Bull. Spec. Astrophys. Obs., 1996, 39, 124–130

Fast image processing method for PC: 1. Median smoothing and detaching of residual image

Ts. B. Georgiev

Rozhen National Observatory, BG-4700 Smolyan, Bulgaria
Visiting astronomer to SAO of the Russian AS

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

Abstract. The method of median filtering performs cleaning, smoothing and detaching of residual images. The corresponding computer program, written in the environment of Microsoft C – language, version 5.1, and software *PCVISTA* (Treffers and Richmond, 1989), is described. The program uses a circular window, whose diameter W is an input parameter. It treats the whole frame, including the periphery. The processing of big images is possible since only the horizontal band of the frame, with height W, is currently stored in the processor memory. The fast median algorithm of Huang et al. (1978) is realized. The program is many times faster than the corresponding *MIDAS* program. The text of the program is published.

Key words: aperture photometry – CCD data processing

1. Introduction

The median is a robust estimation of the mean value and is very useful in data analysis, when impulse noise is present. Initially the sliding median value was successfully applied by Tukey (1971) for smoothing economic data rows. After Huang et al. (1978) found the fast algorithm in the two-dimensional case, the method of median filtering became widely used in image processing. Fast algorithms are also described by Frieden (1980) and Korsun (1986).

The essence of median filtering in the two-dimensional case may be described as follows. A square or circle data window of size W slides across the frame row by row. Let $M(i,j)$ be the value of the current pixel on which the window is centred, and MED the mean value of the pixels in the window. Two processes are assumed under the name "median filtering": the process when MED changes $M(i,j)$ is called "median smoothing", and the process when $M(i,j) - MED$ changes $M(i,j)$ is called "extracting the residual image". Two additional processes exist if the brightness deviation threshold $D > 0$ is input: smoothing or extracting the residual image under condition $|M(i,j) - MED| > D$. In these cases the median filtering method only smoothes or detaches strong impulses, keeping the "usual" noise unchanged.

Median filtering is a highly efficient nonlinear procedure. Its main applications to astronomical image processing are as follows: i) small scale smoothing, where the cosmic events, as well as "hot" or "cool" pixels, may be removed, ii) large scale smoothing when the fundamental shape of an extended object (galaxy, comet) must be elucidated, iii) extracting the residual frame as the difference between the original frame and smoothed frame. In the last mentioned case the significant background gradient may be flattened and the contrast of the faint images may be increased.

This paper gives a description of the algorithms and practical solutions for applying the median filtering method to PC. The author's previous efforts in this direction (on computers of type PDP 11-34) were pointed out earlier (Georgiev 1987, 1990, 1991). Here the computer program *MEDFIL* is realized as an addition to the *PCVISTA* (Treffers and Richmond, 1989) package. The program may be easily changed and included in any other software.

The author believes that this detailed description of the median filtering method and the corresponding computer program will force other investigators to include this important procedure in their own software, trying to make the method and the program better.

2. General description of the image processing algorithm

Hereafter we assume that the dimensions of the processed frame are NR (number of rows) and NC (number of columns). The values of the pixels are $M(i,j)$, where $i = 0, 1, \ldots, NR-1$ and $j = 0, 1, \ldots, NC-1$. Note that i and j play the role of the Y and X coordinates of the pixels, but the Y axis is opposite directed

(downward).

The program MEDFIL is made in the compact environment of language Microsoft C, version 5.1 and *PCVISTA*. It processes FITS frames with the integer type of pixel values. Each single call of the program performs smoothing or extracting of the residual image. The arguments of the program are as follows: input file name, output file name, window diameter W, threshold of the brightness deviation D, and zero point of the histogram Z.

The input value of W may be positive or negative, but the program will apply the corresponding window using the value of $|W|$. The sign of W tells the program to output the smoothed (if "+") or residual (if "−") image.

The value of D may be zero or positive. If $D = 0$ the output of the program will depend only on W. In the case $D > 0$ the processing of each frame pixel will be done depending on the difference $DM = M(i,j) - MED$ (see Section 1). Note that if $W < 0$ and $D > 0$ a special kind of residual image will be outputted. Its pixel values will be DM if $|DM| > D$ or 0 in the other cases. Having such a frame we can find, p.e., how the median smoothing cuts the central peaks of the stellar images.

The full size of the histogram used in the program is 32767 levels. This number depends on the used program language and defines the usable brightness interval. In the present program if the pixel values are smaller than the zero point Z it is changed to be equal to Z, and if the pixel values are higher than the upper limit $Z + 32767$ it is changed to be equal to $Z + 32767$. So, if negative pixel values must be processed the user must change the zero–point of the histogram to negative.

The program $MEDFIL$ uses a preliminarily defined circle window with the diameter (full width) $FW = |W|$. The usable forms of the windows, depending on FW (odd or even), are shown in Fig.1. The "half window" value HW is assumed to be the result of integer dividing of FW by 2, so if FW is odd, $FW = 28HW + 1$. However, if FW is inputted as 'even' then after computation of the vector LW (see below) the program will use the value of $FW = |W| + 1$! The program uses the window with the help of the vector of its row limits $LW(k)$. Here $k = 1, 2, \ldots, HW$ is the relative distance of the window row from the centre of the window. Consider the last window in Fig.1, where FW=8. The right limit of the central row, where $k = 0$, is $LW(0) = 4$. The k–numbers of the rows which are below the central row are 1, 2, 3, and 4, and the $LW(k)$ coordinates of their right limits are 3, 3, 2, and 0, respectively. Such k–numbers may be used for the rows which are above the central row, where the program uses the value of $|k|$. Computation of the limits of the window $LW(k)$, its area $NPIX$ and the half of the area HLF are performed at the end of the "preliminary part" of the program $MEDFIL$. The value of HLF defines the median of the histogram.

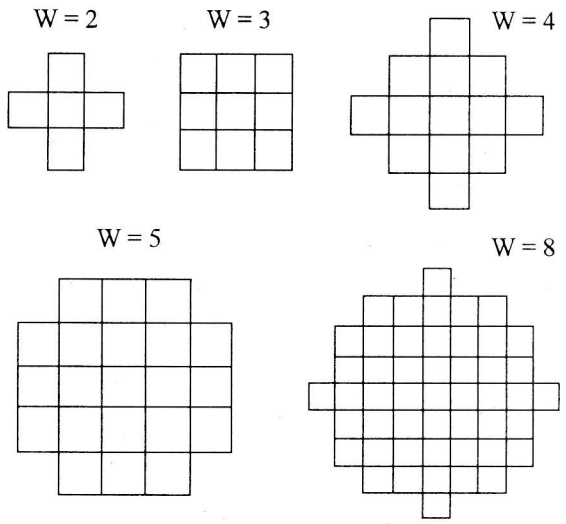

Figure 1: *The initial part of the possible data windows, used in the program MEDFIL and the programs, described in the next papers of the series. The corresponding input diameter of the window W is denoted.*

The program $MEDFIL$ treats the whole frame, including the periphery. This is why the frame must preliminarily be enlarged by adding rows or columns, taken from the opposite sides of the frame. This process is illustrated in Fig.2. So, the dimensions of the whole frame become $(NR + W - 1) \times (NC + W - 1)$ pixels. The program $MEDFIL$ processes just such a kind of image, but only a band with the number of rows equal to FW currently resides in the processor memory. The height of this "sliding" band is FW rows and its width is $NC + W - 1$ columns. The corresponding memory in the program is denoted as $M(i,j)$. One of the fixed positions of the band and the window are shown in Fig.2b.

The organization of the frame band is the most complicated part of the program. It is made at the beginning of the "main process" (see the text of $MEDFIL$). The formal number i counts the rows of the enlarged image. The counter *iinp* corresponds to the input frame rows. When $iinp < 0$ or $iinp > NR - 1$ the corresponding input rows fill the periphery added in the upper and lower part of the frame. The counter *iout* corresponds to the output rows and manages the output. The counters k and np define the position number of each input row in the memory $M(i,j)$. The vector $PN(n)$ contains the position numbers of the frame band rows in the memory band.

For example, the values of the counters in the image processing, shown in Fig.2, are given in Table 1.

Table 1: *The values of the counters of the program MEDFIL when the image with $NR = 8$, $NC = 4$, and $FW = 5$ is processed*

i	$iinp$	$iout$	k	np	$PN(0)$	$PN(1)$	$PN(2)$	$PN(3)$	$PN(4)$
−2	6	−4	0	0	–	–	–	–	0
−1	7	−3	1	1	–	–	–	0	1
0	0	−2	2	2	–	–	0	1	2
1	1	−1	3	3	–	0	1	2	3
2	2	0	4	4	0	1	2	3	4
3	3	1	5	0	1	2	3	4	0
4	4	2	6	1	2	3	4	0	1
5	5	3	7	2	3	4	0	1	2
6	6	4	8	3	4	0	1	2	3
7	7	5	9	4	0	1	2	3	4
8	0	6	10	0	1	2	3	4	0
9	1	7	11	1	2	3	4	0	1

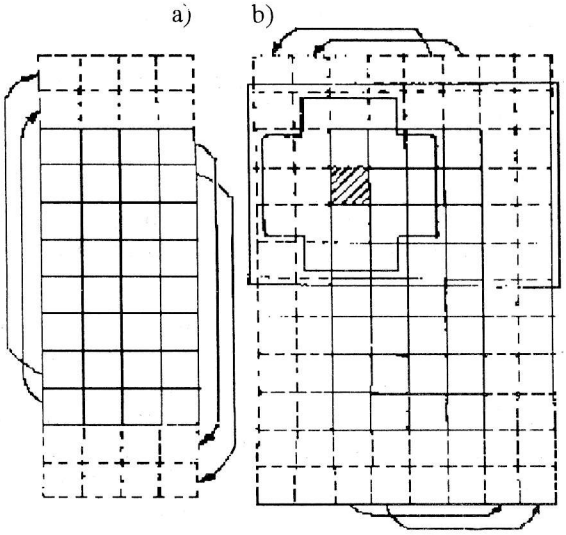

Figure 2: *Artificial enlargement and processing of the frame with dimensions 8 × 4 pixels, when the window diameter is $W = 5$ pixels. The transfer of the rows and columns is denoted by arrows: a) the original frame, and its increasing after addition of the upper and down rows, b) enlargement of the frame after addition of left and right columns. One fixed position of the frame band and the window are drawn, when the numbers of the current pixel (hatched) are $i = 2$ and $j = 1$.*

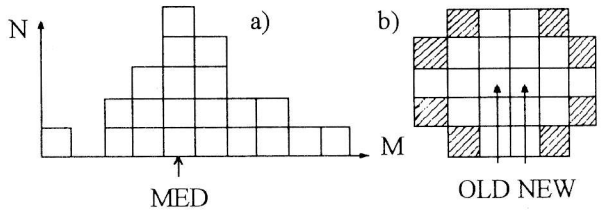

Figure 3: *Two illustrations for the fast median algorithm in the case of the window diameter $W = 5$: a) the histogram of the pixels in the window, where the abscissa is the pixel value M and the ordinate is the number of the pixels N with the value of M. In this example the median value MED corresponds to the position of the highest histogram column, b) the "old" and "new" current pixels, shown by arrows, and the peripheral pixels of the corresponding window positions (hatched). When the "new" current centre changes the "old" one, the program subtracts from the histogram the contributions of the left peripheral pixels and adds the contributions of the right ones.*

3. The fast median algorithm

The method of fast median filtering is based on three solutions (Huang et al., 1978; Huang, 1981).

Firstly, it uses the histogram of the pixel values in the window. One example is given in Fig.3a. At the beginning of the processing of each frame row the content of the histogram is turned to zero. After this, at the first position of the window in the current row, each pixel with value $n = M(i, j)$ gives a contribution of 1 to the histogram column with number n. So, the values of the pixels occur ranged and the problem of sorting the great amount of numbers is overcome.

Secondly, the fast algorithm implements partial removing and adding of the pixel values in the histogram. This process is illustrated in Fig.3b. For each new current pixel in the frame row (after the first pixel in the row), the contributions in the histogram of the pixels from the left periphery of the old window position are removed. After this the contributions of the pixels which occupy the right periphery of the new position of the window are added. In this way the number of the manipulation with the histogram is a linear function of W.

The median value of the histogram MED, cor-

responding to the first pixel in the row, is found directly. Let SUM be the sum of the contents of the histogram columns with numbers from 1 to n. Then, if $SUM < HLF$ the value of n grows and SUM increases, otherwise the median value $MED = n$ is considered as reached.

The third solution of the fast algorithm, which increases the computation speed several times, is based on the fact that the median value usually changes slowly along the frame (see Frieden, 1980). That is why the search for the current median begins from the old median. For this reason the auxiliary number SUM is used together with the removing and adding of the contributions of the peripheral pixels. The values of SUM permits the changes in the left half of the histogram. The fast median algorithm is realized in the program $MEDFIL$ in the "loop in the current row". Note that the final search for the median always goes from the low to high values.

The processing time with $FW = 61$ of an image with dimensions 400 × 580 pixels, using a 40 MHz IBM PC/AT 386 DX, is about 80 seconds. When the processor memory contains an image band up to 61 × 512 pixels the whole memory for the program is 160 kb. Using 600 kb memory and $W = 61$ pixels the maximal frame size may be about 4100 pixels.

4. One example

One application of the median filtering method is shown in the case of the extraction of the residual image.

The object is the dwarf irregular galaxy NGC 2537 = Mkn 86, observed with help of the 6 m telescope. This galaxy has a high surface brightness, and the structure of its central part is peculiar. One CCD frame in V band is used, where the seeing is about 1.2 arcsec (6 pixels) and the exposure is 600 sec. The image dimensions are 580 × 400 pixels. The standard preliminary processing of the frame, including subtraction of the bias and dark frame and flat fielding, is done by $PCVISTA$ and a few auxiliary programs of the author.

The map of the image is shown in the upper part of Fig.4a. The stars and the diffuse objects, fainter than $22^{m}.5$, are not visible on this map. Most of the images in the frame are situated against a strongly biased background.

The isophote map of the residual image, obtained from that given in Fig.4a by median filtering with $W = 31$ pixels (4.3 arcseconds), is shown at the bottom of Fig.4b. Many objects with $V > 23^{m}$ become visible.

5. Final remarks

The method of median filtering is a very useful tool for different situations in image processing. The program $MEDFIL$, which reflects the knowledge and experience of the author, may be perfected by other people and included in any software.

The author expresses gratitude to his colleagues in the Special Astrophysical Observatory (SAO) of the Russian Academy of Sciences T.N.Sokolova, A.I.Kopylov and N.A.Tikhonov for help and useful discussions, as well as to G.G.Korotkova for the help in the preparation of this paper. The author is grateful to the administration of SAO for the excellent conditions in which this work was accomplished.

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

References

Frieden B.R.: 1980, The computer in optical research. Methods and applications. Springer–Verlag. (in Russian: 1985).

Georgiev Ts.B.: 1987, PhD Thesis, Bulgarian Academy of Sciences, Sofia.

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

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

Huang T.S. (ed.): 1981, Topics in Applied Physics, 43, Two dimensional digital signal processing, Springer-Verlag, New York (in Russian: 1985).

Huang T.S., Yang G.F., Tang G.Y.: 1978, IEEE Conference on pattern recognition and image processing, Chicago.

Korsun P.P.: 1986, Kinematika i fizika nebesnykh tel, 2/2, 81.

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

Tukey J.W.: 1971, Exploratory data analysis, Addison-Wesley Publ.Co. 1977. (in Russian: 1981).

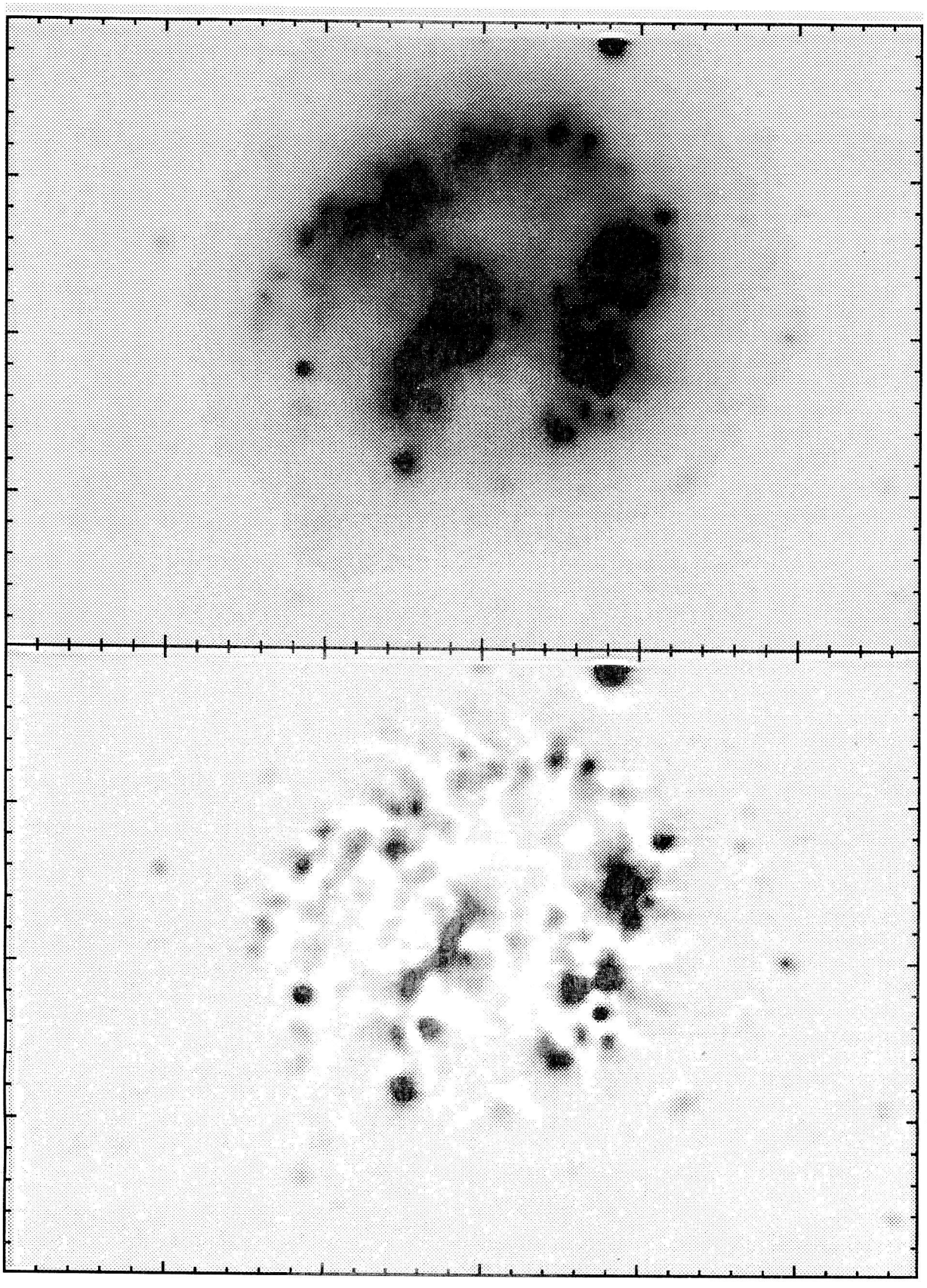

Figure 4: *Elucidation of the complicate structure of the dwarf irregular galaxy Mkn 86 using one V CCD frame from the 6 m telescope (see the text). The maps of the galaxy are given: top — original image, bottom — residual image, extracted with* $W = 31$.

Appendix

```
/*-------------------------------------------------------------
  MEDFIL: Fast median filtering, general case, v.1.0, Nov 93
  Ts. Georgiev, Rozhen Observatory, BG-4700 Smolyan, Bulgaria
  e-mail:   tsgeorg@bgearn.bitnet
-------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
        #include "pcvista.h"
#include "fits.h"
#define   MAXNC  1024   /* max image width in pixels */
#define   MAXFW  99 /* max window diameter in pixels */
#define   MAXHS  32767          /* max histogram size */
        #ifdef PROTO
        void main(int, char **);
#endif
void main (argc, argv)
int argc;  char *argv[]; { char *gotit; FITS_HANDLE finp,fout;
static int huge m[MAXFW][MAXNC];   /* the current image band */
static int huge h[MAXHS];          /* histogram in the window */
static int r[MAXNC];            /* input or output image row  */
static int pn[MAXFW];   /* number of the row in memory m[][] */
static int lw[MAXFW/2+1];     /* limits of the circle window */
int nr,nc,fw1,fw,hw,dev,d1,d2,zp,sum,med,hlf,nh,npix;
int i,iinp,iout,j,k,l,lk,n,np,npc,rr;     long numcor=0;
   fw1=-75;   dev=0;   zp=0;                        /* defaults */
if (argc < 3)  {  Usage:
printf("Usage: MEDFIL  inp_file  out_file  [w=W d=D z=Z])\n");
printf("W:window diameter, may be +- 2/3/4/5/6/7..%d;\n",MAXFW);
printf("  if W>0: the output will be the smoothed image --  \n");
printf("  the current median M changes the current pixel N; \n");
printf("  if W<0: the output will be the residual image --  \n");
printf("  the deviation N-M changes N;     DEFAULT W=%d;\n",fw1);
printf("D:deviation threshold, may be 0 or positive:        \n");
printf("  if D=0: the output depends only on the value of W;\n");
printf("  if D>0: M changes N if abs(N-M)>D; DEFAULT D=0;   \n");
printf("Z:histogram zero-point, if you want to save negative\n");
printf("  pixel value, you may enter Z<0;  DEFAULT Z=%d; \n",zp);
printf("  histogram size: %d; max image: %d pix;",MAXHS,MAXNC);
return; }
if(argc>=4){gotit=find("w",argv[3]);fw1=(int)(evaluate(gotit));}
if(argc>=5){gotit=find("d",argv[4]);dev=(int)(evaluate(gotit));}
if(argc>=6){gotit=find("z",argv[5]);zp= (int)(evaluate(gotit));}
/* ---------------------- PRELIMINARY PART ------------------ */
fw=abs(fw1);  /* fw: full width of the smooth window; fw=|W| */
if(fw<2) fw=2;  if(fw>MAXFW) fw=MAXFW;
hw=fw/2;  fw=hw*2+1;  /* hw: half-width of the smooth.window */
nh=MAXHS-2;                  /* histogram range: from 1 to nh */
d2=abs(dev); d1=-d2; /* negative and positive dev.thresholds */
/* ------------- the limits of the circle window ----------- */
hw=fw/2; if(hw*2==fw)rr=hw*hw; else rr=(int)((hw+0.5)*(hw+0.5));
npix=1; for (k=0; k<=hw; k++) { lw[k]=0;    /* window limits */
for(l=0;l<=hw;l++) if(k*k+l*l<=rr) {lw[k]=l; if(k>0) npix+=4;}}
fw=2*hw+1;   hlf=npix/2+1;    /* half of the histogram total */
/* ----------------------- MAIN PROCESS -------------------- */
finp=fits_open(argv[1],"r",&nr,&nc);  if(nc>MAXNC) goto Usage;
```

```
fout=fits_open(argv[2],"w",&nr,&nc);/* nr,nc:image dimension */
printf("%dx%d w=%d d=%d z=%d npix=%d; ",nr,nc,fw,dev,zp,npix);
  if (fw1>0) printf("smoothed image: ");
  if (fw1<0) printf("residual image: ");
/* ------------------ MAIN LOOP BY IMAGE 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 numbers */
k=i+hw; np=k-k/fw*fw; pn[fw-1]=np;  /* pos.number of inp.row */
npc=pn[hw];/* pos.number of curr.out.row for eventual output */
fits_get_data (finp,iinp,0,r,nc);   /* READING THE IMAGE ROW */
for(j=0; j<nc; j++) { r[j]-=zp;   /* zero-pointing, checking */
if(r[j]<1) r[j]=1; if(r[j]>nh) r[j]=nh; m[np][hw+j]=r[j];}
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;
/* ------------- LOOP IN THE CURRENT IMAGE ROW ------------- */
  for (j=hw; j<nc+hw; j++) {
if (j==hw) {       /* FIRST PIXEL IN THE CURRENT ROW */
for (n=0; n<MAXHS; n++) h[n]=0;  /* histogram initialization */
for (k=0; k<fw; k++)         { np=pn[k]; l=abs(k-hw); lk=lw[l];
for (l=hw-lk; l<=hw+lk; l++) { n=m[np][l];  h[n]++;  } }
med=0; sum=0; while(sum<hlf) { med++; sum += h[med]; } }/*MED*/
else {            /* OTHER PIXELS IN THE CURRENT ROW */
for (k=0; k<fw; k++) { /* along the window edge columns only */
np=pn[k]; l=abs(k-hw); lk=lw[l];
n = m[np][j-lk-1];  h[n]--;  if(n<=med)  sum--;  /* removing */
n = m[np][j+lk];    h[n]++;  if(n<=med)  sum++; }  /* adding */
while(sum >=hlf) { sum-=h[med]; med--;} /* SEARCH FOR MEDIAN */
while(sum < hlf) { med++; sum+=h[med];}            /* MEDIAN */
 }  /* end of "OTHER PIXELS" */
/* ---------- DIFFERENT KINDS OF OUTPUT IMAGES ------------- */
if(fw1>0) { if(dev==0) r[j-hw] = med+zp;          /* smoothed */
else {l=m[npc][j]-med; if(l>d1&&l<d2) r[j-hw]=l+med+zp;/*raw */
   else { r[j-hw]=med+zp; numcor++;  } } }         /* cleaned */
if(fw1<0) { if(dev==0) r[j-hw]=m[npc][j]-med+1;  /* residual */
else {l=m[npc][j]-med; if(l>d1&&l<d2) r[j-hw] = 0;   /* zero */
   else { r[j-hw]=m[npc][j]-med;  numcor++; } } }   /* peak  */
   }/* END OF j-LOOP */
Next: if(iout>=0) {fits_put_data(fout,iout,0,r,nc);/* OUTPUT */
if(iout/100*100==iout) printf("%d ",iout); }
   }/* END OF i-LOOP */
fits_close (finp);  fits_close (fout);
if(dev!=0) printf("\nNumber of corrections: %d%d",numcor);  }
```