

# Image Processing Elaborazione Immagini in Fortran

Andrea De Paoli

Astrophysics project - 1996

## Contents

<b>1</b>	<b>Introduction (English section)</b>	<b>2</b>
<b>2</b>	<b>Summary of source code</b>	<b>2</b>
2.1	pass 1, Processing (Elaborazione) . . . . .	2
2.2	Image filters . . . . .	2
2.3	Image sums . . . . .	2
<b>3</b>	<b>Algorithms</b>	<b>2</b>
3.1	somma1.for . . . . .	2
3.2	somma2.for . . . . .	3
3.3	somma3.for . . . . .	3
3.4	Special algorithms to handle partial matrices . . . . .	3
<b>4</b>	<b>Introduzione (sezione in italiano)</b>	<b>3</b>
<b>5</b>	<b>Sommario dei codici sorgenti</b>	<b>4</b>
5.1	pass 1, Elaborazione . . . . .	4
5.2	Filtri immagini . . . . .	4
5.3	Somma immagini . . . . .	4
<b>6</b>	<b>Algoritmi</b>	<b>4</b>
6.1	somma1.for . . . . .	4
6.2	somma2.for . . . . .	4
6.3	somma3.for . . . . .	4
6.4	Algoritmo particolare per gestire matrici parziali . . . . .	5
<b>7</b>	<b>source codes</b>	<b>5</b>
7.1	somma1.for . . . . .	5
7.2	somma2.for . . . . .	11
7.3	somma3.for . . . . .	13

## 1 Introduction (English section)

(This version is lacking images which I will include asap.)

This was a project which I developed to do image processing. The problem was to “clean” pictures of globular clusters. The algorithms below proved satisfactory in matching various pictures and then using filtering techniques “cleaning” them (removing noise) and summing them. Unfortunately the technique is not general and only works for globular clusters (even other astronomical images were not as satisfactory).

The request was that it be in Fortran and run on a PC (DOS at the time). But despite the limitations I think this may be of interest, I would especially like to know if anyone can use this code, perhaps adapt it. I have not had the time since then to translate it into C, even though that was the original intention.

We had a CCD camera available, and we also used a public domain program to view the images so there is, I’m sorry to say, no code to view the images.

## 2 Summary of source code

### 2.1 pass 1, Processing (Elaborazione)

1. Find min and max brightness points in image `pixel1.for`
2. ( determine “burnt” (overflow) pixels `pixel2.for`
3. substitute burnt pixels *excepting border pixels* `pixel3.for`

### 2.2 Image filters

1. low band filter (filtro passa basso) `smooth.for`  
1 1 1  
1 4 1  
1 1 1
2. high pass filter (filtro passa alto) `smooth2.for`  
0 -1 0  
-1 20 -1  
0 -1 0
3. custom filter (filto da introdurre) `smooth3.for`

### 2.3 Image sums

Overlaps two images and sums the pixels, the sources are `somma1.for` `somma2.for` `somma3.for`

## 3 Algorithms

### 3.1 `somma1.for`

The first program, using the first image (call it **a0.img**) as a reference, determines the coordinates of the brightest stars and, using the same criteria, does

the same for the other images (say **a1.img**). It then determines which stars correspond in the various images and calculates their coordinate difference  $\Delta x$  and  $\Delta y$  so that the images can be shifted accordingly and matched up, and writes this *shift* value corresponding to image *a1* into the file **a1.dat**

### 3.2 **somma2.for**

The second program reads the original file **a1.img** and rewrites it shifted by  $\Delta x$  and  $\Delta y$  into the output file **a1.out** using the values obtained from file **a1.dat** generated by the program **somma1**.

### 3.3 **somma3.for**

Reads the reference image file **a0.img** and the shifted image file **a1.out**, and then “sums” them into an output file called **a0.new**. This procedure is then repeated with other *shifted* images.

### 3.4 **Special algorithms to handle partial matrices**

An algorithm for using a temporary matrix, *buffer matrix*, when due to memory limitations, it is not possible to read all of the lines of a matrix.

Variables:

*m* number of lines in original (input) matrix

*n* = *m* - 1 tail of the buffer matrix

*n* + *ℓ* number of rows in matrix buffer. It is best to choose the number to be a divisor of the number of lines in the original matrix.

**pass 1** read *n* + *ℓ* and copy them into buffer matrix

**begin loop**

**pass 2** execute calculations with an *m* row matrix from row 1 to row *n* + *ℓ* - *m*, that is until the last scanned matrix row reaches the last buffer matrix row.

**pass 3** Copy the last *n* rows from the end of the buffer to the beginning of the buffer itself (moving the tail end to the head of the matrix)

**pass 4** read *ℓ* lines from the original matrix and copy them after the *n* rows at the head of the buffer. Exit if finished, otherwise go back to beginning of loop (pass 2).

## 4 **Introduzione (sezione in italiano)**

Questo progetto di elaborazioni immagini serviva a “pulire” foto di ammassi globulari. Gli algoritmi i rivelarono soddisfacenti nell’applicazione ma purtroppo non generalizzabile per altri tipi di immagini.

Fu richiesto il programma in Fortran a doveva “girare” su un PC (sotto DOS). Gradirei sapere se a qualcuno il codice è risultato utile.

Le foto erano fatte con ana macchina CCD e il programma per visualizzare le immagini era di dominio pubblico.

## 5 Sommario dei codici sorgenti

### 5.1 pass 1, Elaborazione

1. determinazione massimo e minimo punto di brillantezza immagine `pixel1.for`
2. conteggio pixels “bruciati” (sovraesposti) `pixel2.for`
3. sostituzione pixels bruciati interni all’immagine *i pixels ai bordi non possono essere eliminati* `pixel3.for`

### 5.2 Filtri immagini

1. low band filter (filtro passa basso) `smooth.for`  
1 1 1  
1 4 1  
1 1 1
2. high pass filter (filtro passa alto) `smooth2.for`  
0 -1 0  
-1 20 -1  
0 -1 0
3. custom filter (filto da introdurre) `smooth3.for`

### 5.3 Somma immagini

Effettua la sovrapposizione e la somma di due immagini), tle sorgenti sono `somma1.for` `somma2.for` `somma3.for`

## 6 Algoritmi

### 6.1 `somma1.for`

Il primo programma, usando il file `a0.img` (per es.) determina le coordinate delle stelle più luminose, e opi esegue lo stesso algoritmo per gilaltri file immagine (per es. `a1.img`). In seguito determina quali stelle corrispondono fra le diverse immagini e calcola le differenze fra le loro coordinate  $\Delta x$  e  $\Delta y$  e scrive il risultato dello spostamento *shift* corrispondente all’immagine *a1* nel file `a1.dat`.

### 6.2 `somma2.for`

Il secondo programma legge il file originale `a1.img` e lo riscrive spostato di  $\Delta x$  e  $\Delta y$  nel file di output `a1.dat` prodotto da `somma1`.

### 6.3 `somma3.for`

Legge il file di riferimento `a0.img` e il file con l’immagine già spostata dal file `a1.out`. Le “somma” in un file di output `a0.new`. Si ripete il procedimento co eventuali altri file già shiftati. La parte elaborata sarà solo l’intersezione di tutte le immagini spostate.

## 6.4 Algoritmo particolare per gestire matrici parziali

Un algoritmo per utilizzare una matrice di appoggio temporanea, *matrice buffer*, nel caso non si possa usare una matrice originale intera (per limitazioni sul numero di righe), e poter quindi scorrere comunque con una matrice di un numero piccolo di righe la matrice originale.

Variabili:

$m$  numero di righe della matrice (originale) di scorrimento

$n = m - 1$  la “*coda*” della matrice buffer

$n + \ell$  numero di righe della matrice buffer. È opportuno scegliere il numero del buffer che sia un divisore del numero di righe della matrice originale.

**passo 1** si leggono le  $n + \ell$  righe e si copiano nella matrice buffer.

**inizio loop**

**pass 2** eseguire i calcoli scorrendo con una matrice di  $m$  righe da riga 1 fino a  $n + \ell - m$ , cioè fino a che l’ultima riga della matrice di scorrimento arriva all’ultima riga del buffer.

**pass 3** si copiano le ultime  $n$  righe dalla fine del buffer nell’inizio del buffer stesso (così che la coda diventa la testa)

**pass 4** si leggono  $\ell$  righe dalla matrice originale e si copiano a seguire le  $n$  in testa al buffer. Se si ha finito si esce altrimenti si torna all’inizio del *loop* (passo 2).

## 7 source codes

### 7.1 somma1.for

```
program SOMMA1
C -----
C -- i pixel sono lunghi 2 byte ciascuno, quindi usare
C -- integer*2, ma appoggiarsi ad una variabile di comodo
C -- di integer.
C --
C -- versione 1.0    08 August, 1996
C -----
      integer*2    pxl (1:405,1:576)
      integer     y, x, ind, valpix
      integer     maxb(1:3), mxb(1:3), myb(1:3)
      integer     maxc(1:3), mxc(1:3), myc(1:3)
      character*12 nmfileb
      character*12 nmfilec
      character*12 nmfilef
      integer     l, delx, dely
      real        mediax,mediay
      integer     media, somma, shiftx, shifty
```

```

C ----- inizializzazione
    do 2450, ind = 1, 3
        maxb(ind)=0
        mxb(ind)=0
        myb(ind)=0
        maxc(ind)=0
        mxc(ind)=0
        myc(ind)=0
2450 continue

        l = 3

C ----- ottenere il nome file dall'utente

        write(*,*) ' '
        write(*,*) ' Programma per calcolare la somma di due immagini'
        write(*,*) ' '
        write(*,*) 'Scrivere il nome del file di base',
-           ' (senza estensione):'
        read(*,1000) nmfileb
        write(*,*) 'Scrivere il nome del file di confronto ',
-           ' (senza estensione):'
        read(*,1000) nmfilec
        write(*,*) 'Scrivere il nome del file finale ',
-           ' (senza estensione):'
        read(*,1000) nmfilef

1000 format(A)

c-----creazione file util

        open(unit=10, file='utilb', err=8300,status='unknown')
        write(10,*)nmfileb
        close (10)

        open(unit=11, file='utilc', err=8300,status='unknown')
        write(11,*)nmfilec
        close (11)

        open(unit=12, file='utilf', err=8300,status='unknown')
        write(12,*) nmfilef
        close (12)

C ----- apertura files
        open(unit=20, FILE=nmfileb//'.img', err=8000, status='OLD',
& form='BINARY')

        open(unit=30, FILE=nmfilec//'.img', err=8100, status='OLD',
& form='BINARY')

```

```

open(unit=40, FILE=nmfilec//'.dat', err=8200, status='unknown')

C -----
C -- rilevazione delle tre box piu' brillanti distanti
c -- aumento l tra di loro del file base
C -----

C ----- lettura del file di base
do 2005 y = 1, 576
do 2000 x = 1, 405
read (20) pxl(x,y)
2000 continue
2005 continue

close (unit=20, status='KEEP')

C ----- RICERCA DELLA box 3x3 PIU' BRILLANTE

do 2025, y = 2, 575
do 2020, x = 2, 404

valpix = 0

do 2040, delx = -1, 1
do 2050, dely = -1, 1
valpix = valpix + pxl(x+delx, y+dely)
2050 continue
2040 continue

if (maxb(1) .LE. valpix) then
maxb(1) = valpix
mxb(1) = x
myb(1) = y
endif

2020 continue
2025 continue

C ----- RICERCA DELLA SECONDA

do 2125, y = 2, 575
do 2120, x = 2, 404

valpix = 0

do 2140, delx = -1, 1
do 2150, dely = -1, 1
valpix = valpix + pxl(x+delx, y+dely)
2150 continue
2140 continue

```

```

        if ((maxb(1)      .GE. valpix) .and.
&         (maxb(2)      .LE. valpix) .and.
&         (abs(x - mxb(1)) .GE. 1)   .and.
&         (abs(y - myb(1)) .GE. 1)   ) then
            maxb(2) = valpix
            mxb(2)  = x
            myb(2)  = y
        endif

2120     continue
2125     continue

C ----- RICERCA DELLA TERZA

        do 2225 , y = 2, 575
          do 2220, x = 2, 404

            valpix = 0
            do 2240, delx = -1, 1
              do 2250, dely = -1, 1
                valpix = valpix + pxl(x+delx, y+dely)
2250         continue
2240         continue

            if ((maxb(2)      .GE. valpix) .and.
&         (maxb(3)      .LE. valpix) .and.
&         (abs(x - mxb(1)) .GE. 1)   .and.
&         (abs(y - myb(1)) .GE. 1)   .and.
&         (abs(x - mxb(2)) .GE. 1)   .and.
&         (abs(y - myb(2)) .GE. 1)   ) then
                maxb(3) = valpix
                mxb(3)  = x
                myb(3)  = y
            endif

2220     continue
2225     continue

C -----
C -- rilevazioni delle tre box dal file di confronto
C -----

C ----- lettura del file di input

        do 3005 y = 1, 576
          do 3000 x = 1, 405
            read (30) pxl(x,y)

```

```
3000 continue
3005 continue
```

```
close (unit=30, status='KEEP')
```

```
C ----- RICERCA DELLA PIU' BRILLANTE
```

```
do 3025, y = 2, 575
do 3020, x = 2, 404
```

```
valpix = 0
do 3040, delx = -1, 1
do 3050, dely = -1, 1
valpix = valpix + pxl(x+delx, y+dely)
3050 continue
3040 continue
```

```
if (maxc(1) .LE. valpix) then
maxc(1) = valpix
mxc(1) = x
myc(1) = y
endif
```

```
3020 continue
3025 continue
```

```
C ----- RICERCA DELLA SECONDA
```

```
do 3125, y = 2, 575
do 3120, x = 2, 404
```

```
valpix = 0
do 3140, delx = -1, 1
do 3150, dely = -1, 1
valpix = valpix + pxl(x+delx, y+dely)
3150 continue
3140 continue
```

```
if ((maxc(1) .GE. valpix) .and.
& (maxc(2) .LE. valpix) .and.
& (abs(x - mxc(1)) .GE. 1) .and.
& (abs(y - myc(1)) .GE. 1) ) then
maxc(2) = valpix
mxc(2) = x
myc(2) = y
endif
```

```
3120 continue
3125 continue
```

```

C ----- RICERCA DELLA TERZA

do 3225 , y = 2, 575
do 3220, x = 2, 404

    valpix = 0
do 3240, delx = -1, 1
do 3250, dely = -1, 1
    valpix = valpix + pxl(x+delx, y+dely)
3250     continue
3240     continue

    if ((maxc(2)      .GE. valpix) .and.
&      (maxc(3)      .LE. valpix) .and.
&      (abs(x - mxc(1)) .GE. 1)   .and.
&      (abs(y - myc(1)) .GE. 1)   .and.
&      (abs(x - mxc(2)) .GE. 1)   .and.
&      (abs(y - myc(2)) .GE. 1) ) then
        maxc(3) = valpix
        mxc(3)  = x
        myc(3)  = y
    endif

3220     continue
3225     continue

```

```

C -----
C -- calcolo dello shift dell'immagine da conforntare
c----rispetto all'immagine di base
C -----

```

```

C ----- calcolo del DELTA

media = 0
somma = 0

do 3660, ind = 1, 3
    if (ABS(mxb(ind) - mxc(ind)) .LT. 10) then
        media = media + REAL(mxb(ind) - mxc(ind))
        somma = somma + 1
    endif
3660 continue

mediax = media/ somma
shiftx = NINT(mediax)

media = 0
somma = 0

do 3760, ind = 1, 3

```

```

        if (ABS(myb(ind) - myc(ind)) .LT. 10) then
            media = media + REAL(myb(ind) - myc(ind))
            somma = somma + 1
        endif
3760 continue

        mediay = media/ somma
        shifty = NINT(mediay)

C ----- scrittura degli shift nel file d'appoggio

        write (40,5000) shiftx, shifty
5000 format ('',i3,'',i3)
        close (unit=40, status='KEEP')

C -----
C---                               chiusura programma                               -----
C-----
        goto 9000

C ----- messaggi di errore

8000 write(*,*) 'Errore in lettura file immagine di base '
        goto 9000

8100 write(*,*) 'Errore in lettura file immagine di confronto'
        goto 9000

8200 write(*,*) ' Errore in scrittura file '
        goto 9000
8300 write(*,*) ' Errore in scrittura file util'
        goto 9000

9000 continue
end

```

## 7.2 somma2.for

```

        program SOMMA2
C -----
C -- i pixel sono lunghi 2 byte ciascuno, quindi il pixel
C -- di lettura deve essere integer*2
C -- altrimenti tratterebbe due pixel alla volta.
C -- Se si devono fare calcoli bisogna appoggiarsi ad una
C -- variabile di comodo di integer (= integer*4).
C -- Versione 1.1    30 July, 1996
C -----

C ----- dichiarazioni variabili

```

```

integer*2  bufpix
integer*2  pxl (1:405,1:16)

integer    deltax, deltay
integer    x, y, maxx, maxy
integer    lp, valy
character*12 nmfilec

C ----- inizializzazioni
      maxx = 405
      maxy = 576

C -----  ottenere nome dei file dal file nome

      open(unit=11, file='utilc', err=8200, status='OLD')
      read(11,1000,end=1200) nmfilec

1200  continue
      close (11)

1000  format(A)

C -----  aprire i file di lettura e di scrittura

      open(unit=30, file=nmfilec//'.img', err=8000, status='OLD',
& FORM='BINARY')
      open(unit=50, file=nmfilec//'.out', err=8100, status='NEW',
& FORM='BINARY')
      open(unit=40, file=nmfilec//'.dat', err=8200, status='OLD')

C -----  ottieni i valori dei delta dal file

      read(40,*) deltax, deltay

C -----
C -- fa un loop su Y 36 volte perche' l'intera matrice non ci sta
C -- legge il valore del pixel e fa lo shift in base ai valori letti
C -- controlla che non fuoriesce dai limiti

      do 2015, lp = 1, 36
      do 2005, y = 1, 16
      do 2000, x = 1, 405

          read (30) pxl(x,y)

C -----
C   calculation routines
C -----

```

```

        valy = (lp-1)*16 + y

        IF ( (x + deltax .LE. maxx) .AND.
-          (x + deltax .GE. 1) .AND.
-          (valy + deltay .LE. maxy) .AND.
-          (valy + deltay .GE. 1)) THEN

        bufpix = pxl(x,y)

        write (50) bufpix

        ENDIF
C -----

2000  continue
2005  continue
2015  continue

C ----- chiude i file di lettura e scrittura

        close (unit=30, status='KEEP')
        close (unit=40, status='KEEP')
        close (unit=50, status='KEEP')

        goto 9000

C -----
C -- messaggi di errore

8000  write(*,*) 'Errore in lettura file immagine '
        goto 9000

8100  write(*,*) 'Errore in scrittura file immagine '
        goto 9000

8200  write(*,*) 'Errore in lettura file dati '
        goto 9000

C ----- fine programma

9000  continue
      END

```

### 7.3 somma3.for

```

      program SOMMA3
C -----
C -- Questa routine prende i file allineati (shifted) e li
C -- somma.

```

```

C -- I pixel sono lunghi 2 byte ciascuno, quindi si deve
C -- usare integer*2 altrimenti tratterebbe due pixel alla volta,
C -- ma poi ci si deve appoggiare ad una variabile integer
C -- di comodo.
C -- versione 1.1    30 July, 1996
C -----

```

```

integer*2  bufpix
integer*2  pxlbas (1:405,1:16)
integer*2  pxlnew (1:405,1:16)

```

```

integer    deltax, deltay
integer    ny, x, y, lp
integer    maxy, maxx

```

```

character*12 nmfileb
character*12 nmfilec
character*12 nmfilef

```

```

maxx = 405
maxy = 576

```

```

C -----
      open(unit=10, file='utilb', err=8000, status='old')

      read(10,1400,end=1000) nmfileb

1000 continue
      close (10)

      open(unit=11, file='utilc', err=8000, status='old')

      read(11,1400,end=1100) nmfilec

1100 continue
      close (11)

      open(unit=12, file='utilf', err=8000, status='old')

      read(12,1400,end=1200) nmfilef
1200 continue
1400 format(A)
      close (12)

```

```

c-----apertura files

```

```

      open(unit=20, file=nmfileb//'.img', err=8100, status='OLD',
& form='BINARY')
      open(unit=50, file=nmfilec//'.out', err=8100, status='OLD',
& form='BINARY')

```

```

        open(unit=60, file=nmfilef//' .img', err=8000, status='unknown',
& form='BINARY')
        open(unit=40, file=nmfilec//' .dat', err=8000,status='old')

        read(40,*) deltax, deltax

        do 2015 lp = 1, 36
        do 2005 y = 1, 16
            do 2000 x = 1, 405

                read (20) pxlbas(x,y)
                bufpix = pxlbas(x,y)

C -----
C   routine di calcolo
C -----

                ny = (lp-1)*16 + y
                if ((ny - deltax .LE. maxy) .AND.
-                 (ny - deltax .GE. 1) .AND.
-                 (x - deltax .LE. maxx) .AND.
-                 (x - deltax .GE. 1)) THEN

                    read (50) pxlnew(x,y)
                    bufpix = pxlbas(x,y) + pxlnew(x,y)
                endif

                write (60) bufpix

C -----

2000    continue
2005    continue
2015    continue

                close (unit=50, status='KEEP')
                close (unit=20, status='KEEP')
                close (unit=60, status='KEEP')
                close (unit=40, status='KEEP')

                write(*,*) ' '
                write(*,*) 'Immagine trattata registrata nel file ',nmfilef
                write(*,*) ' '

                goto 9000

8000    write(*,*) 'Errore in lettura '
                goto 9000

8100    write(*,*) 'Errore in scrittura file immagine '

```

```
        goto 9000

8200  write(*,*) 'Errore in lettura file dati '
      goto 9000

9000  continue
      END
```

## References

- [1] Buil, Christian *CCD Astronomy: construction and use of an astronomical CCD camera* Willmann-Bell, Richmond VA, 1991
- [2] Lindley, Craig *Practical Image Processing in C* John Wiley & Sons, 1991
- [3] McLean *Electronic Imaging in Astronomy* John Wiley & Sons, 1997