电子科技大学成都学院

实验报告

课程名称 数字图像处理

分 院 计算机学院

年 级 2020级

专　　业 人工智能

班 级 三班、四班

任课教师 陈东祥

2021年12月制

选题：图像拼接

小组成员：黄月、陈彦洁、李贝、敬炳华、白苏彤、陈为铮、郑榕松

分工：PPT制作：陈彦洁、黄月

资料搜寻：李贝、敬炳华、白苏彤

报告：陈为铮、郑榕松

摘 要

图像拼接技术作为计算机视觉和数字图像处理技术的重要基础之一,现如今已经成为计算机图像领域的重要研究内容。图像拼接技术是将多张相互有重叠部分的图像,根据重合区域选取合适的方法进行匹配和融合,最终拼接成一张无缝隙的图像。该技术已经广泛应用于图像分析、场景虚拟和事物勘测等方面。本文以图像拼接技术为研究背景,针对图像配准过程中特征点提取速度慢、匹配准确度低以及在图像融合过程中图像有重影的问题,研究并设计性能更好的图像匹配和图像融合算法,以此提高图像拼接的效率和准确度。

关键词：图像拼接、SIFT原理

Abstract

 Image Mosaic technology, as one of the important foundations of computer vision and digital image processing technology, has become an important research content in the field of computer images. Image Mosaic technology is to select the appropriate method to match and fuse multiple overlapping images according to the overlapping area, and finally Mosaic into a seamless image. This technology has been widely used in image analysis, scene virtual and object survey. In this paper, the image Mosaic technology as the research background, aiming at the problem of slow extraction of feature points in the process of image registration, low matching accuracy and image ghosting in the image fusion process, the research and design of image matching and image fusion algorithm with better performance, so as to improve the efficiency and accuracy of image Mosaic

Keywords: image mosaic、SIFT theory

目录

[第一章 引言 6](#_Toc90661066)

[第二章 研究的理论和基础 8](#_Toc90661067)

[基本思路 8](#_Toc90661068)

[一．SIFT特征点提取： 8](#_Toc90661069)

[1．定义： 8](#_Toc90661070)

[2．背景： 8](#_Toc90661071)

[3.SIFT算法特点： 9](#_Toc90661072)

[4.SIFT算法实质 9](#_Toc90661073)

[5.SIFT算法描述 9](#_Toc90661074)

[6.SIFT算法实现特征匹配主要有以下三个流程： 25](#_Toc90661075)

[二．特征匹配 26](#_Toc90661076)

[（1）特征匹配的作用： 26](#_Toc90661077)

[（2）特征匹配的方法： 26](#_Toc90661078)

[（3）特征匹配的前提： 26](#_Toc90661079)

[三．BFMatcher 27](#_Toc90661080)

[1.BFMatcher含义： 27](#_Toc90661081)

[2.实现原理 27](#_Toc90661082)

[四．KNN算法原理及实现 28](#_Toc90661083)

[1、KNN算法概述 28](#_Toc90661084)

[2.KNN算法介绍 28](#_Toc90661085)

[（1）KNN的产生 28](#_Toc90661086)

[（2）KNN算法思路 28](#_Toc90661087)

[（3）举例说明算法 29](#_Toc90661088)

[3、KNN算法的思想总结 30](#_Toc90661089)

[4.KNN缺点 30](#_Toc90661090)

[5. KNNMatch 30](#_Toc90661091)

[五．透视变换 31](#_Toc90661092)

[【知识点】一、线性变化: 31](#_Toc90661093)

[【知识点】二、透视变换透视矩阵的一般表达形式为： 31](#_Toc90661094)

[【了解】三、几种典型的仿射变换： 32](#_Toc90661095)

[【知识点】五、根据对应坐标求透视矩阵 35](#_Toc90661096)

[【实例解释】单应性矩阵（H矩阵） 36](#_Toc90661097)

[六．代码实现效果及解释 36](#_Toc90661098)

[第三章 图像拼接技术的不足 42](#_Toc90661099)

[第四章 基于本图像拼接技术的优化 42](#_Toc90661100)

[第五章 研究展望 42](#_Toc90661101)

[致谢 42](#_Toc90661102)

1. 引言

1.1图像拼接技术的拼接背景

数字图像处理方法的研究源于两个主要应用领域:其一是为了使于人们分析而对图像信息进行改进:其二是为了便于机器自动理解而对图像数据进行存储、传输及显示"。早在20世纪20年代,数字图像处理就曾用于报纸业,当时,图像传输第一次引入 Bartlane 电缆把一幅图像通过海底电缆从伦敦传往纽约。这是数字图像的最早应用。随着计算机处理能力的不断提高,硬件装置价格的不断下降,数字图像技术广泛地用于医学图像、地球遥感监测和天文学领域。它所涉及的内容有点运算、几何处理、图像增强、图像复原、图像形态学处理、图像编码、图像重建、模式识别等。  
 图像拼接技术最早应用于遥感技术领域,由于遥感监测所用的摄像机仅能拍摄某一地区局部区域,为能获得大范围的遥感图像,需要将拍投"为区域图像通过拼接以获得一幅完整的遥感图像。而当我们使用普通相机拍摄照片时,必须调节焦距才可以拍摄到完整的场景,但这样拍得的照片分辨率较低。反之,如果焦距调近,我们又拍不到一幅完整的场景照片。然而全景相机、广角镜头等设备不仅价格昂贵,而且失真也比较严重,所以,人们就想到用多幅图片拼成一张大的图片。

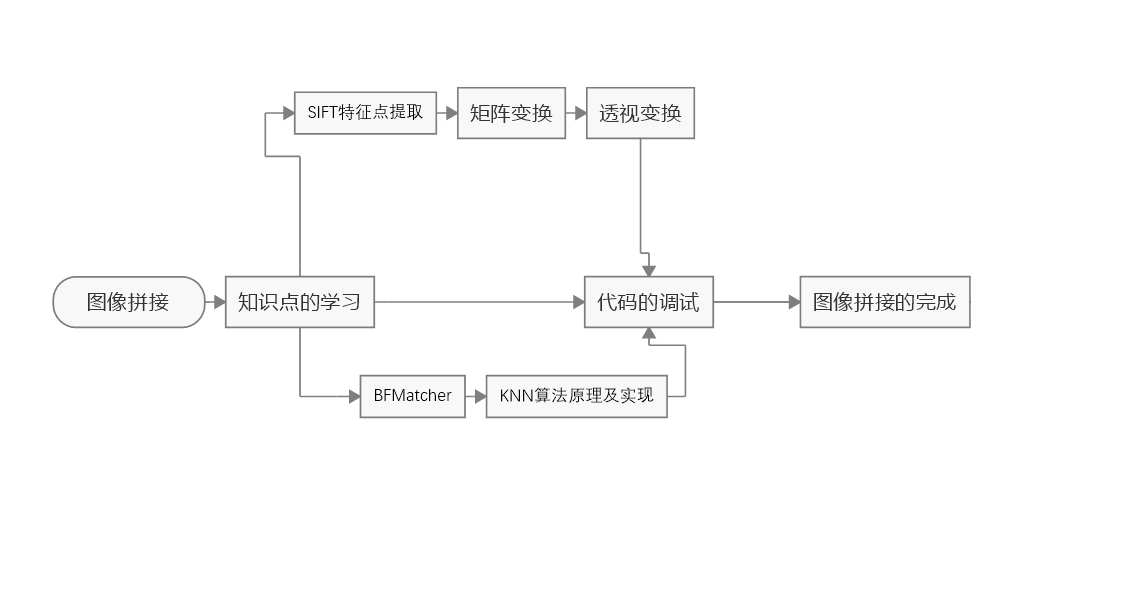
1.2 图像拼接技术的应用领域

图像拼接技术在宇宙空间探测、海底勘测、医学、气象、地质勘测、军事、视频压缩和传输、档案的数字化保存、视频的索引和检索、物体的三维重建、军事侦察和公安取证、数码相机的超分辨处理等领域都有广泛的应用。主要变现为：

1. 遥感图像处理  
   随着航空、航天技术的发展,侦查卫星或航空遥感器能够实时获得目标的高清晰图像,为获得高质量、高分辨率的图像,需要将来自不同传感器的两幅或多幅遥感图像拼接成一幅影像图。遥感图像最典型的应用是天气预报。地球接收站每天两次在固定时间里接受某一轨道的卫星云图,几条轨道的图像拼接成区域云图,是预报员的重要参考资料。例如,通过将卫星图片或航空图片拼接成大范围的场景图片来实现对某一地区整体检测。
2. 医学图像分析  
   这个方面的应用主要有三个,一是外科手术,通过图像拼接可将标准的解剖图谱叠加到病人的数据上,这样可以帮助医生进行外科手术的规划。例如视网膜激光手术,由于缺少对视网膜整体状况进行检查的工具,成功率低,有学者提出了利用图像拼接的方法构建视网膜的整体视图8;二是脑功能区的定位,在心理学、药理学等其它的实验中,往往需要观察大脑在某一刺激下,某些区域产生的变化情况,由于这是一项需要多人共同参与的统计学意义下的实验,因而需要将许多形态各异的 MRI 图像放到同一个参考系统中,这当然离不开图像配准技术;三是脑结构变化的研究,人的性别、年龄及所患的疾病将导致大脑的解剖结构的差异,通过图像配准,可以对这种差别进行定量分析,从而有助于从数量上阐明相应机理。
3. 视频监控  
   在日常生活中,图像拼接技术可用于视频监控系统。将多个摄像头获取的图像合成一幅宽视野的高分辨率的全景图,可以实现现场操作人员和指挥专家之间的远程协助系统和远程遥控系统，现场工作人员通过将拍摄得的视频用无线通讯方式传递给远程专家,远程专家在收到视频图像后构建出现场全景图,然后根据现场情况提出建议并通知现场工作人员进行  
    军事领试中的监控视频大气大层造出来的全景图可以用来描述场景中的运动目标。

第二章 研究的理论和基础

## 基本思路



## 一．SIFT特征点提取：

### 1．定义：

SIFT即尺度不变特征变换，是用于图像处理领域的一种描述。这种描述具有尺度不变性，可在图像中检测出关键点，是一种局部特征描述子。

### 2．背景：

SIFT算法由D.G.Lowe 1999年提出，2004年完善总结。后来Y.Ke将其描述子部分用PCA代替直方图的方式，对其进行改进。

### 3.SIFT算法特点：

(1)、具有较好的稳定性和不变性，能够适应旋转、尺度缩放、亮度的变化，能在一定程度上不受视角变化、仿射变换、噪声的干扰。

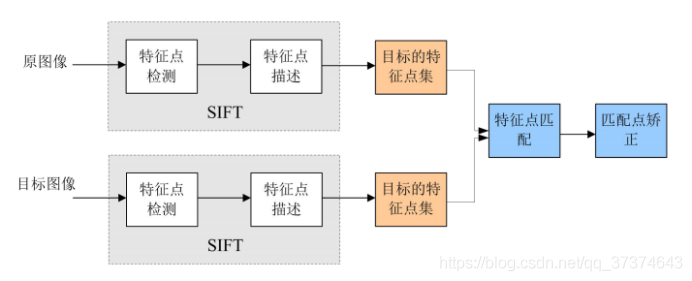
(2)、区分性好，能够在海量特征数据库中进行快速准确的区分信息进行匹配

(3)、多量性，就算只有单个物体，也能产生大量特征向量

(4)、高速性，能够快速的进行特征向量匹配

(5)、可扩展性，能够与其它形式的特征向量进行联合

### 4.SIFT算法实质

在不同的尺度空间上查找关键点，并计算出关键点的方向。  


### 5.SIFT算法描述

SIFT特征不只具有尺度不变性，即使改变旋转角度，图像亮度或拍摄视角，仍然能够得到好的检测效果。整个算法分为以下几个部分：

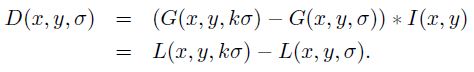
1. 构建尺度空间

这是一个初始化操作，尺度空间理论目的是模拟图像数据的多尺度特征。

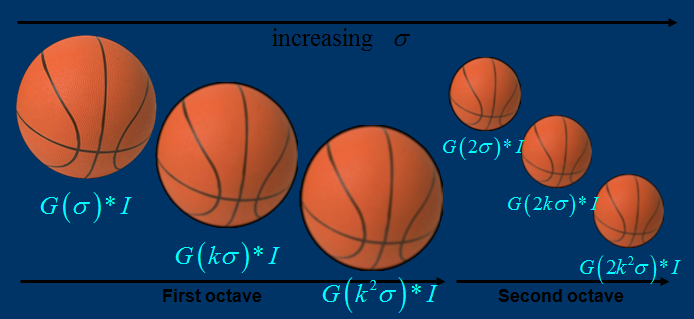
高斯卷积核是实现尺度变换的唯一线性核，于是一副二维图像的尺度空间定义为：



其中 G(x,y,σ) 是尺度可变高斯函数 x，y）是空间坐标，是尺度坐标。σ大小决定图像的平滑程度，大尺度对应图像的概貌特征，小尺度对应图像的细节特征。大的σ值对应粗糙尺度(低分辨率)，反之，对应精细尺度(高分辨率)。为了有效的在尺度空间检测到稳定的关键点，提出了高斯差分尺度空间（DOG scale-space）。利用不同尺度的高斯差分核与图像卷积生成。

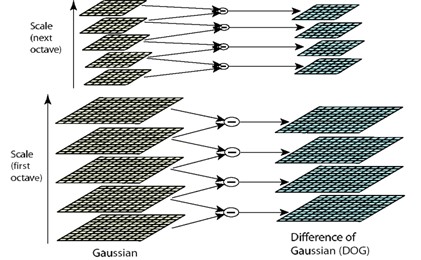


下图所示不同σ下图像尺度空间：



关于尺度空间的理解说明：2kσ中的2是必须的，尺度空间是连续的。在 Lowe的论文中 ，将第0层的初始尺度定为1.6（最模糊），图片的初始尺度定为0.5（最清晰）. 在检测极值点前对原始图像的高斯平滑以致图像丢失高频信息，所以 Lowe 建议在建立尺度空间前首先对原始图像长宽扩展一倍，以保留原始图像信息，增加特征点数量。尺度越大图像越模糊。

图像金字塔的建立：对于一幅图像I,建立其在不同尺度(scale)的图像，也成为子八度（octave），这是为了scale-invariant，也就是在任何尺度都能够有对应的特征点，第一个子八度的scale为原图大小，后面每个octave为上一个octave降采样的结果，即原图的1/4（长宽分别减半），构成下一个子八度（高一层金字塔）。



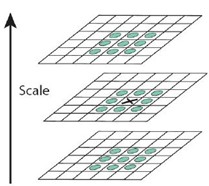


尺度空间的所有取值，i为octave的塔数（第几个塔），s为每塔层数

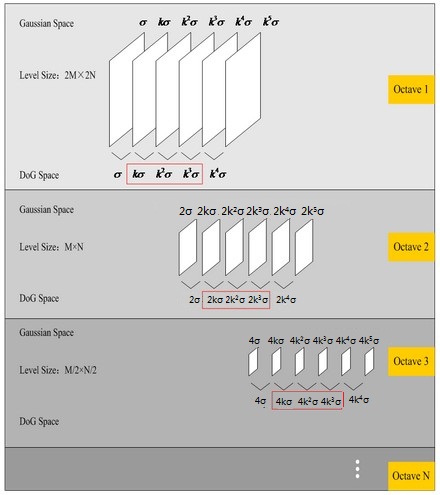
由图片size决定建几个塔，每塔几层图像(S一般为3-5层)。0塔的第0层是原始图像(或你double后的图像)，往上每一层是对其下一层进行Laplacian变换（高斯卷积，其中σ值渐大，例如可以是σ, k\*σ, k\*k\*σ…），直观上看来越往上图片越模糊。塔间的图片是降采样关系，例如1塔的第0层可以由0塔的第3层down sample得到，然后进行与0塔类似的高斯卷积操作。

2. LoG近似DoG找到关键点<检测DOG尺度空间极值点>

为了寻找尺度空间的极值点，每一个采样点要和它所有的相邻点比较，看其是否比它的图像域和尺度域的相邻点大或者小。如图所示，中间的检测点和它同尺度的8个相邻点和上下相邻尺度对应的9×2个点共26个点比较，以确保在尺度空间和二维图像空间都检测到极值点。 一个点如果在DOG尺度空间本层以及上下两层的26个领域中是最大或最小值时，就认为该点是图像在该尺度下的一个特征点,如图所示。



同一组中的相邻尺度（由于k的取值关系，肯定是上下层）之间进行寻找



在极值比较的过程中，每一组图像的首末两层是无法进行极值比较的，为了满足尺度变化的连续性（下面有详解）

，我们在每一组图像的顶层继续用高斯模糊生成了 3 幅图像，高斯金字塔有每组S+3层图像。DOG金字塔每组有S+2层图像.

==========================================

这里有的童鞋不理解什么叫“为了满足尺度变化的连续性”，现在做仔细阐述：

假设s=3，也就是每个塔里有3层，则k=21/s=21/3，那么按照上图可得Gauss Space和DoG space 分别有3个（s个）和2个（s-1个）分量，在DoG space中，1st-octave两项分别是σ,kσ; 2nd-octave两项分别是2σ,2kσ;由于无法比较极值，我们必须在高斯空间继续添加高斯模糊项，使得形成σ,kσ,k2σ,k3σ,k4σ这样就可以选择DoG space中的中间三项kσ,k2σ,k3σ（只有左右都有才能有极值），那么下一octave中（由上一层降采样获得）所得三项即为2kσ,2k2σ,2k3σ，其首项2kσ=24/3。刚好与上一octave末项k3σ=23/3尺度变化连续起来，所以每次要在Gaussian space添加3项，每组（塔）共S+3层图像，相应的DoG金字塔有S+2层图像。

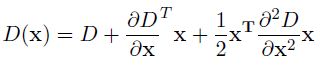
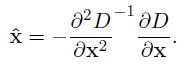
==========================================

使用Laplacian of Gaussian能够很好地找到找到图像中的兴趣点，但是需要大量的计算量，所以使用Difference of Gaussian图像的极大极小值近似寻找特征点.DOG算子计算简单，是尺度归一化的LoG算子的近似,有关DOG寻找特征点的介绍及方法详见

3. 除去不好的特征点

这一步本质上要去掉DoG局部曲率非常不对称的像素。

通过拟和三维二次函数以精确确定关键点的位置和尺度（达到亚像素精度），同时去除低对比度的关键点和不稳定的边缘响应点(因为DoG算子会产生较强的边缘响应)，以增强匹配稳定性、提高抗噪声能力，在这里使用近似Harris Corner检测器。

①空间尺度函数泰勒展开式如下：，对上式求导,并令其为0,得到精确的位置, 得

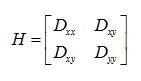
②在已经检测到的特征点中,要去掉低对比度的特征点和不稳定的边缘响应点。去除低对比度的点：把公式(2)代入公式(1)，即在DoG Space的极值点处D(x)取值，只取前两项可得：



若  ，该特征点就保留下来，否则丢弃。

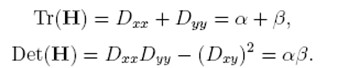
③边缘响应的去除

一个定义不好的高斯差分算子的极值在横跨边缘的地方有较大的主曲率，而在垂直边缘的方向有较小的主曲率。主曲率通过一个2×2 的Hessian矩阵H求出:



导数由采样点相邻差估计得到。

D的主曲率和H的特征值成正比，令α为较大特征值，β为较小的特征值，则



令α=γβ，则



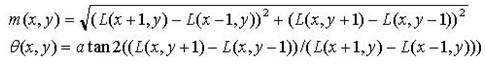
(r + 1)2/r的值在两个特征值相等的时候最小，随着r的增大而增大，因此，为了检测主曲率是否在某域值r下，只需检测



if (α+β)/ αβ> (r+1)2/r, throw it out. 在Lowe的文章中，取r＝10。

4. 给特征点赋值一个128维方向参数

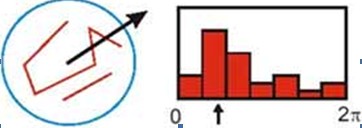
上一步中确定了每幅图中的特征点，为每个特征点计算一个方向，依照这个方向做进一步的计算， 利用关键点邻域像素的梯度方向分布特性为每个关键点指定方向参数，使算子具备旋转不变性。



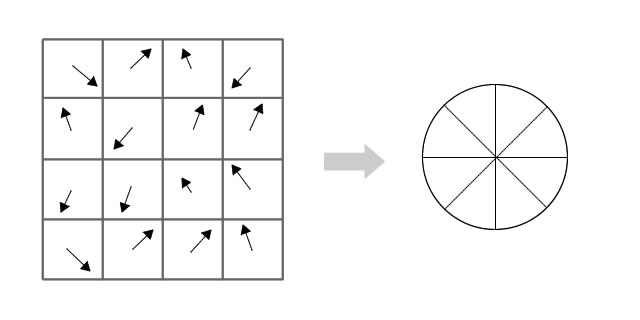
为(x,y)处梯度的模值和方向公式。其中L所用的尺度为每个关键点各自所在的尺度。至此，图像的关键点已经检测完毕，每个关键点有三个信息：位置，所处尺度、方向，由此可以确定一个SIFT特征区域。

梯度直方图的范围是0～360度，其中每10度一个柱，总共36个柱。随着距中心点越远的领域其对直方图的贡献也响应减小.Lowe论文中还提到要使用高斯函数对直方图进行平滑，减少突变的影响。

在实际计算时，我们在以关键点为中心的邻域窗口内采样，并用直方图统计邻域像素的梯度方向。梯度直方图的范围是0～360度，其中每45度一个柱，总共8个柱, 或者每10度一个柱，总共36个柱。Lowe论文中还提到要使用高斯函数对直方图进行平滑，减少突变的影响。直方图的峰值则代表了该关键点处邻域梯度的主方向，即作为该关键点的方向。

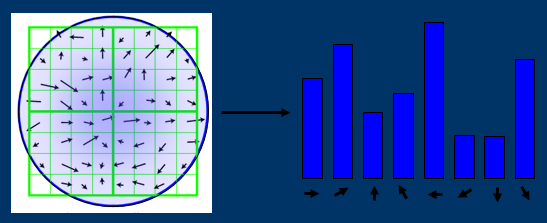


直方图中的峰值就是主方向，其他的达到最大值80%的方向可作为辅助方向



由梯度方向直方图确定主梯度方向

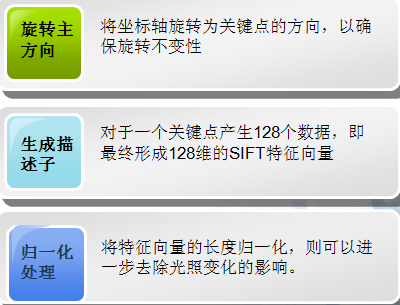
该步中将建立所有scale中特征点的描述子（128维）



Identify peak and assign orientation and sum of magnitude to key point.

The user may choose a threshold to exclude key points based on their assigned sum of magnitudes.

关键点描述子的生成步骤



通过对关键点周围图像区域分块，计算块内梯度直方图，生成具有独特性的向量，这个向量是该区域图像信息的一种抽象，具有唯一性。

5. 关键点描述子的生成

首先将坐标轴旋转为关键点的方向，以确保旋转不变性。以关键点为中心取8×8的窗口。

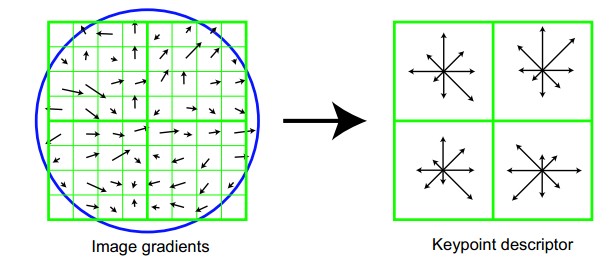
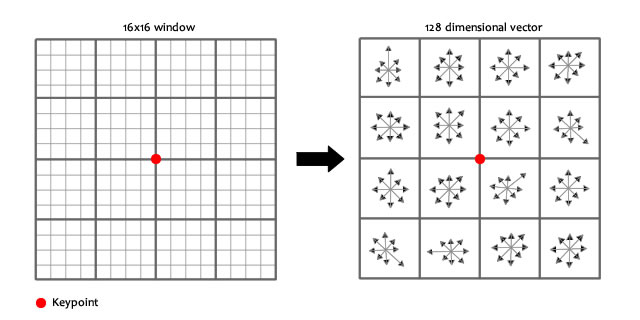


Figure.16\*16的图中其中1/4的特征点梯度方向及scale，右图为其加权到8个主方向后的效果。

图左部分的中央为当前关键点的位置，每个小格代表关键点邻域所在尺度空间的一个像素，利用公式求得每个像素的梯度幅值与梯度方向，箭头方向代表该像素的梯度方向，箭头长度代表梯度模值，然后用高斯窗口对其进行加权运算。

图中蓝色的圈代表高斯加权的范围（越靠近关键点的像素梯度方向信息贡献越大）。然后在每4×4的小块上计算8个方向的梯度方向直方图，绘制每个梯度方向的累加值，即可形成一个种子点，如图右部分示。此图中一个关键点由2×2共4个种子点组成，每个种子点有8个方向向量信息。这种邻域方向性信息联合的思想增强了算法抗噪声的能力，同时对于含有定位误差的特征匹配也提供了较好的容错性。

计算keypoint周围的16\*16的window中每一个像素的梯度，而且使用高斯下降函数降低远离中心的权重。



在每个4\*4的1/16象限中，通过加权梯度值加到直方图8个方向区间中的一个，计算出一个梯度方向直方图。

这样就可以对每个feature形成一个4\*4\*8=128维的描述子，每一维都可以表示4\*4个格子中一个的scale/orientation. 将这个向量归一化之后，就进一步去除了光照的影响。

5. 根据SIFT进行Match

生成了A、B两幅图的描述子，（分别是k1\*128维和k2\*128维），就将两图中各个scale（所有scale）的描述子进行匹配，匹配上128维即可表示两个特征点match上了。

实际计算过程中，为了增强匹配的稳健性，Lowe建议对每个关键点使用4×4共16个种子点来描述，这样对于一个关键点就可以产生128个数据，即最终形成128维的SIFT特征向量。此时SIFT特征向量已经去除了尺度变化、旋转等几何变形因素的影响，再继续将特征向量的长度归一化，则可以进一步去除光照变化的影响。 当两幅图像的SIFT特征向量生成后，下一步我们采用关键点特征向量的欧式距离来作为两幅图像中关键点的相似性判定度量。取图像1中的某个关键点，并找出其与图像2中欧式距离最近的前两个关键点，在这两个关键点中，如果最近的距离除以次近的距离少于某个比例阈值，则接受这一对匹配点。降低这个比例阈值，SIFT匹配点数目会减少，但更加稳定。为了排除因为图像遮挡和背景混乱而产生的无匹配关系的关键点,Lowe提出了比较最近邻距离与次近邻距离的方法,距离比率ratio小于某个阈值的认为是正确匹配。因为对于错误匹配,由于特征空间的高维性,相似的距离可能有大量其他的错误匹配,从而它的ratio值比较高。Lowe推荐ratio的阈值为0.8。但作者对大量任意存在尺度、旋转和亮度变化的两幅图片进行匹配，结果表明ratio取值在0. 4~0. 6之间最佳，小于0. 4的很少有匹配点，大于0. 6的则存在大量错误匹配点。(如果这个地方你要改进，最好给出一个匹配率和ration之间的关系图，这样才有说服力)作者建议ratio的取值原则如下:

ratio=0. 4　对于准确度要求高的匹配；

ratio=0. 6　对于匹配点数目要求比较多的匹配；

ratio=0. 5　一般情况下。

也可按如下原则:当最近邻距离<200时ratio=0. 6，反之ratio=0. 4。ratio的取值策略能排分错误匹配点。

当两幅图像的SIFT特征向量生成后，下一步我们采用关键点特征向量的欧式距离来作为两幅图像中关键点的相似性判定度量。取图像1中的某个关键点，并找出其与图像2中欧式距离最近的前两个关键点，在这两个关键点中，如果最近的距离除以次近的距离少于某个比例阈值，则接受这一对匹配点。降低这个比例阈值，SIFT匹配点数目会减少，但更加稳定。

===============================

基本概念及一些补充

什么是局部特征？

　　•局部特征从总体上说是图像或在视觉领域中一些有别于其周围的地方

　　•局部特征通常是描述一块区域，使其能具有高可区分度

　　•局部特征的好坏直接会决定着后面分类、识别是否会得到一个好的结果

局部特征需具备的特性

　　•重复性

　　•可区分性

　　•准确性

　　•数量以及效率

　　•不变性

局部特征提取算法-sift

　　•SIFT算法由D.G.Lowe 1999年提出，2004年完善总结。后来Y.Ke将其描述子部分用PCA代替直方图的方式，对其进行改进。

　　•SIFT算法是一种提取局部特征的算法，在尺度空间寻找极值点，提取位置，尺度，旋转不变量

　　•SIFT特征是图像的局部特征，其对旋转、尺度缩放、亮度变化保持不变性，对视角变化、仿射变换、噪声也保持一定程度的稳定性。

　　•独特性好，信息量丰富，适用于在海量特征数据库中进行快速、准确的匹配。

　　•多量性，即使少数的几个物体也可以产生大量SIFT特征向量。

　　•可扩展性，可以很方便的与其他形式的特征向量进行联合。

尺度空间理论

　　•尺度空间理论目的是模拟图像数据的多尺度特征

　　•其基本思想是在视觉信息图像信息处理模型中引入一个被视为尺度的参数, 通过连续变化尺度参数获得不同尺度下的视觉处理信息, 然后综合这些信息以深入地挖掘图像的本质特征。

描述子生成的细节

　　•以极值点为中心点，并且以此点所处于的高斯尺度sigma值作为半径因子。对于远离中心点的梯度值降低对其所处区域的直方图的贡献，防止一些突变的影响。

　　•每个极值点对其进行三线性插值，这样可以把此极值点的贡献均衡的分到直方图中相邻的柱子上

归一化处理

　　•在求出4\*4\*8的128维特征向量后，此时SIFT特征向量已经去除了尺度变化、旋转等几何变形因素的影响。而图像的对比度变化相当于每个像素点乘上一个因子，光照变化是每个像素点加上一个值，但这些对图像归一化的梯度没有影响。因此将特征向量的长度归一化，则可以进一步去除光照变化的影响。

　　•对于一些非线性的光照变化，SIFT并不具备不变性，但由于这类变化影响的主要是梯度的幅值变化，对梯度的方向影响较小，因此作者通过限制梯度幅值的值来减少这类变化造成的影响。

PCA-SIFT算法

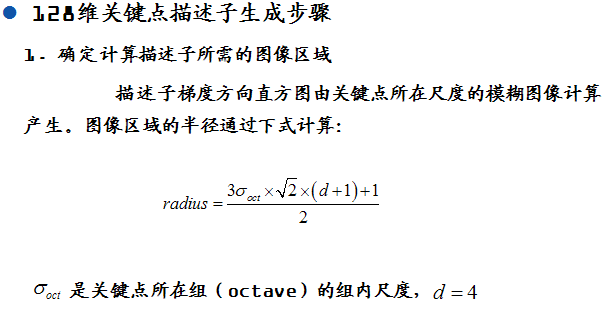
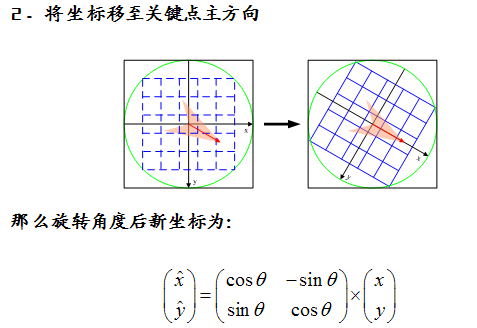
　　•PCA-SIFT与标准SIFT有相同的亚像素位置，尺度和主方向。但在第4步计算描述子的设计，采用的主成分分析的技术。

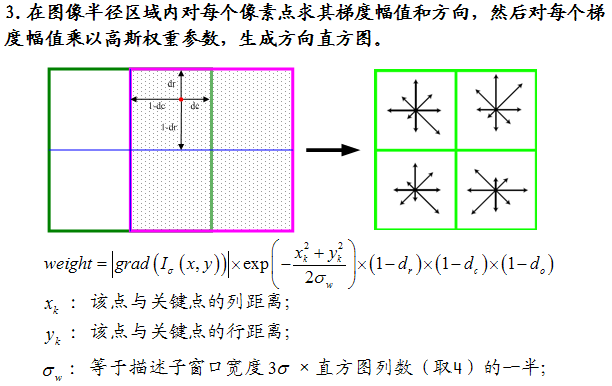
　　•下面介绍一下其特征描述子计算的部分：

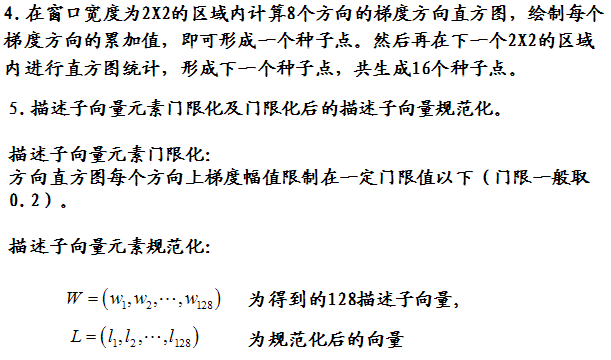
　　　　•用特征点周围的41×41的像斑计算它的主元，并用PCA-SIFT将原来的2×39×39维的向量降成20维，以达到更精确的表示方式。

　　　　•它的主要步骤为，对每一个关键点：在关键点周围提取一个41×41的像斑于给定的尺度，旋转到它的主方向 ；计算39×39水平和垂直的梯度，形成一个大小为3042的矢量；用预先计算好的投影矩阵n×3042与此矢量相乘；这样生成一个大小为n的PCA-SIFT描述子。

===============================





### 6.SIFT算法实现特征匹配主要有以下三个流程：

(1)、提取关键点：关键点是一些十分突出的不会因光照、尺度、旋转等因素而消失的点，比如角点、边缘点、暗区域的亮点以及亮区域的暗点。此步骤是搜索所有尺度空间上的图像位置。通过高斯微分函数来识别潜在的具有尺度和旋转不变的兴趣点。

(2)、定位关键点并确定特征方向：在每个候选的位置上，通过一个拟合精细的模型来确定位置和尺度。关键点的选择依据于它们的稳定程度。然后基于图像局部的梯度方向，分配给每个关键点位置一个或多个方向。所有后面的对图像数据的操作都相对于关键点的方向、尺度和位置进行变换，从而提供对于这些变换的不变性。

(3). 通过各关键点的特征向量，进行两两比较找出相互匹配的若干对特征点，建立景物间的对应关系。

## 二．特征匹配

### （1）特征匹配的作用：

1).在图像拼接中，需要进行特征匹配，方便求出单应矩阵以拼接两幅图像；

2).在三维重建中，需要进行特征匹配，方便求出变换矩阵以及三角化特征点；

3).在图像检索中，可以通过特征点在数据库中检索，查找到特征匹配数量最多的识别结果；

### （2）特征匹配的方法：

在本文中，只介绍两种常用的方法。

1. Flann快速最近邻搜索匹配法；

2．BruteForce暴力匹配法；（找出特征点、特征向量、然后欧氏距离计算特征向量间距离）

### （3）特征匹配的前提：

特征描述子的特异性和可重复性：特征描述子是在图像中利用邻域信息对关键点进行描述，得到一组描述向量。通常对于两帧具有部分重复纹理的地方，同一个位置的特征描述子应该是一样的。但是由于光照、遮挡、旋转和尺度等变化，导致同样的地方看起来却又不太一样。但是总体来说，只要是同一个地方，特征描述子的差别就不应该太大。

相似度度量方法：基于这个事实，特征匹配的目的就是在两个有重复纹理的图像中，将同一个位置的特征描述子关联起来。通常特征描述子是一组向量，比如 x=x1,x2,...,xnx=x1,x2,...,xn，同理，另一张图像中的同个位置描述向量为 y=y1,y2,...,yny=y1,y2,...,yn，将其在超平面中画出来，就可以通过度量其距离来判断其相似度。比如 n=2/3n=2/3 时，在坐标系中画出来，可以非常直观的看出来两个向量的距离。

## 三．BFMatcher

### 1.BFMatcher含义：

BFMatche总是尝试所有可能的匹配，从而使得它总能够找到最佳匹配，这也是Brute Force（暴力法）的原始含义。

而FlannBasedMatcher中FLANN的含义是Fast Library forApproximate Nearest Neighbors，从字面意思可知它是一种近似法，[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "_blank)更快但是找到的是最近邻近似匹配，所以当我们需要找到一个相对好的匹配但是不需要最佳匹配的时候往往使用FlannBasedMatcher。当然也可以通过调整FlannBasedMatcher的参数来提高匹配的精度或者提高算法速度，但是相应地算法速度或者算法精度会受到影响。

### 2.实现原理

* 发现两幅图片分别提取出来N，M个特征向量
* 然后对N和M的特征向量进行匹配，找到最佳匹配
* 然后再画出匹配的特征显示出来

## 四．[KNN算法原理及实现](https://www.cnblogs.com/sxron/p/5451923.html)

### 1、KNN算法概述

KNN算法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

### 2.KNN算法介绍

### （1）KNN的产生

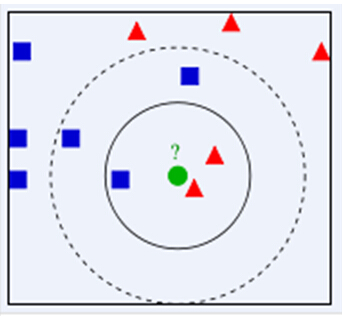
 　　最简单最初级的分类器是将全部的训练数据所对应的类别都记录下来，当测试对象的属性和某个训练对象的属性完全匹配时，便可以对其进行分类。但是怎么可能所有测试对象都会找到与之完全匹配的训练对象呢，其次就是存在一个测试对象同时与多个训练对象匹配，导致一个训练对象被分到了多个类的问题，基于这些问题呢，就产生了KNN。

### （2）KNN算法思路

   KNN是通过测量不同特征值之间的距离进行分类。它的的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。K通常是不大于20的整数。KNN算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

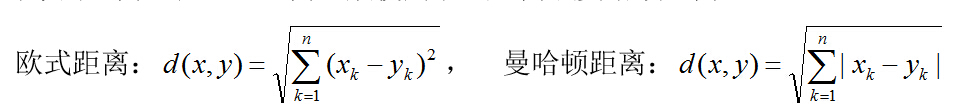
### （3）举例说明算法

     下面通过一个简单的例子说明一下：如下图，绿色圆要被决定赋予哪个类，是红色三角形还是蓝色四方形？如果K=3，由于红色三角形所占比例为2/3，绿色圆将被赋予红色三角形那个类，如果K=5，由于蓝色四方形比例为3/5，因此绿色圆被赋予蓝色四方形类。（后者的k=5，所以就在绿色的圆周围找最邻近的5个点，然后会发现蓝色有3，红色有2，所以占主要的是蓝色，所以绿色圆被划分给蓝色）



由此也说明了KNN算法的结果很大程度取决于K的选择。

     在KNN中，通过计算对象间距离来作为各个对象之间的非相似性指标，避免了对象之间的匹配问题，在这里距离一般使用欧氏距离或曼哈顿距离：



同时，KNN通过依据k个对象中占优的类别进行决策，而不是单一的对象类别决策。这两点就是KNN算法的优势。

### 3、KNN算法的思想总结

   接下来对KNN算法的思想总结一下：就是在训练集中数据和标签已知的情况下，输入测试数据，将测试数据的特征与训练集中对应的特征进行相互比较，找到训练集中与之最为相似的前K个数据，则该测试数据对应的类别就是K个数据中出现次数最多的那个分类，其算法的描述为：

1）计算测试数据与各个训练数据之间的距离；

2）按照距离的递增关系进行排序；

3）选取距离最小的K个点；

4）确定前K个点所在类别的出现频率；

5）返回前K个点中出现频率最高的类别作为测试数据的预测分类。

### 4.KNN缺点

KNN确实有缺点，其中包括较高的预测成本，这对于大型数据集而言更糟。 KNN对异常值也很敏感，因为异常值会对最近的点产生影响。 此外，它们不适用于高维数据集，并且分类特征不能很好地工作。 由于模型需要存储所有这些数据点以便确定它们之间的距离，因此KNN算法在处理更多数据时会变慢。

### 5. KNNMatch

匹配过程中很可能发生错误的匹配，错误的匹配主要有两种：

①匹配的特征点是错误的，

②图像上的特征点无法匹配。

常用的删除错误的匹配有： **1.RANSAC方法（随机抽样一致算法）计算基础矩阵，并细化匹配结果**

**2.使用RANSAC方法计算基础矩阵后可以得到一个status向量，用来删除错误的匹配**

3.**计算单应矩阵H，并细化匹配结果**

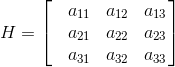
## 五．透视变换

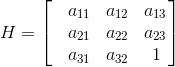
### 【知识点】一、线性变化:

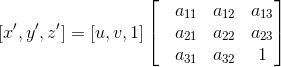
C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\6H7%8L62QI]CBBRO1Q)D7CA.JPG

原点不变，且原有的平行关系和线性关系不变

### 【知识点】二、透视变换透视矩阵的一般表达形式为：



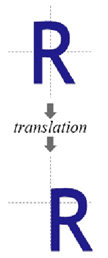
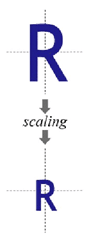
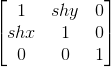
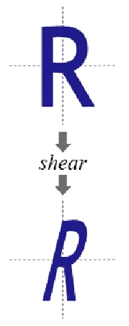
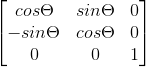
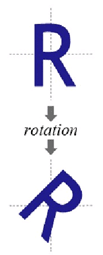
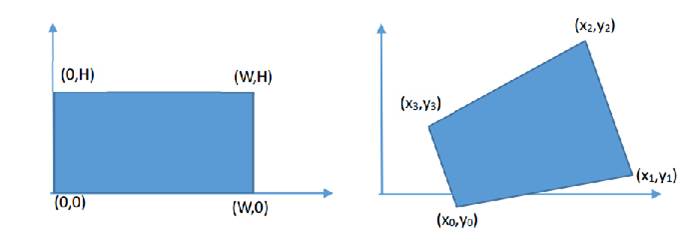
在计算完透视矩阵之后都会把H矩阵的所有元素都除以a33，于是，我们最后得到的H矩阵是这样的：  
——式2.1

  获得了透视矩阵之后，就可以进行图像的变换了：  
——式2.2

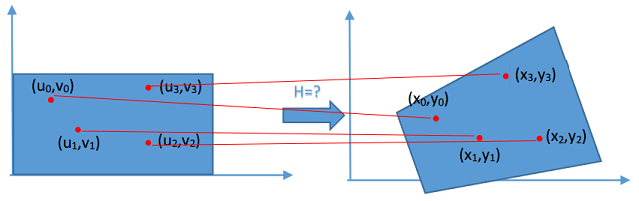
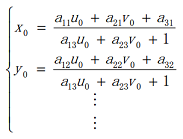
  其中，u,v分别为当前图片的横坐标和纵坐标，变换后的图像坐标是x’/z’和y’/z’，即以下表达式中的x和y：  
C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\K~U)_DZ%AUV91T)NR$[[5[D.JPG  
     
C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\}GJHB2ZTF3S{K(IY3LZ$`_3.JPG --式2.3

  x和y分别为变换后图像的横坐标和纵坐标。透视矩阵中的元素组合有着其特殊功能：a11、a12、a21、a22能控制图像的缩放、修剪以及旋转，a31、a32能控制图像的平移，a13、a23能将图像进行透视变换。

### 【了解】三、几种典型的仿射变换：

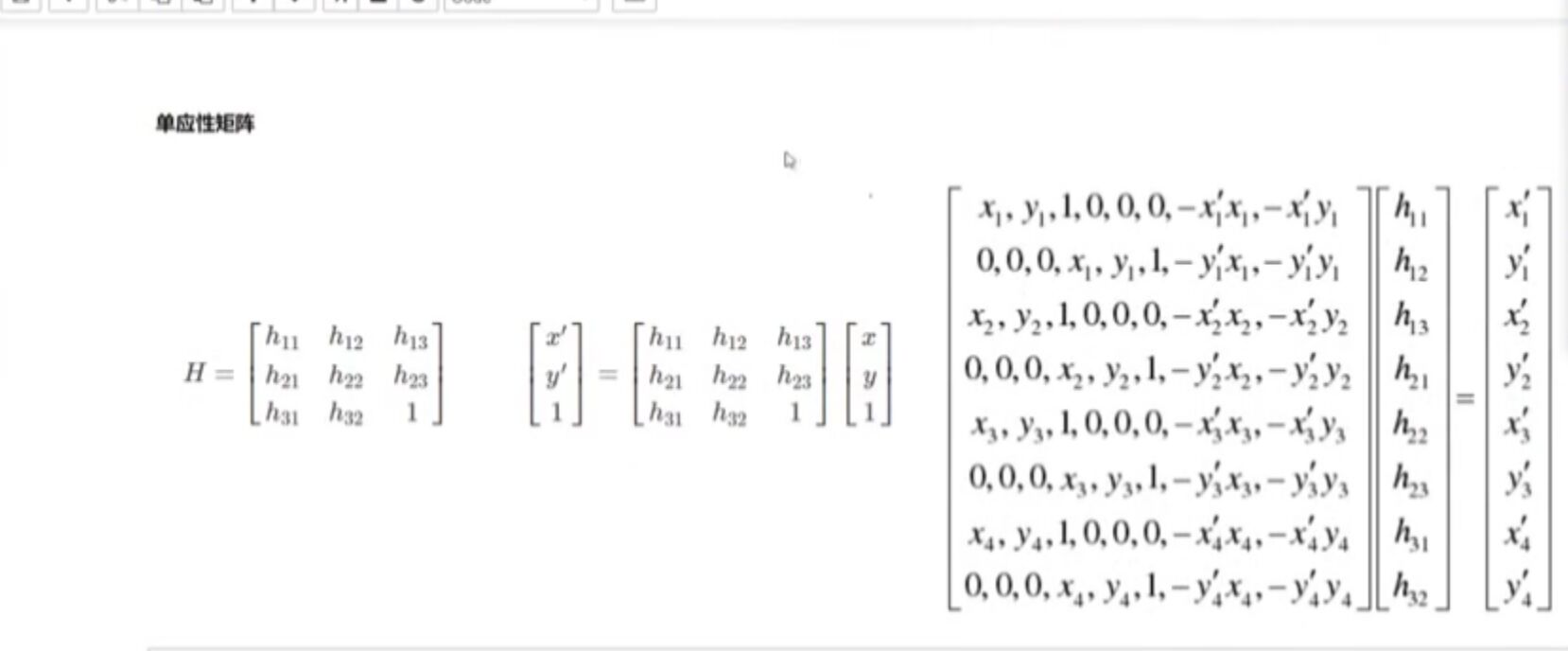
1. 平移变换（Translation）：   将图像的每一像素点平移（x+tx,y+ty），变换矩阵为  
     
   ，变换效果如下：   
     
    2、缩放变换（scale）：将整体图像横坐标放大（缩小）sx倍，纵坐标放大（缩小）sy倍，变换矩阵为  
     
   ，变换效果如下：   
     
    3、剪切变换（shear）：   剪切变换并不是把图像进行裁剪，而是将方形变平行四边形的过程，其变换矩阵如下：   
     
   ——纵向剪切  
     
   ——横向剪切    
     
   ——相当于横向剪切和纵向剪切的组合   变换效果如下：   
     
   4、旋转变换（rotation）   指的是将图像以其中心位置旋转某一角度，其变换矩阵为：  
     
       变换效果如下：  
     
   5.确定变换后图像的尺寸大小需要确定原图像四个顶点经过透视变换计算后的新坐标，即可确定新图像的大小。  
     
      按照这个思路，将原图四个顶点的坐标分别代入2.3式中的u和v变量，可以得到转换过后的顶点坐标：  
   C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\Q`{H1L~~OB1P1P@Q9`~{XN5.JPG  
   C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\IVHDB{JRMZ%874AR9%[@0WA.JPG  
   C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\EHEEYQ@7TE900@RU_`2K2OF.JPG  
   C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\LW47_F{[@LAOQKKO}3]}`[2.JPG  
      由以上四个坐标可以获得变换后的图像尺寸：    
   C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\3AMJ(G@`6Y_WZT`HHC7{(HU.JPG  
   C:\Users\86177\Documents\Tencent Files\2402637869\Image\C2C\}@)MX6}V{5@5A{LAWBG(12G.JPG

### 【知识点】五、根据对应坐标求透视矩阵

有时，我们已经找到了两图中对应的匹配点，需要通过这些匹配点求其相应的透视矩阵，这个问题的典型应用就是图像的无缝拼接，它首先需要分别找到两图中的特征点（sift、surf特征等等），然后在两图中匹配对应的特征点（即图1的特征点在图2中出现的位置），之后通过这些匹配特征点的坐标，计算出透视矩阵。  
  
   要计算透视矩阵，我们还是要回到公式2.3，透视矩阵所求的变量共有8个（a33=1，其余都是要求的对象），根据方程组求解的基本知识，我们需要8个等式才能求出这8个变量，而取4对匹配点代入公式2.3正好可以组成8个等式，剩下的就是方程组求解的问题了。  
  
 如果已经知道图片将要变换后的样子（比如要将图片变形放入某一固定模板），四个顶点正好可以作为四个匹配点代入上式，从而获得透视矩阵

### 【实例解释】单应性矩阵（H矩阵）

通过opencv大概知道：利用两个图像中至少四个特征点能够求解一个单应性矩阵(homography matrix)，然后用这个单应性矩阵H，能够将图像1中的某个坐标( u , v )变换到图像2中对应的位置(u',v')



单应性矩阵不需要太多了解，只需明白在这里要用它进行映射，[x’,y’,1]=H\*[x,y,1],通过之前的RANSAC算法找到最少四组最优的点计算H，通过比较（进行了H矩阵变换后的两幅图相同特征点位置之间距离）与（原本未经变换的两点间的距离），越短则证明变换越成功。

## 六．代码实现效果及解释

from cv2 import cv2

import numpy as np

#其实该语句就是在python环境下对文件，文件夹执行操作的一个模块

import os

def imshow(img):

cv2.imshow('img', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

def imdiv(img, div):

#img.shape[:2]取彩色图片的长、宽

(x, y) = img.shape[:2]

#将float型转为int型

x = int(float(x)/div)

y = int(float(y)/div)

img = cv2.resize(img, (y, x))

return img

#os.listdir() 方法用于返回指定的文件夹包含的文件或文件夹的名字的列表

files = os.listdir()

print(files)

if '2.jpg' not in files:

#os.getcwd返回当前进程的工作目录

path = os.getcwd()

path += '\\project3'

os.chdir(path)

img1 = cv2.imread('2.jpg')

img2 = cv2.imread('1.jpg')

img1 = imdiv(img1, 5)

img2 = imdiv(img2, 5)

#将待拼接的图像显示

show = np.zeros((max(img1.shape[0], img2.shape[0]), img1.shape[1] + img2.shape[1], 3), np.uint8)

show[0:img1.shape[0], 0:img1.shape[1]] = img1

show[0:img2.shape[0], img1.shape[1]:img1.shape[1] + img2.shape[1]] = img2

imshow(imdiv(show, 1.5))

gray1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

gray2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)

#特征提取

sift = cv2.xfeatures2d.SIFT\_create()#蛮力化匹配

kp1, des1 = sift.detectAndCompute(gray1, None)

#关键点及关键点对应特征向量

kp2, des2 = sift.detectAndCompute(gray2, None)

bf = cv2.BFMatcher()#这里的BF就是暴力匹配法的缩写

#knn就是一个点对应k个点，然后选取最佳选项

matches = bf.knnMatch(des1, des2, k=2)

good = []

for m, n in matches:

#是一种过滤的方法，通过比较H和实际两图连接的距离，肯定是H越小越好就证明不同角度上相同位置的特征点就基本对应起来了，在计算选择特征点的时候选最优的特征点

if m.distance < 0.75 \* n.distance:

good.append(m)

good = sorted(good, key=lambda x: x.distance)

#排序，通过距离度量排序

#找到对应关键点并连起来，drawMatches找到对应关键点用来连线

show = cv2.drawMatches(img1, kp1, img2, kp2, good[:20], None, flags=2)

imshow(imdiv(show, 1.5))

#到这里特征匹配完成

#记录特征点的x，y坐标

pos1 = np.float32([kp.pt for kp in kp1])

pos2 = np.float32([kp.pt for kp in kp2])

#k对的最佳匹配

#特征值匹配

bf = cv2.BFMatcher()

matches = bf.knnMatch(des1, des2, k=2)#这个k就是将之前的1对1匹配变为了k对匹配

good = []

for m, n in matches:

#通过knn算法筛选优良匹配

if m.distance < 0.75 \* n.distance:

#记录优良匹配的两个匹配点的index

good.append((m.trainIdx, m.queryIdx))

#记录匹配点对中左图点的x, y坐标放在pts1中

pts1 = np.float32([pos1[i] for (\_, i) in good])

#记录匹配点对中右图点的x, y坐标放在pts2中

pts2 = np.float32([pos2[i] for (i, \_) in good])

示例1运行结果:

第一幅图：将要拼接在一起的两幅图先简单拼在一起



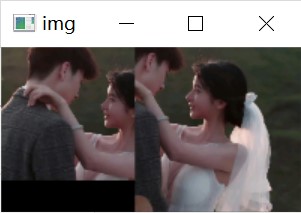
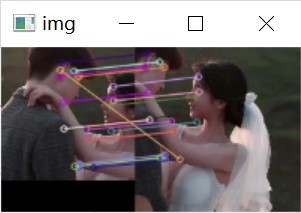
第二幅图：实现特征点匹配以及算法处理后关键点之间的连接

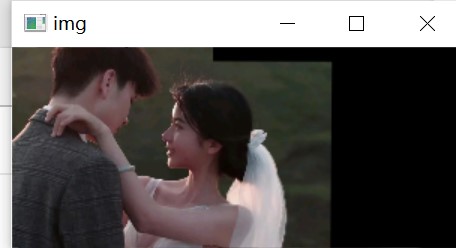


第三幅图：成功实现两幅图之间的全景拼接



示例二：



图片拼接成功率：100%

第三章 图像拼接技术的不足

图像拼接最终结果的好坏主要取决于算法稳定性。但就目前情况而言，多数拼接算法都会产生图形轻度畸变，拼接图片因光线问题处理不当而无法实现完美融合等等问题。针对不同的拼接图像，其采用的拼接算法并不相同。因此，到目前为止，并没有一种通用性的图像拼接评价体系是通用的

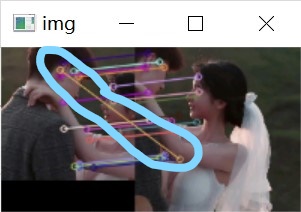
第四章 基于本图像拼接技术的优化

单应性矩阵的精细化算法：LM算法

第五章 研究展望

图片拼接技术涉及知识范围广，本文虽实现了两幅图像处于不同或相同角度时的全景拼接，但仍有多个问题待研究与完善：

1. 从本篇文章的处理结果可以观察出对于不同光线下的图片拼接时会出现因光线而产生的色差问题。
2. 本文章仅实现了两幅图片的拼接，对于多幅图片的拼接仍需研究。
3. 在本篇文章中出现了特征点匹配错误的问题，例如示例二的运行结果图二。特征点匹配出错会对之后的矩阵计算产生影响，进而对拼接结果产生影响。可否利用更好、更稳定的算法提高拼接后图片的质量也是继续需要研究的问题。



# 致谢

通过这次的学习，我学会了许多关于图像拼接的知识点。首先需要感谢的是陈东祥老师对我们的悉心教导，不仅让我们学会了关于数字图像处理方面的知识点，同时也教会了我们关于实验报告规范方面的知识，这位我们大四的毕业论文书写打下了扎实的基础。其次，还需要感谢这一组的同学们，是大家的不懈努力，共同合作，才有了我们这次的课题成果。最后还需要感谢的是学校环境的大力支持，没有课程的设立，我是不会有这次课题的研究的。