课程设计 Part1 全景图拼接（Opencv C++）

——刘崇轩 2312801 智能科学与技术

一、实验目的

1. 掌握基于特征点的图像拼接方法。
2. 理解 SIFT 特征检测、描述与匹配原理，并应用于图像配准。
3. 学习简单图像融合与黑边裁剪技巧，提高拼接结果的视觉效果。

二、实验原理

1.SIFT 特征检测与描述

利用尺度不变特征变换（SIFT）算法检测图像关键点，并提取旋转、尺度不变的 128 维特征描述子。

2.特征匹配

使用 FLANN（快速近似最近邻）算法结合 Lowe 比例测试剔除错误匹配，得到两幅图像的对应点集。

3.单应矩阵估计

基于 RANSAC 方法从匹配点中剔除离群点，计算 3×3 单应矩阵 H，以实现平面视角变换。

4.图像扭曲与融合

利用 warpPerspective 将待拼图像变换到参考图像坐标系，并在重叠区域通过像素最大值融合消除黑线。

5.自动黑边裁剪

提取非零像素区域的最小外接矩形，去除拼接后两侧及底部的黑色填充。

三、实验步骤

1.图像加载

程序在内部依次调用 imread("S1.jpg") … imread("S7.jpg"),将七张图像读取到内存。

2.特征检测与配准

对每一对图像，调用 detectAndMatchSIFT 提取 SIFT 特征并使用 BFMatcher 进行匹配，得到匹配点集。

统计每个灰度级（0~255）的像素数量。

3.单应变换计算

使用 findHomography 与 RANSAC 算法，根据匹配点估计单应矩阵 H。

4.图像扭曲与融合

通过 warpPerspective 将第二张图扭曲到参考图坐标系，并在重叠区域采用像素最大值融合策略消除黑线。

5.黑边裁剪

调用 boundingBoxNonZero 找出非零像素区域的边界，并裁剪得到紧凑的全景图。

6.结果显示与保存

使用 imshow 在固定窗口中展示拼接结果，并调用 imwrite("panorama_cropped.jpg")将其保存到磁盘。

四、程序代码

```
#include <opencv2/opencv.hpp>
#include <opencv2/features2d.hpp>
#include <iostream>
#include <vector>
using namespace cv;
using namespace std;

//SIFT 特征检测与匹配
void detectAndMatchSIFT(const Mat& img1, const Mat& img2, vector<Point2f>& pts1,
vector<Point2f>& pts2) {
    Ptr<SIFT> sift = SIFT::create();
    vector<KeyPoint> kp1, kp2; Mat desc1, desc2;
    sift->detectAndCompute(img1, noArray(), kp1, desc1);
    sift->detectAndCompute(img2, noArray(), kp2, desc2);
    FlannBasedMatcher matcher;
    vector<vector<DMatch>> knnMatches;
    matcher.knnMatch(desc2, desc1, knnMatches, 2);
    const float ratioThresh = 0.75f;
    for (auto& m : knnMatches) {
        if (m[0].distance < ratioThresh* m[1].distance) {
            pts2.push_back(kp2[m[0].queryIdx].pt);
            pts1.push_back(kp1[m[0].trainIdx].pt);
        }
    }
}

//自动裁剪黑边
Rect boundingBoxNonZero(const Mat& img) {
    Mat gray;
    cvtColor(img, gray, COLOR_BGR2GRAY);
    Mat mask = gray > 0;
    vector<Point> pts;
    findNonZero(mask, pts);
    return boundingRect(pts);
}

//简单融合拼接（消除黑线）
Mat stitchPair(const Mat& img1, const Mat& img2) {
    Mat gray1, gray2;
    cvtColor(img1, gray1, COLOR_BGR2GRAY);
    cvtColor(img2, gray2, COLOR_BGR2GRAY);
    vector<Point2f> pts1, pts2;
```

```
    detectAndMatchSIFT(gray1, gray2, pts1, pts2);
    Mat H = findHomography(pts2, pts1, RANSAC);

    //创建画布并扭曲图像 2
    Mat warped;
    int width = img1.cols + img2.cols;
    warpPerspective(img2, warped, H, Size(width, img1.rows));

    //简单 max 融合（解决黑线问题）
    Mat result = warped.clone();
    for (int y = 0; y < img1.rows; ++y) {
        for (int x = 0; x < img1.cols; ++x) {
            Vec3b p1 = img1.at<Vec3b>(y, x);
            Vec3b p2 = warped.at<Vec3b>(y, x);
            if (p2 == Vec3b(0, 0, 0)) result.at<Vec3b>(y, x) = p1;
            else for (int c = 0; c < 3; ++c)
                result.at<Vec3b>(y, x)[c] = max(p1[c], p2[c]);
        }
    }

    //裁剪黑边
    Rect crop = boundingBoxNonZero(result);
    return result(crop);
}

int main() {
    //读取 S1.jpg 到 S7.jpg
    vector<string> paths;
    for (int i = 1; i <= 7; i++) paths.push_back("S" + to_string(i) + ".jpg");

    //读取 D1.jpg 到 D8.jpg
    //vector<string> paths;
    //for (int i = 1; i <= 8; i++) paths.push_back("D" + to_string(i) + ".jpg");

    //读取 P1.jpg 到 P5.jpg
    //vector<string> paths;
    //for (int i = 1; i <= 5; i++) paths.push_back("P" + to_string(i) + ".jpg");

    vector<Mat> imgs;
    for (auto& p : paths) {
        Mat im = imread(p);
        if (im.empty()) { cerr << "无法加载:" << p << endl; return -1; }
        imgs.push_back(im);
    }
```

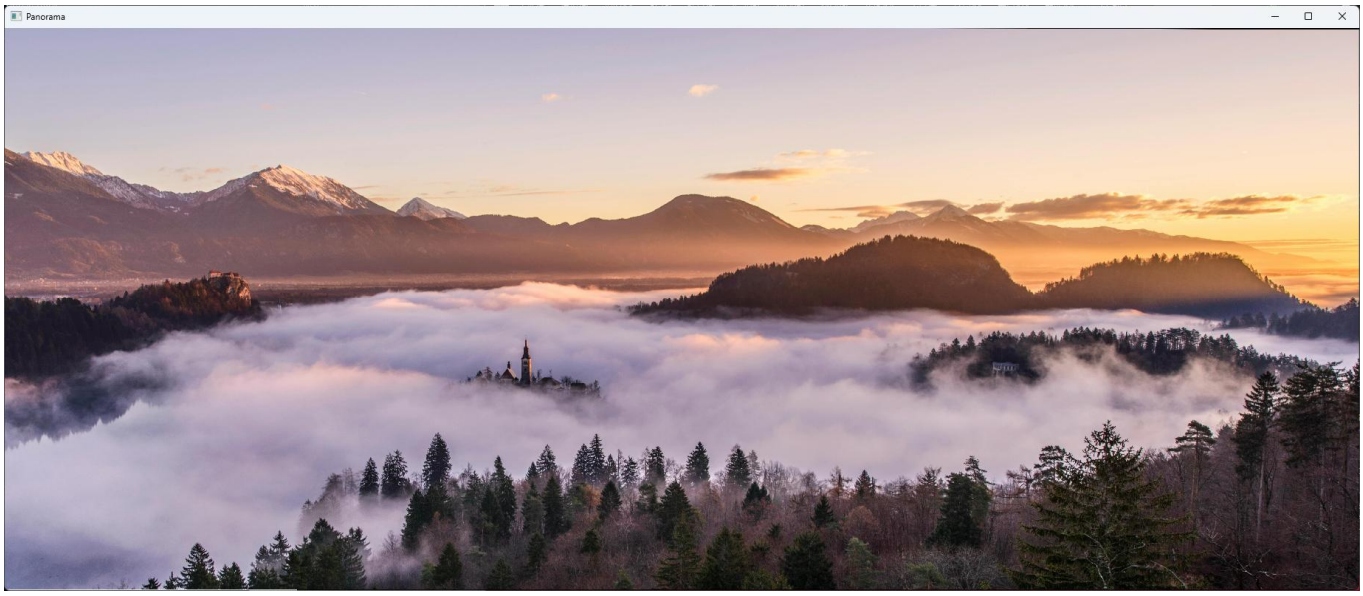
```
    //连续拼接
    Mat pano = imgs[0];
    for (size_t i = 1; i < imgs.size(); i++) pano = stitchPair(pano, imgs[i]);

    //显示与保存
    namedWindow("Panorama", WINDOW_NORMAL);
    imshow("Panorama", pano);
    imwrite("panorama_cropped.jpg", pano);
    waitKey(0);
    return 0;
}
```

五、实验结果显示

（1）简单图像

（2）困难图像

（3）自行采集图像（网上下载）



六、实验分析总结

1. 成功要点：
   - SIFT 算法在尺度与旋转变化场景下具有鲁棒性，能提取丰富匹配点；
   - 简单的最大值融合有效填补缝隙，消除了黑线；
   - 自动裁剪去除黑边，使结果更紧凑。
2. 改进空间：
   - 融合区域尚可采用权重渐变或多频带融合以消除亮度突变；
   - 对于高度透视失真或动态场景，可尝试多重投影模型或暴光补偿；
   - 增加并行加速，提高大规模图像拼接的效率。