Subject: Re: Median filter the hard way
Posted by mmiller3 on Fri, 17 Oct 2003 01:21:12 GMT
View Forum Message <> Reply to Message

>>>> > "Peter" == Peter Payzant <pce@accesswave.ca.nospam> writes:

> He is applying a median filter to a 2-dimensional image.
[...]
> Obviously, the nested loops are the source of the
> problem. Is there any other way to accomplish this, in a
> more IDL-esque way?

Try the median function.  (?median)

Mike

Subject: Re: Median filter the hard way
Posted by Dick Jackson on Fri, 17 Oct 2003 21:34:30 GMT
View Forum Message <> Reply to Message

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
```

OK, here goes:

```
IDL> m3=Median(a,3)
IDL> nGood = Convol(Finite(a), Replicate(1B,3,3))
IDL> m3[Where(nGood LT 7)] = !Values.F_NaN
IDL> print,m3,Format='(7F4.1)'
 NaN NaN NaN NaN NaN NaN NaN
 NaN 7.0 7.0 NaN 3.0 4.0 NaN
 NaN 5.0 NaN NaN NaN 4.0 NaN
 NaN NaN NaN NaN NaN NaN NaN
 NaN 5.0 NaN NaN NaN 4.0 NaN
 NaN 6.0 5.0 NaN 4.0 4.0 NaN
 NaN NaN NaN NaN NaN NaN NaN
```

Hope this helps!

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

---

## 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

---

Subject: Re: Median filter the hard way
Posted by Peter Payzant on Fri, 17 Oct 2003 22:50:54 GMT
View Forum Message <> Reply to Message

Dick and JD-

Thanks so much for your helpful suggestions. I'm looking forward to trying
them out on Monday. There is definitely an "IDL way" of doing things which
works well, and a "brute force and ignorance" way which works as well, but a
lot more slowly! It's always instructive to see things done right.

Regards

Peter Payzant

---

Subject: Re: Median filter the hard way
Posted by the_cacc on Sat, 18 Oct 2003 06:08:04 GMT
View Forum Message <> Reply to Message

>  There is definitely an "IDL way" of doing things...

Oh yes.

(Tell him about histogram, tell him about histogram!!!)

---

Subject: Re: Median filter the hard way
Posted by Dick Jackson on Tue, 21 Oct 2003 22:52:28 GMT
View Forum Message <> Reply to Message

"JD Smith" <> wrote in message
news:pan.2003.10.17.22.22.14.73337.24537@as.arizona.edu...
> On Fri, 17 Oct 2003 14:34:30 -0700, Dick Jackson wrote:
>
>> Here's an array, a:
>>
>> IDL> a=Float(Byte(RandomU(seed,7,7)*10)) IDL>
a[2:4,2:4]=!values.F_nan
>
> [...]
>
>> 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))
>
> 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)
>
> [...]
>
> Notice it treats edges better (as far as this problem is concerned),

Granted, if edge pixels are not going to be dropped anyway for having
too few 'good' neighbors...

> and should definitely be faster.

A good guess, but in this case, I guess Convol can keep everything as
Byte type and it is indeed faster:

```
IDL> a=Float(Byte(RandomU(seed,2000,2000)*10))
IDL>  tstart&b=Convol(Finite(a),Replicate(1B,3,3))&treport
```

```
    0.510 seconds.
IDL>  tstart&b=Convol(Finite(a),Replicate(1B,3,3))&treport
    0.510 seconds.
IDL> help,b
B               BYTE      = Array[2000, 2000]
IDL>  tstart&b=smooth(finite(a)*9,3,/EDGE_TRUNCATE)&trepor t
    0.681 seconds.
IDL>  tstart&b=smooth(finite(a)*9,3,/EDGE_TRUNCATE)&trepor t
    0.661 seconds.
IDL> help,b
B               INT       = Array[2000, 2000]
```

You may want these for testing:

```
=====

PRO TStart, msg   ; Timer Start
                ; Save current time for use by TReport
COMMON Timer, t0
IF N_Elements(msg) NE 0 THEN Print, msg
t0 = SysTime(1)
END

PRO TReport, msg  ; Timer Report
                ; Print elapsed time since last TStart
COMMON Timer, t0
IF N_Elements(msg) EQ 0 THEN msg = ''
Print, Format='(A0, D10.3," seconds.")', msg, SysTime(1)-t0
END

=====
```

Cheers,
--
-Dick

Dick Jackson                    /            dick@d-jackson.com
D-Jackson Software Consulting /       http://www.d-jackson.com
Calgary, Alberta, Canada     / +1-403-242-7398 / Fax: 241-7392

---

Subject: Re: Median filter the hard way
Posted by JD Smith on Tue, 21 Oct 2003 23:29:54 GMT
View Forum Message <> Reply to Message

On Tue, 21 Oct 2003 15:52:28 -0700, Dick Jackson wrote:

> "JD Smith" <> wrote in message
> news:pan.2003.10.17.22.22.14.73337.24537@as.arizona.edu...
>> On Fri, 17 Oct 2003 14:34:30 -0700, Dick Jackson wrote:
>>
>>> Here's an array, a:
>>>
>>> IDL> a=Float(Byte(RandomU(seed,7,7)*10)) IDL>
> a[2:4,2:4]=!values.F_nan
>>
>> [...]
>>
>>> 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))
>>
>> 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)
>>
>> [...]
>>
>> Notice it treats edges better (as far as this problem is concerned),
>
> Granted, if edge pixels are not going to be dropped anyway for having
> too few 'good' neighbors...
>
>> and should definitely be faster.
>
> A good guess, but in this case, I guess Convol can keep everything as
> Byte type and it is indeed faster:

This is the key (byte type).  Change it to smooth(finite(a)*9b... and you
should see similar performance.  Obviously, in this case, we're limited
by something other than the details of the addition.

JD

---

Subject: Re: Median filter the hard way
Posted by Dick Jackson on Tue, 21 Oct 2003 23:51:38 GMT
View Forum Message <> Reply to Message

"JD Smith" <jdsmith@as.arizona.edu> wrote in message
news:pan.2003.10.21.23.29.45.108433.5869@as.arizona.edu...
> On Tue, 21 Oct 2003 15:52:28 -0700, Dick Jackson wrote:
>
>
>> "JD Smith" <> wrote in message
>> news:pan.2003.10.17.22.22.14.73337.24537@as.arizona.edu...
>>> On Fri, 17 Oct 2003 14:34:30 -0700, Dick Jackson wrote:
>>>
>>>> Here's an array, a:
>>>>
>>>> IDL> a=Float(Byte(RandomU(seed,7,7)*10)) IDL>
>> a[2:4,2:4]=!values.F_nan
>>>
>>> [...]
>>>
>>>> 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))
>>>
>>> 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)
>>>
>>> [...]
>>>
>>> Notice it treats edges better (as far as this problem is
concerned),
>>
>> Granted, if edge pixels are not going to be dropped anyway for
having
>> too few 'good' neighbors...
>>
>>> and should definitely be faster.
>>
>> A good guess, but in this case, I guess Convol can keep everything
as
>> Byte type and it is indeed faster:
>
> This is the key (byte type).  Change it to smooth(finite(a)*9b... and
you

>  should see similar performance.  Obviously, in this case, we're
limited
>  by something other than the details of the addition.

Argh, you're right. (I don't know why I thought Smooth wouldn't do
bytes, but it does!) I think you win on the edge-handling, nice one!

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392