画像処理による C++ 入門

道川 隆士 (michi@den.rcast.u-tokyo.ac.jp)

2013 年 10 月 7 日

1　はじめに

本文書は．画像処理を題材に C++ プログラムの作成法について学習する教材である．画像処理を題材として選んだのは，計算結果を画像として確認できるためであり，文字列で”Hello World”と打つよりは動機付けになると考えたためである．一部学生の要望に応じて Java によるサンプルコードも同梱している．

本文書の目標は，大きく分けて２つある．１つは，C++ のクラスを利用したプログラムをかけるようになることである．C++ は，その仕様の複雑さから，学習自体が面倒なものであった．しかし，標準的なデータ構造を定義した Standard Template Library (STL) や，線形代数ライブラリの Eigen など，開発の生産性を向上させるような便利なライブラリが提供されている現状を考えると，C++ を利用する術を知っておくことは有益である．もう一つは，簡単な画像処理アルゴリズムを理解できるようになることである．画像処理は身近になっているため，Photoshop や GIMP を使えばすぐに所望の結果が得られる．また，プログラム開発についてもオープンソースのライブラリ（例えば OpenCV など）を利用すれば，有名な画像処理アルゴリズムは，数行でかける場合が多い．本文書では，あえて１から書くことで画像処理のアルゴリズムを身をもって理解することを目指す．

1.1　ソースコードのダウンロード

ソースコードは，github のリポジトリで管理している．URL は以下の通りである．

- https://github.com/tmichi/image

2　サンプルファイル

サンプルファイルは，図 2 に示すような BMP 画像の各画素の値を反転させるプログラムが入っている．

(a) 入力画像　　(b) 計算結果

図 1　画素値反転プログラム

3　サンプルファイルのコンパイル

3.1　C++

C++ のソースコードは，cpp/フォルダに格納されている．g++ 等のコンパイラが使用できる場合，以下でコンパイルおよび実行確認できる．

```
$ g++ -O3 -Wall -o sample main.cpp
$ ./sample -i sample.bmp -o negate.bmp
```

ただし，-i オプションには入力ファイル名，-o オプションには出力をそれぞれ与える．フォルダ内で，編集するファイルは，main.cpp のみである．以下に，main.cpp と簡単な説明を示す．

```
1  #include <cstdlib>
2  #include "Image.h"
3  #include "Option.h"
4  int main ( int argc, char** argv )
5  {
6          char* input_file  = getOptChar ( argc, argv, "-i", NULL );
7          char* output_file = getOptChar ( argc, argv, "-o", NULL );
8
9          Image inImage;
10         if ( !inImage.read ( input_file ) ) return EXIT_FAILURE;
11         const int width  = inImage.getWidth();
12         const int height = inImage.getHeight();
13
14         Image outImage ( width, height );
15         for ( int y = 0 ; y < height ; y++ ) {
16                 for ( int x = 0 ; x < width ; x++ ) {
17                         Pixel p = inImage.getPixel ( x, y );
18                         // TODO : add your code here.
19                         const int r = 255 - p.getR();
20                         const int g = 255 - p.getG();
21                         const int b = 255 - p.getB();
22                         p.set ( r, g, b );
23                         outImage.setPixel ( x, y, p );
24                 }
25         }
26         if ( !outImage.write ( output_file ) ) return EXIT_FAILURE;
27         return EXIT_SUCCESS;
28 }
```

- 1 行目: C の場合は，stdlib.h であるが，C++ の場合は，cstdlib と書いた方がよい．
- 2-3 行目: 画像を扱うクラスとオプション (-i など）を処理する関数が格納されている．
- 6-7 行目: 引数から文字列を取り出す関数．1,2 個目の引数は，main 関数の引数である argc, argv をそのまま入れる．3 個目の引数は，オプションのキー，4 個目の引数はデフォルトの値である．数字を取り出したい場合は，getOptInt(int argc, char* argv, char* key, int defaultValue) を使う．詳しくは，Option.h を参照のこと．
- 9 行目: 画像格納クラス（大雑把にいうと構造体に関数がくっついたもの）．この時はサイズは 0 x 0 であり、空の状態である．
- 10 行目:BMP ファイルから画像を読み込む．引数にはファイル名を与える．bool 型 (true か false) を返す．失敗した場合，false を返すので，その場合は，EXIT_FAILURE(cstdlib で定義されている) を返す．
- 11-12 行目: 画像の幅 (width) と高さ (height) を取得する．int の前にある const は，変更を許可しないという意味の修飾子である．例えば，13 行目以降で， width = 0 と書くとコンパイルエラーを起こす．予期せぬ変更を防ぐ点で有用である．
- 14 行目: サイズを指定した画像オブジェクトの作成．この場合は，width x height のサイズで初期化されている．
- 18 行目: inImage.getPixel(x,y) で (x,y) におけるピクセルの値を取得．Pixel は画素値 (RGB) をもっているクラスである．
- 19-21 行目: p.getR(), p.getG(), p.getB() で画素の RGB 成分を取得．それぞれの最大値は 255 である．
- 22 行目: RGB 値をピクセルにセットする．
- 23 行目: 22 行目で作成したピクセルを画像にセットする．
- 26 行目: 画像を保存する．
- 27 行目: 成功したという値 (EXIT_SUCCESS) を返す．

以上に示す通り，基本的な処理は，ピクセルの色情報を画像から取得，各画素の取得，各画素をピクセルにセット，ピクセルを画像にセットという段階を経て行う．例えば，ピクセル処理は，19-21 行目を編集すればよい．

3.1.1 Java の場合

Java の場合のコンパイルと実行は以下の通り行う．

```
$ javac ImageProcessingSample.java
$ java ImageProcessingSample sample.bmp negate.bmp
```

Java のコードは, java フォルダ内にある．フォルダ内で編集するファイルは, ImageProcessingSample.java のみである．以下に，ImageProcessingSample.java を示す．説明は省略するが，基本的な構造は C++ と同じである．

```java
/**
 * Sample program for image processing in Java.
 * by Takashi Michikawa
 */
import java.io.*;
import java.awt.Color;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;

public class ImageProcessingSample {
        private BufferedImage _inputImage;
        private BufferedImage _outputImage;

    public ImageProcessingSample(final String filename) throws IOException {
        this._inputImage = ImageIO.read(new File ( filename ) );

        final int width  = this._inputImage.getWidth();
        final int height = this._inputImage.getHeight();
        this._outputImage = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);

        return;
    }

    public boolean compute() {
        final int width  = this._inputImage.getWidth();
        final int height = this._inputImage.getHeight();
        for( int y = 0 ; y < height; y++) {
            for( int x = 0 ; x < width ; x++) {
                final Color c = new Color(this._inputImage.getRGB(x,y));

                final int r = 255 - c.getRed();
                final int g = 255 - c.getGreen();
                final int b = 255 - c.getBlue();

                final Color nc = new Color (r, g, b);
                this._outputImage.setRGB(x, y, nc.getRGB());
            }
        }
        return true;
    }

    public void writeImage(final String filename ) throws IOException {
        ImageIO.write(this._outputImage, "jpeg", new File(filename));
        return;
    }

    public static void main ( String[] args ) throws IOException {
        if( args.length < 2 ) {
            System.out.println("Error. Insufficient arguments.");
            System.out.println("Usage : java ImageProcessingSample input.bmp output.jpg");
            return;
        }
        try {
            ImageProcessingSample sample = new ImageProcessingSample ( args[0] ) ;
            sample.compute();
            sample.writeImage(args[1]);

        } catch ( java.lang.Exception e) {
            e.printStackTrace();
        }
        return;
    }
}
```