目录

1 实验目标

利用课上学习的Harris角点检测和SIFT特征检测对图像进行处理，具体要求包括:

(1)输入一张图像

(2)检测图像的Harris角点特征，并用绿色”+”标记

(3)检测图像的SIFT特征，并用红色”o”标记

提交内容:

a.源码

两种方法均使用opencv函数（60分);

一种使用opencv函数，一种自行编码（100分）

b.报告

包含实验结果分析和运行结果截图。原图加2、3效果图。Pdf文件（10分）

2 程序

2.1 程序简介

使用OpenCV库实现Harris角点检测和SIFT特征算法，并在图像中用不同的符号标记检测到的特征。

程序首先加载图像“tju.jpg”，显示图像，然后等待按键。接下来，程序调用自定义的myCornerHarris函数，对图像进行Harris角点检测。检测结果存储在dst中，并被归一化以便显示。然后，程序遍历角点响应图像中的每个像素，并标记角点。标记使用绿色十字形表示。最后，程序显示带有角点标记的图像，并将其保存到文件“tju_harris.jpg”中。

然后，程序加载同一图像“tju.jpg”并将其转换为灰度图像。程序使用SIFT算法检测灰度图像中的关键点，并使用红色圆圈标记检测到的关键点。标记后的图像被命名为“tju_sift.jpg”，并保存。

2.2 算法

2.2.1 Harris角点检测

Harris角点检测算法是一种经典的计算机视觉算法，用于检测图像中的角点。它的基本思想是比较小的局部窗口在不同方向上移动时，图像强度的变化程度。如果在多个方向上都有较大的变化，那么这个点就很可能是角点。具体步骤如下：

1. 计算每个像素的梯度：对于灰度图像，可以使用Sobel算子或其他算子计算水平和垂直方向上的梯度。
2. 计算每个像素的自相关矩阵：对于每个像素，以其为中心的局部窗口内的像素值和梯度信息，计算自相关矩阵。
3. 计算每个像素的角点响应函数：对于每个像素，根据其自相关矩阵计算角点响应函数。最常用的是Harris响应函数：

$$R = det(M) - k \times trace^2(M)$$

其中，M是自相关矩阵，$det(M)$和$trace(M)$分别是它的行列式和迹，k是一个常数，通常取值为0.04到0.06。

对角点响应函数进行非极大值抑制：找到响应函数的局部最大值，即认为该位置是角点。

Harris角点检测算法的核心在于角点响应函数的计算。它基于二阶微分近似，可以表示为：

$$M = \sum_{i,j} w_{i,j} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}_{i,j}$$

其中，I_x和I_y分别是图像在x和y方向上的梯度，$w_{i,j}$是权重系数，通常使用高斯函数或矩形函数。

通过对每个像素计算角点响应函数，可以得到图像中所有可能的角点。然后可以使用非极大值抑制等方法对角点进行筛选和优化，得到最终的角点检测结果。'

2.2.2 SIFT特征算法

SIFT（Scale-Invariant Feature Transform）特征算法是一种计算机视觉算法，用于提取图像中的关键特征点。它能够有效地处理图像的尺度、旋转和亮度变化，具有很高的鲁棒性和可靠性。SIFT特征算法主要分为四个步骤：尺度空间极值检测、关键点定位、方向分配和描述符生成。

1. 尺度空间极值检测：首先，在不同的尺度上使用高斯滤波器对原始图像进行模糊处理，得到图像的高斯金字塔。然后，在每个尺度上，通过比较相邻像素的灰度值，找到图像中的极值点，即关键点的候选。
2. 关键点定位：对于每个关键点候选，通过在尺度空间和空间中进行极值点的精确定位，得到关键点的位置和尺度大小。为了提高关键点的稳定性，还需要对关键点进行边缘响应、主曲率等方面的筛选。
3. 方向分配：对于每个关键点，需要确定其主方向，以便后续生成描述符。通常使用关键点周围的梯度方向直方图来计算主方向，并将关键点方向转换为主方向。
4. 描述符生成：根据关键点周围的图像信息，生成一个128维的特征向量，描述关键点的局部特征。描述符的生成过程可以通过将关键点周围的像素分为若干个子区域，并计算每个子区域内的梯度直方图来实现。

SIFT特征算法的核心在于描述符的生成，它基于局部像素梯度的统计信息，可以表示为：

$$d_i = [f(x, y, \theta_1), f(x, y, \theta_2), \ldots, f(x, y, \theta_n)]$$

其中，d_i是第i个关键点的描述符向量，$f(x, y, \theta)$表示在位置(x, y)处，以θ为主方向的梯度方向直方图。

SIFT特征算法通过对图像中的关键点进行提取和描述符生成，可以实现对图像的特征匹配、目标检测、图像分类等任务。

2.3 实现细节

2.3.1 输入一张图像

使用 `imread()` 函数读取 "tju.jpg" 图片，并创建一个窗口来显示它。

```
    //读取tju.jpg
    Mat src = imread("tju.jpg");
    //创建窗口
    namedWindow("tju", WINDOW_AUTOSIZE);
    //显示图像
    imshow("tju", src);
    //等待按键
    waitKey(0);
    return 0;
```

2.3.2 自行编码实现Harris角点检测

程序首先进行了图像的预处理，包括将彩色图像转换为灰度图像。

```
Mat gray1;
cvtColor(src, gray1, COLOR_BGR2GRAY);
```

然后利用Sobel算子计算了图像每个像素的x和y方向的梯度，接着计算了每个像素的梯度乘积，然后对每个像素的局部邻域内的梯度乘积进行了加权平均得到了每个像素的哈里斯响应函数。

```
void myCornerHarris(const Mat& src, Mat& dst, int blockSize, int apertureSize,
double k)
{
    // 计算梯度
    Mat dx, dy;
    Sobel(src, dx, CV_32FC1, 1, 0, apertureSize);
    Sobel(src, dy, CV_32FC1, 0, 1, apertureSize);

    // 计算每个像素的梯度乘积
    Mat A = dx.mul(dx);
    Mat B = dy.mul(dy);
    Mat C = dx.mul(dy);

    // 计算局部邻域上梯度乘积的和
    Mat S_A, S_B, S_C;
    boxFilter(A, S_A, CV_32FC1, Size(blockSize, blockSize));
    boxFilter(B, S_B, CV_32FC1, Size(blockSize, blockSize));
    boxFilter(C, S_C, CV_32FC1, Size(blockSize, blockSize));

    // 计算每个像素的哈里斯响应函数
    dst.create(src.size(), CV_32FC1);
    for (int i = 0; i < src.rows; i++) {
        for (int j = 0; j < src.cols; j++) {
            float a = S_A.at<float>(i, j);
            float b = S_B.at<float>(i, j);
            float c = S_C.at<float>(i, j);
            float det = a * b - c * c;
            float trace = a + b;
            float response = det - k * trace * trace;
            dst.at<float>(i, j) = response;
        }
    }
}
```

最后，对角点响应值进行归一化，并标记出响应值大于给定阈值的像素位置。

```
    int blockSize = 2;
    int apertureSize = 3;
    double k = 0.04;
    int threshold = 140;
```

```
    Mat dst;
    myCornerHarris(gray1, dst, blockSize, apertureSize, k);
    // 归一化角点响应值
    normalize(dst, dst, 0, 255, NORM_MINMAX, CV_32FC1, Mat());

    // 标记角点
    for (int i = 0; i < dst.rows; i++)
    {
        for (int j = 0; j < dst.cols; j++)
        {
            if (dst.at<float>(i, j) > threshold)
            {
                //circle(src, Point(j, i), 3, Scalar(0, 255, 0), -1);
                //用十字标注
                line(src, Point(j - 3, i), Point(j + 3, i), Scalar(0, 255, 0), 1);
                line(src, Point(j, i - 3), Point(j, i + 3), Scalar(0, 255, 0), 1);
            }
        }
    }

    // 显示带有角点的图像
    imshow("Harris corner detection", src);
    // 等待按键
    waitKey(0);
    // 保存图像
    imwrite("tju_harris.jpg", src);
```

2.3.3 使用opencv函数实现SIFT特征检测

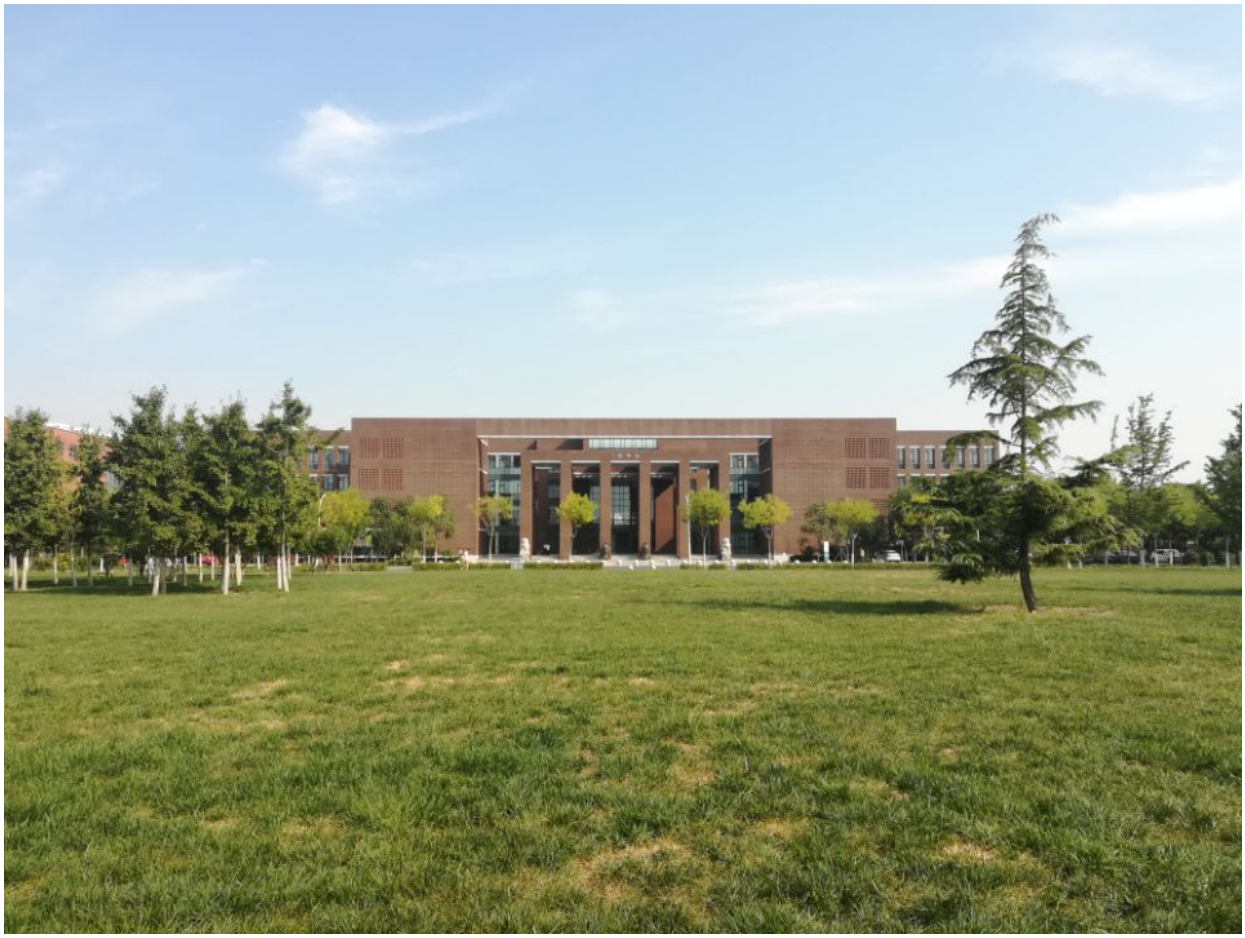
函数`cvtColor`将输入图像从BGR格式转换为灰度格式，`SIFT::create`创建了一个SIFT特征检测器，其中传入的参数分别是：尺度空间的层数、每层尺度空间中的高斯核的大小、边缘响应阈值、高斯金字塔的Octaves数目和每个Octave中的尺度数目。`sift->detect`函数对输入图像进行SIFT特征检测，返回特征点的位置和大小信息，并存储在`vector<KeyPoint> keypoints`中。`drawKeypoints`函数将特征点用红色圆圈标记在原始图像上，参数`img_keypoints`即为标记后的图像，`Scalar(0, 0, 255)`表示圆圈的颜色为红色。最后，通过`imshow`函数显示标记后的图像，并通过`imwrite`函数将其保存。

```
//检测图像的SIFT特征，并用红色”o”标记
Mat src2 = imread("tju.jpg");
Mat gray;
cvtColor(src2, gray, COLOR_BGR2GRAY);
Ptr<SIFT> sift = SIFT::create(0, 3, 0.09, 10, 1.6);//设置阈值
vector<KeyPoint> keypoints;
sift->detect(gray, keypoints);
Mat img_keypoints;
drawKeypoints(src2, keypoints, img_keypoints, Scalar(0, 0, 255),
DrawMatchesFlags::DEFAULT);
imshow("tju", img_keypoints);
waitKey(0);
//保存图像
imwrite("tju_sift.jpg", img_keypoints);
```

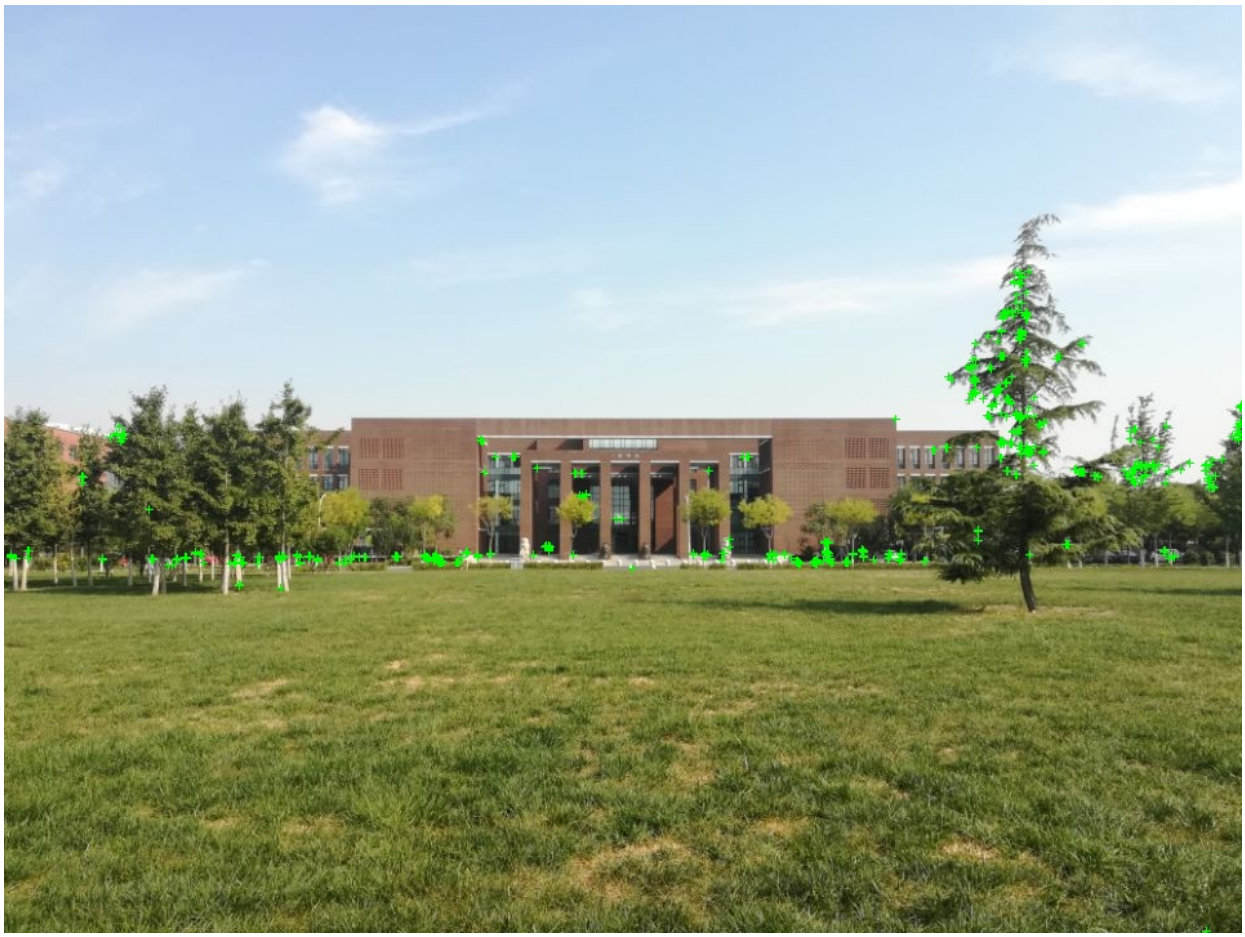
3 结果与讨论

3.1 实验结果

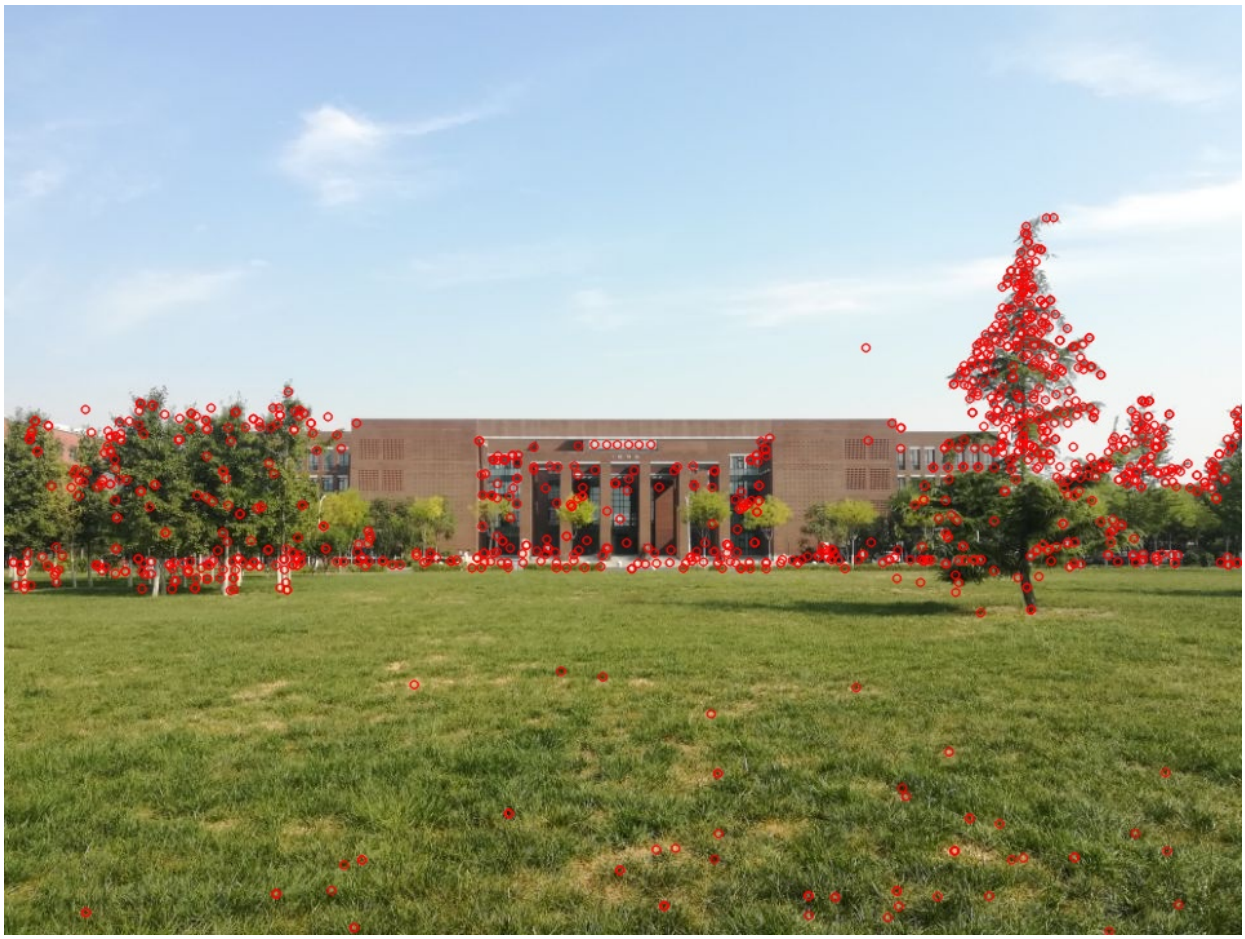
3.1.1 输入一张图像



3.1.2 Harris角点检测



3.1.3 SIFT特征检测



3.2 实验分析

Harris角点检测方法可以检测出图像中的角点，这些角点在图像中通常具有一定的边缘和纹理信息。而SIFT特征检测方法可以检测出图像中的SIFT特征点，这些特征点通常具有一定的尺度不变性和旋转不变性，可以用于图像匹配和目标跟踪等应用场景。

3.3 讨论

通过这个实验我掌握了Harris角点检测和SIFT特征检测这两种图像处理方法，也让我对图像处理方法和OpenCV函数有了更深入的了解。我相信这些知识和经验对我的未来学习和工作会产生一定的帮助。