

# 西北农林科技大学

Northwest A&F University

## 信息工程学院

## 实验报告

实验名称: 图像增强

专业班级: 计算机 141

学号: 2014012537

姓名: 刘朝洋

指导教师: 杨龙

2016~2017 学年第一学期

使用 L<sup>A</sup>T<sub>E</sub>X 撰写于 2016 年 12 月 8 日

# 目录

实验一	图像平滑：以中值滤波为例	2
1	实验目的	2
2	实验要求	2
3	实验过程	2
4	实验结果	4
5	实验总结	4

# 实验一 图像平滑：以中值滤波为例

## 1 实验目的

1. 掌握图像平滑的基本概念，以及常用的方法；
2. 了解不同滤波方法的原理，并可以编程实现。

## 2 实验要求

1. 使用 C++ 编程实现中值滤波，但不能直接调用 OpenCV 中的滤波函数；
2. 将处理结果与 OpenCV 中的平滑处理函数 `medianBlur()` 的处理结果做对比。

## 3 实验过程

中值滤波的原理很简单，它把以某像素为中心的小窗口内的所有像素的灰度按从小到大排序，取排序结果的中间值作为该像素的灰度值。

首先是放置窗口，这里我们通过改变开始的索引值来放置窗口。

```
1 for (int m = 1; m < M - 1; ++m)
2     for (int n = 1; n < N - 1; ++n)
```

注意，这里是从第 1 个元素开始，而不是第 0 个元素；倒数第 1 个元素结束，而不是最后一个元素。问题就是我们无法从第 0 个元素开始，因为在这种情况下，过滤窗口的左半部分是空的。为了解决这个问题，我们需要在处理之前，对图像的扩展，方法如下图1.1所示：因此，在对图像进行中值滤波之前，先对图片进行扩展，代码实现如下所示：

```
1 // Allocate memory for signal extension
2 element* extension = new element[(N + 2) * (M + 2)];
3 // Check memory allocation
4 if (!extension)
5     return;
6 // Create image extension
7 for (int i = 0; i < M; ++i)
8 {
9     memcpy(extension + (N + 2) * (i + 1) + 1, image + N * i, N * sizeof(element));
10    extension[(N + 2) * (i + 1)] = image[N * i];
11    extension[(N + 2) * (i + 2) - 1] = image[N * (i + 1) - 1];
12 }
13 // Fill first line of image extension
14 memcpy(extension, extension + N + 2, (N + 2) * sizeof(element));
```

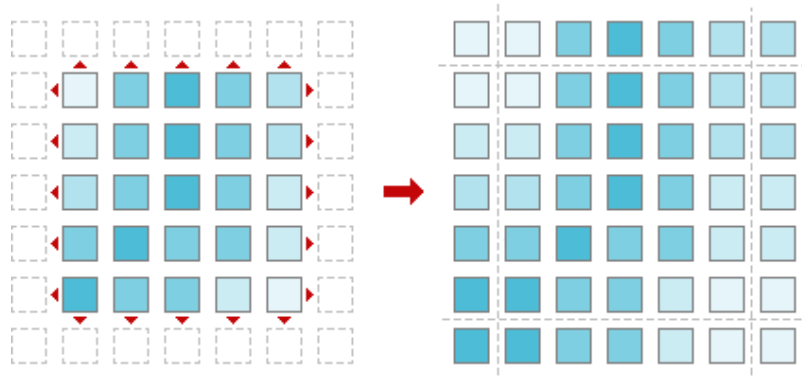


图 1.1: 图像扩展

```

15 // Fill last line of image extension
16 memcpy(extension + (N + 2) * (M + 1), extension + (N + 2) * M, (N + 2) * sizeof(element));
17 // Call median filter implementation

```

接下来第二步就是取出窗口中的元素，保存在数组`window[]`中。

```

1 // Pick up window elements
2 int k = 0;
3 element window[9];
4 for (int j = m - 1; j < m + 2; ++j)
5     for (int i = n - 1; i < n + 2; ++i)
6         window[k++] = image[j * N + i];

```

第三步就是把窗口中的元素排序。但是在这里我们将会使用一个代码优化的技巧：由于我们需要的仅仅是中值，所以只需要对一半的元素排序就可以了。

```

1 // Order elements (only half of them)
2 for (int j = 0; j < 5; ++j)
3 {
4     // Find position of minimum element
5     int min = j;
6     for (int l = j + 1; l < 9; ++l)
7         if (window[l] < window[min])
8             min = l;
9     // Put found minimum element in its place
10    const element temp = window[j];
11    window[j] = window[min];
12    window[min] = temp;
13 }

```

最后一步，将得到的中值作为该点像素的灰度值。

```

1 // Get result - the middle element
2 result[(m - 1) * (N - 2) + n - 1] = window[4];

```

上述过程即为中值滤波的完整的处理过程，为了方便使用，我们将上述处理过程写在两个函数中。处理前的扩展函数`medianfilter()`，负责对图像进行扩展，并将扩展好的图像传递给中值滤波函数，其函数参数为原图像指针`image`、结果图像指针`result`、图像宽度`N`、图像高度`M`；中值滤波函数`_medianfilter()`的工作就是对扩展好的函数进行滤波处理，其函数参数与`medianfilter()`相

同。为了将处理结果与 OpenCV 中的平滑处理函数`medianBlur()`的处理结果做对比，我们将函数`medianBlur()`的第三个参数卷积核的长度设置为 3，即与前面的窗口宽度值保持一致。

```
1 medianBlur(src, dst, 3);
```

## 4 实验结果

下图1.2为原始图像，该图像具有大量的椒盐噪声。



图 1.2: 椒盐噪声图像

下图1.6为不使用 OpenCV 库函数对图像进行中值滤波的处理结果，卷积核的大小为  $3 \times 3$ 。



图 1.3: 未使用 OpenCV 处理后的图像



图 1.4: 使用 OpenCV 库函数处理后的图像

图1.5为一张充满了抓痕噪声的图片，同样采用两种方式对其做中值滤波。

## 5 实验总结



图 1.5: 充满了抓痕的图像



图 1.6: 未使用 OpenCV 处理后的图像



图 1.7: 使用 OpenCV 库函数处理后的图像