画像工学 レポート

— pgm ファイルの解析とヒストグラム生成 —

IE5 (9) 片岡 駿之介

2016 年 10 月 27 日

1　課題

PGM は画像フォーマット PNM[1] の一種で，グレースケールを表現できる形式となっている．PGM は「portable graymap format」と呼ばれている．PNM には他にも 2 値画像，RGB 画像に対応したものも存在し，それぞれ PBM，PPM と呼ばれている．
本レポートでは，PGM 画像のヘッダから画像のメタデータを取得した後に画像データ部を解析し，それによってヒストグラムを生成する．

2　解析

今回の演習では，PGM ファイルを以下の手順で解析していくこととした．

1. 入力されたコマンドが正しいかチェック（正しくない場合は Usage を表示して終了）
2. コマンドで指定された PGM ファイルを読み込む
3. 読み込んだ PGM ファイルのヘッダ解析のスタート
4. マジックナンバーをチェック（マジックナンバーが異なる場合はその旨を表示して終了）
5. 画像の横のサイズを取得
6. 画像の縦のサイズを取得
7. 画像の最大輝度値設定を取得
8. 以降，画像データの読み込みをスタートし，ヒストグラムを生成

この解析によって得られたヒストグラムのデータを txt ファイルに保存し，gnuplot でプロットを行うことにより，ヒストグラムのグラフを出力する．

3 ソースリスト

ソースコード 1　histogram.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4
5  FILE *fp;
6
7  int width;
8  int height;
9  int max_value;
10
11 int status;
12 char buffer[128];
13
14 void image_open(char *file_name) {
15   if ((fp = fopen(file_name, "r")) == NULL) {
16    printf("file␣open␣error!!\n");
17    exit(-1); /* (3) エラーの場合は通常、異常終了する */
18  }
19   // printf("%s\n", file_name);
20 }
21
22 int main(int argc,char *argv[]) {
23
24   /* 入力コマンドのチェック */
25   if(argc == 1) {
26     printf("Usage␣:␣./histogram␣<pgm␣file␣name>\n");
27     exit(0);
28   } else if (argc == 2) {
29     image_open(argv[1]);
30   }
31
32   /* ヘッダ取得部 */
33   int ch;
34
35   while (status < 3) {
36     ch = getc(fp);
37
38     if(ch == '#') { // コメントのスキップ
39       while (( ch = getc(fp)) != '\n')
40         break;
41     }
42
43     if(ch == 'P') { // マジックナンバーのチェック
44       if(getc(fp) != '5') {
45         printf("Magic␣Number␣is␣wrong.\n");
46         break;
47       } else {
48         // printf("Magic Number is P5\n");
49       }
50     }
51
52     if(isdigit((unsigned char)ch)) { // 数値取得部
```

```
53          buffer[0] = ch;
54          int i=1;
55          while(1) {
56              char c = getc(fp);
57              if(isdigit((unsigned char)c)) {
58                buffer[i]=c;
59                i++;
60              } else
61                break;
62          }
63          buffer[i] = '\0';
64
65          switch (status) {
66            case 0: // width 取得部
67              width = atoi(buffer);
68              // printf("width=%d\n", width);
69              break;
70            case 1: // height 取得部
71              height = atoi(buffer);
72              // printf("height=%d\n", height);
73              break;
74            case 2: // max_value 取得部
75              max_value = atoi(buffer);
76              // printf("max_value=%d\n", max_value);
77              break;
78          }
79          status++; // 次のステータスへ
80        }
81      }
82
83      /* ヒストグラム生成部 */
84      int histogram[1024] = {};
85
86      while ((ch=getc(fp)) != -1) {
87        histogram[(int)ch]++;
88      }
89
90      for(int k=0; k <= max_value; k++) {
91        printf("%4d␣%5d\n", k, histogram[k]);
92      }
93    }
```

4　結果

本演習で与えられた 2 枚の PGM ファイル「cup.pgm」と「source.pgm」を図 1，図 2 に示す.

図 1　cup.pgm

図 2　source.pgm

「2. 解析」で示した手順で「cup.pgm」「source.pgm」を解析し，グラフにプロットしたものを図 3，図 4 に示す.

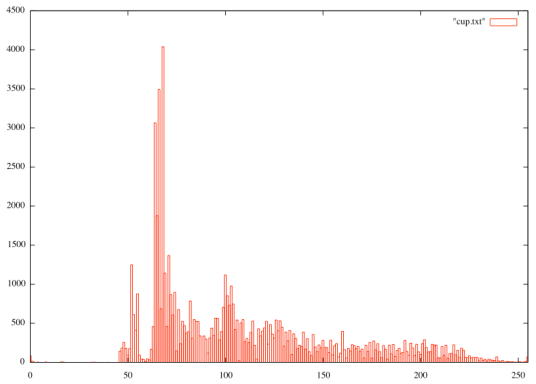

図 3　cup.eps のヒストグラム

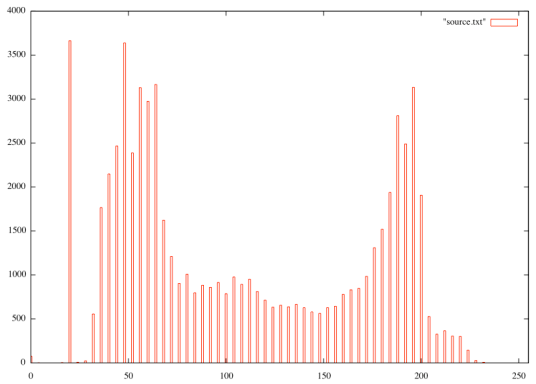

図 4　source.eps のヒストグラム

以下に，画像編集ソフト「GIMP」で 2 つの PGM ファイルを開き，ヒストグラムを観察したものを示す.

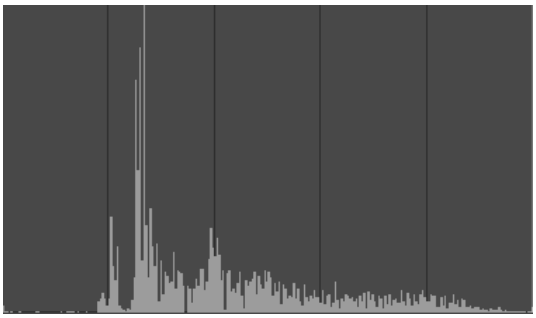

図 5　cup.eps のヒストグラム (GIMP)

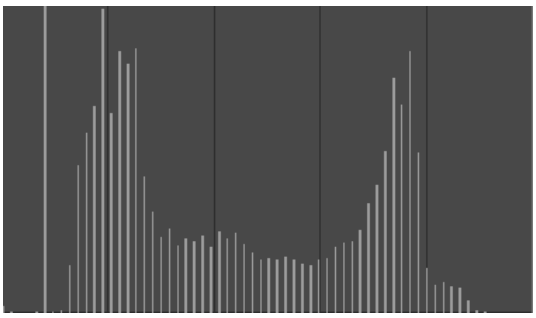

図 6　source.eps のヒストグラム (GIMP)

それぞれを見比べると，同一形状のヒストグラムが生成されていることがわかる.

5　考察

演習で与えられた PGM ファイルは，2 つともヘッダの記述フォーマットが少し異なっており，そこを如何に解決するかが本演習の肝だった.
PNM フォーマットのヘッダは一番はじめにマジックナンバーが来るという性質を利用してマジックナンバーを確定し，その後は順番に横幅，縦幅，最大輝度値という順番でヘッダを解析するという方針でプログラムを実装した．また，ヘッダに記載されている情報は改行・空白で区切られているため，コメント・改行・空白以外の部分で情報を一度文 1 字列として取得し，数値に変換するという手法を取った.
実際にプログラムを動かしてみると，正しく情報を更に取得できていることが確認できたため，正しく実装できたと考えている.

6　感想

最近は Ruby や Python といったスクリプト言語しか書いていなかったので，やはり C 言語での文字列操作はつらいという印象を受けた.

参考文献

[1] PNM (画像フォーマット) — https://ja.wikipedia.org/wiki/PNM_(画像フォーマット)