画像処理

電子情報工学科 5 年
学籍番号:17404

提出日:2021 年 12 月 20 日

1　目的

本科目達成度の確認のため課題を提出する。

2　予備知識

本課題では用いる画像はすべてラスター形式の画像である。特に無圧縮の BMP 形式を用いる。BMP 形式ではヘッダと呼ばれるファイル自体の情報などが記載された部分がある。本課題では各ピクセルの画素値の情報のみが必要であるためヘッダを削除する必要がある。ヘッダを削除したファイルの INC 形式として作成した変換プログラムによって画素値を変換する。その後取り除いたヘッダを追加することで既存の画像ビューアソフトによって閲覧することができる。図 1 に一連の処理の流れを示す。

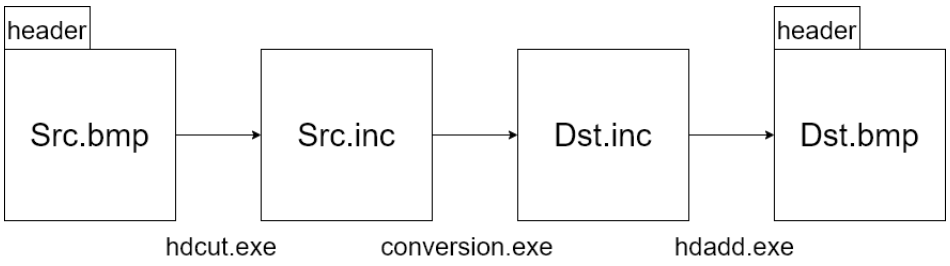

図 1: 画像変換の処理

3　課題 1(1)

本章では課題 1(1) について課題内容、結果およびプログラムを示す。

3.1　課題内容

次に課題内容を示す.

課題 1(1)

画像処理のデータ変換において，次の変換の入力と出力の関係を表す各関数をグラフで表せ.

1. 輝度調整
2. 等輝度線

3.2　理論

本課題では入力画像としてグレースケール画像を用いる。そのため 1 ピクセルごとにその濃度値を何らかの関数によって変換する。式 (1) に輝度調整の変換関数を示す。MAX,MIN は指定す

る閾値である。

$$f(x) = \begin{cases} 0 & x < \mathrm{MIN} \\ \frac{255(x-\mathrm{MIN})}{\mathrm{MAX}-\mathrm{MIN}} & \mathrm{MIN} \leqq x \leqq \mathrm{MAX} \\ 255 & x > \mathrm{MAX} \end{cases} \tag{1}$$

式 (2) に等輝度線の変換関数を示す。RANGE はグレースケールで塗り分ける輝度幅である。

$$f(x) = \begin{cases} x & \mathrm{RANGE} \leqq 1 \\ \frac{255 \cdot (x\%\mathrm{RANGE})}{(\mathrm{RANGE}-1)} & else \end{cases} \tag{2}$$

3.3 結果

図 2 に輝度調整を行う変換関数を示す。また、図 3 に等輝度線に変換する関数を示す。

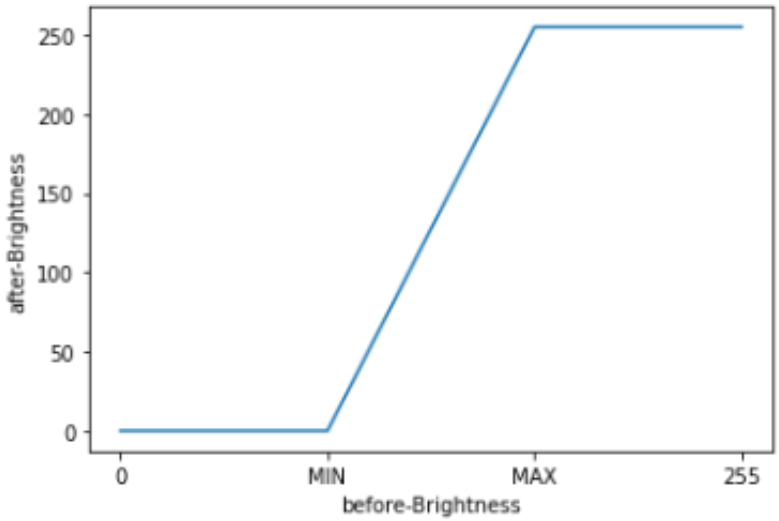

図 2: 輝度調整

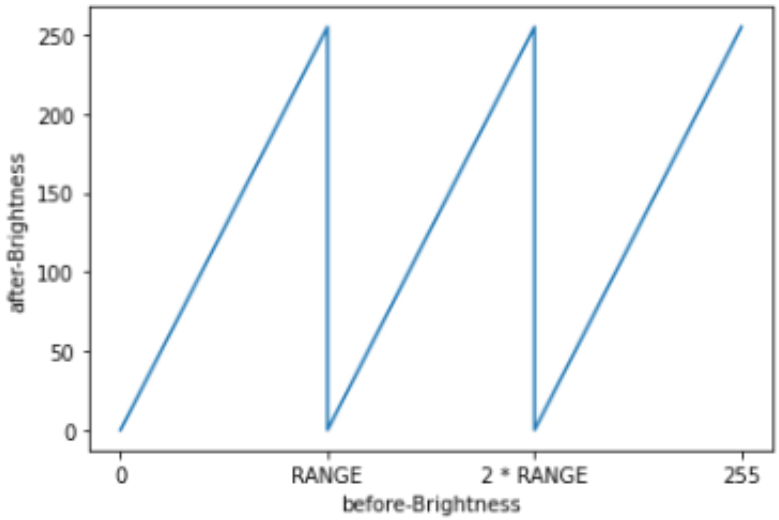

図 3: 等輝度線

3.4 考察

図 2 より指定した MIN 以下の濃度値では 0 に変換し、MAX 以上である場合は濃度値を 255 に変換する。MIN を超過し、MAX 未満の濃度値は線形的に変換する。等輝度線では図 3 のように指定した RANGE 間隔で線形的に濃度値が上昇する。

4 課題 1(2)

本章では課題 1(2) について課題内容、結果およびプログラムを示す。

4.1 課題内容

次に課題内容を示す.

課題 1(2)

次の変換について元の画像と作成した画像の濃度値ヒストグラムを示せ.

1. 輝度調整
2. 等輝度線

4.2 結果

図 4,5 にそれぞれ輝度調整を行う画像の濃度値ヒストグラム、輝度調整を行った画像の濃度値ヒストグラムを示す。MIN=75,MAX=170 である。

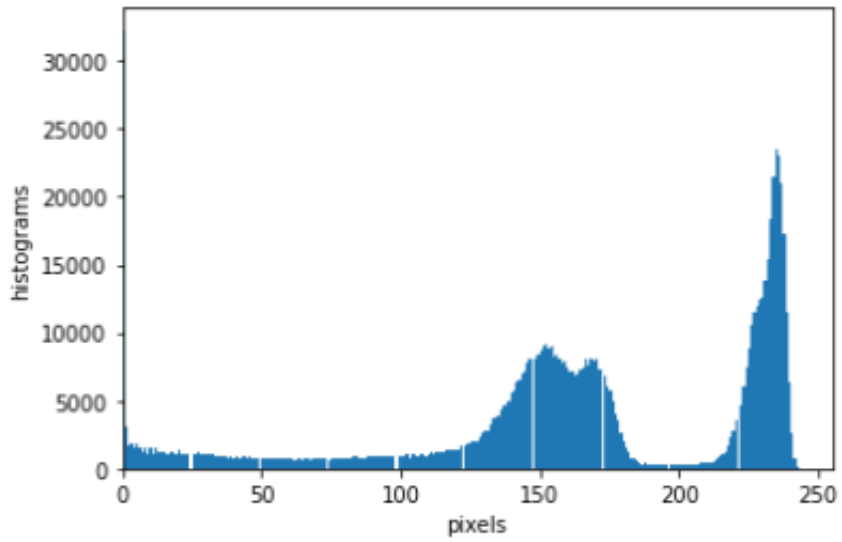

図 4: 元画像の濃度値ヒストグラム

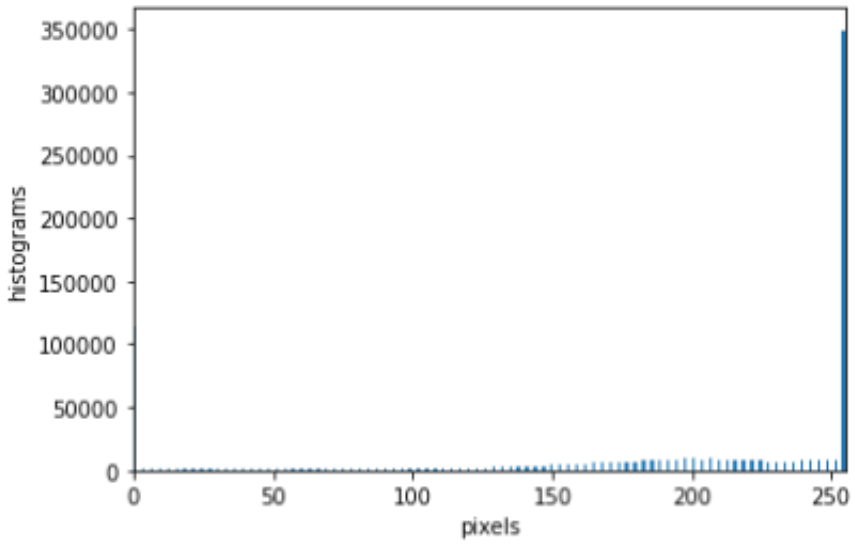

図 5: 輝度調整後の画像の濃度値ヒストグラム

図 6,7 にそれぞれ等輝度線に変換する画像の濃度値ヒストグラム、等輝度線画像の濃度値ヒストグラムを示す。RANGE=20 である。

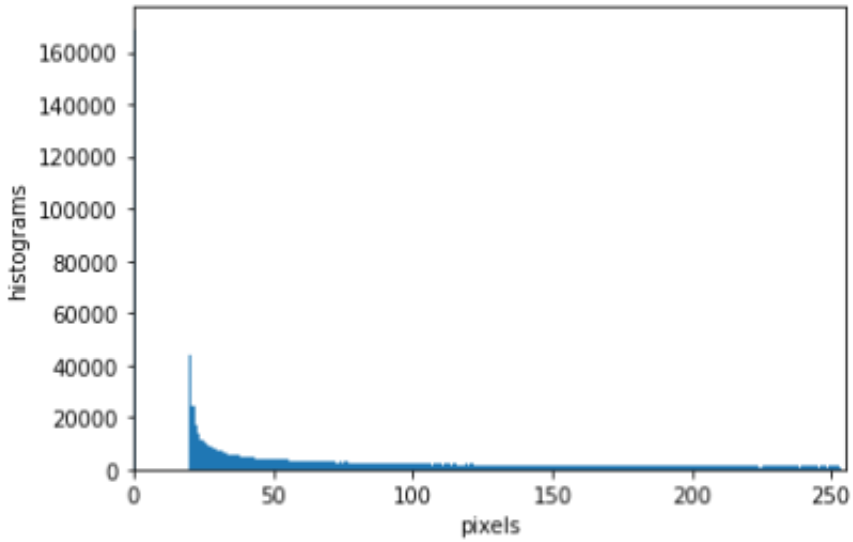

図 6: 元画像の濃度値ヒストグラム

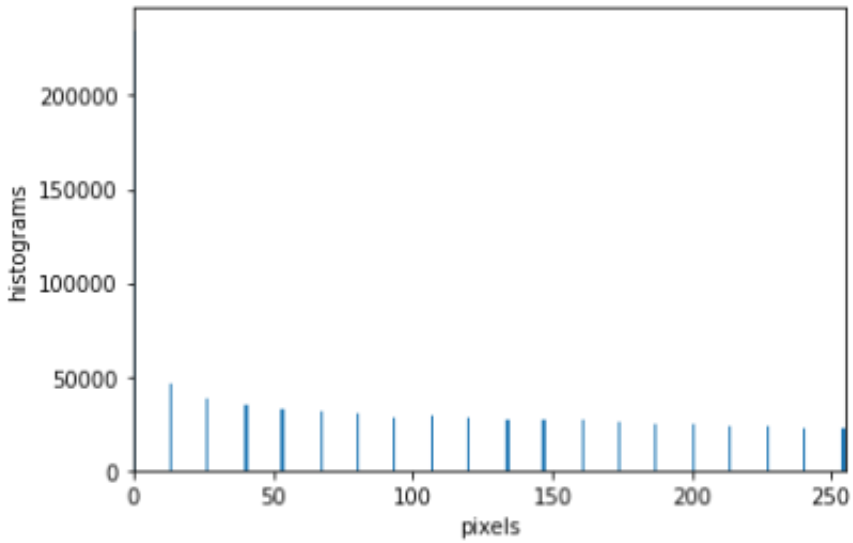

図 7: 等輝度線画像の濃度値ヒストグラム

4.3 考察

図 4,5 より元画像では濃度値 230 程度の値が多いことがわかる。そのため輝度調整を行った画像の濃度値ヒストグラムでは 255 が一番多いことがわかる。図 7 より等輝度線では RANGE によって分割された濃度値のみが存在することがわかる。

5 課題 1(3)

本章では課題 1(3) について課題内容、結果およびプログラムを示す。

5.1 課題内容

次に課題内容を示す.

課題 1(3)

次の変換を行うプログラムを示せ.

1. 輝度調整
2. 等輝度線

5.2 プログラム

プログラムリスト 1 に輝度調整を行うプログラムを示す。

プログラムリスト 1: 輝度調整

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  int main(int argc,char *argv[])
5  {
6      FILE *fin,*fout;
7      int c,k=255;
8      int min,max;
9
10
11     if(argc < 5){
12         puts("usage: brightnessControl [Input filename] [Output filename] [min]
               [max]");
13         exit(-1);
14     }
15
16     fin = fopen(argv[1],"rb");
17     fout = fopen(argv[2],"wb");
18     min = atoi(argv[3]);
19     max = atoi(argv[4]);
20
21     if(fin == NULL ){
22         printf("Can't open file %s\n",argv[1]);
23         exit(-1);
24     }
25
26     while((c = getc(fin)) != EOF){ /* input image data */
27             putc(c < min ? 0 : c > max ? k : (int)((k * c - k * min) / (max -
                   min)), fout); /* output the result */
28     }
29
30     fclose(fin);
31     fclose(fout);
32 }
```

コマンドライン引数として入出力ファイル、閾値となる min,max を指定する。getc 関数によって入力バッファから 1 バイトずつ読み取り int 型変数 c に格納する。putc 関数により出力バッファに書き込む。このとき式 (1) を基に三項演算子によって場合分けを行う。

プログラムリスト 2 に等輝度線を出力するプログラムを示す。

プログラムリスト 2: 等輝度線

```
#include<stdio.h>
#include<stdlib.h>

int main(int argc,char *argv[])
{
    FILE *fin,*fout;
    int c,k=255;
    int range;


    if(argc < 4){
        puts("usage: isohighlightLine [Input filename] [Output filename] [range
            ]");
        exit(-1);
    }
    fin = fopen(argv[1],"rb");
    fout = fopen(argv[2],"wb");
    range = atoi(argv[3]);

    if(fin == NULL ){
        printf("Can't open file %s\n",argv[1]);
        exit(-1);
    }
    if(fout == NULL)puts("out");
    while((c = getc(fin)) != EOF){ /* input image data */
        if(range <= 1){
            putc(c, fout);
        }else{
            putc((int)(k * (c % range) / (range - 1)), fout); /* output the
                result */
        }
    }

    fclose(fin);
    fclose(fout);
}
```

コマンドライン引数として入出力ファイル、輝度幅となる range を指定する。getc 関数によって入力バッファから 1 バイトずつ読み取り int 型変数 c に格納する。putc 関数により出力バッファに書き込む。このとき式 (1) を基に変換を行う。

5.3　結果

図 8,9 にそれぞれ輝度調整を行う画像、輝度調整を行った画像を示す。MIN=75,MAX=170 である。

図 8: 元画像

図 9: 輝度調整後の画像

図 10,11 にそれぞれ等輝度線に変換する画像、等輝度線画像を示す。RANGE=20 である。

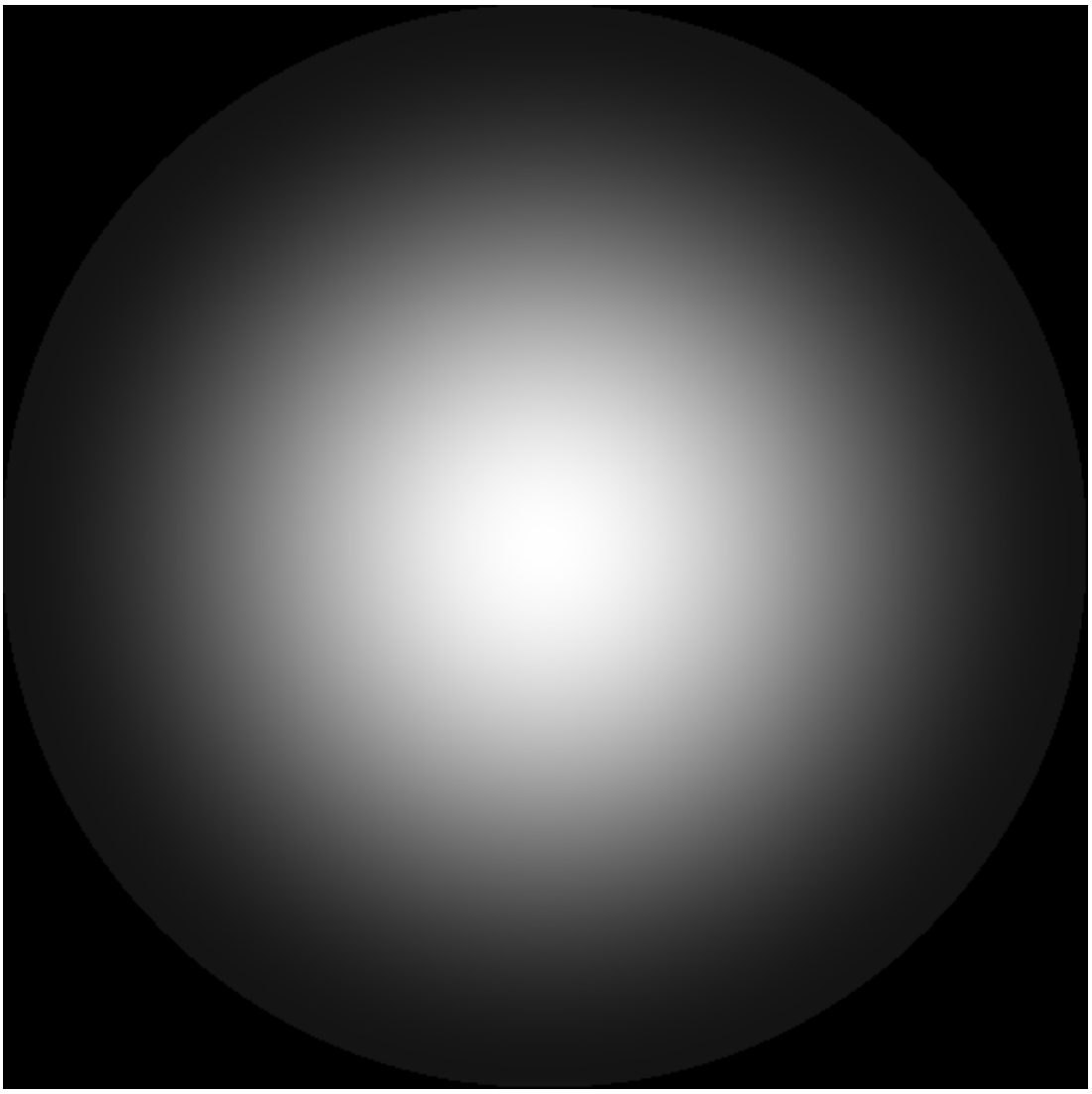

図 10: 元画像

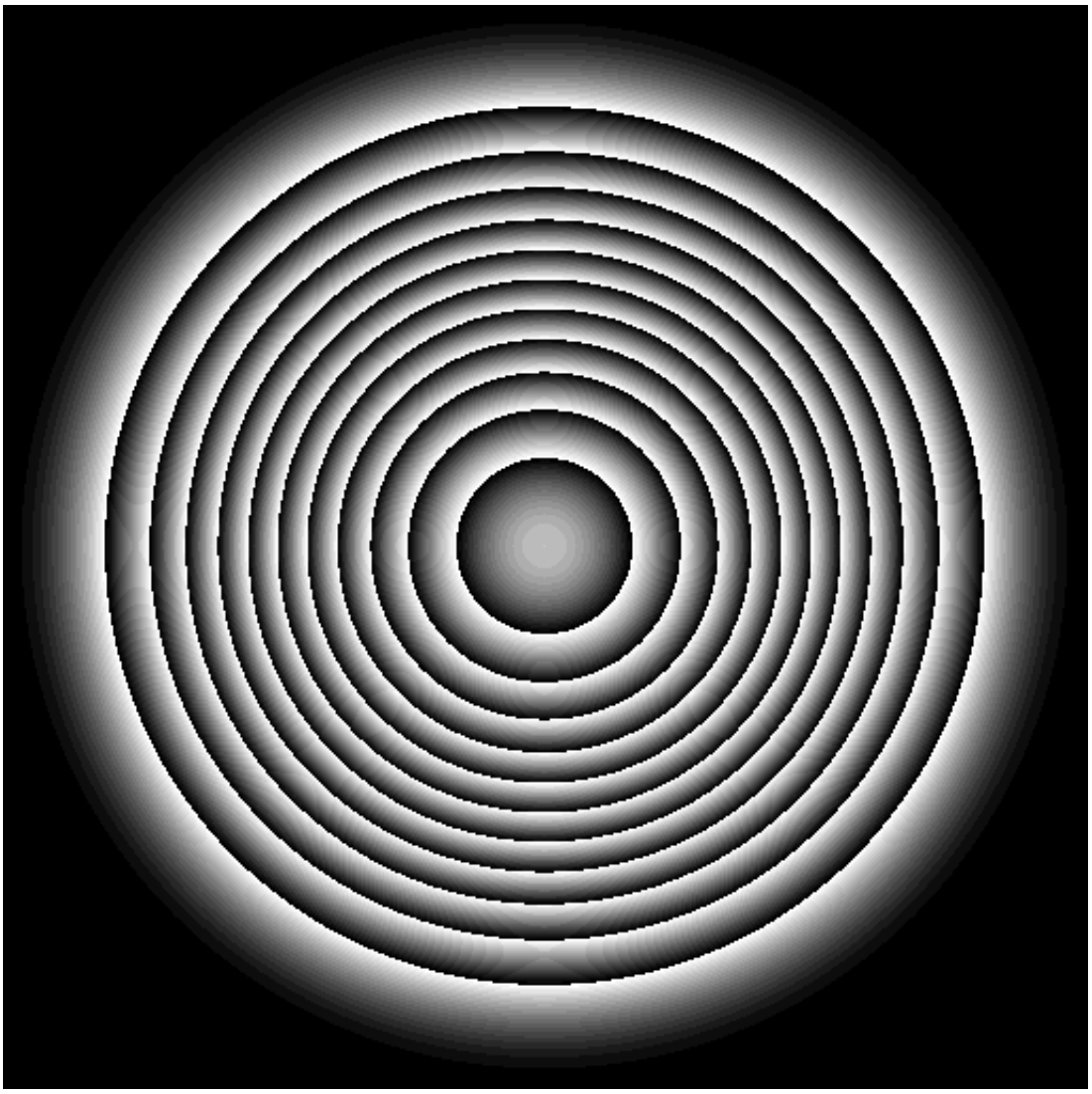

図 11: 等輝度線画像

5.4 考察

図 8,9 より元画像の空の部分は白く、木の部分は黒くなっていることがわかる。従って輝度調整ができていると言える。

6 課題 2(1)

本章では課題 2(1) について課題内容、結果およびプログラムを示す。

6.1 課題内容

次に課題内容を示す.

課題 2(1)

3×3 空間フィルタによる 8 近傍ラプラシアンを実行し，エッジ抽出を行え．

6.2 理論

6.3 プログラム

プログラムリスト 3 に空間フィルタによりラプラシアン 8 近傍を適用するプログラムを示す。

プログラムリスト 3: ラプラシアン

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<malloc.h>
4
5  int main(int argc,char *argv[])
6  {
7      FILE *fin,*fout;
8      int c,k=255;
9      int i,j;
10     int x,y;
11     int** src;
12     int ** dst;
13     int mask[3][3] = { {1, 1, 1},
14                        {1, -8, 1},
15                        {1, 1, 1}};
16     int offset[3] = {-1, 0, 1};
17
18     if(argc < 4){
19         puts("usage: laplacian [Input filename] [Output filename] [size]");
20         exit(-1);
21     }
22     fin = fopen(argv[1],"rb");
23     fout = fopen(argv[2],"wb");
24     int size = atoi(argv[3]);
25
26     //mat malloc
27     src = malloc(sizeof(int*) * size);
28     dst = malloc(sizeof(int*) * size);
29     for(i = 0;i < size;i++){
30         src[i] = malloc(sizeof(int) * size);
31         dst[i] = malloc(sizeof(int) * size);
32     }
33
34     if(fin == NULL ){
35         printf("Can't open file %s\n",argv[1]);
36         exit(-1);
37     }
38
39     i = 0;
```

```
40      j = 0;
41      while((c = getc(fin)) != EOF){ /* input image data */
42          src[j][i] = c;
43          j++;
44          if(j == size){
45              i++;
46              j = 0;
47          }
48      }
49
50      for(i = 0; i < size;i++){
51          for(j = 0; j < size; j++){
52              int sum = 0;
53              for(x = 0; x < 3; x++){
54                  for(y = 0; y < 3; y++){
55                      if(i + offset[x] >= 0 && i + offset[x] < size && j +
                            offset[y] >= 0 && j + offset[y] < size){
56                          sum += mask[y][x] * src[j + offset[y]][i + offset[x]];
57                      }
58                  }
59              }
60              dst[i][j] = sum;
61              putc(sum < 0 ? 0 : sum > k ? 255 : sum, fout);
62          }
63      }
64
65
66      fclose(fin);
67      fclose(fout);
68  }
```

コマンドライン引数として入出力ファイル、画像サイズとなる size を指定する。本プログラムでは画像の画素データを保持しておく必要があるため二次元配列を用いて各ピクセルの画素値を保存する。その際 malloc によって領域を確保する. その後各ピクセルとその近傍 8 つのピクセルにマスクを適用し、その積和を変換後のピクセルの画素値として putc 関数にて出力バッファに書き込む。このとき積和が 0 以下であれば 0 として出力する。また 255 以上であれば 255 として出力する。

6.4 結果

図 12,13 にラプラシアン 8 近傍の空間フィルタを適用する画像と適用した画像を示す。

図 12: 元画像

図 13: 空間フィルタ適用後の画像

6.5 考察

図 12,13 よりラプラシアン 8 近傍の空間フィルタを適用することでエッジ部分が白くなっていることがわかる。

7 課題 2(2)

本章では課題 2(2) について課題内容、結果およびプログラムを示す。

7.1 課題内容

次に課題内容を示す.

課題 2(2)

画像の先鋭化を行うプログラムを示せ。

7.2 理論

7.3 プログラム

プログラムリスト 3 に空間フィルタにより先鋭化を行うプログラムを示す。

プログラムリスト 4: 先鋭化

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>

int main(int argc,char *argv[])
{
    FILE *fin,*fout;
    int c,k=255;
    int i,j;
    int x,y;
    int** src;
    int ** dst;
    int mask[3][3] = { {-1, -1, -1},
                       {-1, 9, -1},
                       {-1, -1, -1}};
    int offset[3] = {-1, 0, 1};
    if(argc < 4){
        puts("usage: sharp [Input filename] [Output filename]");
        exit(-1);
    }
    fin = fopen(argv[1],"rb");
    fout = fopen(argv[2],"wb");
    int size = atoi(argv[3]);

    //mat malloc
    src = malloc(sizeof(int*) * size);
    dst = malloc(sizeof(int*) * size);
    for(i = 0;i < size;i++){
        src[i] = malloc(sizeof(int) * size);
        dst[i] = malloc(sizeof(int) * size);
    }

    if(fin == NULL ){
        printf("Can't open file %s\n",argv[1]);
        exit(-1);
    }

    i = 0;
    j = 0;
```

```
40      while((c = getc(fin)) != EOF){ /* input image data */
41          src[j][i] = c;
42          j++;
43          if(j == size){
44              i++;
45              j = 0;
46          }
47      }
48
49      for(i = 0; i < size;i++){
50          for(j = 0; j < size; j++){
51              int sum = 0;
52              for(x = 0; x < 3; x++){
53                  for(y = 0; y < 3; y++){
54                      if(i + offset[x] >= 0 && i + offset[x] < size && j +
                            offset[y] >= 0 && j + offset[y] < size){
55                          sum += mask[y][x] * src[j + offset[y]][i + offset[x]];
56                      }
57                  }
58              }
59              dst[i][j] = sum;
60              putc(sum < 0 ? 0 : sum > k ? 255 : sum, fout);
61          }
62      }
63
64
65      fclose(fin);
66      fclose(fout);
67  }
```

コマンドライン引数として入出力ファイル、画像サイズとなる size を指定する。本プログラムでは画像の画素データを保持しておく必要があるため二次元配列を用いて各ピクセルの画素値を保存する。その際 malloc によって領域を確保する. その後各ピクセルとその近傍 8 つのピクセルにマスクを適用し、その積和を変換後のピクセルの画素値として putc 関数にて出力バッファに書き込む。このとき積和が 0 以下であれば 0 として出力する。また 255 以上であれば 255 として出力する。

7.4 結果

図 14,15 にラプラシアン 8 近傍の空間フィルタを適用する画像と適用した画像を示す。

図 14: 元画像

図 15: 空間フィルタ適用後の画像

7.5 考察

図 12,13 より画像を先鋭化することでエッジ部分が鮮明に見えるようになった。

8 課題 2(3)

本章では課題 2(3) について課題内容、結果およびプログラムを示す。

8.1 課題内容

次に課題内容を示す.

課題 2(2)

ラプラシアンの空間フィルタを適用した結果の画像を，適当な閾値を設定して 2 値化し，輪郭線画像を作成せよ.

8.2 理論

式 (3) に輝度調整の変換関数を示す。MAX,MIN は指定する閾値である。

$$f(x) = \begin{cases} 0 & x < \text{threshold} \\ 255 & x > \text{threshold} \end{cases} \tag{3}$$

8.3 プログラム

プログラムリストに二値化プログラムを示す。

プログラムリスト 5: 二値化

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  int main(int argc,char *argv[])
5  {
6      FILE *fin,*fout;
7      int c,k=255;
8      int threshold;
9
10     if(argc < 4){
11         puts("usage: binarize [Input filename] [Output filename] [threshold]");
12         exit(-1);
13     }
14
15     fin = fopen(argv[1],"rb");
16     fout = fopen(argv[2],"wb");
17     threshold = atoi(argv[3]);
18
19
20     if(fin == NULL ){
21         printf("Can't open file %s\n",argv[1]);
22         exit(-1);
23     }
24
25     while((c = getc(fin)) != EOF){ /* input image data */
26
27             putc((c > threshold ? k : 0),fout); /* output the result */
28     }
29
30     fclose(fin);
31     fclose(fout);
32 }
```

コマンドライン引数として入出力ファイル、閾値となる threshold を指定する。getc 関数によって入力バッファから 1 バイトずつ読み取り int 型変数 c に格納する。putc 関数により出力バッファに書き込む。このとき式 (3) を基に三項演算子によって場合分けを行う。

8.4 結果

図 16,17 にそれぞれ輝度調整を行う画像、輝度調整を行った画像を示す。threshold=160 である。

図 16: 元画像

図 17: 二値化後の画像

8.5 考察

図 16,17 より閾値を 160 にすることで輪郭線画像を作成することができた。