〔非公開〕

ＴＲ－Ｍ－００２０

特徴点の平面への写像作成

坂　井　　克　也
Katsuya　SAKAI

灰　塚　　凡　樹
Tsuneki　HAIZUKA

井　上　　誠　喜
Seiki　INOUE

１９９７．３．１９

ＡＴＲ知能映像通信研究所

特徴点の平面への写像作成

1997年3月19日

ATR知能映像通信研究所　第三研究室
法政大学工学部電子情報学科

坂井克也

1 背景・目的

人に分かりやすい映像検索を提供するための方法としてシナリオに基づく映像検索方法が提案されている [1]。

この方法においてはカメラワークを正確に算出することが重要なポイントとなる。文献 [1] では、カメラワークを算出するために 3 次元カメラモデルを用いてプログラムによる自動検出を試みている。この結果は、極大極小点は実際の動きと一致していたが実際の動きとは異なる部分も数ヵ所あった。

そこで本実習ではカメラワークの検出を手作業で行い主演者の移動軌跡を作成し、カメラワークと人の動きの対応を確認してみた。

2 処理手順

次のような手順で作業を行った。

- カメラワークの測定
- 人物の移動軌跡の抽出
- 写像作成

3 カメラワークの測定

前準備

- MTX1 を ProntoVideo につなげ、スイッチャの AUX1 の AUX BUS を MTX1 にする。(ProntoVideo(RD2500) の出力を DME9000 に通す)
- マスタモニタを DVS ME1 につなげる。(スイッチャでモニタに出力する映像を切り替えられるようにする)
- 編集機や pronto_gui を用いて能の映像の 1 フレーム目を表示する。
- この映像を特殊効果をかける装置で半分のサイズに縮小する。これは画像が画面の外にはみ出さないようにするためである。
- 次にスイッチャのフレームメモリ 1 に、表示されている画面を保存しておく。(フレームメモリに取り込んだ画像は原画像よりも多少下にずれるので、画像の位置を比較する場合には注意する必要がある)

測定手順

- 編集機や pronto_gui を使用して能の映像を表示させる。

- スイッチャでDME9000の出力とフレームメモリ内の画像を切り替えながら、トラックボールを操作してProntoVideoの画像を移動・拡大・縮小し、フレームメモリに保存してある画像と背景の位置・大きさを合わせる。
- その時のsystem-gのLocXYZの数値を記録する。

これらの操作を適当なフレーム間隔で繰り返す。

測定結果

測定結果をグラフにしたものが図1と図2である。これらのグラフの横軸は時間、縦軸はsystem-gでの数値をあらわしている。

この結果は、3次元カメラモデルに基づいたプログラムによる検出結果(図3, 4)とはかなりグラフの形が異なっている。これはプログラムで特徴点を抽出するときに問題が起こっているのではないかと思われる。なお、図3と図4の横軸はフレーム数、縦軸はプログラムでの数値である。

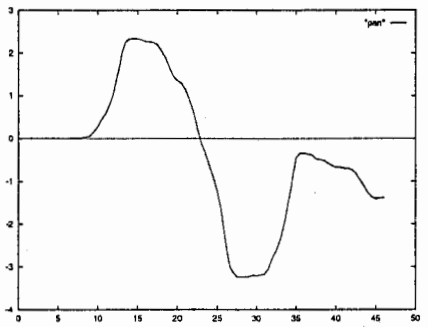

図 1: パンの測定結果

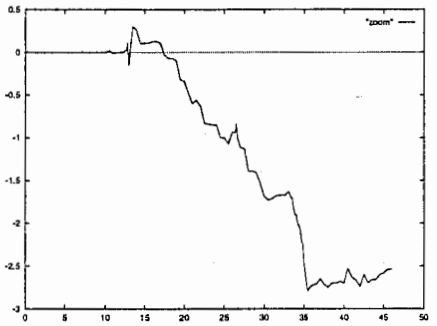

図 2: ズームの測定結果

4 人物の移動軌跡の抽出

軌跡の抽出手順

- 計測データをsysganm.exe用のデータに書き換える。

  計測データをエディタで多少整形してコマンドラインから

  > convsysg ファイル名 > anmset.dat

  などとする。(convsysgの詳しい使い方については付録参照)

- 変換したデータをsystem-gに読み込ませる。

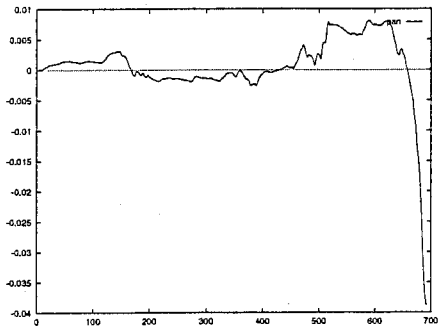

図 3: プログラムによるパン検出結果

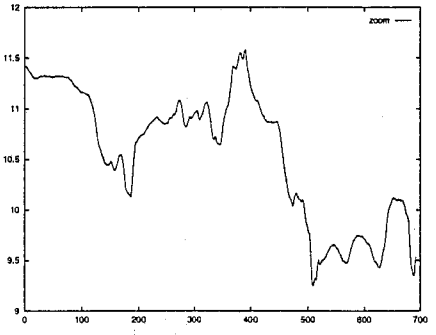

図 4: プログラムによるズーム検出結果

c:/sysg/data/anm に anmset.dat をおき sysganm.exe を実行する。

- 次に編集機を用いて、再生している映像に system-g で先ほど読み込ませた特殊効果をかけビデオに録画する。このときの映像はカメラを固定して撮影したようなものになる。

- vcp を使用してビデオをムービーファイルにおとす。

  vcp の設定を適切に行った後にコンソールで

  > dmrecord -t 46 -p video,device=ev1,engine=cosmo,quality=70 出力ファイル

  とし、録画したい場面が再生されたところで Enter キーを押す。

- movieconvert を用いてムービーファイルから 30 フレーム毎に画像を保存する。

- それぞれの画像に次の処理を行い足の位置を取り出す。

  - imgworks を使用してコントラストを変える。
    (Shadows:0.7, Highlights:0.3 とする)
  - imgworks を使用してエッジを検出する。
    (エッジ検出のアルゴリズムは Roberts を使用)
  - imgworks を使用してグレースケールにする。
  - bott を使用して足の部分だけ取り出す。

    コマンドラインで、

    > bott 入力ファイル名 出力ファイル名 100 480

    とする。

足の部分の輝度は足より下の他の部分に比べて高いので、imgworksを使用した処理は省略してもある程度の軌跡は得られる。ただし、bottのパラメータの閾値の部分を200程度にする必要がある。

- すべての画像の足の位置が取り出せたら、maxを用いて画像を重ね合わせて軌跡をつくる。

  コマンドラインで、

  > max 入力ファイル1 入力ファイル2 出力ファイル

  とする。

- 舞台上での軌跡を分かりやすくするため舞台の画像を合成する。

  コマンドラインから、

  > max 軌跡ファイル名 背景ファイル名 出力ファイル名

  とする。

結果

前述の方法で軌跡を取り出した過程とその結果が図5〜図10である。

今回はimgworksでのコントラストの変化率やbottでの閾値をすべての画像に対して同じ値を用いたが、画像にあわせて値を変えると、もう少し軌跡が見やすいものになるのではないかと思われる。

図5: 原画像

図6: コントラストを変化

5 舞台上の点の平面上への写像作成

- perspectを用いて軌跡の画像を変形させる。

図 7: エッジ検出

図 8: 足の位置を取り出した画像

図 9: 軌跡

図 10: 背景合成

コマンドラインで、

> perspect inimage outimage 33 210 0 62 0 0 31 0 60 2

とすればよい。
パラメータは他の位置から撮った映像の場合には変更する必要がある。

図10の画像に、この処理を施した画像が図11である。この図から舞台上での主演者の移動軌跡を確認することができた。

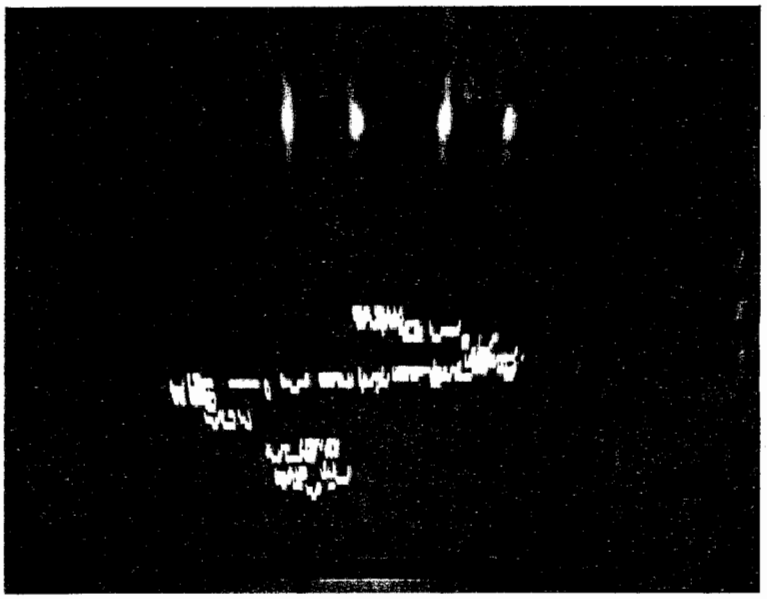

図 11: 透視変換をした画像

6 他の映像での結果例

前述の方法で、別の人が舞っている同じ場面の「能」映像の軌跡を作ってみた。(図 12〜図 14)

ただし、imgworks を用いたコントラスト変更、エッジ検出、グレースケール化の処理は行わず閾値操作のみで作成した。

なお、透視変換の際のパラメータは次の通りである。

> perspect infile outfile 48 210 0 72 0 0 31 0 60 1.3

図 12: 原画像の例

図 13: 軌跡

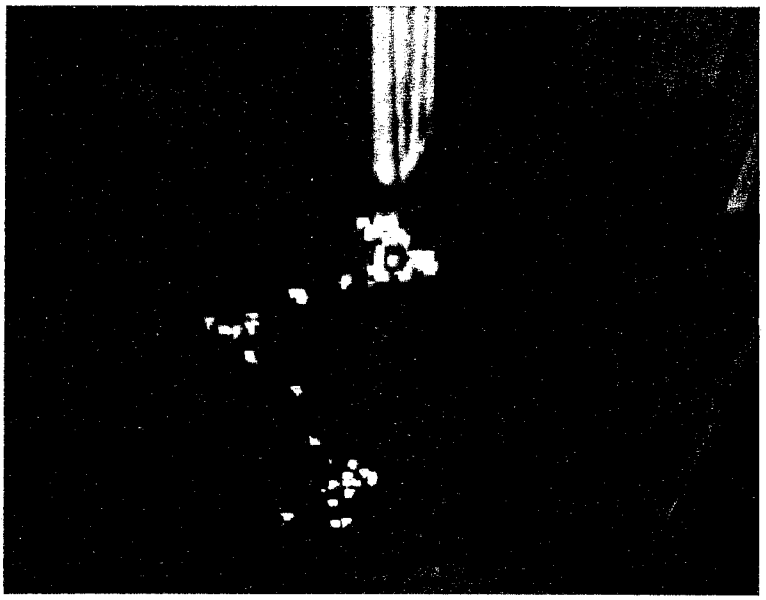

図 14: 背景を合成し透視変換した画像

参考文献

[1] 灰塚凡樹, 井上誠喜, “シナリオに基づく映像検索方法”, 信学技報 HIP96-35, pp.61-66, 1997

[2] 八木伸行, 井上誠喜 他, “C 言語で学ぶ実践画像処理 ”, オーム社

A　計測データ

計測データを以下に示す。

時間	パン	チルト	ズーム
0:00	0	-0.05	0
1:00	0	-0.05	0
2:00	0	-0.05	0
3:00	0	-0.05	0
4:00	0	-0.05	0
5:00	0	-0.05	0
6:00	0	-0.05	0
7:00	0.01	-0.04	0
8:00	0.015	-0.045	0
9:00	0.055	-0.045	0
10:00	0.28	-0.045	0
10:15	0.44	-0.045	0.02
11:00	0.58	-0.045	-0.01
11:15	0.79	-0.04	0
12:00	1.05	-0.04	0
12:05	1.2	-0.04	0
12:10	1.33	-0.04	0
12:15	1.48	-0.04	0.02
12:20	1.61	-0.03	0.02
12:25	1.77	-0.03	0.1
13:00	1.97	-0.035	-0.15
13:15	2.25	-0.035	0.3
14:00	2.33	-0.035	0.26
14:15	2.33	-0.045	0.1
15:00	2.33	-0.05	0.11
15:15	2.31	-0.05	0.11
16:00	2.27	-0.05	0.13
16:15	2.26	-0.06	0.13
17:00	2.25	-0.06	0.11
17:15	2.19	-0.06	-0.03
18:00	2.04	-0.04	-0.07
18:15	1.88	-0.06	-0.07
19:00	1.65	-0.06	-0.1
19:15	1.46	-0.05	-0.32
20:00	1.36	-0.05	-0.34
20:15	1.29	-0.06	-0.48
21:00	1.13	-0.06	-0.6
21:15	0.89	-0.07	-0.56
22:00	0.61	-0.06	-0.63
22:15	0.22	-0.05	-0.83
23:00	-0.12	-0.05	-0.84
23:15	-0.35	-0.07	-0.85
24:00	-0.63	-0.105	-0.85
24:15	-0.91	-0.145	-1
25:00	-1.24	-0.14	-1
25:15	-1.69	-0.14	-1.07
26:00	-2.31	-0.12	-0.93
26:05	-2.51	-0.13	-0.94
26:10	-2.69	-0.13	-0.94
26:15	-2.87	-0.13	-0.85
26:20	-3.01	-0.14	-1
26:25	-3.07	-0.14	-1.05
27:00	-3.13	-0.14	-1.11
27:15	-3.24	-0.14	-1.13
28:00	-3.23	-0.14	-1.39
28:15	-3.23	-0.14	-1.39
29:00	-3.23	-0.14	-1.41
29:15	-3.2	-0.13	-1.53
30:00	-3.21	-0.12	-1.69
30:15	-3.19	-0.1	-1.73
31:00	-3.17	-0.1	-1.71
31:15	-3.05	-0.11	-1.68
32:00	-2.81	-0.12	-1.67
32:15	-2.63	-0.12	-1.67
32:20	-2.54	-0.13	-1.67
32:25	-2.46	-0.12	-1.65
33:00	-2.37	-0.13	-1.63

33:05	-2.27	-0.13	-1.65
33:10	-2.13	-0.13	-1.69
33:15	-1.99	-0.135	-1.71
33:20	-1.87	-0.16	-1.81
33:25	-1.68	-0.17	-1.9
34:00	-1.49	-0.19	-1.9
34:05	-1.34	-0.22	-2
34:10	-1.18	-0.22	-2.03
34:15	-0.96	-0.24	-2.07
34:20	-0.79	-0.24	-2.2
34:25	-0.62	-0.24	-2.25
35:00	-0.45	-0.25	-2.47
35:15	-0.34	-0.25	-2.79
36:00	-0.34	-0.25	-2.73
36:15	-0.36	-0.23	-2.71
37:00	-0.38	-0.23	-2.65
37:15	-0.47	-0.23	-2.71
38:00	-0.49	-0.23	-2.75
38:15	-0.51	-0.23	-2.7
39:00	-0.57	-0.23	-2.7
39:15	-0.63	-0.23	-2.68
40:00	-0.67	-0.21	-2.7
40:15	-0.67	-0.23	-2.53
41:00	-0.69	-0.23	-2.63
41:15	-0.69	-0.21	-2.67
42:00	-0.73	-0.21	-2.74
42:15	-0.81	-0.21	-2.6
43:00	-0.96	-0.21	-2.7
43:15	-1.11	-0.25	-2.66
44:00	-1.27	-0.25	-2.66
44:15	-1.37	-0.25	-2.6
45:00	-1.41	-0.25	-2.58
45:15	-1.39	-0.25	-2.54
46:00	-1.39	-0.25	-2.54

B　作成したプログラムの概要

特徴点の平面への写像を作成するため4つのプログラムを作成した。また、画像処理のプログラムを作成するための練習として3つのプログラムを作成した。それぞれのプログラムの概要を表Bに示す。

ソース名	概要
convsysg.c convsysg2.c	sysganm.exe 用のデータを出力する
maxrgb.c	2画像の画素のうち rgb 値の合計が大きい方を出力する
bottom.c	画像を下からスキャンしていき閾値以上の輝度の画素がはじめて出てきた行を出力する
perspect.c	透視変換
threshld.c	輝度が閾値以下の画素の rgb 値を変化させる
zoom.c	拡大・縮小
affine.c	拡大・縮小・移動・回転

表 1: プログラム概要

画像処理のプログラムは画像ファイルをいったんメモリに読み込んでから処理を行うので入力ファイルと出力ファイルで同じファイル名を指定することが可能になっている。扱える画像のフォーマットは、IRIS rgb 形式と ppm/pgm(raw) 形式の画像である。

なお、ソースは、miris37:/home/sakai/Source/ 内に格納されている。ディレクトリ構造を以下に示す。

```
/home/sakai/Souce/
                |
                +-------- convsysg/          anmset.dat 生成プログラム
                |
                +-------- ppmtools/          ppm/pgm 用の画像処理プログラム
                |           |
                |           +-- bottom/
                |           +-- max/
                |           +-- perspect/
                |           +-- threshld/
                |           +-- zoom/
                |           +-- test/
                |
                +-------- rgbtools/          IRIS rgb 用の画像処理プログラム
                |           |
                |           +-- affine/
                |           +-- bottom/
                |           +-- max/
                |           +-- perspect/
                |           +-- threshld/
                |           +-- zoom/
                |           +-- test/
                |
                +-------- test/              その他のツール
```

C　プログラム使用法

convsysg.c

convsysg2.c

機能:

絶対値で記述されたファイルを sysganm.exe で読める形式に変換し標準出力に表示する。

使用法:

> convsysg infile
> convsysg2 infile

引数:

infile　　入力ファイル名

入力ファイルのフォーマット:

```
+-----------------------------------(convsysg.c)--+
|  LocSizeZ Step StartSec StartFrame                |
|  zoom pan tilt rot                                |
|  zoom pan tilt rot                                |
|          ...                                      |
|          ...                                      |
+---------------------------------------------------+
```

データ区切り文字はスペース。
データはすべて絶対値で記述する。
Step はフレーム単位で記述する。
StartSec, StartFrame は開始時間。
rot は z 軸を軸とした回転の値である。

```
+-----------------------------------(convsysg2.c)-+
|  LocSizeZ                                         |
|  Sec:Frame pan tilt zoom                          |
|  Sec:Frame pan tilt zoom                          |
|          ...                                      |
|          ...                                      |
+---------------------------------------------------+
```

データ区切り文字はスペース。
データはすべて絶対値で記述する。

例:

LocSizeZ が 0.5、 8 フレーム刻み、 00:00 から始まる場合の例

```
+-- infile ----------------------------------------+
|  0.5  8  0  0                                    |
|  0       0       0       0                       |
|  0.1   -0.0022  0.001   0                        |
|  0.1   -0.0254 -0.0023  0                        |
|  0.3   -0.0312 -0.0162  0                        |
+--------------------------------------------------+
```

LocSizeZ が 0.5 の場合の convsysg2.c の例

```
+-- infile ----------------------------------------+
|  0.5                                             |
|  0:0       0       0       0                     |
|  0:8    -0.0022  0.001   0.1                      |
|  0:16   -0.0254 -0.0023  0.1                      |
|  0:24   -0.0312 -0.0162  0.3                      |
+--------------------------------------------------+
```

補足:

convsysg と convsysg2 でデータ部の並び順が違うので注意すること。また、convsysg.c の 46 行目からのコメントの部分を有効にするとカメラワーク測定プログラムの出力を sysganm.exe 用の入力ファイルに変換できるようになる。ただし、そのままでは正しく変換されないので変換式の部分を修正してください。

max.c

機能:

2 つの入力画像の rgb 値の合計を比べて、値が大きい方の画素の値を出力画像に適用する。

使用法:

> max infile1 infile2 outfile

引数:

infile1	入力画像ファイル名 1
infile2	入力画像ファイル名 2
outfile	出力画像ファイル名

bottom.c

機能:

画像を line 行目から上にスキャンしていき、閾値以上の輝度の画素がはじめてあらわれた行を出力する。他の行は黒で埋められる。

使用法:

> bottom infile outfile threshold line

引数:

infile	入力画像ファイル名
outfile	出力画像ファイル名
threshold	閾値 (0 以上の整数)
line	行数 (0 から入力画像の ysize までの範囲)

補足:

プログラムソース中の #define WID の値を変えることで出力する行数を増やすことができる。

perspect.c

機能:

透視変換を行う。遠くのものほど小さく見えるような効果が得られる。

使用法:

> perspect infile outfile zx zy mx my mz rx ry rz v s

引数:

infile	入力画像ファイル名
outfile	出力画像ファイル名
zx	拡大率 (横)
zy	拡大率 (縦)
mx	移動量 (x)
my	移動量 (y)
mz	移動量 (z)
rz	回転角 (z 度)
rx	回転角 (x 度)
ry	回転角 (y 度)
v	視点の位置 (z)
s	スクリーンの位置 (z)

threshld.c

機能：

輝度が閾値以下の画素の rgb 値を rate の割合に変化させる。

使用法:

> thrshld infile outfile rate threshold

引数:

infile	入力画像ファイル名
outfile	出力画像ファイル名
rate	変化させる割合 (0 以上の実数)
threshold	閾値 (0 以上の整数)

zoom.c

機能：

画像の真ん中を中心に拡大縮小を行う。画像の範囲からはみ出した部分は捨てられる。

使用法:

> zoom infile outfile zx zy

引数:

infile	入力画像ファイル名
outfile	出力画像ファイル名
zx	拡大率 (横)
zy	拡大率 (縦)

affine.c

機能:

画像の拡大・縮小・移動・回転を行う。画像の範囲からはみ出した部分は捨てられる。

使用法:

> affine infile outfile zx zy r mx my

引数:

infile	入力画像ファイル名
outfile	出力画像ファイル名
zx	拡大率 (横)
zy	拡大率 (縦)
deg	回転角 (度)
mx	移動量 (x)
my	移動量 (y)

```
/*

   絶対値で記述されたデータをsysganm.exeで読める形式に変換するプログラム

   Usage: convsysg infile > anmset.dat

   入力ファイルフォーマット
     LocSizeZ Step StartSec StartFrame
         zoom pan tilt rotZ
         zoom pan tilt rotZ
         ...

*/

#include <stdio.h>

main(argc, argv)
int argc;
char *argv[];
{
        FILE *fp;
        float z, x, y, r, z2, x2, y2, r2, z1, x1, y1, r1, Z, lim;
        unsigned int i, j, STEP;

        if(argc!=2){
          fprintf(stderr, "Usage: %s infile\n\n", argv[0]);
          return -1;
        }

        if((fp=fopen(argv[1],"r"))==NULL){
          fprintf(stderr, "cannot open file: %s\n\n", argv[1]);
          return -2;
        }

        z1=x1=y1=r1=0.;
        fscanf(fp,"%f %d %d %d", &Z, &STEP, &i, &j);  /* 一行目の読み込み */
        printf("0:0,0,0,0,%4f,0,0,0\n", Z);

        while(1){
                fscanf(fp,"%f %f %f %f", &z, &x, &y, &r); /* 二行目以降の読み込み */
                if(feof(fp)) break;

                x2=x; y2=y; z2=z; r2=r;

/*  カメラワーク測定プログラムの数値をSYSTEM-Gでの数値に変換  */
/*
                x2=x*788.8;
                y2=-y*9.32;
                z2=(z-11.42)*0.6985;
                r2=r*0.000001;
*/

/*  上限下限のチェック  */
/*
                lim = 4./Z;
                if(x2>lim) x2=lim; else if(x2<-lim) x2=-lim;
                if(y2>lim) y2=lim; else if(y2<-lim) y2=-lim;
*/

                printf("%d:%d,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f\n"
                        , i, j, x2-x1, y2-y1, z2-z1, Z, 0, 0, r2-r1);

                j=j+STEP;
                if(j>29){ i=i+(j/30); j=j%30; }
                x1=x2; y1=y2; z1=z2; r1=r2;
        }
```



```
        fclose(fp);
}
```

```
/*
   絶対値で記述されたデータをsysganm.exeで読める形式に変換するプログラム

   Usage: sysgconv infile > anmset.dat

   入力ファイルフォーマット
         time x y z
         time x y z
         ...
*/

#include <stdio.h>

main(argc, argv)
int argc;
char *argv[];
{
        FILE *fp;
        float z, x, y, r, z2, x2, y2, r2, z1, x1, y1, r1, Z, lim;
        unsigned int  sec, frm;

        if(argc!=2){
          fprintf(stderr, "Usage: %s infile\n\n", argv[0]);
          return -1;
        }

        if((fp=fopen(argv[1],"r"))==NULL){
          fprintf(stderr, "cannot open file: %s\n\n", argv[1]);
          return -2;
        }

        z1=x1=y1=r1=0.;
        fscanf(fp,"%f", &Z);  printf("0:0,0,0,0,%4f,0,0,0\n", Z);

        while(1){
        fscanf(fp,"%d:%d %f %f %f", &sec, &frm, &x, &y, &z);
                if(feof(fp)) break;
                x2=x; y2=y; z2=z; r2=r;
                printf("%3d:%02d,% 8.4f,% 8.4f,% 8.4f,% 8.4f,% 8.4f,% 8.4f,% 8.4f\n"
                        , sec, frm, x2-x1, y2-y1, z2-z1, Z, 0, 0, 0);
                x1=x2; y1=y2; z1=z2; r1=r2;
        }

        fclose(fp);
}
```

```
/*
    2つの画像を比べてrgbの合計が大きい画像の値を出力画像に適用する

       第1引数:入力ファイル1(rgb)
       第2引数:入力ファイル2(rgb)
       第3引数:出力ファイル(rgb)

            (入力ファイルと出力ファイルで同じ名前を指定することも可能)
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "image.h"

#define COLOR 0
#define GREY 1

struct img {
        unsigned char type;
        unsigned int  xsize, ysize, depth;
        unsigned char *rdata;
        unsigned char *gdata;
        unsigned char *bdata;
};

int readrgb(char *, struct img *);
int writergb(char *, struct img *);
int comph(struct img *, struct img *, struct img *);

main(argc, argv)
int argc;
char *argv[];
{
        struct img input1, input2, output;

        if(argc!=4){
          printf("Usage: %s infile1 infile2 outfile\n\n", argv[0]);
          return -1;
        }

        if(readrgb(argv[1], &input1)<0){puts("error"); return -2;}
        if(readrgb(argv[2], &input2)<0){puts("error"); return -2;}

        if(input1.xsize!=input2.xsize || input1.ysize!=input2.ysize || input1.type!=i
nput2.type || input1.depth!=input2.depth){
          puts("  "); return -3;
        }

        comph(&input1, &input2, &output);
        writergb(argv[3], &output);
}

int comph(input1, input2, output)
struct img *input1;
struct img *input2;
struct img *output;
{
        int i, size,  d1, d2;

        size = input1->xsize*input1->ysize;
        output->rdata = (char *)malloc( size );
        if(input1->type==COLOR){
                output->gdata = (char *)malloc( size );
                output->bdata = (char *)malloc( size );
        }
```



```
        output->type  = input1->type;
        output->xsize = input1->xsize;
        output->ysize = input1->ysize;
        output->depth = input1->depth;

        for(i=0;i<size;i++){
          d1 = input1->rdata[i];
          d2 = input2->rdata[i];

          if(input1->type==COLOR){
            d1 = d1 + input1->gdata[i] + input1->bdata[i];
            d2 = d2 + input2->gdata[i] + input2->bdata[i];
          }
          if(d1<d2){
            output->rdata[i]=input2->rdata[i];
            if(input1->type==COLOR){
              output->gdata[i]=input2->gdata[i];
              output->bdata[i]=input2->bdata[i];
            }
          } else {
            output->rdata[i]=input1->rdata[i];
            if(input1->type==COLOR){
              output->gdata[i]=input1->gdata[i];
              output->bdata[i]=input1->bdata[i];
            }
          }
        }
        return 0;
}


/*
   IRIS rgb形式のファイルをimg構造体に読み込む

     返り値  読み込み成功   0
                    失敗  -1
*/
readrgb(filename, data)
char *filename;         /* ファイル名 */
struct img *data;       /* 画像を格納したい構造体 */
{
  IMAGE *image;
  short r[8192], g[8192], b[8192];
  int size, x, y;

  if( (image=iopen(filename,"r"))==NULL ){
    fprintf(stderr, "cannot open file -%s\n",filename);
    return -1;
  }

  data->xsize = image->xsize;
  data->ysize = image->ysize;

  if(image->zsize==1){
    data->type = GREY;
  } else if(image->zsize==3 || image->zsize==4){
    data->type = COLOR;
  } else {
    fprintf(stderr, "unknown file type\n");
    return -1;
  }
  data->depth = 255;

  size = data->xsize*data->ysize;
  data->rdata = (unsigned char *)malloc(size);
  if(data->type==COLOR){
```

```
    data->gdata = (unsigned char *)malloc(size);
    data->bdata = (unsigned char *)malloc(size);
  }

  for(y=0;y<data->ysize;y++){
    getrow(image,r,y,0);
    if(data->type==COLOR){
      getrow(image,g,y,1);
      getrow(image,b,y,2);
    }
    for(x=0;x<data->xsize;x++){
      data->rdata[x+data->xsize*(data->ysize-1-y)] = r[x];
      if(data->type==COLOR){
                data->gdata[x+data->xsize*(data->ysize-1-y)] = g[x];
                data->bdata[x+data->xsize*(data->ysize-1-y)] = b[x];
      }
    }
  }
  iclose(image);
  return 0;
}


/*
   img構造体の中身ををIRIS rgb形式のファイルに書き出す

       返り値   書き込み成功   0
                        失敗  -1
*/
writergb(filename, data)
char *filename;                 /* ファイル名 */
struct img *data;               /* 書き込みたい画像の入った構造体 */
{
  IMAGE *image;
  int y,x;
  unsigned short  r[8192], g[8192], b[8192];

  if(data->type==COLOR)
    image = iopen( filename, "w", RLE(1), 3, data->xsize, data->ysize, 3);
  else
    image = iopen( filename, "w", RLE(1), 2, data->xsize, data->ysize, 1);

  if(image==NULL){
       return -1;
  }

  for(y=0;y<data->ysize;y++){
    for(x=0;x<data->xsize;x++){
      r[x]=data->rdata[x + data->xsize*(data->ysize-y-1)];
      if(data->type==COLOR){
                g[x]=data->gdata[x + data->xsize*(data->ysize-y-1)];
                b[x]=data->bdata[x + data->xsize*(data->ysize-y-1)];
      }
    }
    putrow(image, r, y, 0);
    if(data->type==COLOR){
      putrow(image, g, y, 1);
      putrow(image, b, y, 2);
    }
  }
  iclose(image);
  return 0;
}
```

```
/*

   画像を指定ラインから上にスキャンしていき、
   閾値以上の点がはじめてあらわれたラインを出力するプログラム

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "image.h"

#define COLOR 0
#define GREY 1

#define WID 1

struct img {
        unsigned char type;
        unsigned int  xsize, ysize, depth;
        unsigned char *rdata;
        unsigned char *gdata;
        unsigned char *bdata;
};

int readrgb(char *, struct img *);
int writetrgb(char *, struct img *);
int bott(struct img *, struct img *, int, unsigned char);

main(argc, argv)
int argc;
char *argv[];
{
  struct img in,out;
  int line;

  if(!(argc==4 || argc==5)){
        printf("Usage:%s infile outfile threshold line \n", argv[0]);
        return -1;
  }

  if(readrgb(argv[1], &in)<0){
        fprintf(stderr, "cannot open file - %s\n", argv[1]);
        return -2;
  }

  if(argc==5) line = atoi(argv[4]);
  else line = in.ysize;

  bott(&in, &out, line, (unsigned char)atoi(argv[3]));
  writergb(argv[2], &out);

}



int bott(image_in, image_out, line, th)
struct img *image_in;
struct img *image_out;
int line;
unsigned char th;
{
  int x, y, xsize, ysize, bot, d;

  bot=0;
  xsize = image_in->xsize;
```



```
  ysize = image_in->ysize;

  if(line>ysize) line=ysize;

  for(y=line-1;y>=0;y--){
        for(x=0;x<xsize;x++){
          d=image_in->rdata[x + y*xsize];
          if(image_in->type==COLOR){
                d = 0.3*(float)d + 0.59*(float)image_in->gdata[x + y*xsize]
                  + 0.11*(float)image_in->bdata[x + y*xsize];
          }
          if(d>=th){
                bot=y;
                x=xsize;
                y=-1;
          }
        }
  }

  image_out->type  = image_in->type;
  image_out->xsize = xsize;
  image_out->ysize = ysize;
  image_out->depth = image_in->depth;

  image_out->rdata = (unsigned char *)malloc(xsize*ysize);
  if(image_in->type==COLOR){
        image_out->gdata = (unsigned char *)malloc(xsize*ysize);
        image_out->bdata = (unsigned char *)malloc(xsize*ysize);
  }

  for(y=0;y<ysize;y++){
        for(x=0;x<xsize;x++){
          image_out->rdata[x + y*xsize] = 0;
          if(image_in->type==COLOR){
                image_out->gdata[x + y*xsize] = 0;
                image_out->bdata[x + y*xsize] = 0;
          }
        }
  }

  for(y=bot-WID+1;y<=bot;y++)
        for(x=0;x<xsize;x++){
          d=image_in->rdata[x + y*xsize];
          if(image_in->type==COLOR){
                d = 0.3*(float)d + 0.59*(float)image_in->gdata[x + y*xsize]
                  + 0.11*(float)image_in->bdata[x + y*xsize];
          }
          if(d>=th){
                image_out->rdata[x + y*xsize] = image_in->rdata[x + y*xsize];
                if(image_in->type==COLOR){
                  image_out->gdata[x + y*xsize] = image_in->gdata[x + y*xsize];
                  image_out->bdata[x + y*xsize] = image_in->bdata[x + y*xsize];
                }
          } else {
                image_out->rdata[x + y*xsize] = 0;
                if(image_in->type==COLOR){
                  image_out->gdata[x + y*xsize] = 0;
                  image_out->bdata[x + y*xsize] = 0;
                }
          }
        }

  return 0;
}
```

```
readrgb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  short r[8192], g[8192], b[8192];
  int size, x, y;

  if( (image=iopen(filename,"r"))==NULL ){
    fprintf(stderr, "cannot open file -%s\n",filename);
    return -1;
  }

  data->xsize = image->xsize;
  data->ysize = image->ysize;

  if(image->zsize==1){
    data->type = GREY;
  } else if(image->zsize==3 || image->zsize==4){
    data->type = COLOR;
  } else {
    fprintf(stderr, "unknown file type\n");
    return -2;
  }
  data->depth = 255;

  size = data->xsize*data->ysize;
  data->rdata = (unsigned char *)malloc(size);
  if(data->type==COLOR){
    data->gdata = (unsigned char *)malloc(size);
    data->bdata = (unsigned char *)malloc(size);
  }

  for(y=0;y<data->ysize;y++){
    getrow(image,r,y,0);
    if(data->type==COLOR){
      getrow(image,g,y,1);
      getrow(image,b,y,2);
    }
    for(x=0;x<data->xsize;x++){
      data->rdata[x+data->xsize*(data->ysize-1-y)] = r[x];
      if(data->type==COLOR){
                data->gdata[x+data->xsize*(data->ysize-1-y)] = g[x];
                data->bdata[x+data->xsize*(data->ysize-1-y)] = b[x];
      }
    }
  }
  iclose(image);
  return 0;
}

writergb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  int y,x;
  unsigned short  r[8192], g[8192], b[8192];

  if(data->type==COLOR)
    image = iopen( filename, "w", RLE(1), 3, data->xsize, data->ysize, 3);
  else
    image = iopen( filename, "w", RLE(1), 2, data->xsize, data->ysize, 1);

  for(y=0;y<data->ysize;y++){
    for(x=0;x<data->xsize;x++){
```



```
      r[x]=data->rdata[x + data->xsize*(data->ysize-y-1)];
      if(data->type==COLOR){
                g[x]=data->gdata[x + data->xsize*(data->ysize-y-1)];
                b[x]=data->bdata[x + data->xsize*(data->ysize-y-1)];
      }
    }
    putrow(image, r, y, 0);
    if(data->type==COLOR){
      putrow(image, g, y, 1);
      putrow(image, b, y, 2);
    }
  }
  iclose(image);
  return 0;
}
```

```
/*
   透視変換

   第１引数:入力ファイル(IRIS rgb)
   第２引数:出力ファイル(IRIS rgb)
   第３引数:拡大率(横)
   第４引数:拡大率(縦)
            移動量(x)
            移動量(y)
            移動量(z)
            回転角(z度)
            回転角(x度)
            回転角(y度)
            視点の位置(z)
   第12引数:スクリーンの位置(z)

   (入力ファイルと出力ファイルで同じ名前を指定することも可能)
*/

#include    <stdio.h>
#include    <stdlib.h>
#include    <string.h>
#include        <math.h>
#include    "image.h"

/* 画像タイプ */
#define COLOR 0
#define GREY 1

/* 画像データを格納する構造体 */
struct img {
        char type;
        unsigned int xsize, ysize, depth;
        char *rdata;
        char *gdata;
        char *bdata;
};


main(argc, argv)
int argc;
char *argv[];
{
        struct img input, output;

        if(argc!=13){
          fprintf(stderr, "Usage: %s  infile outfile zx zy mx my mz rz rx ry v s\n\n"
, argv[0]);
          return -1;
        }

        if(readrgb(argv[1], &input)<0) return -2;     /* ファイル読み込み */

        perspect(&input, &output, atof(argv[3]), atof(argv[4]),
                        atof(argv[5]), atof(argv[6]), atof(argv[7]),
                        atof(argv[8]), atof(argv[9]), atof(argv[10]),
                        atof(argv[11]), atof(argv[12]));     /* 透視変換処理 */

        if(writergb(argv[2], &output)<0) return -3;   /* ファイル書き込み */

        return 0;
}


perspect(input, output, ax, ay, px, py, pz, rz, rx, ry, v, s)
/* 透視変換 (線形補間法)  */
```



```
struct img *input;              /* 入力画像配列 */
struct img *output;     /* 出力画像配列 */
float ax;                               /* 拡大率 (横)  */
float ay;                               /* 拡大率 (縦)  */
float px;                               /* 移動量 ( x )  */
float py;                               /* 移動量 ( y )  */
float pz;                               /* 移動量 ( z )  */
float rz;                               /* 回転角 ( z 度) */
float rx;                               /* 回転角 ( x 度) */
float ry;                               /* 回転角 ( y 度) */
float v;                                /* 視点の位置 ( z ) */
float s;                                /* スクリーンの位置 ( z ) */
{
        int         i, j, m, n;
        float   x, y, w, p, q;
        float   k[9];
        int         xs, ys;
        int     r, g, b, xsize;

        xs = input->xsize/2;  ys = input->ysize/2;
        xsize = input->xsize;

        output->rdata = (char *)malloc(input->xsize*input->ysize);
        if(input->type==COLOR){
          output->gdata = (char *)malloc(input->xsize*input->ysize);
          output->bdata = (char *)malloc(input->xsize*input->ysize);
        }
        output->type  = input->type;
        output->xsize = input->xsize;
        output->ysize = input->ysize;
        output->depth = input->depth;

        param_pers(k,ax,ay,px,py,pz,rz,rx,ry,v,s,xs,ys); /* 変換パラメータ決定 */
        for (i = -ys; i < ys; i++) {
          for (j = -xs; j < xs; j++) {
            w = k[0]*j + k[1]*i + k[2];
            x = k[3]*j + k[4]*i + k[5];
            y = k[6]*j + k[7]*i + k[8];
            x = x/w;
            y = y/w;
            if (y > 0) m = y; else m = y-1;
            if (x > 0) n = x; else n = x-1;
            q = y - m;
            p = x - n;

            if((m>=-ys)&&(m<ys)&&(n>=-xs)&&(n<xs)){
              r=(1.0-q)*((1.0-p)*input->rdata[(m+ys)*xsize + n+xs]
                        +p*input->rdata[(m+ys)*xsize + n+1+xs])
                +q*((1-p)*input->rdata[(m+1+ys)*xsize + n+xs]
                    +p*input->rdata[(m+1+ys)*xsize + n+1+xs]);
              if(input->type==COLOR){
                        g=(1.0-q)*((1.0-p)*input->gdata[(m+ys)*xsize + n+xs]
                                          +p*input->gdata[(m+ys)*xsize + n+1+xs])
                          +q*((1-p)*input->gdata[(m+1+ys)*xsize + n+xs]
                                    +p*input->gdata[(m+1+ys)*xsize + n+1+xs]);
                        b=(1.0-q)*((1.0-p)*input->bdata[(m+ys)*xsize + n+xs]
                                          +p*input->bdata[(m+ys)*xsize + n+1+xs])
                          +q*((1-p)*input->bdata[(m+1+ys)*xsize + n+xs]
                                    +p*input->bdata[(m+1+ys)*xsize + n+1+xs]);
              }
            } else r=g=b=0;

            if(r<0) r=0;  if(r>255) r=255;
            output->rdata[ (i+ys)*xsize + j+xs ] = r;

            if(input->type==COLOR){
```

```
            if(g<0) g=0;  if(g>255) g=255;
            if(b<0) b=0;  if(b>255) b=255;
            output->gdata[ (i+ys)*xsize + j+xs ] = g;
            output->bdata[ (i+ys)*xsize + j+xs ] = b;
          }
        }
      }
}

param_pers(k, a, b, x0, y0, z0, z, x, y, t, s, xs, ys)
/* 透視変換のパラメータ計算 */
float k[9];             /* 変換パラメータ                  */
float a;                /* 拡大率（x方向）                 */
float b;                /* 拡大率（y方向）                 */
float x0;               /* 移動量（x方向）                 */
float y0;               /* 移動量（y方向）                 */
float z0;               /* 移動量（z方向）                 */
float z;                /* 回転角（z方向，度）             */
float x;                /* 回転角（x方向，度）             */
float y;                /* 回転角（y方向，度）             */
float t;                /* 視点の位置（z方向）             */
float s;                /* スクリーンの位置（z方向）       */
int xs;
int ys;
{
        float   l[4][4],m[4][4],n[4][4],k1,k2,k3,k4,k5,k6,k7,k8,k9;
        double  u,v,w;
        int     i;

        u=x*3.141592/180.0;     v=y*3.141592/180.0;     w=z*3.141592/180.0;
        l[0][0]= 1.0/xs;  l[0][1]= 0;       l[0][2]= 0;         l[0][3]= 0;
        l[1][0]= 0;       l[1][1]= -1.0/xs; l[1][2]= 0;         l[1][3]= 0;
        l[2][0]= 0;       l[2][1]= 0;       l[2][2]= 1;         l[2][3]= 0;
        l[3][0]= 0;       l[3][1]= 0;       l[3][2]= 0;         l[3][3]= 1;
        m[0][0]= a;       m[0][1]= 0;       m[0][2]= 0;         m[0][3]= 0;
        m[1][0]= 0;       m[1][1]= b;       m[1][2]= 0;         m[1][3]= 0;
        m[2][0]= 0;       m[2][1]= 0;       m[2][2]= 1;         m[2][3]= 0;
        m[3][0]= 0;       m[3][1]= 0;       m[3][2]= 0;         m[3][3]= 1;
        matrix(l,m,n);  /* 正規化マトリックス × 拡大縮小マトリックス */
        l[0][0]= 1;       l[0][1]= 0;       l[0][2]= 0;         l[0][3]= 0;
        l[1][0]= 0;       l[1][1]= 1;       l[1][2]= 0;         l[1][3]= 0;
        l[2][0]= 0;       l[2][1]= 0;       l[2][2]= 1;         l[2][3]= 0;
        l[3][0]= x0;      l[3][1]= y0;      l[3][2]= z0;        l[3][3]= 1;
        matrix(n,l,m);  /* × 移動マトリックス */
        n[0][0]=  cos(w); n[0][1]= sin(w);  n[0][2]= 0;         n[0][3]= 0;
        n[1][0]= -sin(w); n[1][1]= cos(w);  n[1][2]= 0;         n[1][3]= 0;
        n[2][0]= 0;       n[2][1]= 0;       n[2][2]= 1;         n[2][3]= 0;
        n[3][0]= 0;       n[3][1]= 0;       n[3][2]= 0;         n[3][3]= 1;
        matrix(m,n,l);  /* × z軸の回転マトリックス */
        m[0][0]= 1;       m[0][1]= 0;       m[0][2]= 0;         m[0][3]= 0;
        m[1][0]= 0;       m[1][1]= cos(u);  m[1][2]= sin(u);    m[1][3]= 0;
        m[2][0]= 0;       m[2][1]= -sin(u); m[2][2]= cos(u);    m[2][3]= 0;
        m[3][0]= 0;       m[3][1]= 0;       m[3][2]= 0;         m[3][3]= 1;
        matrix(l,m,n);  /* × x軸の回転マトリックス */
        l[0][0]= cos(v);  l[0][1]= 0;       l[0][2]= sin(v);    l[0][3]= 0;
        l[1][0]= 0;       l[1][1]= 1;       l[1][2]= 0;         l[1][3]= 0;
        l[2][0]= -sin(v); l[2][1]= 0;       l[2][2]= cos(v);    l[2][3]= 0;
        l[3][0]= 0;       l[3][1]= 0;       l[3][2]= 0;         l[3][3]= 1;
        matrix(n,l,m);  /* × y軸の回転マトリックス */
        n[0][0]= 1;       n[0][1]= 0;       n[0][2]= 0;         n[0][3]= 0;
        n[1][0]= 0;       n[1][1]= 1;       n[1][2]= 0;         n[1][3]= 0;
        n[2][0]= 0;       n[2][1]= 0;       n[2][2]= -1;        n[2][3]= 0;
        n[3][0]= 0;       n[3][1]= 0;       n[3][2]= t;         n[3][3]= 1;
        matrix(m,n,l);  /* × 視点座標変換マトリックス */
        m[0][0]= 1;       m[0][1]= 0;       m[0][2]= 0;         m[0][3]= 0;
        m[1][0]= 0;       m[1][1]= 1;       m[1][2]= 0;         m[1][3]= 0;
```



```
        m[2][0]= 0;       m[2][1]= 0;       m[2][2]= 1/s;       m[2][3]= 1/s;
        m[3][0]= 0;       m[3][1]= 0;       m[3][2]= -1;        m[3][3]= 0;
        matrix(l,m,n);  /* × 透視変換マトリックス */
        l[0][0]= xs;      l[0][1]= 0;       l[0][2]= 0;         l[0][3]= 0;
        l[1][0]= 0;       l[1][1]= -xs;     l[1][2]= 0;         l[1][3]= 0;
        l[2][0]= 0;       l[2][1]= 0;       l[2][2]= 1;         l[2][3]= 0;
        l[3][0]= 0;       l[3][1]= 0;       l[3][2]= 0;         l[3][3]= 1;
        matrix(n,l,m);  /* × 正規化逆マトリックス */
        k1=m[0][3];     k2=m[1][3];     k3=m[3][3];
        k4=m[0][0];     k5=m[1][0];     k6=m[3][0];
        k7=m[0][1];     k8=m[1][1];     k9=m[3][1];
        k[0]=k7*k2-k8*k1; k[1]=k5*k1-k4*k2; k[2]=k4*k8-k7*k5;
        k[3]=k8*k3-k9*k2; k[6]=k9*k1-k7*k3; k[4]=k6*k2-k5*k3;
        k[7]=k4*k3-k6*k1; k[5]=k5*k9-k8*k6; k[8]=k7*k6-k4*k9;
/*      これ以下のコメントは必要に応じて生かして下さい.
        変換マトリックスを印字します.
        printf("paremeters\n");
        printf(" p= %f\n",k[0]);
        printf(" q= %f\n",k[1]);
        printf(" r= %f\n",k[2]);
        printf(" a= %f\n",k[3]);
        printf(" b= %f\n",k[4]);
        printf(" c= %f\n",k[5]);
        printf(" d= %f\n",k[6]);
        printf(" e= %f\n",k[7]);
        printf(" f= %f\n",k[8]);
        コメント終わり
*/
}

matrix(l, m, n)
/* マトリックス計算 */
float l[4][4];  /* 入力マトリックス1 */
float m[4][4];  /* 入力マトリックス2 */
float n[4][4];  /* 出力マトリックス  */
{
        int     i, j, k;
        float   p;

        for (i = 0; i < 4; i++) {
          for (j = 0; j < 4; j++) {
            p = 0;
            for (k = 0; k < 4; k++) p = p + l[i][k]*m[k][j];
            n[i][j] = p;
          }
        }
}


/*
   IRIS rgb形式のファイルをimg構造体に読み込む

     返り値  読み込み成功   0
                 失敗  -1
*/
readrgb(filename, data)
char *filename;         /* ファイル名 */
struct img *data;       /* 画像を格納したい構造体 */
{
  IMAGE *image;
  short r[8192], g[8192], b[8192];
  int size, x, y;

  if( (image=iopen(filename,"r"))==NULL ){
    fprintf(stderr, "cannot open file -%s\n",filename);
    return -1;
```

```
  }

  data->xsize = image->xsize;
  data->ysize = image->ysize;

  if(image->zsize==1){
    data->type = GREY;
  } else if(image->zsize==3 || image->zsize==4){
    data->type = COLOR;
  } else {
    fprintf(stderr, "unknown file type\n");
    return -1;
  }
  data->depth = 255;

  size = data->xsize*data->ysize;
  data->rdata = (unsigned char *)malloc(size);
  if(data->type==COLOR){
    data->gdata = (unsigned char *)malloc(size);
    data->bdata = (unsigned char *)malloc(size);
  }

  for(y=0;y<data->ysize;y++){
    getrow(image,r,y,0);
    if(data->type==COLOR){
      getrow(image,g,y,1);
      getrow(image,b,y,2);
    }
    for(x=0;x<data->xsize;x++){
      data->rdata[x+data->xsize*(data->ysize-1-y)] = r[x];
      if(data->type==COLOR){
                data->gdata[x+data->xsize*(data->ysize-1-y)] = g[x];
                data->bdata[x+data->xsize*(data->ysize-1-y)] = b[x];
      }
    }
  }
  iclose(image);
  return 0;
}


/*
   img構造体の中身ををIRIS rgb形式のファイルに書き出す

       返り値   書き込み成功   0
                失敗   -1
*/
writergb(filename, data)
char *filename;                 /* ファイル名 */
struct img *data;               /* 書き込みたい画像の入った構造体 */
{
  IMAGE *image;
  int y,x;
  unsigned short  r[8192], g[8192], b[8192];

  if(data->type==COLOR)
    image = iopen( filename, "w", RLE(1), 3, data->xsize, data->ysize, 3);
  else
    image = iopen( filename, "w", RLE(1), 2, data->xsize, data->ysize, 1);

  if(image==NULL){
        return -1;
  }

  for(y=0;y<data->ysize;y++){
    for(x=0;x<data->xsize;x++){
```



```
      r[x]=data->rdata[x + data->xsize*(data->ysize-y-1)];
      if(data->type==COLOR){
                g[x]=data->gdata[x + data->xsize*(data->ysize-y-1)];
                b[x]=data->bdata[x + data->xsize*(data->ysize-y-1)];
      }
    }
    putrow(image, r, y, 0);
    if(data->type==COLOR){
      putrow(image, g, y, 1);
      putrow(image, b, y, 2);
    }
  }
  iclose(image);
  return 0;
}
```

```
/*

   輝度がthreshold未満の画素のr,g,b値を変えるプログラム

        第１引数：入力ファイル
            第２引数：出力ファイル
            第３引数：割合
            第４引数：閾値

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "image.h"

#define COLOR 0
#define GREY 1

struct img {
        unsigned char type;
        unsigned int  xsize, ysize, depth;
        unsigned char *rdata;
        unsigned char *gdata;
        unsigned char *bdata;
};

int readrgb(char *, struct img *);
int writergb(char *, struct img *);
int theshold(struct img *, struct img *, int, float);

main(argc, argv)
int argc;
char *argv[];
{
        struct img input, output;

        if(argc!=5){
          printf("usage: %s infile outfile ratio threshold\n\n", argv[0]);
          return -1;
        }

        if(readrgb(argv[1], &input)<0){puts("error"); return -2;}
        threshold(&input, &output, atoi(argv[4]), atof(argv[3]));
        writergb(argv[2], &output);
}

int threshold(input1, output, thsh, rt)
struct img *input1;
struct img *output;
int thsh;
float rt;
{
        int i, size,  d1;
        float r, g, b;

        size = input1->xsize*input1->ysize;
        output->rdata = (char *)malloc( size );
        if(input1->type==COLOR){
                output->gdata = (char *)malloc( size );
                output->bdata = (char *)malloc( size );
        }
        output->type  = input1->type;
        output->xsize = input1->xsize;
        output->ysize = input1->ysize;
        output->depth = input1->depth;
```



```
        for(i=0;i<size;i++){
          d1 = input1->rdata[i];
          if(input1->type==COLOR){
            d1 = d1 + input1->gdata[i] + input1->bdata[i];
            d1 = d1/3.;
          }

          output->rdata[i]=input1->rdata[i];
          if(input1->type==COLOR){
            output->gdata[i]=input1->gdata[i];
            output->bdata[i]=input1->bdata[i];
          }

          if(d1<=thsh){
            r = (float)output->rdata[i]*rt;
            if(r>255.) output->rdata[i]=255;
            else if(r<0.) output->rdata[i]=0;
            else output->rdata[i]=(unsigned char)r;
            if(input1->type==COLOR){
              g = (float)output->gdata[i]*rt;
              b = (float)output->bdata[i]*rt;
              if(g>255.) output->gdata[i]=255;
                  else if(g<0.) output->gdata[i]=0;
              else output->gdata[i]=(unsigned char)g;
              if(b>255.) output->bdata[i]=255;
              else if(b<0.) output->bdata[i]=0;
                  else output->bdata[i]=(unsigned char)b;
            }

          }
        }
        return 0;
}


readrgb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  short r[8192], g[8192], b[8192];
  int size, x, y;

  if( (image=iopen(filename,"r"))==NULL ){
    fprintf(stderr, "cannot open file -%s\n",filename);
    return -1;
  }

  data->xsize = image->xsize;
  data->ysize = image->ysize;

  if(image->zsize==1){
    data->type = GREY;
  } else if(image->zsize==3 || image->zsize==4){
    data->type = COLOR;
  } else {
    fprintf(stderr, "unknown file type\n");
    return -2;
  }
  data->depth = 255;

  size = data->xsize*data->ysize;
  data->rdata = (unsigned char *)malloc(size);
  if(data->type==COLOR){
    data->gdata = (unsigned char *)malloc(size);
```

```
    data->bdata = (unsigned char *)malloc(size);
  }

  for(y=0;y<data->ysize;y++){
    getrow(image,r,y,0);
    if(data->type==COLOR){
      getrow(image,g,y,1);
      getrow(image,b,y,2);
    }
    for(x=0;x<data->xsize;x++){
      data->rdata[x+data->xsize*(data->ysize-1-y)] = r[x];
      if(data->type==COLOR){
        data->gdata[x+data->xsize*(data->ysize-1-y)] = g[x];
        data->bdata[x+data->xsize*(data->ysize-1-y)] = b[x];
      }
    }
  }
  iclose(image);
  return 0;
}

writergb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  int y,x;
  unsigned short  r[8192], g[8192], b[8192];

  if(data->type==COLOR)
    image = iopen( filename, "w", RLE(1), 3, data->xsize, data->ysize, 3);
  else
    image = iopen( filename, "w", RLE(1), 2, data->xsize, data->ysize, 1);

  for(y=0;y<data->ysize;y++){
    for(x=0;x<data->xsize;x++){
      r[x]=data->rdata[x + data->xsize*(data->ysize-y-1)];
      if(data->type==COLOR){
        g[x]=data->gdata[x + data->xsize*(data->ysize-y-1)];
        b[x]=data->bdata[x + data->xsize*(data->ysize-y-1)];
      }
    }
    putrow(image, r, y, 0);
    if(data->type==COLOR){
      putrow(image, g, y, 1);
      putrow(image, b, y, 2);
    }
  }
  iclose(image);
  return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "image.h"

#define COLOR 0
#define GREY 1
#define LOW 0
#define HIGH 255

struct img {
        unsigned char type;
        unsigned int  xsize, ysize, depth;
        unsigned char *rdata;
        unsigned char *gdata;
        unsigned char *bdata;
};

main(argc, argv)
int argc;
char *argv[];
{
        struct img input, input2, output;

        if(argc!=6){
          printf("Usage: %s infile outfile threfile thl thh\n\n", argv[0]);
          return -1;
        }

        if(readrgb(argv[1], &input)<0){puts("error"); return -2;}
        if(readrgb(argv[3], &input2)<0){puts("error"); return -2;}

        threshold(&input, &output, &input2, argv[4], argv[5]);
        writergb(argv[2], &output);
}


int threshold(input1, output, input2, th, th2)
struct img *input1;
struct img *output;
struct img *input2;
char *th;
char *th2;
{
        int i, size,  d1, thsh, thsh2, kk;
        float rt, r;

        thsh = atoi( th );
        thsh2 = atoi( th2 );

        size = input1->xsize*input1->ysize;
        output->rdata = (char *)malloc( size );
        output->gdata = (char *)malloc( size );
        output->bdata = (char *)malloc( size );

        output->type  = input1->type;
        output->xsize = input1->xsize;
        output->ysize = input1->ysize;
        output->depth = input1->depth;

        for(i=0;i<size;i++){
          d1 =  input1->bdata[i];
          kk = (d1-thsh)/(thsh2-thsh);
          if(kk>HIGH){
                output->rdata[i]=input1->rdata[i];
                output->gdata[i]=input1->gdata[i];
```



```
                output->bdata[i]=input1->bdata[i];
          }else if(kk<LOW){
                output->rdata[i]=input2->rdata[i];
                output->gdata[i]=input2->gdata[i];
                output->bdata[i]=input2->bdata[i];
          }else{
                output->rdata[i]=(input1->gdata[i]*kk+input2->rdata[i]*(HIGH-kk))/(HI
GH-LOW);
                output->gdata[i]=(input1->gdata[i]*kk+input2->gdata[i]*(HIGH-kk))/(HI
GH-LOW);
                output->bdata[i]=(input1->gdata[i]*kk+input2->bdata[i]*(HIGH-kk))/(HI
GH-LOW);
          }
        }
        return 0;
}


readrgb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  short r[8192], g[8192], b[8192];
  int size, x, y;

  if( (image=iopen(filename,"r"))==NULL ){
    fprintf(stderr, "cannot open file -%s\n",filename);
    return -1;
  }

  data->xsize = image->xsize;
  data->ysize = image->ysize;

  if(image->zsize==1){
    data->type = GREY;
  } else if(image->zsize==3 || image->zsize==4){
    data->type = COLOR;
  } else {
    fprintf(stderr, "unknown file type\n");
    return -2;
  }
  data->depth = 255;

  size = data->xsize*data->ysize;
  data->rdata = (unsigned char *)malloc(size);
  if(data->type==COLOR){
    data->gdata = (unsigned char *)malloc(size);
    data->bdata = (unsigned char *)malloc(size);
  }

  for(y=0;y<data->ysize;y++){
    getrow(image,r,y,0);
    if(data->type==COLOR){
      getrow(image,g,y,1);
      getrow(image,b,y,2);
    }
    for(x=0;x<data->xsize;x++){
      data->rdata[x+data->xsize*(data->ysize-1-y)] = r[x];
      if(data->type==COLOR){
                data->gdata[x+data->xsize*(data->ysize-1-y)] = g[x];
                data->bdata[x+data->xsize*(data->ysize-1-y)] = b[x];
      }
    }
  }
  iclose(image);
```

```
  return 0;
}

writergb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  int y,x;
  unsigned short  r[8192], g[8192], b[8192];

  if(data->type==COLOR)
    image = iopen( filename, "w", RLE(1), 3, data->xsize, data->ysize, 3);
  else
    image = iopen( filename, "w", RLE(1), 2, data->xsize, data->ysize, 1);

  for(y=0;y<data->ysize;y++){
    for(x=0;x<data->xsize;x++){
      r[x]=data->rdata[x + data->xsize*(data->ysize-y-1)];
      if(data->type==COLOR){
                g[x]=data->gdata[x + data->xsize*(data->ysize-y-1)];
                b[x]=data->bdata[x + data->xsize*(data->ysize-y-1)];
      }
    }
    putrow(image, r, y, 0);
    if(data->type==COLOR){
      putrow(image, g, y, 1);
      putrow(image, b, y, 2);
    }
  }
  iclose(image);
  return 0;
}
```

```
/*

    拡大縮小プログラム

    第１引数:入力ファイル(IRIS rgb)
    第２引数:出力ファイル(IRIS rgb)
    第３引数:拡大率(横)
    第４引数:拡大率(縦)

    (入力ファイルと出力ファイルで同じ名前を指定することも可能)

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "image.h"

#define COLOR 0
#define GREY 1

struct img {
        char type;
        unsigned int xsize, ysize, depth;
        unsigned char *rdata;
        unsigned char *gdata;
        unsigned char *bdata;
};

int readrgb(char *, struct img *);
int writergb(char *, struct img *);
int zoom(struct img *, struct img *, char *, char *);

main(argc, argv)
int argc;
char *argv[];
{
        struct img input, output;

        if(argc!=5){
          fprintf(stderr, "Usage: %s infile outfile zx zy\n\n", argv[0]);
          return -1;
        }

        if(readrgb(argv[1], &input)<0) return -2;
        zoom(&input, &output, argv[3], argv[4]);
        writergb(argv[2], &output);
}


int zoom(input, output, xr, yr)
struct img *input;
struct img *output;
char *xr;
char *yr;
{
        float zx, zy;
        float x, y, p, q;
        int i, j, m, n;
        int xs, ys, r, g, b, xsize;

        zx = atof(xr);  zy = atof(yr);
        xs = input->xsize/2;  ys = input->ysize/2;
        xsize = input->xsize;

        output->rdata = (char *)malloc(input->xsize*input->ysize);
```



```
        if(input->type==COLOR){
                output->gdata = (char *)malloc(input->xsize*input->ysize);
                output->bdata = (char *)malloc(input->xsize*input->ysize);
        }

        for(i=-ys; i<ys; i++){
                for(j=-xs; j<xs; j++){
                        x = (float)j/zx;  y = (float)i/zy;
                        if(x>0) n=x; else n=x-1;
                        if(y>0) m=y; else m=y-1;
                        p = x-n;  q = y-m;
                        if((m>=-ys)&&(m<ys)&&(n>=-xs)&&(n<xs)){
                          r=(1.0-q)*((1.0-p)*input->rdata[(m+ys)*xsize+n+xs]
                               +p*input->rdata[(m+ys)*xsize+n+1+xs])
                               +q*((1-p)*input->rdata[(m+1+ys)*xsize+n+xs]
                               +p*input->rdata[(m+1+ys)*xsize+n+1+xs]);
                          if(input->type==COLOR){
                                g=(1.0-q)*((1.0-p)*input->gdata[(m+ys)*xsize+n+xs]
                                                  +p*input->gdata[(m+ys)*xsize+n+1+x
s])
                                +q*((1-p)*input->gdata[(m+1+ys)*xsize+n+xs]
                                                  +p*input->gdata[(m+1+ys)*xsize+n+1+xs]);
                                b=(1.0-q)*((1.0-p)*input->bdata[(m+ys)*xsize+n+xs]
                                                  +p*input->bdata[(m+ys)*xsize+n+1+x
s])
                                +q*((1-p)*input->bdata[(m+1+ys)*xsize+n+xs]
                                                  +p*input->bdata[(m+1+ys)*xsize+n+1+xs]);
                          }
                        } else r=g=b=0;

                        if(r<0) r=0;  if(r>255) r=255;
                        if(g<0) g=0;  if(g>255) g=255;
                        if(b<0) b=0;  if(b>255) b=255;

                        output->rdata[ (i+ys)*xsize + j+xs ] = r;
                        if(input->type==COLOR){
                                output->gdata[ (i+ys)*xsize + j+xs ] = g;
                                output->bdata[ (i+ys)*xsize + j+xs ] = b;
                        }
                }
        }
        output->type = input->type;
        output->xsize = input->xsize;
        output->ysize = input->ysize;
        output->depth = input->depth;

        return 0;
}


readrgb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  short r[8192], g[8192], b[8192];
  int size, x, y;

  if( (image=iopen(filename,"r"))==NULL ){
    fprintf(stderr, "cannot open file -%s\n",filename);
    return -1;
  }

  data->xsize = image->xsize;
  data->ysize = image->ysize;
```

```
  if(image->zsize==1){
    data->type = GREY;
  } else if(image->zsize==3 || image->zsize==4){
    data->type = COLOR;
  } else {
    fprintf(stderr, "unknown file type\n");
    return -2;
  }
  data->depth = 255;

  size = data->xsize*data->ysize;
  data->rdata = (unsigned char *)malloc(size);
  if(data->type==COLOR){
    data->gdata = (unsigned char *)malloc(size);
    data->bdata = (unsigned char *)malloc(size);
  }

  for(y=0;y<data->ysize;y++){
    getrow(image,r,y,0);
    if(data->type==COLOR){
      getrow(image,g,y,1);
      getrow(image,b,y,2);
    }
    for(x=0;x<data->xsize;x++){
      data->rdata[x+data->xsize*(data->ysize-1-y)] = r[x];
      if(data->type==COLOR){
        data->gdata[x+data->xsize*(data->ysize-1-y)] = g[x];
        data->bdata[x+data->xsize*(data->ysize-1-y)] = b[x];
      }
    }
  }
  iclose(image);
  return 0;
}

writergb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  int y,x;
  unsigned short  r[8192], g[8192], b[8192];

  if(data->type==COLOR)
    image = iopen( filename, "w", RLE(1), 3, data->xsize, data->ysize, 3);
  else
    image = iopen( filename, "w", RLE(1), 2, data->xsize, data->ysize, 1);

  for(y=0;y<data->ysize;y++){
    for(x=0;x<data->xsize;x++){
      r[x]=data->rdata[x + data->xsize*(data->ysize-y-1)];
      if(data->type==COLOR){
        g[x]=data->gdata[x + data->xsize*(data->ysize-y-1)];
        b[x]=data->bdata[x + data->xsize*(data->ysize-y-1)];
      }
    }
    putrow(image, r, y, 0);
    if(data->type==COLOR){
      putrow(image, g, y, 1);
      putrow(image, b, y, 2);
    }
  }
  iclose(image);
  return 0;
}
```

```
/*

   拡大縮小プログラム

   第１引数：入力ファイル(IRIS rgb)
   第２引数：出力ファイル(IRIS rgb)
   第３引数：拡大率(横)
   第４引数：拡大率(縦)

   (入力ファイルと出力ファイルで同じ名前を指定することも可能)

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "image.h"

#define COLOR 0
#define GREY 1

struct img {
        char type;
        unsigned int xsize, ysize, depth;
        unsigned char *rdata;
        unsigned char *gdata;
        unsigned char *bdata;
};

int readrgb(char *, struct img *);
int writergb(char *, struct img *);
int zoom(struct img *, struct img *, char *, char *);


main(argc, argv)
int argc;
char *argv[];
{
        struct img input, output;

        if(argc!=5){
          fprintf(stderr, "Usage: %s infile outfile zx zy\n\n", argv[0]);
          return -1;
        }

        if(readrgb(argv[1], &input)<0) return -2;
        zoom(&input, &output, argv[3], argv[4]);
        writergb(argv[2], &output);
}


int zoom(input, output, xr, yr)
struct img *input;
struct img *output;
char *xr;
char *yr;
{
        float zx, zy;
        float x, y, p, q;
        int i, j, m, n;
        int xs, ys, r, g, b, xsize, size;
        int ox, oy;

        zx = atof(xr);  zy = atof(yr);
        xs = input->xsize/2;  ys = input->ysize/2;
        xsize = input->xsize;
```



```
        output->type = input->type;
        output->xsize = (int)((float)input->xsize*zx);
        output->ysize = (int)((float)input->ysize*zy);
        output->depth = input->depth;

        size = output->xsize * output->ysize;
        output->rdata = (char *)malloc( size );
        if(input->type==COLOR){
          output->gdata = (char *)malloc( size );
          output->bdata = (char *)malloc( size );
        }

        ox = output->xsize/2;  oy = output->ysize/2;

        for(i=-oy; i<oy; i++){
          for(j=-ox; j<ox; j++){
                x = (float)j/zx;  y = (float)i/zy;
                if(x>0) n=x; else n=x-1;
                if(y>0) m=y; else m=y-1;
                p = x-n;  q = y-m;
                if((m>=-ys)&&(m<ys)&&(n>=-xs)&&(n<xs)){
                  r=(1.0-q)*((1.0-p)*input->rdata[(m+ys)*xsize+n+xs]
                                    +p*input->rdata[(m+ys)*xsize+n+1+xs])
                        +q*((1-p)*input->rdata[(m+1+ys)*xsize+n+xs]
                                +p*input->rdata[(m+1+ys)*xsize+n+1+xs]);
                  if(input->type==COLOR){
                        g=(1.0-q)*((1.0-p)*input->gdata[(m+ys)*xsize+n+xs]
                                          +p*input->gdata[(m+ys)*xsize+n+1+xs])
                          +q*((1-p)*input->gdata[(m+1+ys)*xsize+n+xs]
                                  +p*input->gdata[(m+1+ys)*xsize+n+1+xs]);
                        b=(1.0-q)*((1.0-p)*input->bdata[(m+ys)*xsize+n+xs]
                                          +p*input->bdata[(m+ys)*xsize+n+1+xs])
                          +q*((1-p)*input->bdata[(m+1+ys)*xsize+n+xs]
                                  +p*input->bdata[(m+1+ys)*xsize+n+1+xs]);
                  }

                } else r=g=b=0;
                if(r<0) r=0;
                if(r>255) r=255;
                output->rdata[ (i+oy)*output->xsize + j+ox ] = r;

                if(input->type==COLOR){
                  if(g<0) g=0;  if(g>255) g=255;
                  if(b<0) b=0;  if(b>255) b=255;
                  output->gdata[ (i+oy)*output->xsize + j+ox ] = g;
                  output->bdata[ (i+oy)*output->xsize + j+ox ] = b;
                }
          }
        }
        return 0;
}



readrgb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  short r[8192], g[8192], b[8192];
  int size, x, y;

  if( (image=iopen(filename,"r"))==NULL ){
    fprintf(stderr, "cannot open file -%s\n",filename);
    return -1;
```

```
  }

  data->xsize = image->xsize;
  data->ysize = image->ysize;

  if(image->zsize==1){
    data->type = GREY;
  } else if(image->zsize==3 || image->zsize==4){
    data->type = COLOR;
  } else {
    fprintf(stderr, "unknown file type\n");
    return -2;
  }
  data->depth = 255;

  size = data->xsize*data->ysize;
  data->rdata = (unsigned char *)malloc(size);
  if(data->type==COLOR){
    data->gdata = (unsigned char *)malloc(size);
    data->bdata = (unsigned char *)malloc(size);
  }

  for(y=0;y<data->ysize;y++){
    getrow(image,r,y,0);
    if(data->type==COLOR){
      getrow(image,g,y,1);
      getrow(image,b,y,2);
    }
    for(x=0;x<data->xsize;x++){
      data->rdata[x+data->xsize*(data->ysize-1-y)] = r[x];
      if(data->type==COLOR){
                data->gdata[x+data->xsize*(data->ysize-1-y)] = g[x];
                data->bdata[x+data->xsize*(data->ysize-1-y)] = b[x];
      }
    }
  }
  iclose(image);
  return 0;
}

writergb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  int y,x;
  unsigned short  r[8192], g[8192], b[8192];

  if(data->type==COLOR)
    image = iopen( filename, "w", RLE(1), 3, data->xsize, data->ysize, 3);
  else
    image = iopen( filename, "w", RLE(1), 2, data->xsize, data->ysize, 1);

  for(y=0;y<data->ysize;y++){
    for(x=0;x<data->xsize;x++){
      r[x]=data->rdata[x + data->xsize*(data->ysize-y-1)];
      if(data->type==COLOR){
                g[x]=data->gdata[x + data->xsize*(data->ysize-y-1)];
                b[x]=data->bdata[x + data->xsize*(data->ysize-y-1)];
      }
    }
    putrow(image, r, y, 0);
    if(data->type==COLOR){
      putrow(image, g, y, 1);
      putrow(image, b, y, 2);
    }
```



```
  }
  iclose(image);
  return 0;
}
```

```
/*

   拡大縮小プログラム

   第１引数:入力ファイル(IRIS rgb)
   第２引数:出力ファイル(IRIS rgb)
   第３引数:拡大率(横)
   第４引数:拡大率(縦)

   (入力ファイルと出力ファイルで同じ名前を指定することも可能)

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "image.h"

#define COLOR 0
#define GREY 1

struct img {
        char type;
        unsigned int xsize, ysize, depth;
        unsigned char *rdata;
        unsigned char *gdata;
        unsigned char *bdata;
};


main(argc, argv)
int argc;
char *argv[];
{
        struct img input, output;

        if(argc!=8){
          fprintf(stderr, "Usage: %s infile outfile zx zy deg mx my\n\n", argv[0]);
          return -1;
        }

        if(readrgb(argv[1], &input)<0) return -2;
        affine(&input, &output, atof(argv[3]),
                atof(argv[4]), atof(argv[5]), atof(argv[6]), atof(argv[7]));
        if(writergb(argv[2], &output)<0) return -3;

        return 0;
}

int affine(input, output, zx, zy, deg, mx, my)
struct img *input;
struct img *output;
float zx;
float zy;
float deg;
float mx;
float my;
{


        float x, y, p, q, c, s, u, v;
        double rr;
        int i, j, m, n;
        int xs, ys, r, g, b, xsize;
```



```
        rr = deg*3.141592/180.;
        c = cos(rr);
        s = sin(rr);

        xs = input->xsize/2;  ys = input->ysize/2;
        xsize = input->xsize;

        output->type = input->type;
        output->xsize = input->xsize;
        output->ysize = input->ysize;
        output->depth = input->depth;

        output->rdata = (char *)malloc(input->xsize*input->ysize);
        if(input->type==COLOR){
          output->gdata = (char *)malloc(input->xsize*input->ysize);
          output->bdata = (char *)malloc(input->xsize*input->ysize);
        }

        for(i=-ys; i<ys; i++){
          for(j=-xs; j<xs; j++){
                u = j - mx;   v = i - my;
                x = (u*c-v*s)/zx;   y = (u*s+v*c)/zy;
                if(x>0) n=x; else n=x-1;
                if(y>0) m=y; else m=y-1;
                p = x-n;  q = y-m;
                if((m>=-ys)&&(m<ys)&&(n>=-xs)&&(n<xs)){
                  r=(1.0-q)*((1.0-p)*input->rdata[(m+ys)*xsize+n+xs]
                                        +p*input->rdata[(m+ys)*xsize+n+1+xs])
                        +q*((1-p)*input->rdata[(m+1+ys)*xsize+n+xs]
                                +p*input->rdata[(m+1+ys)*xsize+n+1+xs]);
                  if(input->type==COLOR){
                        g=(1.0-q)*((1.0-p)*input->gdata[(m+ys)*xsize+n+xs]
                                        +p*input->gdata[(m+ys)*xsize+n+1+xs])
                          +q*((1-p)*input->gdata[(m+1+ys)*xsize+n+xs]
                                  +p*input->gdata[(m+1+ys)*xsize+n+1+xs]);
                        b=(1.0-q)*((1.0-p)*input->bdata[(m+ys)*xsize+n+xs]
                                        +p*input->bdata[(m+ys)*xsize+n+1+xs])
                          +q*((1-p)*input->bdata[(m+1+ys)*xsize+n+xs]
                                  +p*input->bdata[(m+1+ys)*xsize+n+1+xs]);
                  }
                } else r=g=b=0;

                if(r<0) r=0;  if(r>255) r=255;
                if(g<0) g=0;  if(g>255) g=255;
                if(b<0) b=0;  if(b>255) b=255;

                output->rdata[ (i+ys)*xsize + j+xs ] = r;
                if(input->type==COLOR){
                  output->gdata[ (i+ys)*xsize + j+xs ] = g;
                  output->bdata[ (i+ys)*xsize + j+xs ] = b;
                }
          }
        }

        return 0;
}


readrgb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  short r[8192], g[8192], b[8192];
  int size, x, y;
```

```
  if( (image=iopen(filename,"r"))==NULL ){
    fprintf(stderr, "cannot open file -%s\n",filename);
    return -1;
  }

  data->xsize = image->xsize;
  data->ysize = image->ysize;

  if(image->zsize==1){
    data->type = GREY;
  } else if(image->zsize==3 || image->zsize==4){
    data->type = COLOR;
  } else {
    fprintf(stderr, "unknown file type\n");
    return -2;
  }
  data->depth = 255;

  size = data->xsize*data->ysize;
  data->rdata = (unsigned char *)malloc(size);
  if(data->type==COLOR){
    data->gdata = (unsigned char *)malloc(size);
    data->bdata = (unsigned char *)malloc(size);
  }

  for(y=0;y<data->ysize;y++){
    getrow(image,r,y,0);
    if(data->type==COLOR){
      getrow(image,g,y,1);
      getrow(image,b,y,2);
    }
    for(x=0;x<data->xsize;x++){
      data->rdata[x+data->xsize*(data->ysize-1-y)] = r[x];
      if(data->type==COLOR){
        data->gdata[x+data->xsize*(data->ysize-1-y)] = g[x];
        data->bdata[x+data->xsize*(data->ysize-1-y)] = b[x];
      }
    }
  }
  iclose(image);
  return 0;
}

writergb(filename, data)
char *filename;
struct img *data;
{
  IMAGE *image;
  int y,x;
  unsigned short  r[8192], g[8192], b[8192];

  if(data->type==COLOR)
    image = iopen( filename, "w", RLE(1), 3, data->xsize, data->ysize, 3);
  else
    image = iopen( filename, "w", RLE(1), 2, data->xsize, data->ysize, 1);

  for(y=0;y<data->ysize;y++){
    for(x=0;x<data->xsize;x++){
      r[x]=data->rdata[x + data->xsize*(data->ysize-y-1)];
      if(data->type==COLOR){
        g[x]=data->gdata[x + data->xsize*(data->ysize-y-1)];
        b[x]=data->bdata[x + data->xsize*(data->ysize-y-1)];
      }
    }
    putrow(image, r, y, 0);
```



```
    if(data->type==COLOR){
      putrow(image, g, y, 1);
      putrow(image, b, y, 2);
    }
  }
  iclose(image);
  return 0;
}
```