## Subject: 2D Savitzky-Golay derivative filter?
Posted by dg86 on Sun, 03 Feb 2013 18:55:32 GMT
View Forum Message <> Reply to Message

Dear Folks,

I would like to use Savitzky-Golay filters to calculate gradients of images.
Before I roll my own code to calculate the filter coefficients, I was wondering
if anyone here already had working routines that they would be willing to share.

IDL's built-in SAVGOL routine only computes one-dimensional filters, and I need
two-dimensional filters. Erik Rosolowsky's SAVGOL2D computes two-dimensional
smoothing filters, but not two-dimensional derivative filters.

Alternatively, is there an equivalently good IDL way to compute image gradients?
A useful replacement would offer the noise rejection of Savitzky-Golay without
suppressing peaks the way that SMOOTH() does.

Many thanks,

David

## Subject: Re: 2D Savitzky-Golay derivative filter?
Posted by d.poreh on Thu, 07 Feb 2013 07:33:36 GMT
View Forum Message <> Reply to Message

On Thursday, February 7, 2013 3:19:01 AM UTC+1, David Grier wrote:
> Dear Folks,
>
>
>
> I append a routine that computes two-dimensional Savitzky-Golay filters for smoothing and taking derivatives of images. This is based substantially on Erik Rosolowsky's savgol2d() routine, with some code simplification and the addition of capabilities for computing derivatives of specified order along each direction.
>
>
>
> The benefit of Savitzky-Golay filters for image analysis is that they suppress noise while retaining features of interest such as peaks and ridges. They therefore are particularly useful for computing gradients of images with additive noise.
>
>
>
> Comments and suggestions are warmly solicited.
>
>

>
> All the best,
>
>
>
> David
>
>
>
> ;+
>
> ; NAME:
>
> ;     savgol2d()
>
> ;
>
> ; PURPOSE:
>
> ;     Calculate two-dimensional Savitzky-Golay filters for smoothing images or
>
> ;     computing their derivatives.
>
> ;
>
> ; CALLING SEQUENCE:
>
> ;     filter = savgol2d(dim, order)
>
> ;
>
> ; INPUTS:
>
> ;     dim: width of the filter [pixels]
>
> ;     order: The degree of the polynomial
>
> ;
>
> ; KEYWORD PARAMETERS:
>
> ;     dx: order of the derivative to compute in the x direction
>
> ;         Default: 0 (no derivative)
>
> ;     dy: order of derivative to compute in the y direction
>
> ;         Default: 0 (no derivative)

```
>
> ;
>
> ; OUTPUTS:
>
> ;    filter: [dim,dim] Two-dimensional Savitzky-Golay filter
>
> ;
>
> ; EXAMPLE:
>
> ; IDL> dadx = convol(a, savgol2d(11, 6, dx = 1))
>
> ;
>
> ; MODIFICATION HISTORY:
>
> ;  Algorithm based on SAVGOL2D:
>
> ;  Written and documented
>
> ;  Fri Apr 24 13:43:30 2009, Erik Rosolowsky <erosolo@A302357>
>
> ;
>
> ;  02/06/2013 Revised version by David G. Grier, New York University
>
> ;-
>
>
>
> function savgol2d, dim, order, dx = dx, dy = dy
>
>
>
> COMPILE_OPT IDL2
>
>
>
> umsg = 'USAGE: filter = dgsavgol2d(dim, order)'
>
>
>
> if n_params() ne 2 then begin
>
>    message, umsg, /inf
>
>    return, -1
```

```
>
> endif
>
> if ~isa(dim, /scalar, /number) then begin
>
>   message, umsg, /inf
>
>   message, 'DIM should be the integer width of the filter', /inf
>
>   return, -1
>
> endif
>
>
>
> if ~isa(order, /scalar, /number) then begin
>
>   message, umsg, /inf
>
>   message, 'ORDER should be the integer order of the interpolaying polynomial', /inf
>
>   return, -1
>
> endif
>
> if ~(order lt dim) then begin
>
>   message, umsg, /inf
>
>   message, 'ORDER should be less than DIM', /inf
>
>   return, -1
>
> endif
>
>
>
> if ~isa(dx, /scalar, /number) then dx = 0
>
> if ~isa(dy, /scalar, /number) then dy = 0
>
> if dx lt 0 or dy lt 0 then begin
>
>   message, umsg, /inf
>
>   message, 'DX and DY should be non-negative integers', /inf
>
>   return, -1
```

```
>
> endif
>
> if (dx + dy ge order) then begin
>
>    message, umsg, /inf
>
>    message, 'DX + DY should not be greater than ORDER', /inf
>
>    return, -1
>
> endif
>
>
>
> npts = dim^2
>
>
>
> x = rebin(findgen(dim)-dim/2, dim, dim)
>
> y = transpose(x)
>
> x = reform(x, npts)
>
> y = reform(y, npts)
>
>
>
> Q = findgen((order+1)*(order+2)/2, npts)
>
>
>
> n = 0
>
> for nu = 0, order do begin
>
>    ynu =  y^nu
>
>    for mu = 0, order-nu do begin
>
>       Q[n++, *] = x^mu * ynu
>
>    endfor
>
> endfor
>
>
```

```
>
>  a = transpose(invert(Q # transpose(Q)) # Q)
>
>  filter = fltarr(npts)
>
>  b = [1., fltarr(npts-1)]
>
>  ndx = dx + (order + 1) * dy
>
>  for i = 0, npts-1 do begin
>
>     filter[i] = (a ## b)[ndx]
>
>     b = shift(b, 1)
>
>  endfor
>
>
>
>  return, reform(filter, dim, dim)
>
>  end
```

I wish you put an example (visual) to see the benefit of this over others (sobel etc.)...
Cheers,
Dave

---

Subject: Re: 2D Savitzky-Golay derivative filter?
Posted by Klemen on Thu, 07 Feb 2013 13:52:35 GMT
View Forum Message <> Reply to Message

Dear David,

the solution I have over here works also over a larger domain. It doesn't compute the gradient but a sky-view factor. We use it for visualization of terrain to expose subtle terrain features, which is important for instance in archeology.
So just in case you are looking for a visual solution to find edges and you don't necessary need gradients, try the following link:

http://iaps.zrc-sazu.si/en/svf#v

We have there a SAV file you can download and test the data. I guess you aren't working with georeferenced data, so it might be that you need to set the vertical exaggeration factor to an appropriate value (try different values). You can find also a paper describing the theory behind the method. If you need the original code, contact me.

Cheers, Klemen

---

Subject: Re: 2D Savitzky-Golay derivative filter?
Posted by dg86 on Sat, 09 Feb 2013 00:39:48 GMT
View Forum Message <> Reply to Message

Dear Klemen,

Thanks for the helpful post.

I actually need to compute derivatives, specifically the gradients of experimentally measured holograms. Roberts and Sobel are not appropriate for my application because they compute the magnitude of the gradient, rather than the gradient itself. As an example, I've uploaded a typical noisy hologram of a colloidal sphere together with the x-component of its gradient computed with finite differences, and the equivalent component computed with the SG filter:

http://physics.nyu.edu/grierlab/noisygradient.png

For what it's worth, the SG version is more useful for my subsequent analysis.

All the best,

David