Bull. Spec. Astrophys. Obs., 1996, 39, 131–139

Fast image processing methods for PC: 2. Regression smoothing without loss of resolution

Tsvetan B. Georgiev

Rozhen National Observatory, BG-4700 Smolyan, Bulgaria e-mail: tsgeorg@bgearn.bitnet
Visiting astronomer to SAO of the RAS

Received May, 10, 1994; accepted July, 10, 1995.

Abstract. The image smoothing method by direct digital convolution, corresponding to the sliding regression surface of 3rd or 5th degree, is described. The coefficients of the convolution nuclei in the one– and two–dimensional cases are derived. The C–program, added to the *PCVISTA* (Treffers and Richmond, 1989) package, is described. It uses circular window and treats the whole frame, including the periphery. The fast convolution algorithm, which decreases several times the number of the arithmetic multiplications is realized. The text of the program is published.

Key words: aperture photometry – CCD data processing

1. Introduction

The practice of image processing, especially the methods of restoration, needs a good smoothing procedure. Under reasonable conditions of a narrow smoothing window it should be able to suppress the noise, keeping the resolution unchanged. The method of blurring by convolution of the frame with the gaussian nucleus always decreases the resolution.

One natural method for smoothing is the regression estimation of each pixel value, using the values of the neighbouring pixels. The method may be described as a local regression surface of 3rd or higher order, which slides across the frame row by row. To conserve the resolution, the regression must be derived over a small enough data window, centred on the current pixel. However, the direct performance of the least squares method (LSM) for each pixel of the image demands a great amount of computations.

The approach pointed out by Heasley (1984) avoids this problem with the computer time. The LSM procedure is linear, and the estimation of the current pixel value must be found as a linear combination of the pixels in the window with suitable constant coefficients. Consequently, the regression smoothing may be done by means of a convolution transform, where the coefficients of the convolution nucleus, depending on the window size, must be preliminarily derived.

Regression smoothing has been successfully used by the author for the last few years (Georgiev, 1990; 1991). However, this method has not been fully described in the literature till now, and maybe for this reason is not very popular.

The present paper aims to give the theoretical and practical solutions, making the regression smooth available to any PC. The formulae for the convolution coefficients in the one– and two–dimensional cases are derived in Sections 2 and 3, respectively. The used fast convolution algorithm, based on the symmetry of the convolution nucleus, is described in Section 4. The computer program *SMOOTH* for the two-dimensional case is described in Section 5. The examples and the conclusions are given in Sections 6 and 7. The text of the program is given as the Appendix. The basic conception of realized image processing, including the method of circular windows and sliding frame band, are described in the first paper of this series, hereafter referred to as Paper I (Georgiev, 1996).

2. One–dimensional case

Let m_1 be the data row and the odd number W is the size of the smoothing window, centred on the current pixel. We specified the coordinate axis Oi with the origin in the point of the current pixel. Thus, the i–coordinates of the pixels in the window are $-W/2, \dots, -1, 0, 1, \dots, W/2$, and the value of the current pixel is m_0. Note that $W/2$ is the result of integer type division.

Here we assume the cubic smoothing polynomial in the form

$$m = A_0 + A_1 \cdot i + A_2 \cdot i^2 + A_3 \cdot i^3. \qquad (2.1)$$

Simplifying the regression formulae, we use the denotations

$$\langle i^k \rangle = \sum_i i^k / W \quad \text{and}$$

$$\langle\langle i^k, i^l \rangle\rangle = \sum_i i^k \cdot i^l - W \cdot \langle i^k \rangle \cdot \langle i^l \rangle. \qquad (2.2)$$

Hereafter in Section 2 i changes from $-W/2$ to $W/2$.

The free term of the regression (2.1), which will be the estimation of the current pixel value m_0, can be expressed as

$$A_0 = \langle m \rangle - A_1 \cdot \langle i \rangle - A_2 \cdot \langle i^2 \rangle - A_3 \cdot \langle i^3 \rangle. \qquad (2.3)$$

Then the reduced set of the normal LSM equations has the form

$$\left| \begin{array}{ll} \langle\langle i, i \rangle\rangle \cdot A_1 + \langle\langle i, i^2 \rangle\rangle \cdot A_2 + \langle\langle i, i^3 \rangle\rangle \cdot A_3 & = \langle\langle i, m \rangle\rangle \\ \langle\langle i^2, i \rangle\rangle \cdot A_1 + \langle\langle i^2, i^2 \rangle\rangle \cdot A_2 + \langle\langle i^2, i^3 \rangle\rangle \cdot A_3 & = \langle\langle i^2, m \rangle\rangle \\ \langle\langle i^3, i \rangle\rangle \cdot A_1 + \langle\langle i^3, i^2 \rangle\rangle \cdot A_2 + \langle\langle i^3, i^3 \rangle\rangle \cdot A_3 & = \langle\langle i^3, m \rangle\rangle. \end{array} \right. \qquad (2.4)$$

Because of the special choice of the coordinate system we have

$$\langle i \rangle = \langle i^3 \rangle = \langle\langle i, i^2 \rangle\rangle = \langle\langle i^2, i^3 \rangle\rangle = 0. \qquad (2.5)$$

Using (2.3) and the second equation of (2.4) we derive the system of two equations for derivation of A_0 :

$$\left| \begin{array}{l} A_0 = \langle m \rangle - A_2 \cdot \langle i^2 \rangle \\ A_2 \cdot \langle\langle i^2, i^2 \rangle\rangle = \langle\langle i^2, m \rangle\rangle. \end{array} \right. \qquad (2.6)$$

Then the solution for A_0 is

$$A_0 = \langle m \rangle - \langle\langle i^2, m \rangle\rangle \cdot \langle i^2 \rangle / \langle\langle i^2, i^2 \rangle\rangle. \qquad (2.7)$$

The value of A_0 is the estimation of the value of the current pixel m_0. It is found as a linear combination of the neighbouring pixels. This solution may be expressed as the one dimensional discrete convolution

$$A_0 = \sum C_i \cdot m_i, \qquad (2.8)$$

and the formula, giving the coefficients C_i, is

$$C_i = 1/W - \langle i^2 \rangle (i^2 - \langle i^2 \rangle) / \langle\langle i^2, i^2 \rangle\rangle. \qquad (2.9)$$

The fulfilment of the condition $C_i = C_{-i}$ and $\sum C_i \cdot m = 1$ is done.

As an example, in the case $W = 5$, we have $C_i = 17/35 - i^2/7$, and the values of the coefficients are $C_0 = 17/35$, $C_1 = C_{-1} = 12/35$ and $C_2 = C_{-2} = -3/35$. Note, that in the case of $W = 3$, we have $C_0 = 1$ and $C_1 = C_{-1} = 0$, i.e. the method does not give useful coefficients. The suitable artificial coefficients, that smooth, but decrease the resolution, are $C_0 = 3/4$ and $C_1 = C_{-1} = 1/8$.

In the case of the 5th degree we have a set of equations

$$\left| \begin{array}{l} A_0 = \langle m \rangle - A_2 \cdot \langle i^2 \rangle - A_4 \cdot \langle i^4 \rangle \\ \langle\langle i^2, i^2 \rangle\rangle \cdot A_2 + \langle\langle i^2, i^4 \rangle\rangle \cdot A_4 = \langle\langle i^2, m \rangle\rangle \\ \langle\langle i^2, i^4 \rangle\rangle \cdot A_2 + \langle\langle i^2, i^2 \rangle\rangle \cdot A_4 = \langle\langle i^4, m \rangle\rangle. \end{array} \right. \qquad (2.10)$$

We denote

$$\begin{array}{ll} B_0 & = \langle\langle i^4, i^4 \rangle\rangle \cdot \langle\langle i^2, i^2 \rangle\rangle - \langle\langle i^2, i^4 \rangle\rangle \cdot \langle\langle i^2, i^4 \rangle\rangle, \\ B_1 & = (\langle\langle i^2, i^2 \rangle\rangle - \langle\langle i^2, i^2 \rangle\rangle)/B_0, \\ B_2 & = (\langle\langle i^2, i^2 \rangle\rangle - \langle\langle i^2, i^4 \rangle\rangle)/B_0. \end{array} \qquad (2.11)$$

Then the solution for A_0 is similar to that in the case (2.8), but the coefficients are

$$C_i = 1/W - B_1 \cdot (i^2 - \langle i^2 \rangle) + B_2 \cdot (i^4 - \langle i^4 \rangle).$$

3. Two-dimensional case

We apply the approach already used in Section 2. Let the origin of the local coordinate system coincide with the current pixel. The coordinate axes are Oi, where i corresponds to the row number, and Oj, where j corresponds to the column number. The pixel values are $m_{i,j}$ and the current pixel has the value m_{00}.

The regression model of the cubic polynomial surface is

$$\begin{array}{rcl} m & = & A_0 + A_i \cdot i + A_2 \cdot j + A_3 \cdot i \cdot j \\ & & +A_4 \cdot i^2 + A_5 \cdot j^2 + A_6 \cdot i^2 \cdot j \\ & & +A_7 \cdot i \cdot j^2 + A_8 \cdot i^3 + A_9 \cdot j^3. \end{array} \qquad (3.1)$$

With simplification of the formulae, as in Section 2, we denote

$$\langle i^k \rangle = \sum_i \sum_j i^k / N \quad \text{and}$$

$$\langle\langle i^k, j^l \rangle\rangle = \sum_i \sum_j i^k j^l - N \cdot \langle i^k \rangle \cdot \langle j^l \rangle. \qquad (3.2)$$

In Section 3 i and j acquire all possible values in the window (with size of W) and N is the number of pixels in the window. The reduced LSM system now consists of 9 equations. Because of the special choice of coordinates many regression sums, as in (2.5), occur equal to zero. Then the system of equations for A_0, that corresponds to (2.6), is

$$\left| \begin{array}{l} A_0 = \langle m \rangle - A_4 \cdot \langle i^2 \rangle - A_5 \cdot \langle j^2 \rangle \\ A_4 \cdot \langle\langle i^2, i^2 \rangle\rangle + A_5 \cdot \langle\langle i^2, j^2 \rangle\rangle = \langle\langle i^2, m \rangle\rangle \\ A_4 \cdot \langle\langle j^2, i^2 \rangle\rangle + A_5 \cdot \langle\langle j^2, i^2 \rangle\rangle = \langle\langle j^2, m \rangle\rangle. \end{array} \right. \qquad (3.3)$$

However, $\langle\langle i^2, i^2 \rangle\rangle = \langle\langle j^2, j^2 \rangle\rangle$ and $\langle\langle i^2, j^2 \rangle\rangle = \langle\langle j^2, i^2 \rangle\rangle$ solutions for A_0 are

$$\begin{array}{rcl} A_0 & = & \langle m \rangle - \langle i^2 \rangle \cdot (\langle\langle i^2, m \rangle\rangle + \langle\langle j^2, m \rangle\rangle)/ \\ & & (\langle\langle i^2, i^2 \rangle\rangle + \langle\langle i^2, j^2 \rangle\rangle). \end{array} \qquad (3.4)$$

Thus, the coefficient of the convolution nucleus is expressed as

$$\begin{array}{rcl} C_{ij} & = & 1/N - \langle i^2 \rangle (i^2 + j^2 - 2 \cdot \langle i^2 \rangle)/ \\ & & (\langle\langle i^2, i^2 \rangle\rangle + \langle\langle i^2, j^2 \rangle\rangle). \end{array} \qquad (3.5)$$

In the case of the surface of the 5th degree we have

$$m = \sum_{k=0}^{5}\sum_{l=0}^{5} A_{kl} \cdot i^k \cdot j^l.$$

The reduced system of regression equations now consists of 20 equations. Again, because of the special choice of the coordinate system, many regression sums occur equal to zero. We denote the nonzero sums as follow:

$$\begin{aligned}
S_1 &= \langle\langle i^2, i^2\rangle\rangle\\
S_2 &= \langle\langle i^2, j^2\rangle\rangle\\
S_3 &= \langle\langle i^2, j^2, i^2\rangle\rangle\\
S_4 &= \langle\langle i^2, j^2, i^2, j^2\rangle\rangle\\
S_5 &= \langle\langle i^2, i^2\rangle\rangle\\
S_6 &= \langle\langle i^2, j^2\rangle\rangle\\
S_7 &= \langle\langle i^2, j^2, i^4\rangle\rangle\\
S_8 &= \langle\langle i^4, j^4\rangle\rangle\\
S_9 &= \langle\langle i^4, i^4\rangle\rangle.
\end{aligned}$$

Let denote also:

$$\begin{aligned}
S_{12} &= S_1 + S_2\\
S_{56} &= S_5 + S_6\\
S_{89} &= S_8 + S_9\\
D_{11} &= S_{12} \cdot S_4 - 2 \cdot S_3 \cdot S_3\\
D_{22} &= S_{89} \cdot S_4 - 2 \cdot S_7 \cdot S_7\\
D_{12} &= S_{56} \cdot S_4 - 2 \cdot S_3 \cdot S_7\\
D &= D_{11} \cdot D_{22} - D_{12} \cdot D_{12}\\
D_1 &= (D_{22} \cdot S_3 - D_{12} \cdot S_7)/D\\
D_2 &= (D_{12} \cdot S_7 - D_{12} \cdot S_3)/D\\
A_1 &= A_{01} + A_{20}\\
A_2 &= A_{22}\\
A_3 &= A_{04} + A_{40}.
\end{aligned}$$

Then the simplified system of regression equations is

$$\begin{aligned}
S_{12} &\cdot a_1 + 2 \cdot S_3 \cdot A_2 + S_{56} \cdot A_3 = \langle\langle i^2, z\rangle\rangle + \langle\langle j^2, m\rangle\rangle\\
S_3 &\cdot A_1 + S_4 \cdot A_2 + S_7 \cdot A_3 = \langle\langle i^2, j^2, m\rangle\rangle\\
S_{56} &\cdot A_1 + 2 \cdot S_7 \cdot A_2 + S_{89} \cdot A_3 = \langle\langle i^4, z\rangle\rangle + \langle\langle j^2, m\rangle\rangle\\
A_{00} &= \langle m\rangle - \langle i^2 + j^2\rangle \cdot A_1 - \langle i^2 j^2\rangle \cdot A_2 - \langle i^4 + j^4\rangle \cdot A_3.
\end{aligned}$$

The solution for A_{00} is the regression estimation of the value of the current pixel m_{00}. It may be presented as the convolution

$$A_{00} = \sum_i \sum_j C_{ij} \cdot m_{ij},$$

where i and j run inside the current position of the smoothing window. The convolution coefficient has the form

$$\begin{aligned}
C_{ij} &= 1/N - B_1 \cdot (i^2 + j^2 - 2 \cdot \langle i^2\rangle) +\\
&\quad -B_2 \cdot (i^1 \cdot j^2 - \langle i^2 j^2\rangle) - B_3 \cdot (i^4 + j^4 - 2 \cdot \langle i^4\rangle),
\end{aligned}$$

where the coefficients B_1, B_2 and B_3 may be written as

$$\begin{aligned}
B_1 = &\ (\langle i^2\rangle \cdot S_4 - \langle i^2 \cdot j^2\rangle \cdot S_3 \cdot D_{22}/D +\\
&+(\langle i^2 \cdot j^2\rangle \cdot S_7 - \langle i^4\rangle \cdot S_4) \cdot D_{12}/D\\
B_2 = &\ 2 \cdot (((\langle i^2 \cdot j^2\rangle \cdot S_3 - \langle i^2\rangle \cdot S_3) \cdot D_1 +\\
&+(\langle i^2 \cdot j^2\rangle \cdot S_7 - \langle i^4\rangle \cdot S_4) \cdot D_2 + \langle i^2 \cdot j^2\rangle)/S_4\\
B_3 = &\ (\langle i^4\rangle \cdot S_4 - \langle i^2 \cdot j^2\rangle \cdot S_7 \cdot D_{11}/D +\\
&+(\langle i^2 \cdot j^2\rangle \cdot S_3 - \langle i^2\rangle \cdot S_4) \cdot D_{12}/D.
\end{aligned}$$

The formulae for the convolution coefficients are efficient if $W > 2$, in the case of the 3rd degree, and if $W > 4$, in the case of the 5th degree polynomial. In all cases the sums of the coefficients are equal to 1.

The values of the convolution coefficients, corresponding to the 3rd and 5th degree surface, for circular windows with sizes 5 and 7 pixels, are given below. These are parts of the convolution nuclei, where the upper left coefficient in each case corresponds to C_{00} and the local coordinates i and j grow up to down and to right, respectively.

3rd degree, $W = 5$

0.2119	0.1612	0.0090
0.1612	0.1104	−0.0418
0.0909	−0.0418	

5th degree, $W = 5$

0.6906	0.2467	0.0125
0.2467	-0.0997	−0.0411
0.0125	−0.0411	

3rd degree, $W = 7$

0.1116	0.0971	0.0536	−0.0188
0.0971	0.0826	0.0392	−0.0333
0.0536	0.0392	−0.0043	
−0.0188	−0.0333		

5th degree, $W = 7$

0.2984	0.2011	0.0027	−0.0162
0.2011	0.1176	−0.0394	0.0107
0.0027	−0.0394	−0.0723	
−0.0162	0.0107		

More detailed profiles of two convolution nuclei in different cases, when the window size is $W = 21$ pixels, are given in Fig.1. It should be pointed out, that negative coefficients appear on the edges of the nuclei, corresponding to regression smoothing. The present nuclei look like the nuclei used for direct restoration of images (see p.e. Frieden, 1975). If the size of the window is small enough, the regression smoothing does not decrease the resolution. Moreover, using sliding surface of the 5th degree, we decrease the FWHM of the stars by about 5-6%.

4. The fast convolution algorithm

Let the frame (i.e. the whole image that must be processed) be presented as a matrix of numbers and

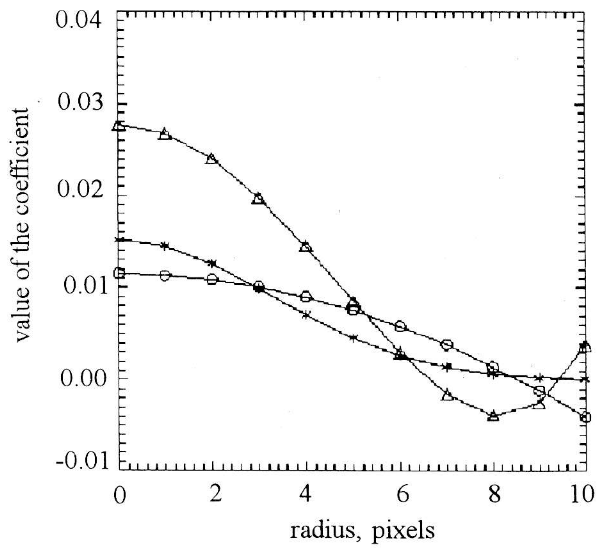

Figure 1: *Comparison of the profiles of the convolution nucleus with W=21 pixels corresponding to the surface of the 3rd degree (circles), the 5th degree (triangles) and gaussian with width of 21 pixels (asterisk Using the 5th degree and window 4< W<1.5 (FWHM) we reach the decrease of the FWHM of the frame of about 5%*

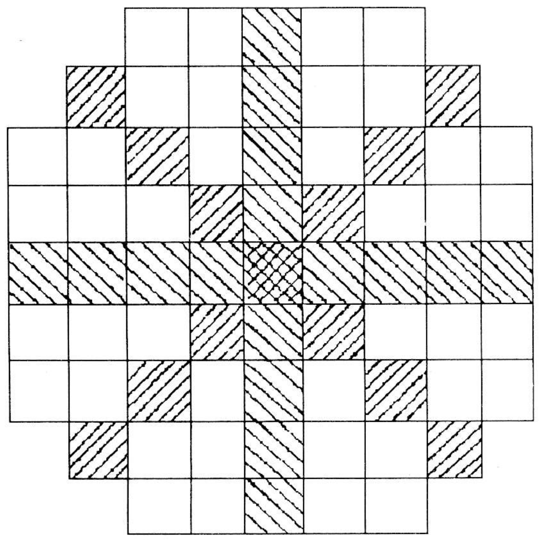

Figure 2: *The symmetry of the pixels of the convolute nucleus with the diameter $W = 9$ pixels. The blank pixels have octal symmetry, the hatched pixels (situated on the axes or diagonals) have quadrant symmetry and only the double hatched central pixel is unique.*

m_{ij} be the value of the current pixel. At the beginning we will use a square window with the size W, where W is an odd integer. This window can move across the image row by row. In each fixed case the central pixel of the window coincides with the current processed pixel. Let the numbers of C_{kl}, where $k, l = -W/2, \dots, 0, \dots, W/2$, be the convolution coefficients. Under these conditions the two–dimensional convolution, that gives new value n_{ij} for each current pixel, may be expressed as the double sum

$$n_{ij} = \sum_{k=-w/2}^{w/2} \sum_{l=-w/2}^{w/2} C_{kl} \cdot m_{i-k,j-l}. \tag{4.1}$$

The expression of (4.1) in the C-language has the form

```
...
s = 0;  for (k = -w/2; k <= w/2; k++)
        for (l = -w/2; l <= w/2; l++)
        s+ = c[k][l] * m[i - k][j - l].
...
```

Here the sum s, which collects the new values for the current pixel, and the convolution coefficients c[ ][ ] are floating point numbers, all other numbers are integer.

The direct realization of the convolution (4.1) leads to a great number of computations. About W floating point multiplications and W floating point additions must be made for each pixel of the frame. The contemporary PCs are fast enough, but for big images we must use the fastest of the available algorithms.

Following the recommendations of Dudgeon and Mersereau (1984), we explore the symmetry of the convolution nucleus. Preliminarily we make the integer summing of the pixels that must be multiplied by equal coefficients, then multiply the sum by the corresponding coefficient. This approach reduces the number of multiplications 4–7 times, and the gain grows with increasing W.

One example of the circular window case with the diameter $W = 9$ is given in Fig.2 (the windows with $W < 9$ are shown in Paper I). Only the central l is unique. The axial and diagonal pixels have quadrant symmetry and the others have octal symmetry.

Based on the symmetry, the present practical realization of the two dimensional convolution with circular window occurs few times faster than the direct method. In the direct method we must make 69 multiplications and 69 additions for each pixel of the frame. Using the symmetry we will make only 13 floating point multiplications, when the preliminary summing may be done by the integer arithmetic.

Let us turn back to the case of the square window. Using the symmetry mentioned above we may apply one complicated but faster C–language expression of the convolution procedure (4.1) as follows:

```
...
s = c[0][0] * m[i][j]; /* central pixel */
for (k=1; k¡=w/2; k++) { /* 4 axial pixels */
ls = m[i-k][j] + m[i+k][j] + m[i][j-k] + m[i][j-k];
s += c[k][0] * ls;
for (l=1; l<=w/2; l++) { /* 4 diagonal pixels */
ls += m[i-k][j-l] + m[i-k][j+l] + m[i+k][j-l] +
m[i+k][j+l];
if(!=k) { /* other 4 symmetric pixels */
ls += m[i-l][j-k] + m[i-l][j+k] + m[i+l][j-k] +
m[i+l][j+k];}
s += c[k][l] * ls.
...
```

Here the floating point values s and c[ ][ ] are the same as in the previous example, and all other numbers are integer. The sum ls must be long integer type.

Practical realization of the convolution loops mentioned above in the program *SMOOTH*, given in Appendix, is more complicated. It reflects the use of the circle window with the system of row limits lw[ ] and frame band with position numbers of the processed image rows pn[ ] (see Paper I).

5. The program *SMOOTH*

The program *SMOOTH* is made in the compact environment of Microsoft C-language, version 5.1, and *PCVISTA* (Treffers, Richmond 1989). It processes *FITS* frames with the integer type of pixel values. Each single call of the program performs smoothing of one specified frame. The input arguments of the program are the input file name, the output file name and the window diameter W. The general algorithm and the designations are the same as in the program *MEDFIL* (Paper I).

When the image dimensions are 400×500 pixels, the processing times of the program *SMOOTH* with $W = 5$, $W = 15$ and $W = 25$ on a 40 MHz IBM PC/AT 386 DX are 14 and 88 and 190 seconds, respectively. The present realization of the program *SMOOTH* demands a 56 kb processor memory.

6. One example

The possibilities of convolution smoothing for elucidating the shape of the extended object, as well as for revealing of faint objects without loss of resolution, are shown in Fig. 3. The object is the dwarf irregular galaxy DDO46 = UGC3966, which has low surface brightness. A part of the B – CCD image, obtained by the 6 m telescope, is used. The exposure time is 900 sec, the seeing is about 1.6 arcseconds (8 pixels), and the dimensions of the frame are 580×400 pixels.

The standard preliminary processing of the frame, including subtraction of the bias and dark frame, flat fielding, etc., is done by *PCVISTA* and a few auxiliary programs of the author.

7. Remarks

The method of regression smooth is a very useful tool for different situation in image processing, and especially before image restoration. To keep the resolution unchanged the window diameter must be smaller than the seeing.

The author expresses his gratitude to Dr. I.D. Karachentsev, Dr. N.A. Tikhonov and Dr. A.I. Kopylov for the useful discussions, as well as G.G. Korotkova for the help in the preparation of this paper. The author is grateful also to the administration of SAO for the excellent conditions in which this work was finished.

This paper is a part of the investigations, supported by the grant F-342/93 of the Bulgarian Ministry of Education and Science.

References

Dudgeon D.E., Mersereau R.M.: 1984, Multidimensional Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs (in Russian: 1988).

Frieden B.R.: 1975, in Huang T.S., ed., Topic in Applied Physics, **6**, Picture Processing and Digital Filtering, Springer-Verlag (in Russian: 1979).

Georgiev Ts.B.: 1990, Astrofiz.Issled. (Izv. SAO), **30**, 127.

Georgiev Ts.B.: 1991, Astrofiz. Issled. (Izv. SAO), **33**, 213.

Georgiev Ts.B.: 1996, Bull. Spec. Astrophys. Obs., **39**, 124, (this issue).

Heasley J.N.: 1984, Publ. Astr. Soc. Pacific, **96**, 767.

Treffers R.R., Richmond M.W.: 1989, Publ. Astr. Soc. Pacific, **101**, 725.

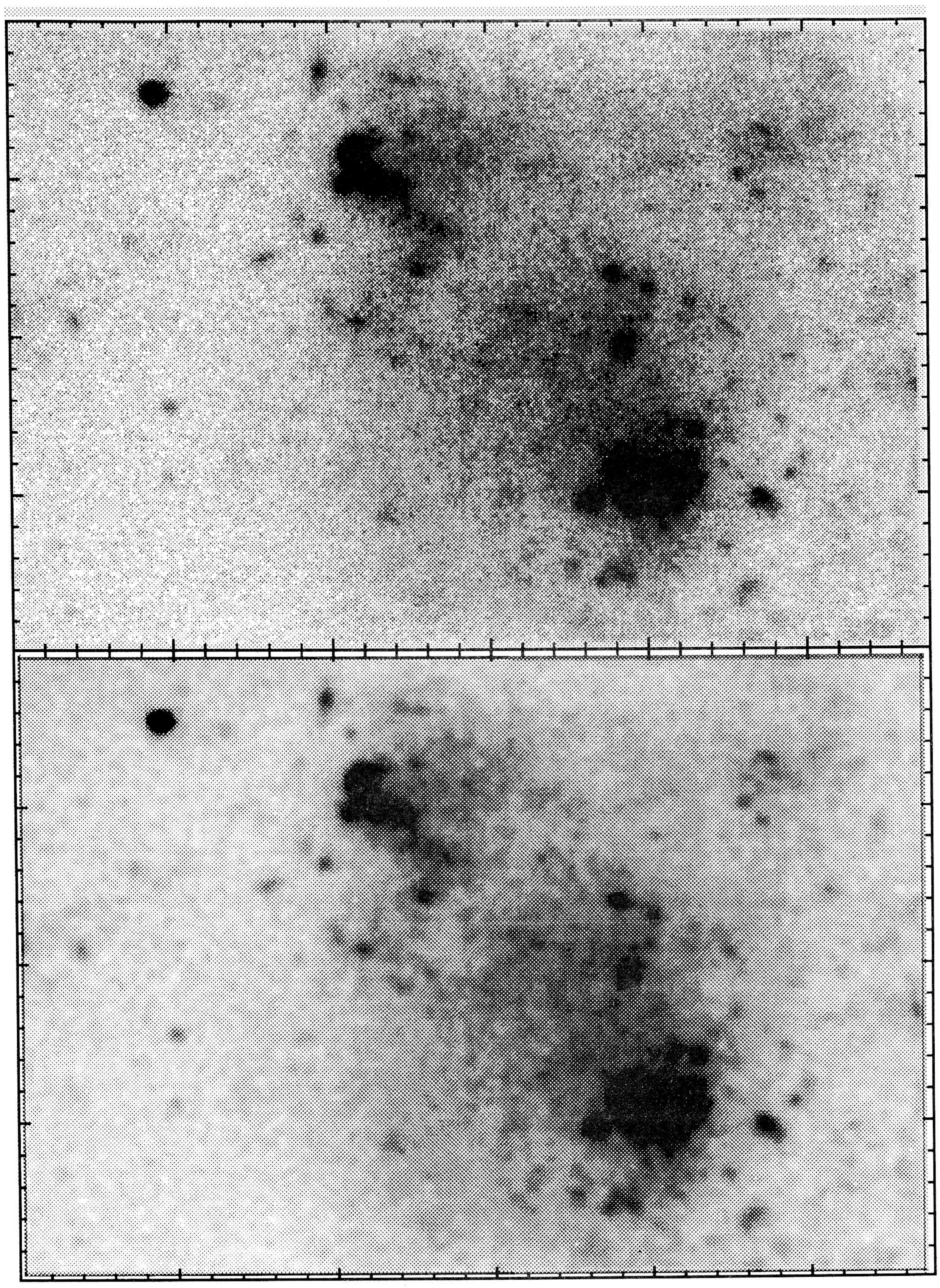

Figure 3: *Smoothing of the galaxy* DDO46, *observed with the 6 m telescope (see the text): upper part — the original, down part — the result of the program SMOOTH with the window diameter W = 9.*

Appendix

```
 /*---------------------------------------------------- SMOOTH - Fast
 regression smooth, general case, v.1.0 nov 93 (the convolute nucleus
 corresponds to 3rd or 5th degree surface)  Ts.Georgiev, Rozhen
Observatory, BG-4700 Smolyan, Bulgaria
----------------------------------------------------------*/
        #include <stdio.h>
        #include <math.h>
        #include "pcvista.h"
        #include "fits.h"
#define  MAXNC 600      /* max image width in pixels */
#define  MAXFW 15         /* max width of the window */
        #ifdef PROTO
        void main(int, char **);
        #endif
  void main (argc, argv)
int argc;  char *argv[]; { char *gotit; FITS_HANDLE finp,fout;
static int r[MAXNC], pn[MAXFW];   /* row and its pos.number */
static int  m[MAXFW][MAXNC];   /* memory for image band */
static int lw[MAXFW/2+1];/* k-limits of the circular window */
static double c[MAXFW/2+1][MAXFW/2+1]; /* convolute nucleus */
static int nr,nc,fw,ca,hw,i,ii,ic,j,k,l,lj;
static int n1,n2,n3,n4,iinp,iout,np,rr;
static double d,a2,a22,a4,c0,c1,a6,a8,a42,a62,a44,di,dj,sum,dn;
static double s1,s2,s3,s4,s5,s6,s7,s8,s9,s12,s56,s89;
static double d11,d22,d12,d10,d20,b1,b2,b3,dm=32767.;   long ls;
    fw=7;  ca=2;   if (argc < 3) {  Usage:
printf("Usage: SMOOTH inp[.fts] out[.fts] [w=W] [c=C] \n");
printf("W=2/3/4/5... - window diameter; default W=%d;\n",fw);
printf("C=0/1/2  -  cases of smoothing; default C=%d;\n",ca);
printf("cases: 0: average, 1: 3rd degree, 2: 5th degree;\n");
printf("max image width: %d, max window diam: %d",MAXNC,MAXFW);
return; }
if(argc>3){ gotit=find("w",argv[3]); fw=(int)(evaluate(gotit));}
if(argc>4){ gotit=find("c",argv[4]); ca=(int)(evaluate(gotit));}
   if(fw<2||fw>MAXFW) goto Usage;    if(ca<0||ca>2) goto Usage;
   if(ca==2&&fw<4) fw=4;
/* -- the limits lw and the area np of the circle window -- */
hw=fw/2; if(hw*2==fw) rr=hw*hw; else rr=(int)((hw+0.5)*(hw+0.5));
np=1; for (k=0; k<=hw; k++) { lw[k]=0;
for (l=0; l<=hw; l++)  if (k*k+l*l<=rr) {
lw[k]=l;  if (k>0) np+=4; } }  fw=2*hw+1;  dn=(double)np;
/* --------- THE COEFFICIENTS FOR THE CONVOLUTION --------- */
  if (ca==0) {  /* ---------- case of averaging ----------- */
for (i=0; i<=hw; i++)  for (j=0; j<=lw[i]; j++) c[i][j]=1./dn;}
  if (ca==1) {  /* ---------- case of 3rd degree ---------- */
if (fw==2) { c[0][0]=0.5; c[0][1]=c[1][0]=0.125; sum=1.; }
if (fw>2) { a2=0.; a4=0.; a22=0.;  for (i=-hw; i<=hw; i++) {
ii=i*i;  ic=abs(i);   for (j=-lw[ic]; j<=lw[ic]; j++) {
a2+=(double)ii; a4+=(double)(ii*ii); a22+=(double)(ii*j*j); } }
a2/=dn; a4/=dn; a22/=dn;  d=(a4+a22-2.*a2*a2)*dn;
c0=(a4+a22)/d;  c1=a2/d;  sum=-c0;  for (i=0; i<=hw; i++) {
ii=i*i;  for (j=0; j<=lw[i]; j++) { c[i][j] = c0-c1*(ii+j*j);
if(i==0||j==0) sum+=2.*c[i][j]; if(i!=0&&j!=0) sum+=4.*c[i][j];
if(j<=8) printf ("%8.4f", c[i][j]); } printf("\n"); }
 printf("Sum of the coefficients: %8.6f\n",sum); } }
```

```
  if (ca==2) {  /* ----------- case of 5th degree ----------- */
/* S1=<<i*i,i*i>>         S2=<<i*i,j*j>>      S3=<<i*i*j*j,i*i>>
   S4=<<i*i*j*j,i*i*j*j>> S5=<<i*i,i*i*i*i>>  S6=<<i*i,j*j*j*j>>
   S7=<<i*i*j*j,i*i*i*i>> S8=<<i*i*i*i,j*j*j*j>>
   S9=<<i*i*i*i,i*i*i*i>>
   b1 = (a2*s4-a22*s3)*d22 + (a22*s7-a4*s4)*d12;
   b2 =2.*(a22 + (a22*s3-a2*s4)*d1 + (a22*s7-a4*s4)*d2)/s4;
   b3 = (a4*s4-a22*s7)*d11 + (a22*s3-a2*s4)*d12;
   c[i,j] = 1/N - b1*(di+dj-2.*a2) - b2*(di*dj-a22)
- b3*(di*di+dj*dj-2.*a4);                          */
 a2=a4=a6=a8=0.;  a22=a42=a62=a44=0.;
 for (i=-hw; i<=hw; i++) { di=(double)(i*i); ic=abs(i);
 for (j=-lw[ic]; j<=lw[ic]; j++) { dj=(double)(j*j);
 a2+=di; a4+=di*di; a6+=di*di*di; a8+=di*di*di*di; a22+=di*dj;
 a42+=di*di*dj; a62+=di*di*di*dj; a44+=di*di*dj*dj; } }
 s1=a4-a2*a2/dn;    s2=a22-a2*a2/dn; s3=a42-a2*a22/dn;
 s4=a44-a22*a22/dn; s5=a6-a4*a2/dn;  s6=a42-a4*a2/dn;
 s7=a62-a22*a4/dn;  s8=a8-a4*a4/dn;  s9=a44-a4*a4/dn;
 s12=s1+s2; s56=s5+s6; s89=s8+s9;    a2/=dn; a22/=dn; a4/=dn;
 d11=s12*s4-2.*s3*s3; d22=s89*s4-2.*s7*s7; d12=s56*s4-2.*s3*s7;
 d=d11*d22-d12*d12;   d11/=d;    d22/=d;    d12/=d;
 d10 = d22*s3-d12*s7;     d20 = d11*s7-d12*s3;
 b1 = (a2*s4-a22*s3)*d22 + (a22*s7-a4*s4)*d12;
 b2 =2.*(a22 + (a22*s3-a2*s4)*d10 + (a22*s7-a4*s4)*d20)/s4;
 b3 = (a4*s4-a22*s7)*d11 + (a22*s3-a2*s4)*d12;    sum=0.;
  for (i=0; i<=hw; i++)    { di=(double)(i*i);
  for (j=0; j<=lw[i]; j++) { dj=(double)(j*j);  c[i][j]=1./dn;
 c[i][j]-=b1*(di+dj-2.*a2)+b2*(di*dj-a22)+b3*(di*di+dj*dj-2.*a4);
 if(i==0||j==0) sum+=2.*c[i][j]; if(i!=0&&j!=0) sum+=4.*c[i][j];
 if(j<=8) printf("%8.4f",c[i][j]); }  printf("\n"); }
 sum-=c[0][0];  printf("Sum of the coeff: %8.6f\n",sum); }
 /* ---------------------- preparation --------------------- */
 finp=fits_open(argv[1],"r",&nr,&nc); if(nc>MAXNC)goto Usage;
 fout=fits_open(argv[2],"w",&nr,&nc);            /* out.file */
 printf("%dx%d w=%d hw=%d c=%d np=%d; ",nr,nc,fw,hw,ca,np);
 if (hw*2==fw) fw=fw+1;
 /* ----------------- MAIN LOOP BY FRAME ROWS ------------- */
 for (i=-hw; i<nr+hw; i++) {
   iinp=i; /* inp.row number for the inner part of the image */
   if(i<0)   iinp=nr+i; /* input row number for upper margin */
   if(i>=nr) iinp=i-nr; /* input row number for lower margin */
   iout=i-hw;                            /* output row number */
 for(k=1;k<fw;k++) pn[k-1]=pn[k];  /* rotate the pos.numbers */
 k=i+hw;  np=k-k/fw*fw; pn[fw-1]=np; /* pos.num. of inp.row  */
 fits_get_data (finp,iinp,0,r,nc);                /* READING */
 for(j=0;j<nc;j++)m[np][hw+j]=r[j];  /* save in memory m[][] */
 for(j=0;j<hw;j++){ m[np][j]=r[nc-hw+j]; m[np][nc+hw+j]=r[j]; }
 if(i<hw) goto Next; np=pn[hw];        /* pos.num. of out.row */
 /* -------------------- LOOP IN THE ROW  ----------------- */
 for (j=hw; j<nc+hw; j++) { sum=c[0][0]*(double)m[np][j];
 for (k=1; k<=hw; k++) { lj=lw[k];  n1=pn[hw-k];  n2=pn[hw+k];
 ls=m[n1][j]; ls+=m[n2][j]; ls+=m[np][j-k];  ls+=m[np][j+k];
 sum+=c[k][0]*(double)(ls); if(k<lj) lj=k; for(l=1;l<=lj;l++){
 ls=m[n1][j-l]; ls+=m[n1][j+l]; ls+=m[n2][j-l]; ls+=m[n2][j+l];
      if (l!=k) {   n3=pn[hw-l]; n4=pn[hw+l];
 ls+=m[n3][j-k];ls+=m[n3][j+k];ls+=m[n4][j-k];ls+=m[n4][j+k];}
      sum+=c[k][l]*(double)ls; } }
```

```
     if (sum<-dm) sum=-dm;  if (sum>dm) sum=dm;
     if (sum<0.) r[j-hw]=(int)(sum-0.5);
    else r[j-hw]=(int)(sum+0.5);  }     /* end of j */
/* ------------------------ OUTPUT -------------------------- */
Next: if(iout>=0) { fits_put_data (fout,iout,0,r,nc);
if(iout/100*100==iout) printf("%d ",iout); } }  /* end of i */
fits_close (finp);  fits_close (fout);  }
```