Moodleの
出席確認を
提出しておいて
下さい。

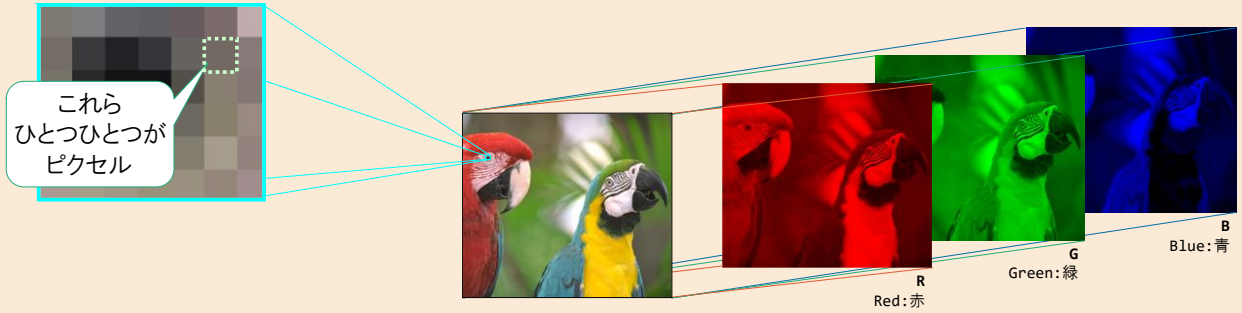
VisualStudio2019(等)で、
C言語+OpenCV のコーディングができる状態に
準備していてください。

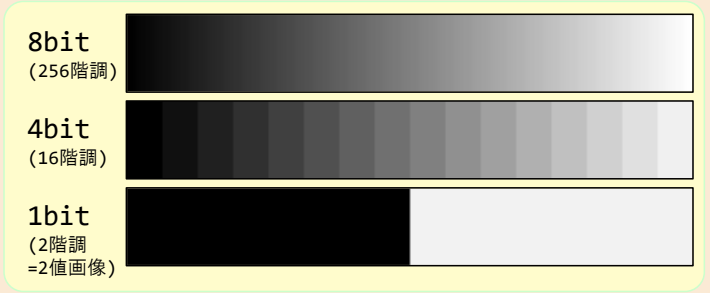
画像処理 (4J)

第08回

第6回のまとめ

- ラスタ画像とベクタ画像 ・・・ この授業では、ピクセル情報の集合であるラスタ画像を扱う
- 解像度 ・・・ 画像の大きさ(細かさ)
- ピクセル(画素) ・・・ ラスタ画像を構成する1つの点
- チャンネル ・・・ 1ピクセルをいくつの値で表現するか (例:RGBの3ch)
- 階調数 ・・・ 濃度を何段階で表現するか (例:8bit(=256段階))



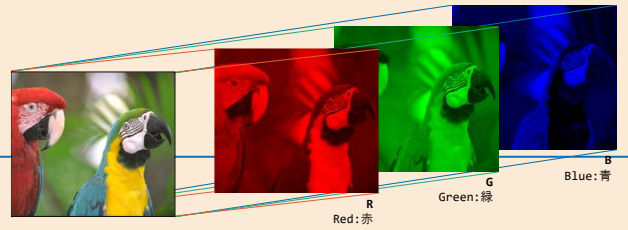


デジタル写真 = 有限の解像度で空間的にサンプリング(標本化)し、
有限の階調値で明るさを表現(量子化) したもの ...と捉えることができる。

※音声信号のデジタル化と対応させると、サンプリング周波数が解像度に、量子化bit数が階調数に、チャンネル数はそのまま対応する

第7回のまとめ

グレイスケール画像



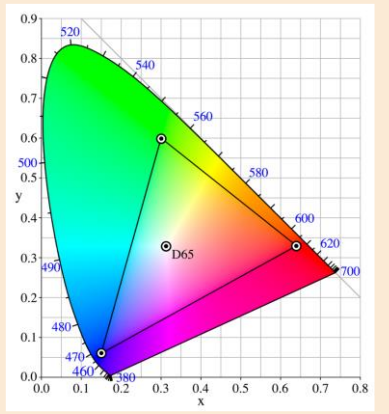
RGBカラー画像

●グレイスケール画像とカラー画像

- ➢グレイスケール画像は1つの(x,y)座標点に1つの濃度値 g(x,y)
- ➢RGBカラー画像は、1つの座標点に、3つの濃度値

●RGBカラー画像

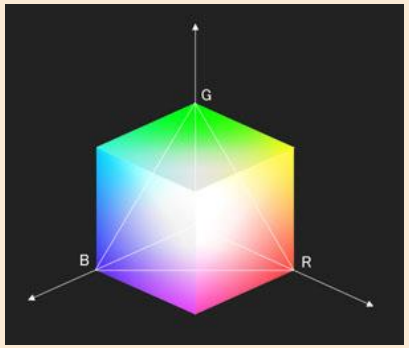
- ➢RGB値が同じでも、同じ色が表示されるとは限らない
- ➢sRGBに準拠させれば、一貫した色表現が可能。
  (ただし表現できる色域が狭い)

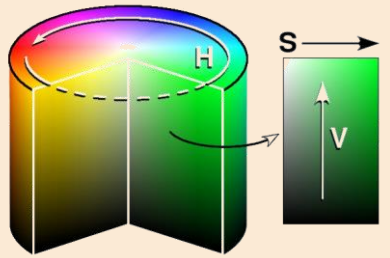


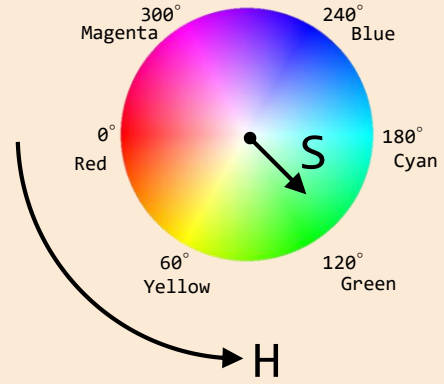
(CIE1931色空間のxy色度図上に示した)
sRGBの色域

●色空間： RGBとHSV

- ➢相互に変換可能
- ➢他にも様々な表色系がある







カラー画像の
グレイスケール化

グレイスケール画像とRGBカラー画像

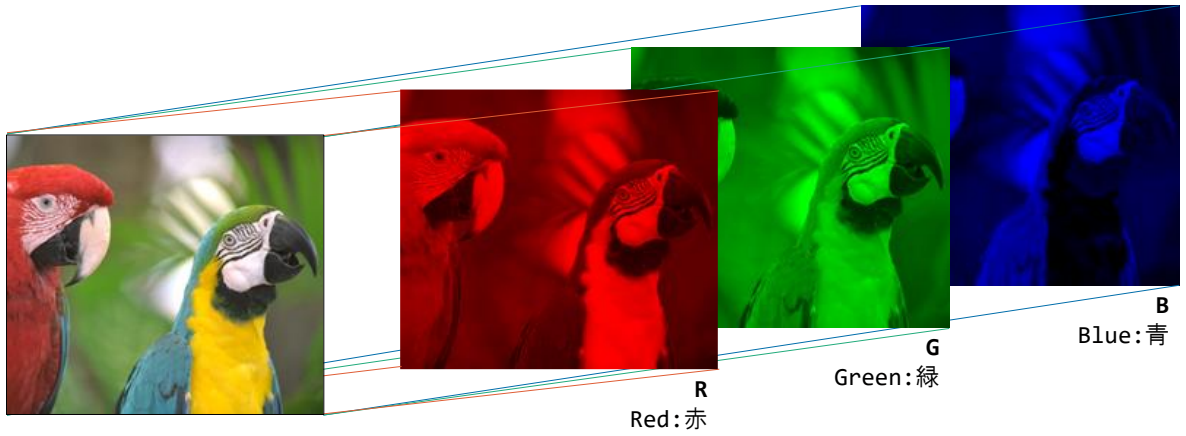
再

●グレイスケール画像　2次元平面上の1つの座標点(x,y) に対応する、
1つの濃度値 g(x,y) の集合として表すことができる。

●カラー画像　1つの座標点(x,y) に対して、
複数の濃度値(カラーチャンネル)を持つ。

➢最も馴染み深いのは、
R,G,B (Red, Gree, Blue) の3チャンネルで表現される“RGBカラー画像”

グレイスケール画像



RGBカラー画像

カラー画像のグレイスケール化

●R,G,Bの3値から、1つの濃度値Y を計算 ･･･ どうやって？

① 単純平均： R,G,Bの平均をとる　　すなわち $Y = \frac{R+G+B}{3}$

② 加重平均： R,G,Bに何らかの係数をかけて平均をとる

- **NTSC加重平均法**

$$Y = (0.298912 \times R + 0.586611 \times G + 0.114478 \times B)$$

グレイスケール画像　　RGBカラー画像

カラー画像のグレイスケール化

●R,G,Bの3値から、1つの濃度値Y を計算 ･･･ どうやって？

① 単純平均： R,G,Bの平均をとる　　すなわち $Y = \frac{R+G+B}{3}$

② 加重平均： R,G,Bに何らかの係数をかけて平均をとる

➢ **NTSC加重平均法**

$$Y = (0.298912 \times R + 0.586611 \times G + 0.114478 \times B)$$

【Gの重みが大きい】

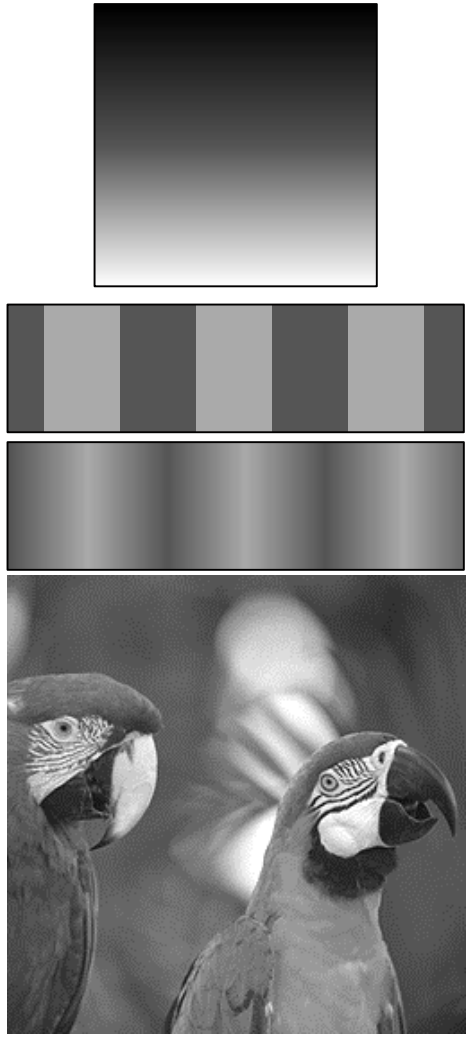
グレイスケール画像　　RGBカラー画像

カラー画像のグレイスケール化

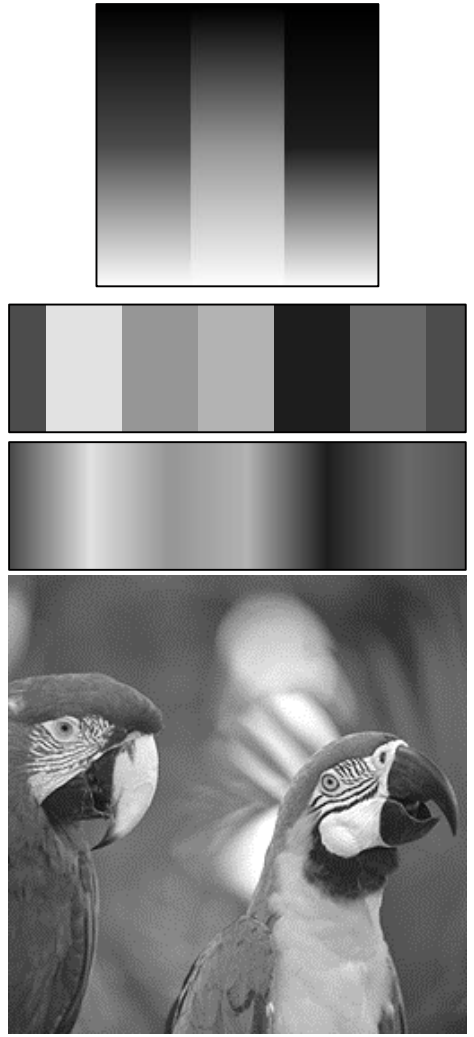
とてもよく使われる

カラー画像(元画像)

単純平均



NTSC加重平均



カラー画像のグレイスケール化

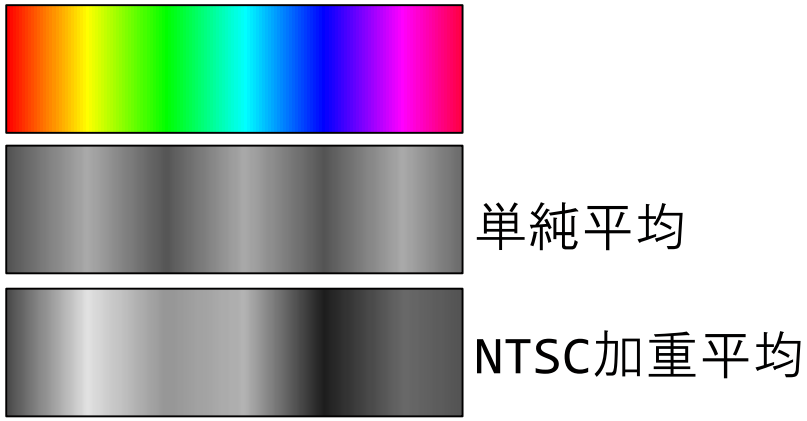
●R,G,Bの3値から、1つの濃度値Y を計算 ・・・ どうやって？

① 単純平均： R,G,Bの平均をとる　　すなわち $Y = \frac{R+G+B}{3}$

② 加重平均： R,G,Bに何らかの係数をかけて平均をとる

➢ **NTSC加重平均法**

$$Y = (0.298912 \times R + 0.586611 \times G + 0.114478 \times B)$$



単純平均の場合、例えば、青色だった部分が明るく、緑色だった部分が暗く感じられるような変換結果となる。

これは人間の眼の分光感度が、光の波長によって異なることが関係している。
(青付近の波長の光に対する眼の感度は、緑付近の波長の光に対する感度よりも悪い。
つまり、同じ強度の光でも、青は暗く感じ、緑は明るく感じる。)

NTSC加重平均法は、このことを考慮した変換式の代表であり、よく使われる。

画像の二値化

二値画像 (binary image)

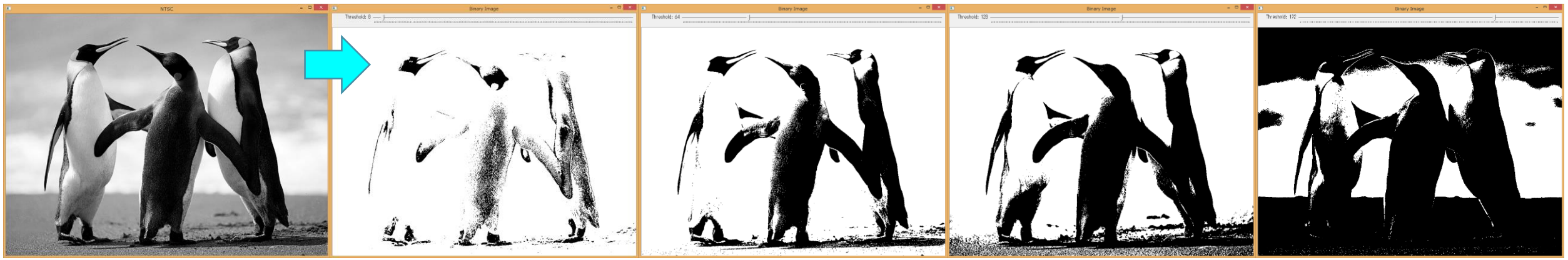
●「**二値画像**」 ･･･ {0,1} の2つの濃度値のみで表される画像

- 1bit(=2階調)のグレイスケール画像と捉えることもできる
- カラー画像や、グレイスケール画像を二値画像に変換することを 「**二値化**」 と呼ぶ
- 例えば、8bit(=256階調)のグレイスケール画像の中央の濃度値127を境界として、

  127以上 ➡ "1"

  127未満 ➡ "0"

  のように二値化する場合、この 127 の値を「**閾値(threshold)**」と呼ぶ。



グレイスケール画像(元画像)
8bit(=濃度値:0～255)

閾値=8

閾値=64

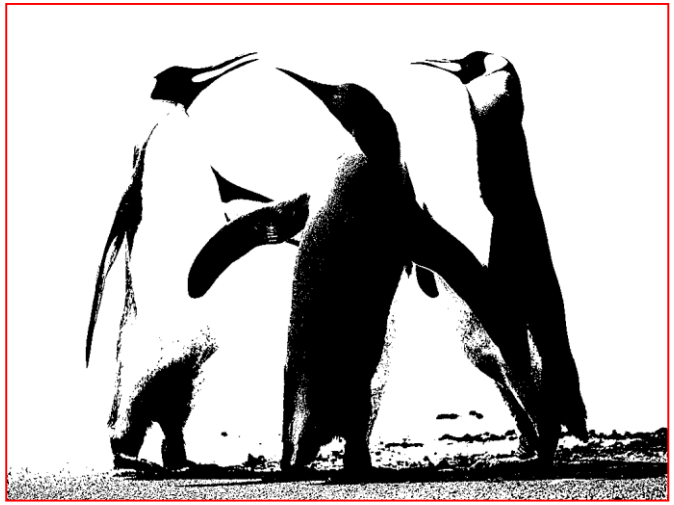
閾値=128

閾値=192

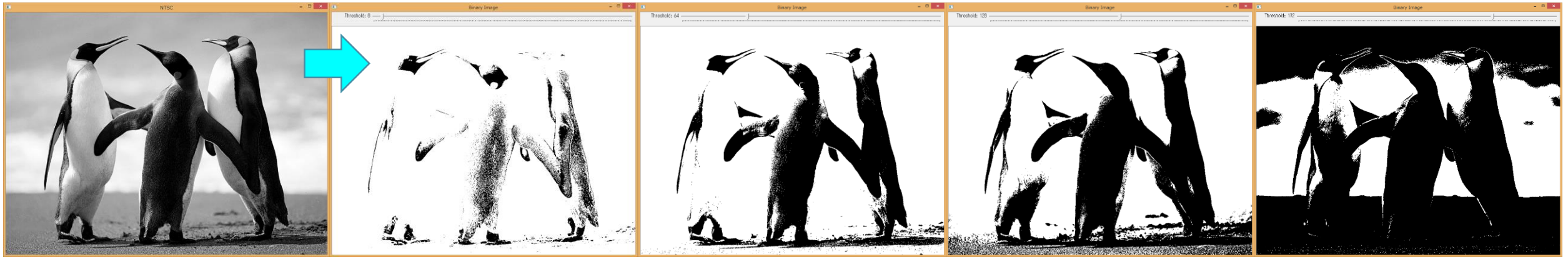
二値画像 (binary image)

- 閾値は任意に決めることができる。

- 閾値を自動決定するアルゴリズムも考案されている。
  - 代表的なものに、画像統計量を用いた「**大津の方法**(判別分析方)」がある。



大津の方法で計算した閾値(111)で
二値化した結果



グレイスケール画像(元画像)
8bit(=濃度値:0～255)

閾値=8

閾値=64

閾値=128

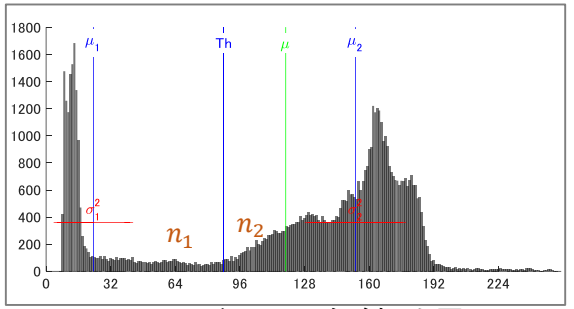
閾値=192

大津の方法 (Otsu's method)

- 大津の方法による二値化閾値の決定方法は、判別分析法(Discriminant analysis method)とも呼ばれ、クラスを2つに分けた時の「分離度」を最大化する方法である。

- 分離度は、クラス間分散とクラス内分散の比として求めることができる。
  - ある閾値Th でピクセルを2つのクラスに分けた時、
    閾値よりもピクセル値が小さいクラスをクラス1、
    閾値よりもピクセル値が大きいクラスをクラス2として、
    それぞれのクラスごとの統計量を表の記号で表すとする。

	クラス1 (< Th)	クラス2 (> Th)	画像全体
ピクセル数	n_1	n_2	n
平均	μ_1	μ_2	μ
分散	σ_1^2	σ_2^2	σ^2



ヒストグラムと各統計量

  - クラス内分散 σ_W^2 は、 $\sigma_W^2 = \frac{n_1\sigma_1^2+n_2\sigma_2^2}{n_1+n_2}$
  - クラス間分散 σ_B^2 は、 $\sigma_B^2 = \frac{n_1(\mu_1-\mu)^2+n_2(\mu_2-\mu)^2}{n_1+n_2}$

  この比として定義した分離度 $\frac{\sigma_B^2}{\sigma_W^2} = \frac{n_1(\mu_1-\mu)^2+n_2(\mu_2-\mu)^2}{n_1\sigma_1^2+n_2\sigma_2^2}$ が最大となる Th を全探索で求め、閾値とする。

  - なお、全分散 σ^2 (画像全体の分散)は、クラス内分散 σ_W^2 とクラス間分散 σ_B^2 の和として $\sigma^2 = \sigma_W^2 + \sigma_B^2$ と求めることができるので、これを用いて分離度の式を再度整理すると、 $\frac{\sigma_B^2}{\sigma_W^2} = \frac{\sigma_B^2}{\sigma^2-\sigma_B^2}$ と書ける。
  - ここで、 σ^2 は Th によらず一定なので、クラス間分散 σ_B^2 が最大になるときに分離度は最大になる
    (σ_B^2 が大きいほど分母は小さく、分子は大きくなるため)と言い換えることが出来る。
  - さらに、クラス間分散(式2.2)の分母も、Thによらず一定なので、
    結局は、 分子 $V = n_1(\mu_1-\mu)^2 + n_2(\mu_2-\mu)^2$ が最大となる Th を探索すれば良いことになる。

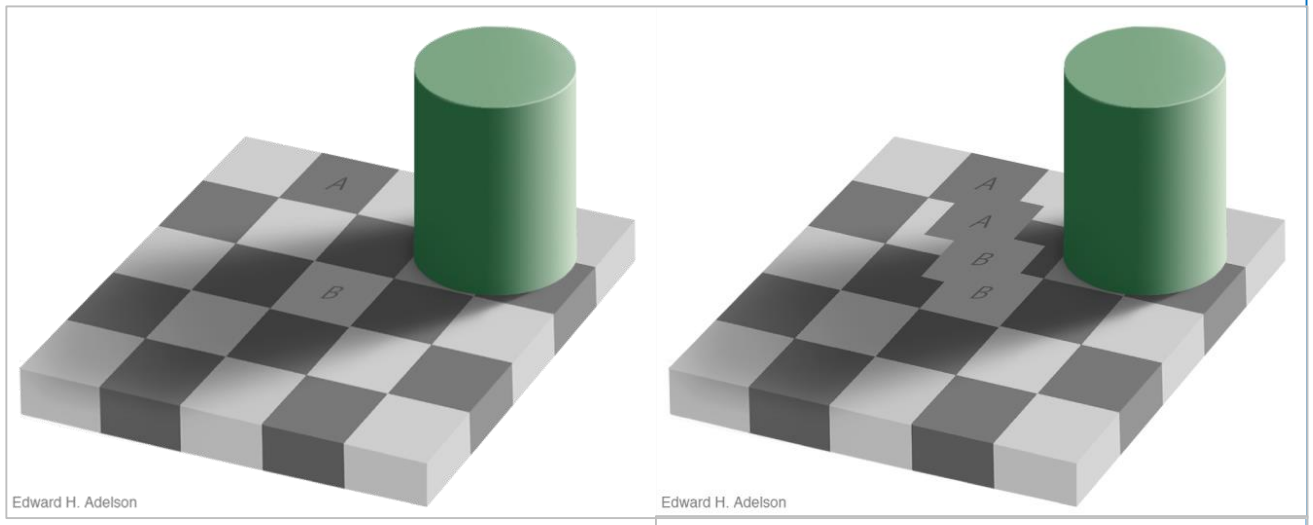
大津の方法の実装は、
ヒストグラムを学んでから、
改めて扱う予定です。

二値化について

●二値化の用途

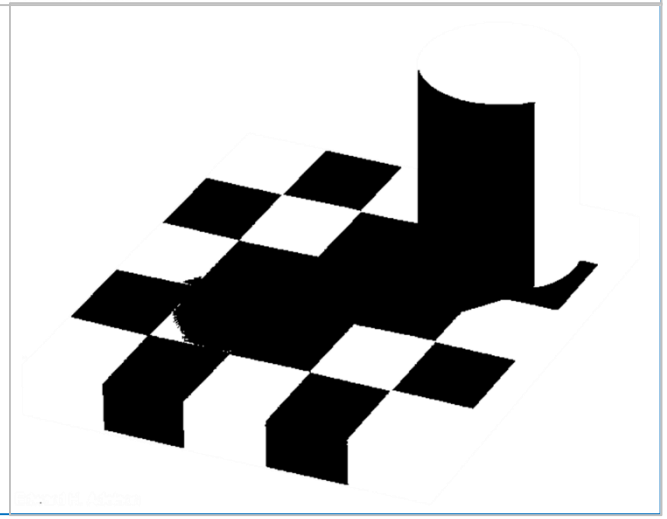
➢ {0,1} の単純化した情報となるため、様々な画像処理や画像認識の**前処理**として頻繁に使われる

- 文字認識
- 物体検出
- …

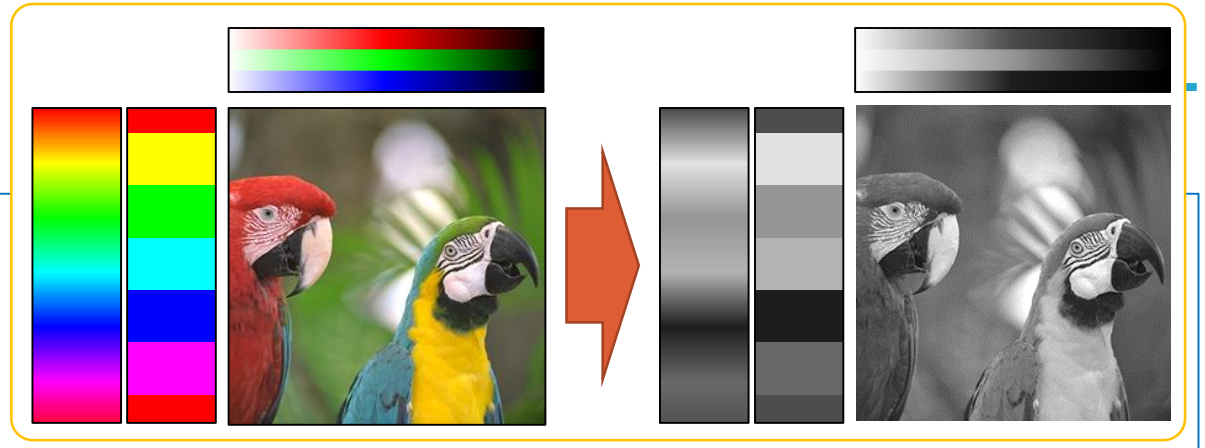


●カラー画像の場合は、
グレイスケール化後に
二値化する場合が多い

●画像全体にグラデーションがかかっている場合等、
画像全体で単一の閾値を使う二値化(大域的二値化)
では良好な結果が得られない場合もある。



まとめ



●グレイスケール化

➢**NTSC加重平均法**がよく使われる

$$Y = (0.298912 \times R + 0.586611 \times G + 0.114478 \times B)$$

●二値化

➢**閾値**を堺に、{0,1} の二値の画像に変換

➢閾値は任意に決められるが、画像統計量から閾値を自動決定する方法として**大津の方法**(判別分析法)が有名。

演習：
グレイスケール化と二値化

OpenCVを使った画像生成の流れ

再

【画像生成時の流れ】

```
IplImage* img = cvCreateImage(CvSize size, IPL_DEPTH_8U, int channels);
 ↓                                                  ･･･ 画像を扱うための構造体 img を生成する
cvSetZero(img);                ･･･ 画像データ img を 0(=黒) で初期化
 ↓
 ↓      【画像読み込み時の流れ】
 ↓     IplImage* img = cvLoadImage(const char* filename, CV_LOAD_IMAGE_UNCHANGED);
 ↓     ↓                        ･･･ 画像ファイルを読み取り、画像データ img を生成
 ↓     ↓
<img に対する何らかの処理>
 ↓
 ↓
cvSaveImage(img);              ･･･ 画像データ img を画像ファイルとして保存
 ↓
cvReleaseImage(&img);          ･･･ 画像を扱うための構造体 img に割り当てたメモリの開放
```

各種関数のリファレンス(1)


```
IplImage* img
 = cvCreateImage(CvSize size, int depth, int channels);
```

- size:　　画像のサイズ。
- depth:　　ピクセルのデータ形式。
  ※本授業では常に IPL_DEPTH_8U (符号無し8ビット整数 = unsigned char)とする。
- channels: ピクセル毎のチャンネル数。[グレイスケール = 1 , カラー = 3]

※ロードに失敗した場合は NULL が返る。
内部でmalloc()されているので、cvReleaseImage()で開放する必要がある。

```
typedef struct CvSize {
  int width;    /* 横幅 */
  int height:   /* 高さ */
} CvSize;
```

各種関数のリファレンス(2)

再


```
void cvSetZero(IplImage *img);
```

- img: cvCreateImage() が返した IplImage* のアドレス。
  全ピクセルデータを 0(黒)で初期化する

```
IplImage* img
 = cvLoadImage(const char* filename, int iscolor);
```

- filename:　　ファイル名。対応ファイル形式は(表1)を参照。
- iscolor:　　読み込む画像のカラーの種類。
  ※本授業では常に CV_LOAD_IMAGE_UNCHANGED とする。

指定した画像ファイルを IplImage 形式に読み込む

※内部でmalloc()されているので、cvReleaseImage()で開放する必要がある。

各種関数のリファレンス(3)

再


```
int cvSaveImage(const char* filename, IplImage* image);
```

- filename:　ファイル名。拡張子で保存形式が決まる。→ (表1)を参照。
- image:　保存する画像データ

IplImage を、画像ファイルとして保存する。

※保存に成功した場合は 1 、失敗した場合は 0 が返る(らしい)。

```
void cvReleaseImage(IplImage** img);
```

- img: cvCreateImage() が返した IplImage* のアドレス。

cvCreateImage()やcvLoacImage()で確保された領域を開放する。

(表 1) cvLoacImage()、 cvSaveImage() の対応形式と、指定する拡張子

形式	Windows bitmaps	Jpeg	Portable Network Graphics	Portable image format	Sun rasters	TIFF files	OpenEXR HDR images	JPEG 2000 images
拡張子	BMP,DIB	JPEG, JPG,JPE	PNG	PGM,PGM PPM	SR,RAS	TIFF, TIF	EXR	Jp2

IplImage 構造体（types_c.h 内で定義）再


```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;       /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                         IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;     /* 0 - interleaved color channels, 1 - separate color channels.
                           cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "             " */
    struct _IplTileInfo *tileInfo;  /* "             " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

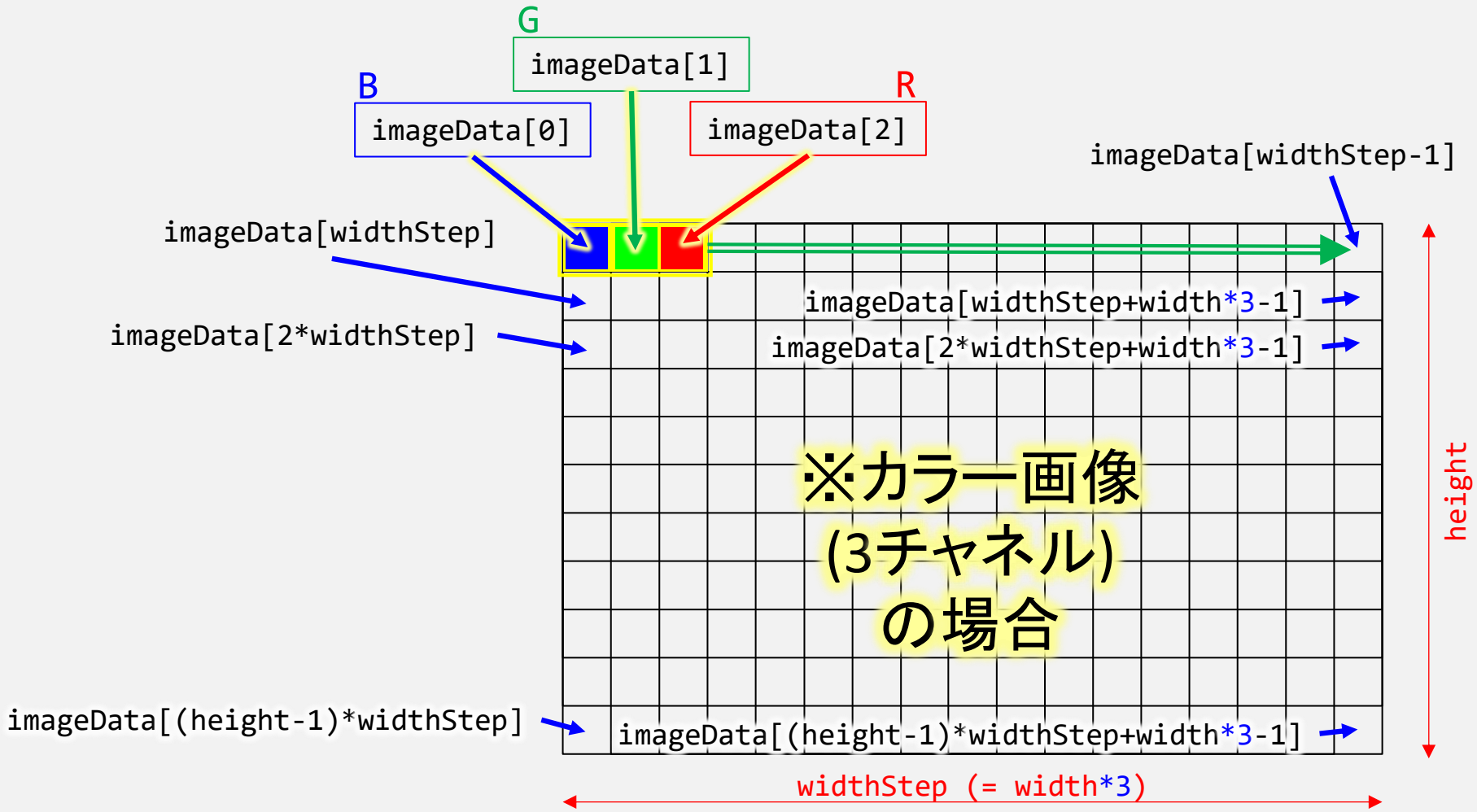
カラー画像／グレイスケール画像の判定 再

- IplImage の メンバ変数の nChannels
  - 3の場合カラー画像
  - 1の場合グレイスケール画像

(※本授業では、nChannelsが1か3の場合のみ、取り扱うものとする)

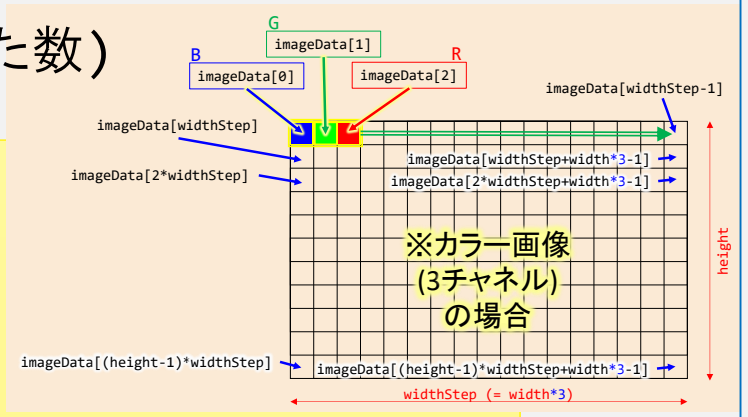
IplImageのRGB値へのアクセス 再

RGBカラー画像の個々のRGB値は、
下図のような順に一次元配列 imageData[] に格納される。



IplImageのRGB値へのアクセス 再

- IplImage のメンバ変数を用いて、個々のピクセルへアクセスする。
- imageDataには、
  BGRBGRBGRBGR……の順で格納されていることに注意 （RGBの順ではない！）。
  - char* imageData ・・・ 画像データへのポインタ
  - int widthStep ・・・ 画像データ1ライン分のバイト数(= char で数えた数)



【例】
IplImage *img の画像(RGBカラー画像)に対して、
座標点 (x, y) のカラーチャネルごとのピクセル値(RGB値)へは、

```
b = img->imageData[img->widthStep * y + x * 3 + 0];    // B値
g = img->imageData[img->widthStep * y + x * 3 + 1];    // G値
r = img->imageData[img->widthStep * y + x * 3 + 2];    // R値
```

としてアクセスすることが出来る。

IplImageのRGB値へのアクセス（!少し変更!）

- imageData は、構造体の定義通り、 char型 として宣言されています。
- しかし、格納されるデータは unsigned char型 で入っています。
  - 単に char と記述した場合、 signed char と扱われるか、 unsigned char と扱われるかは実は処理系依存です。
  - VCは char == signed char として扱うようです

●大変ややこしいですが、対応は簡単で、
読み出す時は必ず（unsigned char）でキャストすればOKです。

【例】
IplImage *img の画像(RGBカラー画像)に対して、
座標点 (x, y) のカラーチャネルごとのピクセル値(RGB値)へは、

```
b = (unsigned char)img->imageData[img->widthStep * y + x * 3 + 0];
g = (unsigned char)img->imageData[img->widthStep * y + x * 3 + 1];
r = (unsigned char)img->imageData[img->widthStep * y + x * 3 + 2];
```

としてアクセスすることが出来る。

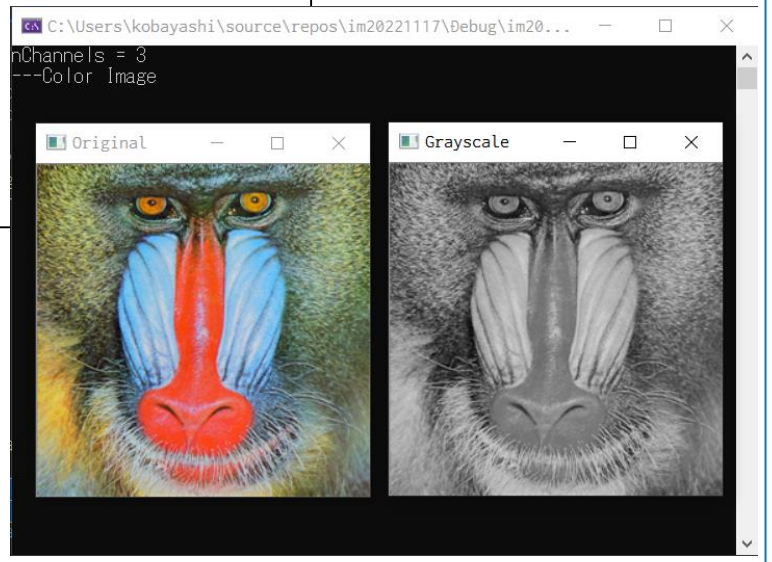
演習：
グレイスケール化

Sample Program [p03.cpp]

●カラー画像グレイスケール化する(NTSC加重平均法)

$$Y = (0.298912 \times R + 0.586611 \times G + 0.114478 \times B)$$

```
// p03 : カラー画像をグレイスケール化する

#include <stdio.h>
#include <opencv/highgui.h>

// カラー画像をグレイスケール化
void bgr2gray(IplImage* gray, IplImage* bgr) {
   for (int y = 0; y < gray->height; y++) {
      for (int x = 0; x < gray->width; x++) {
         gray->imageData[gray->widthStep * y + x]
            = … // この部分を実装
      }
   }
}


void main()
{
   IplImage* img;
   IplImage* img_gray;
   char filename[] = "Mandrill.bmp";

   // 画像データの読み込み
   if ((img = cvLoadImage(filename, CV_LOAD_IMAGE_UNCHANGED)) == NULL) {
      printf("画像ファイルの読み込みに失敗しました。¥n");
      return;
   }
   // 読み込んだ画像と同じサイズのグレイスケール画像(nChannels=1)を生成
   img_gray = cvCreateImage(cvSize(img->width, img->height), img->depth, 1);

   cvNamedWindow("Original");
   cvShowImage("Original", img);

   printf("nChannels = %d¥n", img->nChannels);
   if (img->nChannels == 3) {
      // カラー画像だった場合、グレイスケール化した画像を表示
      printf("---Color Image¥n");
      bgr2gray(img_gray, img);  // グレイスケール化

      cvNamedWindow("Grayscale");
      cvShowImage("Grayscale", img_gray);
   }
   else if(img->nChannels == 1) {
      // グレイスケール画像だった場合は何もしない
      printf("---Grayscale Image¥n");
   }

   cvWaitKey(0);
   cvDestroyAllWindows();
   cvReleaseImage(&img);
   return;
}
```



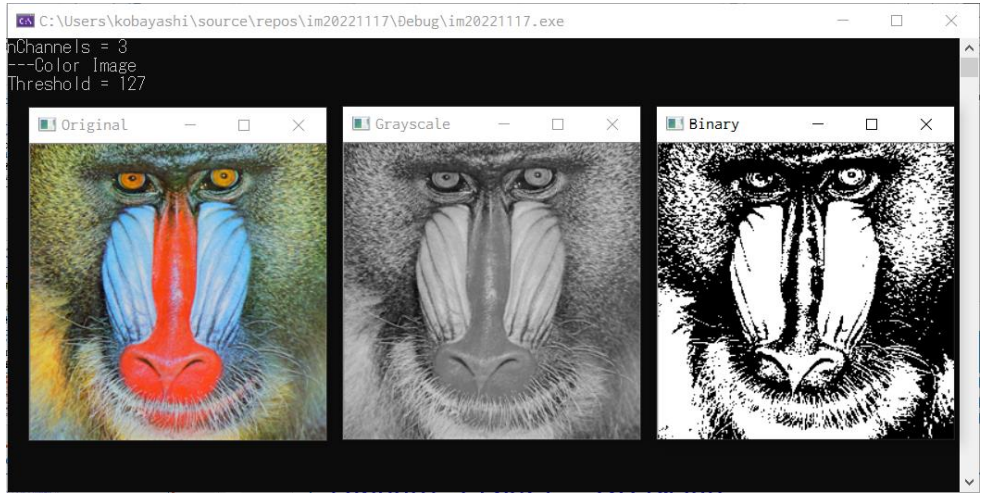
課題 No.08

課題 No.08

●カラー画像/グレイスケール画像を二値化する

二値画像は、0,1ではなく、
グレイスケール画像の 0,255
として生成すること。

```
// No.08 : カラー画像/グレイスケール画像を二値化する

#include <stdio.h>
#include <opencv/highgui.h>

// グレイスケール画像を二値化
void gray2bin(IplImage* bin, IplImage* gray, char th) {
   // ここを実装する
}



// カラー画像をグレイスケール化
void bgr2gray(IplImage* gray, IplImage* bgr) {
   // ここは p03 と同じ
}
```




```
void main()
{
   // 前半は p03 と同じ
   IplImage* img_bin;
   char th = 127;    // 閾値

   printf("nChannels = %d¥n", img->nChannels);
   if (img->nChannels == 3) {
      // カラー画像だった場合、グレイスケール化した画像を表示
      printf("---Color Image¥n");
      bgr2gray(img_gray, img);  // グレイスケール化

      cvNamedWindow("Grayscale");
      cvShowImage("Grayscale", img_gray);
   }
   else if(img->nChannels == 1) {
      // グレイスケール画像だった場合はは、img_gray に元画像をコピーする
      printf("---Grayscale Image¥n");
      cvCopy(img, img_gray);
   }

   // グレイスケール画像と同じ大きさの画像を生成 (※0,1 は グレイスケール画像の 0,255 で表現)
   img_bin = cvCreateImage(cvSize(img->width, img->height), img->depth, img_gray->nChannels);
   gray2bin(img_bin, img_gray, th);

   cvNamedWindow("Binary");
   cvShowImage("Binary", img_bin);
   printf("Threshold = %d¥n", th);

   cvWaitKey(0);
   cvDestroyAllWindows();
   cvReleaseImage(&img);
   return;
}
```