Subject: Re: Chunk Array Decimation
Posted by Wayne Landsman on Tue, 01 Oct 2002 21:34:21 GMT

View Forum Message <> Reply to Message

> Of course, anyone familiar at all with histogram() would realize
> there's a better route when many indices are repeated:
>
> ```
>   mx=max(inds)
>   vec3=fltarr(mx+1)
>   h=histogram(inds,reverse_indices=ri,OMIN=om)
>   for j=0L,n_elements(h)-1 do if ri[j+1] gt ri[j] then $
>     vec3[j+om]=total(data[ri[ri[j]:ri[j+1]-1]])
> ```
>
> This taps into the ever-so useful reverse indices vector to pick out
> those elements of data which fall in each "bin" of the index
> histogram. Notice I'm using OMIN to save time in case the minimum
> index is greater than 0. This is much faster than the where() method,
> and can be a factor of 2 or 3 faster than the literal loop approach,
> if indices are repeated at least a few times on average (a few drops
> in each histogram bin). If indices are never repeated, or especially
> if many indices are skipped (a *sparse* set), the literal loop method
> can be much faster than histogram.

The problem that discussed by JD is actually a very practical one, that can be used in "drizzling" algorithms (e.g. http://www-int.stsci.edu/~fruchter/dither/drizzle.html)   This a method of combining or warping images that preserves flux -- every pixel in the input image is equally represented in the output image.
Instead of starting with an input pixel and mapping to an output image (e.g. as with POLY_2D) , one starts with an output pixel and determines which input pixels get mapped into it.   The flux conservation property is one very dear to astronomers, and for which there are no existing IDL tools.

My solution to the problem combined the REVERSE_INDICiES aproach of JD, with the "accumlate based on the index" approach.   For the drizzle problem, one is probably only going to sum at most 3-4 pixels together, so it makes sense to loop over the number of distinct histogram values (i.e. loop only 3-4 times).

My solution is below, but I have to admit that I haven't looked at it for a while.

--Wayne

P.S. I never finished the drizzle algorithm, because I couldn't figure out a quick way to compute partial pixel overlaps in IDL...

```
pro fdrizzle, vector, index, values
;+
; NAME:
;    FDRIZZLE
; PURPOSE:
;    Add values to an array at specified indicies.     The basic usage is

;       FDRIZZLE, vector, index, values
;    where INDEX and VALUES should have same number of elements.   If
there are
;       no duplicates in INDEX then FDRIZZLE simply performs the
assignment
;    VECTOR[INDEX] = VECTOR[INDEX] + VALUES
;       But if INDEX contains repeated elements then the corresponding
VALUES
;       will be summed together.
;
; METHOD:
;    Use the REVERSE_ELEMENTS keyword of histogram to determine the
repeated
;    values in INDEX and vector sums these together.
;-

 h = histogram(index,reverse = ri,min=0,max=N_elements(vector)-1)

;Add locations with at least one pixel
 gmax = max(h)         ;Highest number of duplicate indicies

 for i=1,gmax do begin
       g = where(h GE i, Ng)
     if Ng GT 0 then  vector[g] = vector[g] + values[ri[ ri[g]+i-1]]
 endfor

 end
```