

机器视觉技术 第五次实验

实验题目——使用差分滤波器的边缘检测

实验目的：

通过差分滤波的方法实现图像边缘检测，并比较不同算子的性能。

实验原理：

边缘检测是图像处理中提取图像边缘信息的关键技术，常用方法为差分滤波。通过使用 Sobel、Laplace 和 Canny 等不同算子对图像中相邻像素灰度值进行差分运算，可以突出图像中的高频成分，从而实现边缘检测。

实验步骤：

1. 预处理：对图像进行平滑处理，减少噪声对边缘检测的影响。
2. 差分滤波，边缘检测：使用不同算子对差分图像进行边缘检测。

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 convolution(Mat img, Mat kernel) {
    // 获得尺寸
    auto Row = img.rows;
    auto Col = img.cols;
    auto row = kernel.rows;
    auto col = kernel.cols;

    // 转化
    img.convertTo(img, CV_32F);
    normalize(img, img, 0, 1, NORM_MINMAX, CV_32F);

    // 返回值初始化
    Mat Filter2D(Row, Col, CV_32F);
    Filter2D.setTo(Scalar(0));

    // 进行卷积
    for (int i = row / 2; i < Row - row / 2; i++) {
```

```

        for (int j = col / 2; j < Col - col / 2; j++) {
            for (int m = 0; m < row; m++) {
                for (int n = 0; n < col; n++) {
                    Filter2D.at<float>(i, j) += img.at<float>(i - col / 2 + m, j - row
/ 2 + n) * kernel.at<float>(m, n);
                }
            }
        }

// 归一化
normalize(Filter2D, Filter2D, 0, 255, NORM_MINMAX, CV_8U);
return Filter2D;
}

Mat myEdgeDetect(Mat img)
{
    // 定义卷积核
    Mat kernel = (Mat_<float>(3, 3) <<
        -1, -2, -1,
        0,  0,  0,
        1,  2,  1);

    // 进行卷积
    Mat EdgeImg = convolution(img, kernel);

    // 返回结果
    return EdgeImg;
}

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

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

    // 利用线性滤波, 使用差分滤波器的边缘检测
    Mat EdgeImg = myEdgeDetect(gray);

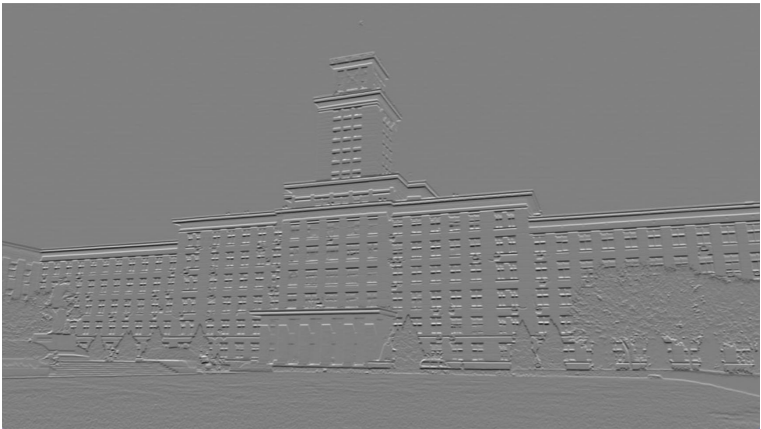
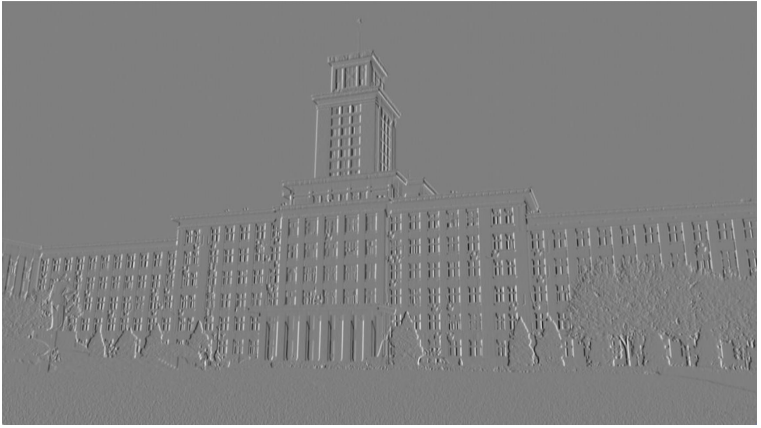
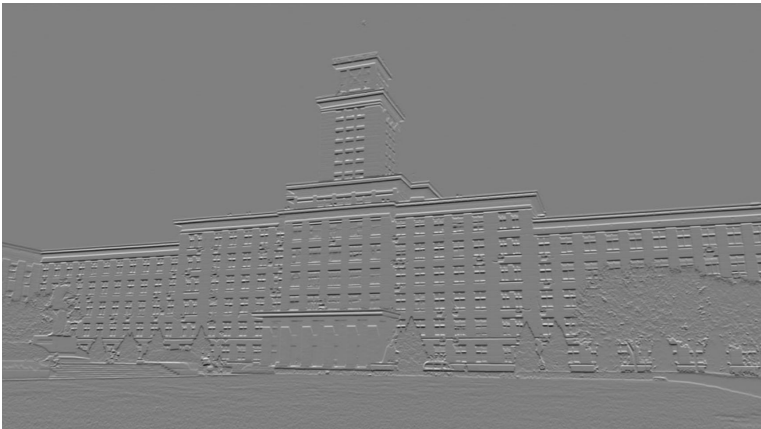
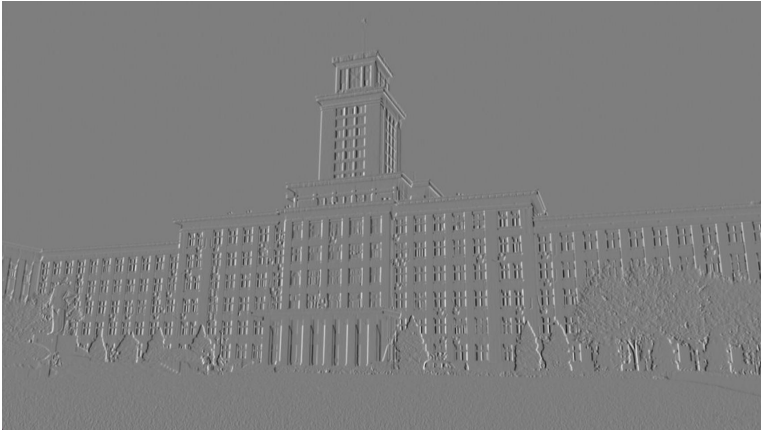
    imshow("EdgeImg", EdgeImg);
    waitKey(0);
}


```

实验结果显示：
测试图片：



算子		图像
Robert 算子	135° $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	
	45° $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	

Prewitt 算子	水平 $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	
	竖直 $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	
Sobel 算子	水平 $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	
	竖直 $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	

Laplace 算子	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
---------------	--	--

实验分析总结：

以上得到了四种算子在不同角度对同一图像的边缘检测结果,它们各有特点,适合不同的应用场景。

以下还尝试了利用 Sobel 算子连续进行水平、竖直两个维度的边缘检测。

