Subject: Re: Chunk Array Decimation
Posted by Craig Markwardt on Thu, 03 Oct 2002 08:58:13 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@as.arizona.edu> writes:

> On Tue, 01 Oct 2002 14:34:21 -0700, Wayne Landsman wrote:
[...]
>>
>> 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.
>>
>
>> 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
>
> That's a very interesting approach, Wayne. People who need to understand
> the reverse indices vector would do well to study this one. I put it
> into the same terms as my problem for testing:
>
> mx=max(inds)
> vec5=fltarr(mx+1)
> h=histogram(inds,REVERSE_INDICES=ri,omin=om)
> gmax = max(h) ;Highest number of duplicate indicies
> for j=1,gmax do begin
> g = where(h GE j, Ng)
> if Ng GT 0 then vec5[om+g] = vec5[om+g] + data[ri[ri[g]+j-1]]
> endfor
>
> I was interested to see that your method beat mine for normal
> densities by about a factor of 2! This should provide some cannon
> fodder for Craig in his loop-anti-defamation campaign: keep loops
> small, and they're not bad. The only change I added was using OMIN as

> opposed to fixing MIN=0, but that shouldn't account for much if any
> improvement.
>
> However, one thing still bothered me about the your method: even
> though the loop through the bin depth is small (e.g. maybe up to 5-10
> for DRIZZLE-type cases), you're using WHERE to search a potentially
> very large histogram array linearly each time. What's the solution?
> Why, just use another histogram to sort the histogram into bins of
> repeat count, of course. Now this is a true histogram of a histogram.
[...]

Here I come late to the game again. This topic actually came up
before by Liam Gumley in September 2000.

My solution then was the following loop (expressed in today's variable
names):

```
  n = n_elements(vec)
  hh = histogram(inds, min=0, max=n-1, reverse=rr)
  wh = where(hh GT 0) & mx = max(hh(wh), min=mn)
  for i = mn, mx do begin
    wh = wh(where(hh(wh) GE i, ct))          ;; Get IND cells with GE i entries
    vec(wh) = vec(wh) + data(rr(rr(wh)+i-1)) ;; Add into the total
  endfor
```

This is essentially the same as Wayne's FDRIZZLE routine, with the
difference that the WHERE-generated index array is slowly whittled
away by repeated thinning. Thus, the WHERE() function gets faster and
faster as the loop proceeds. At the time, I was crowned the victor by
Pavel :-), but I don't know how I will do against this round of
competitors.

However, all of these optimized techniques that Wayne and JD have
proposed in the end game here, including mine, suffer if the dynamic
range of the histogram is very large. For example, if the input array
contains a million 1s, then any of the proposed loops will still take
1 million iterations. There are even ways around that, which reminds
me to finish an old routine named CMHISTOGRAM...

Craig

PS. In my case, NO, I do not have to check the count returned by
WHERE, because by the construction of the loop, there is always
guaranteed to be at least one element.

--
-- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu