［演習 05］　色空間変換

画像情報処理においては，R 信号，G 信号，B 信号に対して直接処理を施すことは稀であり，通常は，（我々が処理の内容を直感的に理解できるように）これらの 3 原色信号を我々の知覚に調和した信号に変換してから処理を施す．このような変換のことを色空間変換という．

本資料は，本演習で使用する色空間とその変換方法について説明するものである．

1.　演習で使用する色空間

本演習では，ITU-R BT.601 という規格で定められている Y 信号，Cb 信号，Cr 信号を用いた色空間において処理を行う．ここで，Y 信号は輝度（明るさ）を表す信号であり，Cb 信号，Cr 信号は，色差（色）を表す信号である．

2.　色空間変換

2.1　　RGB 信号から YCbCr 信号への変換

R，G，B 信号から Y，Cb，Cr 信号への変換式を以下に示す．

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5000 \\ 0.5000 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (1)$$

ここで，R，G，B 信号と Y 信号は正の値をとるが，Cb，Cr 信号は，正または負の値をとるので，プログラムを作成する際には注意が必要である．

2.2　　YCbCr 信号から RGB 信号への変換

Y，Cb，Cr 信号から R，G，B 信号への変換式を以下に示す．

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0000 & 0.0000 & 1.4020 \\ 1.0000 & -0.3441 & -0.7141 \\ 1.0000 & 1.7720 & 0.0000 \end{bmatrix} \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} \qquad (2)$$

ここで，R，G，B 信号と Y 信号は正の値をとるが，Cb，Cr 信号は，正または負の値をとる．また，言うまでもなく，式(2)の変換行列は，式(1)の変換行列の逆行列である．

3. 色空間変換を含む画像情報処理の流れ

色空間変換を含む画像情報処理の流れを図 5.1 に示す．以下，具体的な手順を示す．

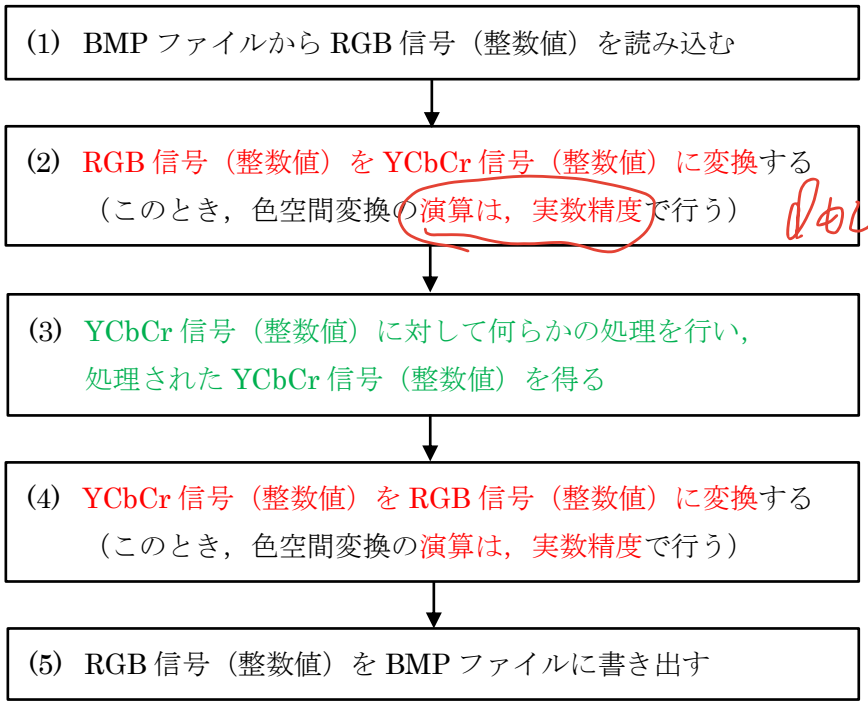

図 5.1

(1) BMP ファイルから R，G，B 信号を読み込む．本演習においては，R，G，B 信号は，0～255 の整数値をもつ．

(2) 読み込んだ RGB 信号を，以下に示す手順で，Y，Cb，Cr 信号に変換する．

(a) 式(1)を用いて実数精度で色空間変換を行い，Y，Cb，Cr 信号を実数値として得る．

(b) 実数値として得られた Y，Cb，Cr 信号を四捨五入する．
（Cb，Cr 信号は，正または負の値をとるので，四捨五入する際には注意すること）

(c) 四捨五入された Cb，Cr 信号にオフセット値 128 を加算する．
（Y 信号に対しては何もしない）

(d) Y，Cb，Cr 信号が，255 より大きい場合には 255 に，0 未満の場合には 0 に，強制的に置き換える．

(e) こうして得られた Y，Cb，Cr 信号を格納する．
このようにして得られた格納データと，そのデータが示す実際の Y，Cb，Cr 信号の値との関係を表 5.1 に示す．この関係をよく理解しておこう．

(3) 上記格納データに対して何らかの処理を（一般的には実数精度で）行う．そして，処理後の Y，Cb，Cr 信号の値を，処理前と同様に，それぞれ 1 バイトのデータとして（すなわち，実数値として得られる場合には，上記(2)と同様な手順で 1 バイトのデータにして），プログラム内の変数に格納する．

(4) 処理後の Y，Cb，Cr 信号を，以下に示す手順で，RGB 信号に変換する．

(a) Cb，Cr 信号から，オフセット値 128 を減算する．（Y 信号に対しては何もしない．）

(b) 式(2)を用いて実数精度で色空間変換を行い，R，G，B 信号を実数値として得る．

(c) 実数値として得られた R，G，B 信号を四捨五入する．

(d) R，G，B 信号が，255 より大きい場合には 255 に，0 未満の場合には 0 に，強制的に置き換える．

(5) RGB 信号を BMP ファイルに書き出す．

表 5.1

（補足）Y 信号を 1 バイトで表す場合，Y 信号は正の値のみを持つため，「2 進数の 0000 0000」は「10 進数の 0」に，「2 進数の 1111 1111」は「10 進数の 255」に対応させている（すなわち，一般的な符号なし 2 進数である）．

一方で，Cb，Cr 信号は，正と負の両方の値を持ち，これを 1 バイトの 2 進数で表さなくてはならない．そのため，BMP フォーマットでは，「2 進数の 0000 0000」は「10 進数の-128」に，「2 進数の 1000 0000」は「10 進数の 0」に，「2 進数の 1111 1111」は「10 進数の 127」に対応させている．（このような対応のさせ方をオフセットバイナリという．一般的な 2 の補数を用いた符号付き 2 進数ではないので注意すること．）

格納データ（1 バイト）		格納データが示す実際の値		
2 進数表現	10 進数表現	Y 信号	Cb 信号	Cr 信号
1111 1111	255	255 以上	127 以上	127 以上
1111 1110	254	254	126	126
…		…	…	…
1000 0001	129	129	1	1
1000 0000	128	128	0	0
0111 1111	127	127	-1	-1
…		…	…	…
0000 0001	1	1	-127	-127
0000 0000	0	0 以下	-128 以下	-128 以下

4. 参考プログラム

図 5.1 の(2)，(4)の色空間変換を行うための参考プログラムを ex_05_ref_1 に示す．また，その実行結果を ex_05_ref_1(result)に示す．プログラムが何を行っているのか，実行結果を参考にコードを 1 行ずつ読み進め，処理の内容をよく理解しておこう．

ex_05_ref_1

```
/*--- ex_05_ref_1 ---*/

#include <stdio.h>

#define CH 3
#define Ych 0

#define ROW 3
#define COL 3

int main(void)
{
    double rgb_to_ybr[ROW][COL] =
    {
      { 0.2990, 0.5870, 0.1140},
      {-0.1687,-0.3313, 0.5000},
      { 0.5000,-0.4187,-0.0813}
    };

    double ybr_to_rgb[ROW][COL] =
    {
      { 1.0000, 0.0000, 1.4020},
      { 1.0000,-0.3441,-0.7141},
      { 1.0000, 1.7720, 0.0000}
    };

    int rgb_in[CH] = { 0, 255, 0 }; // 入力RGB信号（整数値）
    int ybr[CH]; // YCbCr信号（整数値）
    int rgb_out[CH]; // 出力RGB信号（整数値）

    double dtemp[CH];
    int ch;
    int i;
    int itemp;

    //--- 入力RGB信号（整数値）の表示 ---
    printf("\n＜入力RGB信号（整数値）＞\n");
    for (ch = 0; ch < CH; ch++)
        printf("%4d\n", rgb_in[ch]);

    //--- RGB信号（整数値）からYCbCr信号（実数値）への変換 ---
    for (ch = 0; ch < CH; ch++)
    {
        dtemp[ch] = 0.0;
        for (i = 0; i < COL; i++)
            dtemp[ch] += rgb_to_ybr[ch][i] * (double)rgb_in[i];
    }

    //--- 変換されたYCbCr信号（実数値）の表示 ---
    printf("\n＜変換されたYCbCr信号（実数値）＞\n");
    for (ch = 0; ch < CH; ch++)
        printf("%10.4f\n", dtemp[ch]);
```

（次ページに続く）

```
//--- YCbCr信号（実数値）からYCbCr信号（整数値）への変換 ---
for (ch = 0; ch < CH; ch++)
{
    // 四捨五入
    if (dtemp[ch] > 0.0)
        itemp = (int)(dtemp[ch] + 0.5);
    else
        itemp = (int)(dtemp[ch] - 0.5);

    // Cb，Cr信号にオフセット値128を加える
    if (ch != Ych)
        itemp += 128;

    // 信号値を0～255の範囲内に制限する
    if (itemp > 255)
        itemp = 255;
    else if (itemp < 0)
        itemp = 0;

    // YCbCr信号値（整数値）を格納
    ybr[ch] = itemp;
}

//--- 変換されたYCbCr信号（整数値）の表示 ---
printf("\n＜変換されたYCbCr信号（整数値）＞\n");
for (ch = 0; ch < CH; ch++)
    printf("%4d\n", ybr[ch]);

//--- YCbCr信号（整数値）からRGB信号（実数値）への変換 ---
for (ch = 0; ch < CH; ch++)
{
    dtemp[ch] = 0.0;
    for (i = 0; i < COL; i++)
    {
        if (i == Ych)
            dtemp[ch] += ybr_to_rgb[ch][i] * (double)ybr[i];
        else // Cb，Cr信号は，オフセット値128を減ずる
            dtemp[ch] += ybr_to_rgb[ch][i] * (double)(ybr[i] - 128);
    }
}

//--- 変換されたRGB信号（実数値）の表示 ---
printf("\n＜変換されたRGB信号（実数値）＞\n");
for (ch = 0; ch < CH; ch++)
    printf("%10.4f\n", dtemp[ch]);
```

（次ページに続く）

```
    //--- RGB信号（実数値）からRGB信号（整数値）への変換 ---
    for (ch = 0; ch < CH; ch++)
    {
        // 四捨五入
        if (dtemp[ch] > 0.0)
            itemp = (int)(dtemp[ch] + 0.5);
        else
            itemp = (int)(dtemp[ch] - 0.5);

        // 信号値を0～255の範囲内に制限する
        if (itemp > 255)
            itemp = 255;
        else if (itemp < 0)
            itemp = 0;

        // RGB信号値（整数値）を格納
        rgb_out[ch] = itemp;
    }

    //--- 出力RGB信号（整数値）の表示 ---
    printf("\n＜出力RGB信号（整数値）＞\n");
    for (ch = 0; ch < CH; ch++)
        printf("%4d\n", rgb_out[ch]);

    return 0;
}
```

ex_05_ref_1(result)

```
＜入力RGB信号（整数値）＞
   0
 255
   0

＜変換されたYCbCr信号（実数値）＞
  149.6850
  -84.4815
 -106.7685

＜変換されたYCbCr信号（整数値）＞
 150
  44
  21

＜変換されたRGB信号（実数値）＞
   -0.0140
  255.3131
    1.1520

＜出力RGB信号（整数値）＞
   0
 255
  1
```