Subject: I think my circular median filter is very inefficient
Posted by JRP on Sun, 03 Aug 2014 03:00:59 GMT
View Forum Message <> Reply to Message

Hi, I have written a circular median filter for removing noise from a noisy signal, which runs through a loop of radii (r=2 -> r=20) and then calculates a peak signal-noise ratio to determine which radius does well (this does not take long). I am by no means at all experienced in any kind of programming, so if anyone is able to offer me any assistance in reducing the time it would take me to do this it would be greatly appreciated! Here is the part of the median filter code:

```
for k=0,size3[1]-size2[1]-1 do begin
  for l=0,size3[2]-size2[2]-1 do begin
    for i=0, size2[1]-1 do begin
      for j=0, size2[2]-1 do begin

        holder[i,j] = (se[i,j]*padding[i+k,j+l])
        med = MEDIAN(holder)
        clean[k,l] = med

      endfor
    endfor

  endfor
endfor
```

Here, size2 is the size of the circular structuring element and size3 is the size of the padded noisy image. So basically I multiply the structuring element by the noisy image (which is padded), store in the values that fill the circle, then find the median and assign it to a "cleaned" image. Then the loop moves the structuring element 1 unit over...

At the moment, the program has done up to r=17, and has been running for 7.2 hours. I've been printing the time it takes for each radii to complete and it will take something like a further 5.5 hours to complete! :(

Cheers!

Subject: Re: I think my circular median filter is very inefficient
Posted by Fabzi on Sun, 03 Aug 2014 19:21:45 GMT
View Forum Message <> Reply to Message

Hi,

I am not sure to understand well what you try to do but what is sure is
that you are overwriting many times a new median value to the same place
in clean[k,l] while you are still in the i, j loop, thus making many
useless operations without writing their results.

I *think* you might want to move the block:

```
med = MEDIAN(holder)
clean[k,l] = med
```

out of the two inner loops and also replace it by:

```
clean[k,l] = MEDIAN(holder)
```

But this you should check by yourself

Fabien

Subject: Re: I think my circular median filter is very inefficient
Posted by on Sun, 03 Aug 2014 22:35:12 GMT
View Forum Message <> Reply to Message

Den söndagen den 3:e augusti 2014 kl. 05:00:59 UTC+2 skrev JRP:
> Hi, I have written a circular median filter for removing noise from a noisy signal, which runs through a loop of radii (r=2 -> r=20) and then calculates a peak signal-noise ratio to determine which radius does well (this does not take long). I am by no means at all experienced in any kind of programming, so if anyone is able to offer me any assistance in reducing the time it would take me to do this it would be greatly appreciated! Here is the part of the median filter code:
>
>
>
> for k=0,size3[1]-size2[1]-1 do begin
>
> for l=0,size3[2]-size2[2]-1 do begin
>
> for i=0, size2[1]-1 do begin
>
> for j=0, size2[2]-1 do begin
>
>
>
> holder[i,j] = (se[i,j]*padding[i+k,j+l])
>
> med = MEDIAN(holder)
>
> clean[k,l] = med
>
>
>
> endfor
>
> endfor

>
>
>
>   endfor
>
> endfor
>
>
>
> Here, size2 is the size of the circular structuring element and size3 is the size of the padded noisy image. So basically I multiply the structuring element by the noisy image (which is padded), store in the values that fill the circle, then find the median and assign it to a "cleaned" image. Then the loop moves the structuring element 1 unit over...
>
>
>
> At the moment, the program has done up to r=17, and has been running for 7.2 hours. I've been printing the time it takes for each radii to complete and it will take something like a further 5.5 hours to complete! :(
>
>
>
> Cheers!

Like Fabien, I think you should lose the inner two loops. They are equivalent to something like:

```
holder = (se*padding[k:k+size2[1]-1,l:l+size2[2]-1])
med = MEDIAN(holder)
clean[k,l] = med
```

But I wonder if your code really does calculate the medians you want, with or without this change. If the array "se" is your "structuring element", which is a binary mask defining the circular area over which you want to calculate the median, then the number you calculate will be biased toward zero by the elements that are zeroed by multiplication with the mask.

So if I understand what you are trying to do, you may actually want to write the whole thing as:

```
indx = where(se)
for k=0,size3[1]-size2[1]-1 do begin
  for l=0,size3[2]-size2[2]-1 do begin
    clean[k,l] = median( ( padding[k:k+size2[1]-1,l:l+size2[2]-1] )[indx] )
  endfor
endfor
```

Subject: Re: I think my circular median filter is very inefficient
Posted by JRP on Mon, 04 Aug 2014 03:15:41 GMT

On Monday, 4 August 2014 08:35:12 UTC+10, Mats Löfdahl wrote:
> Den söndagen den 3:e augusti 2014 kl. 05:00:59 UTC+2 skrev JRP:
>
>> Hi, I have written a circular median filter for removing noise from a noisy signal, which runs through a loop of radii (r=2 -> r=20) and then calculates a peak signal-noise ratio to determine which radius does well (this does not take long). I am by no means at all experienced in any kind of programming, so if anyone is able to offer me any assistance in reducing the time it would take me to do this it would be greatly appreciated! Here is the part of the median filter code:
>
>>
>
>>
>
>>
>
>> for k=0,size3[1]-size2[1]-1 do begin
>
>>
>
>>   for l=0,size3[2]-size2[2]-1 do begin
>
>>
>
>>   for i=0, size2[1]-1 do begin
>
>>
>
>>   for j=0, size2[2]-1 do begin
>
>>
>
>>
>
>>
>
>>
>
>>   holder[i,j] = (se[i,j]*padding[i+k,j+l])
>
>>
>
>>   med = MEDIAN(holder)
>
>>
>
>>   clean[k,l] = med
>
>>
>

>>
>
>>
>
>>   endfor
>
>>
>
>>   endfor
>
>>
>
>>
>
>>
>
>>
>
>>   endfor
>
>>
>
>> endfor
>
>>
>
>>
>
>>
>
>>
>
>> Here, size2 is the size of the circular structuring element and size3 is the size of the padded noisy image. So basically I multiply the structuring element by the noisy image (which is padded), store in the values that fill the circle, then find the median and assign it to a "cleaned" image. Then the loop moves the structuring element 1 unit over...
>
>>
>
>>
>
>>
>
>>
>
>> At the moment, the program has done up to r=17, and has been running for 7.2 hours. I've been printing the time it takes for each radii to complete and it will take something like a further 5.5 hours to complete! :(
>
>>
>
>>
>
>>
>

>
>> Cheers!
>
>
>
> Like Fabien, I think you should lose the inner two loops. They are equivalent to something like:
>
>
>
> holder = (se*padding[k:k+size2[1]-1,l:l+size2[2]-1])
>
> med = MEDIAN(holder)
>
> clean[k,l] = med
>
>
>
> But I wonder if your code really does calculate the medians you want, with or without this change. If the array "se" is your "structuring element", which is a binary mask defining the circular area over which you want to calculate the median, then the number you calculate will be biased toward zero by the elements that are zeroed by multiplication with the mask.
>
>
>
> So if I understand what you are trying to do, you may actually want to write the whole thing as:
>
>
>
> indx = where(se)
>
> for k=0,size3[1]-size2[1]-1 do begin
>
> for l=0,size3[2]-size2[2]-1 do begin
>
> clean[k,l] = median((padding[k:k+size2[1]-1,l:l+size2[2]-1])[indx])
>
> endfor
>
> endfor

Thank you so so much for this. Amazing how the run time can go from life time scales to coffee scales :D.

Also I realised that the 0's in my mask will create a bias and shift in the median, I was going to change all the zero's in my SE to NaN's, but will it suffice to replace:

indx = where(se)

with

indx = where(se EQ 1)?

Cheers.

Subject: Re: I think my circular median filter is very inefficient
Posted by on Mon, 04 Aug 2014 07:29:45 GMT
View Forum Message <> Reply to Message

Den måndagen den 4:e augusti 2014 kl. 05:15:41 UTC+2 skrev JRP:
>
> Thank you so so much for this. Amazing how the run time can go from life time scales to coffee scales :D.

I'm glad it was useful. :o)

> Also I realised that the 0's in my mask will create a bias and shift in the median, I was going to change all the zero's in my SE to NaN's, but will it suffice to replace:
>
> indx = where(se)
>
> with
>
> indx = where(se EQ 1)?

If se is binary, those two operations are equivalent.
