Subject: Re: Maximum value array resampling
Posted by JD Smith on Mon, 15 Aug 2005 18:13:10 GMT
View Forum Message <> Reply to Message

>
> Just to clarify things, your 8/8/05 routine above is essentially
> identical to the revised routine I posted on 8/5/05 (see below), and I was
> referring to this new routine when I said it was faster than what you had
> posted. I had also posted a lame rebinning approach to the problem
> previously, and I think that is the one you used when you found such poor
> performance.

Sorry about that Dick... looks very similar (except of course for
reform vs #). Here's a version which can do max or min, and any box
size (not just 2x2). Keep in mind that for arrays which are not
multiples of the box size, the edges will be lost. Another
interesting question is how to do a sliding box max/min efficiently,
ala MEDIAN.

JD

```
;; BOX_MAX: Compute local maximum (or minimum. with /MIN) box
;; downsampling, with box size boxx, boxy (default 2,2). Pre-computed
;; INDS may be passed.
;;   JD Smith (c) 2005.
function box_max,array,boxx,boxy,INDS=inds, MIN=min
  if n_elements(boxx) eq 0 then boxx=2
  if n_elements(boxy) eq 0 then boxy=2
  min=keyword_set(min)
  d=size(array,/DIMENSIONS)
  nx=d[0] & ny=d[1]

  if n_elements(inds) eq 0 then begin
     nx_out=nx/boxx & ny_out=ny/boxy
     inds=rebin(lindgen(nx_out)*boxx,nx_out,ny_out,/SAMPLE)+ $
          rebin(transpose(lindgen(ny_out)*boxy*nx),nx_out,ny_out,/SAMP LE)
  endif

  ret=array[inds]
  for i=0L,boxx-1L do begin
     for j=0L,boxy-1L do begin
        if i eq 0 && j eq 0 then continue
        if min then ret <= array[inds+i+j*nx] else ret >= array[inds+i+j*nx]
     endfor
  endfor

  return,ret
end
```