东南大学自动化学院

机器视觉第一次课程设计报告

——图像分类与全景图像生成

姓　　名： 董胜豪　　**学　　号：** 08020328

目录

1 设计需求

按照全景图像采集要求，采集不同场景的多组图像，场景数量不少于三个，每个场景图像不少于三张，设计全景图像识别与图像拼接算法，实现图像按场景分组以及对应场景图像拼接。

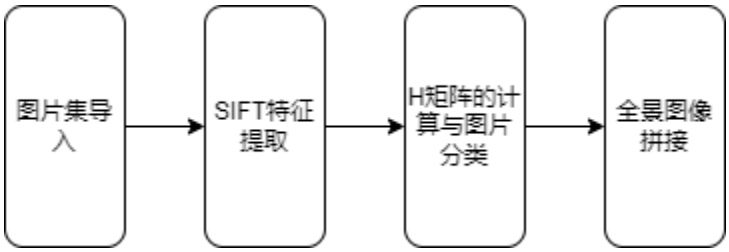
2 算法设计

2.1 流程分析

基于 SIFT 特征和 RANSAC 算法的全景图像拼接技术是一种常用的全景图像拼接方法。SIFT（尺度不变特征变换）是一种用于图像特征提取和匹配的算法，能够提取出具有尺度不变性、旋转不变性和光照不变性的图像特征。RANSAC（随机抽样一致性）是一种用于估计模型参数的算法，能够从一组包含噪声和异常值的数据中估计出最佳模型参数。

基于 SIFT 特征和 RANSAC 算法的全景图像拼接技术主要包括以下步骤：首先，对多张图片进行 SIFT 特征提取和匹配，得到每张图片之间的匹配点对。然后，使用 RANSAC 算法从匹配点对中估计出每张图片之间的变换矩阵，以保证它们在全景图像中的位置和方向正确。最后，使用图像拼接算法将多张图片拼接在一起，形成一个完整的全景图像。

基于 SIFT 特征和 RANSAC 算法的全景图像拼接技术能够有效地处理图像之间的错位和重叠问题，得到较为准确和稳定的全景图像。该技术在许多应用领域得到了广泛应用，例如虚拟现实、地图制作、室内设计等。



2.2 SIFT 特征提取

SIFT（Scale-Invariant Feature Transform）特征是一种用于计算计算机视觉中的局部特征的算法。它可以在不同的缩放尺度和旋转角度下提取出稳定的特征点，并且对于光照、噪声等因素也具有一定的鲁棒性。

SIFT 算法的基本流程是：

尺度空间极值检测：通过不同尺度的高斯滤波器来检测图像中的极值点，这些点在不同尺度下具有稳定的特征。

关键点定位：在尺度空间中找到关键点，这些关键点是在不同尺度下的极值点，并且具有高斯差分值的局部极大值。

方向分配：为每个关键点分配一个主方向，用于后续的特征描述。

特征描述：在每个关键点周围的区域内提取局部特征向量，该向量可以表示关键点周围区

域的特征。
SIFT 特征在图像匹配、物体识别、图像检索等领域有广泛的应用。

2.2.1 代码实现

```
% Read the first image from the image set.
for i = 1:length(img.Files)
    I= readimage(img,i);
    % Initialize features for I(1)
    grayImage = im2gray(I);
    points = detectSIFTFeatures(grayImage);
    [features, points] = extractFeatures(grayImage,points);
    n = 1000;
    %random_features = features(randperm(points.Count,n),:);
    %random_num = sort(random_num);
    %random_features=reshape(random_features,[1,n*128]);
    features_list{i,1}=features;
    points_list{i,1}=points;
end
```

2.3 H 矩阵计算与图片分类

2.3.1 H 矩阵计算

单应矩阵（homography matrix）是计算机视觉中的一个重要概念，用于描述两个平面之间的投影变换关系。在计算机视觉中，我们经常需要对不同视角或者不同摄像机拍摄到的图像进行配准（registration）或者叠加（overlay）。这时候就需要将一个图像中的特征点通过一个变换映射到另一个图像中。这个变换就可以由单应矩阵来描述。

单应矩阵可以将一个平面上的点映射到另一个平面上，它由一个 3x3 的矩阵表示，其形式如下：

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

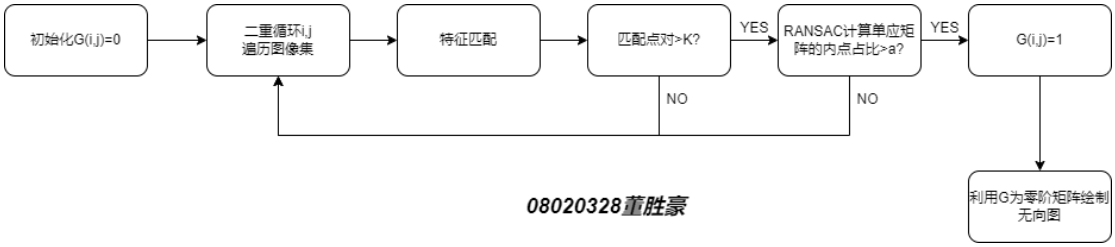
其中，h11，h12，h13，h21，h22，h23，h31，h32 和 h33 是矩阵中的 9 个元素，通过这些元素，可以描述一个平面到另一个平面的仿射变换或者透视变换关系。在计算单应矩阵时，通常使用最小二乘法或者 RANSAC 算法来求解。

在此我们选择使用 RANSAC 算法求解，RANSAC 算法通过迭代的方式，在每次迭代中随机抽取一部分数据来拟合模型，然后用该模型去评估所有数据点，并将符合某个阈值条件的点归为内点（inliers），不符合条件的点称为外点（outliers）。通过重复这个过程，RANSAC 算法可以找到符合内点数量达到最大的最优模型。

在进行单应矩阵计算之前，首先要对两幅图片进行特征匹配，特征匹配的过程是匹配两幅图像中对应的 SIFT 特征，同时需要注意的是一个特征点只能与一个特征点相匹配，并且需要尽可能的消除模糊匹配。

2.3.2 图像分类

图像分类过程采用了一种自己思考的方法实现，对于特征的匹配，如果不是同一个场景下的图像，那么匹配的特征点对的数目则会比较小，这也是所符合我们认知的，但是如果单单利用匹配的特征点对图片的分类结果相比较于实际会有比较大的出入，因此我们使用匹配的特征点数目进行一个初步判断。如果通过初步判断，那么就利用 RANSAC 算法计算单应矩阵，利用返回的内点数目占比，如果内点数目占比大于一定值，那么则可以认为此两幅图像属于同一类别。



具体的实现方法如上图所示，而最终输出的 G 矩阵，可以看作一个无向图的邻接矩阵，那么绘制出这个无向图，求出这个无向图的连通分量，即可对图像进行分类。
利用这个算法思想，可以实现全自动的图像分类，不需要定义种类数于每一类的而图片数目，但是需要定义 K 值的大小和 a 值的大小，最终分类的效果与这两个参数的关系很大。

2.3.3 代码实现

```
function [H, corrPtIdx] = CalcH(pts1, pts2)

Points=[pts1.Location,pts2.Location];
coef.minPtNum = 4;
coef.iterNum = 50;
coef.thDist = 4;
coef.thInlrRatio = 0.1;
% 使用RANSAC算法求解最优的H
%[H, corrPtIdx] = ransac(pts1, pts2, coef, @CalcHDetail, @calcDist);
%仿射变换
if pts1.Count < coef.minPtNum
    H=0;
    corrPtIdx=false;
    return
end
[H, corrPtIdx] = ransac(Points,@CalcHDetail,@calcDist,4,4);
%h
end
```

利用 ransac 算法进行单应矩阵的求解，代码实现利用 matlab 中 ransac()函数实现，模型需要自己构建，求解 Projective transformations 的单应矩阵所需要八个自由度，提供四个点输入即可求解。

```
function H = CalcHDetail(Points)
pts1=Points(:,1:2)';
pts2=Points(:,3:4)';
% tform = fitgeotform2d(pts1',pts2',"projective");
% H=tform.A;
n = size(pts1,2);
A = zeros(2*n,9);
A(1:2:2*n,1:2) = pts1';
A(1:2:2*n,3) = 1;
A(2:2:2*n,4:5) = pts1';
A(2:2:2*n,6) = 1;
x1 = pts1(1,:)';
y1 = pts1(2,:)';
x2 = pts2(1,:)';
y2 = pts2(2,:)';
A(1:2:2*n,7) = -x2.*x1;
A(2:2:2*n,7) = -y2.*x1;
A(1:2:2*n,8) = -x2.*y1;
A(2:2:2*n,8) = -y2.*y1;
A(1:2:2*n,9) = -x2;
A(2:2:2*n,9) = -y2;
[evec,~] = eig(A'*A);
```

上图为通过四个点求解单应矩阵的计算过程。

```
for i = 1:length(img.Files)
    for j = 1:length(img.Files)
        [a,b]=matchFeatures(features_list{i,1},features_list{j,1},'MaxRatio',0.2
        p1=points_list{i,1};
        p2=points_list{j,1};
        matchedPoints1 = p1(a(:,1),:);
        matchedPoints2 = p2(a(:,2),:);
        num_match=numel(a)/2;
        K=0.01/128*max([numel(features_list{i,1}),numel(features_list{j,1})]);%0
        if num_match>K
            [H, corrPtIdx] = CalcH(matchedPoints1,matchedPoints2);
            table = tabulate(corrPtIdx);
            if table{end,3}>10
                cell_H{i,j}=H;
                Graph_classify(i,j)=1;
            end
        end

    end
```

循环遍历 imageDatastore 中的所有图像对。对于每对图像，它使用 matchFeatures 函数查找两张图像之间对应的 SIFT 特征，使用 CalcH 函数计算单应性矩阵 H，如果匹配点的数量大于一个阈值 K 且至少有 10 个匹配点，则将 H 存储在 cell_H 中。

然后创建一个图，其中每个图像都由一个节点表示，并在两个节点之间添加一条边，如果两个图像之间存在单应性矩阵 H。使用 conncomp 计算图的连通分量。

最终输出 bin 和 binsize 是图的连通分量及其大小。

2.4 全景图像生成

2.4.1 matlab 生成全景图像

下图全景图像的生成过程参考 matlab 官方示例 Feature Based Panoramic Image Stitching 实现，具体过程如流程图所示

考虑到在 2.3 中求出的子图并不是完全两两连通的，因此我们需要找到子图之中的一条连接所有点的路径，也就是找到一个最小生成树，后即可计算出任意两幅图片之间的单应矩阵。

关键代码：

求解全景图像中间图像编号

```
for i = 1:numel(tforms)
    [xlim(i,:), ylim(i,:)] = outputLimits(tforms(i), [1 imageSize(i,2)], [1 imageSize(i,1)]);
end
avgXLim = mean(xlim, 2);
[~,idx] = sort(avgXLim);
centerIdx = floor((numel(tforms)+1)/2);
centerImageIdx = idx(centerIdx);
Tinv = invert(tforms(centerImageIdx));
for i = 1:numel(tforms)
    tforms(i).A = Tinv.A * tforms(i).A;
end
for i = 1:numel(tforms)
    [xlim(i,:), ylim(i,:)] = outputLimits(tforms(i), [1 imageSize(i,2)], [1 imageSize(i,1)]);
end

maxImageSize = max(imageSize);
%maxImageSize = sum(imageSize);

% Find the minimum and maximum output limits.
xMin = min([1; xlim(:)]);
xMax = max([maxImageSize(2); xlim(:)]);
```

全景图像拼接，在全景图像拼接的过程之中，将每一个图像经过单应矩阵变换后映射投影到全景画幅之上，为了克服即便，考虑使用圆柱面投影的方法来减小畸变，具体的投影方法采用 matlab 之中的 AlphaBlender 函数进行，通过参数选择对两个图像的重合部分进行线性结合以优化拼接结果。

```
panorama = zeros([height width 3], 'like', I);
%blender = vision.AlphaBlender('Operation','Blend','MaskSource', 'Input port');
blender = vision.AlphaBlender('Operation', 'Binary mask', ...
    'MaskSource', 'Input port');

xLimits = [xMin xMax];
yLimits = [yMin yMax];
panoramaView = imref2d([height width], xLimits, yLimits);

% Create the panorama.
for i = 1:numImages

    I = readimage(img, i);

    % Transform I into the panorama.
    warpedImage = imwarp(I, tforms(i), 'OutputView', panoramaView);

    % Generate a binary mask.
    mask = imwarp(true(size(I,1),size(I,2)), tforms(i), 'OutputView', panoramaView);

    % Overlay the warpedImage onto the panorama.
    %panorama = step(blender, panorama, warpedImage);%,mask);
    panorama = step(blender, panorama, warpedImage, mask);
end
```

柱面投影代码

```
for n = 1:length(img.Files)
    I=readimage(img,n);
    [height, width, depth] = size(I);
    % A = zeros(size(I));
    A = I;
    centerX = width / 2;
    centerY = height / 2;
    alpha = pi / 8;
    f = width / (2 * tan(alpha/2));
    for i = 1 : width
        for j = 1 : height
            theta = asin((i - centerX) / f);
            pointX = int32(f * tan((i - centerX) / f) + centerX);
            pointY = int32((j - centerY) / cos(theta) + centerY);
            if pointX >= 1 && pointX <= width && pointY >= 1 && pointY <= height
                for k = 1:depth
                    A(j,i,k) = I(pointY,pointX,k);
                end
            else
                for k = 1:depth
                    A(j,i,k) = 0;
                end
            end
        end
    end
```

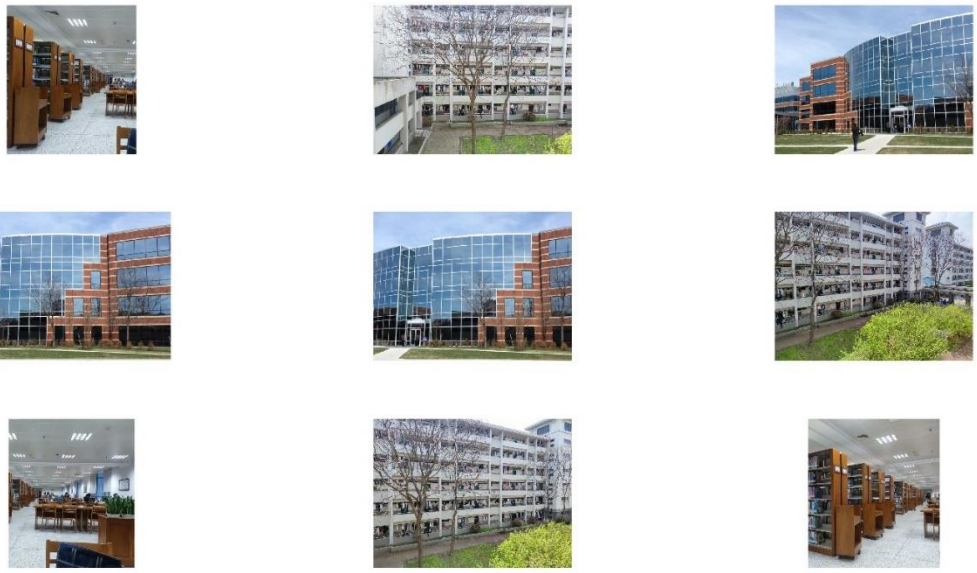
2.4.2 OpenCV 生成全景图像

完成基本图像拼接后，可以发现拼接结果还是存在瑕疵，例如接缝处理和畸变的处理，因此利用 OpenCV 中的 stitch 函数进行对比

```
import cv2
import sys
import os
#filename = sys.argv[1]
filename = '新建文件夹'
imagelist = os.listdir(filename)
image_paths=imagelist
#image_paths=['3.jpg','4.jpg','5.jpg']
# initialized a list of images
imgs = []

for i in range(len(image_paths)):
    imgs.append(cv2.imread(filename + '/' + image_paths[i]))
    imgs[i]=cv2.resize(imgs[i],(0,0),fx=0.4,fy=0.4)

# showing the original pictures
cv2.imshow('1',imgs[0])
cv2.imshow('2',imgs[1])
cv2.imshow('3',imgs[2])

stitchy=cv2.Stitcher.create()
(dummy,output)=stitchy.stitch(imgs)

if dummy != cv2.STITCHER_OK:

    print("stitching ain't successful")
else:
    print('Your Panorama is ready!!!')
```

2.5 实验结果

拼接素材展示

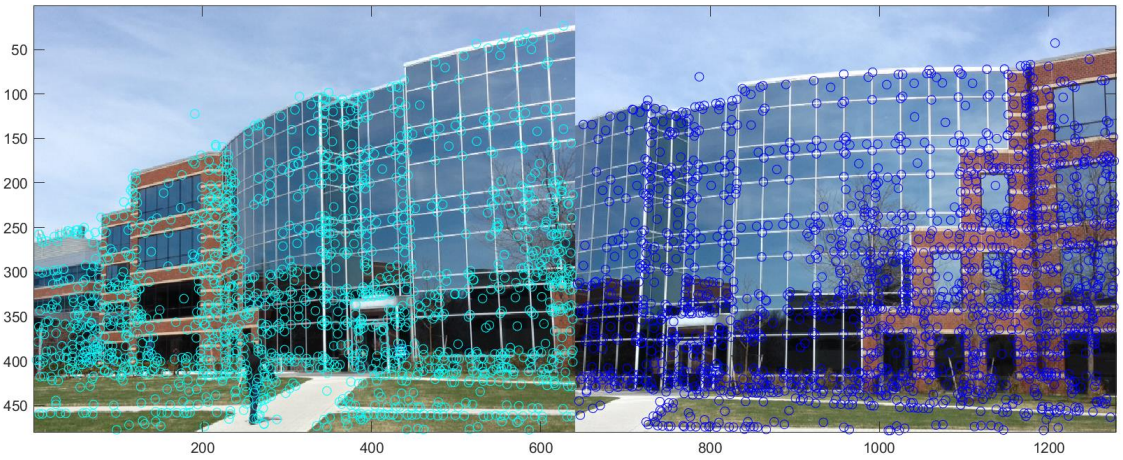


共九张图片，每组三张，打乱顺序存储在同一个文件夹中

2.5.1 SIFT 特征点提取与匹配

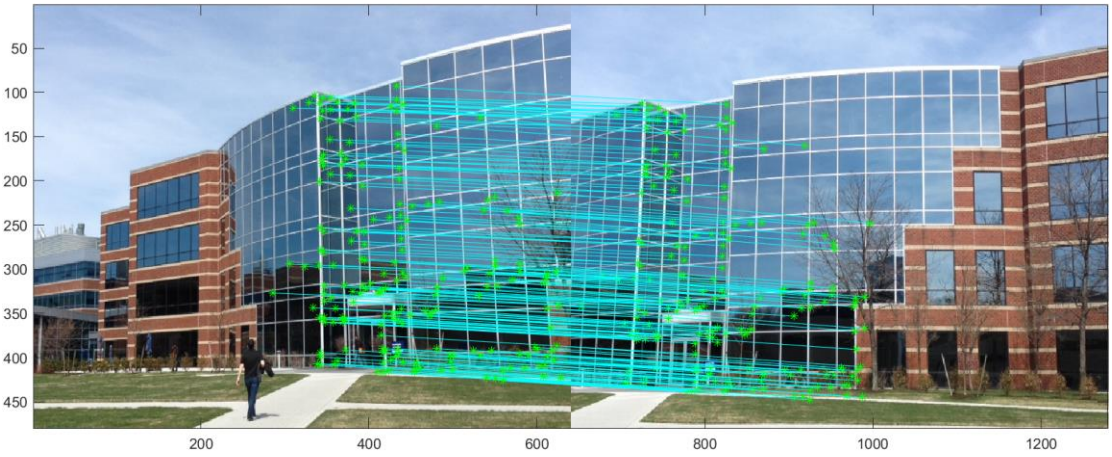
以其中的一类图像为例

SIFT 特征点



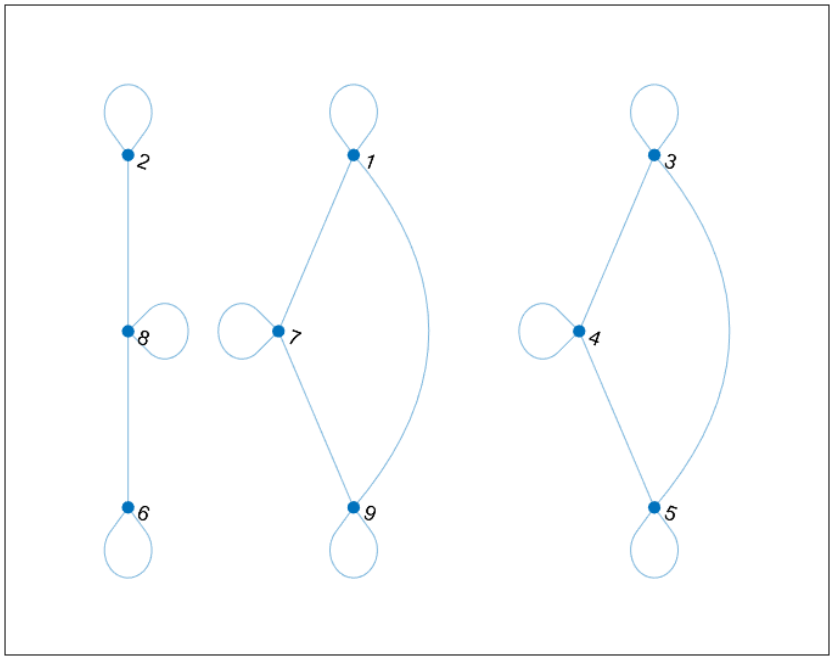
特征点匹配

Ransac 算法求解单应矩阵的内点



2.5.2 图像分类

求解单应矩阵后利用邻接矩阵做出无向图



共将 9 张图片分成三类，无向图存在三个连通分量
对每一个连通分量提取对应的图片，由求出的单应矩阵进行全景图像拼接。

2.5.3 全景图像拼接

Matlab 拼接结果

Opencv 实现

3 总结

通过拼接的结果我们可以看出，matlab 中采取直接拼接的方法和 opencv 中函数 stitch 互有优劣。直接拼接，即不采用柱面投影的方法在处理视角较广的照片拼接是会出现图像的畸变，并且没有采用多波段融合（Multiban blending）的方法，会出现一定的拼接缝隙。Opencv 中的 stitch 函数很明显采用了柱面投影加多波段融合的方法，效果相对更好，但是对于视角较窄的图片集进行拼接时，会出现很大的图像畸变，因此在实际运用的过程中需要酌情考虑视角的问题以及算力问题。