Subject: Re: Local max filter
Posted by rkj on Tue, 21 Aug 2001 21:55:11 GMT
View Forum Message <> Reply to Message

Thanks!

I was afraid any potential solution would involve "histogram" ;-)

As an aside, would this work in Matlab? Some people around here
just won't make the switch . . .

Kyle

Craig Markwardt (craigmnet@cow.physics.wisc.edu) wrote:

: rkj@dukebar.crml.uab.edu (R. Kyle Justice) writes:
: > I am trying to implement a local max filter
: > without loops. Has this been done?
: >
: > (Given an array and a filter width, return an
: > array containing the array value if it is
: > a local max, 0 if not)
: >
: > For instance,
: >
: > 3 4 7 2 6 4 9 8 3
: >
: > would be
: >
: > 0 0 7 0 0 0 9 0 0
: >
: > for a width of 5.

: JD and I had a contest doing this kind of thing -- finding maxima -- a
: year or so ago. Of course I popped his socks off, but he will tell
: you a different story :-)

: Perhaps the easiest way to do this is with a bunch of vector compares.

```
: arr = [3, 4, 7, 2, 6, 4, 9, 8, 3]  ;; Initial data
: arr2 = arr(2:*)                    ;; Center points
: b = (arr2 GE arr(0:*)) AND (arr2 GE arr(1:*)) AND $
:     (arr2 GE arr(3:*)) AND (arr2 GE arr(4:*))  ;; Compare against neighbors
: result = [0, 0, b*arr2, 0, 0]  ;; Replace boundaries
```

: The trick is that you are comparing arr(2:*) to each of its neighbors.

: I am using the little trick I've published a couple of times, which is
: that when two vectors of unequal lengths are compared, the longer one
: is truncated to the other one's size. Otherwise you need to do "arr2
: GE arr(0:n_elements(arr)-2)" and so on.

: If you really need variable widths then the above can be formulated
: into a loop over the width. This is not a hurtful loop because the
: arrays are still compared vectorially. Try this function out:

```
: function locmax, arr, width
:   if n_elements(arr) LT width then message, 'ERROR: arr is too small'
:   if (width MOD 2) EQ 0 then       message, 'ERROR: width must be odd'
:   ic = (width-1)/2
:   arrc = arr(ic:*)
:   b = bytarr(n_elements(arr)-width+1) + 1b
:   for i = 1, ic do $
:     b = b AND (arrc GE arr(ic-i:*)) AND (arrc GE arr(ic+i:*))
:   return, [arr(0:ic-1)*0, b*arrc, arr(0:ic-1)*0]
: end
```

: I play the same tricks, plus a few tricks to preserve the type of the
: original array.

: Craig

: --
: -- --------------
: Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
: Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
: -- --------------