Un problema nella digitalizzazione di libri e manoscritti antichi: la correzione della distorsione indotta dalla curvatura della pagina

Maurizio Antonelli, Maresa Bertolo, Fabrizio Celentano, Martin Zürn

Marzo 2000 / Marzo 2001

Sommario

Abbiamo effettuato uno studio su alcuni algoritmi per correggere la distorsione indotta dalla curvatura delle pagine di antichi scritti. Questa curvatura, si presenta in quasi tutti i libri e manoscritti antichi che di solito sono molto delicati. È necessario quindi intervenire non fisicamente per non danneggiare l'oggetto. La digitalizzazione delle immagini ci ha permesso quindi un intervento completamente non distruttivo. Il reperto viene maneggiato solo per l'acquisizione, senza il minimo contatto fisico. L'algoritmo viene poi applicato da un calcolatore sull'immagine digitale composta solo da valori numerici.

Indice

1 Introduzione

La carta delle pagine di uno scritto antico, nella migliore condizione di conservazione, non è mai perfettamente liscia come la carta che si vede solitamente nei libri moderni, ma a causa del tempo e dell'umidità ha perso la sua originale conformazione liscia e presenta quindi delle zone con convessitá o concavitá più o meno accentuate.
Questo fatto pone un problema abbastanza grande nel momento in cui si decide di fotografare questi testi antichi per scopi di catalogazione, pubblicazione o per una semplice immagine da poter inserire in un qualunque articolo.
La deformazione della superficie delle pagine causa in primo luogo una notevole distorsione dei caratteri del testo; in secondo luogo presenta anche una disuniformità dell'illuminazione, creando zone di luce illuminate con angolature diverse e quindi effetti poco belli esteticamente ed abbastanza fastidiosi per gli osservatori.
Si è presentata quindi l'esigenza di studiare sistemi per la correzione di queste distorsioni.
Per cercare sempre di non sollecitare mai più del dovuto l'oggetto in esame, che potrebbe essere benissimo un antico manoscritto di estremo valore, si deve sempre cercare un sistema non distruttivo, in cui l'analisi e la successiva elaborazione vengano fatte a debita distanza, andando il meno possibile a contatto fisico con l'oggetto.
Da qui l'esigenza di digitalizzare l'oggetto. Questa azione è molto simile al fotografare, ma, invece di un supporto cartaceo, si utilizza un supporto informatico: l'immagine è prelevata come una serie di informazioni numeriche che un computer in qualsiasi momento interpreta e ricostruisce visivamente attraverso un monitor.
Successivamente c'è l'operazione di stiramento, non più delicatissima come nel caso in cui fosse stata svolta direttamente sul reperto, ma eseguita a livello computazionale come elaborazione aritmetica delle informazioni numeriche costituenti le immagini digitali.
Quindi nessun contatto con le pagine degli scritti; risultato: l'uso di un sistema totalmente non distruttivo.

Figura 1: Testa di scansione 3030

2 Strumentazione utilizzata

Rivelatore per acquisire e digitalizzare immagini in 3 dimensioni. Questo strumento oltre che prelevare dati per ottenere un'immagine piana a colori, deve fornirci anche l'informazione relativa alla distanza a cui si trova ogni punto dell'immagine, così da avere tutte le informazioni geometriche necessarie relative alla conformazione della pagina.
Un computer con software adatto per eseguire le giuste operazioni di stiramento.

2.1 Cyberware "Rapid 3D Color Digitizer Model 3030 RGB-Hirez [1]"

Come rivelatore è stato utilizzato questo *white scanner*. All'interno della testa dello scanner, la scatola nera che si vede nella figura 1, c'è un sistema di specchi che serve a dirigere i raggi laser del rivelatore, disposti in linea verticale, verso l'oggetto in esame e poi a rispedire la luce riflessa alle due telecamere con rivelatore CCD che si trovano all'interno. Una telecamera serve a rivelare i colori dell'oggetto in scansione; l'altra, invece, vede, da una certa angolatura, la linea rossa tracciata dai laser sull'oggetto e dal profilo riesce a ricostruire la forma del bersaglio.
La linea di raggi laser che viene proiettata è di 15 cm e di conseguenza per oggetti più alti sono necessarie diverse scansioni e la successiva ricostruzione dell'immagine totale.
Il laser utilizzato dal sistema è un laser He-Ne di lunghezza d'onda 633nm. L'acquisizione dei dati è gestita da un pc con processore 286. Questo pc memorizza i dati esclusivamente per poi trasferirli via rete ad una *WorkStation* remota.

Figura 2: Piattaforma LN

2.2 Cyberware "Desktop motion platform LN"

Lo strumento di acquisizione è montato su questa piattaforma (Figura 2) che permette un movimento esclusivamente orizzontale, con un'ampiezza massima di 500mm.

2.3 Cyberware "MS motion platform"

A disposizione avevamo anche quest'altro tipo di piattaforma, studiato più per acquisizioni cilindriche (Figura 3). Infatti, in questo caso, il sistema di acquisizione rimaneva fermo, mentre la pedana circolare su cui è l'oggetto ruotava di 360 gradi permettendo l'acquisizione su tutta la superficie. La piattaforma MS permette anche una scansione piana poiché la pedana circolare può scorrere su un binario predisposto di circa 1 metro di lunghezza.
Nel caso del lavoro sui testi antichi, comunque, quest'ultimo sistema è abbastanza inutile, quindi il lavoro è stato svolto esclusivamente sulla piattaforma LN.

2.4 Silicon Graphics "WorkStation Iris Indigo XS4000"

I dati dell'immagine vengono trasferiti dal PC 286 a questa potente *WorkStation* progettata per grafica informatica.
La SG utilizza come sistema operativo l'IRIX 4.0.5, un sistema operativo UNIX tipico delle *Silicon Graphics*.

2.5 Il software "Echo 2.0"

Questo programma permette alla Silicon Graphics di prelevare, visualizzare e fare semplici studi e semplici modifiche su un'immagine. Si utilizza *echo* caricandolo dalla shell dell'utente *echo* col comando `ee`. Prima però bisogna comunicare alla SG se si vuole utilizzare il banco LN o il banco MS, tramite una piccola manipolazione del file di configurazione *.echo*.

Figura 3: Piattaforma MS

Attivato il programma *echo*, la prima cosa da fare è regolare la sensibilità tramite il comando `view` o `vi`. Sul monitor compare una schermata con una linea rossa che dovrebbe rappresentare visivamente il profilo dell'oggetto. Tramite i tasti *freccia su* e *freccia giù* si ottimizza la sensibilità facendo in modo di vedere sullo schermo di `view` un profilo nitido, continuo e senza disturbi. Questa operazione è tanto importante quanto facile, grazie alla chiarezza con cui il comando `view` visualizza il profilo che lo scanner vede.
Prima di prelevare un'immagine, bisogna comunicare al software la larghezza di acquisizione e la risoluzione. Per far questo ci sono a disposizione questi 8 comandi:

```
LIN150  : scansione lineare,  150mm di movimento
LIN256  : scansione lineare,  256mm di movimento
LIN384  : scansione lineare,  384mm di movimento
LIN512  : scansione lineare,  512mm di movimento
LIN1024 : scansione lineare, 1024mm di movimento
CYL180  : scansione cilindrica, 180 gradi
CYL360  : scansione cilindrica, 360 gradi
CYL1024 : scansione cilindrica, 360 gradi,
          risoluzione doppia rispetto alla precedente
```

Le ultime tre sono naturalmente per l'uso della piattaforma MS.
In tutti i casi, lo scanner dá in uscita un'immagine di 512 x 512 pixel; quindi, maggiore sará la dimensione dell'acquisizione, minore sará la risoluzione.
A questo punto si procede facilmente al prelevamento dell'immagine tramite il comando `image` o semplicemente `im`.
Successivamente l'immagine può essere visualizzata con indifferentemente i comandi `display, dis, di, ed, edit`.
Da qui col semplice uso del mouse è possibile, tramite funzioni in appositi menù a tendina, ruotare, ingrandire, visualizzare dei falsi colori indicanti la profonditá e molte altre operazioni.
La strumentazione precedentemente illustrata è stata messa a disposizione dal Laborato-

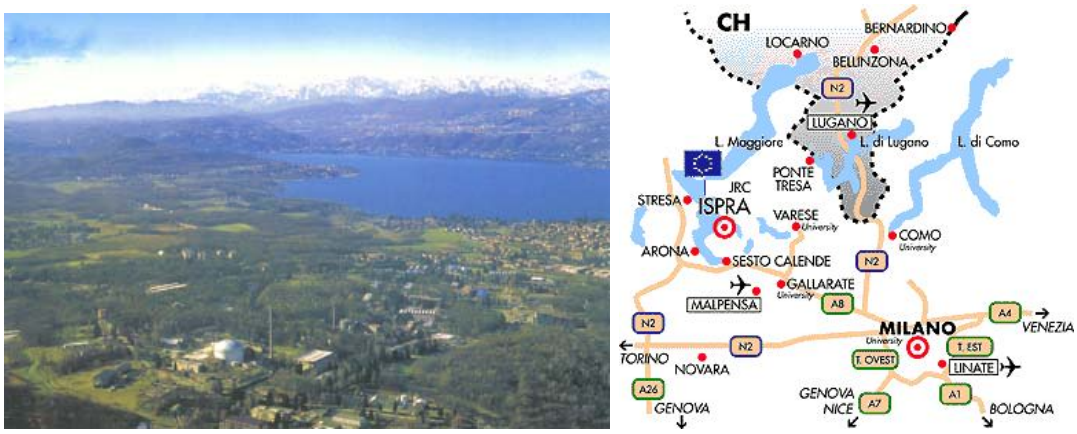

Figura 4: Centro Comune di Ricerca di Ispra [4]

rio di Profilometria (Unitá RMDS, Istituto ISIS) del Centro Comune di Ricerca sito di Ispra (VA).

3 Procedura sperimentale

Il primo mese di lavoro è stato dedicato esclusivamente allo studio del funzionamento degli strumenti di acquisizione.

3.1 La struttura dei file contenenti i dati di profonditá

Per continuare è stato necessario comprendere la struttura dei file delle profonditá dati in uscita da *echo*.
Per far ciò, sono stati confrontati due file di una stessa immagine, di cui uno convertito in *ASCII* con il programma *echoascii* e l'altro trattato con l'istruzione Unix "`od`", che dà in uscita i valori presi dal file originale, 16 bit alla volta.
Siamo arrivati così a sapere che nei file delle profonditá vi è un *header* composto da 252 caratteri più il nome del file in esame; ecco come si presenta un header dei file:

```
NAME=nome del file
DATE=Thu Jan  1 01:00:00 1970
SPACE=CARTESIAN
COLOR=SGI
THETA_RIGHTHAND=TRUE
NLG=512
LGINCR=293
LGMIN=0
LGMAX=511
NLT=512
LTINCR=300
LTMIN=0
LTMAX=511
RMIN=0
RMAX=262136
RSHIFT=3
LGSHIFT=0
SCALE=100.00
RPROP=100.00
DATA=
```

Poi, dopo l'uguale, vi sono in sequenza i valori di profondità che per ogni pixel sono dati da 16 bit più altri tre bit fittizi posti uguale a 0 dal programma. Ogni valore di profonditá dato da *echo* è quindi un multiplo di 8, con unità di misura il micrometro; per rappresentare i VOID, valori fuori dal range dello scanner, si ha un valore di 262144, equivalente ai 19 bit con solo il primo uguale ad 1; se il valore dato è superiore al valore indicante il VOID, allora il numero va interpretato in maniera negativa sottraendogli il valore 65536 x 8.

3.1.1 Il programma ECH, per avere file di facile elaborazione

A questo punto si è deciso di scrivere un programma in linguaggio C [2] per convertire i file Cyberware in un formato più comodo per successive elaborazioni.
Il programma *ech* chiama in input il file di *echo*, contenente i dati di profondità, ed il file delle tonalità di grigio dell'immagine in formato *sunraster* (estensione `.sun`), conversione ottenibile facilmente tramite *xv* [3]. Questo nostro programma dà in uscita un file *ascii* con lo stesso nome, ma con estensione `.ech`, in cui ogni record è formato da quattro numeri separati da una virgola. Questi numeri sono nell'ordine: la coordinata x (in micron), la coordinata y (in micron), la coordinata z (profondità, in micron) ed infine la tonalità di grigio del pixel in esame.
Il listato del programma lo si può trovare nell'appendice A.

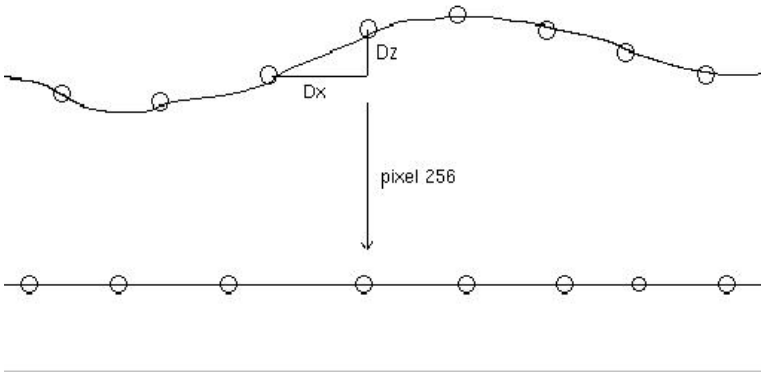

Figura 5: Disegno A: Algoritmo per lo stiramento

3.2 I programmi STIRAX e STIRAY

Questi programmi effettuano lo stiramento lungo l'asse X e lungo l'asse Y.

3.2.1 L'algoritmo

Per ogni pixel dell'immagine, si hanno il valore della tonalità di grigio e il valore della profondità (z) in micrometri. Inoltre è conosciuta la distanza tra i centri di due pixel adiacenti (Dx). Ora, tramite il teorema di Pitagora, applicato tra la differenza delle z di 2 pixel adiacenti e Dx si puo' calcolare la distanza che ci sarebbe tra i centri dei due pixel dopo lo stiramento dell'immagine.

Per effettuare i calcoli su tutta l'immagine, si inizia, per ogni riga, dal punto centrale, di coordinata 512 / 2 = 256. Da qui si applica la tecnica precedente per tutti i pixel a sinistra e poi per tutti quelli a destra. Ad ogni pixel verrà assegnata la nuova distanza dal centro, data dalla somma delle distanze di ogni pixel che lo precede con il rispettivo precedente, fino al pixel centrale. A questo punto tra due pixel allontanati (A e B), ne rimangono alcuni vuoti. A questi viene assegnata la stessa tonalità di grigio del più vicino tra A e B.

Il disegno A può aiutare a capire meglio il procedimento.

Per la scrittura del programma, sono bastate delle subroutines per il calcolo aggiunte al programma *ech*. I listati dei programmi *stirax* e *stiray* sono riportati negli Appendici B e C. Le figure 6, 7, 8, 9 rappresentano delle immagini stirate con questi due programmi.

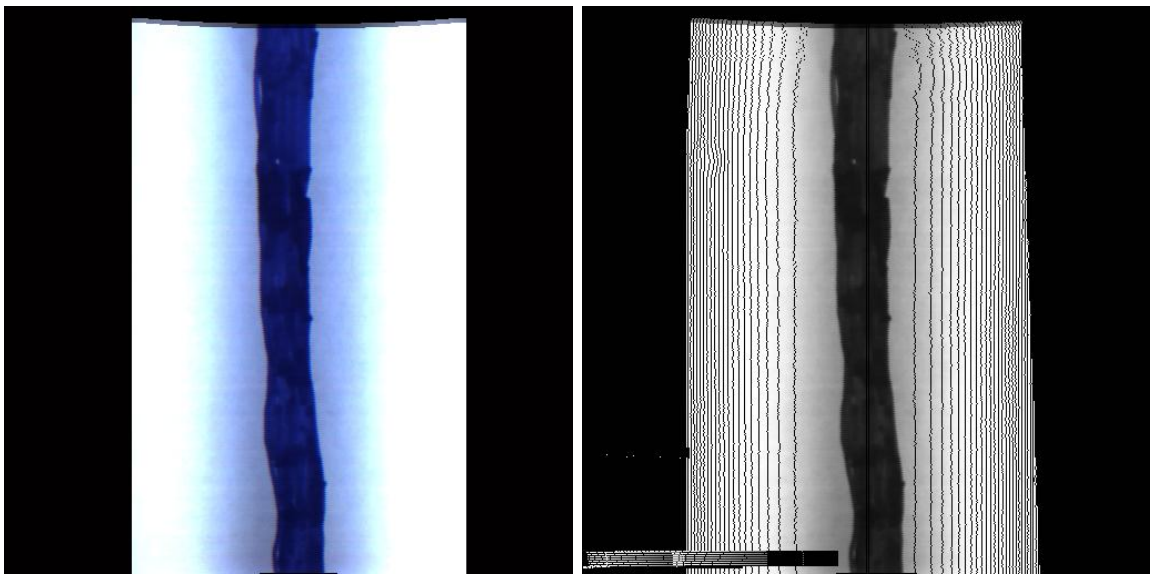

Figura 6: Cilindro con riga azzurra in mezzo, stirato lungo la X

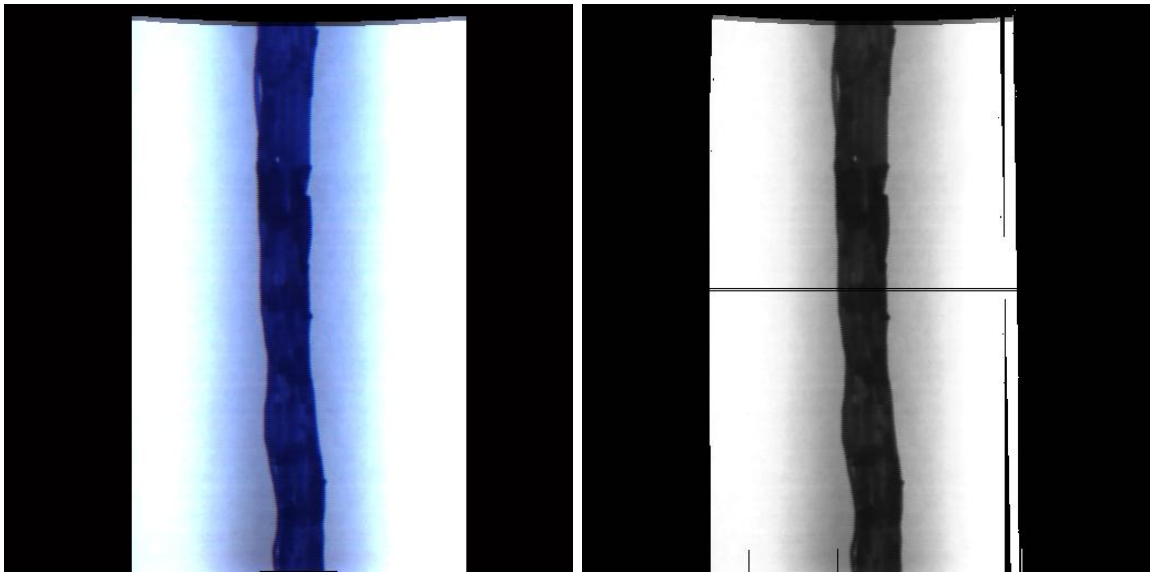

Figura 7: Cilindro con riga azzurra in mezzo, stirato lungo la Y

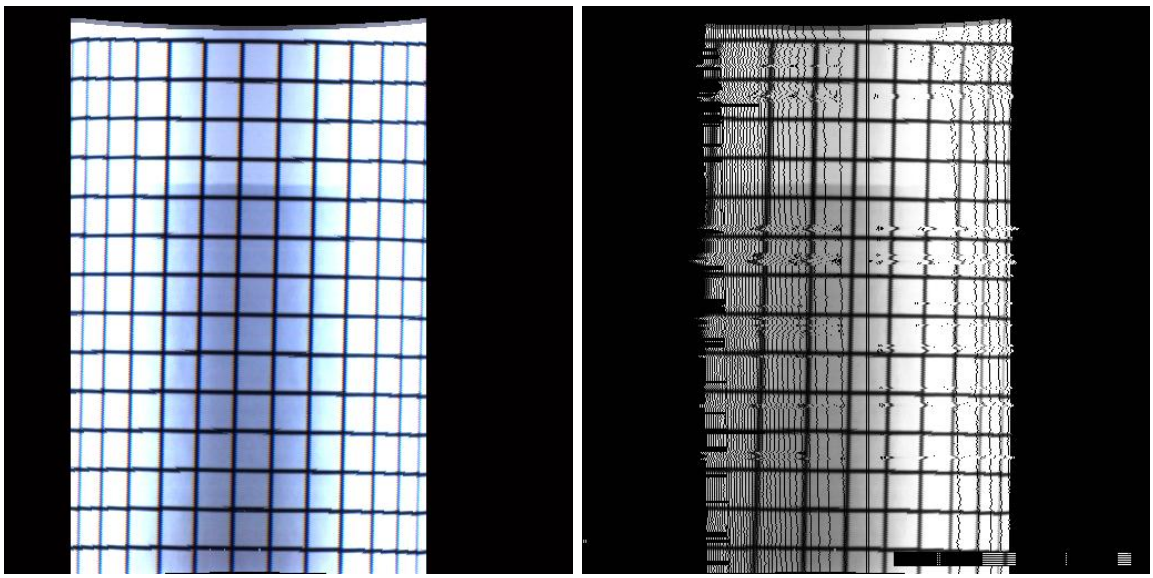

Figura 8: Cilindro con quadrettatura, stirato lungo la X

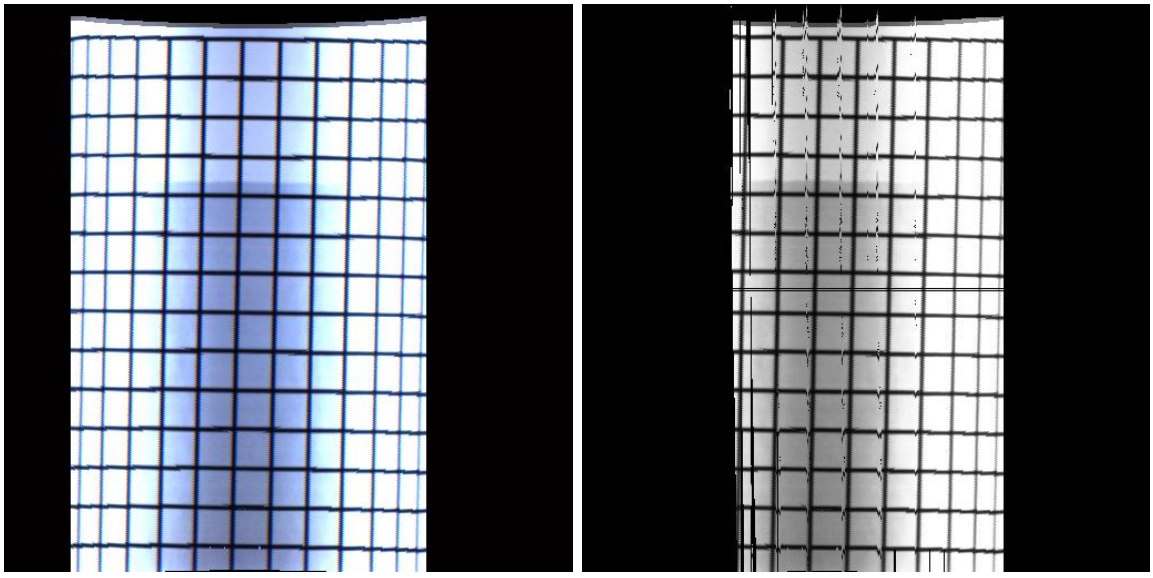

Figura 9: Cilindro con quadrettatura, stirato lungo la Y

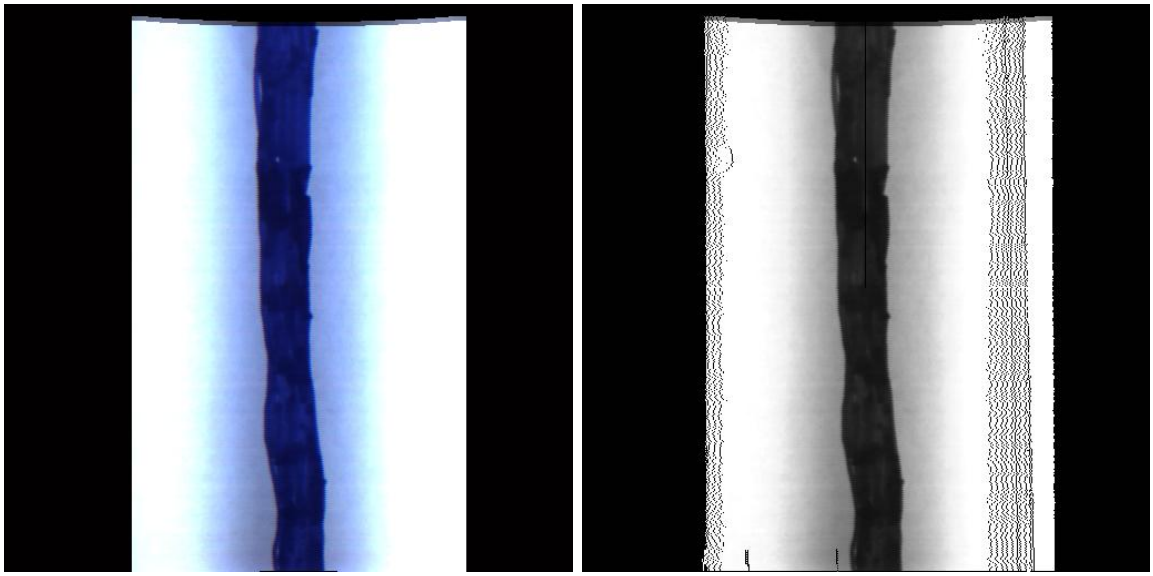

Figura 10: Cilindro con riga azzurra in mezzo

3.3 Il programma STIRA

A questo punto è stato scritto un programma che applicasse lo stiramento alle due direzioni X e Y contemporaneamente. Il listato è presente nell'appendice D.
Il programma è stato provato sulle immagini delle figure 10, 11, 12, 13, 14, 15, 16 riportate: un cilindro di ferro con incollato un foglio bianco con una riga azzurra in mezzo tracciata con un pennarello, un cilindro con un foglio con tracciata una quadrettatura, il foglio con la quadrettatura con due livelli diversi di stropicciamento e su un foglio con riportata una poesia di Giacomo Leopardi anch'esso sul cilindro e poi con due livelli diversi di stropicciamento.

Nelle figure 10, 11, 12 e 13 si vede uno stiramento solo nella parte più esterna. Nelle immagini non sono stati ricostruiti i pixel mancanti, per mettere meglio in evidenza il lavoro del programma.
Lo stiramento ottenuto va aumentando man mano che si va verso l'esterno dell'immagine. Questo è dovuto alla presenza di un errore sistematico che continua a sommarsi sempre di più man mano che ci si sposta verso i pixel esterni. L'errore è dovuto probabilmente al fatto che le distanze in micron vengono convertite in distanze in pixel, subendo così una discretizzazione eccessiva che rovina il procedimento.
Sulle ultime immagini, lo stiramento ha addirittura causato un'esplosione completa. Questo probabilmente per colpa di qualche malfunzionamento dei rivelatori dello scanner che quindi danno un file di partenza con valori errati.

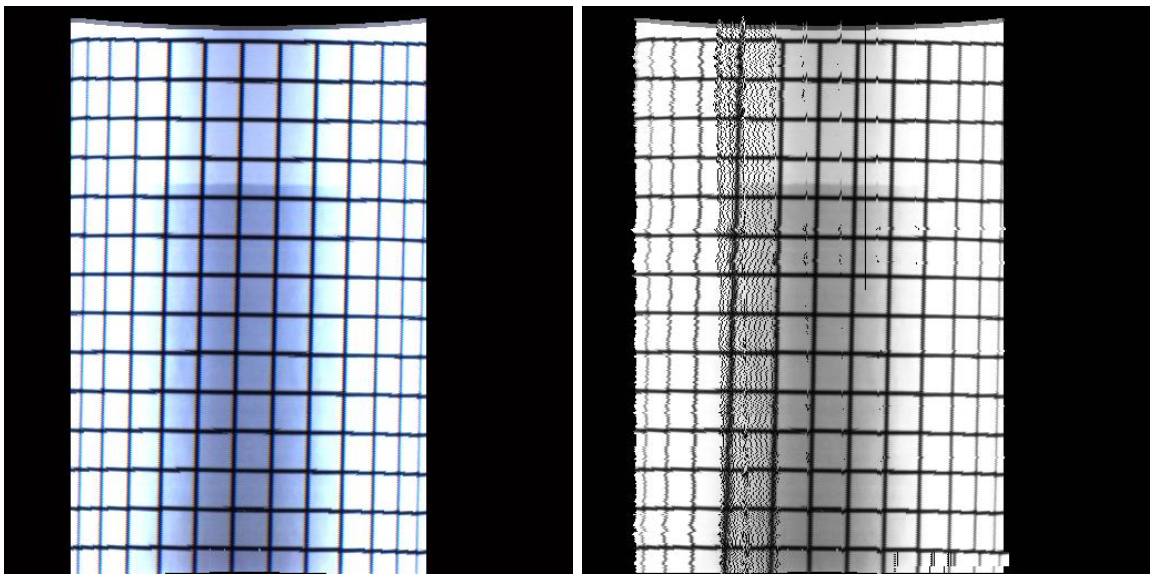

Figura 11: Cilindro con quadrettatura

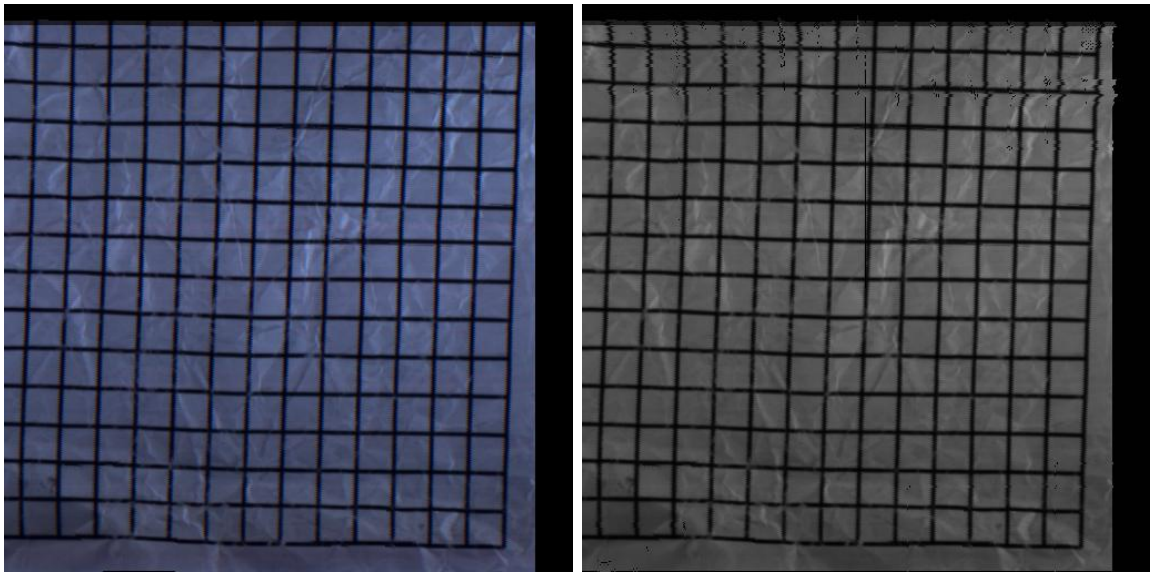

Figura 12: Foglio poco stropicciato con quadrettatura

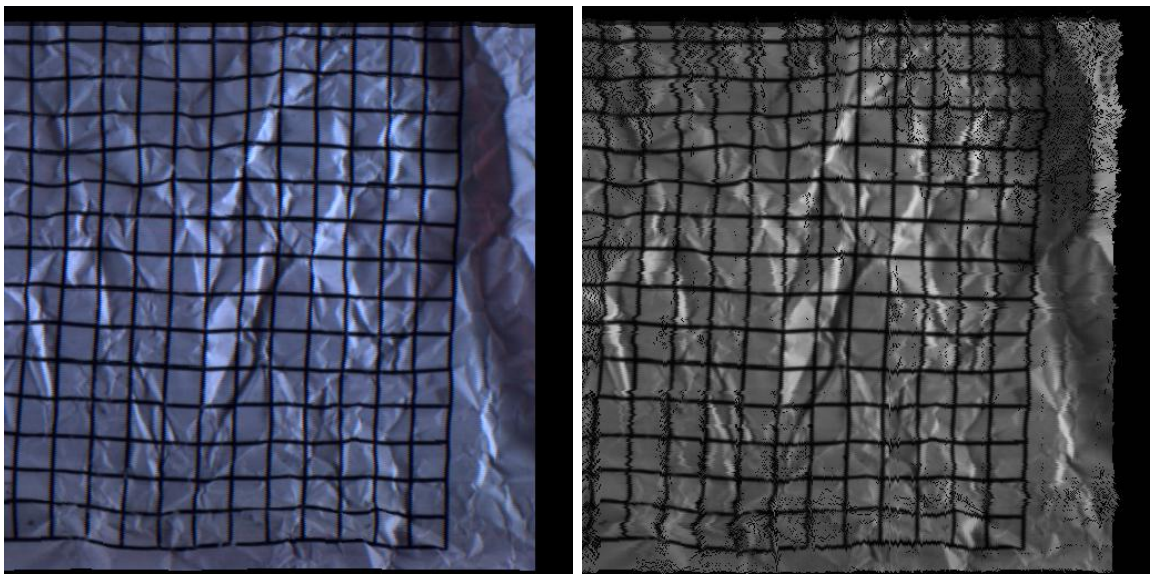

Figura 13: Foglio molto stropicciato con quadrettatura

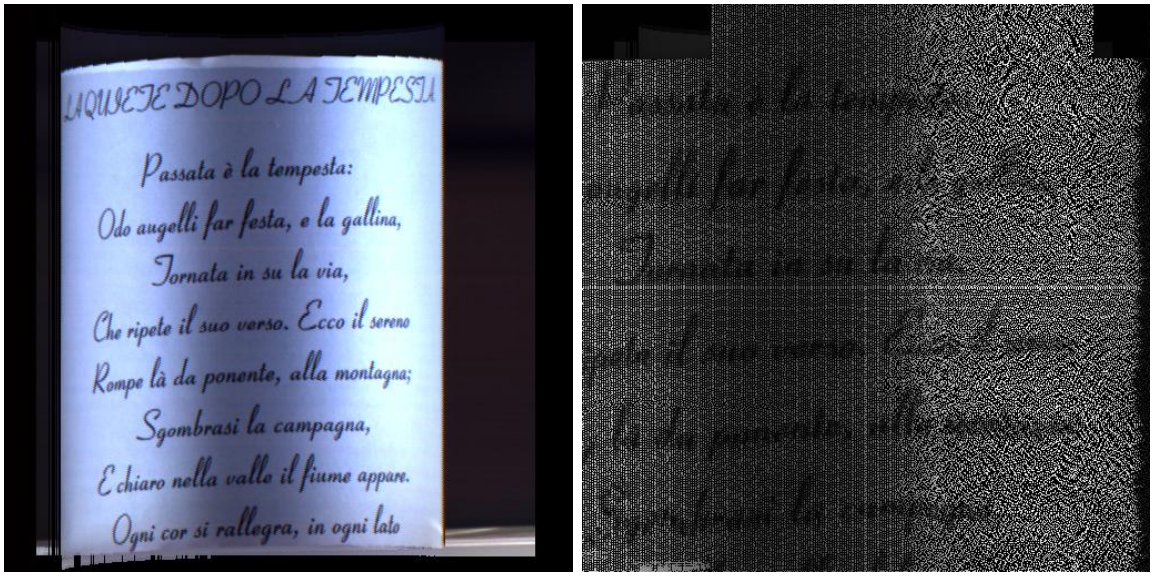

Figura 14: Cilindro con poesia di Leopardi

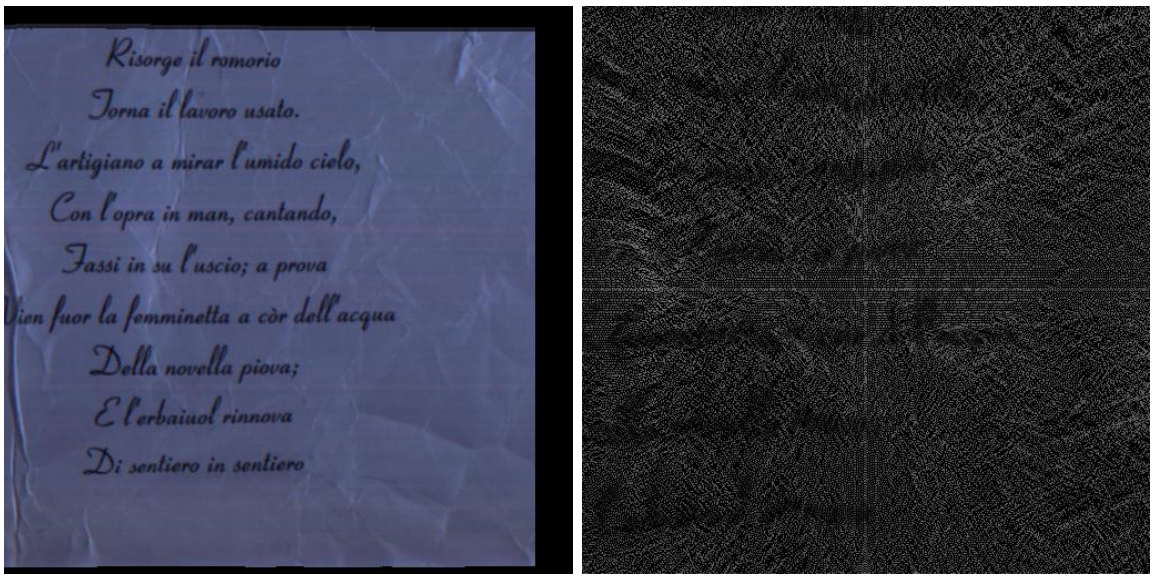

Figura 15: Foglio poco stropicciato con poesia di Leopardi

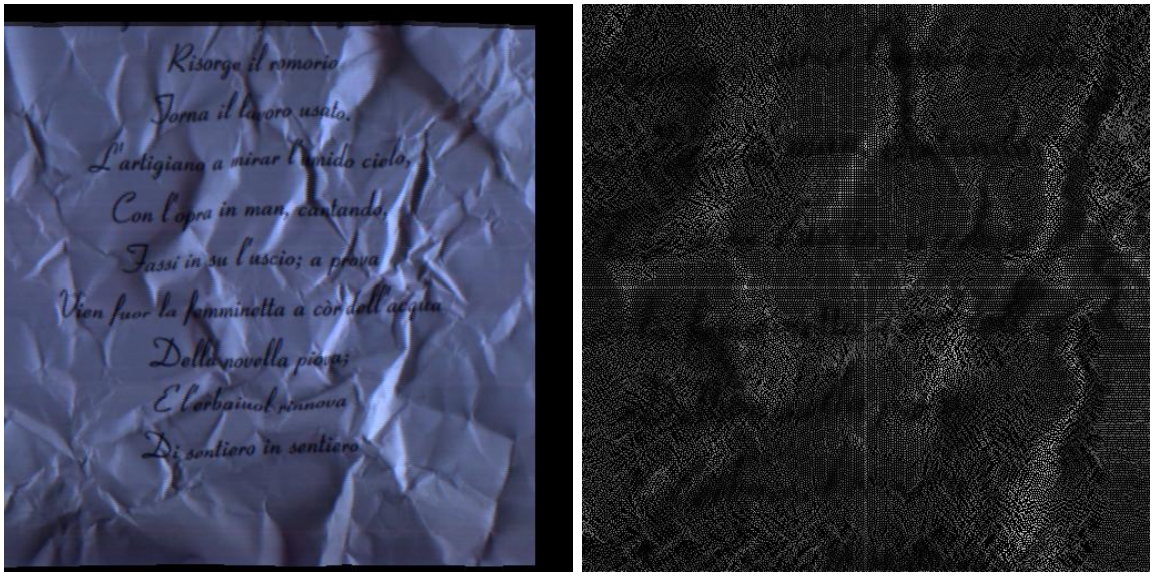

Figura 16: Foglio molto stropicciato con poesia di Leopardi

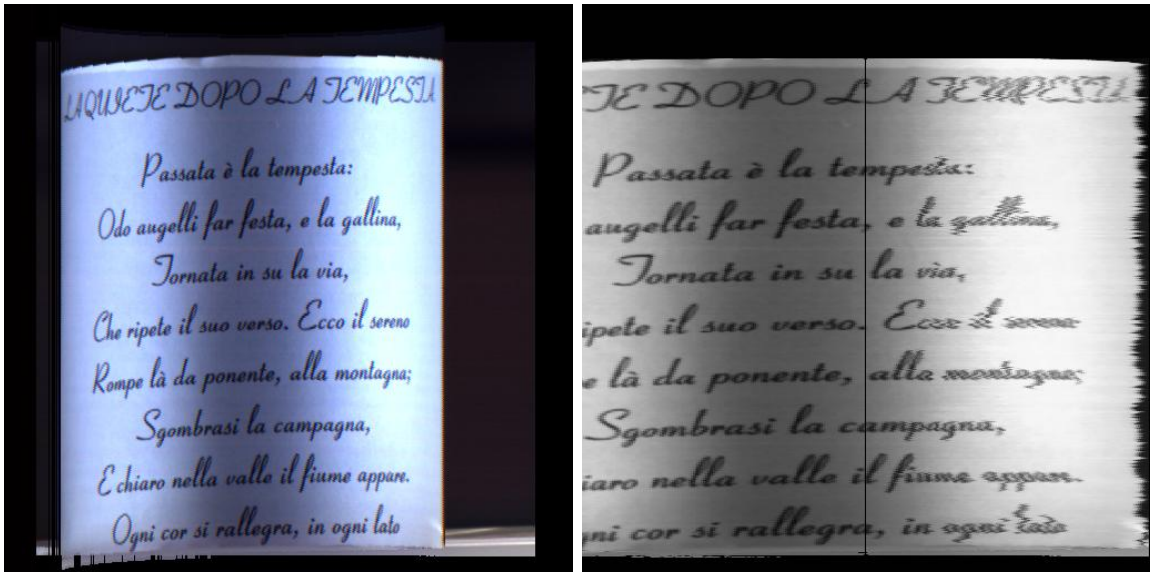

Figura 17: Cilindro con poesia di Leopardi, algoritmo STI

3.4 Un altro algoritmo, il programma STI

E' stata infine provata una modifica all'algoritmo utilizzato.
Poiché per il momento si è trattato solo di una prova, questo algoritmo è stato applicato solo lungo la direzione X.
Una volta ottenuta la matrice 512 x 512 con i valori delle distanze reali di ogni pixel dalla linea verticale di ascissa 256, si andava a prendere ogni pixel dell'immagine, si calcolava la sua distanza dalla linea di riferimento (300 micron x l'ascissa del pixel) e a questo punto si trovavano dalla matrice delle distanze reali i due pixel tra i quali questo sarebbe dovuto stare. A questo punto gli si assegnava il colore tramite la media dei colori dei due pixel circostanti pesata sulla distanza a cui essi stanno dal pixel in esame (più un pixel dista da quello in esame, meno il suo colore influisce).
Il listato del nuovo programma, chiamato *sti.c*, lo si può vedere nell'appendice E. Nelle figure 17, 18, 19 si può vedere l'applicazione di quest'altro algoritmo.

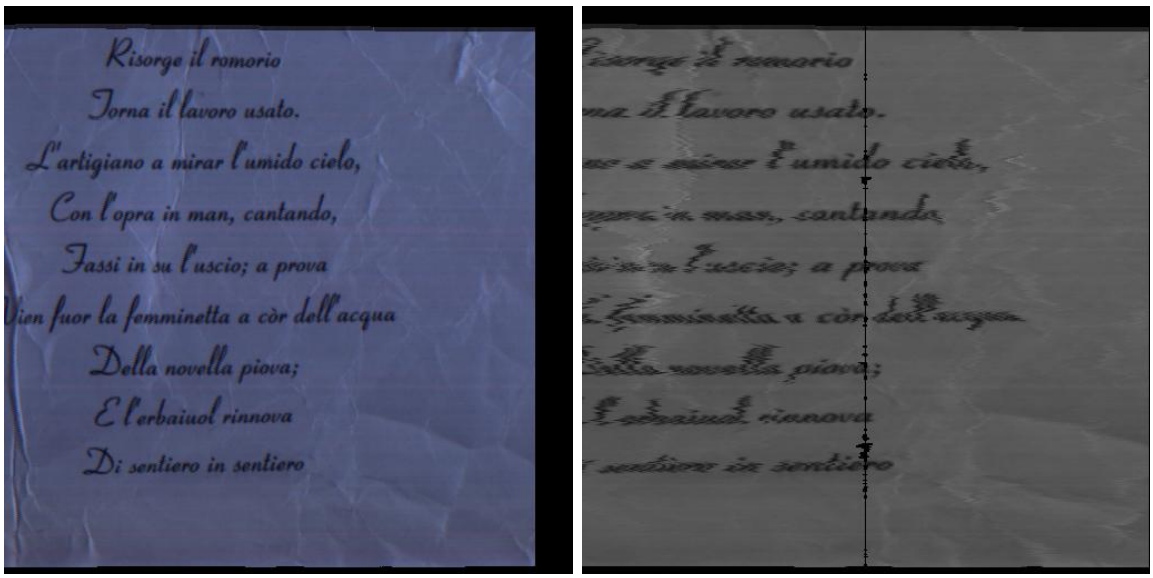

Figura 18: Foglio poco stropicciato con poesia di Leopardi, algoritmo STI

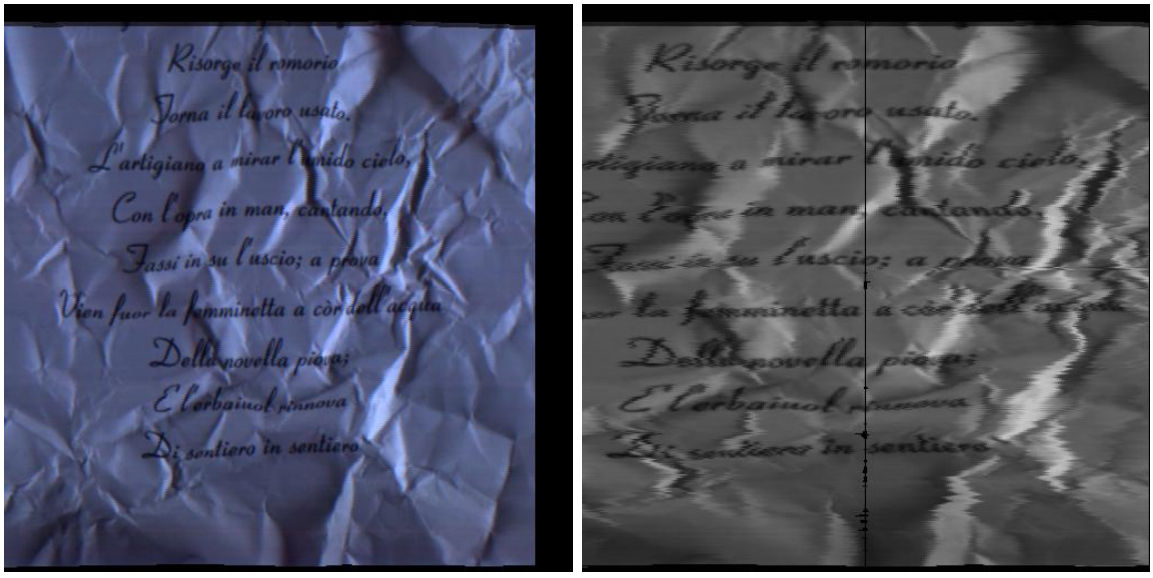

Figura 19: Foglio molto stropicciato con poesia di Leopardi, algoritmo STI

4 Conclusioni

4.1 Primo algoritmo

Dalle prime immagini, anche se la qualitá non è quella sperata, si puó concludere comunque che l'algoritmo non è completamente da scartare. La bassa qualitá delle immagini risultanti è probabilmente dovuta ad una definizione troppo bassa dello strumento di acquisizione. Infatti, probabilmente, per un lavoro di questo tipo, per ottenere un'immagine stirata accettabile per l'occhio umano, è necessario avere una distanza tra due pixel inferiore a quella che si aveva (300 micron).
L'algoritmo utilizzato dovrebbe avere come suo unico errore il fatto che, invece della lunghezza di una sezione curva, va a considerare la corda, quindi la lunghezza è inferiore a quella reale. Peró, date le distanze bassissime, questo errore dovrebbe essere talmente piccolo da non influire sul livello visivo dell'immagine risultante. Naturalmente con una strumentazione a più alta definizione, il risultato potrebbe migliorare.
Probabilmente l'errore maggiore è dovuto al rumore intrinseco dei rivelatori. Per limitare questo si sarebbe dovuto trovare il sistema di fare la media tra più scansioni della stessa immagine, oppure si potrebbe provare a regolarizzare la superficie prelevata con una tecnica di spline.

4.2 Secondo algoritmo

Con l'applicazione di questo algoritmo si ottiene un'immagine chiara, ma anche qui con uno stiramento eccessivo man mano che si va verso l'esterno. Non siamo riusciti a comprenderne il motivo.
Anche qui, inoltre, è presente un rumore intrinseco dei rivelatori. Ne è la prova la presenza di stiramenti maggiori a determinate ordinate rispetto al resto dell'immagine.

A Listato del programma ECH

```
/*
                          ech3_completo.c

Conversione di file di echo in un file leggibile

Il file di input deve essere un file di echo (ad esempio un file
creato con lo scanner della Cyberware.
Il file di output sarà chiamato input.ech.

*/

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <malloc.h>

int **imatrix(nrl,nrh,ncl,nch)
int nrl,nrh,ncl,nch;
{
int i,**m;

m=(int **)malloc((unsigned) (nrh-nrl+1)*sizeof(int*));
if (!m) printf("Allocazione 1 fallita in imatrix()");
m -= nrl;

for(i=nrl;i<=nrh;i++) {
m[i]=(int *)malloc((unsigned) (nch-ncl+1)*sizeof(int));
if (!m[i]) printf("Allocazione 2 fallita in imatrix()");
m[i] -= ncl;
}
return m;
}

main(argc,argv)
int argc;
char *argv[];
{

FILE *fp_in, *fp_out, *fp_imm;
char infile[32],
     outfile[32],
     immfile[32],
     c,
     a,
     dato_char[65536],
     lunghezza_mappa[4];

unsigned short u;

int **z,
    **imatrix(),
    int_a,
    header,  // Lunghezza header del file delle profonditá
    i,j,contatore,                  /* Indici */
    xposts,               /* Numero di punti in X */
    yposts,               /* Numero di punti in Y */
    n,                    /* Totale dei punti xposts*yposts */
    s,
    range,                /* Dimensione della scansione */
    passo_della_i,        /* Passo della lg in micron */
    dato_colore[65536],   /* Dati della mappa */
    immagine[512][512],   /* Immagine in scala di grigi */
    lunghezza_mappa_int;


/* Test degli argomenti */
if ( argc == 1 )
        { printf("Devi dare un argomento\n");
          exit(1);
};
```

```
/* File di input */
strcpy(infile,argv[1]);
if ((fp_in = fopen (infile, "r")) == NULL)
   { printf("File di input inesistente\n"); exit(); };

/* File di output */
strcpy(outfile,argv[1]);
strcat(outfile,".ech");


xposts=512; yposts=512;
n=xposts*yposts;

/* Richiede il range di scansione */
  do {
  printf("\n\nInserire il range di scansione (150, 256, 384 mm): ");
  scanf("%d", &range);
  printf("\n");
}

/* Controlla esattezza del range */
while (range!=150 && range!=256 && range!=384);


/* Definizione della matrice delle profondità */
z = imatrix ( 0, xposts-1, 0, yposts-1 );

header = strlen( argv[1] ) + 252;
for (j=0;j<header;j++){
fscanf(fp_in,"%c",&c);
}


/* Matrice delle profondità */
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
        fscanf(fp_in,"%2c",&u);
        z[i][j]=8*(int)u;

if (z[i][j] > 262144)
z[i][j] = - 65536 * 8 + z[i][j];

}};

/* Chiusura del file di input */
fclose (fp_in);


/* Preleva dati colore dal file sunraster */

/* File sun */
strcpy(immfile,argv[1]);
strcat(immfile,".sun");


/* Apre il file sunraster */
if ((fp_imm = fopen (immfile, "r")) == NULL)
   { printf("File sun di input inesistente\n"); exit(); };

/* Salta l'header che non interessa */
for (j=0;j<28;j++){
fscanf(fp_imm,"%c",&c);
}

/* Legge la lunghezza della mappa */
for (j=0;j<4;j++){
fscanf(fp_imm,"%c",&lunghezza_mappa[j]);
}
lunghezza_mappa_int=lunghezza_mappa[3]+
                    lunghezza_mappa[2]*256+
                    lunghezza_mappa[1]*65536+
```

```
                    lunghezza_mappa[0]*16777216;


/* Legge la mappa */
for (contatore=0;contatore<lunghezza_mappa_int;contatore++){
fscanf(fp_imm,"%c",&dato_char[contatore]);
        dato_colore[contatore]=dato_char[contatore];
}


/* Matrice dell'immagine */
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
fscanf(fp_imm,"%c", &a);
        int_a=(int)a;
immagine[i][j]=dato_colore[int_a];
}};


/* Chiude il file sun */
fclose (fp_imm);


/* Scrittura sul file
   x e y vengono dati in micrometri */

/* L'incremento della i deve essere di
       293  per le scansioni di 150 mm
       500  per le scansioni di 256 mm
       750  per le scansioni di 384 mm
*/

switch (range) {

  case 150: { passo_della_i=293;
              break; }

  case 256: { passo_della_i=500;
              break; }

  case 384: { passo_della_i=750;
              break; }
}


fp_out = fopen (outfile,"w");

for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
      if (z[i][j] == 262144) continue;
      fprintf(fp_out,"%d,%d,%d,%d\n", \
       (i*passo_della_i), \
    (j*300), \
                              z[i][j], \
immagine[i][j]  );
}}
fclose (fp_out);


}
```

B Listato del programma STIRAX

```
/*
                        stirax2.c

Stiramento lungo la dirzione X.

In input dovranno esserci un file di profondità di echo ed
il file coi colori in formato sunraster a scala di grigi.
L'output sarà un file chiamato input_stiratox2.sun.

*/
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <malloc.h>


int **imatrix(nrl,nrh,ncl,nch)
int nrl,nrh,ncl,nch;
{
int i,**m;

m=(int **)malloc((unsigned) (nrh-nrl+1)*sizeof(int*));
if (!m) printf("Allocazione 1 fallita in imatrix()");
m -= nrl;

for(i=nrl;i<=nrh;i++) {
m[i]=(int *)malloc((unsigned) (nch-ncl+1)*sizeof(int));
if (!m[i]) printf("Allocazione 2 fallita in imatrix()");
m[i] -= ncl;
}
return m;
}

main(argc,argv)
int argc;
char *argv[];
{

FILE *fp_in, *fp_out, *fp_imm;
char infile[32],
     outfile[32],
     immfile[32],
     c,
     a,
     dato_char[65536],
     lunghezza_mappa[4];

unsigned short u;

double pass;

double distanza[512][512]; /* Matrice distanze per lo stiramento */


int **z,
    **imatrix(),
    int_a,
    header,  // Lunghezza header del file delle profonditá
    i,j,contatore,                    /* Indici */
    arrivo,
    arrivo2,
    xposts,              /* Numero di punti lungo la X */
    yposts,              /* Numero di punti lungo la Y */
    n,                   /* Totale dei punti xposts*yposts */
    s,
    range,               /* Dimensione della scansione */
    passo_della_i,       /* Passo della lg in micron */
    dato_colore[65536],  /* Dati della mappa */
    immagine[512][512],  /* Immagine in scala di grigi */
    immagine_2[512][512], /* Immagine stirata */
```

```
    lunghezza_mappa_int,
    distanza_pix,         /* Distanza in pixel tra due dati */
    distanza2pix,         /* Usata per lo riempimento */
    somma_dz[512][512];   /* Matrice distanze dal centro */



/* Test degli argomenti */
if ( argc == 1 )
        { printf("Devi dare un argomento\n");
          exit(1);
};

/* File di input */
strcpy(infile,argv[1]);
if ((fp_in = fopen (infile, "r")) == NULL)
   { printf("File inesistente\n"); exit(); };

/* File di output */
strcpy(outfile,argv[1]);
strcat(outfile,"_stiratox2.sun");



xposts=512; yposts=512;
n=xposts*yposts;

/* Richiede il range di scansione */
  do {
  printf("\nInserire il range di scansione (150, 256, 384 mm): ");
  scanf("%d", &range);
  printf("\n");
}

/* Controlla esattezza del range */
while (range!=150 && range!=256 && range!=384);


/* Definizione della matrice delle profondità */
z=imatrix(0,xposts-1,0,yposts-1);


header = strlen( argv[1] ) + 252;
for (j = 0; j < header; j++){
fscanf(fp_in,"%c",&c);
}

/* Matrice delle profondità */
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
        fscanf(fp_in,"%2c",&u);
        z[i][j]=8*(int)u;

if (z[i][j] > 262144)
z[i][j] = - 65536 * 8 + z[i][j];
}};

/* Chiusura file di input */
fclose (fp_in);


/* Preleva dati colore dal file sunraster */

/* File sun */
strcpy(immfile,argv[1]);
strcat(immfile,".sun");


/* Apre il file sunraster */
if ((fp_imm = fopen (immfile, "r")) == NULL)
   { printf("File sun inesistente\n"); exit(); };
```

```
/* Salta l'header che non interessa */
for (j=0;j<28;j++){
fscanf(fp_imm,"%c",&c);
}

/* Legge la lunghezza della mappa */
for (j=0;j<4;j++){
fscanf(fp_imm,"%c",&lunghezza_mappa[j]);
}
lunghezza_mappa_int=lunghezza_mappa[3]+
                    lunghezza_mappa[2]*256+
                    lunghezza_mappa[1]*65536+
                    lunghezza_mappa[0]*16777216;


/* Legge la mappa */
for (contatore=0;contatore<lunghezza_mappa_int;contatore++){
fscanf(fp_imm,"%c",&dato_char[contatore]);
        dato_colore[contatore]=dato_char[contatore];
}


/* Matrice dell'immagine */
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
fscanf(fp_imm,"%c", &a);
        int_a=(int)a;
immagine[i][j]=dato_colore[int_a];
}};


/* Chiude il file sun */
fclose (fp_imm);

/* L'incremento della x deve essere di
       293  per le scansioni di 150 mm
       500  per le scansioni di 256 mm
       750  per le scansioni di 384 mm
*/

switch (range) {

  case 150: { passo_della_i=293;
              break; }

  case 256: { passo_della_i=500;
              break; }

  case 384: { passo_della_i=750;
              break; }
}

/* STIRAMENTO */

/* Crea la matrice delle distanze */

for (j=0; j<yposts; j++) {

/* Sinistra */
  for (i = xposts / 2; i > 0; i--) {
    distanza[i][j] = passo_della_i;
    if (z[i][j]==262144 || z[i][j]==1024) continue;

      distanza[i][j] = sqrt (passo_della_i*passo_della_i \
            +(z[i-1][j]-z[i][j])*(z[i-1][j]-z[i][j]));

    somma_dz[i][j]=0;
    for (contatore=xposts/2; contatore>=i; contatore--)
      somma_dz[i][j]=somma_dz[i][j]+distanza[contatore][j];

  }
```

```
/* Destra */
  for (i = xposts / 2; i < xposts-1; i++) {
    distanza[i][j] = passo_della_i;
    if (z[i][j]==262144 || z[i][j]==1024) continue;

      distanza[i][j] = sqrt (passo_della_i*passo_della_i \
            +(z[i+1][j]-z[i][j])*(z[i+1][j]-z[i][j])) ;

    somma_dz[i][j]=0;
    for (contatore=xposts/2; contatore<=i; contatore++)
      somma_dz[i][j]=somma_dz[i][j]+distanza[contatore][j];

  }

}


/* Ricostruzione dell'immagine stirata */

for (j=0; j<yposts; j++) {

  /* Stiramento a sx del centro */
  distanza_pix = 1;
  arrivo = xposts/2;
  for (i = xposts/2; i >= 0; i--) {
    arrivo = arrivo - distanza_pix;
    pass = passo_della_i;
    distanza_pix = somma_dz[i][j] / pass;
    if ((somma_dz[i][j] / pass) > 256) continue;
    if (xposts / 2 - distanza_pix > 0)
      contatore = xposts / 2 - distanza_pix;
    else
      contatore = 1;
    immagine_2[contatore][j] = immagine[i][j];



// Ricostruzione pixel mancanti
    distanza2pix = distanza[i][j] / pass;
    if ((distanza2pix > 1) && (i - distanza2pix > 0)) {
      arrivo2 = i - distanza2pix / 2;
      if (arrivo2 < 1) arrivo2 = 1;
      for (contatore = i; contatore >= arrivo2; contatore--)
        immagine_2[contatore][j] = immagine[i][j];

      arrivo2 = i + distanza_pix / 2;
      if (arrivo2 > xposts) arrivo2 = xposts;
      for (contatore = i; contatore <= arrivo2; contatore++)
        immagine_2[contatore][j] = immagine[i-1][j];
    }
  }


  /* Stiramento a dx del centro */
  distanza_pix = 1;
  arrivo = xposts / 2;
  for (i = xposts / 2; i < xposts; i++) {
    arrivo = arrivo + distanza_pix;
    pass = passo_della_i;
    distanza_pix = somma_dz[i][j] / pass;
    if ((somma_dz[i][j] / pass) > 256) continue;
    if (xposts / 2 + distanza_pix < xposts)
      contatore = xposts / 2 + distanza_pix;
    else
      contatore = xposts;

    immagine_2[contatore][j] = immagine[i][j];



// Ricostruzione pixel mancanti
    distanza2pix = distanza[i][j] / pass;
```

```
    if ((distanza2pix > 1) && (i + distanza2pix < xposts)) {
      arrivo2 = i + distanza2pix / 2;
      if (arrivo2 > xposts) arrivo2 = xposts;
      for (contatore = i; contatore <= arrivo2; contatore++)
        immagine_2[contatore][j] = immagine[i][j];

      arrivo2 = i - distanza_pix / 2;
      if (arrivo2 < 1) arrivo2 = 1;
      for (contatore = i; contatore >= arrivo2; contatore--)
        immagine_2[contatore][j] = immagine[i+1][j];
    }
  }


}

/* SCRITTURA DEL FILE SUNRASTER */

/* Apre il file */
fp_out = fopen (outfile,"w");

/* Numero magico */
fprintf(fp_out,"%c%c%c%c",89,166,106,149);

/* Numero di colonne */
fprintf(fp_out,"%c%c%c%c",0,0,2,0);

/* Numero di linee */
fprintf(fp_out,"%c%c%c%c",0,0,2,0);

/* Numero di bit per pixel */
fprintf(fp_out,"%c%c%c%c",0,0,0,8);

/* Lunghezza dell'immagine in byte */
fprintf(fp_out,"%c%c%c%c",0,4,0,0);

/* Tipo */
fprintf(fp_out,"%c%c%c%c",0,0,0,1);

/* Tipo mappa */
fprintf(fp_out,"%c%c%c%c",0,0,0,1);

/* Lunghezza mappa */
fprintf(fp_out,"%c%c%c%c",0,0,3,0);

/* Scrittura mappa */
for (j=0; j<3; j++) {
  for (i=0; i<256; i++)
    fprintf(fp_out,"%c",i);
}

/* Scrittura matrice dell'immagine */
for (i=0; i<xposts; i++){
for (j=0; j<yposts; j++){
  fprintf(fp_out,"%c",immagine_2[i][j]);
}}

/* Chiusura del file */
fclose (fp_out);

}
```

C Listato del programma STIRAY

```
/*
                        stiray2.c

Stiramento lungo la direzione Y.

Il file delle profondità deve essere un file di echo.
Il file coi colori deve essere in formato sunraster a scala di grigi.
Il file di output sarà chiamato input_stiratoy2.sun.

*/
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <malloc.h>



int **imatrix(nrl,nrh,ncl,nch)
int nrl,nrh,ncl,nch;
{
int i,**m;

m=(int **)malloc((unsigned) (nrh-nrl+1)*sizeof(int*));
if (!m) printf("Allocazione 1 fallita in imatrix()");
m -= nrl;

for(i=nrl;i<=nrh;i++) {
m[i]=(int *)malloc((unsigned) (nch-ncl+1)*sizeof(int));
if (!m[i]) printf("Allocazione 2 fallita in imatrix()");
m[i] -= ncl;
}
return m;
}

main(argc,argv)
int argc;
char *argv[];
{
/***************** Dichiarazioni ***********************/
FILE *fp_in, *fp_out, *fp_imm;
char infile[32],
     outfile[32],
     immfile[32],
     c,
     a,
     dato_char[65536],
     lunghezza_mappa[4];

unsigned short u;

double pass;

double distanza[512][512]; /* Matrice distanze per lo stiramento */


int **z,
    **imatrix(),
    int_a,
    header,  // Lunghezza header del file delle profonditá
    i,j,contatore,                    /* Indici */
    arrivo,
    arrivo2,
    xposts,              /* Numero dei punti lungo la direzione X */
    yposts,              /* Numero dei punti lungo la direzione Y */
    n,                   /* Totale dei punti xposts*yposts */
    s,
    range,               /* Dimensione della scansione */
    passo_della_i,       /* Passo della lg in micron */
    dato_colore[65536],  /* Dati della mappa */
    immagine[512][512],  /* Immagine in scala di grigi */
```

```
    immagine_2[512][512], /* Immagine stirata */
    lunghezza_mappa_int,
    distanza_pix,         /* Distanza in pixel tra due dati */
    distanza2pix,         /* Usata per lo riempimento */
    somma_dz[512][512];   /* Matrice distanze dal centro */
/****************** Fine dichiarazioni ******************/


/* Test degli argomenti */
if ( argc == 1 )
        { printf("Devi dare un argomento\n");
          exit(1);
};

/* File di input */
strcpy(infile,argv[1]);
if ((fp_in = fopen (infile, "r")) == NULL)
   { printf("File inesistente\n"); exit(); };

/* File di output */
strcpy(outfile,argv[1]);
strcat(outfile,"_stiratoy2.sun");


xposts=512; yposts=512;
n=xposts*yposts;

/* Richiede il range di scansione */
  do {
  printf("\nInserire il range di scansione (150, 256, 384 mm): ");
  scanf("%d", &range);
  printf("\n");
}

/* Controlla esattezza del range */
while (range!=150 && range!=256 && range!=384);


/* Definizione della matrice delle profondità */
z=imatrix(0,xposts-1,0,yposts-1);


header = strlen( argv[1] ) + 252;
for (j = 0; j < header; j++){
fscanf(fp_in,"%c",&c);
}

/* Matrice delle profondità */
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
        fscanf(fp_in,"%2c",&u);
        z[i][j]=8*(int)u;

if (z[i][j] > 262144)
z[i][j] = - 65536 * 8 + z[i][j];
}};

/* Chiusura file di input */
fclose (fp_in);



/* Preleva dati colore dal file sunraster */

/* File sun */
strcpy(immfile,argv[1]);
strcat(immfile,".sun");


/* Apre il file sunraster */
if ((fp_imm = fopen (immfile, "r")) == NULL)
   { printf("File sun inesistente\n"); exit(); };
```

```
/* Salta l'header che non interessa */
for (j=0;j<28;j++){
fscanf(fp_imm,"%c",&c);
}

/* Legge la lunghezza della mappa */
for (j=0;j<4;j++){
fscanf(fp_imm,"%c",&lunghezza_mappa[j]);
}
lunghezza_mappa_int=lunghezza_mappa[3]+
                    lunghezza_mappa[2]*256+
                    lunghezza_mappa[1]*65536+
                    lunghezza_mappa[0]*16777216;


/* Legge la mappa */
for (contatore=0;contatore<lunghezza_mappa_int;contatore++){
fscanf(fp_imm,"%c",&dato_char[contatore]);
        dato_colore[contatore]=dato_char[contatore];
}


/* Matrice dell'immagine */
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
fscanf(fp_imm,"%c", &a);
        int_a=(int)a;
immagine[i][j]=dato_colore[int_a];
}};


/* Chiude il file sun */
fclose (fp_imm);

/* L'incremento della x deve essere di
       293  per le scansioni di 150 mm
       500  per le scansioni di 256 mm
       750  per le scansioni di 384 mm
*/

switch (range) {

  case 150: { passo_della_i=293;
              break; }

  case 256: { passo_della_i=500;
              break; }

  case 384: { passo_della_i=750;
              break; }
}

/* STIRAMENTO */

/* Crea la matrice delle distanze */

for (i=0; i<xposts; i++) {

/* Sinistra */
  for (j = yposts / 2; j > 0; j--) {
    distanza[i][j] = 300;
    if (z[i][j]==262144 || z[i][j]==1024) continue;

      distanza[i][j] = sqrt (300*300 \
            +(z[i][j-1]-z[i][j])*(z[i][j-1]-z[i][j]));

    somma_dz[i][j]=0;
    for (contatore=yposts/2; contatore>=j; contatore--)
      somma_dz[i][j]=somma_dz[i][j]+distanza[i][contatore];

  }
```

```
/* Destra */
  for (j = yposts / 2; j < yposts-1; j++) {
    distanza[i][j] = 300;
    if (z[i][j]==262144 || z[i][j]==1024) continue;

      distanza[i][j] = sqrt (300*300 \
            +(z[i][j+1]-z[i][j])*(z[i][j+1]-z[i][j])) ;

    somma_dz[i][j]=0;
    for (contatore=yposts/2; contatore<=j; contatore++)
      somma_dz[i][j]=somma_dz[i][j]+distanza[i][contatore];

  }

}


/* Ricostruzione dell'immagine stirata */

for (i=0; i<xposts; i++) {

  /* Stiramento a sx del centro */
  distanza_pix = 1;
  arrivo = yposts/2;
  for (j = yposts/2; j > 0; j--) {
    arrivo = arrivo - distanza_pix;
    pass = 300;
    distanza_pix = somma_dz[i][j] / pass;
    if ((somma_dz[i][j] / pass) > 256) continue;
    if (yposts / 2 - distanza_pix > 0)
      contatore = yposts / 2 - distanza_pix;
    else
      contatore = 1;
    immagine_2[i][contatore] = immagine[i][j];



// Ricostruzione pixel mancanti
    distanza2pix = distanza[i][j] / pass;
    if ((distanza2pix > 1) && (j - distanza2pix > 0)) {
      arrivo2 = j - distanza2pix / 2;
      if (arrivo2 < 1) arrivo2 = 1;
      for (contatore = j; contatore >= arrivo2; contatore--)
        immagine_2[i][contatore] = immagine[i][j];

      arrivo2 = j + distanza_pix / 2;
      if (arrivo2 > yposts) arrivo2 = yposts;
      for (contatore = j; contatore <= arrivo2; contatore++)
        immagine_2[i][contatore] = immagine[i][j-1];
    }
  }


  /* Stiramento a dx del centro */
  distanza_pix = 1;
  arrivo = yposts / 2;
  for (j = yposts / 2; j < yposts; j++) {
    arrivo = arrivo + distanza_pix;
    pass = 300;
    distanza_pix = somma_dz[i][j] / pass;
    if ((somma_dz[i][j] / pass) > 256) continue;
    if (yposts / 2 + distanza_pix < yposts)
      contatore = yposts / 2 + distanza_pix;
    else
      contatore = yposts;


    immagine_2[i][contatore] = immagine[i][j];
```

```
// Ricostruzione pixel mancanti
    distanza2pix = distanza[i][j] / pass;
    if ((distanza2pix > 1) && (j + distanza2pix < yposts)) {
      arrivo2 = j + distanza2pix / 2;
      if (arrivo2 > yposts) arrivo2 = yposts;
      for (contatore = j; contatore <= arrivo2; contatore++)
        immagine_2[i][contatore] = immagine[i][j];

      arrivo2 = j - distanza_pix / 2;
      if (arrivo2 < 1) arrivo2 = 1;
      for (contatore = j; contatore >= arrivo2; contatore--)
        immagine_2[i][contatore] = immagine[i][j+1];
    }
  }


}

/* SCRITTURA DEL FILE SUNRASTER */

/* Apre file in scrittura */
fp_out = fopen (outfile,"w");

/* Numeo magico */
fprintf(fp_out,"%c%c%c%c",89,166,106,149);

/* Numero di colonne */
fprintf(fp_out,"%c%c%c%c",0,0,2,0);

/* Numero di linee */
fprintf(fp_out,"%c%c%c%c",0,0,2,0);

/* Numero di bit per pixel */
fprintf(fp_out,"%c%c%c%c",0,0,0,8);

/* Lunghezza dell'immagine in byte */
fprintf(fp_out,"%c%c%c%c",0,4,0,0);

/* Tipo */
fprintf(fp_out,"%c%c%c%c",0,0,0,1);

/* Tipo mappa */
fprintf(fp_out,"%c%c%c%c",0,0,0,1);

/* Lunghezza mappa */
fprintf(fp_out,"%c%c%c%c",0,0,3,0);

/* Scrittura mappa */
for (j=0; j<3; j++) {
  for (i=0; i<=255; i++)
    fprintf(fp_out,"%c",i);
}

/* Scrittura matrice dell'immagine */
for (i=0; i<xposts; i++){
for (j=0; j<yposts; j++){
  fprintf(fp_out,"%c",immagine_2[i][j]);
}}

/* Chiusura del file */
fclose (fp_out);

}
```

D Listato del programma STIRA

```
/*
                                stira.c

Stira files Echo 3D dopo averli convertiti in un formato
leggibile da questo programma.

Il file di input contenente i dati di profondità deve essere nel
formato standard di Echo, ad esempio un file creato dallo
scanner Cyberware.
Il file dei colori deve essere in formato sunraster a scala di grigi.
L'output sarà un file nominato input_stirato.sun.

*/


#include <stdio.h>
#include <string.h>
#include <math.h>
#include <malloc.h>


int **imatrix(nrl,nrh,ncl,nch)
int nrl,nrh,ncl,nch;
{
int i,**m;

m=(int **)malloc((unsigned) (nrh-nrl+1)*sizeof(int*));
if (!m) printf("Allocazione fallita 1 in imatrix()");
m -= nrl;

for(i=nrl;i<=nrh;i++) {
m[i]=(int *)malloc((unsigned) (nch-ncl+1)*sizeof(int));
if (!m[i]) printf("Allocazione fallita 2 in imatrix()");
m[i] -= ncl;
}
return m;
}




main(argc,argv)
int argc;
char *argv[];
{

FILE *fp_in, *fp_out, *fp_imm;
char infile[32],
     outfile[32],
     immfile[32],
     c,
     a,
     dato_char[65536],
     lunghezza_mappa[4];

unsigned short u;

double pass;

double alfa_righe[512][512],
       alfa_colonne[512][512];

int **z,
    **imatrix(),
    int_a,
    header,  // Lunghezza header del file delle profonditá
    i,j,contatore,                    /* Indici */
    arrivo,
    arrivo2,
    xposts,              /* Dimensione x dell'immagine */
```

```
    yposts,               /* Dimensione y dell'immagine */
    n,                    /* Numero totale di punti (x*y) */
    s,
    range,                /* Dimensione della scansione */
    passo_della_i,        /* Passo della lg in micron */
    dato_colore[65536],   /* Dati della mappa */
    immagine[512][512],   /* Immagine in scala di grigi */
    immagine_2[512][512], /* Immagine stirata */
    lunghezza_mappa_int,
    distanza_pixel_righe[512][512],
    distanza_pixel_colonne[512][512],

    somma_dz[512][512],   // Matrice distanze dal centro
    distanza_righe[512][512],
    distanza_colonne[512][512],
    distanza_vera_righe[512][512],
    distanza_vera_colonne[512][512],
    distanza[512][512];


/* Test dell'argomento */
if ( argc == 1 )
        { printf("Devi dare un argomento\n");
          exit(1);
};

/* Quale file è in input? */
strcpy(infile,argv[1]);
if ((fp_in = fopen (infile, "r")) == NULL)
   { printf("Input file inesistente\n"); exit(); };

/* Usa questo se vuoi mettere l'output in un file */
strcpy(outfile,argv[1]);
strcat(outfile,"_stirato.sun");


xposts=512; yposts=512;
n=xposts*yposts;

/* Richiede il range di scansione */
  do {
  printf("\nInserire il range di scansione (150, 256, 384 mm): ");
  scanf("%d", &range);
  printf("\n");
}

/* Controlla esattezza del range */
while (range!=150 && range!=256 && range!=384);


/* Definisce la dimensione della matrice e il range degli indici */
z=imatrix(0,xposts-1,0,yposts-1);

header = strlen( argv[1] ) + 252;
for (j = 0; j < header; j++){
fscanf(fp_in,"%c",&c);
}

/* Matrice delle profondità */
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
        fscanf(fp_in,"%2c",&u);
        z[i][j]=8*(int)u;

if (z[i][j] > 262144)
z[i][j] = - 65536 * 8 + z[i][j];
}};

/* Ora abbiamo i dati e conosciamo il file di input */
fclose (fp_in);

printf("Dati profondità prelevati\n\n");
```

```
/* Preleva dati colore dal file sunraster */

/* File sun */
strcpy(immfile,argv[1]);
strcat(immfile,".sun");


/* Apre il file sunraster */
if ((fp_imm = fopen (immfile, "r")) == NULL)
   { printf("Input file sun inesistente\n"); exit(); };

/* Salta l'header che non interessa */
for (j=0;j<28;j++){
fscanf(fp_imm,"%c",&c);
}

/* Legge la lunghezza della mappa */
for (j=0;j<4;j++){
fscanf(fp_imm,"%c",&lunghezza_mappa[j]);
}
lunghezza_mappa_int=lunghezza_mappa[3]+
                    lunghezza_mappa[2]*256+
                    lunghezza_mappa[1]*65536+
                    lunghezza_mappa[0]*16777216;


/* Legge la mappa */
for (contatore=0;contatore<lunghezza_mappa_int;contatore++){
fscanf(fp_imm,"%c",&dato_char[contatore]);
        dato_colore[contatore]=dato_char[contatore];
}


/* Matrice dell'immagine */
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
fscanf(fp_imm,"%c", &a);
        int_a=(int)a;
immagine[i][j]=dato_colore[int_a];
}};


/* Chiude il file sun */
fclose (fp_imm);

/* L'incremento della x deve essere di
       293  per le scansioni di 150 mm
       500  per le scansioni di 256 mm
       750  per le scansioni di 384 mm
*/

printf("Dati colore prelevati\n\n");


switch (range) {

  case 150: { passo_della_i=293;
              break; }

  case 256: { passo_della_i=500;
              break; }

  case 384: { passo_della_i=750;
              break; }
}

/* STIRAMENTO */

/* Inizializzazioni */
for (j=0; j<yposts; j++)
```

```
{
  for (i=0; i<xposts; i++)
  {
    alfa_righe[i][j] = 0;
    alfa_colonne[i][j] = 0;
    distanza_righe[i][j] = 0;
    distanza_colonne[i][j] = 0;
    distanza_vera_righe[i][j] = 0;
    distanza_vera_colonne[i][j] = 0;
    distanza_pixel_righe[i][j] = 0;
    distanza_pixel_colonne[i][j] = 0;
    immagine_2[i][j] = 0;
  }
}
printf("Inizializzazione eseguita\n\n");


/* Crea le matrici degli angoli alfa */

   /* Sulle righe */

   for (j = 0; j<yposts; j++)
   {
     for (i = 0; i<xposts-1; i++)
     {
       alfa_righe[i][j] = atan ((z[i+1][j] - z[i][j]) / passo_della_i);
     }
   }

   /* Sulle colonne */
   for (i = 0; i<xposts; i++)
   {
     for (j = 0; j<yposts-1; j++)
     {
       alfa_colonne[i][j] = atan ((z[i][j+1] - z[i][j]) / 300);
     }
   }
printf("Angoli alfa calcolati\n\n");


/* CREA LE MATRICI DELLA DISTANZA TRA PIXEL ADIACENTI */

/* Sulle righe */

for (j=0; j<yposts; j++)
{
  for (i=xposts/2; i>=0; i--)
  {
    distanza_righe[i][j]=passo_della_i /(cos(atan(alfa_righe[i][j])));
  }
  for (i=xposts/2; i<xposts; i++)
  {
    distanza_righe[i][j]=passo_della_i /(cos(atan(alfa_righe[i][j])));
  }
}

/* Sulle colonne */

for (i=0; i<xposts; i++)
{
  for (j=yposts/2; j>=0; j--)
  {
    distanza_colonne[i][j]=300/(cos(atan(alfa_colonne[i][j])));
  }
  for (j=yposts/2; j<yposts; j++)
  {
    distanza_colonne[i][j]=300/(cos(atan(alfa_colonne[i][j])));
  }
}
printf("Distanze tra pixel adiacenti calcolate\n\n");
```

```
/* CREA LE MATRICI DELLE DISTANZE VERE DAL CENTRO */

/* Sulle righe */

for (j=0; j<yposts; j++)
{
  for (i=xposts/2; i>=0; i--)
  {
    for (contatore=xposts/2; contatore>=i; contatore--)
      distanza_vera_righe[i][j] += distanza_righe[contatore][j];
  }
  for (i=xposts/2; i<xposts; i++)
  {
    for (contatore=xposts/2; contatore<=i; contatore++)
      distanza_vera_righe[i][j] += distanza_righe[contatore][j];
  }
}


/* Sulle colonne */

for (i=0; i<xposts; i++)
{
  for (j=yposts/2; j>=0; j--)
  {
    for (contatore=yposts/2; contatore>=j; contatore--)
      distanza_vera_colonne[i][j] += distanza_colonne[i][contatore];
  }
  for (j=yposts/2; j<yposts; j++)
  {
    for (contatore=yposts/2; contatore<=j; contatore++)
      distanza_vera_colonne[i][j] += distanza_colonne[i][contatore];
  }
}
printf("Distanze dal centro calcolate\n\n");


/* CREA LE MATRICI DELLE DISTANZE IN PIXEL */

for (j=0; j<yposts; j++)
{
  for (i=0; i<xposts; i++)
  {
    distanza_pixel_righe[i][j] = distanza_vera_righe[i][j] / passo_della_i + 0.5;
    distanza_pixel_colonne[i][j] = distanza_vera_colonne[i][j] / 300 + 0.5;
  }
}
printf("Matrice delle distanze in pixel calcolata\n\n");


/* CREA LA NUOVA MATRICE DELL'IMMAGINE STIRATA */

for (j=0; j<=yposts/2; j++)
{
  for (i=0; i<=xposts/2; i++)
  {
    if ((xposts/2-distanza_pixel_righe[i][j]<0) || (yposts/2-distanza_pixel_colonne[i][j]<0)) continue;
    immagine_2[xposts/2-distanza_pixel_righe[i][j]][yposts/2-distanza_pixel_colonne[i][j]]=immagine[i][j];
  }
}

for (j=0; j<=yposts/2; j++)
{
  for (i=xposts/2; i<xposts; i++)
  {
    if ((xposts/2+distanza_pixel_righe[i][j]>xposts) || (yposts/2-distanza_pixel_colonne[i][j]<0)) continue;
    immagine_2[xposts/2+distanza_pixel_righe[i][j]][yposts/2-distanza_pixel_colonne[i][j]] = immagine[i][j];
  }
}

for (j=yposts/2; j<yposts; j++)
{
```

```
  for (i=xposts/2; i<xposts; i++)
  {
    if ((xposts/2+distanza_pixel_righe[i][j]>xposts) || (yposts/2+distanza_pixel_colonne[i][j]>yposts)) continue;
    immagine_2[xposts/2+distanza_pixel_righe[i][j]][yposts/2+distanza_pixel_colonne[i][j]] = immagine[i][j];
  }
}

for (j=yposts/2; j<yposts; j++)
{
  for (i=0; i<=xposts/2; i++)
  {
    if ((xposts/2-distanza_pixel_righe[i][j]<0) || (yposts/2+distanza_pixel_colonne[i][j]>yposts)) continue;
    immagine_2[xposts/2-distanza_pixel_righe[i][j]][yposts/2+distanza_pixel_colonne[i][j]] = immagine[i][j];
  }
}
printf("Matrice nuova immagine pronta\n\n");


/* SCRITTURA DEL FILE SUNRASTER */

/* Open file in scrittura */
fp_out = fopen (outfile,"w");

/* Magic number */
fprintf(fp_out,"%c%c%c%c",89,166,106,149);

/* Number of columns */
fprintf(fp_out,"%c%c%c%c",0,0,2,0);

/* Number of lines */
fprintf(fp_out,"%c%c%c%c",0,0,2,0);

/* Number of bits per pixel */
fprintf(fp_out,"%c%c%c%c",0,0,0,8);

/* Image data lenght in byte */
fprintf(fp_out,"%c%c%c%c",0,4,0,0);

/* Type */
fprintf(fp_out,"%c%c%c%c",0,0,0,1);

/* Map type */
fprintf(fp_out,"%c%c%c%c",0,0,0,1);

/* Map lenght */
fprintf(fp_out,"%c%c%c%c",0,0,3,0);

/* Scrittura mappa */
for (j=0; j<3; j++) {
  for (i=0; i<256; i++)
    fprintf(fp_out,"%c",i);
}

/* Scrittura matrice dell'immagine */
for (i=0; i<xposts; i++){
for (j=0; j<yposts; j++){
  fprintf(fp_out,"%c",immagine_2[i][j]);
}}

/* Close file */
fclose (fp_out);

}
```

E Listato del programma STI

```
/*
                                sti.c

Stira files Echo 3D dopo averli convertiti in un formato
leggibile da questo programma.

Il file di input contenente i dati di profondità deve essere nel
formato standard di Echo, ad esempio un file creato dallo
scanner Cyberware.
Il file dei colori deve essere in formato sunraster a scala di grigi.
L'output sarà un file nominato input_sti.sun.

*/


#include <stdio.h>
#include <string.h>
#include <math.h>
#include <malloc.h>


int **imatrix(nrl,nrh,ncl,nch)
int nrl,nrh,ncl,nch;
{
int i,**m;

m=(int **)malloc((unsigned) (nrh-nrl+1)*sizeof(int*));
if (!m) printf("Allocazione 1 fallita in imatrix()");
m -= nrl;

for(i=nrl;i<=nrh;i++) {
m[i]=(int *)malloc((unsigned) (nch-ncl+1)*sizeof(int));
if (!m[i]) printf("Allocazione 2 fallita in imatrix()");
m[i] -= ncl;
}
return m;
}




main(argc,argv)

#define PASSO_DELLA_J 300

int argc;
char *argv[];
{

FILE *fp_in, *fp_out, *fp_imm;
char infile[32],
     outfile[32],
     immfile[32],
     c,
     a,
     dato_char[65536],
     lunghezza_mappa[4];

unsigned short u;

double pass;

double alfa_righe[512][512],
       alfa_colonne[512][512],

    distanza_righe[512][512],
    distanza_colonne[512][512],
    distanza_vera_righe[512][512],
    distanza_vera_colonne[512][512];
```

```
int **z,
    **imatrix(),
    int_a,
    header,  // Lunghezza header del file delle profonditá
    i,j,contatore,                    // Indici
    arrivo,
    arrivo2,
    xposts,               // Dimensione x dell'immagine
    yposts,               // Dimensione y dell'immagine
    n,                    // Numero totale di punti (x*y)
    s,
    range,                // Dimensione della scansione
    passo_della_i,        // Passo della lg in micron
    dato_colore[65536],   // Dati della mappa
    immagine[512][512],   // Immagine in scala di grigi
    immagine_2[512][512], // Immagine stirata
    lunghezza_mappa_int;


// Test dell'argomento
if ( argc == 1 )
        { printf("Devi dare un argomento\n");
          exit(1);
};

// Quale file è in input?
strcpy(infile,argv[1]);
if ((fp_in = fopen (infile, "r")) == NULL)
   { printf("Input file inesistente\n"); exit(); };

// File di output
strcpy(outfile,argv[1]);
strcat(outfile,"_sti.sun");


xposts=512; yposts=512;
n=xposts*yposts;

// Richiede il range di scansione
  do {
  printf("\nInserire il range di scansione (150, 256, 384 mm): ");
  scanf("%d", &range);
  printf("\n");
}

// Controlla esattezza del range
while (range!=150 && range!=256 && range!=384);


// Definisce la dimensione della matrice e il range degli indici
z=imatrix(0,xposts-1,0,yposts-1);

header = strlen( argv[1] ) + 252;
for (j = 0; j < header; j++){
fscanf(fp_in,"%c",&c);
}

// Matrice delle profondità
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
        fscanf(fp_in,"%2c",&u);
        z[i][j]=8*(int)u;

if (z[i][j] > 262144)
z[i][j] = - 65536 * 8 + z[i][j];
}};

// Ora abbiamo i dati e conosciamo il file di input
fclose (fp_in);

printf("Dati profondità prelevati\n\n");
```

```
// PRELEVA I DATI COLORE DAL FILE SUNRASTER

// File sun
strcpy(immfile,argv[1]);
strcat(immfile,".sun");


// Apre il file sunraster
if ((fp_imm = fopen (immfile, "r")) == NULL)
   { printf("Input file sun inesistente\n"); exit(); };

// Salta l'header che non interessa
for (j=0;j<28;j++){
fscanf(fp_imm,"%c",&c);
}

// Legge la lunghezza della mappa
for (j=0;j<4;j++){
fscanf(fp_imm,"%c",&lunghezza_mappa[j]);
}
lunghezza_mappa_int=lunghezza_mappa[3]+
                    lunghezza_mappa[2]*256+
                    lunghezza_mappa[1]*65536+
                    lunghezza_mappa[0]*16777216;


// Legge la mappa
for (contatore=0;contatore<lunghezza_mappa_int;contatore++){
fscanf(fp_imm,"%c",&dato_char[contatore]);
        dato_colore[contatore]=dato_char[contatore];
}


// Matrice dell'immagine
for (i=0;i<xposts;i++){
for (j=0;j<yposts;j++){
fscanf(fp_imm,"%c", &a);
        int_a=(int)a;
immagine[i][j]=dato_colore[int_a];
}};


// Chiude il file sun
fclose (fp_imm);

/* L'incremento della x deve essere di
       293  per le scansioni di 150 mm
       500  per le scansioni di 256 mm
       750  per le scansioni di 384 mm
*/

printf("Dati colore prelevati\n\n");


switch (range) {

  case 150: { passo_della_i=293;
              break; }

  case 256: { passo_della_i=500;
              break; }

  case 384: { passo_della_i=750;
              break; }
}

// STIRAMENTO

// Inizializzazioni
for (j=0; j<yposts; j++)
```

```
{
  for (i=0; i<xposts; i++)
  {
    alfa_righe[i][j] = 0;
    alfa_colonne[i][j] = 0;
    distanza_righe[i][j] = 0;
    distanza_colonne[i][j] = 0;
    distanza_vera_righe[i][j] = 0;
    distanza_vera_colonne[i][j] = 0;
    immagine_2[i][j] = 0;
  }
}
printf("Inizializzazione eseguita\n\n");


// Crea le matrici degli angoli alfa

   // Sulle righe

   for (j = 0; j<yposts; j++)
   {
     for (i = 0; i<xposts-1; i++)
     {
       alfa_righe[i][j] = atan ((z[i+1][j] - z[i][j]) / passo_della_i);
     }
   }

   // Sulle colonne
   for (i = 0; i<xposts; i++)
   {
     for (j = 0; j<yposts-1; j++)
     {
       alfa_colonne[i][j] = atan ((z[i][j+1] - z[i][j]) / PASSO_DELLA_J);
     }
   }
printf("Angoli alfa calcolati\n\n");


// CREA LE MATRICI DELLA DISTANZA TRA PIXEL ADIACENTI

// Sulle righe

for (j=0; j<yposts; j++)
{
  for (i=xposts/2; i>=0; i--)
  {
    distanza_righe[i][j]=passo_della_i /(cos(atan(alfa_righe[i][j])));
  }
  for (i=xposts/2; i<xposts; i++)
  {
    distanza_righe[i][j]=passo_della_i /(cos(atan(alfa_righe[i][j])));
  }
}

// Sulle colonne

for (i=0; i<xposts; i++)
{
  for (j=yposts/2; j>=0; j--)
  {
    distanza_colonne[i][j]=PASSO_DELLA_J/(cos(atan(alfa_colonne[i][j])));
  }
  for (j=yposts/2; j<yposts; j++)
  {
    distanza_colonne[i][j]=PASSO_DELLA_J/(cos(atan(alfa_colonne[i][j])));
  }
}
printf("Distanze tra pixel adiacenti calcolate\n\n");


// CREA LE MATRICI DELLE DISTANZE VERE DAL CENTRO
```

```
// Sulle righe

for (j=0; j<yposts; j++)
{
  for (i=xposts/2; i>=0; i--)
  {
    for (contatore=xposts/2; contatore>=i; contatore--)
      {distanza_vera_righe[i][j] += distanza_righe[contatore][j];}

  }
  for (i=xposts/2; i<xposts; i++)
  {
    for (contatore=xposts/2; contatore<=i; contatore++)
      {distanza_vera_righe[i][j] += distanza_righe[contatore][j];}

  }
}


/*
// Sulle colonne

for (i=0; i<xposts; i++)
{
  for (j=yposts/2; j>=0; j--)
  {
    for (contatore=yposts/2; contatore>=j; contatore--)
      {distanza_vera_colonne[i][j] += distanza_colonne[i][contatore];}

  }
  for (j=yposts/2; j<yposts; j++)
  {
    for (contatore=yposts/2; contatore<=j; contatore++)
      {distanza_vera_colonne[i][j] += distanza_colonne[i][contatore];}

  }
}
*/
printf("Distanze dal centro calcolate\n\n");



// CREA LA NUOVA MATRICE DELL'IMMAGINE STIRATA


for (j=0; j<yposts; j++)
{
  for (i=xposts/2; i<xposts; i++)
  {
    for (contatore=xposts/2; contatore<xposts-1; contatore++)
    {
      if ( (((i-xposts/2)*passo_della_i) >distanza_vera_righe[contatore][j]) &&
           (((i-xposts/2)*passo_della_i) <distanza_vera_righe[contatore+1][j]))
         immagine_2[i][j]=(immagine[contatore][j]*
                 distanza_righe[contatore+1][j]+
                 immagine[contatore+1][j]*distanza_righe[contatore][j])/
          (distanza_righe[contatore][j]+distanza_righe[contatore+1][j]);


    }

  }


  for (i=xposts/2; i>0; i--)
  {
    for (contatore=xposts/2; contatore>0; contatore--)
    {
      if ( (((xposts/2-i)*passo_della_i) >distanza_vera_righe[contatore][j]) &&
           (((xposts/2-i)*passo_della_i) <distanza_vera_righe[contatore-1][j]))
       immagine_2[i][j]=(immagine[contatore][j]*
                 distanza_righe[contatore-1][j]+
```

```
                immagine[contatore-1][j]*distanza_righe[contatore][j])/
          (distanza_righe[contatore][j]+distanza_righe[contatore-1][j]);

    }

  }
}

printf("Matrice nuova immagine pronta\n\n");


// SCRITTURA DEL FILE SUNRASTER

// Open file in scrittura
fp_out = fopen (outfile,"w");

// Numero magico del formato SUN
fprintf(fp_out,"%c%c%c%c",89,166,106,149);

// Numero di colonne
fprintf(fp_out,"%c%c%c%c",0,0,2,0);

// Numero di linee
fprintf(fp_out,"%c%c%c%c",0,0,2,0);

// Numero di bit per pixel
fprintf(fp_out,"%c%c%c%c",0,0,0,8);

// Numero dei byte dei dati immagine
fprintf(fp_out,"%c%c%c%c",0,4,0,0);

// Tipo
fprintf(fp_out,"%c%c%c%c",0,0,0,1);

// Tipo mappa
fprintf(fp_out,"%c%c%c%c",0,0,0,1);

// Lunghezza mappa
fprintf(fp_out,"%c%c%c%c",0,0,3,0);

// Scrittura mappa
for (j=0; j<3; j++) {
  for (i=0; i<256; i++)
    fprintf(fp_out,"%c",i);
}

// Scrittura matrice dell'immagine
for (i=0; i<xposts; i++){
for (j=0; j<yposts; j++){
  fprintf(fp_out,"%c",immagine_2[i][j]);
}}

// Chiusura file
fclose (fp_out);

}
```

Riferimenti bibliografici

[1] Cyberware, “Echo User and Reference Manual”, Revision 8-93

[2] Brian W. Kernighan, Dennis M. Ritchie, ”The C Programming Language”, Prentice-Hall, 1977

[3] John Bradley, “XV image viewer”, University of Pennsylvania, 1994

[4] Foto aerea del Centro Comune di Ricerca sito di Ispra presa dal sito internet: `http://www.varesenews.com/cronaca/99/dicembre/13_12ccr.htm`