Bull. Spec. Astrophys. Obs., 1996, 39, 140–145

Fast image processing methods for PC: 3. Partial restoration by intensified Richardson-Lucy method

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 purpose of the intensified Richardson-Lucy method (IRLM) is to reach a resolution gain of about 1.5 after 5–6 iterations. The improved image $I(n+1)$, obtained after the iteration no. n, depends only on the current image $I(n)$ and the convolution nucleus $C: I(n+1) = ((I(n)/(I(n)*C)*C) \cdot I(n)$, where the division (/) and multiplication $(\cdot)$ are done point-wise, and $(*)$ means two-dimensional convolution. The IRLM is few times faster than the classical Richardson-Lucy procedure. The IRLM is realized in the form of a C-program written in a limited version (but the fastest), when the whole frame is stored in the processor memory. The text of the program is published.

Key words: aperture photometry – CCD data processing

1. Introduction

Following the discussion of Frieden (1975), a linear and shift-invariant image formation system with additive noise, giving the observing image I_0, can be modelled mathematically by two-dimensional convolution (denoted by $*$) plus noise:

$$I_0 = O * S + N. \tag{1}$$

Here O is the object, S is the point spread function (PSF) and N is the noise. All functions are two dimensional.

Richardson (1972) and Lucy (1974) found an iterative procedure, giving a sequence of images I_n, which are improved estimations of the object O. Following Heasley (1984), we can express the iterative solution of (1) as

$$I_{n+1} = (\frac{I_0}{I_n * S} * S) \cdot I_n. \tag{2}$$

Here I_0 is the raw image, and multiplication and division are done point-wise. This is the Richardson-Lucy method (RLM). It is the subject of different discussions and development (White, 1993; Starck & Murtagh, 1994).

The RLM has been widely explored in image restoration because of its useful characteristics. White (1993) pointed them out as follows:

1. The RLM converges to the maximum likelihood solution for Poisson statistics in the data (White, 1993), which is appropriate for astronomical CCD data.

2. The RLM forces the restored image to be non-negative and conserves global and local flux at each iteration. This means that the restored image has good photometric linearity.

3. The image, restored by the RLM, is robust against small deviation of the used PSF toward true PSF.

4. The RLM requires significant, but manageable, computer time, representing a reasonable compromise between quick (but unsatisfactory) methods, based on Wiener filtering, and much slower (but usually superior) maximum entropy method.

2. The intensified Richardson–Lucy method

Our experiments with procedure (2) show that:

1. In the case of a relatively low ratio of signal to noise (between 50 and 200), we can really make only partial restoration. This means that the restoration gain (RG), the relative decrease in the full width at half maximum (FWHM) of the stellar images, may be about 1.5.

2. An analysis of the procedure (2) shows that each iteration amplifies the faint images which are in the same spatial scale as the PSF. So, bearing in mind (1), we conclude that the true PSF may be changed

by one two-dimensional Gaussian with suitable width.

3. We found that the first iteration which occurs is the most efficient. Its RG is usually 1.05–1.10, when the RG of the next iteration is 2–3 times lower and the process is slowly convergent. It should be noted that the result of the first convolution is an image with pixel values close to 1. We are forced to store it in the integer memory after subtraction of 1 and multiplication by 10,000. However, the Gaussian approximation of the PSF and the arithmetic errors, when we use only integer memory, probably decreases the speed of the RLM convergence.

The upper mentioned considerations lead us to the intensified Richardson-Lucy method (IRLM), where each next image I_{n+1} depends only on the previous image I_n and the PSF, modelled by the Gaussian G:

$$I_{n+1} = (\frac{I_n}{I_n * G} * G) \cdot I_n. \qquad (3)$$

Usually the RG of the IRLM grow up with increasing iteration number, i.e. the process (3) is non-convergent. However, when the number of the iteration is only 3–5, the RG of the IRLM is 1.3–1.6. Thus, the IRLM is a number of times faster than the RLM. Moreover, the IRLM may be easily realized as a program, where only one input/output image-file is necessary, where the computer memory may store the intermediate image.

3. The algorithm and program IRLM

The program IRLM is created in the environment of the language C 5.1 of Microsoft. Its input parameters are an input/output integer file, a desired number of iterations, a diameter of the convolution window, a seeing (that will be used as "2.sigma" for the Gaussian), and coordinates and a size of the stellar area for estimation of the RG. The text of the program is given as an Appendix to this paper. A description of how to use the window limits and the fast convolution procedure, based on the octal symmetry of the convolution nucleus, is given in the previous papers of Georgiev (1996a,b).

The algorithm of the program IRLM is given below: some of the denotations are used as follows: Fn is the input/output file, In is the image that is current in the iteration no. n, Im is the image that is currently stored in the computer memory, Ic is the current result of the convolution, and m[ ][ ] is the computer memory that is able to save one whole image.

Note that the dimension of m[ ][ ] must be $(NR + W - 1) \times (NC + W - 1)$ pixels, where NR and NC are the numbers of the rows and columns of the processed image, and W is the diameter of the circle convolute window. The additional periphery — $W/2$ rows or columns by each side of the image — is necessary for processing the whole image. More detailed descriptions of the image processing principals used in the program IRLM (as in the previously published programs of the author) are given in the papers of Georgiev (1996a,b).

The sequence of actions in the program IRLM may be presented as follows.

3.1. Preparation

- Open the file Fn for reading;
- write the content of Fn in the left upper part of m [ ][ ], as Im;
- close the file Fn;
- initialize the iteration number: n=0.

3.2. Convolutions

- Increase the iteration number: n=n+1;
- shift the image In to the central part of the memory m[ ][ ];
- fill the peripheral margins, using the rows and columns from the opposite side of the image;
- open the file Fn for writing;
- read one row of the image In from Fn for the future division or multiplication;
- make the convolution for each pixel: Im * C = Ic;
- Case of convolution 1: Im = (In/Ic-1)·10000;
- Case of convolution 2: Im = (Ic/10000+1)·In;
- write Im in Fn;
- close the file Fn;
- else — end of the process.

One important distinction of the IRLM is that the intermediate image Im is stored currently in the memory m[ ][ ] when the convolution process goes on. It is possible, even in the use of square convolution window. Note that when the program finishes the computation (p.e.) of the first pixel $(W/2, W/2)$ of the image Im (centred in m[ ][ ]), the first pixel (0,0) of the memory m[ ][ ] is already free for storing! For this reason after each convolution the image Im is centred.

The processing time for one IRLM iteration with $W = 35$, if the image dimensions are 400×580 pixels and a 40 MHz IBM PC/AT 386 DX is used, is about 23 min. The whole memory used by the IRLM program is 560 kb.

4. One example

An example of applying IRLM is given in Fig.1. The object is the dwarf irregular galaxy UGC 4115 twice observed by CCD on the 6-m telescope in the B band. The scale is 0.2 arcsec by pixel, the exposure times

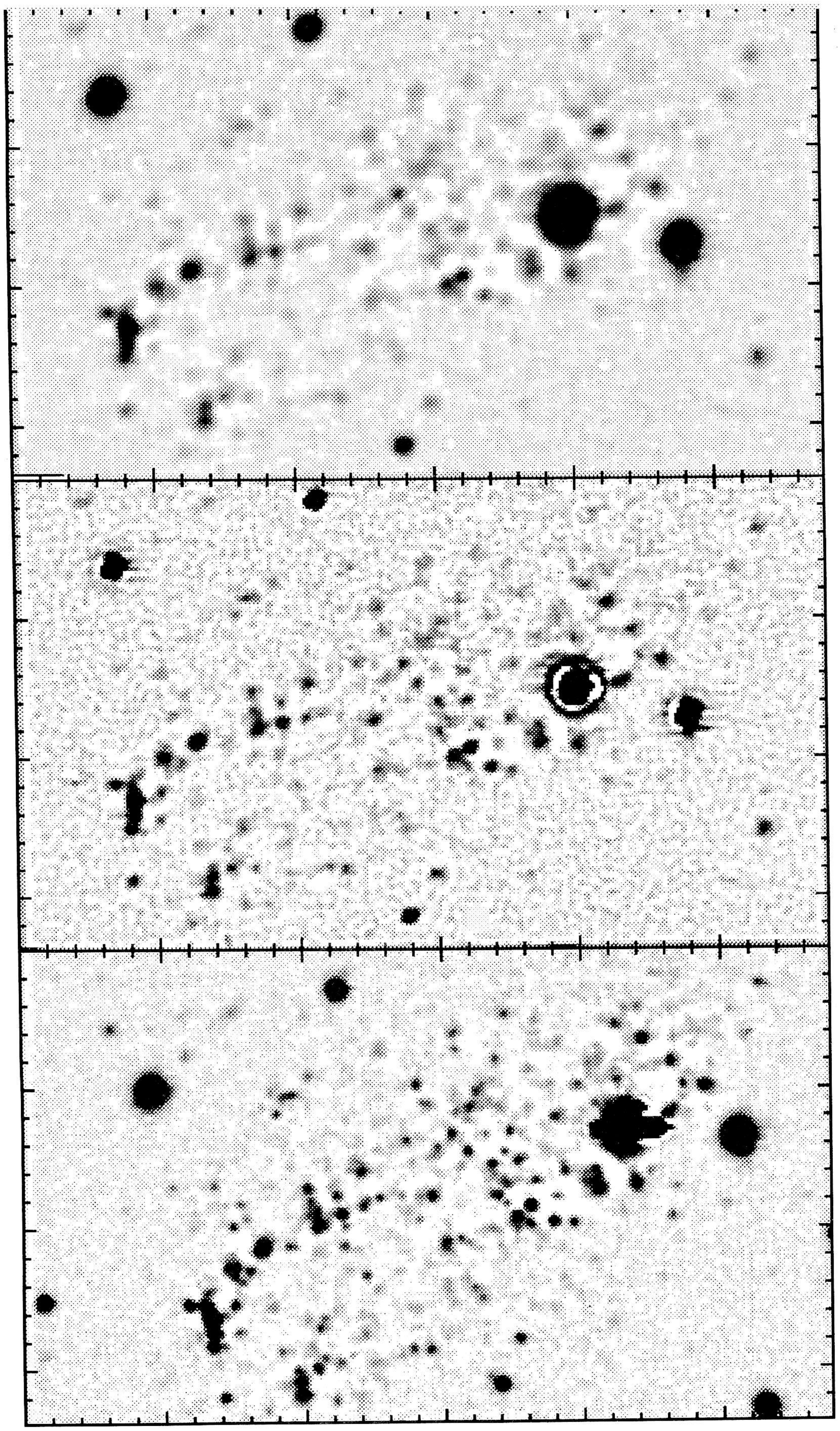

Figure 1: *Application of the IRLM on the galaxy UGC 4115. Upper part: raw image with FWHM 2.1 arcsec; middle part: restored image where the FWHM is 1.6 times better; down part: raw image with FWHM of 1.0 arcsec.*

are 600 s, and the seeings are about 2.1 and 1.0 arcsec. Standard processing of the images (bias and dark frame subtraction, flat fielding, etc., including the mapping) is done with help of the *PCVISTA* (Treffers, Richmond, 1989) and some auxiliary programs of the author. The IRLM demonstration is made with help of the residual images, obtained with the median filtering program MEDFIL (Georgiev, 1996a), and is smoothed with the fast regression method, realized in the program SMOOTH (Georgiev, 1996b).

The "raw" and restored images are given in the upper and middle parts of the Fig.1, respectively. The RG is about 1.6 times. Another CCD image under seeing of 1 arcsec is given in the down part of Fig.1.

5. Concluding remarks

It should be noted, that the IRLM, like the original RLM, conserves the shape of the objects, if they are different from the shape of PSF. The IRLM is several times faster than the classical RLM procedure, so it turns out to be a very useful tool for partial restoration. The author considers that the IRLM, as well as the program IRLM, can be developed and included in any software.

It should be especially emphasized that the circular window, corresponding to the shapes of the objects in the frame, gives better results in the IRLM than the square one. That is why, we do not use the fast separable convolution where the image can be convoluted into two one dimensional convolutions (Heasley, 1984).

The author is grateful to Dr.I.D. Karachentsev, Dr. N.A. Tikhonov and Dr. A.I. Kopylov of the Special Astrophysical Observatory (SAO) of the Russian Academy of Sciences for their discussions, to G.G. Korotkova for the help in the preparation of this paper, as well as to the administration of SAO for the excellent conditions in which this work was finished.

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.: 1975, in Picture Processing and Digital Filtering, ed.: T.S. Huang, Springer–Verlag. (in Russian: 1979).
Georgiev Ts.B.: 1996a, Bull. Spec. Astrophys. Obs., **39**, 124, (this issue).
Georgiev Ts.B.: 1996b, Bull. Spec. Astrophys. Obs., **39**, 131, (this issue).
Heasley J.N.: 1984, Publ. Astr. Soc. Pacific, **96**, 767.
Lucy L.B.: 1974, Astron. J., **79**, 745.
Richardson B.H.: 1972, J.Opt.Soc.Am., **62**, 55.
Starck J.-L., Murtagh F.: 1994, ESO scientific preprint No.978.
Treffers R.R., Richmond M.W., 1989, Publ. Astr. Soc. Pacific, **101**, 725.
White R.L.: 1993, Restoration, Newsletter of STScI's Image Restoration Project, 1,11.

Appendix

```
/* ---- IRLM - Intensified Richardson-Lucy
  Method, v.1, Feb 94 Purpose: partial image restoration  J = ((I/(I*c))*c).I
  where I and J are the images before and after iteration;
  "." and "/" mean point-wise multiplication and division;
  "*"  means convolution; "c" is a gauss' convolution nucleus
  Georgiev Ts.B., Bull. SAO, v.39;  tsgeorg@bgearn.bitnet
  Rozhen Observatory, BG-4700 Smolyan, Bulgaria  ------------ */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define   MAXNR  300     /* max image lenght in pixels */
#define   MAXNC  200     /* max image width  in pixels */
#define   MAXFW  15             /* max window diameter */
  void main (argc, argv)
int argc;  char *argv[]; { FILE *fn;
static int huge m[MAXNR+MAXFW-1][MAXNC+MAXFW-1];   /*image Im */
static int r[MAXNC],lw[MAXFW/2+1];  /*image row,window limits */
static double c[MAXFW/2+1][MAXFW/2+1];   /* convolute nucleus */
static int nr,nc,i,j,k,np,fw,hw, l,ll,it,ni,conv;    long ls;
double s,ss,c1,c2,rad,gw, dl=32767.,dc=10000.,pi=3.14159;
   ni=5;   gw=4.;  fw=MAXFW;                       /* DEFAULTS */
if (argc<4) { Use: printf(
"Usage:  IRLM  file  nrows  ncols  niter  sgaus  wdiam\n");
printf("Intensified Richardson-Lucy Method, v.1, Feb 94;\n");
printf("file: integer image-file without heder;\n");
printf("nrows & ncols: lenght and width of the image (pix);\n");
printf("niter: desired number of iterations,   DEF: %d;\n",ni);
printf("sgaus: gaussian width(pix),(0.8*FWHM), DEF: %2.1f\n",gw);
printf("wdiam:  window  diam.(pix),( 10*FWHM), DEF: %d\n",fw);
printf("  max image: %dx%d;  max window: %d",MAXNR,MAXNC,MAXFW);
  return; }
nr=atoi(argv[2]);  nc=atoi(argv[3]);  if(argc>4) ni=atoi(argv[4]);
if(argc>5) gw=atof(argv[5]);  if(argc>6) fw=atoi(argv[6]);
if(nr>MAXNR||nc>MAXNC||fw>MAXFW) goto Use;
/* ----------- Limits of the circle convolute nucleus ------- */
hw=fw/2; if(hw*2==fw) ll=hw*hw; else ll=(int)((hw+.5)*(hw+.5));
np=1; for (k=0; k<=hw; k++) { lw[k]=0;  for (l=0; l<=hw; l++)
if(k*k+l*l<=ll) { lw[k]=l; if (k>0) np+=4; } }  fw=2*hw+1;
/* ----------- Gaussian nucleus for partial restoration ----- */
s=gw/2.;   c1=1./s/2./pi;   c2=-1./2./s;   ss=0.;
for (k=0; k<=hw; k++) {  for (l=0; l<=hw; l++) c[k][l]=0.;
for (l=0; l<=lw[k]; l++) {  rad=(double)(k*k+l*l);
c[k][l]=c1*exp(c2*rad); if(l!=0)ss+=4.*c[k][l];}} ss+=c[0][0];
     s=0.; for (k=0; k<=hw; k++)  for (l=0; l<=lw[k]; l++) {
     c[k][l]/=ss; if(l!=0) s+=4.*c[k][l]; } s+=c[0][0];
/* printf("Coef: init.sum = %6.4f;  morm.sum = %6.4f\n",ss,s);
   for (k=0; k<=hw; k++) { for (l=0; l<=lw[k]; l++)
   if (l<9) printf ("%7.4f ",c[k][l]);  printf("\n"); } */
/* ------------ Writing the inp.file in m[][] as Io --------- */
fn = fopen (argv[1],"rb");  for(i=0; i<nr; i++) {
fread (r,2,nc,fn); for(j=0; j<nc; j++) m[i][j]=r[j];} fclose (fn);
printf("%dx%d ni=%d gw=%3.2f fw=%d np=%d; R E S T O R A T I O N :",
nr,nc,ni,gw,fw,np);
/* -------------------- CONVOLUTIONS  1 and 2 --------------- */
it=0;   Next_iter:  it++;                  /* iteration number */
if (it/2*2!=it) conv=1;  else conv=2;   /* convolution number */
if (conv==1)  printf("\niter %d ",it/2+1);
/* ----------- centering the image in the memory m[][] ------ */
for(i=nr-1;i>=0;i--)for(j=nc-1;j>=0;j--)m[i+hw][j+hw]=m[i][j];
/* - enlarging the periphery by adding left and right margins */
for (i=hw; i<nr+hw; i++)  for (j=0; j<hw; j++)  {
m[i][hw-j] = m[i][nc+j];    m[i][nc+hw+j] = m[i][hw+j];  }
/* - enlarging the periphery by adding upper and down margins */
for (i=0;i<hw;i++)  for (j=0; j<nc+fw-1; j++)  {
m[i][j] = m[nr+i][j];   m[nr+hw+i][j] = m[hw+i][j]; }
/* ----------------------- convolution ---------------------- */
fn = fopen (argv[1],"rb");
for (i=hw; i<nr+hw; i++) {           /* making convoluted image */
fread (r,2,nc,fn);            /* geting image row from output file */
for (j=hw; j<nc+hw; j++) {  s=c[0][0]*(double)m[i][j];
for (l=1; l<=hw; l++) {       /* 4 symetric pixels on the axis */
ls=m[i-l][j];  ls+=m[i+l][j];  ls+=m[i][j-l];  ls+=m[i][j+l];
   s += c[l][0]*(double)ls;     ll=lw[l]; if(l<ll) ll=l;
for (k=1; k<=ll; k++) {      /* 4 symetric pixels on diagonals */
ls=m[i-l][j-k];ls+=m[i-l][j+k];ls+=m[i+l][j-k];ls+=m[i+l][j+k];
    if (k!=l) {  /* +4 symetric pixels in other cases */
ls+=m[i-k][j-l]; ls+=m[i-k][j+l];
ls+=m[i+k][j-l]; ls+=m[i+k][j+l]; }
       s += c[l][k]*(double)ls; } }    /* end of k and l-loops */
if (conv==1) { /* making In/Ic and saving the result in m[][] */
  s=((double)r[j-hw]/s-1.)*dc; if(s<-dl) s=-dl; if(s>dl) s=dl;
  if (s<0.)  m[i-hw][j-hw]=(int)(s-.5);
       else  m[i-hw][j-hw]=(int)(s+.5); }
if (conv==2){  /* making In.Ic and saving the result in m[][] */
  s=(double)r[j-hw]*(s/dc+1.); if(s<-dl) s=-dl; if(s>dl) s=dl;
  if (s<0.)  m[i-hw][j-hw]=(int)(s-.5);
       else  m[i-hw][j-hw]=(int)(s+.5);
      r[j-hw]=m[i-hw][j-hw]; } }            /* END OF j */
if((i-hw)/100*100==(i-hw))printf("%d ",i-hw); }   /* END OF i */
  fclose (fn);
if (conv==2) {              /* saving the result in the out.file */
  fn = fopen (argv[1],"wb");  for (i=0; i<nr; i++ )  {
  for (j=0; j<nc; j++)  r[j]=m[i][j];
  fwrite (r,2,nc,fn);  }  fclose (fn);  }
    if (it/2<ni) goto Next_iter;  }
```