论学线代的必要性

第一次大作业报告

*** | 高级语言程序设计 | 2019 年 12 月 4 日

195****

新生院工科试验班（信息类）17 班

lena 老了依旧风姿绰约

——陈宇飞

设计思路与功能描述

0. 灵感来自于卡西欧 991CN 计算器，上海学生最后的尊严。
1. 为了优化输入体验，我想：我来模拟卡西欧计算器那种
   可以按上下左右方向键的输入吧！没想到一写就是半天。
   （真是越来越期待扫雷大作业了呢（完全不）
2. 支持空格和右键和回车作为右移光标的方式；支持触碰行末自动换行；
3. 为了让矩阵显示优美，当数据（包括负号）超过四个字符位时，将只显示后四位；转而，在矩阵的左上角会显示当前位置的完整值。
4. 事实上，对于大矩阵(10*10)以上如此大的矩阵一般并不会手动一个一个输入。因此做了一个文件式批量导入的功能来完善这个程序。该程序支持右键将矩阵直接复制黏贴来批量输入矩阵。
5. 同理，对计算得到的结果也做了一个导出功能。比较简单，直接存在 D 盘根目录下的 answer.txt
6. 关于功能 7：就跟着要求一步一步来。见招拆招慢慢做。努力有了结果真是太好了 w
   具体思路见下一栏目功能 7 和整理
8. 没有 7777777
9. 关于功能 8：为了确定自己的算法的适应性，除了 OJ 的图包外还测试了其它图片
10. 经过试验，OTSU 算法得到的结果对风景照更为良好，对 OJ 图包和二次元图像（轮廓和背景对比鲜明的图）效果不尽如人意，因此最终没有采用 OTSU 算法，自己写了一个
    （对比图找不到了嘤）
11. 功能 8 的设计思路：（详见代码注释）
    a) 二值化时，把中位数作为阈值
    b) 求剩下的（中位数以上）的像素点的亮度平均值

c) 据此判断图像是否需要提亮，提亮倍数向下取整，最小值为 1 倍（不提亮）
d) 小于中位数的抹黑
e) 其余图像未抹黑部分先按倍数提亮
f) 将虽然高于中位数但亮度仍然过低的再抹黑
g) 计算非黑色连通块的面积大小（即像素点个数）//BFS 算法避开栈存储，避开爆栈
若连通块面积过小则抹黑（用来去处非中心物体（和头皮屑））
h) 计算黑色连通块的面积大小（即像素点个数），方法同上
若过少则判定为中心物体被不小心抹黑，恢复原来的颜色
i) 求出图像四个角边缘上未抹黑的像素的个数和他们的亮度值（三通道）总和
若存在这样的像素点则计算他们的亮度值平均值
将图像中亮度值接近该平均值的像素点抹黑
j) 重复一边 g→h→i 这三个操作

在实验过程中遇到的问题及解决方法

首先是玄学问题

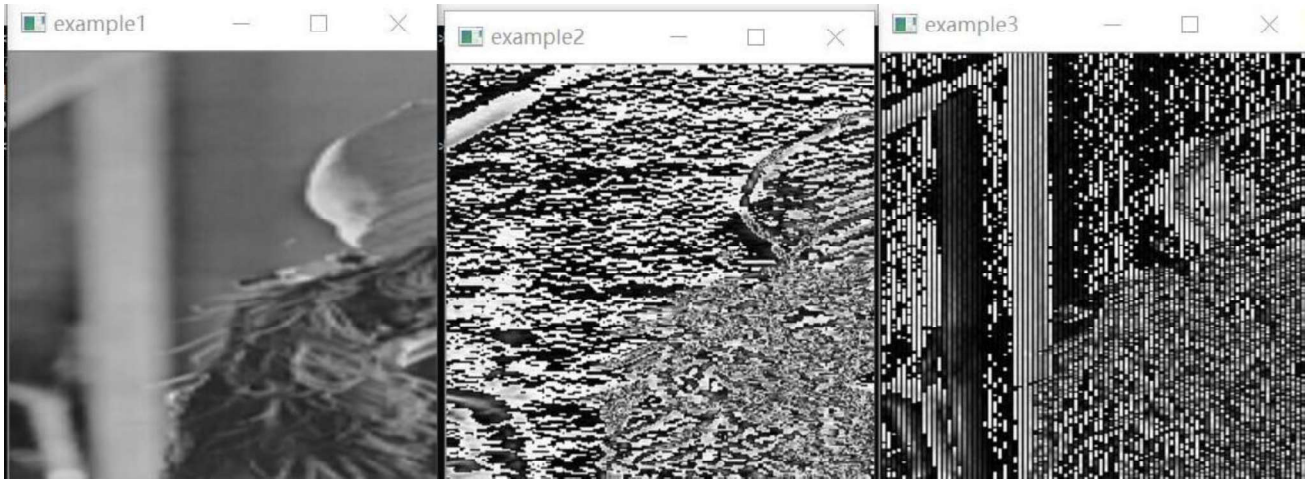
重启以后程序不能正常运行，显示缺少 dll。打开环境变量一看，配置好的 bin 消失了。若非误操作，初步怀疑是“win10 系统关机前会记录当前打开的内容”相关功能引发的问题。尤其是在两台 win10 设备上登录了同一个账户更容易复现。

关于功能 1-6

0. 为了能跳动式移动光标和模拟选择框，专门写了"tools.h"，包括 void cursor(int y, int x);int getCursor() 等函数来获取和设置光标位置；void gen();void opt()来设置字体颜色和背景色。自带的库函数中{x,y}代表坐标轴。所以事实上，y 代表了行，x 代表了列。一开始搞错了，卡了五分钟。
1. 为了同时支持数据的批量复制黏贴导入，我没有把回车键作为矩阵写入结束的标志。但是当光标置于矩阵右下角时我进行了特判，接受回车键作为矩阵写入结束的标志。来优化操作体验。
2. 为了支持负号花了一番功夫。最后索性做齐了完整的读入判断，程序不会因为输入了非法数据而崩溃了。
3. 为了输出矩阵的左右大括号翻了好久的特殊字符
4. 少数特定情况下需要两次回车才能结束矩阵的读入，比较难修。另外在对方向键等特殊键进行 _getch()时，会先传入一个 224，再传入对应键的 ascii 值。这个特性导致了一个相当难找的 bug，又卡了五分钟。

5. -1 用 freopen 这会导致 system 的 stdout 也被重定向，从而在 freopen 到文件流再 freopen 回控制台之后 getscreenbufferinfo 失效，从而导致 system("cls")失效。而且由于 crt 闭源，很难让他恢复。事实上，除算法竞赛外，fopen 更常用。

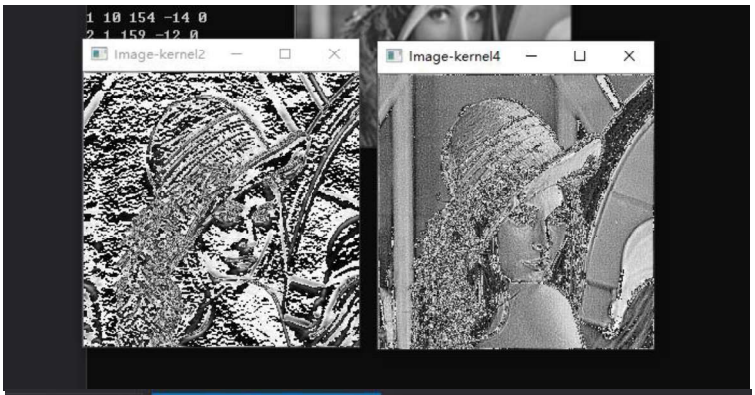
功能 7



一开始显示半边灰暗半边正常，但是基本上都保持原图模样（卷积无效了）；后来发现原因出在即使这是张灰白图，保存格式依然是 RGB 三通道的，所以二维数组向 Mat 赋值时使用了 Vec3b 赋值。然后发现图像被横向拉长了，显示不完全。最后在 Mat 向二维数组赋值时从 uchar 也改成了 Vec3b，发现问题解决。（Vector 真香）（逃）

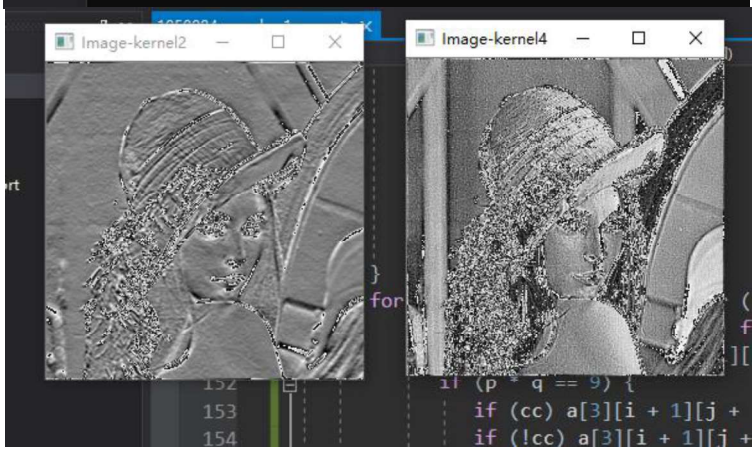
得到图像→→→

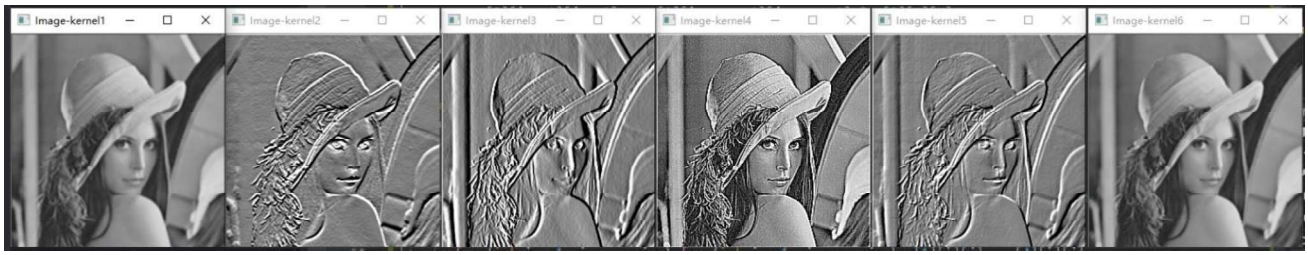
由此初步判断，Kernel2 对应的是浮雕滤波器，Kernel4 对应的是锐化。由于 kernel2 存在大量黑白，我对所有卷积核总和为 0 的生成图像加上了 128 的偏移



得到结果→→→

此时发现 两张图都在边缘部分产生了黑白噪点，推测是经过卷积以后生成了越界值，经过 unsign 强转以后由很黑突变为很白（或由很白突变为很黑）。最后在赋值给 Mat 前将>255 的值抹平为 255，<0 的值抹平为 0。最终成功全部得到了期望卷积结果：





功能 8

0. 首先是 DFS 他爆栈了，做这份作业的时候也没讲指针。想了一会到底怎么避免 newnewenew

   一直 new 一直爽，忘记 delete 火葬场——wws

   最后直接换成了 BFS，世界和平。

1. 一个一个开开来的框乱七八糟的排列，也不好拖动。在 opencv 里还真找到了个叫 moveWindow 的函数，我好了。
2. 发现使用 waitkey(0)的话 程序必须要所有图片都关闭后才会响应。
   解决方法：if (waitKey(20000)) destroyAllWindows();
   //等待 20 秒，或者在图像界面按任意键后关闭所有图像
3. 剩下的都在设计思路里面了。事实上这些步骤中许多都是在测试较多图片后，发现不满意之处，再想出来的解决办法。

整理了（主要是功能 7）中发现和解决的问题

（包括高程荣誉群里还出现过的问题，在此尝试给出解答：）

//是比较早整理的了，新的问题没全部更新进来 qwq

0. VS 报错要求把数组移进堆：没有使用全局变量也没有使用动态申明
1. VS 报错从数组中读取的数据无效：可读大小 balabala，但可能读取了 balabala：初始化 /memset 部分没写/写错·
2. VS 可以编译，但运行时出错，提示缺少 dll：检查环境变量
3. 输出的图像一部分很暗一部分和原图没区别，中间夹杂着彩色丝线或者彩点：数组赋值给 Mat 时用了单通道 uchar，而读入的图片格式为三通道的。
4. 输出图像被横向拉长，显示不完全：虽然最后数组赋值给 Mat 时正确赋值了三通道，但最开始 Mat 向二维数组赋值时用了 uchar
5. 图像整体过暗甚至大部分全黑：除以卷积核总和过程中发生问题（如变量冲突，在不同地方声明了两次相同的变量然后没有发现（例如将矩阵列数和卷积核总和都用变量 c 表示了））

6. 图像金属化：直接把图像的矩阵值赋值给了(unsigned)char 类型的数组，导致运算过程中不断越界。这会导致生成图像高频率的从黑到灰到白再突变会黑，我称之为"金属化"形状。（见右图）
7. 矩阵 2,3,5 相比矩阵 1,4,6 黑白对比更强烈：卷积核总和为 0（大致上代表着平均灰度为 127，无视原图）的生成图像没有做处理（抹平<0 及>255 的值）
8. 矩阵 2,3,5 相比矩阵 1,4,6 更黑：做了抹平处理，但是没有加上 128 的偏移
9. 图像大部分较为正常，而对应原图的边缘部分产生了黑白噪点：卷积以后生成了越界值，经过 unsign 强转以后由很黑突变为很白（或由很白突变为很黑）。抹平即可。

心得体会

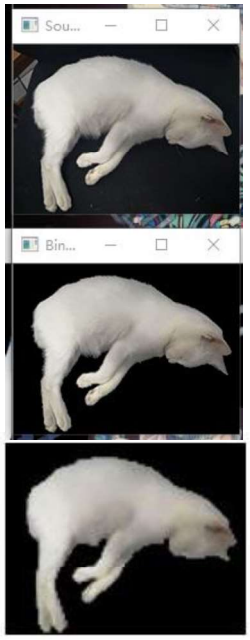
编程过程的体会/对图像（矩阵）卷积操作的理解/吐槽

二值化的改进之处

全局二值化对（如小船）低对比度高底噪的图表现不好

待改进的地方：通过计算图像灰度图来实现找到更合理的阈值。

只写了黑底的处理，没写白底的处理（虽然只是符号方向上的区别...）



Result

猫咪

和 OJ 上展示的图的对比，本程序对（猫咪等）照片成功保留了完整边缘和边缘细节。（从上到下：原图、本程序结果、OJ 结果）

语法

熟悉了 extern 等平时不常用的语法。

编程规范

一定要用模块化编程。习惯于写函数。感觉很 ok 了，不会再做大改动了，就写成*.h 头文件封装起来。不然上千行的代码，翻起来都累。

其次是写注释。被助教的 Graphic.cpp 这种没什么人会看（懂）但是依然写了详尽注释的代码感动了，补上了自己的注释，希望能帮助理解 w

然后是函数命名，火葬场火葬场，我以后一定不贪图简单好好命名。

最后是写代码，因为谢谢改改，还有很多格式上改进的地方，但是现在的代码也不难理解，时间复杂度也得到了保证(O(r*c))，暂时就这样了~

调试

一定要大量使用输出调试（或者 watch 变量），不要对着一张不大好看的图苦思冥想半天，盲猜错在哪里。

可以输出 Mat image 来看它的 data 保存方式和发现通道数问题；可以输出全图左上角 15*15 的矩阵来判断自己的卷积函数是否存在问题；以及发现总和为 0/不为 0 的卷积核的不同之处，等等。

实用性

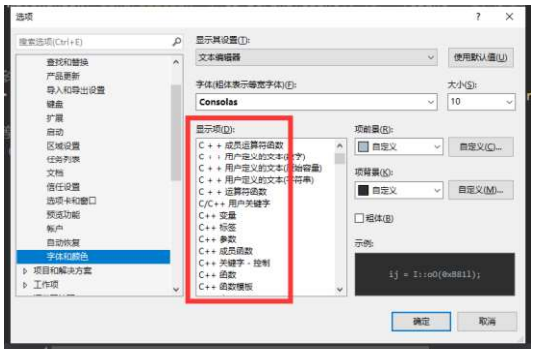
我的 APP 开发选修课也要做一个大作业。我打算写一个功能齐全的（包括矩阵运算）的计算器。这次的就当一个练手啦！

前端

信息竞赛极少对前段页面做出要求。虽然大一一直在学网页、小程序和手机 APP 的前端；整个作业，1 个小时花在功能 0-7，1 个小时花在拓展功能 8 OTSU 算法，1 个小时花在写这份报告，15 个小时花在了写界面。

王维斯大佬说，最恶心的是前端。我宣布，大佬说得对。

VS2019 真的香



平时用习惯了 sublime 。为了上手 Visual Studio 花了好多时间调色；甚至复习了不同项目的名称。

因为一开始没有用 VS 写，新建头文件没有自动加上 #pragma once，这才意识到这一行代码的宝贵之处。

另外，ctrl+k+f 解千愁。

论学线代的必要性（标题回收√）

这次大作业让我重新熟悉了矩阵卷积，为线性代数做出了巨大贡献（）

你还有什么废话想说的啊(ﾍ'□')ﾍ ┻┻

想必助教们批这么多份大作业一定也很累，辛苦了（绝对不是为了想要多 0.1 分）

最后，请加 大 力 度（指 lena 和大作业）

源代码

195****_ZONGHE_1.CPP

```
#define _CRT_SECURE_NO_WARNINGS
#include "tools.h"
#include "func1-6.h"
#include "func7.h"
#include "func8.h"
void m1(); void m2(); void m3(); void m4(); void m5(); void m6(); void m7(); void m8();//八个功能
int n, r, c, d, k, nr, nc, a[8][800][800], f[800][800], cl[800], t;
int main() {

    //设置标题；设置控制台窗口长宽防止行数/列数过长（255）时渲染出错
    system("title 1950084_zonghe_1");
    char cmd[30];
    sprintf(cmd, "mode con cols=%d lines=%d", 1300, 1000);
    system(cmd);

    //主菜单界面
    do {
        int f;
        do {
            f = 0;
            system("cls");
            for (int i = 1; i <= 57; ++i) cout << "*";
            cout << "\n *      1 矩阵加法       2 矩阵数乘       3 矩阵转置       *";
            cout << "\n *      4 矩阵乘法       5 Hadamard 乘积  6 矩阵卷积       *";
            cout << "\n *      7 卷积应用       8 OTSU 算法      0 退出系统       *\n";
            for (int i = 1; i <= 57; ++i) cout << "*";
            cout << "\n 选择菜单项<0~8>:";
            n = _getche() - 48;
            if (n < 0 || n>8) {
                f = 1;
                cout << "\n 输入错误\n 按任意键继续...";
                k = _getch();
            }
            if (!n) {
                ifexit();//键盘左右方向键控制 yes 或 no，因此放进 tools.h 了
            }
        } while (f);
        system("cls");
        memset(a, 0, sizeof(a));
        switch (n) {
        case 1: m1(); break;
        case 2: m2(); break;
        case 3: m3(); break;
        case 4: m4(); break;
        case 5: m5(); break;
        case 6: m6(); break;
        case 7: m7(); break;
        case 8: m8(); break;
        }
    } while (1 == 1);
    return 0;
}
```