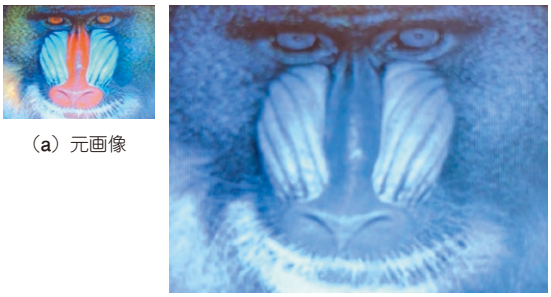
第2章 モノクロ化から色相/彩度/明度の変更，ディープ・ラーニング向け圧縮まで

明るさ&色

米田 幸生



（a）元画像　（b）変換後

図1　カラー画像をCIE XYZでグレー・スケール化

本章では，画像の色合いを変えてみます．人工知能の学習データ作りや画像処理による特定の物体検出に役立ちます．また，モノクロ化は画像のデータ量削減にも役立ちます．

画像のモノクロ変換と2値化

● 別の色空間に置き換え輝度情報だけを利用する「グレー・スケール化」

グレー・スケールは，画像の輝度表現方法の一種で，画像を色味のない明るさの度合いだけで表現します．今回は，RGB平均を使用せずに，CIE XYZを使用しました．CIE XYZは，CIE（国際照明委員会）がまとめた規格で，RGBとは別の軸を持つ色空間です．XYZ軸のY軸が輝度（明るさ）で，XとZが色味に対応し以下の計算式で表せます．

$$Y = 0.2126 \times R + 0.7152 \times G + 0.0722 \times B$$

元の画像を**図1**（a）に，画像処理の結果を**図1**（b）に示します．ソースコードを**リスト1**に示します．

● 白と黒の2値で表現する「2値化」

2値化するには，あらかじめしきい値を決めておき，画素の値（輝度）がしきい値より大きければ白，小さければ黒に変換します．今回は，まずグレー・スケール化した後に，しきい値を128に設定して2値化しました．画像処理の結果を**図2**に，ソースコードを**リスト2**に示します．

リスト1　グレー・スケール化処理

```
#ifndef GRAYSCALE_H
#define GRAYSCALE_H

#include "Original.h"

class GrayScale : public Original {

public:
  virtual void display(const char* fname);

};

#endif // GRAYSCALE_H
```

Original画像処理クラスを継承してグレー・スケール・クラスを作っている．以降のほとんどの画像処理クラスで同様の構成になっている

（a）ヘッダ（GrayScale.h）

```
#include "GrayScale.h"

void
GrayScale::display(const char *fname){
  uint8_t *pImg;
  uint8_t *pImg_disp;
  int x,y,bx,by;

  M5.Lcd.clearDisplay(BLACK);

  JpegDec.decode(fname);

  while(JpegDec.read()){
    pImg = JpegDec.pImage;
    pImg_disp = pImg;

    for(by=0; by<JpegDec.MCUHeight; by++){
      for(bx=0; bx<JpegDec.MCUWidth; bx++){
        uint8_t tmp = grayChg(pImg);
        *pImg = *(pImg + 1) = *(pImg + 2) = tmp;
        pImg += 3;
      }
    }
    x = JpegDec.MCUx * JpegDec.MCUWidth;
    y = JpegDec.MCUy * JpegDec.MCUHeight;
    displayPixel(x, y, JpegDec.MCUWidth,
                 JpegDec.MCUHeight, pImg_disp);
  }
  JpegDec.finish();
}
```

JPEG画像の終端までマクロ・ブロック単位でJPEGを読み込む

JPEGのデコードを開始する

グレー・スケール化を行う

（b）メイン（GrayScale.cpp）