Subject: Re: not-quite meidan filter
Posted by Gray on Thu, 30 Sep 2010 20:29:18 GMT
View Forum Message <> Reply to Message

On Sep 30, 3:38 pm, JJ <j...@cornell.edu> wrote:
> I would like to do something that is similar to, but not quite the
> same as, a median filter to a 2D array. Instead of choosing the
> median value in a box surrounding each pixel, I would like to chose
> the value in that box that occurs most frequently.
>
> For example, if I had
>
> 1 1 1 1 1
> 1 1 1 1 1
> 2 2 2 2 2
> 2 2 3 3 3
> 3 3 3 3 3
>
> The median would be 2, but I would want the value 1 (it occurs 10
> times, which is more than the 8 instances of the value 3 or the 7
> instances of the value 2).
>
> Can anyone think of a clever way to do this that would be fast in IDL
> (ie, no looping through the pixels)? I need it to work for box sizes
> up to around 21. Ties may be broken arbitrarily.
>
> Is there already a name for this concept?
>
> Thanks.
>
> -Jonathan

So you're using a "mode" filter. Try max(histogram(pixels)). You
should be careful to choose your binsize appropriately if you have non-
integer data, however; or, alternatively you can use my statistical
mode which finds the mode of a continuous distribution by maximizing
the kernel density estimation. If you'd like the code for that, let
me know.

Subject: Re: not-quite meidan filter
Posted by jeanh on Thu, 30 Sep 2010 20:51:42 GMT
View Forum Message <> Reply to Message

On 30/09/2010 3:38 PM, JJ wrote:
> I would like to do something that is similar to, but not quite the
> same as, a median filter to a 2D array. Instead of choosing the
> median value in a box surrounding each pixel, I would like to chose

> the value in that box that occurs most frequently.
>
> For example, if I had
>
> 1 1 1 1 1
> 1 1 1 1 1
> 2 2 2 2 2
> 2 2 3 3 3
> 3 3 3 3 3
>
> The median would be 2, but I would want the value 1 (it occurs 10
> times, which is more than the 8 instances of the value 3 or the 7
> instances of the value 2).
>
> Can anyone think of a clever way to do this that would be fast in IDL
> (ie, no looping through the pixels)? I need it to work for box sizes
> up to around 21. Ties may be broken arbitrarily.
>
> Is there already a name for this concept?
>
> Thanks.
>
> -Jonathan

Hi,
this is what I use (similar to Gray's suggestion)

```
function getMajority,data, central=central
;This function returns the majority value of the input array.
;If there is a tie, a random mode is returned.
;If the central keyword is specified, if there is a tie and the value of
the central cell of the array is in one of
;the mode, then this value is returned.
;
;
;Written by Jean-Gabriel Hasbani   (firstname@lastname.ca)
;December 2007


 majority = -1
 ;Use histogram to find the mode
 histo = histogram(data, min=0, r=ri)
 ;Get the number of time the mode(s) value is repeated
 MaxVal = max(histo)
 ;Find every entry in the histogram that have this frequency
 maxValInd = where(histo eq maxVal, count)
 ;If there is only 1 mode, return its value
 if count eq 1 then return, data[ri[ri[maxValInd]]]


 ;else, look at all the possible values (modes) and return a random one

 ;If the keyword is specified, if the central cell is of the type of one
of the majority, return this value
 if keyword_set(central) then begin
  ;get the central cell value
  centralValue = data[n_elements(data)/2 - 1]
  ;For each mode, get the value and compare is with the central value
  for i = 0, count-1 do begin
   histoData = data[ri[ri[maxValInd[i]]]]
   if histoData eq centralValue then return, centralValue
  endfor
 endif

 ;Create a random index
 randomID = randomu(seed,count)
 randomID = (sort(randomID))[0]

 ;return the random mode value
 return,  data[ri[ri[maxValInd[randomID]]]]
end
```

Jean

Subject: Re: not-quite meidan filter
Posted by JJ on Thu, 30 Sep 2010 21:06:01 GMT
View Forum Message <> Reply to Message

>
> So you're using a "mode" filter. Try max(histogram(pixels)). You
> should be careful to choose your binsize appropriately if you have non-
> integer data, however; or, alternatively you can use my statistical
> mode which finds the mode of a continuous distribution by maximizing
> the kernel density estimation. If you'd like the code for that, let
> me know.

I think I see what you're suggesting, but unless I'm missing
something, it seems that I would still have to loop over all the
pixels in my image - which is exactly what I don't want to do. It's
good to know the name "mode" though.

I have worked up a solution which will work as long as I have a
limited number of possible values - my test case is only 53 distinct
values, which is OK. I create a 3D array with a slice for each
distinct value, where value of each slice is the coverage (0 or 1) for
that particular value. I then convol each slice with a simple box
kernel filled with 1's, which gives me number of instances of that
value in the box. Then I do a max (dim = 3) on the cube and get the
"max_subscripts", which I can then easily convert to the disticnt
value that I want.

This method is reasonably fast (especially compared to how long a loop
would take), but seems a little clunky and might cause trouble with a
lot more values. Come to think of it though, I don't need the whole
cube - I could just process one value at a time and just keep the
maximum so far.

Any better solutions appreciated.

Thanks.

-Jonathan

Subject: Re: not-quite meidan filter
Posted by David Fanning on Thu, 30 Sep 2010 21:22:46 GMT
View Forum Message <> Reply to Message

JJ writes:

> I would like to do something that is similar to, but not quite the
> same as, a median filter to a 2D array. Instead of choosing the
> median value in a box surrounding each pixel, I would like to chose
> the value in that box that occurs most frequently.
>
> For example, if I had
>
> 1 1 1 1 1
> 1 1 1 1 1
> 2 2 2 2 2
> 2 2 3 3 3
> 3 3 3 3 3
>
> The median would be 2, but I would want the value 1 (it occurs 10
> times, which is more than the 8 instances of the value 3 or the 7
> instances of the value 2).
>
> Can anyone think of a clever way to do this that would be fast in IDL
> (ie, no looping through the pixels)? I need it to work for box sizes

> up to around 21. Ties may be broken arbitrarily.
>
> Is there already a name for this concept?

I don't know. This article might give you some ideas,
though:

   http://www.dfanning.com/idl_way/smregval.html

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: not-quite meidan filter
Posted by Jeremy Bailin on Fri, 01 Oct 2010 11:42:56 GMT
View Forum Message <> Reply to Message

On Sep 30, 3:38 pm, JJ <j...@cornell.edu> wrote:
> I would like to do something that is similar to, but not quite the
> same as, a median filter to a 2D array. Instead of choosing the
> median value in a box surrounding each pixel, I would like to chose
> the value in that box that occurs most frequently.
>
> For example, if I had
>
> 1 1 1 1 1
> 1 1 1 1 1
> 2 2 2 2 2
> 2 2 3 3 3
> 3 3 3 3 3
>
> The median would be 2, but I would want the value 1 (it occurs 10
> times, which is more than the 8 instances of the value 3 or the 7
> instances of the value 2).
>
> Can anyone think of a clever way to do this that would be fast in IDL
> (ie, no looping through the pixels)? I need it to work for box sizes
> up to around 21. Ties may be broken arbitrarily.
>

> Is there already a name for this concept?
>
> Thanks.
>
> -Jonathan

Any non-loop solution I've thought of is going to require a lot of
memory, since you're going to end up storing one slice for each
central pixel. For images of size no more than 21, that might be okay,
but it won't work much larger. I'd do something like this:

- create an array with each filter-box-over-an-individual-pixel as a
separate 2D slice in a 3D array
- use VALUE_LOCATE to map your values onto simple integers from 0 to
N-1
- increment the values in slice #i by i*N so that the values in each
slice are unique
- perform the histogram, for which entries i*N through (i+1)*N-1 are
the repeat counts for slice i
- reform the histogram into a 2D array, N by n_slices, and run
SORT_ND on it so that you know where the maximum histogram value for
each slice is
- subtract back out i*N from that value, reform back to the original
dimensions, and do the reverse mapping to go back to your values

-Jeremy.