

ポスタリゼーションプログラム

- グレースケールで画像を入力
4階調に変換して出力
- プロジェクト名：posterization



グレースケール画像



ポスタリゼーション画像

posterization.cpp (1/3)

```
#include <iostream>
#include <opencv2/opencv.hpp>
#define FILE_NAME "../Debug/apple_tree.jpg"

//ウィンドウ名
#define WINDOW_NAME_INPUT "input"
#define WINDOW_NAME_OUTPUT "output"
#define COLOR_NUM (256) //色値

int main(int argc, const char * argv[]) {
    //画像の入力 (グレースケール入力)
    cv::Mat src_img = cv::imread(FILE_NAME, cv::IMREAD_GRAYSCALE);
    if (src_img.empty()) { //入力失敗の場合
        fprintf(stderr, "Cannot read image file: %s.\n", FILE_NAME);
        return (-1);
    }

    //出力画像のメモリ確保 (グレースケール)
    cv::Mat dst_img = cv::Mat(src_img.size(), CV_8UC1);
```

posterization.cpp (2/3)

```
uchar lut[COLOR_NUM]; //ルックアップテーブル
//ルックアップテーブルの生成（4段階）
for (int i=0; i<COLOR_NUM; i++) {
    if (i <= 63) {
        lut[i] = 0;
    }else if (64 <= i && i <= 127) {
        lut[i] = 85;
    }else if (128<= i && i <= 191) {
        lut[i] = 170;
    }else{
        lut[i] = 255;
    }
}
```

posterization.cpp (3/3)

//画像の走査

```
for (int y=0; y<dst_img.rows; y++) {  
    for (int x=0; x<dst_img.cols; x++) {  
        uchar s = src_img.at<uchar>(y, x); //画素値の取得  
        //ルックアップテーブルによるポスタリゼーション  
        dst_img.at<uchar>(y, x) = lut[s];  
    }  
}  
  
cv::imshow(WINDOW_NAME_INPUT , src_img); //入力画像の表示  
cv::imshow(WINDOW_NAME_OUTPUT, dst_img); //出力画像の表示  
cv::waitKey(); //キー入力待ち (止める)  
  
return 0;  
}
```

grayImg.cppと異なる点

- 新しい定数定義

```
#define COLOR_NUM (256) //色値
```

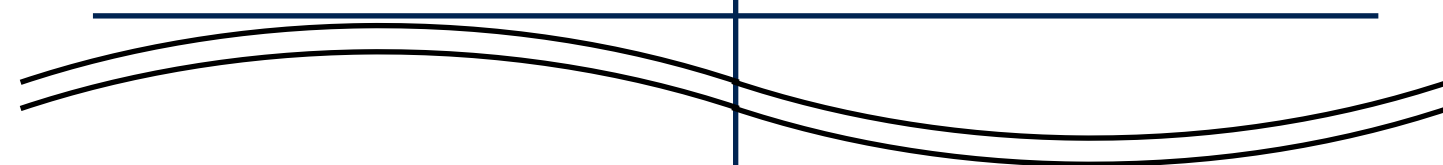
- 画像を最初からグレースケールで入力

```
//画像の入力 (グレースケール入力)
```

```
cv::Mat src_img = cv::imread(FILE_NAME, cv::IMREAD_GRAYSCALE);
```


ルックアップテーブル (LUT)

データ変換のための表 (配列)

| 入力値 | 出力値 |
|--|-----|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
|  | |
| 254 | 255 |
| 255 | 255 |

LUTの定義と作成

```
uchar lut[COLOR_NUM]; //ルックアップテーブル
//ルックアップテーブルの生成（4段階）
for (int i=0; i<COLOR_NUM; i++) {
    if (i <= 63) {
        lut[i] = 0;
    } else if (64 <= i && i <= 127) {
        lut[i] = 85;
    } else if (128 <= i && i <= 191) {
        lut[i] = 170;
    } else {
        lut[i] = 255;
    }
}
```



0以上63以下（64未満）→0に変換

LUTを用いた変換

//画像の走査

```
for (int y=0; y<dst_img.rows; y++) {  
    for (int x=0; x<dst_img.cols; x++) {  
        //画素値の取得  
        uchar s = src_img.at<uchar>(y, x);  
        //ルックアップテーブルによるポスタリゼーション  
        dst_img.at<uchar>(y, x) = lut[s];  
    }  
}
```

入力画素の値 s に応じて
出力画素値の変換 ($\text{lut}[s]$) を行う