实验目的：

使用差分滤波器进行边缘检测

实验原理：

边缘是图像上灰度的不连续点，或者是灰度变化剧烈处。边缘的数学表达：信号一阶微分最大值/两阶微分过零点。实际信号都是有噪声的，按照一阶微分最大值或二阶微分过零点的方法检测边缘，可能检测出噪声引起的假边缘。解决方法是先对原始图像f(x,y) 进行平滑滤波：再对滤波后的信号g(x,y)求一阶或两阶微分以检测边缘。

实验步骤：

1. 将图像转化为灰度图。
2. 对灰度图进行高斯滤波。
3. 对滤波后的图像进行边缘检测。

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 sPrewitt(Mat input\_img)

{

Mat output\_img;

float val\_x[9] = { -1,0,1,-1,0,1,-1,0,1 };//建立Prewitt算子

Mat ker\_x = Mat\_<float>(3, 3, val\_x);

float val\_y[9] = { -1,-1,-1,0,0,0,1,1,1 };

Mat ker\_y = Mat\_<float>(3, 3, val\_y);

output\_img = input\_img;

int height\_x = ker\_x.rows;

int width\_x = ker\_x.cols;

int height\_y = ker\_y.rows;

int width\_y = ker\_y.cols;

for (int row = 1; row < input\_img.rows - 1; row++)

{

for (int col = 1; col < input\_img.cols - 1; col++)

{

float G\_X = 0;

for (int h = 0; h < height\_x; h++)

{

for (int w = 0; w < width\_x; w++)

{

G\_X += input\_img.at<uchar>(row + h - 1, col + w - 1) \* ker\_x.at<float>(h, w);

}

}

float G\_Y = 0;

for (int h = 0; h < height\_y; h++)

{

for (int w = 0; w < width\_y; w++)

{

G\_Y += input\_img.at<uchar>(row + h - 1, col + w - 1) \* ker\_y.at<float>(h, w);

}

}

output\_img.at<uchar>(row, col) = saturate\_cast<uchar>(abs(G\_X) + abs(G\_Y));

}

}

return output\_img;

}

void RGBtoGRAY(Mat& src, Mat& des) { //彩色转灰度的函数

des.create(src.rows, src.cols, CV\_8UC1);

for (int r = 0; r < src.rows; r++) {

for (int c = 0; c < src.cols; c++) {

Vec3b& m = src.at<Vec3b>(r, c);

int gray = (m[2] \* 30 + m[1] \* 59 + m[0] \* 11 + 50) / 100;

des.at<uchar>(r, c) = gray;

}

}

}

void main()

{

Mat input = imread("2233.jpg");

Mat gray;

RGBtoGRAY(input, gray); //彩色图转为灰度图

namedWindow("gray",0);

imshow("gray", gray);//测试是否转化成功

Mat tmp;

Mat result;

GaussianBlur(gray, tmp, Size(15, 15), 1, 9);// 对图像进行高斯滤波

namedWindow("tmpHist", 0);

imshow("tmpHist", tmp);

result = sPrewitt(tmp);

namedWindow("result", 0);

imshow("result", result);

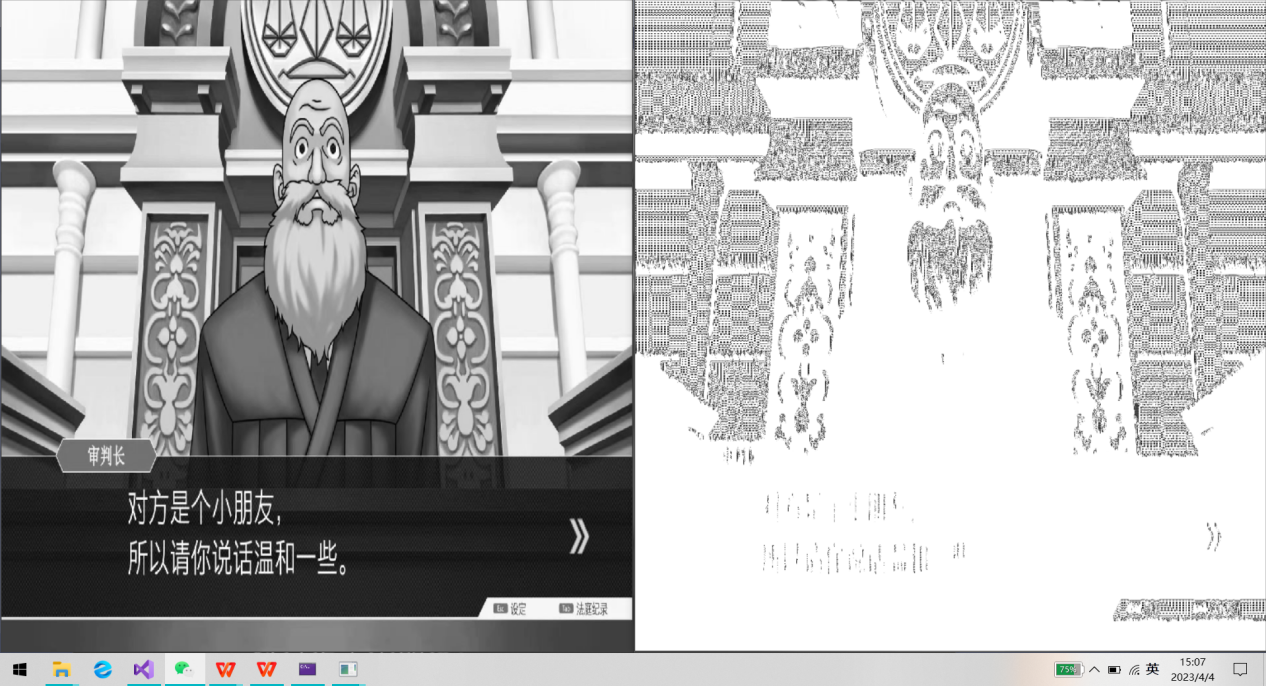
waitKey(0);

}

实验结果显示：



（上为原灰度图和进行高斯滤波后的图像）



（上为原灰度图和进行边缘检测后的图像）

实验分析总结：

体会了边缘检测的原理以及使用高斯滤波去除噪声的意义。提高了找代码bug的能力。