Subject: Re: Median filter the hard way
Posted by JD Smith on Fri, 17 Oct 2003 22:22:15 GMT
View Forum Message <> Reply to Message

On Fri, 17 Oct 2003 14:34:30 -0700, Dick Jackson wrote:

> Hi Peter,
>
> "Peter Payzant" <pce@accesswave.ca.nospam> wrote in message
> news:WvGjb.92982$PD3.4887868@nnrp1.uunet.ca...
>> Hello, all-
>>
>> This is my first posting to the group.
>
> Welcome aboard!
>
>> He is applying a median filter to a 2-dimensional image. [but...] If
>> there less than 7 good values in the 3 x 3 array, he discards the
>> original pixel.
>
> ["loopy" :-) code snipped]
>
>> Obviously, the nested loops are the source of the problem. Is there
> any
>> other way to accomplish this, in a more IDL-esque way?
>
> As Mike mentions, the Median filter is part of it, but your "7 or
> better" requirement makes it a bit more interesting.
>
> Here's an array, a:
>
> IDL> a=Float(Byte(RandomU(seed,7,7)*10)) IDL> a[2:4,2:4]=!values.F_nan
> IDL> print,a,Format='(7F4.1)'
> 8.0 1.0 7.0 2.0 2.0 5.0 4.0
> 5.0 7.0 8.0 8.0 3.0 3.0 4.0
> 3.0 6.0 NaN NaN NaN 9.0 0.0
> 5.0 0.0 NaN NaN NaN 4.0 9.0
> 6.0 5.0 NaN NaN NaN 0.0 0.0
> 9.0 5.0 3.0 4.0 5.0 3.0 4.0
> 7.0 5.0 9.0 7.0 4.0 5.0 3.0
>
> Median will give a result with even one actual number in the 3x3
> neighborhood:
>
> IDL> print,Median(a,3),Format='(7F4.1)'
> 8.0 1.0 7.0 2.0 2.0 5.0 4.0
> 5.0 7.0 7.0 7.0 3.0 4.0 4.0
> 3.0 5.0 7.0 8.0 4.0 4.0 0.0

> 5.0 5.0 5.0 NaN 4.0 4.0 9.0
> 6.0 5.0 4.0 4.0 4.0 4.0 0.0
> 9.0 6.0 5.0 5.0 4.0 4.0 4.0
> 7.0 5.0 9.0 7.0 4.0 5.0 3.0
>
> Let's use the Finite function to figure which other ones to knock out to
> NaN:
>
> IDL> print,Finite(a)
> 1 1 1 1 1 1 1
> 1 1 1 1 1 1 1
> 1 1 0 0 0 1 1
> 1 1 0 0 0 1 1
> 1 1 0 0 0 1 1
> 1 1 1 1 1 1 1
> 1 1 1 1 1 1 1
>
> The Convol function can be used to count up neighborhoods. If you need
> better counting around the edge, you could pad the array before calling
> Convol.
>
> IDL> print,Convol(Finite(a),Replicate(1B,3,3))
> 0 0 0 0 0 0 0
> 0 8 7 6 7 8 0
> 0 7 5 3 5 7 0
> 0 6 3 0 3 6 0
> 0 7 5 3 5 7 0
> 0 8 7 6 7 8 0
> 0 0 0 0 0 0 0
>

Looks good, Dick. CONVOL's a bit heavy-handed for just counting: I'd use smooth instead:

IDL> print,smooth(finite(a)*9,3,/EDGE_TRUNCATE)
```
      9       9       9       9       9       9       9
      9       8       7       6       7       8       9
      9       7       5       3       5       7       9
      9       6       3       0       3       6       9
      9       7       5       3       5       7       9
      9       8       7       6       7       8       9
      9       9       9       9       9       9       9
```

Notice it treats edges better (as far as this problem is concerned), and should definitely be faster.

JD