Savitzky-Golay Filters for 2D Images

John Krumm
Microsoft Research
Microsoft Corporation
Redmond, WA 98052

August 2001

Table of Contents

1. Introduction

Savitzky-Golay filters have been popularized by the *Numerical Recipes* books. The one-dimensional filters presented in the book and the original papers are used for smoothing one-dimensional, tabulated data and also for computing numerical derivatives. The fundamental idea is to fit a different polynomial to the data surrounding each data point. The smoothed points are computed by replacing each data point with the value of its fitted polynomial. Numerical derivatives come from computing the derivative of each fitted polynomial at each data point.

While fitting polynomials for these purposes is obvious, the surprising part is that the polynomial coefficients can be computed with a linear filter. For smoothing, only one coefficient of the polynomial is needed, so the whole process of least squares fitting at every point becomes a simple process of applying the appropriate linear filter at every point. The *Numerical Recipes* books give a description of the filters and source code (FORTRAN and C) for computing these one-dimensional filters for both smoothing and numerical derivatives.

The Savitzky-Golay filters are just as useful for image processing, where the idea is to fit a two-dimensional polynomial to a two-dimensional subsection of the image for smoothing and numerical derivatives. While the Numerical Recipes books give a way of computing the filters for one-dimensional data, this article shows how to compute two-dimensional filters for image data. It gives a MatLab routine for computing the filters and C source code for the actual filter coefficients for different image patch sizes and different polynomial orders.

back to top

2. Example of How to Compute Two-Dimensional Savitzky-Golay Filters

As an example, suppose we want to smooth or differentiate an image based on 5x5 image patches. The image patch is laid out as follows:

		x_i				
		-2	-1	0	1	2
y_i	-2	d(0)	d(1)	d(2)	d(3)	d(4)
	-1	d(5)	d(6)	d(7)	d(8)	d(9)
	0	d(10)	d(11)	d(12)	d(13)	d(14)
	1	d(15)	d(16)	d(17)	d(18)	d(19)
	2	d(20)	d(21)	d(22)	d(23)	d(24)

Table 1: Patch of image data d(i) with its local coordinate system.

d(i) is the pixel value, and the column vector $\underline{d}$ represents all the image data, *i.e.*

$$\underline{d} = (d(0)\ d(1)\ \ldots\ d(24))^T$$

We want to fit a 3rd order, two-dimensional polynomial to this data. The polynomial is

$$d(i) \approx f(x_i,y_i) = a_{00} + a_{10}x_i + a_{01}y_i + a_{20}x_i^2 + a_{11}x_iy_i + a_{02}y_i^2 + a_{30}x_i^3 + a_{21}x_i^2y_i + a_{12}x_iy_i^2 + a_{03}y_i^3$$

Note that the coefficient of x^iy^j is a_{ij}. (x_i,y_i) is the pixel coordinate of d(i).

To compute the coefficients from the data we set up a matrix equation:

$$X\underline{a} = \underline{d} \qquad (1)$$

where

$$X = \begin{matrix} 1 & x_0 & y_0 & x_0^2 & x_0y_0 & y_0^2 & x_0^3 & x_0^2y_0 & x_0y_0^2 & y_0^3 \\ 1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 & x_1^3 & x_1^2y_1 & x_1y_1^2 & y_1^3 \\ : & : & : & : & : & : & : & : & : & : \\ 1 & x_{24} & y_{24} & x_{24}^2 & x_{24}y_{24} & y_{24}^2 & x_{24}^3 & x_{24}^2y_{24} & x_{24}y_{24}^2 & y_{24}^3 \end{matrix}$$

and $\underline{a}$ is the vector of polynomial coefficients:

$$\underline{a} = (a_{00}\ a_{10}\ a_{01}\ a_{20}\ a_{11}\ a_{02}\ a_{30}\ a_{21}\ a_{12}\ a_{03})^T$$

Equation (1) simply reproduces the polynomial for each pixel in the image patch. We solve for the polynomial coefficients using least squares:

$$\underline{a} = (X^TX)^{-1}X^T\underline{d}$$

$C = (X^TX)^{-1}X^T$ is the pseudo-inverse of X, and it is independent of the image data. Each polynomial coefficient is computed as the inner product of one row of C and the column of pixel values d. This is the surprising part about Savitzky-Golay filters: the polynomial coefficients are computed using a linear filter on the data. Just as one can reassemble d back into a rectangular patch of pixels, one can also assemble each row of C into the same size rectangle to get a traditional-looking image filter.

To complete this example, here are the filters for the first three polynomial coefficients. We will use the naming convention that coefficient a_{ij} is computed from rectangular filter C_{ij}. The first three filters are:

$$C_{00} = \begin{matrix} -0.0743 & 0.0114 & 0.0400 & 0.0114 & -0.0743 \\ 0.0114 & 0.0971 & 0.1257 & 0.0971 & 0.0114 \\ 0.0400 & 0.1257 & 0.1543 & 0.1257 & 0.0400 \\ 0.0114 & 0.0971 & 0.1257 & 0.0971 & 0.0114 \\ -0.0743 & 0.0114 & 0.0400 & 0.0114 & -0.0743 \end{matrix}$$

$$C_{10} = \begin{matrix} 0.0738 & -0.0119 & -0.0405 & -0.0119 & 0.0738 \\ -0.1048 & -0.1476 & -0.1619 & -0.1476 & -0.1048 \\ 0 & 0 & 0 & 0 & 0 \\ 0.1048 & 0.1476 & 0.1619 & 0.1476 & 0.1048 \\ -0.0738 & 0.0119 & 0.0405 & 0.0119 & -0.0738 \end{matrix}$$

$$C_{01} = \begin{matrix} 0.0738 & -0.1048 & 0 & 0.1048 & -0.0738 \\ -0.0119 & -0.1476 & 0 & 0.1476 & 0.0119 \\ -0.0405 & -0.1619 & 0 & 0.1619 & 0.0405 \\ -0.0119 & -0.1476 & 0 & 0.1476 & 0.0119 \\ 0.0738 & -0.1048 & 0 & 0.1048 & -0.0738 \end{matrix}$$

back to top

3. Example of How to Use Two-Dimensional Savitzky-Golay Filters

Suppose we want to smooth an image. Using a Savitzky-Golay filters, we are

conceptually fitting a two-dimensional polynomial to the image patch surrounding each pixel and then evaluating this polynomial. The local coordinate system that we use for the image patch (see Table 1) has (x,y) = (0,0) at the pixel of interest in the middle of the patch. Thus to compute the smoothed value of the pixel, we just evaluate the polynomial at (x,y) = (0,0). This turns out to be merely a_{00}, which we can compute by applying filter C_{00} to the image patch.

Suppose we want to compute partial derivatives on the patch. The two partial derivatives of the fitted polynomial are

$$f_x(x_i,y_i) = a_{10} + 2a_{20}x_i + a_{11}y_i + 3a_{30}x_i^2 + 2a_{21}x_iy_i + a_{12}y_i^2$$

$$f_y(x_i,y_i) = a_{01} + a_{11}x_i + 2a_{02}y_i + a_{21}x_i^2 + 2a_{12}x_iy_i + 3a_{03}y_i^2$$

Evaluating at (x,y) = (0,0), the results are simply $f_x(0,0) = a_{10}$ and $f_y(0,0) = a_{01}$, which are computed with filters C_{10} and C_{01} above.

back to top

4. MatLab Routine for Computing Savitzky-Golay Filters

Generalizing the example of Section 2, the following MatLab function called SavGol.m, generates the Savitzky-Golay filters for a user-specified square patch size and polynomial order. This function requires the MatLab Symbolic Math Toolbox. Note that this routine does not check to make sure the patch size is large enough to fit the requested polynomial order. Note also that for square patches with even dimensions, the origin of the local patch coordinates is at the center of the center four pixels.

SavGol.m

back to top

5. MatLab Routine for Checking Savitzky-Golay Filters

The following MatLab routine, called SavGolTest.m, takes the output of SavGol and tests it on an image patch made from a polynomial whose coefficients are all one.

SavGolTest.m

back to top

6. C Program with Savitzky-Golay Filters

The following C code gives the two-dimensional Savitzky-Golay filters for different patch sizes and different polynomial orders. The filters are stored in one-dimensional arrays in row-major order. The code was generated using the SavGol() MatLab routine and another MatLab routine to format the numbers into array declarations.

SavGol.cpp

back to top