实验8 抽象类

# 实验目的

了解为何使用抽象类，学习通过继承实现代码重用的机制和方法

掌握如何声明函数为纯虚函数

掌握如何利用纯虚函数，编写派生类的覆盖函数

# 实验内容

定义滤波器抽象类Filter，并派生两个具体的滤波器：类MeanFilter和类MedianFilter，分别完成对图像的均值滤波和中值滤波。在抽象类Filter中定义纯虚函数Filtering，在两个派生类中实现相应的函数。

在主函数中声明基类指针，通过该指针指向不同的派生类，从而使用不同的滤波器对象对图像进行滤波。并显示滤波后的图像，注意观察两种滤波器对图像滤波效果的差异。在构造滤波器时选择不同的filterSize，观察对滤波输出图像的影响。

**抽象类滤波器 Filter.h**

#ifndef FILTER\_H

#define FILTER\_H

#include “Image.h”

class Filter

{

public:

Filter(int size); //构造函数

virtual ~Filter(); //析构函数;

virtual Matrix Filtering(const Matrix &input) = 0; //滤波函数（纯虚函数）;

protected:

int filterSize;

};

#endif

**派生类 均值滤波器 MeanFilter.h**

#ifndef MEANFILTER\_H

#define MEANFILTER\_H

#include "Filter.h"

class MeanFilter : public Filter

{

public:

MeanFilter(int size);

virtual ~MeanFilter();

virtual Matrix Filtering(const Matrix &input); //均值滤波函数

};

#endif

**派生类 中值滤波器 MedianFilter.h**

#ifndef MEDIANFILTER\_H

#define MEDIANFILTER\_H

#include "Filter.h"

class MedianFilter : public Filter

{

public:

MedianFilter(int size);

virtual ~MedianFilter();

virtual Matrix Filtering(const Matrix &input); // 中值滤波器函数

};

#endif

**请完成MeanFilter.cpp、MedianFilter.cpp及主函数文件**

**使用滤波器对图像进行滤波的参考代码**

Image img("Lena\_gaussian.bmp"或者"Lena\_salt\_and\_pepper.bmp");

Filter \*filter = NULL;

filter = new MeanFilter(5);

Image result\_mean;

result\_mean = filter->Filtering(img);

result\_mean.WriteBMP("Mean.bmp");

delete filter;

filter = new MedianFilter(5);

Image result\_median;

result\_median = filter->Filtering(img);

medianfilter.WriteBMP("Median.bmp");

delete filter;

# 实验要求

完成上述代码，并能得到正确的结果图像，对结果进行比较。

实验附带了两幅添加了不同噪声的图像（高斯白噪声Lena\_gaussian.bmp和椒盐噪声Lena\_salt\_and\_pepper.bmp）。

设计简单的控制台交互，针对不同的图像，更改滤波器大小，观察滤波效果。

# 说明

* 注意，Filtering函数的输入是const的Matrix对象引用，在Filtering函数内部需要获取图像高度宽度的函数也需要重载一个const类型的函数，否则编译不会通过。比如：

int Height() const {return height;}

* 成员变量filterSize记录的是滤波器的大小，一般是奇数，定义在多大的图像块上进行操作，比如filterSize=3，则在输入图像的每一个像素的3\*3区域内进行滤波操作。以3\*3的均值滤波器为例，假设当前操作图像以像素(i,j)为中心的3\*3区域为：

|  |  |  |
| --- | --- | --- |
| 15 | 12 | 87 |
| 25 | 24 | 40 |
| 54 | 21 | 46 |

像素(i,j)的像素值为24，它的3\*3邻域的像素值如上所示，那么均值滤波的操作是：对这个3\*3邻域的像素求均值：(15+12+87+25+24+40+54+21+46)/9=36。那么输出图像的像素(i,j)处的像素值为36。

**针对图像中的每个像素做上述操作，就完成了均值滤波。在编程实现时要特别注意，图像四条边上和四个角的像素应该如何处理；以及计算过程中不能覆盖掉原始图像的像素值，否则会影响其他像素的计算。**

中值滤波的操作类似，只不过把求均值变成了求中值，即把filterSize\*filterSize大小区域内的像素的像素值按照大小排列，结果是位于中间的那个值。比如上例中，如果用中值滤波，那么按像素值从小到大排列后的像素值是：12, 15, 21, 24, 25, 40, 46, 54, 87。中值（中间的值）是25，所以，滤波后的输出图像的像素(i,j)的像素值为25。

filterSize是可变的，表示滤波操作的空间尺度，选取不同的值所产生的效果不一样，请在实验中试着产生不同大小的滤波器，看看对图像有什么影响。