Bull. Kagoshima Pref. Jr. College, 41, 127－139 (1990)

C言語による3次元の運動分析

広　瀬　春　次・志堂寺　和　則*

C言語は，現在では，きめ細かなハードウェアの操作が可能な高級言語として，パソコン用言語の主流をなしてきている。従来，CPUを直接取り扱う言語としてアセンブラが用いられてきたが，ハードウェアにあまりに密着しているためプログラムの意味を一目で理解するのは容易ではなく，他の人にはなかなか活用できないといった問題や，機種が異なると使えなくなるといった問題があった。一方，高級言語であるBASICは人間の言語に近く，理解が容易で移植性も高いが，ハードウェアの細かな操作は不可能で，しかもインタプリタ方式であるために早い速度を必要とする課題には不適切であった。このような相対立する問題点を解決するのにC言語は極めて有効である。心理学でもハードウェア面も含めたパソコン利用が不可欠であるとともに，多くの人が活用できるプログラムの作成が望まれている。C言語は，実験心理学からの要求に最も合致したプログラミング言語であると考えられる。

本論文では，乳幼児のリーチングの3次元分析のためのC言語プログラムを紹介するが，これらのプログラム並びに装置はより一般的な運動の分析にも用いることができるかもしれない。Hofsten (1982) は，Bower (1970) が報告した新生児における目と手の協応，即ち，見た物に手を伸ばすリーチングの現象をより明らかなものにするためビデオ録画による手の詳細な運動分析を行なっている。彼の分析のおもしろい点は，加速と減速局面を含むバリスティック運動を運動継起の単位とみなしたことである。実際，運動の研究領域では，バリスティックな運動は一度喚起されると途中で軌動修正されることなくそれが完了するまで逐行されることが示唆されている。この考えに基づきHofstenは，1つの運動単位における対象物に対する手の運動角度を測定し，新生児において意図的というより源初的・注意喚起的な対象物方向への手の動きがあることを見出した。彼の研究では，ビデオ画面から手の位置を読み取るためXYリーダを用いているが，マウスを利用すればもっと簡単に正確な位置を取得できるはずである。本研究では，手と対象物の実際空間における位置を読み取る方法を紹介する。

装置

PC9800系パーソナルコンピュータ（ディスプレイはビデオ入力の可能なもの），2つのカメラからの映像を同時に録画するためのミキサー（本研究では，National Ouad Syst wj-400 を用いたが4チャンネル用でディスプレイを4分割するため精度はかなり低下した。），ビデオ入力を可能にするビデオコンバータ（本研究では，Central Engneering PVC400を用いた。

* 九州大学文学部

手続

最初に，４角柱の棒で作られたＬ字型のキャリブレーション刺激を被験体がくる位置あたりに水平に置き，正面と横の２台のカメラでとる。カメラはキャリブレーション刺激の直角に交わる辺の延長上に据える。カメラの高さはキャリブレーション刺激と同じにする。この場合，カメラからキャリブレーション刺激を見ると，その辺の端の４角柱が画面の中心に見えるはずである。このようにキャリブレーション用の映像を何コマか録画した後，被験体の運動を録画する。

次に位置データを取得するため，テープに録画された２方向からの映像をコンピュータのディスプレイにグラフィック画面と重複させて映す。まず最初のキャリブレーション用の画面からキャリブレーション刺激の角（基準点あるいは０点）と各辺の端点の座標をマウスを使って取得する。次に被験体の運動を写した画面から手と対象物の座標をマウスカーソルを動かして取得する。従って，１コマにつき正面と横の画面から２点ずつ，計４点の座標が読み込まれることになる。本プログラムでは，キャリブレーション刺激の角と手の座標はマウスの左ボタンを一度押すことによって，キャリブレーション刺激の辺の端点と対象物の座標は左ボタンを二度押すことで読み込まれるようになっている。１コマでの座標取得の終了はマウスの右ボタンを押すことで達成される。

３次元座標の導き方

実際空間での基準点（０点）に対する手（あるいは対象物）の座標（x，y，z）をディスプレイ画面上の座標から導き出すには，まず水平平面上における手（対象物）の座標（x，y）を求める必要がある（図１参照）。

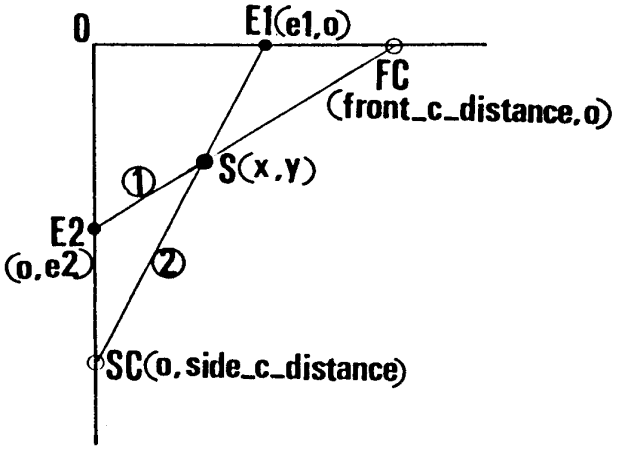

図１　基準点とカメラ，手（対象物）の水平平面上の位置関係《ＦＣ；正面のカメラ，ＳＣ；横のカメラ，Ｅ１；横のカメラに写った手（対象物），Ｅ２；正面のカメラに写った手（対象物）》

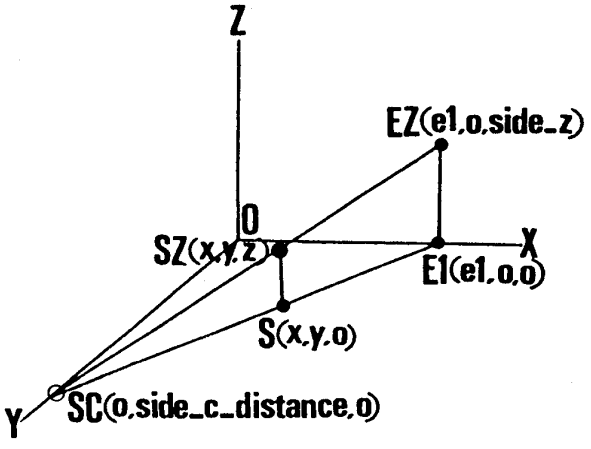

図２　図１にＺ軸を加えた場合の位置関係《ＥＺ，横のカメラに写った手（対象物），ＳＺ；実際空間での手（対象物の位置）》

2つの座標（x_1, y_1），（x_2, y_2）を通る直線は次の式で表わされる。

$(x_2-x_1)(y-y_1)=(y_2-y_1)(x-x_1)$ ――――①

①の式を用いて直線①と②の式が求められる。手（対象物）のx，y座標は，直線①と②の交叉する点であるから，

$$x=(e2-side\_C\_distance)\times\frac{front\_c\_distance\times e1}{e2\times e1-front\_c\_distance\times side\_c\_distance}$$

$$y=\frac{e2}{frnot\_c\_distance}X''x+e2$$

x，yをもとに基準点からの手（対象物）の高さ（Z）を求めるには，横のカメラSCからEIまでの距離（sla1）とSまでの距離（sla2）がわかればよい。図2から次のような式が成立する。

side_Z：Z＝sla1：sla2

従って

Z＝side_Z×sla2/sla2

C言語関数

付表のプログラムは，コンパイラの速さとライブラリの豊富さに優れたTURBO Cを使うことを前提につくられている。以下，各関数の働きを簡単に説明する。

○マウスを操作する関数

mouse_on( )；マウスカーソルの表示。

mouse_off( )；マウスカーソルの消去。

mouse_wait( )；マウスのボタン押しがなくなるまで待つ。

○表示のための関数*

init-graph( )；現在のグラフィックドライバーを検出し，それらが用いられるようにする。

gputs( )；画面の所定の位置に文字を表示。

ilgputs( )；titleで表わされる文字に続けて，numberで示される文字を表示。

display_flame( )；画面を分けるための線を引く。

mark( )；マウスの左ボタンを押した時のマウスカーソルが示す点に，指定された色の“＋”印を表示。

erase_strings( )；何コメ目かを示すcounterを消す。

disp_counter( )；counterを表示。

print_coor( )；手と対象間の実際の距離を表示。

move_disp( )；手が動いているかいないかの表示。

＊これらの関数の中には画面の4分割に対応したものがあり，2分割画面の場合には変更する必要がある。

○座標取得と距離・スピード等を導くための関数

Judge_1_2( )；最初の左ボタン押しの後，そのクリックの持続時間以内に2回目のボタン押しがあれば（即ちj＜iであれば），ダブルクリックとみなし（count＝2），そうでなければシングルクリック（count＝1）とみなす。

setmark( )；xが640/2より小さいか否かで，その点が正面の画像に属するのか，横からの面像に属するのかを判断する。返り値のKind＝0は横カメラからの手の位置，Kind＝1は横カメラからの対象物の位置，Kind＝2は正面カメラからの手の位置，Kind＝3は正面カメラからの対象物の位置の座標が取得されたことを意味する。

translation( )；基準点に対する手（対象物）の画面上の座標を実際値に変換して表わす。横カメラがとった画面上の1ドットは実際空間では，キャリブレーション刺激の辺の長さ(20cm)をその画面上での長さ（cal_dot_F）で割った値となる。

FScomposition( )；図1で示された画面上ではなく実際の水平平面における手（あるいは対象物）の座標（x，y）の算出。

Zcomposition( )；図2で示されたZ値の算出。

disp_distance( )；x，y，z軸それぞれにおける手と対象の距離を算出し，それから実際の距離を導いている。

s_velocity( )；1コマ前の座標と現在の座標の差異（手の移動距離）にコマ数をかけて速度ベクトルを出し，それから実際の速度を求める関数。

oldtopost( )；新たな座標を*(oldx_i)，*(oldY_i)の配列に入れるため，今まで入っていた座標を別の配列に移しかえる関数。

cal_end( )；1コマにつき4座標点がすべて読み込まれた場合1を返し，そうでない場合0を返す。

○主要な関数

main( )；正面カメラと横カメラからキャリブレーション刺激の角までの実際の距離，それに1秒間に何コマ分析するかを入力した後，主要な関数を呼び出し実行する。

cal( )；基準点の画面上での位置，キャリブレーション刺激の辺の画面上での長さを求める。横画面と正面画面における基準点と端点の座標をold_X[4]とold_Y[4]の配列に読み込む。

data_in( )；手と対象物の画面上での座標を取得する。*(oldX＋0）と*(oldX＋1）は横カメラからの画面における手と対象物のx座標，*(oldX＋2）と*(oldX＋3）は正面カメラからの手と物のx座標を表わす。old_Xとold_Yは最後のコマで取得した座標，temp_X，temp_Yは1つ前，post_X，post_Yは2つ前のコマにおいて取得された座標である。

disp( )；連続する3コマのそれぞれにおける手と対象の実際の座標を関数translation( )とcomposition( )によって導き，表示する。さらにその座標点の変化から速度，加速度を求め表示する。コマ数が1より小さいときはS_velocity( )に引数が渡せないのでif(i＞1)以下の命令は実行されない。同様にコマ数が2以下の時はaccは得られないのでif(i＞2)以

下の命令も実行されない。

参 考 文 献

Bower, T. G. R., Broughton, J. M., & Moore, M. K. Demonstration of intention in the reaching behavior of neonate humans. Nature, 1970, 228, 679−681.

Hofsten, C. von. Eye−Hand Coordination in the Newborn. 1982, 18(3), 450−461.

三田典玄　入門，実習，応用C言語，アスキー出版局，1988。

付　　表

```
/*--------------------------------------------------------------------
 *                        3D MOVEMENT ANALYSIS
 *--------------------------------------------------------------------
*/

#include    <stdio.h>
#include    <stdlib.h>
#include    <dos.h>
#include    <process.h>
#include    <alloc.h>
#include    <bios98.h>
#include    <math.h>
#include    <graphics.h>
MOUSE-INIT  mint;
MOUSE-INFO  minf;
float   samp;
int     front_c_distance;
int     side_c_distance;
int     cal_dots_F;
int     cal_dots_S;
int     center_F;
int     center_S;
int     center_Z;

void init-graph(void)                       /*graphic initialize*/
 {

    int grdriver = DETECT, grmode;
    initgraph(&grdriver, &grmode, "");

 }

void gputs(int x, int y, char *string, int color) /*character display*/
```

```
{
    setcolor(color);
    outtextxy(x,y,string);
}

void ilgputs(int x, int y, char *title, int number, int color)
{
    char        i[10], s[80] ;

    itoa(number, i, 10);
    strcpy(s, title);
    strcat(s, i);
    gputs(x, y, s, color);
}

void mouse_on(void)
{
    mint.cmmd=1;                          /*cursor on          */
    bios98mouse_init(&mint);
}

void mouse_off(void)                      /*mouse exit process  */
{
    union REGS st ;

    st.x.ax=2 ;
    int86(51, &st, &st) ;
}

void display_frame(void)                  /*data_in frame       */
{
    rectangle(0, 0, 639, 399);
    line(0, 200, 640, 200);
    line(320, 0, 320, 200);
}

void mark(int x, int y, int color)                /*one click point     */
{
    setcolor(color);
    line(x-5, y, x+5, y);
    line(x, y-5, x, y+5);
}

void mouse_wait(void)                     /*wait for mouse bottons off  */
```

```
{
    while(1) {
        bios98mouse(&minf);
        if (minf.lft_buton==0 && minf.rgt_buton==0) break ;
    }
}

int judge_1_2 (void)                          /*mouse one or double clicks    */
{
    int       count=1 ;
    float     i, j ;

    for (i=0;;i++) {                          /*left button off     */
        bios98mouse(&minf);
        if (minf.lft_buton==0) break ;
    }
    for (j=0:j<i;j++) {                       /*double clicks ?     */
        bios98mouse(&minf);
        if (minf.lft_buton==-1) count=2;
    }
    mouse_wait( );
    return(count);
}
int setmark(int count, int x, int y)
{
    int       kind ;

    if (count==1) { mark(x, y, LIGHTCYAN);
    }else {         mark(x, y, RED);    }
    if (x<640/2) { kind=count-1 ;
    } else {        kind=count+1 ;      }
    return(kind);
}

void erase_strings(int xl, int yl, int x2)    /*erase rectangle area*/
{
    setfillstyle(SOLID_FILL, BLACK);
    bar(x1, y1+2, x2, y1+14);
}

void disp_counter(int i)
{

    erase_strings(10+8*10, 210, 10+8*13);
```

```
    ilgputs(10,210,"counter : ",i,WHITE);
}
                                        /*translate from dots to cm*/
void translation(int *old_x,int *old_y,int *trans_O,int *trans_S)
{
    *(trans_0+0)=20.0*(float)(*(old_x+O)-center_F)/(float)cal_dots_F ;
    *(trans_0+1)=-20.0*(float)(*(old_x+2)-center_S)/(float)cal_dots_S ;
    *(trans_0+2)=20.0*(float)(*(old_y+0)-center_Z)/(float)cal_dots_F ;
    *(trans_S+0)=20.0*(float)(*(old_x+1)-center_F)/(float)cal_dots_F ;
    *(trans_S+1)=-20.0*(float)(*(old-x+3)-center_S)/(float)cal_dots_S ;
    *(trans_S+2)=20.0*(float)(*(old-y+1)-center_Z)/(float)cal_dots_F ;
}

void FScomposition(int *trans,int *f,int *s)        /*FS axis composition     */
{
    int       e1,e2 ;
    float     d1,d2,d3 ;

    e1=*(trans+0);
    e2=*(trans+1);
    d1=e2-side_c_distance ;
    d2=(float)front_c_distance*(float)e1 ;
    d3=(float)e2*(float)e1-(float)front_c_distance*(float)side_c_distance ;
    *f=d1*d2/d3 ;
    *s=-1.0*(float)e2/(float)front_c_distance*(*f)+e2 ;
}

void Zcomposition(int *trans,int f,int s,int *z)    /*Z axis composition      */
{
    double sla1,sla2 ;

    sla1=hypot((double)(*(trans+0)),(double)side_c_distance);
    sla2=hypot((double)f,(double)(side_c_distance-s));
    *z=-1.0*(double)(*(trans+2))*sla2/sla1 ;
}

void composition(int *trans,int *f,int *s,int *z)    /*axis composition*/
{
    FScomposition(trans,f,s);
    Zcomposition(trans,*f,*s,z);
}

print-coor(char *title,int xx,int yy,int x,int y,int z)
{
```

```
    gputs(xx,yy,title,WHITE);
    erase_strings(xx+8*10,yy,xx+8*(10+18));
    ilgputs(xx+8*10,yy," ",x,WHITE);
    ilgputs(xx+8*(10+5),yy,",",y,WHITE);
    ilgputs(xx+8*(10+11),yy,",",z,WHITE);
    gputs(xx+8*(10+17),yy,")",WHITE);
}

void disp_distance(int of,int os,int oz,int sf,int ss,int sz,int *y)
{
    int     dist;
    double  f,s,z;

    f=sf-of;    s=os-ss;    z=oz-sz;
    print_coor("distance(",150,*y,(int)f,(int)s,(int)z);
    dist=(int)(sqrt(f*f+s*s+z*z));
    erase_strings(410+8*15,*y,410+8*(15+5));
    ilgputs(410,*y,"0-S distance=",dist,WHITE);
    *y+=20;
}

int s_velocity(int ff,int ss,int zz,int *y)
{
    int     vel;
    double  f,s,z;

    f=ff*samp;  s=ss*samp;  z=zz*samp;
    vel=(int)(sqrt(f*f+s*s+z*z));
    erase_strings(150+8*15,*y,150+8*(15+5));
    ilgputs(150,*y,"obj velocity=",vel,WHITE);
    *y+=20;
    return(vel);
}

void mov_disp(int x,int *y,char *title,int movement)
{
    erase_strings(x,*y,x+8*20);
    if(movement==0) {
        gputs(x,*y,title,WHITE);
    }else{
        ilgputs(x,*y,title,movement,WHITE);
    }
    *y+=20;
}
```

```
                                                    /*measurements display*/
void disp(int i,int *old_x,int *old_y,int *temp_x,int *temp_y,
          int *post_x,int *post_y,int *movement,int *m_flg)
 {
    int     y=230, of, os, oz, sf, ss, sz ;
    int     tof, tos, toz,  tsf, tss, tsz ;
    int     pof, pos, poz, psf, pss, psz ;
    int     vell=0, vel2=0, acc ;
    int     trans_0[3], trans_S[3] ;

    disp_counter(i);
    translation(old_x, old_y, trans_O, trans_S);
    composition(trans_O, &of, &os, &oz);
    composition(trans_S, &sf, &ss, &sz);
    gputs(222,210,"(front,side,hight)",WHITE);
    erase_strings(20+8*6,y,20+8*10);
    ilgputs(20,y,"frame",i-1,WHITE);
    disp_distance(of, os, oz, sf, ss, sz, &y);
    if (i>1) {
        translation(temp_x, temp_y, trans_O, trans_S);
        composition(trans_O, &tof, &tos, &toz);
        composition(trans_S, &tsf, &tss, &tsz);
        vell=s_velocity(tof_of, tos_os, toz_oz, &y);
        erase_strings(20+8*6,y,20+8*10);
        ilgputs(20,y,"frame",i_2,WHITE);
        disp_distance(tof, tos, toz, tsf, tss, tsz, &y);
        if (i>2) {
            translation(post_x, post_y, trans_O, trans_S);
            composition(trans_O, &pof, &pos, &poz);
            composition(trans_S, &psf, &pss, &psz);
            vel2=s_velocity(pof_tof, pos_tos, poz_toz, &y);
            erase_strings(20+8*6,y,20+8*10);
            ilgputs(20,y,"frame",i-3,WHITE);
            disp_distance(pof, pos, poz, psf, pss, psz, &y);
            acc=vel1-vel2 ;
            erase_strings(150+8*15,y,150+8*(15+5));
            ilgputs(150,y,"              acc.=",acc,WHITE);

            if (acc==0 && vel1==0) {
                *m_flg=0 ;
                mov_disp(20,&y,"no movement",0);
            }else{
                if (*m_flg==0)  ++(*movement);
                *m_flg=1;
```

```
                mov_disp(20,&y,"movement", *movement);
            }
        }
    }
}
                                                    /*old->post        */
void oldtopost(int *old_x,int *old_y,int *post_x,int *post_y)
 {
    int       i;

    for (i=0;i<4;i++) {
        *(post_x+i)=*(old_x+i);
        *(post_y+i)=*(old_y+i);
    }
}

int cal_end(int *flg)
 {
    if (*(flg+0)==0 || *(flg+1)==0 || *(flg+2)==0 || *(flg+3)==0 ) return(0);
    return(1);
}

void cal(void)                                      /*calibulation       */
 {
    int     i,x,y,kind;
    int     old_x[4]={-10,-10,-10,-10},old_y[4]={-10,-10,-10,-10};
    int     flg[4]={0,0,0,0};

    display_frame( );
    mouse_on( );
    gputs(10,210,"cal : 20cm",WHITE);
    while(1) {
        minf.cmmd=3;
        bios98mouse(&minf);
        if (minf.lft_buton==-1) {
            x=(int)minf.abs_h_crs;
            y=(int)minf.abs_v_crs;
            kind=setmark(judge_1_2(  ),x,y);
            mark(*(old_x+kind),*(old_y+kind),BLACK);
            *(old_x+kind)=x;
            *(old_y+kind)=y;
            *(flg+kind)=1;
    }
    if (minf.rgt_buton==-1) {
```

```
            mouse_wait( );
            if (cal_end(flg)==1)     break ;
          }
      }
      mouse_off( );
      cal_dots_F=abs( *(old_x+0)- *(old_x+1));
      cal_dots-S=abs( *(old_x+2)- *(old_x+3));
      center_F= *(old_x+0);
      center_S= *(old_x+2);
      center_Z= *(old_y+0);
}

void data_in(void)
  {
      int     i=0,x,y,kind,movement=0,m_flg ;
      int     old_x[4]={-10,-10,-10,-10},old_y[4]={-10,-10,-10,-10};
      int     temp_x[4],temp_y[4],post_x[4],post_y[4] ;

      cleardevice( );
      clrscr( );
      mouse_on( );
      disp_counter(i);
      display_frame( );
      do {
          minf.cmmd=3 ;
          bios98mouse(&minf);
          if (minf.lft_buton==-1) {
              x=(int)minf.abs_h_crs ;
              y=(int)minf.abs_v_crs ;
              kind=setmark(judge_1_2( ),x,y);
              mark( *(old_x+kind), *(old_y+kind),BLACK);
              *(old_x+kind)=x ;
              *(old_y+kind)=y ;
          }
          if (minf.rgt_buton==-1) {
              disp(++i,old_x,old_y,temp_x,temp_y,post_x,post_y,
                  &movement,&m_flg);
              oldtopost(temp_x,temp_y,post_x,post_y);
              oldtopost(old_x,old_y,temp_x,temp_y);
              mouse_wait( );
          }
      }while(!kbhit( ));
}
```

```
void main(void)
 {
    init_graph( );                                  /*graphic initialization    */
    printf("input the distance from front camera to calibulation");
    scanf("%5d",&front_c_distance);
    printf("input the distance from side camera to calibulation");
    scanf("%5d",&side_c_distance);
    printf("input sampling");
    scanf("%5f",&samp);
    clrscr( );
    cal( );
    data_in( );
    mouse_off( );
    closegraph( );
 }
```

(平成2年9月17日受理。)