Turbo-Cによる画像処理プログラムについて

吉村　卓

On Programming of Image Processing in Turbo-C

Takashi YOSHIMURA

(昭和63年10月17日受理)

1. はしがき

さきに，BASIC語と機械語を用いてパソコン用画像処理プログラムを作成した。プログラミングにおいて，機械語を用いたのは，BASICだけでは処理速度が遅く実用にならないからである。初期設定やプログラム全体の流れをコントロールする部分など，繰り返し計算を伴わない部分をBASICで，メニューで選ばれた個々の処理や繰り返し部分を機械語でプログラミングすることにより，処理時間の短縮化が図られた。

ところで，機械語によるプログラミングはソフトウエア作成作業の面からは非能率的であり，一方，BASIC語は構造化プログラミングには不向きである。そこで，今回はC言語によるプログラミングに切換えたわけである。

Cには，構造化プログラミングを行う上で必要な流れ制御構造として，if else構文はもちろんのこと，判定先行型ループにwhile文，判定後行型ループにdo・・・while構文，回数判定型ループにfor文と豊富な構文が用意されている。殊に，Cにはブロック（複文）なる概念があるので，if elseやwhileの制御構造をスマートに書くことができる。BASICやFORTRANはブロックという概念をもたないために構造化プログラミングに向かないのである。

2. 機能概要

今回は，NECのPC-9801VXを用いて，フーリエ変換とフィルタリングのアルゴリズムを用意することを主目的としているが，それに先だち最小限，つぎの機能を準備する必要があるであろう。

(1) 画像データの保存（save）と再生（load）
(2) 濃度ヒストグラム（histgram）
(3) 濃度パターン表示（dot_pattern）とハードコピー（gcopy）

フーリエ変換により画像を空間領域から周波数領域に変換してスペクトルを求め，周波数領域でフィルタ操作を行なってから逆変換により元空間に復元する。アルゴリズムとして基数2の周波数間引方式を用い，1次元FFTの繰り返しにより2次元データの変換処理を行なう。

3. データ構造

3・1 メモリモデル

メモリ・モデルとしてスモール・モデルを用いる。Turbo-Cではスモール・モデルの場合，プログラム・サイズ，データ・サイズともにその上限は64KBである。

1枚の画面を表現するのに128＊128画素（unsigned char型2次元配列imageに格納）とし，各画素は1バイト（256段階）の濃度値をもつものとする。

3・2 ラム・ディスクの使用

1画面128＊128画素でも，フーリエ変換を行なうには，フーリエ変換像の実部ならびに虚部を格納するために，2個のfloat型2次元配列realとimagを用意せねばならないから，2バンク128KBのメモリ領域が必要となり，スモール・モデルでは主メモリだけでは収容しきれない。そのため，アルゴリズムとしてX方向ならびにY方向の1次元FFTの繰り返し法を採用する。すなわち2次元配列データをディスクに格納しておき，処理の対象となる1行あるいは1列分のデータをその都度ディスクから読み込んできて変換処理を施し再びディスクに戻してやるという方法によるわけである。

しかしこの方法では，1画面の変換処理に128＊2回，ディスクとのデータの出し入れを必要とし，フロッピーディスクを用いていたのでは，入出力の

ための時間がかかりすぎて実用にならない。そのために増設RAMボードをラム・ディスクとして用いる。

4. プログラミングの手法

4・1　プロセス制御

親プロセスとしてのメニュー・プログラムで選択された個々の処理を子プロセスとして実行し，処理終了後は再び親プロセスとしてのメニュー・プログラムに制御を戻してやる。そのために，Turbo－Cに標準関数として用意されているspawn関数を用いる。

4・2　各種ライブラリ・ルーチン

Turbo－CのVer 1・5ではＩＢＭ版にテキスト画面処理関数とグラフィック処理関数が用意されているが，PC版には未だこれらの関数が提供されていないので，テキスト画面制御やグラフィック処理を実行するためにはそれに必要な関数を自作しなければならない。

(1)　CRTインターフェイス

Turbo－Cの標準関数には，CRTの任意の位置にカーソルを移動させる等の関数が用意されていない。ところで，Turbo－CはMS－DOSの上で走っている。カーソル・アドレッシングのようなハードウエアに依存する部分はOSの機能を直接コールすることで実現する。

すなわち，エスケープ・コード0×1bではじまる文字列を送出することによりCRT画面の各種制御を行う。最小限つぎのような関数を用意する。

console	画面最下行の使用モード設定
clstxt	テキスト画面のクリア
locate	カーソル位置の指定
clreol	カーソル位置から行末までの消去

```
void locate(x,y)
int x,y;
{
if (x<0 || x>80 || y<0 || y>25)
  return;
printf("¥x1b[");
putchar(y/10+'0');
putchar(y%10+'0');
putchar(';');
putchar(x/10+'0');
putchar(x%10+'0');
putchar('H');
}
```

図1

(2)　グラフィック・インターフェイス

PC－9800でグラフィック処理を行うにはソフトウエア割り込みによりROM内ルーチンを利用することができる。すなわち，グラフLIOのコマンドを実行するには，パラメータ・リストの先頭アドレスのセグメント値をレジスタdsに，オフセット値をレジスタbxに与えてint nを行う。そのめにはTurbo－Cの関数のfar型システム・コールint86×を用いればよい。以下のような関数を用意する。

ginit	グラフィックLIOの初期化
screen	グラフィック画面モード設定
cls	画面クリア
pset	指定位置に点を打つ
line	指定点を直線で結ぶ
getpalete	指定位置の点のパレット番号取得

これらの関数ではつぎの関数を用いている。

setb	バイト・データの設定
setw	ワード・データの設定
lio	glioの割り込み

```
setb(adr,dat)
int adr,dat;
{
  movedata(DSEG,&dat,GSEG,adr,1);
}

setw(adr,dat)
int adr,dat;
{
  movedata(DSEG,&dat,GSEG,adr,2);
}

lio(intno)
int intno;
{
  inregs.x.bx=0;
  segregs.ds=GSEG;
  int86x(intno,&inregs,&outregs,&segregs);
}
```

図2

(3)　プリンタ・インターフェイス

グラフィック画面をプリンタにハード・コピーするプログラムgcopyを作成するためにプリンタ・インターフェイスが必要である。プリンタへの出力法としてはプリンタの出力ポート0×40に直接データを書き出す方法をとる。つぎの２つの関数を用意する。

dlputc	１文字出力
dlputs	文字列の出力

(4)　MS－DOSインターフェイス

画像を空間領域から周波数領域にフーリエ変換してパワー・スペクトルを求め，逆変換するにあたり

```
static dlputc(c)
char c;
{
while ((inport(0x42)&4) !=4)
   ;
   outportb(0x40,c);
   outportb(0x46,14);
   outportb(0x46,15);
}

static dlputs(str)
char *str;
{
while (*str!=0) {
   dlputc(*str++);
   }
}
```

図3

周波数領域でフィルタ操作を行う際，パワー・スペクトルを3次元立体表示すると都合よい。そのようなとき，グラフィック画面上の原画像を一時ラム・ディスクに退避させておき，逆変換後，原画像とフーリエ変換像を並べて表示したい。そのとき，一時退避させておいた原画像をディスクからグラフィック画面に再現しなければならない。そこで，グラフィック画面全体の保存と読み込みを行うプログラムgsaveとgloadを作成する。

グラフィックVRAMのデータは0×a800, 0×b000, 0×b800を先頭とするメモリにそれぞれ32000バイト（640 * 400 ドット）ずつ格納されている。このデータをディスクにセーブするのにTurbo－Cのセグメント間データ転送関数movedataを用いたのでは時間がかかりすぎる。また，スモール・モデルでのread/write関数はデータ・セグメント内のバッファとの間でしかリード／ライトできない。そこで，セグメント外のバッファとの間で直接リード／ライトを行う関数dread/dwriteを準備する。

MS－DOSのシステム・コールをする。すなわち，レジスタahにファンクション・コードを入れ，さらに必要なパラメータをレジスタcx, dxなどに入れてソフトウエア割り込みのint 21hを行なう。これによりファンクション・コードで定められた機能が実行される。far型int 0×21を実行する関数として，Turbo－CではintdosxがⅡ意されている。

```
gsave()
{
  unsigned int fd,p,adr;
  char buf[640],fname[64];
  printf("File name ? ");scanf("%s",fname);
  if ((fd=open(fname,O_CREAT | O_RDWR |
              O_BINARY,S_IWRITE))==-1) {
    printf("Can't creat¥n");
    exit();
  }
  for (p=0xa800;p<=0xb800;p=p+800){
    for (adr=0;adr<80*400;adr=adr+6400){
      dwrite(fd,p,adr,6400);
    }
  }
  close(fd);
}
```

図4

4・3　ファイル処理

原画像データは2次元配列imageに，また，フーリエ変換像の実部，虚部はそれぞれ2次元配列real, imagに格納されている。save,loadではデータimageの読み出し書き込みはシーケンシャル・ファイルとして連続的に行えるので，ストリーム入出力関数のブロック・リード／ライトfread/fwriteを用いる。

```
void load_data()
{
  FILE *f;
  char fname[30];

  clstxt();locate(1,22);
  printf("file name : ");
  scanf("%s",fname);
  f=fopen(fname,"r");
  wx=getw(f);wy=getw(f);
  fread(image,SIZE,1,f);
  fclose(f);
}
```

図5

また，フーリエ変換のプログラムでも，行方向処理の場合は連続してデータを読み出し書き込みできるが，列方向処理の場合は飛びとびにデータを読み書きしなければならない。そのため，ファイル現在位置の移動関数lseekを用いる必要がある。そして，ラム・ディスクとの間のデータの読み書きには低水準ファイル入出力関数のread/writeを用いることにする。

4・4　call by reference

関数間のデータ授受の方法としてCの最も一般的な引数渡しの方法はcall by valueである。call by

```
void fft2(int flag)
{
int i,j;
float rx,ry;

/* FFT / inverse FFT */

if ((fr=open(frname,O_RDWR | O_BINARY))==-1) {
  printf("Can't open real file");exit();
  }
if ((fi=open(finame,O_RDWR | O_BINARY))==-1) {
  printf("Can't open imaginary file");exit();
  }

for (i=0;i<size;i++) {
  for (j=0;j<size;j++) {
    lseek(fr,(long)(i*size+j)*4,0);
    read(fr,&rx,4);x[j]=rx;
    lseek(fi,(long)(i*size+j)*4,0);
    read(fi,&ry,4);y[j]=ry;
    }
    FFT1(flag);
  for (j=0;j<size;j++) {
    rx=x[j];lseek(fr,(long)(i*size+j)*4,0);
    write(fr,&rx,4);
    ry=y[j];lseek(fi,(long)(i*size+j)*4,0);
    write(fi,&ry,4);
    }
  }
for (i=0;i<size;i++) {
  for (j=0;j<size;j++) {
    lseek(fr,(long)(j*size+i)*4,0);
    read(fr,&rx,4);x[j]=rx;
    lseek(fi,(long)(j*size+i)*4,0);
    read(fi,&ry,4);y[j]=ry;
    }
    FFT1(flag);
  for (j=0;j<size;j++) {
    rx=x[j];lseek(fr,(long)(j*size+i)*4,0);
    write(fr,&rx,4);
    ry=y[j];lseek(fi,(long)(j*size+i)*4,0);
    write(fi,&ry,4);
    }
  }
  close(fr);close(fi);
}
```

図6

valueでは実引数の値を仮引数に渡すため，引数を用いて呼び出し側に値を返すことができず，関数名を介して戻り値が一つだけ返される。

ところで，フーリエ変換の結果パワー・スペクトルを立体表示するのに，3次元図形上の点P(x,y,z)を透視変換してグラフィック・ディスプレイ上の2次元座標P'(u,v)に変換しなければならない。

このように，戻り値が二つ以上ある場合はデータ授受の方法としてcall by referenceの手法を用い，実引数のアドレスを仮引数に渡す。呼ばれた関数ではポインタを介して呼び出し側の関数とデータをやりとりする。

```
purse(double qx, double qy, double qz,
                 double *x, double *y)
{
  const double D1=50.0,D2=50.0,
               X_DOT=640.0,Y_DOT=400.0,
               X_CM=24.0,Y_CM=17.0;
  double KX,KY;
  KX=X_DOT/X_CM; KY=Y_DOT/Y_CM;
  *x=D2/(D1+D2-qz)*qx*KX+X_DOT/2-100.0;
  *y=Y_DOT/2-D2/(D1+D2-qz)*qy*KY;
}


void draw_line(double x1,double y1,double z1,
               double x2,double y2,double z2,
               int col)
{
  double xx1,yy1,xx2,yy2,xx,yy,zz;
  henkan(x1,y1,z1,&xx,&yy,&zz);
  purse(xx,yy,zz,&xx1,&yy1);
  henkan(x2,y2,z2,&xx,&yy,&zz);
  purse(xx,yy,zz,&xx2,&yy2);
  line(xx1,yy1,xx2,yy2,col,0);
}
```

図7

参考文献

1. 吉村　卓　パソコン用画像処理プログラム　秋田高専研究紀要第21号　1986年
2. 町田東一，小島紀男　パソコンBASIC数値計算Ⅱ　東海大学出版会　1985年
3. 安居院猛，中島正之，長尾智春　TURBO pascal画像処理の実際　工学社　1988年
4. 河西朝雄　TURBOC初級プログラミング　技術評論社　1988年