机器视觉技术 第三次实验

实验题目 1——卷积

实验目的：

对特征图像进行卷积运算。

实验原理：

卷积核在输入的特征图像上滑动，通常是沿着从左到右、从上到下的次序进行，来提取输入数据中的特征。这个过程可以理解为对输入数据进行加权运算，以突出或抑制某些特定的特征。

实验步骤：

1. 移位：卷积核在特征图像上进行移位操作。
2. 相乘与相加：将特征图像与卷积核在相同位置上的元素进行乘法运算，得到一个新的矩阵。接着，将该矩阵的所有元素值进行相加，得到对应位置的值。

程序代码：

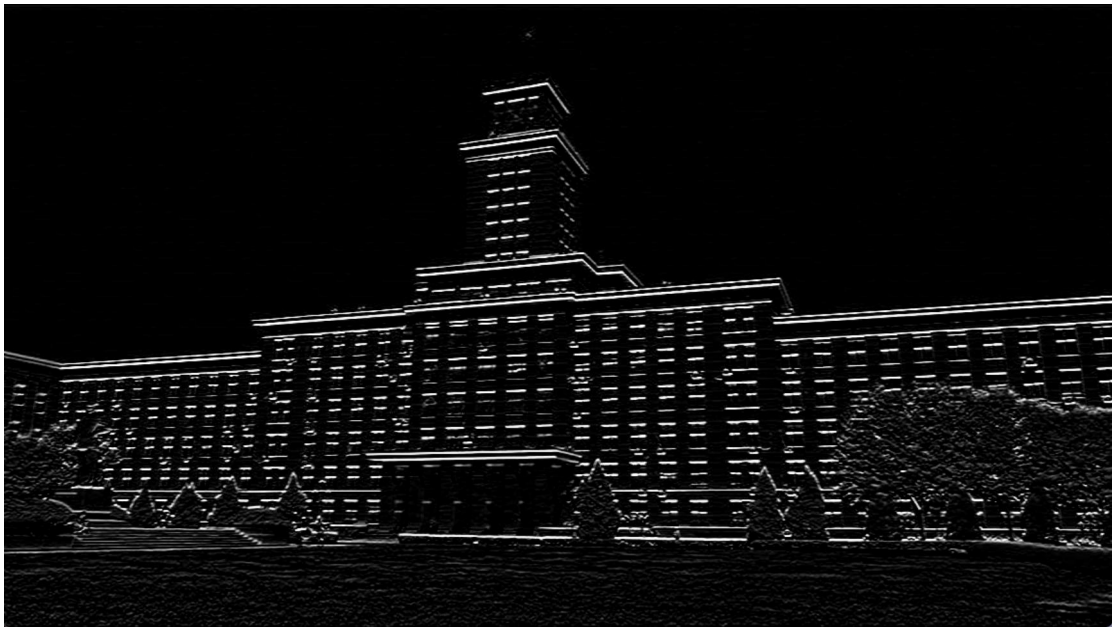
```
#include <iostream>
#include <fstream>
#include "opencv2/opencv.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdio.h>
using namespace cv;
using namespace std;

Mat myConv(Mat img)
{
    // 卷积图像
    Mat Conv_img;

    // 定义卷积核
    Mat kernel = (Mat_<float>(3, 3) <<
         1,  1,  1,
         0,  0,  0,
        -1, -1, -1);

    // 应用卷积
    filter2D(img, Conv_img, img.depth(), kernel);

    // 返回卷积结果
    return Conv_img;
}
```

```
void main()
{
    Mat input = imread("testimg.jpg");

    // 彩色图转为灰度图
    Mat gray;
    cvtColor(input, gray, COLOR_BGR2GRAY);

    // 图像卷积处理
    Mat Conv_img = myConv(gray);

    imshow("Conv_img", Conv_img);
    waitKey(0);
}
```

实验结果显示：

测试图片：

结果 1（卷积核如上）：



结果 2（卷积核为 3x3 的 1/9 矩阵）：

实验分析总结：

以下是通过 Matlab 得到的结果图像：

通过对比以上三个结果，可知，对同一特征图像，使用不同的卷积核，会有不一样的结果。相比于全正的卷积核，有正有负的卷积核卷积的结果更能体现原图的轮廓。

实验题目 2——计算积分图像

实验目的：

通过累加像素点左上角的像素值之和，得到图像的积分图像。

实验原理：

积分图像计算图像中每个像素点左上角所有像素值的和。在处理图像时，只需对图像进行一次扫描，就能快速获取任意矩形区域内像素值的累加和。在计算机视觉和图像处理领域应用广泛，在目标检测、特征提取等任务中可以提高效率。

实验步骤：

1. 用 s(i, j)表示行方向的累加和，初始化 s(i, 0)=0。
2. 用 ii(i, j)表示积分图像，初始化 ii (i, 0)= 0。
3. 逐行扫描图像，递归使用下述公式计算每个像素(i, j)行方向的累加和 s (i, j)与积分图像 ii(i, j)的值。

s（i, j)= s（i, j-1)+ f(i, j)
ii(i, j)= ii(i, j-1)+ s(i, j)
4. 扫描一遍图像，完成积分图像构建。

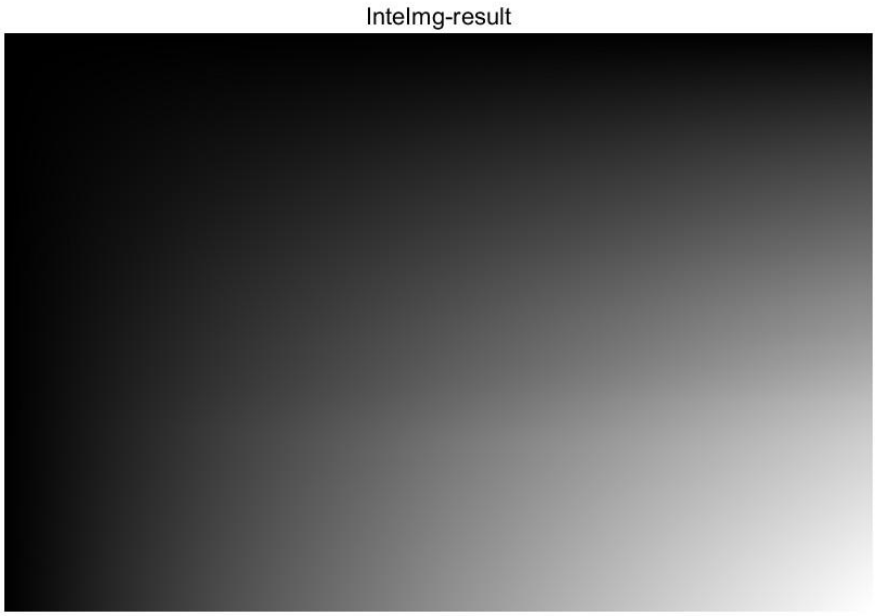
程序代码：
main.m
```
clc;clear all;close all;

% 读入图像
img = imread('testimg2.jpg');

% 将读入的彩色图像转换为灰度图像
img = im2gray(img);

% 调用函数
InteImg_result = MyInteImg(img);

% 输出图像 InteImg_result,[]
figure;
imshow(InteImg_result,[]);
title('InteImg-result');
```

MyInteImg.m
```
% 定义函数 MyInteImg，参数为二值图像 img
function MyInteImg_result=MyInteImg(img)

img = double(img);

% 获取 bw 的行数 rows 和列数 cols
[rows, cols] = size(img);

% 先行方向累加和
s = zeros(rows, cols);
for i = 1:rows
    s(i,1) = img(i,1);
    for j = 2:cols
        s(i,j) = s(i,j - 1) + img(i,j);
    end
end

% 再列方向累加和
ii = zeros(rows, cols);
for j = 1:cols
    ii(1,j) = s(1,j);
```

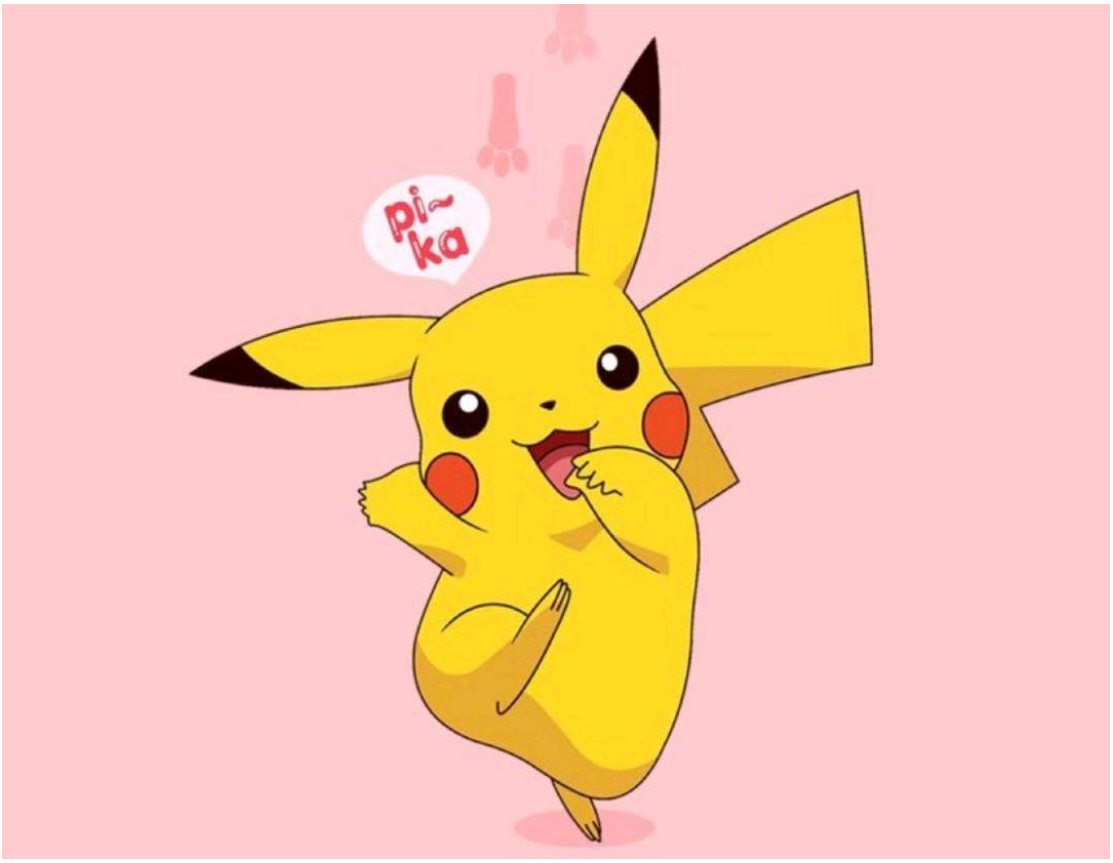
```
    for i = 2:rows
        ii(i,j) = ii(i - 1,j) + s(i,j);
    end
end

% 返回值
MyInteImg_result = ii;

end
```

实验结果显示：
测试图片 1：

结果 1：

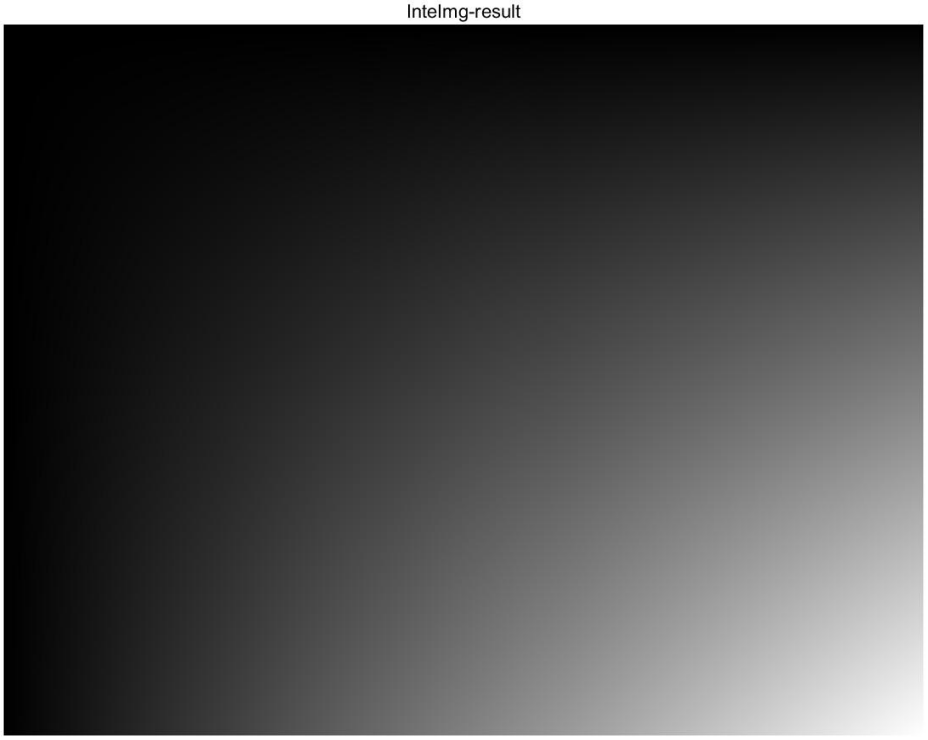


测试图片 2：



结果 2：

IntеImg-result



实验分析总结：

通过比较俩张测试图片的积分图像可知，图 1 比图 2 变化更快，说明其对比度更高。

疑问：

没有“`img = double(img);`”时，为什么会输出不一样的图像？