# 画像・映像情報処理処理
# 第一回実習レポート

学籍番号：201811411
所属：情報学群情報メディア創成学類
氏名：加藤虎之介

2020 年 11 月 4 日

## 1　課題 1

### 1.1　平均値フィルタ

#### 1.1.1　プログラム

Code 1　avg_filter.c

```
1  #include <math.h>
2  #include <stdio.h>
3  #include "data_inout.h"
4
5  #define NPIXELS 256 // 画素数
6  #define N 3 // 窓の大きさ
7
8  int main(void)
9  {
10   unsigned char inputImg[NPIXELS][NPIXELS];
11   char input_filename[] = "data/lenad";
12   unsigned char outputImg[NPIXELS][NPIXELS];
13   char output_filename[] = "data/output_avg7x7";
14   int size = NPIXELS * NPIXELS;
15
16   // データを読み込む
17   read_image(inputImg, size, input_filename);
18
19   for (int i = 0; i < NPIXELS; i++)
20   {
21     for (int j = 0; j < NPIXELS; j++)
22     {
23       int val = 0;
```

```
24        int number = (2 * N + 1) * (2 * N + 1); // 窓に含まれるpixel数
25        for (int k = -N; k <= N; k++)
26        {
27          for (int l = -N; l <= N; l++)
28          {
29            // 窓に含まれるpixelを順次見ていく
30            int i2 = i + k;
31            int j2 = j + l;
32            if (i2 < 0 || i2 > NPIXELS - 1 || j2 < 0 || j2 > NPIXELS - 1)
33              // データが入っていない配列の要素を参照しようとしている時
34              number = number - 1; // 窓に含まれるpixel数を更新
35            else
36              val = val + inputImg[i2][j2];
37          }
38        }
39        // outputImg[i][j] = (unsigned char)((float)val / (float)number + 0.5);
40        outputImg[i][j] = (unsigned char)((float)val / (float)number);
41      }
42    }
43
44    // データを書き込む
45    write_image(outputImg, size, output_filename);
46  }
```

### 1.1.2 実験結果

- $3 \times 3$ の平均値フィルタをかけた結果

図 1　$3 \times 3$ の平均値フィルタ

- $5 \times 5$ の平均値フィルタをかけた結果

図 2　5 × 5 の平均値フィルタ

- 7 × 7 の平均値フィルタをかけた結果

図 3　7 × 7 の平均値フィルタ

#### 1.1.3　実験結果に対する考察

フィルタサイズを大きくするほど、雑音がよく除去されるが、エッジが滑らかになりボヤけた画像になった。

## 1.2　メディアンフィルタ

### 1.2.1　プログラム

Code 2　median_filter.c

```
1 #include <math.h>
2 #include <stdio.h>
3 #include "data_inout.h"
4
5 #define NPIXELS 256 // 画素数
6 #define N 3 // 窓の大きさ
7
8 // 配列内を昇順にsort する
9 void sort(int vals[], int count)
```

```
10 {
11   for (int i = 0; i < count - 1; i++)
12   {
13     for (int j = 1; j < count; j++)
14     {
15       if (vals[j - 1] > vals[j])
16       {
17         int tmp = vals[j - 1];
18         vals[j - 1] = vals[j];
19         vals[j] = tmp;
20       }
21     }
22   }
23
24   return;
25 }
26
27 // sort された配列から中央値を取得し、返す関数
28 int getMedian(int vals[], int count){
29   int median = 0;
30   if (count % 2 == 0)
31   {
32     // 要素数が偶数の場合
33     int i = count / 2;
34     median = vals[i] + vals[i - 1];
35     median /= 2;
36   }else{
37     // 要素数が奇数の場合
38     int i = (count - 1) / 2;
39     median = vals[i];
40   }
41   return median;
42 }
43
44 int main(void)
45 {
46   unsigned char inputImg[NPIXELS][NPIXELS];
47   char input_filename[] = "data/lenad";
48   unsigned char outputImg[NPIXELS][NPIXELS];
49   char output_filename[] = "data/output_median7x7";
50
51   // データを読み込む
52   read_image(inputImg, NPIXELS * NPIXELS, input_filename);
53
54   for (int i = 0; i < NPIXELS; i++)
55   {
```

```
56      for (int j = 0; j < NPIXELS; j++)
57      {
58        // 窓に含まれる画素を順次見ていく
59        int number = (2 * N + 1) * (2 * N + 1); // 窓に含まれるpixel数
60        int vals[number];
61        int index = 0;
62
63        for (int k = -N; k <= N; k++)
64        {
65          for (int l = -N; l <= N; l++)
66          {
67            int i2 = i + k;
68            int j2 = j + l;
69            if (i2 < 0 || i2 > NPIXELS - 1 || j2 < 0 || j2 > NPIXELS - 1)
70              // データが入っていない配列の要素を参照しようとしている時
71              number--; // 窓に含まれる画素数を更新
72            else
73            {
74              vals[index] = inputImg[i2][j2];
75              index++;
76            }
77          }
78        }
79
80        // sort
81        sort(vals, number);
82        // median
83        int median = getMedian(vals, number);
84
85        outputImg[i][j] = (unsigned char)median;
86      }
87    }
88
89    // データを書き込む
90    write_image(outputImg, NPIXELS * NPIXELS, output_filename);
91  }
```

### 1.2.2 実験結果

- $3 \times 3$ のメディアンフィルタをかけた結果

図４　３×３のメディアンフィルタ

- ５×５のメディアンフィルタをかけた結果

図５　５×５のメディアンフィルタ

- ７×７のメディアンフィルタをかけた結果

図６　７×７のメディアンフィルタ

### 1.2.3 実験結果に対する考察

フィルタサイズが大きくなる程、雑音の除去能力が高くなっている。また同一サイズの平均値フィルタに比べ、エッジの鋭さが保存されている。しかし、色の変化の滑らかさは失われ、色むらができている。

# 2 課題 2

## 2.1 ソーベルフィルタ

### 2.1.1 プログラム

Code 3 sobel_filter.c

```c
#include <math.h>
#include <stdio.h>
#include "data_inout.h"

#define NPIXELS 256 // 画素数
#define N 1 // 窓の大きさ

// 畳み込み積分をする関数
int integral(unsigned char pixels[][3], int kernel[][3])
{
  int val = 0;
  for (int i = 0; i < 2 * N + 1; i++)
  {
    for (int j = 0; j < 2 * N + 1; j++)
    {
      val += pixels[i][j] * kernel[i][j];
    }
  }
  return val;
}

int main(void)
{
  unsigned char inputImg[NPIXELS][NPIXELS];
  char input_filename[] = "data/lena";
  unsigned char outputImg[NPIXELS][NPIXELS];
  char output_filename[] = "data/output_sobel_02";

  // sobel filterのkernel
  int Gh[3][3] = {{-1, 0, 1}, {-2, 0, 2}, {-1, 0, 1}}; // 水平方向
  int Gv[3][3] = {{-1, -2, -1}, {0, 0, 0}, {1, 2, 1}}; // 垂直方向

  // データを読み込む
```

```
34   read_image(inputImg, NPIXELS * NPIXELS, input_filename);
35
36   for (int i = 0; i < NPIXELS; i++)
37   {
38     for (int j = 0; j < NPIXELS; j++)
39     {
40       unsigned char pixels[2 * N + 1][2 * N + 1]; // 窓に含まれる画素値
41
42       // pixels に値を格納する
43       for (int k = -N; k <= N; k++)
44       {
45         int ix = k + N; // 配列pixels の index
46         for (int l = -N; l <= N; l++)
47         {
48           int iy = l + N; // 配列pixels の index
49           int i2 = i + k;
50           int j2 = j + l;
51           if (i2 < 0 || i2 > NPIXELS - 1 || j2 < 0 || j2 > NPIXELS - 1)
52             // データが入っていない配列の要素を参照しようとしている時
53             pixels[ix][iy] = 0;
54           else
55             pixels[ix][iy] = inputImg[i2][j2];
56         }
57       }
58
59       // 畳み込み積分をした結果を得る
60       int fx = integral(pixels, Gh);
61       int fy = integral(pixels, Gv);
62       // フィルタの出力結果
63       const double coefficient = 0.20;
64       int g = sqrt(fx * fx + fy * fy) * coefficient;
65       if (g > 255)
66         g = 255;
67       outputImg[i][j] = (unsigned char)g;
68     }
69   }
70
71   // データを書き込む
72   write_image(outputImg, NPIXELS * NPIXELS, output_filename);
73 }
```

### 2.1.2 実験結果

- $coefficient = 0.20$ のゾーベルフィルタをかけた結果

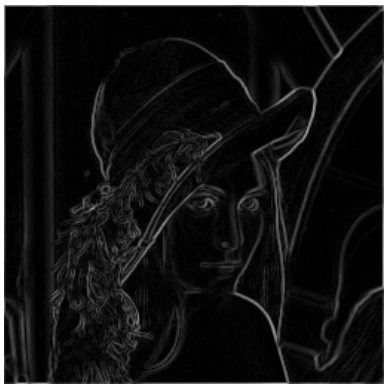

図 7　$coefficient = 0.20$ のゾーベルフィルタ

- $coefficient = 0.50$ のゾーベルフィルタをかけた結果

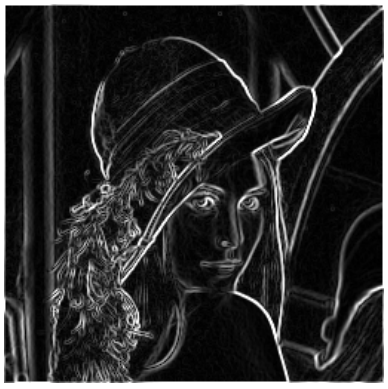

図 8　$coefficient = 0.50$ のゾーベルフィルタ

- $coefficient = 1.0$ のゾーベルフィルタをかけた結果

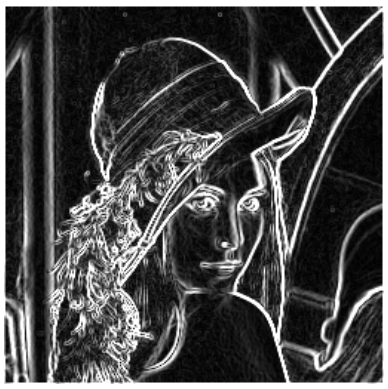

図 9　$coefficient = 1.0$ のゾーベルフィルタ

### 2.1.3 実験結果に対する考察

フィルタの出力値にかける係数（定数 $coefficient$）の値を変化させると、出力される画像が変化した。係数を大きくすると検出されるエッジがより強調されるようになる。