Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

Stereogramma di immagini monocromatiche

P. Andronico, R. Guerrini, A. Marchetti
L. Azzarelli, M. Chimenti

Nota Interna B4-12

Marzo 1990

INDICE

INTRODUZIONE

L'I.E.I. dispone di un sistema per l'acquisizione e l'elaborazione di immagini digitali bidimensionali, chiamato BIS386.

Scopo di questo lavoro è dotare tale sistema di un meccanismo per rappresentare le suddette immagini in forma tridimensionale. L'algoritmo sfruttato per ottenere tale rappresentazione viene denominato STEREOGRAMMA (*).

Questa applicazione permette di evidenziare dettagli dell'immagine talvolta impercettibili nella visione bidimensionale.

Il programma prevede una ristretta gamma di parametri per la rappresentazione dello stereogramma, in modo da semplificare al massimo l'utilizzo da parte dell'utente.

(*) *STEREOGRAMMA (dal greco* στερεοζ *«solido» e* γραμμα *«disegno»):*
rappresentazione in rilievo delle qualità che intervengono in un fenomeno qualunque, cioè una rappresentazione grafica nella quale venga fatto uso di costruzioni geometriche nello spazio. In senso più stretto lo stereogramma è un particolare diagramma in cui le intensità di uno o più fenomeni vengono rappresentate mediante solidi o volumi, così come un diagramma lineare rappresenta intensità mediante linee e un istogramma (diagramma superficiale) mediante superfici.
La denominazione di stereogramma venne suggerita da A. Messedaglia per indicare particolari figure di geometria solida, ideate da G. Zenner e perfezionate e diffuse da I. Perozzo, per dare immagine, in forma chiara e suggestiva, di certi fenomeni statistici e in particolare del movimento di una popolazione attraverso il tempo.
(citazione dall'ENCICLOPEDIA ITALIANA TRECCANI)

MEZZI USATI PER L'ELABORAZIONE

La figura 1 illustra la configurazione hardware generica del sistema BIS386. Finora, sia le immagini digitalizzate, che eventuali grafici, venivano visualizzati sul display controllato dalla PIP, non sfruttando le enormi capacità grafiche della VGA.

Inoltre, vedere contemporaneamente le due rappresentazioni della stessa immagine, permette di trarne, dal confronto, il maggior numero di informazioni.

Si è quindi deciso di visualizzare lo stereogramma sul monitor VGA.

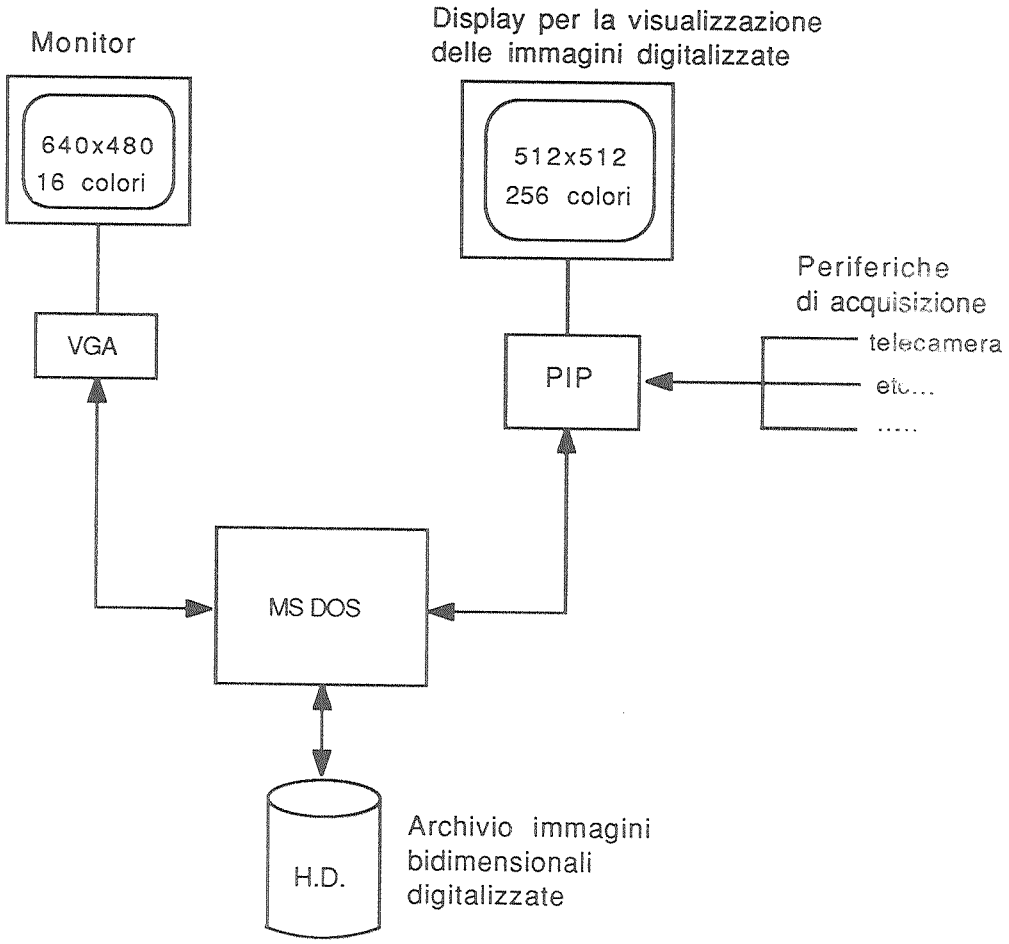


Fig. 1: Configurazione hw del sistema BIS386

DESCRIZIONE GENERALE DELL'ALGORITMO

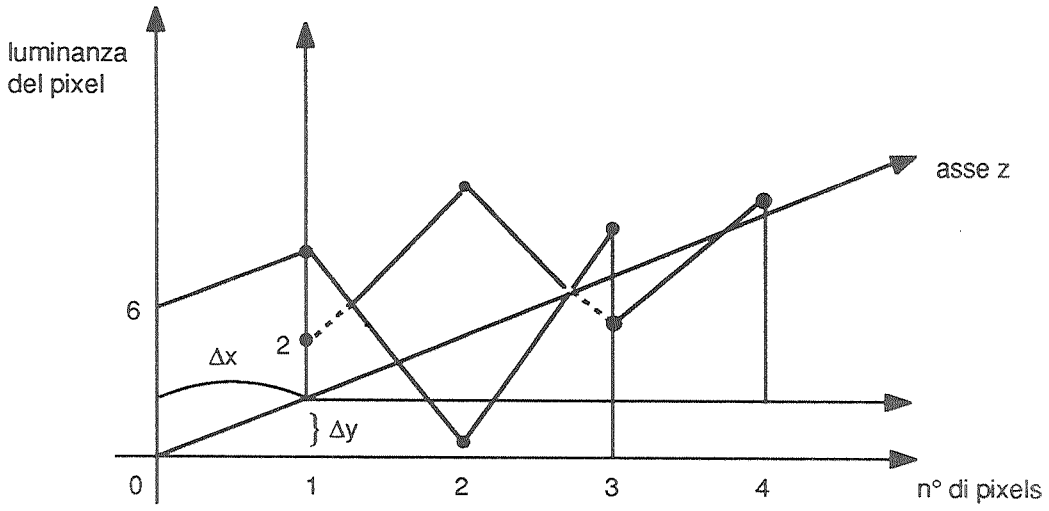
Ogni riga dell'immagine viene rappresentata in due dimensioni, considerando sull'asse x del grafico il pixel in esame e sull'asse y il suo valore di luminosità che può variare nell'intervallo [0÷255].

Nello stereogramma, per ottenere la tridimensionalità, si fa in modo che ogni curva che rappresenta una riga della matrice immagine venga slittata, rispetto alla precedente, lungo l'asse x di un tratto Δx e lungo l'asse y di un tratto Δy (v. Fig. 2a e Fig. 2b). Se $\Delta x = \Delta y = \Delta$, l'asse z apparirà orientato di 45° rispetto alle ascisse.

Il passo di slittamento Δ può essere facilmente variato da programma, in base alle esigenze di lettura dello stereogramma da visualizzare.

2	10	4	8	2ª riga
6	8	1	9	1ª riga

Fig. 2a: Matrice di un'immagine con due righe di quattro pixels.



Ponendo $\Delta x = \Delta y = \Delta$, l'asse z risulterà orientato di 45° rispetto agli altri assi.

Fig. 2b: Stereogramma dell'immagine rappresentata dalla matrice di Fig. 2a.

ADATTAMENTO HARDWARE DELL'ALGORITMO

Le caratteristiche hardware della scheda grafica PIP permettono di memorizzare fino a 4 immagini 512x512 o un'unica immagine 1024x1024 pixels, mentre il video ad essa collegato consente di visualizzare al massimo 512x512 pixels.

Al fine di permettere un confronto diretto tra l'immagine visualizzata e il suo stereogramma, si è deciso di lavorare con matrici di un'immagine di dimensioni massime di 512x512 pixels.

Per rappresentare fedelmente lo stereogramma dell'intera immagine, cioè senza lasciare spazi tra la visualizzazione di una riga e la successiva ($\Delta x = 1$), sarebbero necessarie 512 righe più 255 righe per un eventuale picco massimo nell'ultima riga, per un totale di 767 righe. Per quanto riguarda le colonne, definendo un passo di slittamento pari ad 1 ($\Delta y = 1$), sarebbero occorse 512 colonne più 512 slittamenti, per un totale di 1024 colonne. La VGA a nostra disposizione ci permette di settare una configurazione grafica massima di 640x480 pixels, che è evidentemente insufficiente. Questo ci induce a fare delle scelte di scansione di riga e colonna della matrice immagine, per la rappresentazione dello stereogramma, senza ledere il principio di portabilità del pacchetto.

In base alle dimensioni disponibili del video, abbiamo calcolato che il numero ideale di righe e di colonne da visualizzare si ottiene scandendo la matrice immagine una riga ogni due e tre colonne ogni quattro e moltiplicando i valori di luminosità dei pixels per un fattore di amplificazione uguale a 0,2. Lo stereogramma così ridotto non si discosta molto dalla sua versione integrale.

Calcolo delle dimensioni della finestra video

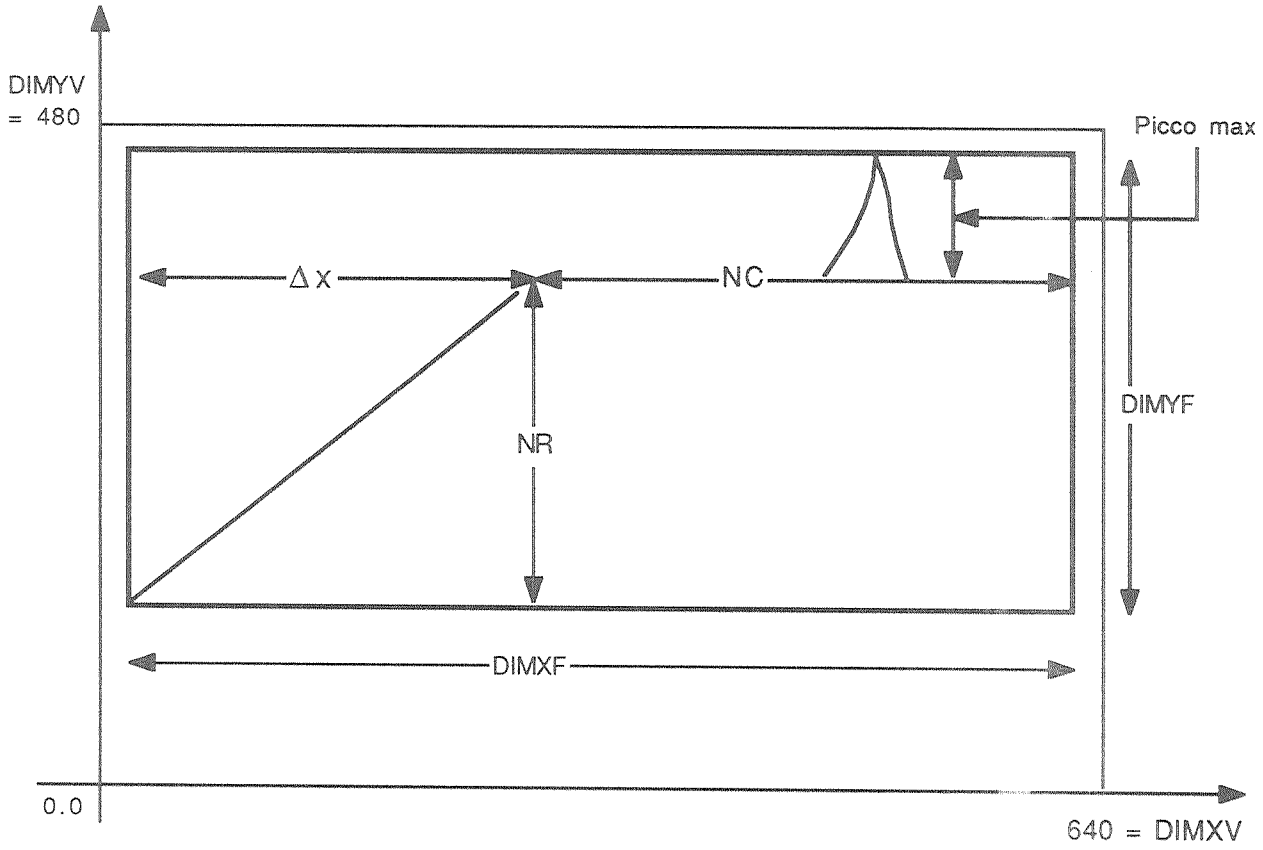


Fig.3: Finestra per la costruzione dello stereogramma sul video della VGA

DIM_X VIDEO = DIMXV = 480

DIM_Y VIDEO = DIMYV = 640

NUMERO COLONNE STEREOGRAMMA = NC = 512 * (3/4) = 341

NUMERO RIGHE STEREOGRAMMA = NR = 512 * (1/2) = 256

PICCO MASSIMO = 255 * 0,2 = 51

$\Delta x = \Delta y = 1$

DIM_Y FINESTRA = DIMYF = NR + PICCO MAX = 256 + 51 = 307 < 480

DIM_X FINESTRA = DIMXF = NC + Δx = 341 + 256 = 597 < 640

Le aree del video destinate rispettivamente alla rappresentazione dello stereogramma (grafica) e ai comandi necessari ed eventuali commenti esplicativi (testo) vengono definite come nel disegno sotto riportato (interfaccia video come appare all'utente).

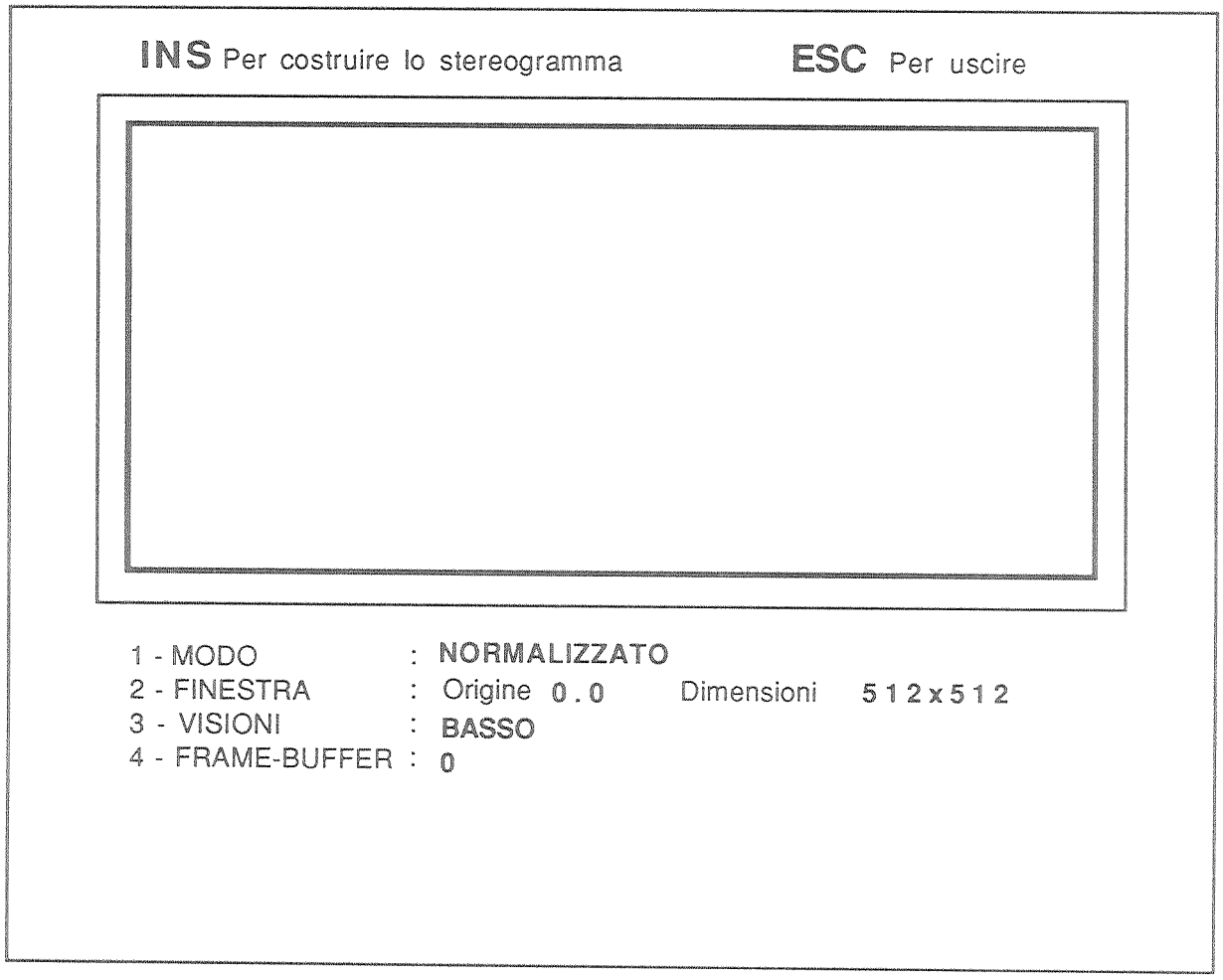


Fig. 4: Interfaccia video all'atto del lancio del programma.

Come già si è accennato precedentemente, per problemi di risoluzione delle due diverse schede usate, si è dovuta fare una scelta nella scansione delle righe e delle colonne della matrice immagine da poter visualizzare nello stereogramma.

Siamo quindi partiti dalla possibilità di avere due modi diversi di scansione, NORMALIZZATO e ZOOMx2.

Il modo NORMALIZZATO consente di costruire lo stereogramma dell'intera immagine visualizzata (512x512 pixels) o di una parte di essa, selezionando una finestra sull'immagine, mediante cursori gestiti da tastiera.

Questo è possibile leggendo una riga ogni due e, per ogni riga, due punti ogni tre. Nel caso in cui si voglia considerare solo una parte dell'immagine, il tipo di scansione rimarrà quella adottata in precedenza, ma lo stereogramma occuperà porzioni più piccole della finestra video.

Il modo ZOOMx2 prevede invece la costruzione dello stereogramma scandendo tutte le righe e tutte le colonne della matrice.

Ciò comporta che solo una porzione di essa possa essere rappresentata in tre dimensioni. Tale porzione si può vedere come una finestra di dimensioni costanti (256x341) con possibilità di spostamento sopra tutta l'immagine, tramite cursori gestiti da tastiera (Fig. 5).

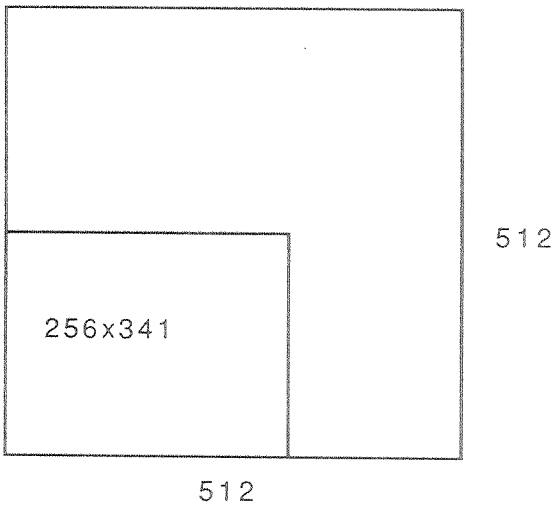


Fig. 5: Posizione iniziale della finestra nel modo ZOOMx2 e sue dimensioni fisse

Le caratteristiche hardware della scheda grafica MATROX PIP-1024A e le dimensioni massime delle immagini a nostra disposizione, ci consentono di disporre di quattro diverse immagini monocromatiche 512x512, memorizzate nella video-RAM della scheda (v. Fig. 6). Per sfruttare questa possibilità, si è previsto che l'utente possa liberamente scegliere su quale immagine di volta in volta intende lavorare. Tale scelta sarà gestita direttamente tramite un comando e può avvenire in qualsiasi momento.

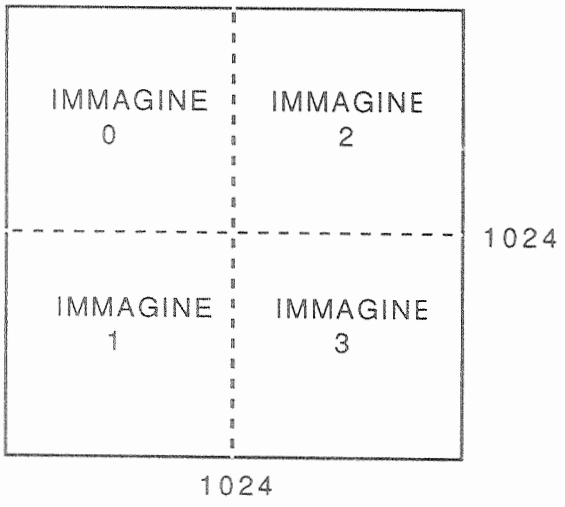


Fig. 6: Quadri memoria immagine

Per fornire maggiori informazioni nella rappresentazione tridimensionale, si è scelto di poter selezionare diversi punti di vista (BASSO, ALTO, DESTRA e SINISTRA).

Tali opzioni possono consentire all'utente di cogliere particolari altrimenti persi.

ARCHITETTURA DEL PACCHETTO

La struttura generale del pacchetto realizzato, è illustrato nella figura 7.

L'utente può intervenire nei seguenti parametri:

1. Scelta del modo di lettura della matrice immagine
2. Scelta della porzione di immagine di cui si vuole costruire lo stereogramma
3. Scelta del lato in cui idealmente si pone l'utente nel guardare l'immagine
4. Scelta di una fra le quattro immagini caricabili sulla video-RAM della scheda PIP.

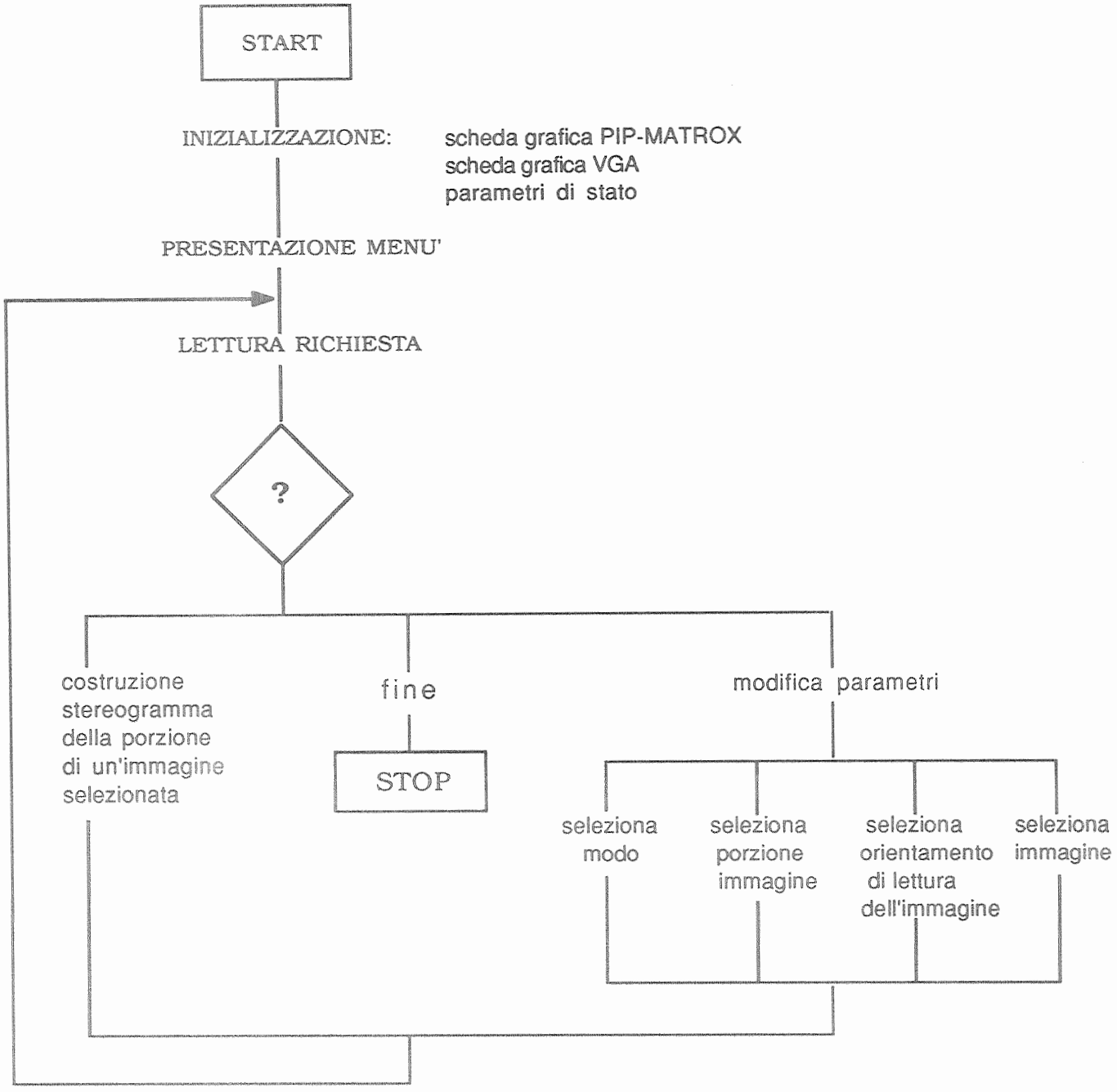


Fig. 7: Grafico dell'architettura del programma

APPENDICE A

ALCUNI ASPETTI DELLA SCANSIONE DELL'IMMAGINE

L'immagine si può leggere nei quattro sensi: alto, basso, sinistra, destra.

Nel caso Normalizzato verranno lette una colonna ogni due e all'interno di ogni colonna, due punti ogni tre, qualunque sia il senso.

Nel caso Zoomx2 si possono seguire due vie:

1. Visualizzare una finestra rettangolare (256x341 punti), che provoca dei tagli nella visualizzazione dello stereogramma da destra e da sinistra
2. Visualizzare una finestra quadrata (256x256 punti), che permette la lettura di tutte le righe e di tutte le colonne, e non provoca perdite di informazioni nella costruzione dello stereogramma da destra e da sinistra.

Quest'ultima alternativa impone la scelta della finestra di rappresentazione dello stereogramma sul monitor della VGA:

- utilizzando la stessa area la finestra non viene occupata completamente dal grafico;
- solo in questo caso, si può costruire una finestra più piccola che verrà quindi riempita completamente.

Si sceglie di utilizzare la finestra rettangolare per mantenere costante l'interfaccia visiva della VGA e per avere una maggiore rappresentazione di informazioni nelle visioni dal Basso e dall'Alto.

APPENDICE B

ALGORITMO DELLA FUNZIONE SETFINESTRA

La funzione SETFINESTRA permette di definire la porzione di immagine di cui si vuole costruire lo stereogramma. In base al modo di lettura della matrice immagine, cambia la modalità di selezione dell'area suddetta.

Modo = NORMALIZZATO

a. cancellazione del rettangolo definito in precedenza
b. disegna un cursore sul monitor PIP al centro dello schermo.
c. gestione del movimento del cursore da tastiera per selezionare la diagonale che definisce il rettangolo settaggio delle variabili:

   XPL_INF, YPL_INF, XPL_SUP, YPL_SUP

d. scrittura su monitor VGA delle dimensioni della finestra definita:

   XPL_SUP = XPL_INF = dx
   YPL_SUP = YPL_INF = dy

   scrittura su monitor VGA dell'origine logica della finestra:

   x_origine = XPL_INF
   y_origine = YPL_INF

e. costruzione del rettangolo (finestra) defintito dai due punti settati sulla PIP.

modo = ZOOMx2

a. cancellazione del rettangolo definito in precedenza
b. disegna un cursore su monitor PIP alle coordinate logiche (341x256) e setta le variabili X_CUR e Y_CUR
c. disegna su monitor PIP una finestra di dimensioni 341x256 tale che il vertice in alto a destra coincida con la posizione del cursore
d. scrittura su monitor VGA delle coordinate di origine della finestra

   x_origine = X_CUR
   y_origine = 511 - Y_CUR

e. scrittura sul monitor VGA delle dimensioni fisse della finestra PIP

   dx = 341
   dy = 256

f. definizione della nuova origine della finestra tramite cursori gestiti da tastiera
   1. inizializzazione variabile direzione
   2. inizializzazione variabile tasto_premuto
   3. richiesta di un input da tastiera per la gestione del movimento del cursore all'interno della finestra predefinita

   se tasto_premuto = ESC
   allora uscita da SETFINESTRA
   aggiornamento delle variabili:
   XPL_INF, XPL_SUP,
   YPL_INF, YPL_SUP

altrimenti se il movimento del cursore avviene all'interno, o sui bordi della finestra
allora

a. aggiornamento delle variabili:
   direzione, X_CUR, Y_CUR

b. cancellazione cursore e finestra precedente su PIP
c. scrittura cursore e finestra nella nuova posizione su PIP
d. scrittura nuova coordinata d'origine della finestra su monitor VGA

altrimenti torna al [3].

ALGORITMO DELLA FUNZIONE STEREOPC

Caso visione_att = BASSO (ALTO)

for ypl = YPL_INF+1 to YPL_SUP-1 step SCANSIONE_R_C

(**for** ypl = YPL_SUP-1 to YPL_INF+1 step SCANSIONE_R_C)

a. memorizzazione nel vettore buffer [512] della riga ypl della matrice immagine

b. scansione dei punti del vettore a partire da XPL_INF+1 fino a XPL_SUP-1 (XPL_SUP-1 a XPL_INF+1) in base al modo settato

c. costruzione dello stereogramma relativo alla riga letta.

Caso visione_att = DESTRA (SINISTRA)

for xpl = XPL_SUP-1 to **if** modo_att = ZOOMx2
XPL_SUP-254
else XPL_INF+1

(**for** xpl = XPL_INF+1 to **if** modo_att = ZOOMx2
XPL_INF+254
else XPL_SUP-1)

a. memorizzazione nel vettore buffer [512] di una colonna della matrice immagine

b. lettura dei punti del vettore a seconda del modo scelto

c. costruzione dello stereogramma relativo alla colonna letta

APPENDICE C

MODALITÀ D'USO DELL'ALGORITMO

L'interfaccia video che appare una volta lanciato il programma, visualizza due tasti funzione (INS e ESC) e quattro tasti per l'impostazione dei parametri.

I parametri impostati per default sono quelli rappresentati in figura 4 (le dimensioni della finestra visualizzata sul monitor PIP sono 512x512).

TASTI FUNZIONE

INS Costruisce lo stereogramma dell'area selezionata
ESC Esce dal programma chiedendo conferma

Per variare i parametri di default è sufficiente premere il tasto funzione corrispondente al parametro che si intende modificare. Il nuovo valore impostato viene evidenziato da una scritta in bianco brillante.

TASTI IMPOSTAZIONE PARAMETRI

F1 Seleziona il modo di lettura della matrice immagine
F2 Permette di scegliere la parte di immagine, utilizzando i tasti direzione, di cui si vuole costruire lo stereogramma
F3 Seleziona il lato dell'immagine da cui viene costruito lo stereogramma
F4 Seleziona un quadro della memoria immagine PIP

APPENDICE D

IMMAGINI

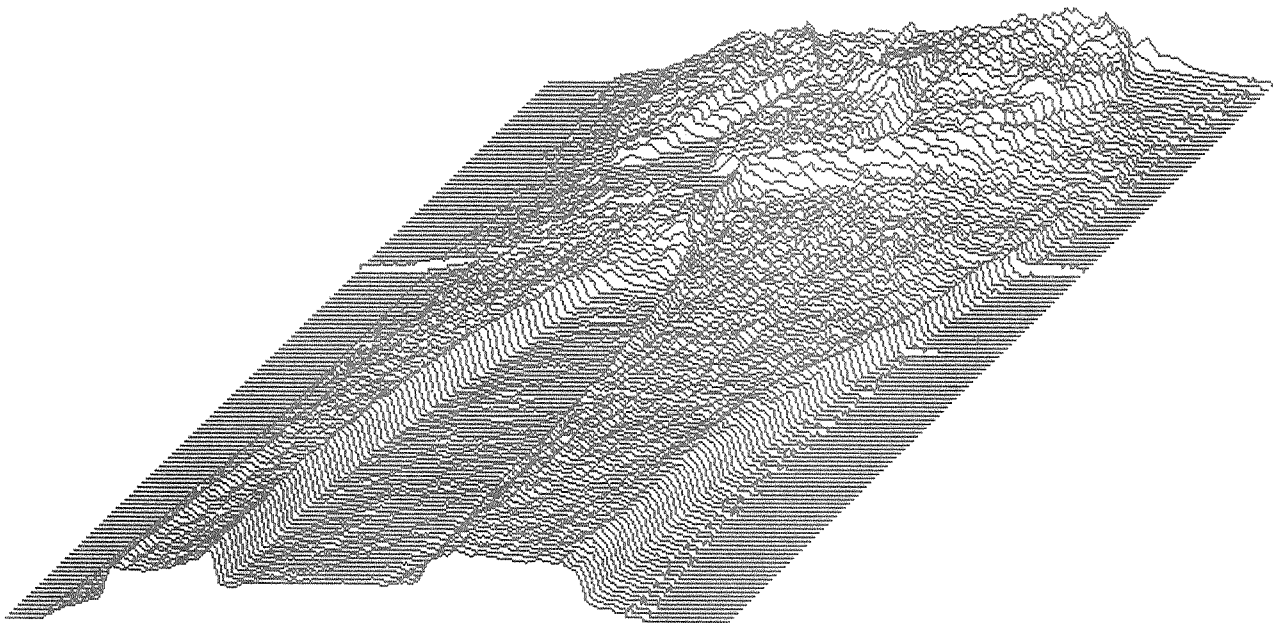


Fig. 8: Stereogramma in modo NORMALIZZATO, visione dal BASSO

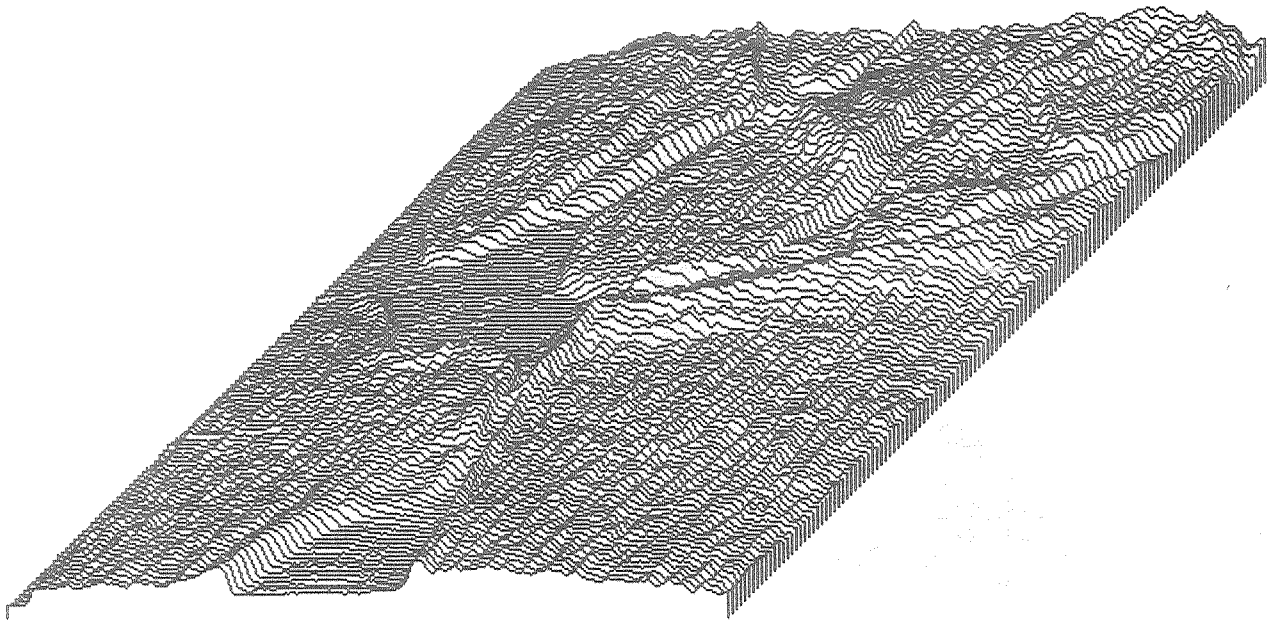


Fig. 9: Stereogramma in modo ZOOMx2, visione dal BASSO

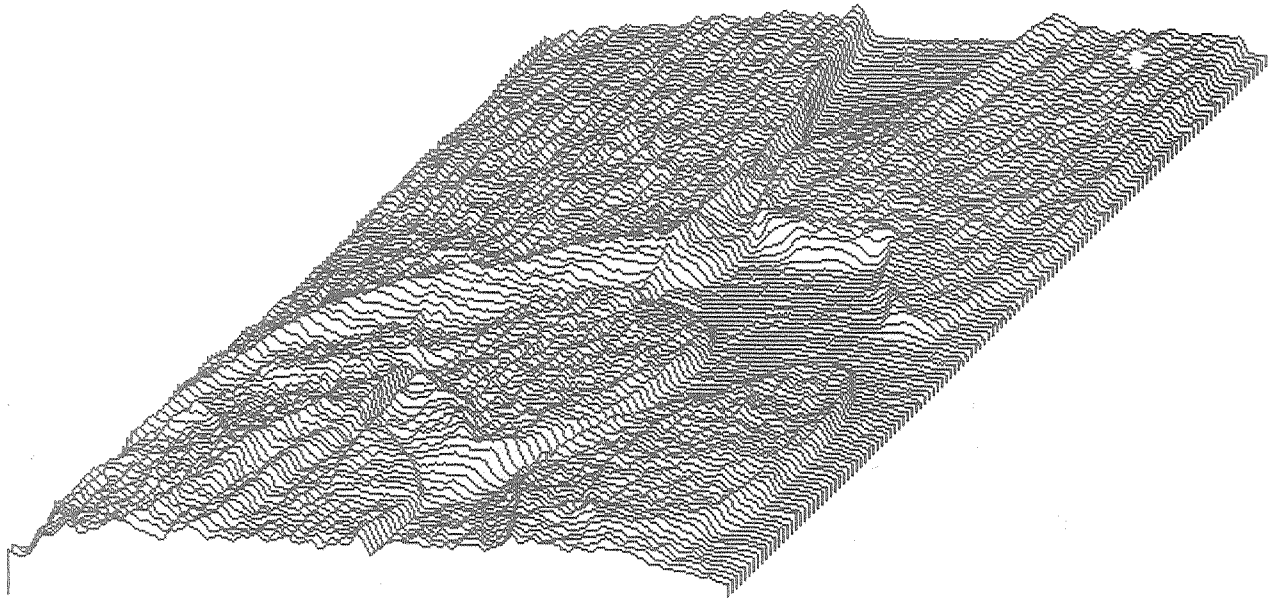


Fig. 10: Stereogramma in modo ZOOMx2, visione dall'ALTO

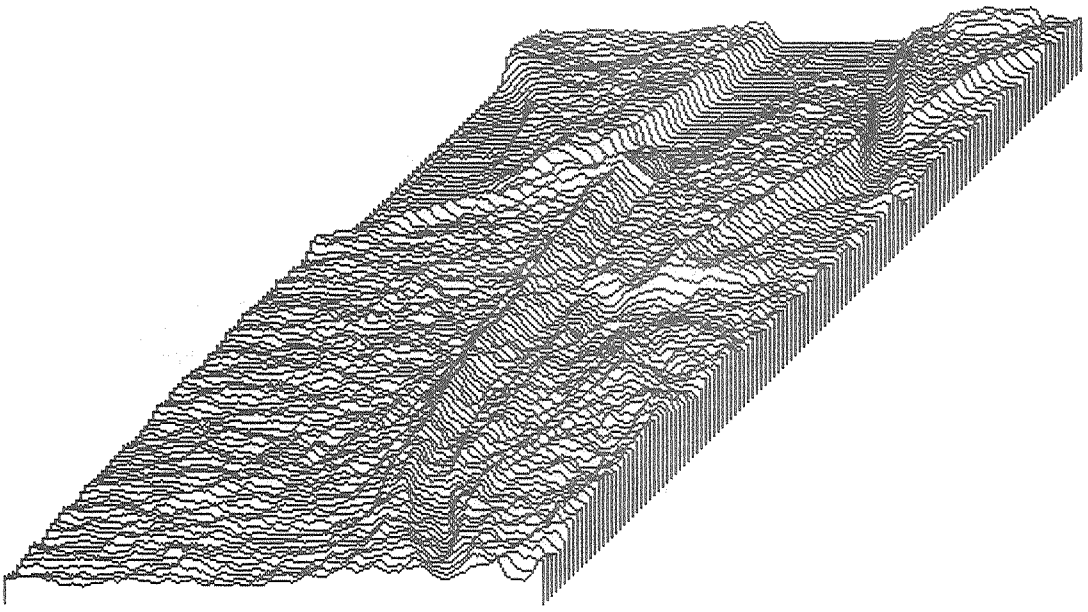


Fig. 11: Stereogramma in modo ZOOMx2, visione da DESTRA

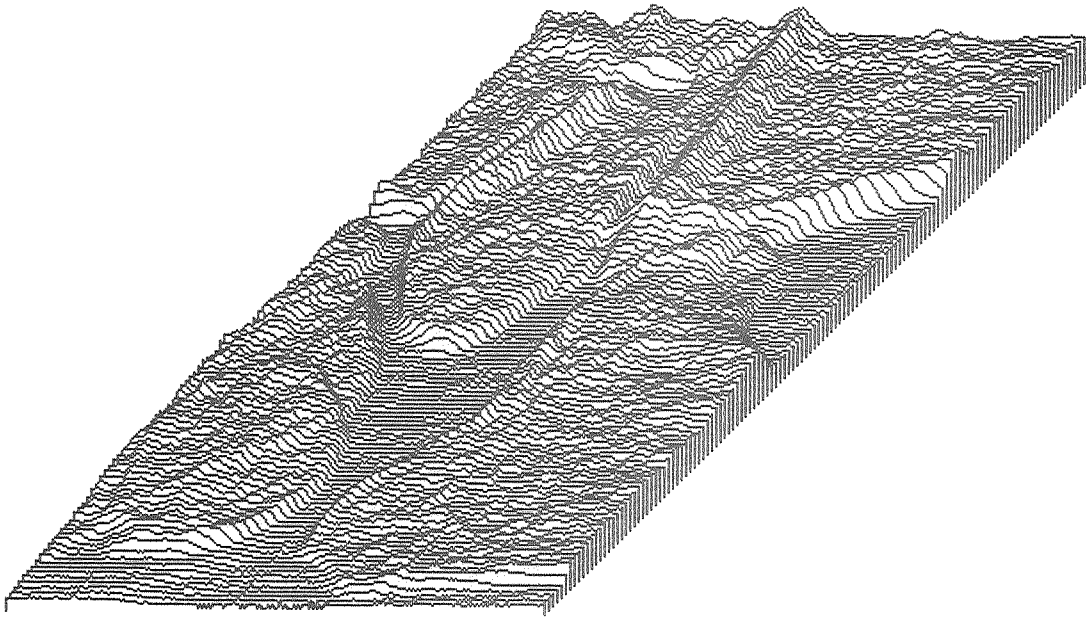


Fig. 12: Stereogramma in modo ZOOMx2, visione da SINISTRA

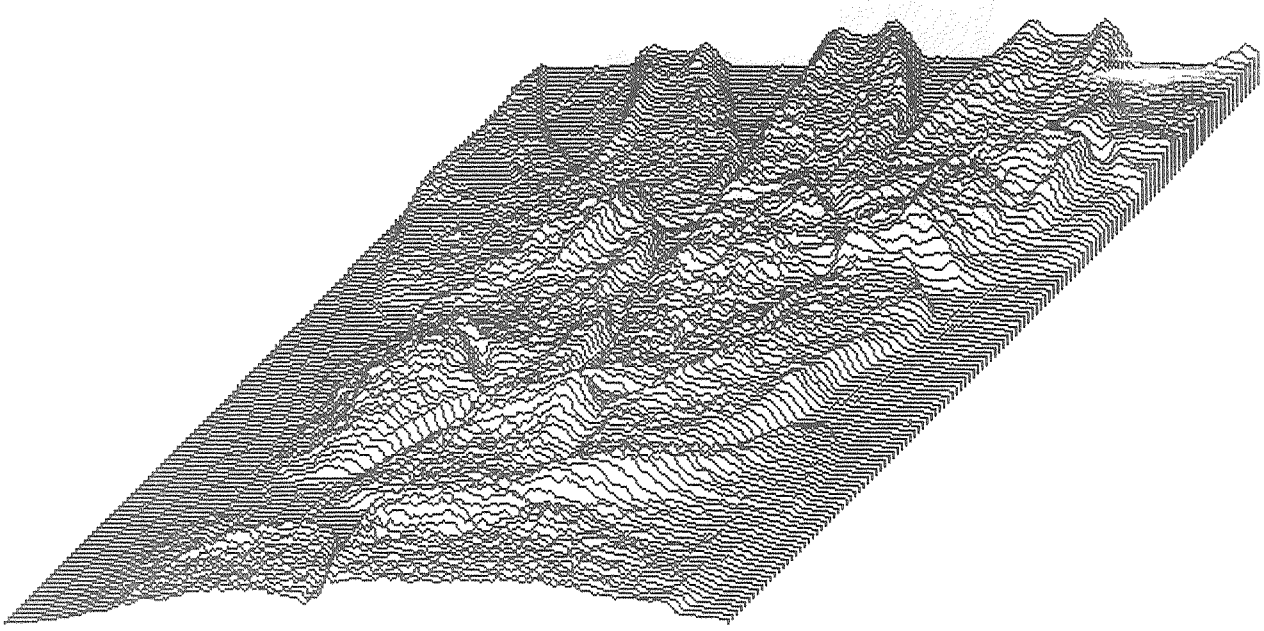


Fig. 13: Stereogramma in modo NORMALIZZATO, visione dal BASSO

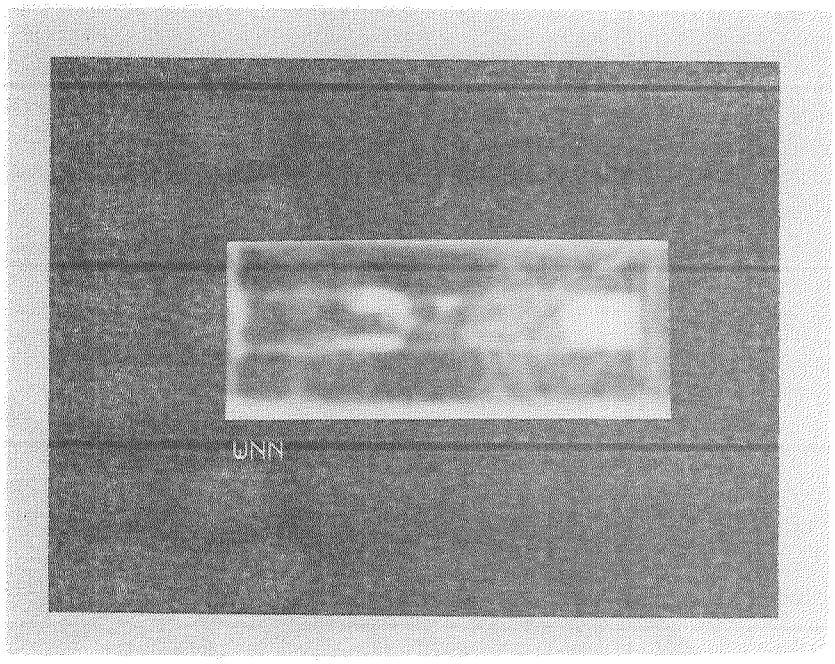


Fig. 14: Campione di materiale con difetto

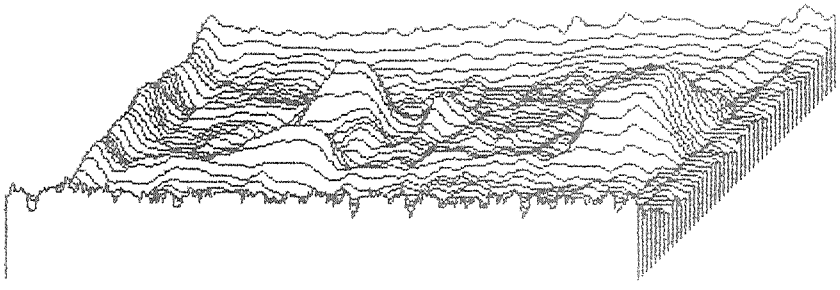


Fig. 15: Stereogramma in modo NORMALIZZATO, visione dal BASSO

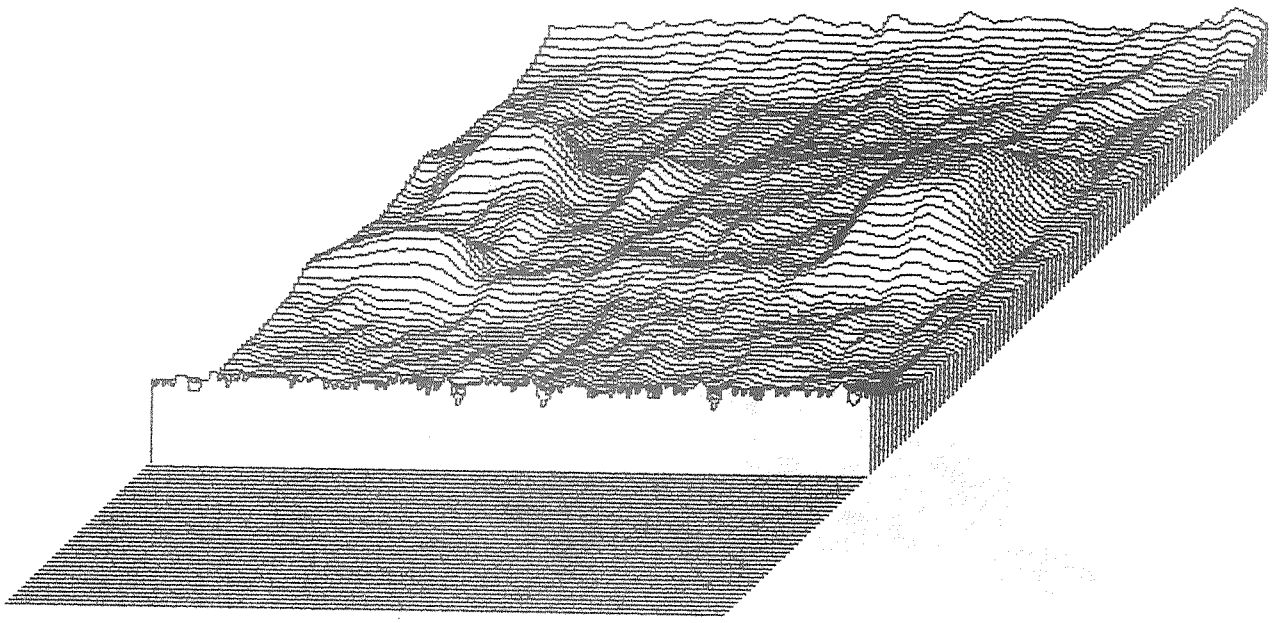


Fig. 16: Stereogramma in modo ZOOMx2, visione dal BASSO

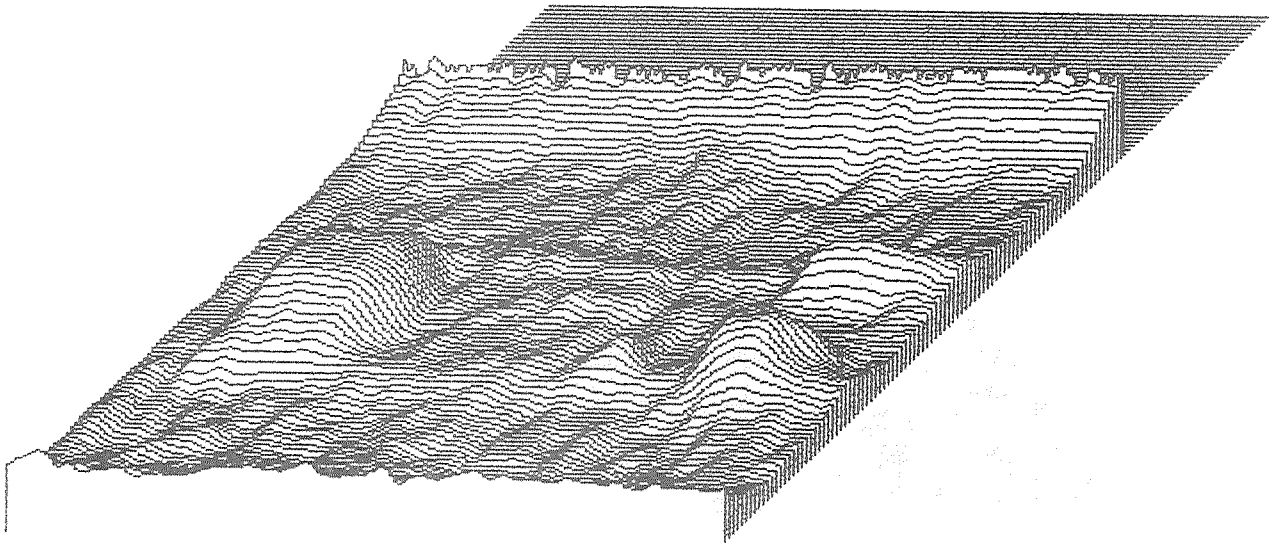


Fig. 17: Stereogramma in modo ZOOMx2, visione dall'ALTO

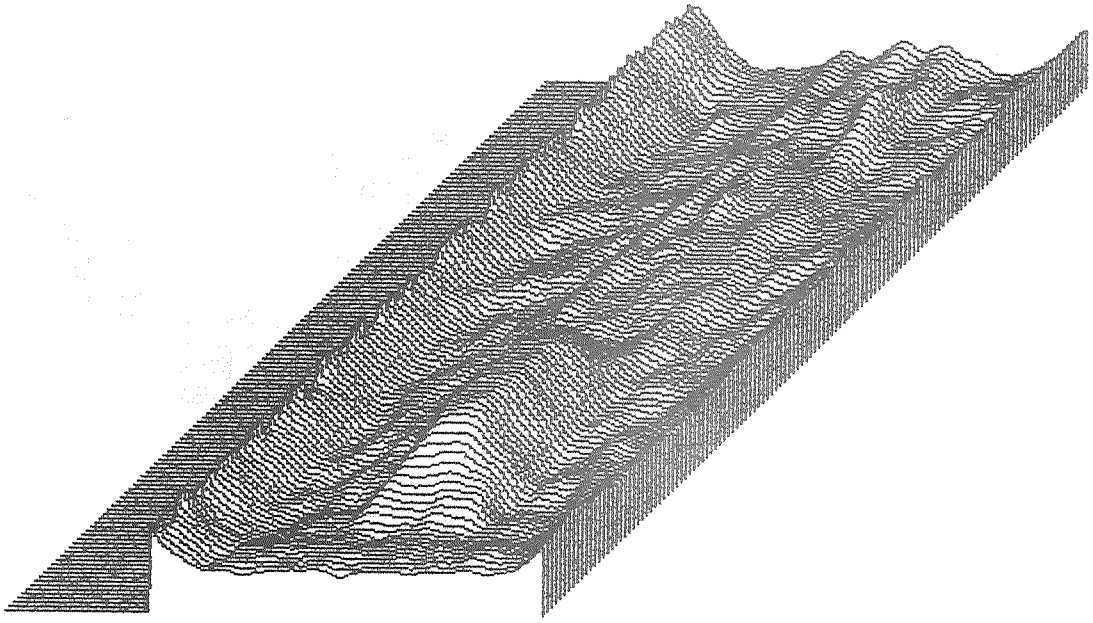


Fig. 18 Stereogramma in modo ZOOMx2, visione da DESTRA

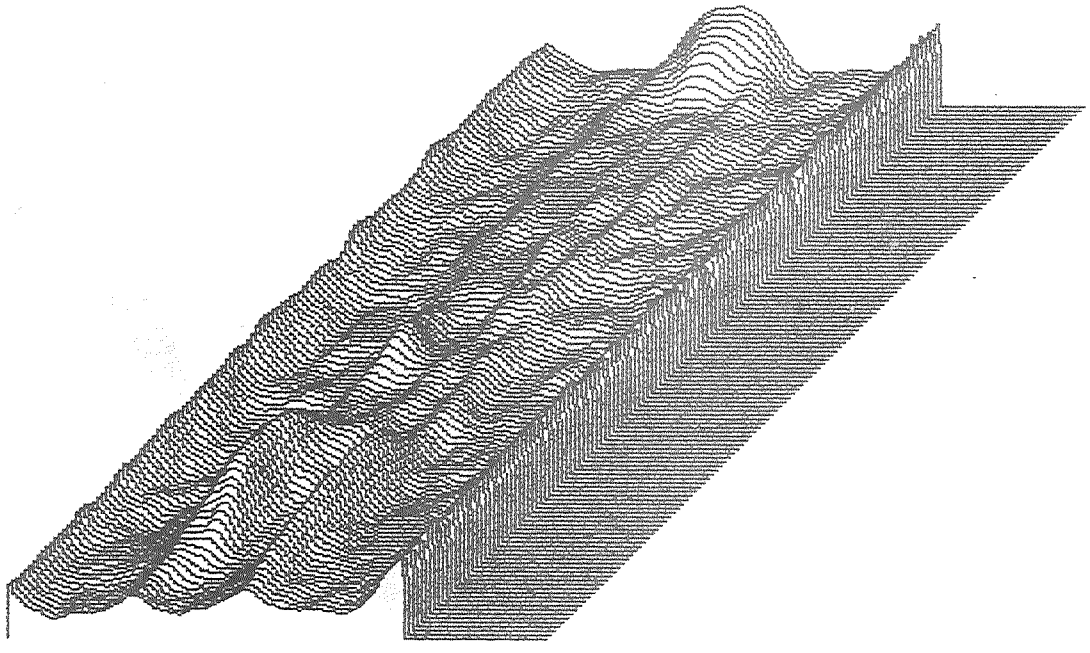


Fig. 19: Stereogramma in modo ZOOMx2, visione da SINISTRA

BIBLIOGRAFIA

[1] Bozzi R., Fantini E., Salvetti O,:
"Il sistema software BIS 386"
Brevetto CNR - rif. COF - 143.

RINGRAZIAMENTI

Si ringrazia la Sig.ra M. Ballati per la preziosa collaborazione nella stesura del presente lavoro.

Contratto AERITALIA - I.E.I. 1989

Progetto e sviluppo a cura di:

Andronico Patrizia, Guerrini Roberto, Marchetti Andrea

```
/*****************************************************************************/
/*      MAINPC.C     (A PARAMETRI COSTANTI)                                  */
/*      FUNZIONI CONTENUTE IN QUESTO MODULO:                                 */
/*               SHELL; INIT;                                                */
/*               SETMODO; SETVISIONE; SETQUADRO; SETFINESTRA;                */
/*               FINE.                                                       */
/*****************************************************************************/

#include <stdio.h>
#include <graph.h>
#include "strut.dat"
#include "video.h"

main()
{

init();

shell(); /* interprete dei comandi da tastiera */

return;

}

/*****************************************************************************/
```

```
/*****************************************************************************/
/*       INIZIALIZZAZIONE DEI PARAMETRI VIDEO (VGA E PIP) E DEI              */
/*          PARAMETRI CHE DEFINISCONO LO STEREOGRAMMA                        */
/*****************************************************************************/

init()
{


/* INIZIALIZZAZIONE VGA */
init_video();


/* INIZIALIZZAZIONE PIP */
a_init();
a_syntyp(0);                    /* ATTIVA IL SINCRONISMO INTERNO */
a_video(1);                     /* SETTA LO STANDARD VIDEO EUROPEO */
a_curtyp (0,1);                 /* DEFINISCE COLORE E FORMA DEL CURSORE */
a_setind(0);                    /* GESTIONE DEL CURSORE DA TASTIERA */


/* INIZIALIZZAZIONE PARAMETRI GENERALI */
modo = NORMALIZZATO;
visione = BASSO;
frame_buffer = 0;
XPL_INF = YPL_INF = 0;
XPL_SUP = YPL_SUP = 511;
X_CUR=255;
Y_CUR=255;


/* VISUALIZZAZIONE SU MONITOR VGA DEI VALORI INIZIALI DEI PARAMETRI */
vis_scelta_modo( modo );
vis_scelta_visione( visione );
vis_scelta_quadro( frame_buffer );
vis_scelta_finestra(  XPL_INF, 511-YPL_SUP, XPL_SUP, 511-YPL_INF  );


/* SETTAGGIO PIP IN BASE AI VALORI DEI PARAMETRI frame_buffer e finestra */
a_displ(frame_buffer);
a_invrec (XPL_INF,511-YPL_INF,XPL_SUP,511-YPL_SUP);


return;

}

/*****************************************************************************/
```

```
/*****************************************************************************/
/*          FUNZIONE PER IL SETTAGGIO DELLA VARIABILE MODO                   */
/*****************************************************************************/

short setmodo(modo_att)
short modo_att;
{

  /* CANCELLA ULTIMA FINESTRA DISEGNATA */
  a_invrec (XPL_INF,511 - YPL_INF,XPL_SUP,511 - YPL_SUP);

if (modo_att == NORMALIZZATO)
{
  modo_att = ZOOMx2;

  /* INIZIALIZZAZIONE FINESTRA PER MODO ZOOMx2 */
  XPL_SUP = 340;
  YPL_SUP = 255;
  XPL_INF = 0;
  YPL_INF = 0;
}
else
{
  modo_att = NORMALIZZATO;

  /* INIZIALIZZAZIONE FINESTRA PER MODO NORMALIZZATO */
  XPL_INF = YPL_INF = 0;
  XPL_SUP = YPL_SUP = 511;
}

/* DISEGNA NUOVA FINESTRA */
a_invrec (XPL_INF,511 - YPL_INF,XPL_SUP,511 - YPL_SUP);

vis_scelta_modo(modo_att);

vis_scelta_finestra(  XPL_INF, 511-YPL_SUP, XPL_SUP, 511-YPL_INF  );

return(modo_att);

}

/*****************************************************************************/
```

```
/*****************************************************************************/
/*          FUNZIONE PER LA REALIZZAZIONE DELLA FINESTRA SU PIP               */
/*****************************************************************************/
short setfinestra(modo_att)
short modo_att;
{
short tasto_premuto;          /* DIREZIONE CURSORE */
short direzione;              /* ULTIMA DIREZIONE SELEZIONATA */
short temp;
short passo=8;                /* N. DI PIXEL CON CUI SI SPOSTA LA FINESTRA
                                 PER OGNI PRESSIONE DI UN TASTO FRECCIA   */
short timer;

/* INIZIALIZZAZIONE VARIABILI PER IL MOVIMENTO DEL CURSORE */
tasto_premuto = NULL;
direzione = NULL;

/* CANCELLA VECCHIA FINESTRA */
a_invrec (XPL_INF,511 - YPL_INF,XPL_SUP,511 - YPL_SUP);

switch (modo_att)
{
 case NORMALIZZATO :

      vis_commento("Selezionare il 1 vertice usando le FRECCE e RETURN");

      /* DISEGNA IL CURSORE ALLE COORDINATE INDICATE */
      a_getxy (&X_CUR,&Y_CUR);

      /* DISEGNA NUOVO CURSORE */
      a_curs (X_CUR,Y_CUR);

      /* VIENE SELEZIONATO IL PRIMO PUNTO DELLA FINESTRA PIP */
      XPL_INF = X_CUR;
      YPL_INF = 511 - Y_CUR;

      vis_commento("Selezionare il 2 vertice usando le FRECCE e RETURN");

      /* VIENE SELEZIONATO IL SECONDO PUNTO DELLA FINESTRA PIP */
      a_getxy (&X_CUR,&Y_CUR);
      XPL_SUP = X_CUR;
      YPL_SUP = 511 - Y_CUR;

      /* CANCELLA VECCHIO CURSORE */
      a_curs (XPL_INF,511 - YPL_INF);

      pulisci_area_commenti();

      /* VENGONO SCAMBIATE LE COORDINATE DEI PUNTI SELEZIONATI */
      if ( XPL_INF > XPL_SUP )
      {
         temp = XPL_INF;
         XPL_INF = XPL_SUP;
         XPL_SUP = temp;
      }
```

```
      if ( YPL_INF > YPL_SUP )
      {
         temp = YPL_INF;
         YPL_INF = YPL_SUP;
         YPL_SUP = temp;
      }

      /* DISEGNA NUOVA FINESTRA */
      a_invrec (XPL_INF,511 - YPL_INF,XPL_SUP,511 - YPL_SUP);

      break;

case ZOOMx2 :
      /* INIZIALIZZAZIONE COORDINATE CURSORE */
      X_CUR=340;
      Y_CUR=256;

      /* DISEGNA NUOVO CURSORE */
      a_curs (X_CUR,Y_CUR);

      /* DISEGNA NUOVA FINESTRA */
      a_invrec (X_CUR-340,Y_CUR+255,X_CUR,Y_CUR);

      /* INIZIO CICLO PER IL MOVIMENTO DEL CURSORE */
      do
      {
         switch (tasto_premuto = getch() )
         {
            case NULL  : /* e' stato premuto un tasto funzione */

                 /* CANCELLA VECCHIA FINESTRA   */
                 a_invrec (X_CUR-340,Y_CUR+255,X_CUR,Y_CUR);

                 /* CANCELLA VECCHIO CURSORE    */
                 a_curs (X_CUR,Y_CUR);

                 direzione = NULL;

                 switch (tasto_premuto = getch())
                 {
                    case RIGHT : /* freccia destra */
                         if (X_CUR + passo <= 511)
                         {
                            X_CUR += passo;
                            direzione = RIGHT;
                         }
                         break;

                    case LEFT  : /* freccia sinistra */
                         if (X_CUR - passo >= 340)
                         {
                            X_CUR -= passo;
                            direzione = LEFT;
                         }
                         break;
```

```
            case UP    : /* freccia alto */
                 if (Y_CUR - passo >= 0)
                 {
                    Y_CUR -= passo;
                    direzione = UP;
                 }
                 break;

            case DOWN  : /* freccia basso */
                 if (Y_CUR + passo <= 256)
                 {
                    Y_CUR += passo;
                    direzione = DOWN;
                 }
                 break;

       }

       /* DISEGNA NUOVO CURSORE */
       a_curs(X_CUR,Y_CUR);

       /* DISEGNA NUOVA FINESTRA */
       a_invrec (X_CUR-340,Y_CUR+255,X_CUR,Y_CUR);
       break;

case SPACE : /* é stata premuta la barra spaziatrice */
             /* movimento libero del cursore nell'ultima */
             /* direzione selezionata */

       /* CANCELLA VECCHIA FINESTRA   */
       a_invrec (X_CUR-340,Y_CUR+255,X_CUR,Y_CUR);


       switch (direzione)
       {
          case RIGHT :
               do
               {
                  a_curs(X_CUR,Y_CUR);
                  a_curs(++X_CUR,Y_CUR);
               }
               while ( (!kbhit())&&(X_CUR < 511) );
               break;

          case LEFT  :
               do
               {
                  a_curs(X_CUR,Y_CUR);
                  a_curs(--X_CUR,Y_CUR);
               }
               while ( (!kbhit())&&(X_CUR > 340) );
               break;
```

```
                    case UP     :
                         do
                         {
                            a_curs(X_CUR,Y_CUR);
                            a_curs(X_CUR,--Y_CUR);
                         }
                         while ( (!kbhit())&&(Y_CUR > 0) );
                         break;

                    case DOWN   :
                         do
                         {
                            a_curs(X_CUR,Y_CUR);
                            a_curs(X_CUR,++Y_CUR);
                         }
                         while ( (!kbhit())&&(Y_CUR < 256) );
                         break;

                  } /* switch( direzione ) */

                  /* DISEGNA NUOVA FINESTRA */
                  a_invrec (X_CUR-340,Y_CUR+255,X_CUR,Y_CUR);
                  direzione=NULL;
                  break;

            default: break;

         } /* switch( tasto_premuto ) */

      }
      while (tasto_premuto!=ENTER);

      /* CANCELLA VECCHIO CURSORE     */
      a_curs (X_CUR,Y_CUR);

      /* COORDINATE DEI 2 PUNTI SULLA DIAGONALE DELLA FINESTRA PIP */
      XPL_SUP = X_CUR;
      YPL_SUP = 511 - Y_CUR;
      XPL_INF = XPL_SUP - 340;
      YPL_INF = YPL_SUP - 255;

      /* fine caso Zoom X 2 */
      break;


}/* switch(modo_att) */

vis_scelta_finestra(  XPL_INF, 511-YPL_SUP, XPL_SUP, 511-YPL_INF  );

return;

}

/*************************************************************************/
```

```
/*****************************************************************************
/*          FUNZIONE PER IL SETTAGGIO DELLA VARIABILE VISIONE                *
/*****************************************************************************

short setvisione(visione_att)
short visione_att;
{

/* CAMBIA LA VARIABILE VISIONE */
if (visione_att == SINISTRA)  visione_att = BASSO;
else visione_att++;

vis_scelta_visione(visione_att);

return(visione_att);

}

/*****************************************************************************
```

```
/*****************************************************************************
/*       FUNZIONE PER IL SETTAGGIO DELLA VARIABILE FRAME_BUFFER              *
/*****************************************************************************

short setquadro(frame_att)
short frame_att;
{

/* CANCELLA VECCHIA FINESTRA                                    */
a_invrec (XPL_INF,511-YPL_INF,XPL_SUP,511-YPL_SUP);

if (frame_att == 3)  frame_att = 0;
else frame_att++;

vis_scelta_quadro (frame_att);

/* SELEZIONA IL QUADRO DELLA MEMORIA IMMAGINE DA VISUALIZZARE */
a_displ (frame_att);

/* DISEGNA NUOVA FINESTRA                                       */
a_invrec (XPL_INF,511-YPL_INF,XPL_SUP,511-YPL_SUP);

return(frame_att);

}

/*****************************************************************************
```

```
/*****************************************************************************/
/*           FUNZIONE SHELL                                                  */
/*****************************************************************************/

shell()
{
char c;

do
{
  switch (c=getch())
  {

  /* SET PARAMETRI */

  case '1' :
             modo = setmodo(modo);
             break;

  case '2' :
             setfinestra(modo);
             break;

  case '3' :
             visione = setvisione(visione);
             break;

  case '4' :
             frame_buffer = setquadro(frame_buffer);
             break;


  /* COMANDI */

  case INS :
              stereogramma(modo, visione);
              break;

  case ESC :
              fine();
              break;


  default  :
              break;

  }
}
while (1);

return;

}
/*****************************************************************************/
```

```
/*****************************************************************************/
/*           FUNZIONE FINE                                                   */
/*****************************************************************************/

fine()
{
   char scelta;

   vis_commento( "Sei proprio sicuro (s/n)? " );

   do
    cscanf("%c",&scelta);
   while ( (scelta!='s')&&(scelta!='S')&&(scelta!='n')&&(scelta!='N') );

   if ( (scelta=='s')||(scelta=='S') )
   {
      /* CANCELLA ULTIMA FINESTRA DISEGNATA */
      a_invrec (XPL_INF,511 - YPL_INF,XPL_SUP,511 - YPL_SUP);
      reset_video( TESTO );
      exit(0);
   }

   pulisci_area_commenti();


   return;

}

/*****************************************************************************/
```

```
/*****************************************************************************/
/*              STEREOPC.C                                                   */
/*                  FUNZIONE CHE COSTRUISCE LO STEREOGRAMMA                  */
/*****************************************************************************/

#include<stdio.h>
#include"video.h"

#define NORMALIZZATO 0              /* MODI DI VISUALIZZAZIONE */
#define ZOOMx2       1

#define BASSO    0                  /* VISIONI */
#define ALTO     1
#define DESTRA   2
#define SINISTRA 3

#define AMP 0.2                     /* FATTORE DI AMPLIFICAZIONE PER I VALORI */
                                    /* DELLA MATRICE IMMAGINE */
#define DIMXF 597                   /* DIMENSIONE X DELLA FINESTRA VGA PER LO */
                                    /* STEREOGRAMMA */
#define VISIBILE 1
#define INVISIBILE 0

extern short X_CUR,Y_CUR;           /* COORDINATE CURSORE */
extern short XPL_INF, YPL_INF;      /* COORDINATE LOGICHE PIP BASSO-SINISTRA */
extern short XPL_SUP, YPL_SUP;      /* COORDINATE LOGICHE PIP ALTO-DESTRA */


stereogramma(modo_att, visione_att)
short modo_att;
short visione_att;
{
short xpl, ypl;             /* CONTATORI RIGA E COLONNA PIP */
short xor, yor;             /* COORDINATE ORIGINE VGA DI UNA RIGA SULL'ASSE Z */
short xvl, yvl;             /* COORDINATE LOGICHE VGA */
short temp;                 /* PER INIZIALIZZARE LA MASCHERA[] */
char  punto_prec;           /* PUNTO VISIBILE */
short unsigned maschera[DIMXF];/* ARRAY PUNTI MAX */
char unsigned buffer[512]; /* ARRAY PER LA LETTURA DELLA MATRICE IMMAGINE */
short scansione_r_c;        /* PASSO DI SCANSIONE NELLA LETTURA DELL'IMMAGINE */
                            /* DIPENDENTE DAL MODO */
short passo_vis;            /* PASSO DI VISUALIZZAZIONE DELLE RIGHE DELLO */
                            /* STEREOGRAMMA */

/* INIZIALIZZAZIONE MASCHERA */
for (temp=0; temp<DIMXF; temp++)
  maschera[temp] = 0;

pulisci_area_disegno();

set_colore(RED);

/* INIZIALIZZAZIONE PASSO_VIS */
passo_vis = 2;
```

```
/* INIZIALIZZAZIONE SCANSIONE_R_C DIPENDENTE DALLA VARIABILE MODO */
if (modo_att == NORMALIZZATO) scansione_r_c = 2 * passo_vis;
else scansione_r_c = passo_vis;

 switch(visione_att)
 {

   case BASSO   :
                  /* SI SCANDISCONO LE RIGHE DELLA FINESTRA CORRENTE */
                  for(ypl = YPL_INF+1, yor=0; ypl <= YPL_SUP-1;
                              yor+=passo_vis,ypl+=scansione_r_c)
                  {
                   /* SI MEMORIZZA UNA RIGA DELLA FINESTRA */
                   a_rowr(XPL_INF+1, 511-ypl, XPL_SUP-XPL_INF-1, buffer);

                   /* GESTIONE PUNTO ORIGINE DELLA RIGA */
                   xor = yor;
                   if (yor >= maschera[xor])  moveto(xor,yor);
                   else  moveto(xor, maschera[xor]);

                   punto_prec = VISIBILE;

                   /* SCANSIONE E VISUALIZZAZIONE DEI PUNTI DELLA RIGA */
                   xpl = 0;  /* con xpl scandisco gli elementi del buffer */
                   xvl = xor;
                   do
                   {
                      /* CALCOLO ORDINATA DEL PUNTO XPL DELLA RIGA */
                      yvl = buffer[xpl]*AMP + yor;

                      /* CONTROLLO VISIBILITA' DEL PUNTO IN ESAME */
                      if (yvl >= maschera[xvl])
                      {
                         /* PUNTO IN ESAME VISIBILE */
                         if (punto_prec == INVISIBILE)
                         {
                            moveto( xvl-1, maschera[xvl-1] );
                            punto_prec = VISIBILE;
                         }
                         maschera[xvl] = yvl;
                         lineto(xvl, yvl);
                      }
                      else
                         /* PUNTO IN ESAME NON VISIBILE */
                         punto_prec = INVISIBILE;

                    switch (modo_att)
                     {
                      /* SI VISUALIZZANO TUTTI I PUNTI DI UNA RIGA */
                      case ZOOMx2:
                                 xpl+=1;
                                 break;
```

```
                /* SI VISUALIZZANO, DI UNA RIGA, 2 PUNTI SU 3 */
                case NORMALIZZATO:
                          if ((xpl+2)%3 == 0)  xpl+=2;
                          else  xpl+=1;
                          break;
                }

                xvl+=1;
               }
               while (xpl < (XPL_SUP-XPL_INF-1));

             /* SI CHIUDE LA RIGA IN ESAME */
             lineto(xvl-1, yor);

           }   /* for sulle righe delle finestre */

           break;

case ALTO    :
           /* SI SCANDISCONO LE RIGHE DELLA FINESTRA CORRENTE */
           for(ypl = YPL_SUP-1, yor=0; ypl >= YPL_INF+1;
                     yor+=passo_vis, ypl-=scansione_r_c)
           {
            /* SI MEMORIZZA UNA RIGA DELLA FINESTRA */
            a_rowr(XPL_INF+1, 511-ypl, XPL_SUP-XPL_INF-1, buffer);

            /* GESTIONE PUNTO ORIGINE DELLA RIGA */
            xor = yor;
            if (yor >= maschera[xor])  moveto(xor,yor);
            else  moveto(xor, maschera[xor]);

            punto_prec = VISIBILE;

            /* SCANSIONE E VISUALIZZAZIONE DEI PUNTI DELLA RIGA */
            xpl = 1;  /* con xpl scandisco gli elementi del buffer */
            xvl = xor;
            do
            {
               /* CALCOLO ORDINATA DEL PUNTO XPL DELLA RIGA */
               yvl = buffer[XPL_SUP-XPL_INF-1-xpl]*AMP + yor;

               /* CONTROLLO VISIBILITA' DEL PUNTO IN ESAME */
               if (yvl >= maschera[xvl])
               {
                  /* PUNTO IN ESAME VISIBILE */
                  if (punto_prec == INVISIBILE)
                  {
                     moveto(xvl-1, maschera[xvl-1]);
                     punto_prec = VISIBILE;
                  }
                  maschera[xvl] = yvl;
                  lineto(xvl, yvl);
               }
```

```
                    else

                      /* PUNTO IN ESAME NON VISIBILE */
                       punto_prec = INVISIBILE;

                    switch (modo_att)
                     {
                      /* SI VISUALIZZANO TUTTI I PUNTI DI UNA RIGA */
                      case ZOOMx2:
                                 xpl+=1;
                                 break;

                      /* SI VISUALIZZANO, DI UNA RIGA, 2 PUNTI SU 3 */
                      case NORMALIZZATO:
                                 if ((xpl+2)%3 == 0)  xpl+=2;
                                 else  xpl+=1;
                                 break;
                     }

                     xvl+=1;
                    }
                    while (xpl <= (XPL_SUP-XPL_INF-1));

                  /* SI CHIUDE LA RIGA IN ESAME */
                  lineto(xvl-1, yor);

                }   /* for sulle righe della finestra */

                break;

case DESTRA   :
                /* SI SCANDISCONO LE COLONNE DELLA FINESTRA CORRENTE */
                for(xpl = XPL_SUP-1, yor=0;
                    xpl >= (modo_att==ZOOMx2 ? XPL_SUP-254 : XPL_INF+1);
                    yor+=passo_vis,xpl-=scansione_r_c)
                {
                 /* SI MEMORIZZA UNA COLONNA DELLA FINESTRA */
                 a_colr(xpl, 511-YPL_SUP+1, YPL_SUP-YPL_INF-1, buffer);

                 /* GESTIONE PUNTO ORIGINE DELLA RIGA */
                 xor = yor;
                 if (yor >= maschera[xor])  moveto(xor,yor);
                 else  moveto(xor, maschera[xor]);

                 punto_prec = VISIBILE;

                 /* SCANSIONE E VISUALIZZAZIONE DEI PUNTI DELLA COLONNA */
                 ypl = 1;  /* con ypl scandisco gli elementi del buffer */
                 xvl = xor;
                 do
                 {
                    /* CALCOLO ORDINATA DEL PUNTO YPL DELLA COLONNA */
                    yvl = buffer[YPL_SUP-YPL_INF-1-ypl]*AMP + yor;
```

```
                /* CONTROLLO VISIBILITA' DEL PUNTO IN ESAME */
                if (yvl >= maschera[xvl])
                {
                   /* PUNTO IN ESAME VISIBILE */
                   if (punto_prec == INVISIBILE)
                   {
                      moveto(xvl-1, maschera[xvl-1]);
                      punto_prec = VISIBILE;
                   }
                   maschera[xvl] = yvl;
                   lineto(xvl, yvl);
                }
                else
                   /* PUNTO IN ESAME NON VISIBILE */
                   punto_prec = INVISIBILE;

                switch (modo_att)
                {
                 /* SI VISUALIZZANO TUTTI I PUNTI DI UNA COLONNA */
                 case ZOOMx2:
                            ypl+=1;
                            break;

                 /* SI VISUALIZZANO, DI UNA COLONNA, 2 PUNTI SU 3 */
                 case NORMALIZZATO:
                            if ((ypl+2)%3 == 0)  ypl+=2;
                            else  ypl+=1;
                            break;
                }

                 xvl+=1;
                }
                while (ypl <= (YPL_SUP-YPL_INF-1));

                /* SI CHIUDE LA COLONNA IN ESAME */
                lineto(xvl-1, yor);

               }   /* for sulle colonne della finestra */

               break;

case SINISTRA:
               /* SI SCANDISCONO LE COLONNE DELLA FINESTRA CORRENTE */
               for(xpl = XPL_INF+1, yor=0;
                   xpl <= (modo_att==ZOOMx2 ? XPL_INF+254 : XPL_SUP-1);
                   yor+=passo_vis,xpl+=scansione_r_c)
               {
                /* SI MEMORIZZA UNA COLONNA DELLA FINESTRA */
                a_colr(xpl, 511-YPL_SUP+1, YPL_SUP-YPL_INF-1, buffer);

                /* GESTIONE PUNTO ORIGINE DELLA RIGA */
                xor = yor;
                if (yor >= maschera[xor])  moveto(xor,yor);
                else  moveto(xor, maschera[xor]);
                punto_prec = VISIBILE;
```

```
                    /* SCANSIONE E VISUALIZZAZIONE DEI PUNTI DELLA COLONNA */
                    ypl = 0;  /* con ypl scandisco gli elementi del buffer */
                    xvl = xor;
                    do
                    {
                       /* CALCOLO ORDINATA DEL PUNTO YPL DELLA COLONNA */
                       yvl = buffer[ypl]*AMP + yor;

                       /* CONTROLLO VISIBILITA' DEL PUNTO IN ESAME */
                       if (yvl >= maschera[xvl])
                       {
                          /* PUNTO IN ESAME VISIBILE */
                          if (punto_prec == INVISIBILE)
                          {
                             moveto(xvl-1, maschera[xvl-1]);
                             punto_prec = VISIBILE;
                          }
                          maschera[xvl] = yvl;
                          lineto(xvl, yvl);
                       }
                       else
                          /* PUNTO IN ESAME NON VISIBILE */
                          punto_prec = INVISIBILE;

                       switch (modo_att)
                       {
                        /* SI VISUALIZZANO TUTTI I PUNTI DI UNA COLONNA */
                        case ZOOMx2:
                                    ypl+=1;
                                    break;

                        /* SI VISUALIZZANO, DI UNA COLONNA, 2 PUNTI SU 3 */
                        case NORMALIZZATO:
                                    if ((ypl+2)%3 == 0)  ypl+=2;
                                    else  ypl+=1;
                                    break;
                       }

                     xvl+=1;
                    }
                    while (ypl < (YPL_SUP-YPL_INF-1));

                    /* SI CHIUDE LA COLONNA IN ESAME */
                    lineto(xvl-1, yor);

                  }   /* for sulle colonne della finestra */

                  break;
  }

}

/*****************************************************************************
```

```
/*****************************************************************************/
/*       VIDEO.C                                                             */
/*       FUNZIONI CONTENUTE IN QUESTO MODULO:                                */
/*                INIT_VIDEO; RESET_VIDEO;                                   */
/*                VIS_SCELTA_MODO; VIS_SCELTA_FINESTRA;                      */
/*                VIS_SCELTA_VISIONE; VIS_SCELTA_QUADRO; VIS_COMMENTI;       */
/*                PULISCI_AREA_DISEGNO; PULISCI_AREA_COMMENTI.               */
/*****************************************************************************/

#include<graph.h>
#include<stdio.h>
#include<stdlib.h>          /* per la funzione abs() */
#include"video.dat"



/*****************************************************************************/
/*                FUNZIONE DI INIZIALIZZAZIONE DEL VIDEO                     */
/*****************************************************************************/

init_video()
{
short y_up, y_down, x_right, x_left;


/* SETTAGGIO DEL MODO GRAFICO */
/* Modo grafico attuale disponendo di una scheda grafica VGA                 *
/* note tecniche :    risoluzione in pixel 640 x 480  (0-639 , 0-479)        *
/*                         "      in char.  80 x  30  (1-80  , 1-30)        *
/*                    dim. char.  in pixel  8 x 16                           *
/*                    numero di colori = 16                                  *
reset_video( GRAFICO );
_getvideoconfig( &conf );

/* altezza in pixel di un carattere */
h_p_char = conf.numypixels / conf.numtextrows;

/* larghezza in pixel di un carattere */
l_p_char = conf.numxpixels / conf.numtextcols;
```

```
/****************** SUDDIVISIONE  VIDEO    IN AREE ****************************
/* Area comandi */
vga.comandi.colore_tenue  = WHITE;      /* colori area comandi   */
vga.comandi.colore_forte  = BRIGHT;
vga.comandi.colore_sfondo = BLACK;
vga.comandi.riga_up       = ROW(1);    /* estremi area comandi  */
vga.comandi.riga_down     = ROW(1);
vga.comandi.col_left      = COL(1);
vga.comandi.col_right     = conf.numtextcols;

/* Area disegno */
vga.disegno.colore_sfondo = BRIGHT;       /* colori area disegno   */
vga.disegno.colore_bordo  = BLUE;
vga.disegno.riga_up       = ROW( 2);     /* estremi area disegno  */
vga.disegno.riga_down     = ROW(21);
vga.disegno.col_left      = COL( 1);
vga.disegno.col_right     = conf.numtextcols;

/* Area commenti */
vga.commenti.colore_tenue  = WHITE;      /* colori area commenti  */
vga.commenti.colore_forte  = BRIGHT;
vga.commenti.colore_sfondo = BLACK;
vga.commenti.riga_up       = ROW(22);    /* estremi area commenti */
vga.commenti.riga_down     = ROW(30);
vga.commenti.col_left      = COL( 1);
vga.commenti.col_right     = conf.numtextcols;
```

```
/***** REALIZZAZIONE DELLA SCHERMATA VIDEO COME DESCRITTA NEL DISEGNO
       SOTTOSTANTE    ******/


/*
                                   AREA COMANDI
    (1)  (6)                                                  (59) (64)
 1   INS  per costruire lo stereogramma                        ESC  per uscire
   ____________________________________________________________________________
 2 :                                                                          :
 3 :                                                                          :
 4 :                                                                          :
 5 :                                                                          :
 6 :                                                                          :
 7 :                                                                          :
 8 :                                                                          :
 9 :                                                                          :
10 :                                                                          :
11 :                              AREA DISEGNO                                :
12 :                                                                          :
13 :                                                                          :
14 :                                                                          :
15 :                                                                          :
16 :                                                                          :
17 :                                                                          :
18 :                                                                          :
19 :                                                                          :
21 :                                                                          :
20 :                                                                          :
21 :                                                                          :
   ----------------------------------------------------------------------------
22  (4)                  (21) (24)                        (45)
23   1.  MODO             :    .....
24   2.  FINESTRA         :    Origine                     Dimensioni
25   3.  VISIONI          :    .....
26   4.  FRAME-BUFFER     :    .....
27                                          AREA COMMENTI
28   Commenti occasionali
29
30


*/
```

```
/****** Trasformazione in pixels degli estremi dell' area comandi *******/
y_up    =  (vga.comandi.riga_up - 1)  * h_p_char    ;
y_down  =   vga.comandi.riga_down     * h_p_char - 1;
x_left  =  (vga.comandi.col_left - 1) * l_p_char    ;
x_right =   vga.comandi.col_right     * l_p_char - 1;

_setcolor( vga.comandi.colore_sfondo );
_rectangle(_GFILLINTERIOR, x_left, y_up, x_right, y_down);

_settextcolor( vga.comandi.colore_forte );
_settextposition( vga.comandi.riga_up , vga.comandi.col_left + 4 );
_outtext("INS");
_settextposition( vga.comandi.riga_up , vga.comandi.col_left + 58 );
_outtext("ESC");

_settextcolor( vga.comandi.colore_tenue );
_settextposition( vga.comandi.riga_up , vga.comandi.col_left + 9 );
_outtext("Per costruire lo stereogramma");
_settextposition( vga.comandi.riga_up , vga.comandi.col_left + 63 );
_outtext("Per uscire");

/****** Trasformazione in pixels degli estremi dell' area commenti ******/
y_up    =  (vga.commenti.riga_up - 1) * h_p_char     ;
y_down  =   vga.commenti.riga_down    * h_p_char - 1 ;
x_left  =  (vga.commenti.col_left - 1) * l_p_char    ;
x_right =   vga.commenti.col_right     * l_p_char - 1;

_setcolor( vga.commenti.colore_sfondo );
_rectangle(_GFILLINTERIOR,  x_left, y_up, x_right, y_down);

_settextcolor( vga.commenti.colore_tenue );
_settextposition( vga.commenti.riga_up + 1, vga.commenti.col_left + 3 );
_outtext("1.  MODO          :");
_settextposition( vga.commenti.riga_up + 2, vga.commenti.col_left + 3 );
_outtext("2.  FINESTRA      :");
_settextposition( vga.commenti.riga_up + 2, vga.commenti.col_left +23 );
_outtext("Origine");
_settextposition( vga.commenti.riga_up + 2, vga.commenti.col_left +44 );
_outtext("Dimensioni");
_settextposition( vga.commenti.riga_up + 3, vga.commenti.col_left + 3 );
_outtext("3.  VISIONI       :");
_settextposition( vga.commenti.riga_up + 4, vga.commenti.col_left + 3 );
_outtext("4.  FRAME-BUFFER :");

/****** Trasformazione in pixels degli estremi dell' area disegno  ******/
y_up    =  (vga.disegno.riga_up - 1) * h_p_char +     GAP;
y_down  =   vga.disegno.riga_down    * h_p_char - 1 - GAP;
x_left  =  (vga.disegno.col_left - 1) * l_p_char    ;
x_right =   vga.disegno.col_right     * l_p_char - 1;

_setcolor( vga.disegno.colore_sfondo );
_rectangle(_GFILLINTERIOR,  x_left, y_up, x_right, y_down);
_setcolor( vga.disegno.colore_bordo  );
_rectangle(_GBORDER,        x_left, y_up, x_right, y_down);
_rectangle(_GBORDER,        x_left + 3, y_up + 3, x_right - 3, y_down - 3);
```

```
/* Inizializzazione di alcune strutture dati della  libreria grafica */

/*************************  ATTENZIONE !!!!!!  ******************************/

_setcliprgn(x_left + 4, y_up + 4, x_right - 4, y_down - 4); /* SET FINESTRA */
_setlogorg( x_left + 4, y_down - 4 );                       /* SET ORIGINE  */

/*************************  ATTENZIONE !!!!!!  ******************************/

/* Queste due ultime chiamate alterano drasticamente le strutture dati di
   tutte le funzioni grafiche in quanto
   a) Set-Clipping-Region : stabilisce un'area di visualizzazione contro cui
      tutte le figure geometriche prodotte dalle funzioni _arc, _ellipse,
      _lineto, _pie, _rectanle, _setpixel, saranno ritagliate (clipping).

   b) Set-Logical-Origin  : sposta l'origine logica nell'angolo in basso a
      sinistra della finestra di clipping, influenzando tutte le funzioni
      che usano le coordinate in pixel, che sono, oltre a quelle specificate
      sopra, _floodfill, _getcurrentposition, _getpixel, _moveto            */

/*************************  ATTENZIONE !!!!!!  ******************************/

_wrapon(_GWRAPOFF); /* tronca il testo che esce dalla text_window impedendo
                       lo scrolling verticale */
_settextcolor(vga.commenti.colore_forte);/* si fissa il colore per il testo */

return;

}

/*****************************************************************************/
```

```
/*****************************************************************************/
/*                  FUNZIONE CHE SELEZIONA LA CONFIGURAZIONE DEL VIDEO       */
/*****************************************************************************/

reset_video( modo )
short modo;
{

  switch( modo )
  {
     case GRAFICO : _setvideomode(_VRES16COLOR);
                    break;

     case TESTO   : _setvideomode(_MRES256COLOR);
                    _setvideomode(_TEXTC80);
                    break;
   }

  return;

}

/*****************************************************************************/
```

```
/*****************************************************************************/
/*             VISUALIZZAZIONE SCELTA MODO SU MONITOR VGA                    */
/*****************************************************************************/

vis_scelta_modo(num_modo)
short num_modo;
{

_settextposition(vga.commenti.riga_up + 1, vga.commenti.col_left + 23);
_settextcolor(vga.commenti.colore_forte);
_outtext(modi[num_modo]);

return;

}

/*****************************************************************************/
```

```
/*****************************************************************************/
/*            VISUALIZZAZIONE SCELTA FINESTRA SU MONITOR VGA                 */
/*****************************************************************************/

vis_scelta_finestra(X_INF,Y_INF,X_SUP,Y_SUP)
short X_INF,Y_INF,X_SUP,Y_SUP;
{
char origine[8];            /* ARRAY VISUALIZZAZIONE PUNTO ORIGINE */
char dimensioni[8];       /* ARRAY VISUALIZZAZIONE DIMENSIONI FINESTRA PIP */
short temp;

_settextcolor(vga.commenti.colore_forte); /* fissa il colore del testo */

/* VISUALIZZA LE COORDINATE DELL'ORIGINE LOGICA DELLA FINESTRA PIP */
temp = sprintf(origine,"%d%c%d",X_INF,',',Y_INF);
while(temp<7)   origine[temp++]=' '; origine[7] = NULL;

_settextposition(vga.commenti.riga_up + 2, vga.commenti.col_left + 33);
_outtext(origine);


/* VISUALIZZA LE DIMENSIONI DELLA FINESTRA PIP */
temp = sprintf(dimensioni,"%d%c%d",abs(X_SUP-X_INF)+1,'x',abs(Y_SUP-Y_INF)+1);
while(temp<7)   dimensioni[temp++]=' '; dimensioni[7] = NULL;

_settextposition(vga.commenti.riga_up + 2, vga.commenti.col_left + 57);
_outtext(dimensioni);

return;

}

/*****************************************************************************/




/*****************************************************************************/
/*            VISUALIZZAZIONE SCELTA VISIONE SU MONITOR VGA                  */
/*****************************************************************************/

vis_scelta_visione(visione_VGA)
short visione_VGA;
{

_settextcolor(vga.commenti.colore_forte);
_settextposition(vga.commenti.riga_up + 3, vga.commenti.col_left + 23);
_outtext(visioni[visione_VGA]);

return;

}

/*****************************************************************************/
```

```
/*****************************************************************************
/*          VISUALIZZAZIONE SCELTA FRAME_BUFFER SU MONITOR VGA                *
/*****************************************************************************

vis_scelta_quadro(frame_VGA)
short frame_VGA;
{

_settextcolor(vga.commenti.colore_forte);
_settextposition(vga.commenti.riga_up + 4, vga.commenti.col_left + 23);
_outtext(buffer[frame_VGA]);

return;

}

/*****************************************************************************
/*****************************************************************************
/*            FUNZIONE PER LA VISUALIZZAZIONE DEI COMMENTI                    *
/*****************************************************************************

vis_commento(comm)
char *comm;
{
   _settextposition(vga.commenti.riga_up + 6, vga.commenti.col_left + 3);
   _settextcolor(vga.commenti.colore_forte);
   _outtext( comm );

   return;
}

/*****************************************************************************
```

```
/*****************************************************************************/
/*    FUNZIONE CHE PULISCE LA FINESTRA CONTENENTE LO STEREOGRAMMA            */
/*****************************************************************************/

pulisci_area_disegno()
{
short x,y;

/* SI PULISCE L'AREA DESTINATA A CONTENERE IL GRAFICO */
_setcolor(vga.disegno.colore_sfondo);

y = (vga.disegno.riga_up-vga.disegno.riga_down-1) * h_p_char + 2*(GAP+4);
x = (vga.disegno.col_right-vga.disegno.col_left+1) * l_p_char - 2*4;

_rectangle(_GFILLINTERIOR,  0, y, x, 0); /* attualmente 0,-306,632,0 */
/*** Nota nell'invocare la funzione RECTANGLE si e' tenuto conto che e' stata
     spostata l'origine logica                                              ***/
return;

}

/*****************************************************************************/




/*****************************************************************************/
/*                    FUNZIONE CHE PULISCE L'AREA COMMENTI                   */
/*****************************************************************************/

pulisci_area_commenti()
{
 _settextcolor(vga.commenti.colore_sfondo);
 _settextposition(vga.commenti.riga_up + 6, vga.commenti.col_left);
 _outtext( "                                                                " );

return;

}

/*****************************************************************************/
```