Assignment 3

姓名：田原

学号：3180101981

开发软件说明

- 主要开发语言
  - C++
- 主要开发环境
  - Visual Studio 2019
  - OpenCV 4.5.0
- 编译系统环境
  - Windows 10 (64-bit)

算法具体步骤及实现要点

执行逻辑

这次的作业需要实现的功能概括起来就是两点：

- 读入摄像头画面并实时显示
- 对暂停画面做harris角点检测

为了方便用户使用，我希望达到这样的效果：

- 用户打开程序之后，是摄像头的实时画面，如果按下空格，就进入Harris角点检测的过程，如果按下esc键就退出程序；
- 进入Harris检测过程之后，先显示暂停画面，按一下空格开始计算，显示最大特征值图，之后每按一次空格依次展示最小特征值图、彩色R图、角点检测结果图。再按一次空格回到摄像头实时画面。

因此，我设计了一个`write()`函数用于摄像头读入画面后的用户交互过程，在`main()`中while循环执行，直到用户选择退出程序，或者摄像头获取画面失败。在`write()`函数内部，当进入角点检测过程时，会调用`harris_corner`函数处理。

以下是我的`main()`函数代码：

```
int main() {
    namedWindow("RESULT", 1); // 创建RESULT窗口
    VideoCapture capture(0); // 摄像头捕获画面
    Mat frame; // 捕获画面到frame
    while (capture.read(frame)) { // 捕获画面成功时
        int i = write(frame); // 进入write交互函数
        if (i == 1) break; // 如果用户选择退出程序，退出
    }
    destroyAllWindows(); // 关闭所有窗口
    return 0;
}
```

write(Mat im)交互函数

我们捕获摄像头画面到frame之后，先把frame显示到RESULT窗口上，之后调用`waitKey(40)`捕捉键盘动作。如果没有键盘输入，就会再次捕获新的摄像头画面到frame，所以我们的视频显示中单个画面的显示时间为40。

- 如果捕获到用户按下空格：调用`harris_corner`函数，进入处理图片过程。结束后等待键盘再次按下空格或者esc键。
- 如果捕获到用户按下esc：关闭所有窗口退出。

具体代码如下所示。

```
int number = 0; // number是程序开始后处理了几张图片，用于harris图片存储时的文件名
int write(Mat im) { // 显示im，时长为40，如果检测到空格则暂停，esc则退出
    imshow("RESULT", im);
    int key = waitKey(40);
    if (key == 32) { // 一帧显示时间为40，期间如果检测到空格，进入harris过程
        // 开始处理图片并显示
        harris_corner(im, ++number);
        // 处理完毕
        int key1 = waitKey(0); // 等待键盘输入
        while (key1 != 32) { // 一直等待键盘输入，直到有空格
            if (key1 == 27) { // 如果暂停期间按到esc键
                goto here1; // 退出
            }
            key1 = waitKey(0);
        }
    }
    else if (key == 27) { // 如果按下esc键
    here1:;
        destroyAllWindows();
        return 1;
    }
    return 0;
}
```

角点检测

原理

根据 $E(u,v)$ 化简结果，我们最后得到了一个 M 矩阵：$M = \sum_{(x,y)\in W} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$。

其中，I_x, I_y 分别是窗口内像素点在 x 方向和 y 方向的梯度值，M是一个协方差矩阵。

将 $E(u,v)$ 近似为二项函数 $E(u,v) = Au^2 + 2Cuv + Bv^2$，本质上就是一个椭圆函数。其中，椭圆的长和宽是由$M$的特征值$\lambda_1, \lambda_2$ 决定的（数学推算可以得到，椭圆的长短轴是矩阵特征值平方根的倒数），椭圆的方向是由M的特征向量决定的。我们有以下模糊的结论：

- 如果两个矩阵特征值都比较大，则像素点的梯度分布比较分散，梯度变化程度比较大，符合角点在窗口区域的特点；
- 如果两个特征值都比较小，窗口区域内的像素点的梯度幅值比较小，应该处于平坦区域；
- 如果是边缘区域，边缘上的像素点的某个方向的梯度幅值变化比较明显，另一个方向上的梯度幅值变化较弱，这样两个特征值理论上应该是一个比较大，一个比较小，具体和窗口的分布情况有关。

所以我们有以下度量角点的办法：

设 $R = det(M) - k(trace(M))^2$，此处$k$一般在0.04到0.06之间。则对于角点 R 很大，平坦的区域 R 很小，边缘的 R 为负值。最后设定R的阈值，进行角点判断。

实现

函数参数如下，其中`im`是输入的彩色图像，`num`是第几张处理图片用于后续文件命名，`ksize`是窗口的大小，默认为3。

```
void harris_corner(Mat im, int num, int ksize = 3)
```

首先显示原始图像，并写入文件：

```
stringstream ss;
ss << num;
string ind = ss.str(); // 获得num转字符串
cout << "Show the original image.." << endl;
imshow("RESULT", im); // 展示原始图像
imwrite(ind + "_original.jpg", im); // 写入文件
waitKey(0);
```

转成灰度图，初始化部分变量：

```
Mat gray; // 灰度图
cvtColor(im, gray, COLOR_BGR2GRAY);
double k = 0.04; // k值
double threshold = 0.01; // 阈值参数设置
int width = gray.cols;
int height = gray.rows;
```

计算梯度值等有关数值，做高斯滤波：

```
// 计算Ix, Iy
Mat grad_x, grad_y;
Sobel(gray, grad_x, CV_16S, 1, 0, ksize);
Sobel(gray, grad_y, CV_16S, 0, 1, ksize);
// 计算Ix^2, Iy^2, Ix*Iy
Mat Ix2, Iy2, Ixy;
Ix2 = grad_x.mul(grad_x);
Iy2 = grad_y.mul(grad_y);
Ixy = grad_x.mul(grad_y);
// 高斯滤波
GaussianBlur(Ix2, Ix2, Size(ksize, ksize), 2);
GaussianBlur(Iy2, Iy2, Size(ksize, ksize), 2);
GaussianBlur(Ixy, Ixy, Size(ksize, ksize), 2);
```

计算特征值和R值：

```
// 计算lambda_max, lambda_min, R
double lambda_max_max = 0, lambda_min_max = 0, R_max = 0;
Mat lambda_max(gray.size(), gray.type());
Mat lambda_min(gray.size(), gray.type());
Mat R_im(gray.size(), gray.type());
vector<vector<double>> R(height, vector<double>(width, 0));
vector<vector<double>> max_array(height, vector<double>(width, 0));
vector<vector<double>> min_array(height, vector<double>(width, 0));

cout << "Calculate the eigen values..." << endl;
for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
        double Marray[2][2] = { Ix2.at<short>(i, j), Ixy.at<short>(i, j),
                                Ixy.at<short>(i, j), Iy2.at<short>(i, j) }; // 得到M矩阵
        Mat M(2, 2, CV_64FC1, Marray);
        Mat lambda, temp;
        eigen(M, lambda, temp); // 计算特征值到lambda矩阵
        double max = lambda.at<double>(1, 0), min = lambda.at<double>(0, 0); //获得max特征值和min特征值
        if (lambda.at<double>(0, 0) > lambda.at<double>(1, 0)) {
            max = lambda.at<double>(0, 0);
            min = lambda.at<double>(1, 0);
        }
        max_array[i][j] = max; // max特征值
        min_array[i][j] = min; // min特征值
        if (max > lambda_max_max) lambda_max_max = max;
        if (min > lambda_min_max) lambda_min_max = min;
        R[i][j] = (max * min - k * (max + min) * (max + min)); // 计算R值
        if (R[i][j] > R_max) R_max = R[i][j];
    }
}
```

得到特征值图像、R图像和结果图像：

```
cout << "Write the eigen values and R values into image.." << endl;
for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
        R_im.at<uchar>(i, j) = (uchar)(R[i][j] * 1.0 / R_max * 1.0 * 255);
```

```
        lambda_max.at<uchar>(i, j) = (uchar)(max_array[i][j] * 1.0/
lambda_max_max * 1.0 * 255);
        lambda_min.at<uchar>(i, j) = (uchar)(min_array[i][j] * 1.0/
lambda_min_max * 1.0 * 255);
        if (R[i][j] > R_max * threshold) { // corner
            im.at<Vec3b>(i, j)[0] = 0;
            im.at<Vec3b>(i, j)[1] = 0;
            im.at<Vec3b>(i, j)[2] = 255; // red
        }
    }
}
Mat colorR; // 彩色R图
applyColorMap(R_im, colorR, COLORMAP_JET);
```

这里使用了`applyColorMap`函数，它能够将一张单通道的图转化成为彩色图，选择适当的参数就能得到彩色R图的效果，其中，灰度值较低的部分对应蓝色，灰度值较高的部分对应红色。

最后依次展示图像并写入文件：

```
cout << "Now you can press space to show the next result .." << endl;
cout << "lambda_max image: " << endl;
imshow("RESULT", lambda_max);                          // lambda max
imwrite(ind + "_lambda_max.jpg", lambda_max);
waitKey(0);
cout << "lambda_min image: " << endl;
imshow("RESULT", lambda_min);                          // lambda min
imwrite(ind + "_lambda_min.jpg", lambda_min);
waitKey(0);
imshow("RESULT", colorR);                              // R
imwrite(ind + "_colorR.jpg", colorR);
waitKey(0);
cout << "Harris corner result: " << endl;
imshow("RESULT", im);                                  // harris corner
imwrite(ind + "_Corner.jpg", im);
```

结果展示及分析

我一共选取了四张图片的测试结果。

1

original:

特征值最大图和特征值最小图：

彩色R图：

角点检测结果图：

2

original:

特征值最大图和特征值最小图：

彩色R图：

角点检测结果图：

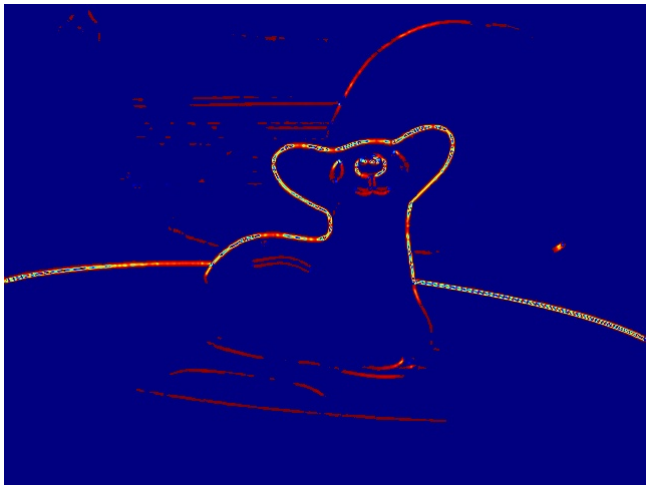
3

这张图选择了轮廓比较平滑的检测对象。

original:

特征值最大图和特征值最小图：

彩色R图：



角点检测结果图：

4

original:

特征值最大图和特征值最小图：

彩色R图：

角点检测结果图：