****

**数字图像处理课程设计**

**题 目 无人机校园航拍全景图像拼接**

**课 程 名 数字图像处理**

**任课教师 李新胜**

**学 院 空天科学与工程学院**

**专 业 飞行器控制与信息工程**

**学生姓名 李铭茁**

**学 号 2017141511015 年级 2017级**

**评阅成绩**

**评审意见**

**报告提交日期 2019.12.31**

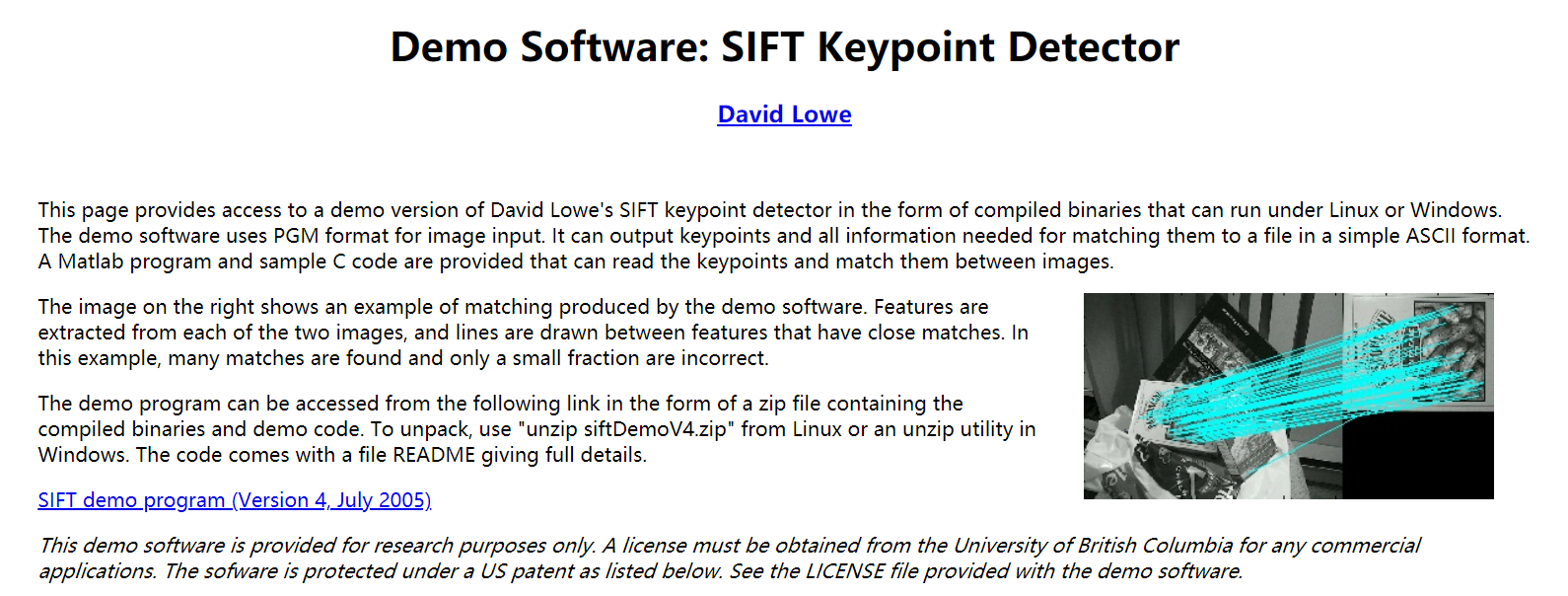
**课题背景**

随着无人机的普及，航拍成为越来越多人的业余爱好。为航拍得到全景图片，在MATLAB中利用SIFT算法与RANSAC算法，对无人机拍摄的校园图片实现全景图像拼接。

**实验方案**

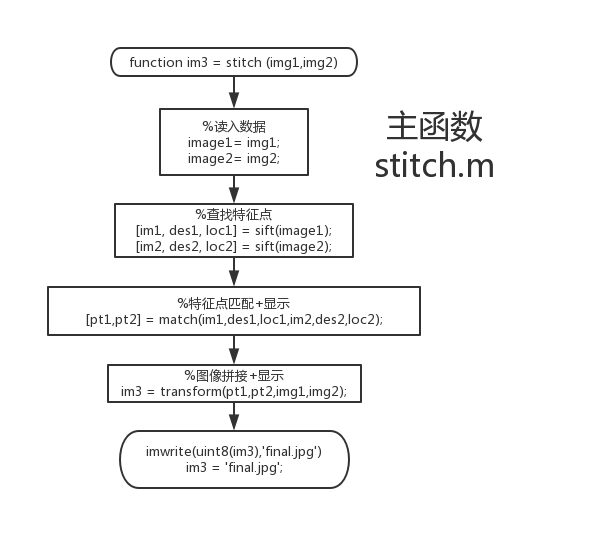
**1.材料准备**

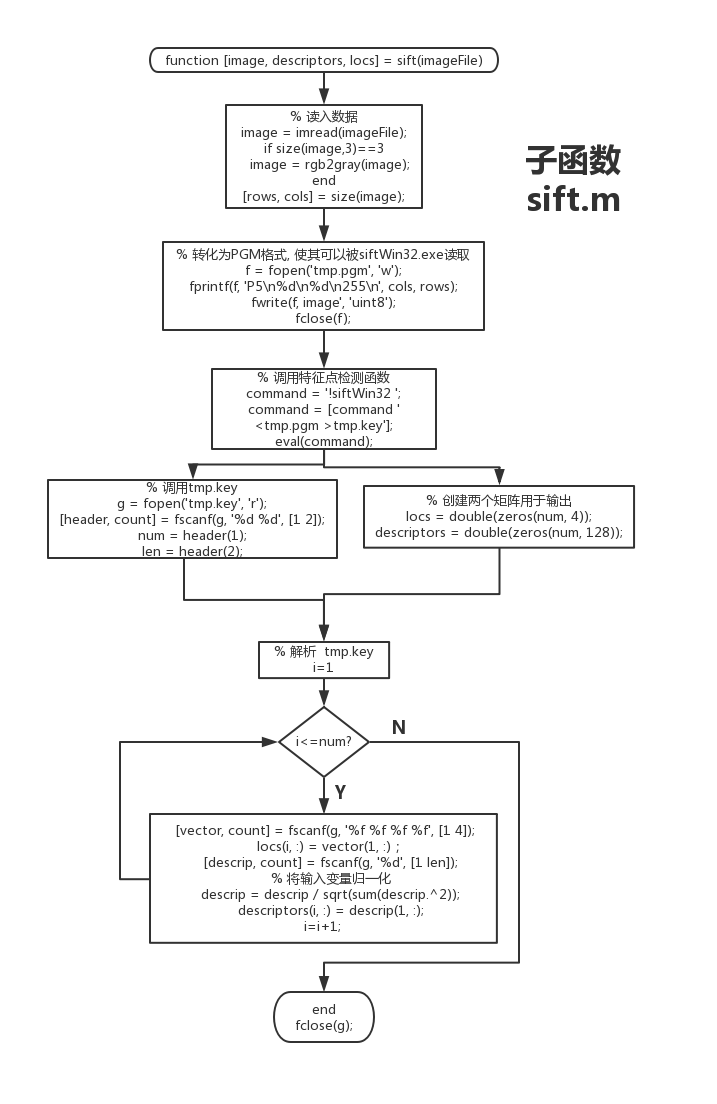
### SIFT算法的专利权归属于英属哥伦比亚大学，由David Lowe提出[1]。在David Lowe的个人主页，我找到了使用C编写的SIFT样例（siftDemoV4中含可执行文件siftWin32.exe，支持MATLAB）。<https://www.cs.ubc.ca/~lowe/keypoints/>

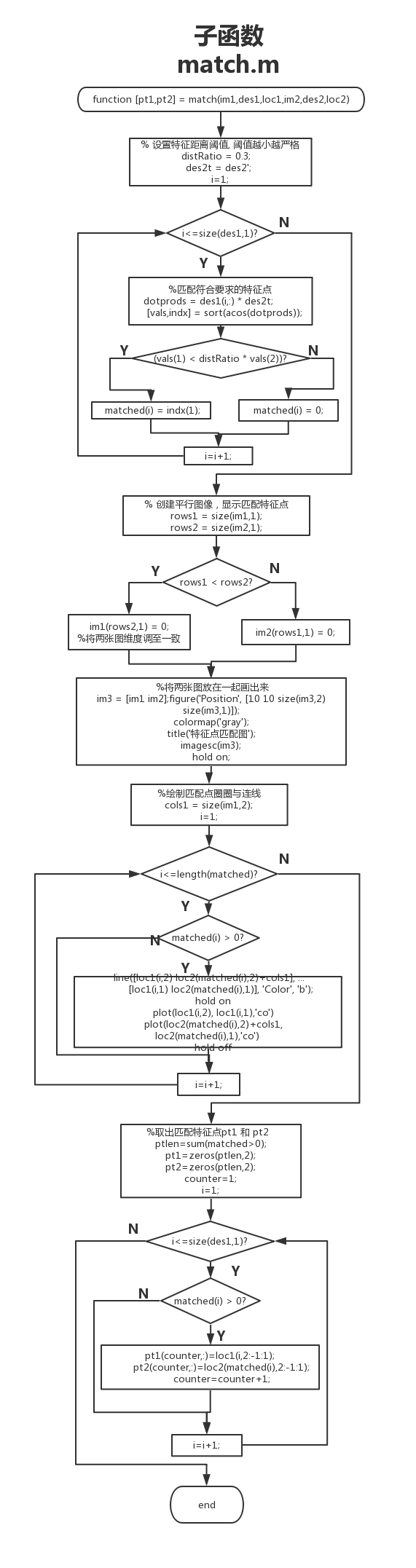
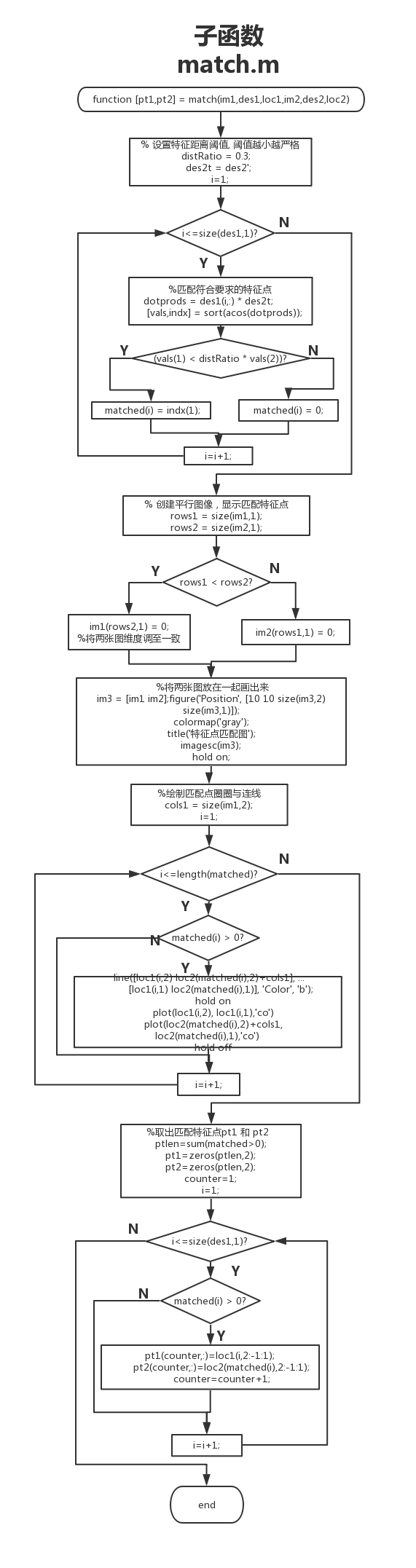


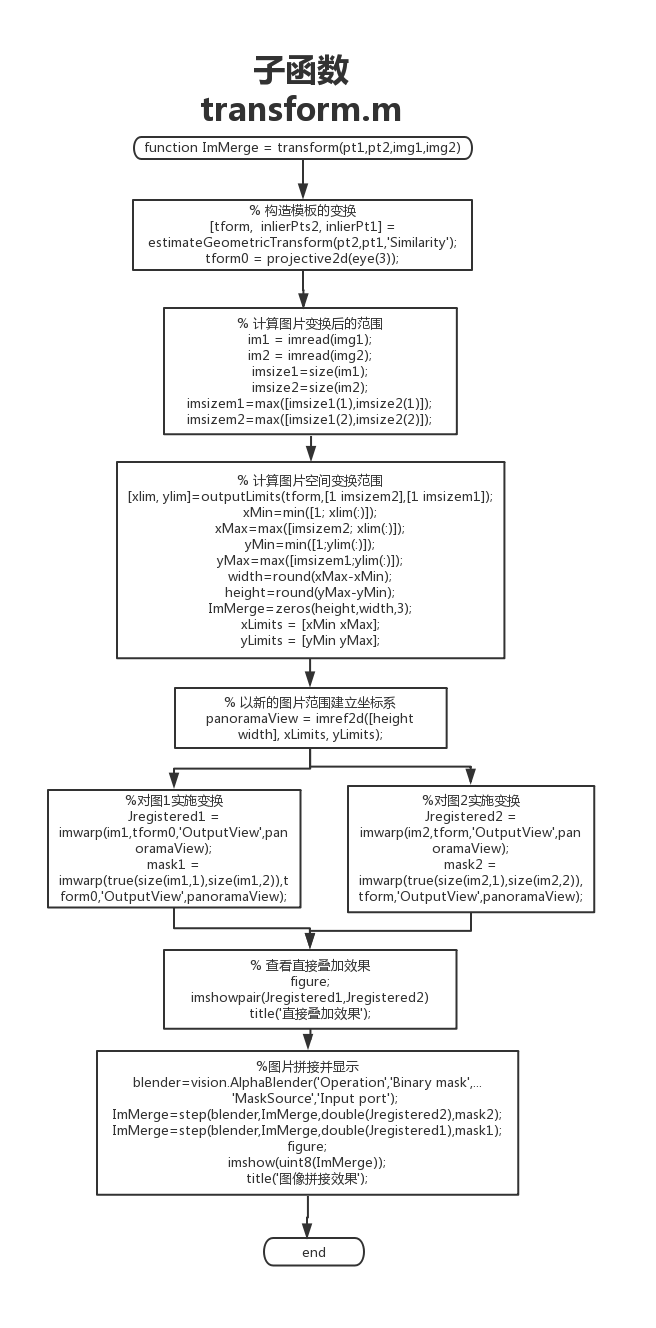
此外，通过自行拍摄和网络检索，得到共计两套四川大学校园图像用于实验。

1. **程序实现（程序流程图)**







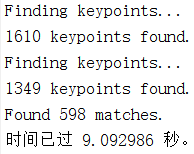


**实验结果与分析**

**实验一 望江校区北门双图拼接**

1. **实验结果**

在MATLAB命令行调用stitch函数：im3 = stitch('1.jpg','2.jpg');可得实验结果，包括图像、特征点数、匹配点数。同时，利用tic toc函数得到运行时间，如下图所示。



图一共计发现1610个特征点，图二共计发现1349个特征点。二者有598对

特征点符合实验所设定的特征距离阈值0.3。程序运行时间为9.092986s。



****

**原图**（望江校区北门） **特征点匹配图**（望江校区北门）



**叠加效果图**（望江校区北门） **最终效果图**（望江校区北门）

**2.结果分析**

1. 图一共计发现1610个特征点，图二共计发现1349个特征点。我认为检索到的特征点数量由图像尺寸决定其量级，由图像自身特征决定其具体大小。同时，我了解到相比SIFT算法，SURF算法和ORB算法的效率更高，相对提取的特征点数量少而精。尤其是SURF算法，作为SIFT算法的优化版本，其特征描述子减少了一半，大大提高了运行效率[2]。同时，opencv中图像拼接中特征点提取主要使用的就是SURF和ORB算法，而非较为原始的SIFT算法。得到结果后，对比其他人同样使用sift算法得到的结果，特征点数量符合预期。

（2）二者有598对特征点符合实验所设定的特征距离阈值0.3。在David Lowe的论文中给出的推荐特征距离阈值为0.2。为了提高容错率，我没有采用较为严格的阈值0.2，而是选择使用0.3作为特征距离阈值。结果表明，采用0.3而非0.2作为特征阈值，匹配到的特征点数量明显增多。（效果图中为0.3的匹配结果）

（3）程序运行时间为9.092986s。由于是图像拼接实验，我并未将效率问题作为首要考虑。经考证，当前主流图像拼接算法均能控制在1s以内，但即使是低分辨率图像也尚未达到30fps[3]。也就是说，图像拼接技术尚未真正实现，实时视频拼接。在本次实验中，利用siftDemoV4得到的9s运行时间，较为合理。

（4）由于两张图像为定点拍摄且主体相同，最终效果图较好，并未出现大面积的失真或者出现重影现象。但是，依旧出现了不希望的黑边。在实验二中，黑边现象将更加明显，因此之后的实验中我对此进行了修正。

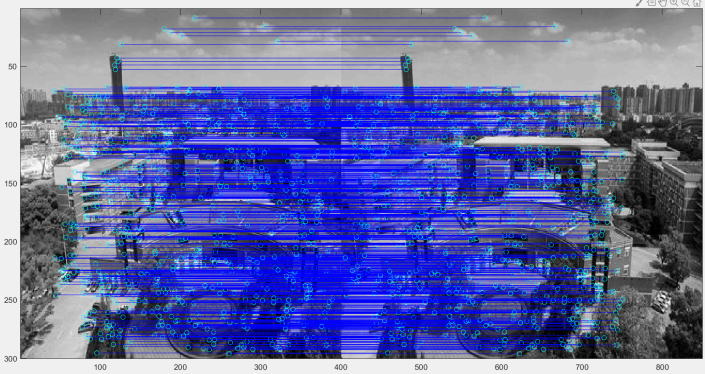
**实验二 望江校区航拍多图拼接**

**1.实验结果**

****

**原图**（航拍图1、2、3）

****

****

**特征点匹配图**（航拍图1、2） **特征点匹配图**（航拍图1、2、3）

****

**叠加效果图**（航拍图1、2） **叠加效果图**（航拍图1、2、3）

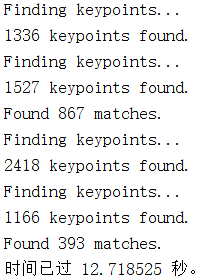
****

当前无人机通过自动控制系统，已经实现了一定程度上的“完美悬停”，因此拍摄可以得到

**最终效果图**（航拍图1、2） **最终效果图**（航拍图1、2、3）

**2.结果分析**

（1）连续调用两次stitch函数，即可实现三图拼接。多次调用，可实现多图拼接。im3 = stitch('1.jpg','2.jpg');im4 = stitch(im3,'3.jpg');实验结果中的特征点数、匹配点数、运行时间如下图所示。



注意到航拍图1、2的拼接图作为中间结果，所发现的特征点数为2418而航拍图1、2各自本身的特征点数分别为1336、1527。

理论上，拼接结果特征数≈（图1特征数+图2特征点数）-匹配点数。实际结果与理论结果相吻合。

1. 对于运算时间，我对两次拼接分别进行了统计，如下所示。

航拍图1、2拼接：

航拍图1、2拼接结果与图3：

第二次拼接时间相比第一次要略短上一些，我推断是由于作为参照图的图1、2拼接结果的特征点数较为充足，且尺寸较大。总体上看，两次拼接的时间差别并不是很大，符合预期。

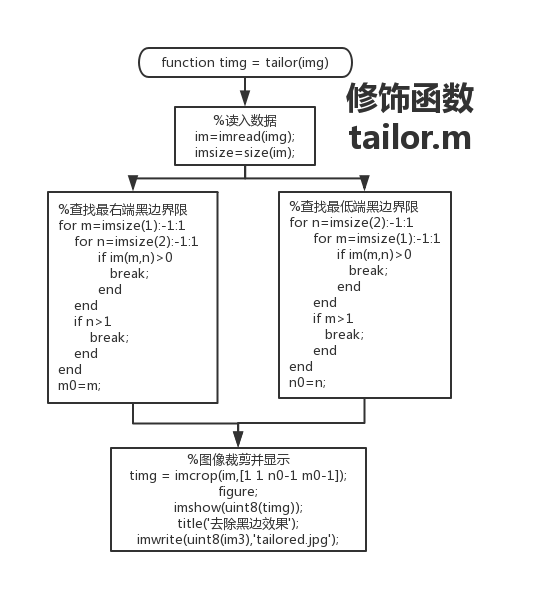
1. 当前市面上的主流无人机（如：大疆Mavic 2），已经实现了一定程度上的“完美悬停”。定点拍摄所得到的图像具有较大的相似度，图像拼接时也较为方便。因此实验所得到的全景图拼接完整，并未出现大面积的失真等现象[3]。总体上，全景图的拼接效果较为满意。
2. 但是最终效果图中还是出现了大面积的黑色边缘，对此我进行了实验修正。

**3.实验修正**

1. **修正方式**

观察发现，由于拼接图像时将左边图像作为基础主图，因此黑边往往拼接结果的右侧及下方。通过修饰函数tailor，检测黑边的范围，对拼接图象进行剪裁。

程序流程图，如下所示。

****

1. **修正结果与分析**

****

对比修正前的图像，这里主要对于图像右侧的黑边进行了修正。虽然依旧存在少量黑色边缘，但总体效果较为满意。

**实验总结**

（1）总体上，实验结果较为满意。但是，由于SIFT算法并非当前最优算法，可以进一步使用SURF、ORB等算法进行特征点提取，并进行算法间的对比分析。

（2）其次，如果希望高效实现360°的全景拼接，对于效率和鲁棒性要求都会较高。本次使用的图片相似度较高，因此效果较好，后续可对效率问题和鲁棒性问题进行进一步的优化。

（3）最后，Chin T J于2013年提出的APAP算法，根据将图像分为密集网格所具有的局部单应性，应用网格优化来解决图像拼接问题。同时，公开了MATLAB的源代码，便于利用假期时间进一步探究[4]。

**附录**

**参考文献**

1. David G.Lowe, Distinctive image features from scale-invariant

keypoints[J], International Journal of Computer Vision,2004

1. Bay,Herbert, Speed-Up Robust Features(SURF)[J], Computer vision and

image understanding,2008

[3]He B,Parallax-Robust Surveillance Video Stitching[J]. Sensors, 2015

[4]Zaragoza J, As-projective-as-possible image stitching with moving DLT[C]//CVPR,2013

**代码**

**主函数stitch.m**

function im3 = stitch (img1,img2)

%读入数据

image1= img1;

image2= img2;

%查找特征点

[im1, des1, loc1] = sift(image1);

[im2, des2, loc2] = sift(image2);

%特征点匹配+显示

[pt1,pt2] = match(im1,des1,loc1,im2,des2,loc2);

%图像拼接+显示

im3 = transform(pt1,pt2,img1,img2);

imwrite(uint8(im3),'final.jpg')

im3 = 'final.jpg';

**子函数sift.m**

function [image, descriptors, locs] = sift(imageFile)

% 读入数据

image = imread(imageFile);

if size(image,3)==3

image = rgb2gray(image);

end

[rows, cols] = size(image);

% 转化为PGM格式, 使其可以被siftWin32.exe读取

f = fopen('tmp.pgm', 'w');

fprintf(f, 'P5\n%d\n%d\n255\n', cols, rows);

fwrite(f, image', 'uint8');

fclose(f);

% 调用特征点检测函数

command = '!siftWin32 ';

command = [command ' <tmp.pgm >tmp.key'];

eval(command);

% 调用tmp.key

g = fopen('tmp.key', 'r');

[header, count] = fscanf(g, '%d %d', [1 2]);

num = header(1);

len = header(2);

% 创建两个矩阵用于输出

locs = double(zeros(num, 4));

descriptors = double(zeros(num, 128));

% 解析 tmp.key

for i = 1:num

[vector, count] = fscanf(g, '%f %f %f %f', [1 4]);

locs(i, :) = vector(1, :)；

[descrip, count] = fscanf(g, '%d', [1 len]);

% 将输入变量归一化

descrip = descrip / sqrt(sum(descrip.^2));

descriptors(i, :) = descrip(1, :);

end

fclose(g);

**子函数match.m**

function [pt1,pt2] = match(im1,des1,loc1,im2,des2,loc2)

% 设置特征距离阈值, 阈值越小越严格

distRatio = 0.3;

des2t = des2';

%匹配符合要求的特征点

for i = 1 : size(des1,1)

dotprods = des1(i,:) \* des2t;

[vals,indx] = sort(acos(dotprods));

if (vals(1) < distRatio \* vals(2))

matched(i) = indx(1);

else

matched(i) = 0;

end

end

% 创建平行图像，显示匹配特征点

rows1 = size(im1,1);

rows2 = size(im2,1);

if (rows1 < rows2)

im1(rows2,1) = 0;%将两张图维度调至一致

else

im2(rows1,1) = 0;

End

%将两张图放在一起画出来

im3 = [im1 im2];figure('Position', [10 10 size(im3,2) size(im3,1)]);

colormap('gray');

title('特征点匹配图');

imagesc(im3);

hold on;

%绘制匹配点圈圈与连线

cols1 = size(im1,2);

for i = 1: length(matched)

if (matched(i) > 0)

line([loc1(i,2) loc2(matched(i),2)+cols1], ...

[loc1(i,1) loc2(matched(i),1)], 'Color', 'b');

hold on

plot(loc1(i,2), loc1(i,1),'co')

plot(loc2(matched(i),2)+cols1, loc2(matched(i),1),'co')

hold off

end

end

hold off;

num = sum(matched > 0);

fprintf('Found %d matches.\n', num);

%取出匹配特征点pt1 和 pt2

ptlen=sum(matched>0);

pt1=zeros(ptlen,2);

pt2=zeros(ptlen,2);

counter=1;

for i = 1: size(des1,1)

if (matched(i) > 0)

pt1(counter,:)=loc1(i,2:-1:1);

pt2(counter,:)=loc2(matched(i),2:-1:1);

counter=counter+1;

end

end

**子函数transform.m**

function ImMerge = transform(pt1,pt2,img1,img2)

% 构造模板的变换

[tform, inlierPts2, inlierPt1] = estimateGeometricTransform(pt2,pt1,'Similarity');

tform0 = projective2d(eye(3));

% 计算图片变换后的范围

im1 = imread(img1);

im2 = imread(img2);

imsize1=size(im1);

imsize2=size(im2);

imsizem1=max([imsize1(1),imsize2(1)]);

imsizem2=max([imsize1(2),imsize2(2)]);

% 计算图片空间变换范围

[xlim, ylim]=outputLimits(tform,[1 imsizem2],[1 imsizem1]);

xMin=min([1; xlim(:)]);

xMax=max([imsizem2; xlim(:)]);

yMin=min([1;ylim(:)]);

yMax=max([imsizem1;ylim(:)]);

width=round(xMax-xMin);

height=round(yMax-yMin);

ImMerge=zeros(height,width,3);

xLimits = [xMin xMax];

yLimits = [yMin yMax];

% 以新的图片范围建立坐标系

panoramaView = imref2d([height width], xLimits, yLimits);

%对图1实施变换

Jregistered1 = imwarp(im1,tform0,'OutputView',panoramaView);

mask1 = imwarp(true(size(im1,1),size(im1,2)),tform0,'OutputView',panoramaView);

%对图2实施变换

Jregistered2 = imwarp(im2,tform,'OutputView',panoramaView);

mask2 = imwarp(true(size(im2,1),size(im2,2)),tform,'OutputView',panoramaView);

% 查看直接叠加效果

figure;

imshowpair(Jregistered1,Jregistered2)

title('直接叠加效果');

%图片拼接并显示

blender=vision.AlphaBlender('Operation','Binary mask',...

'MaskSource','Input port');

ImMerge=step(blender,ImMerge,double(Jregistered2),mask2);

ImMerge=step(blender,ImMerge,double(Jregistered1),mask1);

figure;

imshow(uint8(ImMerge));

title('图像拼接效果');

**修饰函数tailor.m**

function timg = tailor(img)

%读入数据

im=imread(img);

imsize=size(im);

%查找最右端黑边界限

for m=imsize(1):-1:1

for n=imsize(2):-1:1

if im(m,n)>0

break;

end

end

if n>1

break;

end

end

m0=m;

%查找最底端黑边界限

for n=imsize(2):-1:1

for m=imsize(1):-1:1

if im(m,n)>0

break;

end

end

if m>1

break;

end

end

n0=n;

%图像裁剪并显示

timg = imcrop(im,[1 1 n0-1 m0-1]);

figure;

imshow(uint8(timg));

title('去除黑边效果');

imwrite(uint8(im3),'tailored.jpg');