it was fairly slow for even my (densely dithered) 128x128 arrays, and ended up writing a replacement clipper in C. The most fun thing about it: it uses MAKE_DLL to compile the C version of the code to a shared library auto-magically, and, if this fails, just falls back on the slow, but equivalent, IDL version, with no one the wiser. It's still not nearly as fast as coding the whole problem in C, thanks in no small part to the CALL_EXTERNAL overhead, and the need to make a round trip to the clipper so frequently, but it's suitable

for my needs. I suspect drizzling 10's of 2kx2k images would get a bit ugly though.

By the way, this is one problem which cannot, at least insofar as about 2 concerted weeks of deep thought on the problem revealed, be vectorized using the many IDL tricks we discuss here. It's surprising when you run up against these problems, but they do exist. If RSI is listening, they should bone up on Sutherland-Hodgeman or other, fancier clippers, and implement a vectorized version in IDL. I should be able to give a large list of polygons, and have them all clipped to a grid of a specified size, with optional return of the grid pixel(s) and area(s) each polygon clipped to. Somebody would no doubt write a nice, fast DRIZZLE algorithm, maybe with a GUI interface, and that would be one less reason to fire up IRAF. Another very useful routine: given a (long) input list of polygons, compute the internal overlap with areas.

JD
