谭平 Vision Course

# 1 camera

## 1.1 小孔成像模型(pinhole camera)

Q1：场景中有一个物体和一个感光元件（胶片、CMOS），两者之间不加任何东西，那么感光元件上会记录下什么东西呢？

A1：物体地表面会向各个方向反射光线，也就是说物体每一个点都会投影向整个感光元件，那么最后在感光元件上成地像会是一个均匀的平面（比如全灰，全白）。

Q2：怎么解决呢？

A2：在物体与感光原件之间增加一个小孔模型。对于物体上地每一个点，其反射的光只有一束能透过小孔，在感光元件上成像，那么就建立起了物点和像点一一对应的关系。

Q3：孔的大小对于成像有什么影响？

A3：孔太大，会导致物点在感光元件上多处成像，导致成像模糊。

孔太小，会导致物点反射的光线透过小孔的太少，同样的时间也会导致成像模糊，但是通过增长曝光时间可以得到一定的缓解。如果孔真的太小，甚至会出现干涉现象。

### 凸透镜

Q4：有没有更好的解决方法呢？

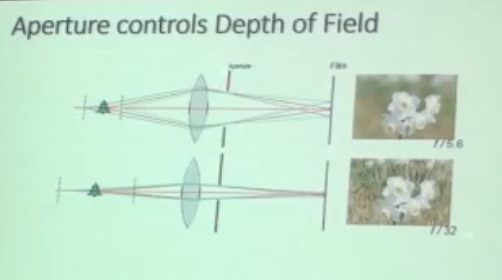
A4：通过凸透镜来取代小孔。凸透镜会聚集所有到达镜面的光，然后汇集成一点。但是距离凸透镜不同距离的物体在另一边成的最清晰的像的位置也是不一样的，所以凸透镜也是由一定问题的。

### 光圈 & depth of field

Q5：如何解决凸透镜的问题呢？

A5：在凸透镜前加光圈（aprture），光圈开的大，光线接受的范围大，那么距离较近的物体会成像清晰，远的物体成像模糊，可以起到强调眼前对象的作用。

光圈开的小，接受的光线范围小，凸透镜后的光线会较为收缩，所以近、远处的物体成像都能清晰。但是由于通过的光线变少了，所以曝光时间需要增长。



## 焦距（focal length）

焦点指的是凸透镜光线汇集成的点，焦距指的是凸透镜光心离成像平面点的距离。

## 视场（field of view）

<https://blog.csdn.net/sw3300255/article/details/83998164>

由上面也可以知道，depth of field越长，焦距也需要越长，能看到的深度虽然增加了，但是能看到的范围却变小了。FOV指的就是相机能看到的场景的范围的大小。

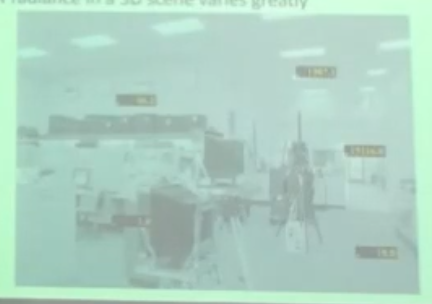
视场与焦距和感光元件的大小有关。计算如链接。

# 2 color

# 3 Radiance/Irradiance

Irradiance：单位时间单位面积所接收到的能量

Radiance：单位时间，单位面积，单位空间角所接收到的能量



相机处于图中位置。假设桌底对相机的radiance为1.，那么墙壁对相机的radiance可能为几十，灯源可能为2000，也就是说不同的物体对相机的radiance是不同的。

Radiance与三个因素有关：光源，物体的表面情况（光滑/粗糙），物体的朝向。

## 3.1 从radiance到pixel values

曝光量Exposure = radiance \* exposure\_time

Pixel value = f (Exposure)

从曝光度到像素值，经过了很多操作，存在很复杂的函数映射关系，下面列举了集中常见的操作。

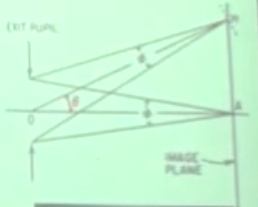
### （1） 自动白平衡（auto white balance）

场景：不同的光源条件下，拍摄相同的物体得出来的照片是不一样的。比如一张白纸，在日光灯下拍出来会偏蓝，在白炽灯下拍出来会发橘（这两种灯的光源里都相对缺少了一种波长）。

不同的光源会对拍摄出来的照片产生不同的影响，我们往往想消除这种影响，这就是auto white balance的目的。

### （2）vignet

相同物体散发出的radiance，在感光元件不同的位置上的其强度是不同（如下图，中间比较集中，上边因为距离远了所以有些涣散），最后导致排除来的照片的强度是不均匀的。

### （2）噪音

会有各种去噪音的方法

### （4） 总结

最后得到的结论是从物体散发出的radiance到srnsor再到形成pixel\_value是一个很复杂的非线性过程，所以照片上的灰度值并不是radiance。

早期的胶片相机中，输入的radiance与灰度值的关系可以通过一条曲线来描述；但是如今的相机非常复杂，甚至对不同场景设置了不同的模式，有不同的算法，所以对于一个现代相机来说，可能已经无法找到这样一条曲线了**，但是对于某一张图片，某一类场景，可能还是有这么一条曲线的**。

## 3.4 Radiometric calibration

既然这条曲线存在，那么如何标定这条曲线成了一个重要问题，Radiometric calibration就是用来拟合这条曲线的。有了这条曲线，我们就可以从pixel values推的场景的radiance，radiance是具有物理意义的，仅为我们可以对场景进行分析，比如应用有shape from shading ，photometric stereo等。

## 3.5 HDR

问题：在一个场景中，可能有些区域的明暗差异非常大，所以在一张照片，单一曝光度下，难以将整个场景的所有位置都拍清楚（曝光时间短，明亮的地方拍清楚了，暗处没拍清楚；曝光时间长，暗处拍清楚了，明亮的地方又曝光过度了）。

解决方法：HDR。对某些场景拍摄多张不同曝光度的照片，目的是为了再这些照片中能把所有场景所有部分都拍清晰，然后通过radiometric calibration拟合出的曲线，得出照片各个位置的radiance，然后对这些radiance进行操作，最后得到一张场景所有地方都拍清楚的图片。



# 4 filter

滤波的操作和CNN中的卷积的操作类似（或者可以称为相同）。

## 4.1 Gaussian filter & box filter

两种滤波器如下图所示，区别在于给周围像素的权重不同。不一样权重导致的区别就是Gaussian会更加突出当前像素的灰度值，为降低周围像素灰度值的影响。而box会平衡地考虑当前像素与周围像素地灰度值。

为了确保亮度相同，所以前面乘了一个所有权重之和的倒数。



## 4.2 Edge detection

边缘（edge）的特征是边缘的一边与另一边的像素的灰度值相差比较大。

在经典的数学理论中，函数值变化较大处，其梯度变化也比较大，可以通过如下公式求梯度：



对于实际的图片，可以设置Δ为具体的像素数量，如果设为1的话，图像在某一个像素处的倒数就可以表示为前后两个像素的灰度值之差（因为2Δ是共有的，所以可以去除）。

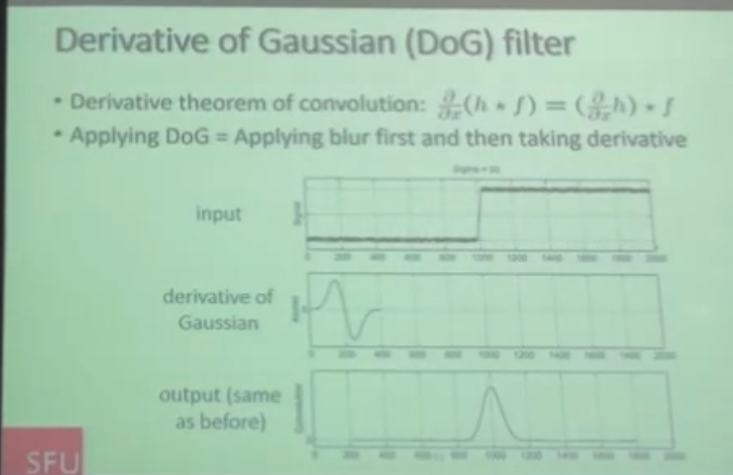
进而可以通过一个一维的filter来表达上诉思想。可以使用[1][0][-1]来检测x方向上的梯度变化，进而求出y方向上的edge。也可以使用竖着的[1][0][-1]来检测y方向上的梯度变化，进而求出x方向上的edge。

**DoG：**

但是对于一张实际的图片，其像素的分布置可能会受到噪声的影响，直接进行求导的话，会导致在很多相邻位置的梯度都很大，显然这是不科学的，所以不能直接对其求导。所以需要先通过gaussian filter先对图片进行滤波，消除噪声。然后进行求导。

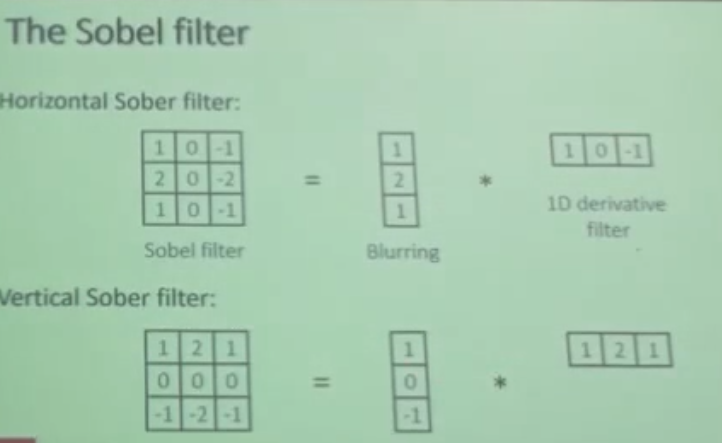
上述相当于先对图片进行了一个Gaussian filter，然后再进行一个derivative filter（也就是[1][0][-1]）。

是要把上述操作顺序改一下。具体的忘了～～～



DOG，先对像素应用Gaussian filter，以消除噪声的影响，然后再进行求导。

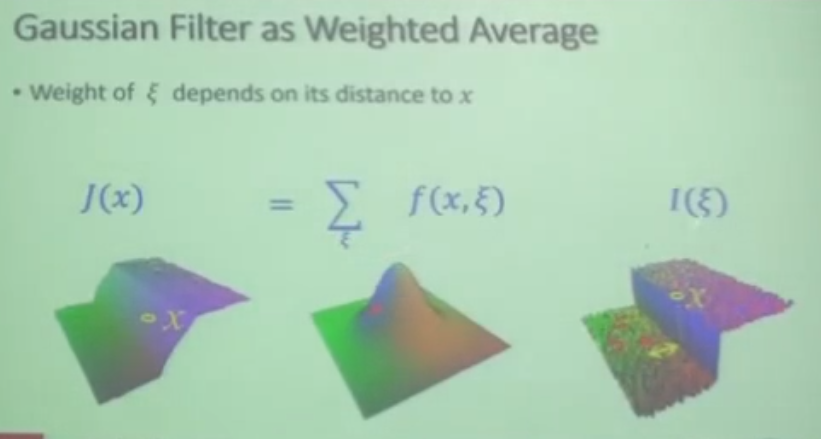
## The Sobel filter



Sobel其实就是gaussian filter和derivitive filter的一个组合。（将两步合成一步）

Bilateral filter（双边滤波）

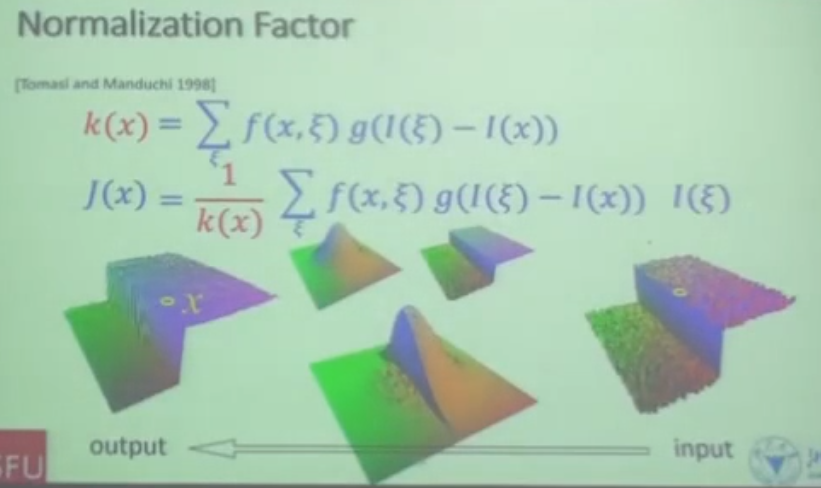
假设有一张一边白一边黑的带有噪声图片，首先要对其应用gaussian filter进行降噪。但是gaussian的降噪方法是将当前像素附近的像素的灰度值进行加权平均取代当前像素的灰度值（也就是仅考虑距离spatial的影响），偏左右的像素加权平均后的结果没问题，但是中间接近edge处的像素因为平均了黑白两种灰度值，会导致出现灰白之间的过渡色，会使得原本陡峭的变化缓和下来，这是我们不希望的。



Bilateral filter的想法是再加上一个权重g(~)来考虑当前像素周围像素灰度值的影响。周围像素灰度值之差越大，权重值越小，反之越大。

如此，就得到了一个既考虑空间位置（spatial）又考虑灰度值关系的filter，示意图如下图所示。在从右边接近边界的像素x处，由上边的两个filter相乘，得到了一个新的filter，这个filter关注于右边的像素。

所以bilateral filter做到的就是，一方面要smooth noise，一方面还要preserve edge details。



# 5 Feature detector

## 5.1 Local features

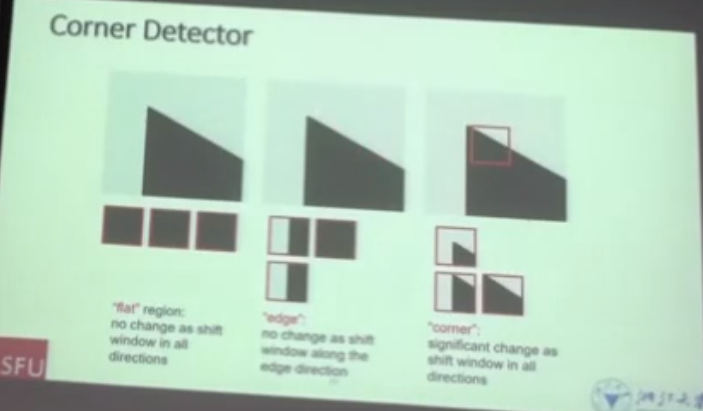
Advantages of local features

1. 抗遮挡与抗混乱性
2. 单张图片中的local features很多
3. 相比于全局特征，具有更高的实时性。

## 5.2 What`s is a good feature？

一个好的feature应该对大部分变化都有一个好的鲁棒性。比如scale-invariant，rotation-invariant，geometry-invariant，photometric-invariant等等。

## 5.3 harris corner detector



**用一个local window框取图片中的一部分，如何确定这个local window框取了一个corner？**

**Intuitively：**

如第三张图片所示，让这个window往上下左右移动，window内的图案都会发生变化，则称之为一个Corner。

比如第一第二张图片中，window分别沿x，y方向移动，window中的pattern都没发生变化，则不能称原始框中的图案为一个corner。

**mathmaticlly**

Ssd：对应元素相减平方再求和

定义一个能量函数E，他表示将两个local window中对应元素做ssd，判断差距的大小。如果我们希望找到Corner，那么移动lw后的能量函数要足够大。

想要能量函数足够大，那么就需要找到最小的特征值。

不过还有一种更方便的计算方法，用到再查。

## 5.4 Invariance of harris

Harris corner detector具有rotation invariance，如果图片所有像素的灰度值都加上一个常量，对hcd的影响也不大。但是如果改变scale，则很有可能会使hcd失效。

那么如何达到scale-invariance？采用不同大小的local window进行实验，选取一个适合的大小的local window。

## 5.5 FAST detector

用决策树算法确定Corner，速度比较快。

# 6 descriptor

## 6.1 SIFT（scale invariance feature transform）

SIFT是一个descriptor，对于每个local window，将其划分为4\*4的网格，对于每个网格内的内容，用一个8维的向量对其进行描述，所以对于每一个local window，使用了128维的向量来描述。

投票！！！

## 6.2 SURF（Speed up robust features）

他的速度比SIFT快，可以认为是SIFT的加速版。

## 6.3 BRIEF（binary robust independent element features）

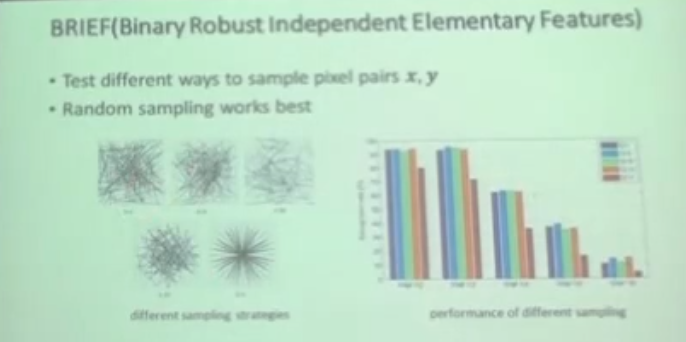
主要思想是建立一个二进制（binary）的descriptor，这样能计算得更快。

具体的做法是随机在local region中选取一对像素点x，y，如果x>y则取值为1，否则为0。比如可以取256对这样的x，y，那么就得到了一个256维的binary的descriptor。

那么如何取这些x，y呢？

考虑的方案有x，y都取均匀分布；取独立的高斯分布；给定两个点一个初始值，然后每个点都在前一个点的周围取；固定一个点，另一个点再随机取。

最后实验证明是第二第三种的效果比较好。



## 6.4 ORB（Oriented FAST and Rotated BRIEF）

a combination of FAST and BRIEF

## 6.5 GIST

GIST的来源在于以前的一些心里学实验的结果：人类能在很短的时间内生成对一张图片、一个场景的描述，从而在后面识别出这张图片。

在这么短的一个时间内，是不太可能对图片的细节产生描述的，所以研究人员就考虑设计一种全局性的描述子，对整张图片进行描述。当然这种描述子没法使用在correspondence中，但是可以使用在image search这类需要考虑图片全局意义的情况下。

## 6.6 How to choose a descriptor？

一般来说速度与精度不可兼得，速度越快，精度会差。

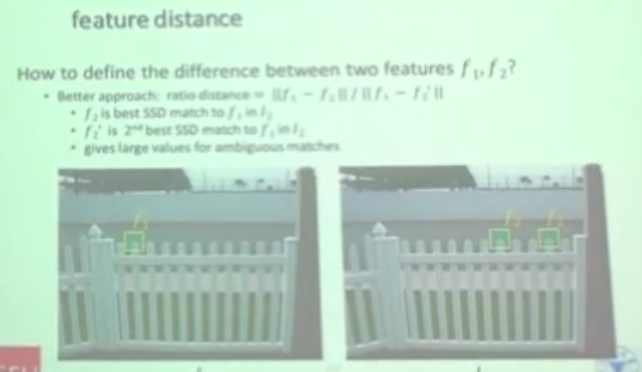
然后根据需要处理的任务，对时效性与精度的要求，如果更看重精度，而不太在乎实时性，选择SIFT不错。如果在SLAM中，那该是选择ORB等速度快的好。

# 7 Feature matching

## 7.1 feature distance

对P1，P2两张图片进行matching，最简单的想法就是固定P1中的一个feature f1，然后遍历P2中的每一个feature，计算L2 distance，distance越小，就说明这两个特征就越匹配。

但是实际情况可能更复杂，有时P2中有两个feature与f1都比较接近，此时就需要计算f1与P2中最近的feature的距离以及与第二近的feature的距离的比值，如果这个值小于threshold，则我们认为这个是一个有效的匹配。

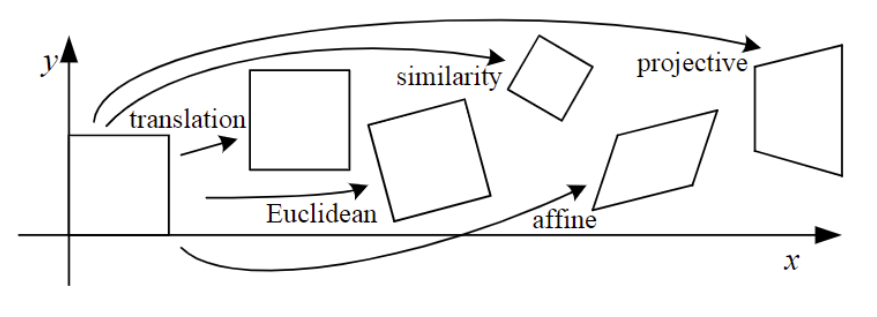


## 7.2 how to evaluate the result?

召回率与准确率都要高。

# 8 Homography

## 8.1 平面集合变换



Planar linear motion Models

从左到右依次为：平移，旋转，相似变换，仿射变换，透视变换

因为2\*2的矩阵无法表示位移，所以引入了其次坐标的概念。

Affine transformation对3\*3变换矩阵的约束是，最后一行要为001.

projective Transformation aka planar perspective maps aka Homography，它能表示out of plane的变换。它对3\*3矩阵的约束是最后一个元素为1.

<https://www.cnblogs.com/shine-lee/p/10950963.html>

<https://www.cnblogs.com/happystudyeveryday/p/10547316.html>

<https://www.zhihu.com/question/20666664>

## 8.2 如何求解Homography matrix？

Define up to a scale ：基于比例来定义的

变换矩阵的最后一个元素为1，所以变换矩阵最多有8个独立变量。P1，P2两张图片中，每对点最多给予2个独立方程，那么需要4对点来求解H。

### DLT（Direct linear transformation）

在Homography问题中，DLT用来求解H matrix，也即

转化为线性方程：

# 9 RANSAC

## 9.1 The function of RANSAC

剔除误匹配点的数量。

## 9.2 The idea of RANSAC

RANSAC的内在思想：inlier都是一致、紧密的，outlier都是相对随机、不一致的。

## 9.3 the implement of RANSAC

Choose a small set of sample randomly,the set should be as small as possible, so that it`s more likely outline-free。

为什么要用最少的点来拟合呢？

因为用最少的点，我们相对就有更高的概率得到所有样本点都是inlier的模型。点数越多，越有可能取到outlier。

例子：

比如仅用两个点来拟合一条直线，有可能取到的一对点都是inlier，也有可能取到的点都是outlier，如果我们有办法判断出这些dot pair中那些是inlier的，那些是outlier的，那么我们就能得到一个正确的拟合结果，进而得到一个正确的模型。

那么如何判断inlier和outlier呢？

对于所有的样本点，可能拟合出多条直线，也就是多个模型。对于一个模型，我们可以定义距离这个模型多少距离的点为inlier，其他就为outlier，这些模型inlier的数量就视为这些模型的评分。最后根据各个模型的评分高低，选取评分最高的模型。

但是选取的这个模型并不是最后的模型。我们根据这个模型的inlier，用M-estimator（不一定是这个词）再做最后的拟合。

### 9.4 RANSAC流程范式：

（1）随机挑选s个样本点（能够拟合模型的最少数量）

（2）拟合一个模型

（3）计算与模型的距离小于阈值的inlier点数量

（4）重复上述步骤N次

（5）choose the model of largest inlier，then fit hre model again

注：在数学上，如果要确保你的模型是真正正确的，就必须保证你的inlier大于50%。这也就是说你的噪声不能太多。

## 9.5 RANSAC的可调参数

（1）inlier shreshlod（距离模型多大距离的点算作inlier）

（2）number of rounds（迭代次数）

## 9.6 迭代次数N如何计算呢？

我们做出如下假设：

1 假设每次挑选s个样本来拟合一个模型

2 假设e是outlier的比例

3 假设迭代了N次

4 假设在这N次中有一个模型是outlier-free的（s个点都是inlier的），就视为成功

从统计学上来说，如果真的要保证一定成功，那N可能是一个我们无法接受的数字，所以一般是通过置信度来度量这个N，比如我有99%的置信度确定运行N次，必有一次能成功。确定置信度计算N的公式如下：

s个元素的set是outlier-free的概率q = （1-e）s

s个元素的set是inlier-free的概率（1-p）=（1-q）s

取置信度为p（比如0.99），则N的计算公式为：

（没太理解）

## 9.7 拟合模型的最少样本数量s怎么确定？

根据实际情况，比如拟合一条直线需要两个点，拟合一个平面参数需要三个点

## 9.8 outlier的比例e如何确定

估计的

## 9.9 RANSAC pros and cons

# 10 一些基础概念

## 10.1 Algebraic Distance代数误差

DLT minimizes ||Ah|| #direct linear transformations

DLT的做法是构造矩阵Ah，使得Ah等于0.则e=Ah就称为resdual error，设ei为e

代数误差不好，我们顶多用它来做一个初始化。

## 10.2 Geometric Distance几何误差

### 单图像几何误差

前提假设：一张图片中的坐标完全正确，另一张图片中的坐标有误差

假设x和x`分别是两张对应图片p1和p2中的点，并且已知两张图片之间的Homography，将x映射到p2中为Hx。欧式距离指的就是Hx与x`之间的欧氏距离。

总结来说，欧氏距离是指一张图片中的点通过H映射到另一张图片中，与对应点之间的欧氏距离。

### 对称几何误差（转移误差）

前提假设：两张图片中的坐标都有噪音。

求p1中的x通过H映射到p2中，与x`的欧氏距离，还要求x`通过H-1映射到p1中，与x的欧式距离。每一对对应点都要考虑对称的两个误差，所以称为对称几何误差。

## 10.3 Reprojection error重投影误差

### 重投影

根据图片上对应像素点的位置以及相机矩阵，triangulate出来三维点的空间坐标。然后根据三维点的空间坐标，相机矩阵，计算三维点在图片上的投影位置，这就是重投影。（第一次投影是指对真实的三维点进行拍摄得到图片，第二次投影是指对重建的三维点通过相机矩阵计算投影点）

### 重投影误差

<https://www.cnblogs.com/Jessica-jie/p/7242179.html>

几何误差的局限性：几何误差仅仅考虑了单应矩阵H的误差，但是往往特征点的提取也是存在噪声的（Harris Corner detector提取角点时本身就存在噪声）。

重投影误差不仅考虑了单应矩阵的误差，而且还考虑了重投影点与检测点之间的误差。重投影误差优化的是H和x，所以精度更高。

### 有标定物的情况

但是如果使用精度很高的标定物时，因为标定物上的点是很准确的，可以认为其没有误差。此时如果使用重投影误差反而会引入错误信息。此时应该使用单边转移误差（单边几何误差）。

# 11 射影几何（projective geometry）

<https://zhuanlan.zhihu.com/p/92504952>

平面直线方程ax+by+c = 0,真正决定这条直线的是系数（a,b,c），而且这些系数乘以任意的非零常数k，表示的都是同一条直线。

用其次坐标表示一个点x = （x1, x2,w）,虽然有三个参数，但是只有因为其次坐标是define up to a scale，所以其实只有两个独立参数。

则有：

如何确定点x在线l上？只要xT · (a,b,c) = 0，就说明。

如何求两线的交点坐标？将两线的向量做叉乘。不过求出来的坐标是齐次坐标。

如何通过两点求出直线方程（向量）？将两点的向量做叉乘

射影几何与平面几何最大的区别就是引入了无穷远点的概念。

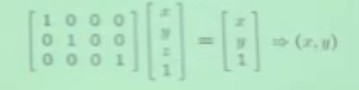
射影几何认为平行线相交于无穷远点。

在射影几何中，一切变换都是define up to a scale的。比如说三维空间点k（x,y,z,w）,不论非零常数k取多少，其经过相机矩阵投影到二维图像上的点是相同的，因为乘以k是将三维点沿着同一条直线变换位置。

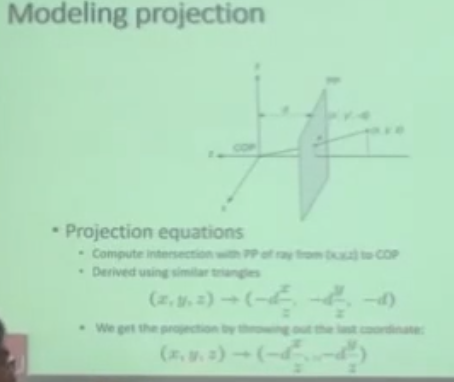
## 11.2 正交投影（orthographic projection）

正交投影是透视投影（homography）的特例，其投影中心在无穷远处（所以也被称为平行投影），体现在坐标变换上就是直接将z值去掉，(x,y)->(x,y,z)。

投影矩阵如下所示：



# 12 相机模型



相机模型中，为了观察方便，将成像平面关于原点对称到光心前面。没有改变比例，只是多了一个负号。

## 12.1 二维与三维点的投影等式

建立世界坐标系如图所示，x伸出屏幕，y向上，z向左，这样定义的好处是构成了一个右手坐标系。

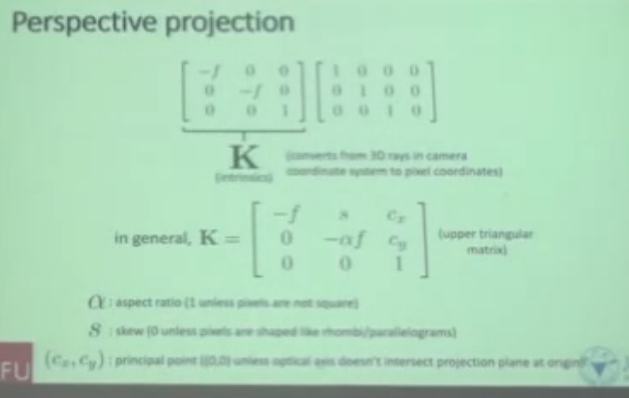
假设三维空间中一点的坐标为（x,y,z）,它投影在成像平面上的点的坐标为（x`， y`），则有：

（x`, y`,z`） = (， ， -d)

## 12.2 相机外参（Extrinsics）

相机外参矩阵的作用是使得世界坐标系和相机坐标系重合，平移矩阵T使得两个坐标系原点重合，旋转矩阵R使得两个坐标系轴对应重合。

## 12.3 相机内参（intrinsics）



K表示的就是相机内参，K右边的矩阵表示的是三维到维的投影。

F：焦距，focal length

α：用来考虑像素长宽比不一样的情况。不过在数码相机时代α基本上都为1了。

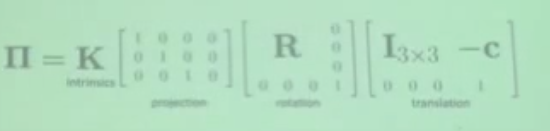
S：用来考虑传感器上的像素两边不垂直的情况，比如不是正方形而是平行四边形。不过在数码相机时代两边基本都垂直，s等于0。

(cx,cy)：用来考虑拍摄相片时相机的光心与相机坐标系z轴不重合的情况。(cx,cy)用来描述两轴的相对偏移量。一般情况下(cx,cy)都等于0，当图片经过裁剪时，他们可能不等于0。

代码中(cx,cy)经常为图片高宽的一半，这应该是在像素坐标系下。如果是在图片坐标系下，那应该就是0了。

## 12.4 相机矩阵（camera matrix）

相机矩阵就是将上述的所有矩阵相乘：



相机矩阵的作用是将世界坐标系下的齐次坐标点投影到图像坐标系中去，右边输入一个4\*1的世界坐标系下的三维空间点，左边输出一个3\*1的图像坐标系的齐次坐标点。

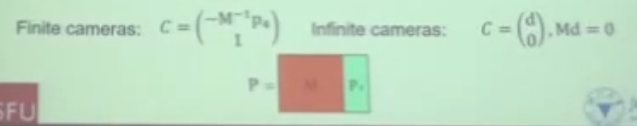
左乘T，将世界坐标系与相机坐标系原点重合；左乘R，使两个坐标系对应的坐标轴重合；左乘K\*3\*4矩阵，将三维坐标投影成二维坐标；

## 12.4 畸变（distortion）

～

## 12.5 从camera matrix分解出相机的中心及朝向

设相机投影矩阵为P，P是一个3\*4的矩阵，将其拆分成一个3\*3和3\*1的矩阵M和p4，相机中心求法如下：



M可逆，则用左边的方法求相机中心；M不可逆则用右边的方法求。

## 12.6 相机矩阵的列向量的含义

将相机矩阵拆分成4个3\*1的列向量：P=（p1, p2, p3, p4）

考虑一个特殊空间点Y=（0 1 0 0）T，这是Y方向上的无穷远点。

PY = p2，那么就说明Y方向上的无穷远点在图像上的投影就是p2.

所以，前三个列向量表示的是X，Y，Z方向上的无穷远点在图像上的投影，第四个列向量表示世界坐标系原点的位置（[0 0 0 1]表示的就是世界坐标系的原点）。

## 12.7 相机矩阵行向量的含义

第三个行向量表示的是主平面，这个平面上的点会被映射到无穷远处。

第三个行向量前3个元素表示的就是相机的朝向。？？？

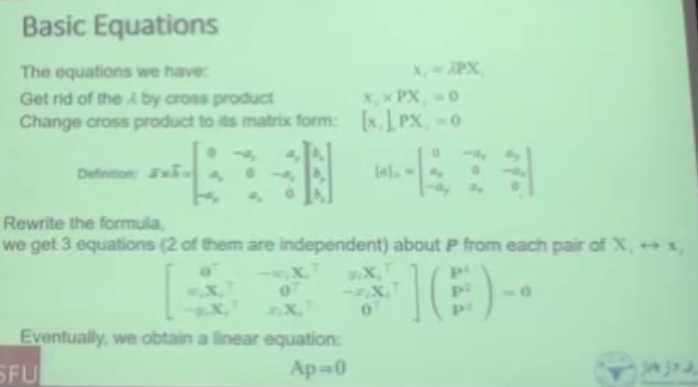
# 13 相机标定

## 13.1 resectioning

假设我们知道空间中一些三维点的具体坐标，并且知道这些三维点与图片中的二维点的对应关系。直觉上讲，知道了上述信息，我们就可以通过摆动相机的位置及朝向使相机光心与二维图片上点的连线，正好通过三维空间中的点。

通过这种方法求相机矩阵的方法，称为resetioning。这种方法需要严格保证三维点的精确坐标以及对应关系。

算法实现：



（将[xi]PXi展开，将已知量和未知量分成两个矩阵，就得到了最后的结果。）

最后根据二维与三维点的对应关系，通过DLT，应用SVD来求解。

需要5个半点来求P，也就是需要6个点。

## 13.2 PnP（perspective-n-points）

PnP用于在相机内参已知的情况下，求解相机外参的一系列算法。这类方法通常先将二维图像坐标转换为相机坐标系下的三维坐标，然后求解相机坐标系与世界坐标系之间的转换关系。

每一对对应点可以给予两个独立约束，平移矩阵T中有3个未知数，旋转矩阵R比较复杂，但是其中也只有3个独立变量，理论上只要有3对对应点就可以求解出外参矩阵。

**P3P algorithm**：通过3个点求出外参矩阵。

但是在p3p的计算过程中会引入一些虚的结果，最后导致会有4个解，那么就需要通过增加对应点来获得唯一解。

**Linear PnP Algorithm**： 要求对应点大于等于5对点。～～～但是这两个算法的计算复杂度都有点高。

**EPnP algorithm：**也是先将二位图像坐标转换为相机坐标系下的三维坐标，然后求解相机坐标系与世界坐标系之间的转换关系。

其计算复杂度为o(n)，EPnP现在使用得比较多。

## 13.3 Circular points

Circular points 指的是二维平面任何圆都要经过的两个点，他们是任何圆与无穷远直线的交点。

假设原点与圆心重合，二维平面上的圆的约束是x1，x2的系数相同，没有x1，x2的交叉项，其方程如下：

求解得：

X12+x22=0

则：

注：拟合一个普通的平面二次曲线需要5个点；

拟合一个二维的圆需要三个点，因为圆默认通过两个circular points。

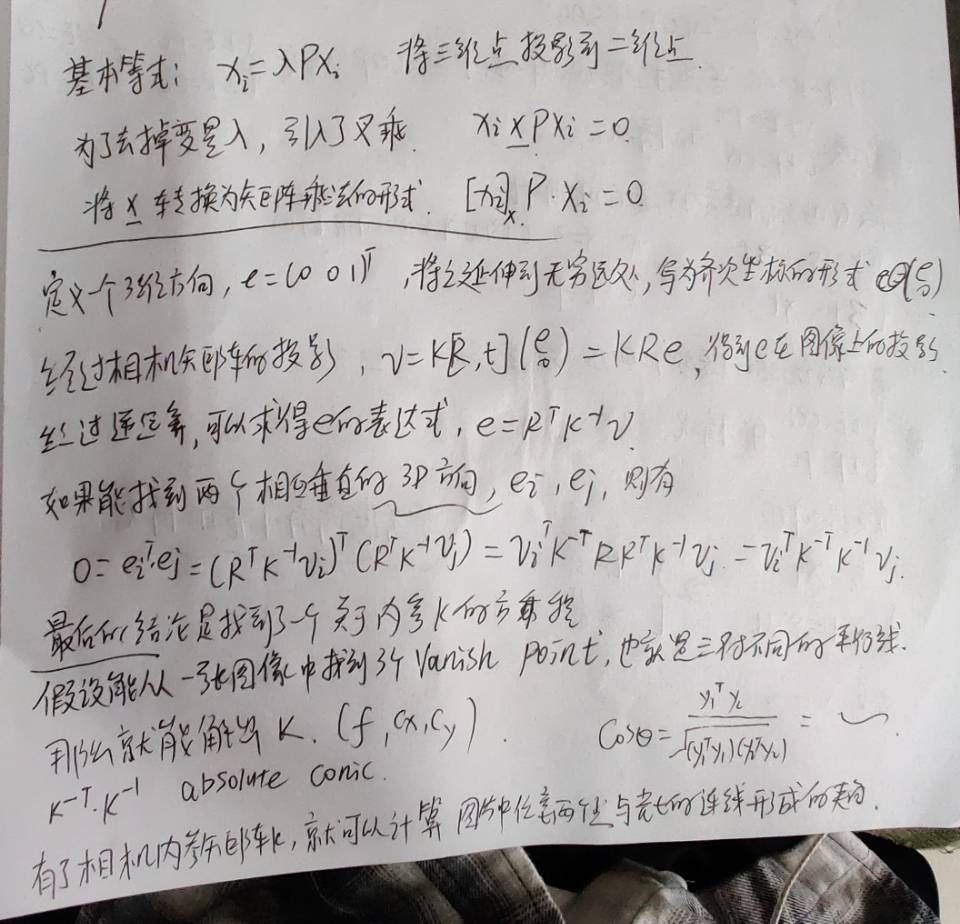
## 13.4 image of absolute conic(IAC)

引出 image of absolute conic

基本等式：xi=λPXi #将三维点投影至二维点。因为射影几何中一切都是define up to scale的，所以要加上系数λ。

为了去掉变量λ，引入了叉乘，等式转换为 xi × PXi = 0

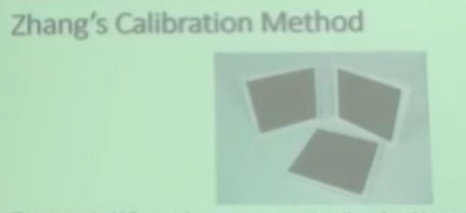
叉乘可以转化为矩阵乘法的形式：[xi] PXi = 0



假设三维空间是由许多二维平面堆叠而成；每个平面上都有两个circular points；所有平面上的circular points构成了一个椭圆曲线，称为absolute conic，记做omega\_i。三维空间中的任何球体都要经过absolute conic。

Circular points是无穷远直线上的两点，absolute conic是无穷远平面上的曲线。

## 13.4 张正友相机标定法



摆三张相同三张标定板并拍照（或者是一张标定板拍三张照片），

（1）首先通过每个标定板（checkboard）与图像的对应关系求出Homography matrix.在标定板上建立一个坐标系，在图片上建立一个坐标系，就可以得到两个坐标系之间的映射关系，也就是Homography。

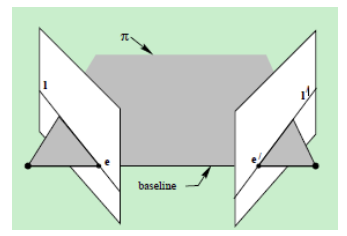
（2）通过Homography求出每个平面上的circular points在图像上的投影点。

（3）三块标定板能得到图片上的6个circular points的投影点。通过这6个点，就可以拟合absolute conic——Omega\_i。根据Omega\_i=K-TK-1，进而可以求出K。

# 14 对极几何（2-view geometry）

## 14.1 对极几何中的名词

<https://blog.csdn.net/tina_ttl/article/details/52749542>



极线（～）:空间点与相机中心的连线

基线（baseline）：两相机中心的连线

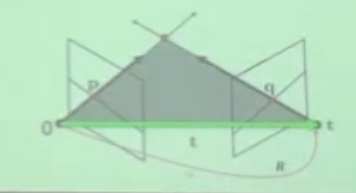
对极平面束（epipolar pencil）：以基线为轴的平面束

对极平面（epipolar plane）：对极平面束中的任何平面都是对极平面

对极线（epipolar line）：对极平面与图像平面的交线。图像1上的点p的对极线是图片2与对极平面的交线，并且对应点p`点一定再对极线上。

对极点（epipole）：摄像机基线与每副图像的交点，比如e和e`，e表示的是相机2在图片1上的投影，e`表示的是相机1在图片2上的投影

## 14.2 对极几何中的约束



假设相机内参已知，那我们就能知道相机坐标系下的3D方向（比如相机1框架下的向量p和相机二坐标系下的向量q）

如果外参已知（R，t如图中的红线与绿线），那么通过RTq，我们就能知道向量q在相机1坐标系下的方向。

最后我们能得到一个约束：p，t，RTq这三个向量要共面。

### 用代数的方法描述约束

可以通过混合基（Mixed product）的方式来描述这种约束。

a·(b×c)的物理意义是以a，b，c为边长的平行六面体的体积。如果体积为0的话那就说明三个向量共面：

p·（t×RTq）= 0

--> pT[t]xRTq = 0

（总体转置）--> qTR[t]xp=qTEp=0

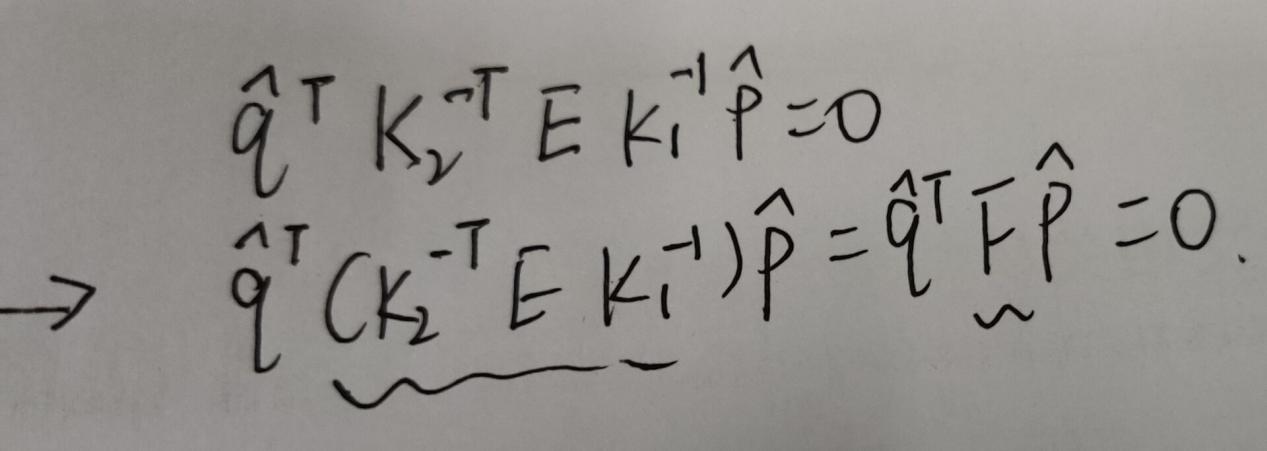
上面R[t]x组成的矩阵E就成为基本矩阵（essential matrix）。

也就是说，两个相机共同看到的点，就要满足基本矩阵的约束。

## 14.3 当相机内参未知时

通过相机内参逆矩阵和图片上的像素位置，我们能够求得这个像素对应的向量。 p=K-1p^

我们可以用,代入qTEp中来替换掉q和p。其中p^, q^指的是像素坐标。则有：



上图中的E不是essential matrix，应该是一个单位矩阵。

在E中，p，q表示的是向量，在F中；p^,q^表示的是图像平面上的像素的齐次坐标。

### 总结

（1）两个平面的对极几何可以通过一个特殊的3×3 fundamental matrix F表示；

（2）F建立起了建立起了图片1中的点到图片2中的对极线之间的映射关系。图片1中的点p点对应与图片2中的对极线为Fp

（3）对应点的对极约束为：qTFp=0，这也就是说q点需要在p点对应的对极线上（射影几何中点与直线点乘为0，就说明点在线上）。

## 14.4 对极点（epipole）

对极点表示的是另一个相机在这张图片上的投影。只要相机位置不变，所有的对极线都要经过这两个对极点。但是epipole不一定在图像的可见范围内。

对极点可以通过F求出。

对于图片1中的任意向量p，他在第二张图片上的向量为Fp。那么必有：

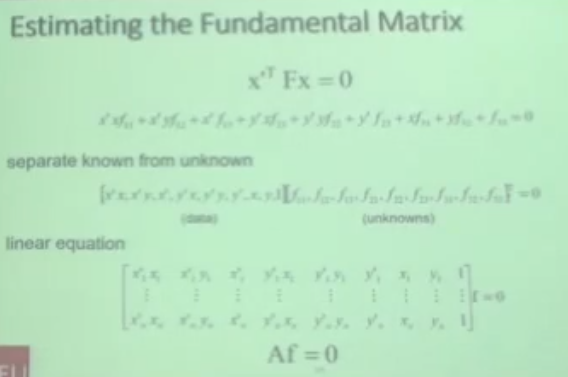
因为p不等于0，那么就有；同理。

所以3\*3矩阵F的秩为2。所以第三列可以由前两列线性表出。

## 14.5 F的计算

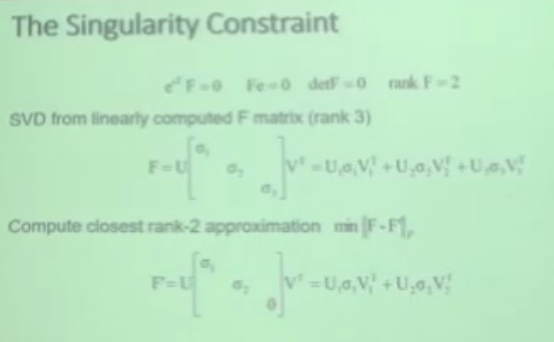
对于两张图片中的对应点x，x`，F满足约束：

将上述等式中的矩阵展开，将已知参数和位置参数分离。得到一个线性等式，然后通过多对已知对应点，应用DLT，SVD，求解F中的未知参数。



但是通过上述方法求出的F，不会是一个秩为2的矩阵，因为得到x与x`的检测方法（harris Corner detector等）会有噪声。

所以需要加一个rank constrain。将上面求出的F进行SVD分解，将中间矩阵的第三个值设为0，就得到了我们要的F。



如果F的秩为3，那么会导致epipole这个点不存在；如果F的秩为2，那么epipole line就会相较于epipole。



**8-points Algorithm**：用8点算法来计算F

**7-points Algorithm**：用7点算法来计算F。3\*3的F中有9个元素，但是因为F是define up to a scale 的，并且rank（F）=2，所以  
F的DoF是7。

## 14.6 E的计算

method 1：首先计算出F，然后把两个内参矩阵相乘求出结果，就得到E

method 2:8点算法，但是数据不是图像平面的坐标而是3D方向。最后加上rank constraint。

method 3:5点算法，得到一个1元10次的方程～。

### 根据E求出K

有了E，通过SVD，就能得到相机参数。因为在小孔成像模型中，我们并没有要求物体在小孔前面，所以会得到4个解。根据两个对应像点，通过triangulation，重建出三维点，如果这个三维点在两个相机前方，那么就是正确的解；如果在两个相机后方，那就是错误的解。

**那么如何判断点是否在相机前方呢？**

假设通过triangulation重建出了三维点X，

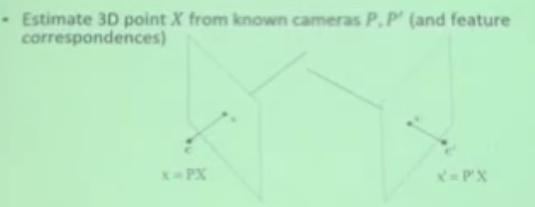
求出相机中心C，那么X-C就是表示从相机中心指向三维点的向量

在看X-C与相机朝向（相机矩阵第三行前三个元素）的家教是否大于90度，如果夹角小于90，说明点在相机前面；反之则在后面。

最后对于这4个解，计算所有重建点在两个相机前数量最多的那个作为最终解。

## 15 triangulation

## 15.1 triangulation 解决的问题



给定两个相机矩阵P&P`和两个图片上的对应点x&x`，如何求出三维点X。

## 15.2 DLT求解

x=PX x`=P`X

转化为叉乘，去掉尺度影响 [x]PX=0 [x`]P`x=0

然后再用SVD的方法求解。

## 15.3 Mid-point algorithm

理想中，我们觉得两条像素与相机光心的连线会在空间中相交于三维点X，但是实际上，由于噪声等因素，这两根线绝对不会相交，他们会是所谓的异面直线。

Mid-point的想法是，求出两条射线的公垂线，然后取公垂线的重点为我们需要重建的三维点。

## 15.4 reprojection error

这种方法的好处是可以不知道相机的内参，这在以前是很重要的，但是现在数码相机时代，相机内参都比较易得了。

最后要解决的是一个非线性优化问题。用列文伯格-马夸尔 Algorithm求解。

# 16 Structure from Motion（SFM）

## 16.1 SFM解决的问题

给定同一场景的多张输入图像，得到这个场景的三维点云以及每个相机的位姿（位置及朝向）。

structure：场景的3D点云

motion：相机的位置及朝向

SfM：get the point cloud from moving cameras

## 16.2 bundle adjustment

### BA的定义

空间中的三维点经过相机矩阵在图片上成的像素点的位置会与算法检测到的位置存在误差，所有三维点在所有图片上的投影与检测点之间都会存在误差，将所有这些误差相加，得到总的reprojection error：

g（X，R，T）

最小化这个总的reprojection error就称为BA。

### 问题的规模

假设三维点的数量为m，相机（图片）的数量为n。因为每个三维点有3个自由度，每个相机有T，R需要确定，且每个矩阵中有三个自由度。则BA中变量的数量为3m+6n。

## 16.3 BA中的算法

### Newton法

牛顿法可以用来求根和优化。

牛顿法的思想是将非线性函数泰勒展开至二次项，也就是在局部用二次方程来拟合函数。

其优点是收敛非常快，对于一元二次方程，能一步收敛；但缺点是如果步长过大，可能会发散。

求解步长时需要求出jacobian矩阵和hessian矩阵，其中hessian矩阵的计算非常复杂。

### Gauss-Newton method

基于牛顿法进行修改，得到了高斯牛顿法。

GN将函数泰勒展开，舍去多次项，只留下一阶导，那么最后就是一个线性优化问题。也就不需要求Hessian矩阵了。

### Levenberg-Maquardt algorithm

梯度下降法一定会收敛，只是收敛速度比较慢。

LM算法就是将高斯牛顿法与梯度下降算法结合。

加入一个因子λ，当λ比较小时，就还是高斯牛顿法；当λ比较大时，就更像梯度下降法了。

在迭代过程中，给定步长△与λ。走了一步后，如果误差下降了，那么就说明迭代方向正确，所以就接受△，并且减小λ，使下一步迭代更像GN，加快收敛速度。如何误差上升了，就说明迭代方向错误或者步长取大了，进而取消这一步迭代，增大λ，使下一步迭代更类似GD，保证迭代的稳定性。

现在用的比较多的LM库是google的Ceres。

## 16.4 Rotation Parameterization

这部分解决的是如何描述一个旋转矩阵。

1. R是一个3\*3的矩阵，用9个未知数表示这个矩阵。

但是这个方法明显不可行，因为R的物理意义只有3个自由度，并且旋转矩阵是一个正交矩阵。9个变量太多了，最后不知道会求出个什么结果。

（2）用欧拉角。用欧拉角来描述R，理论上是可行的，但是很麻烦，因为欧拉角不能直接进行矩阵乘法。而且欧拉角和绕什么轴旋转的顺序有关，最后会导致问题很麻烦，所以不可取。

（3）Axis-angel Repesentation。可行，但是不够好

（4）quaternion。四元数是负数的一个扩展，是相对可行的表示方法。

## 16.5 SfM的分类

### incremental SfM

首先通过两张图片求出E或F，然后SVD求得相机内参矩阵，接着通过内参矩阵和图片上的对应点，triangulate（pnp）出对应的三维点。

加入第三张图片，如果第三个相机内参未知，则先通过resection求出相机内参，在用pnp方法（如果内参已知则直接用pnp方法）重建出新的三维点（前两张图片中没有的，但存在于第三张图片和前两张图片中）

以此类推。

增量式SfM的缺点：

应用resectioning的前提是三维点的位置是绝对精确的，但是pnp出来的三维点并不是绝对精确的，所以增量式sfm会造成误差累加。那么就会要在增加了一定数量的图片后做一次BA，来缩小误差。然后再加图片，再BA。在所有图片都添加完以后，再做一次整体的BA。

### Global SfM

假设有n张图片，对每两张图片计算对应的相机内参、外参（那么就要计算次）；

将所有的图片都转移到同一个相机坐标系下，进行重建。

计算所有三维点在所有图片中总的reprojection error，进行一次总的BA，求出最优解。

## 16.6 如何选择BA的最优初始值

选择R的最优初始值的问题称为Rotation Averaging。

选择T的最优初始值的问题称为Translation Averaging。

# 17 VSLAM

VSLAM可以视作在连续的video sequence上应用SfM，而且是采用增量式的SfM。

因为实时性的限制，所以：

不会将所有的frame用来重建，而是挑出key frame进行重建，

local BA：只对相邻的frame做BA，而不是对当前时刻以前的所有frame都做BA。

# 18 Stereo（立体匹配）

给定两张标定了的照片，求出图片中每个像素的深度。深度的计算是根据每个像素在两张图片中相对的移动距离决定的。

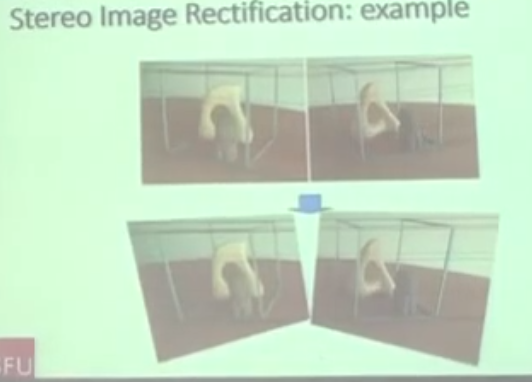
## 18.1 Stereo Image rectification

**scan line**：扫描线，原指电视画面沿着竖直方向切割成一根根细线，这些水平方向的细线构成了电视画面。引入进图像处理中指的就是将图片沿着竖直方向分割成的一根根水平细线。

在实际工程中，如果image scanline和epipolar line重合，那么会给计算带来便利。

具体的的实现：仅旋转两个相机而不移动他们。这相当于对两张图片进行了Homography。

最后会使得像素在两张图片中是水平移动。（原本在对极几何下，也只要沿着epipolar line搜索对应点，做了这样一个变换，进一步简化搜索，使得只需要沿着x方向搜索对应点。）



## 18.2 disparity（视差）

两张图片都以图片的中心为原点建立坐标系（公用坐标系），对应的像素点在两幅图中x坐标的变化值称为视差。

视差与深度的关系：

其中，x，x`为对应点在两张图片中的x坐标

d为视差

b为基线长度

f为焦距长度

Z为三维点的深度值

可以看出，视差与深度成反比。

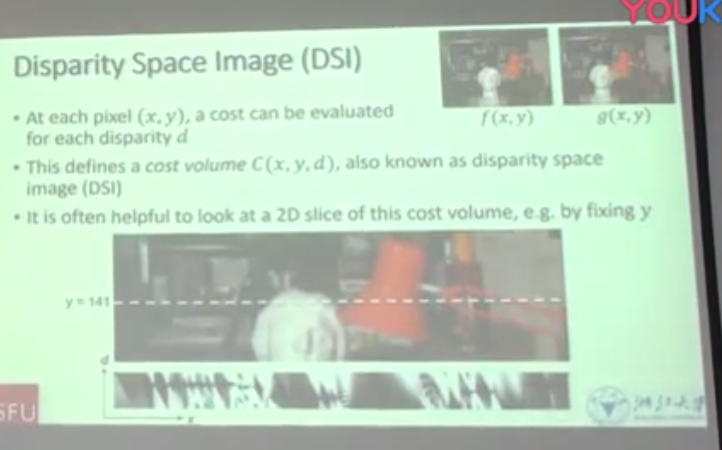
### 如何确定对应点？

对于应用了stereo image rectification的图相对，先在P1中选取一源像素点，然后在P2的epipolar line（对极线）中选取颜色值与源像素点灰度值最接近的一点作为对应点。

## 18.3 disparity space image（DSI）

对于图像P1上的一个像素点（x，y），应用不同的视差值，可以对应于P2上的另一个像素点（x`，y`），计算两个像素点之间灰度值的差距。那么对于P1上的所有像素点，应用同一个视差d，就能得到一张基于视差d的cost map；对于P1的像素点应用不同的视差d，那么便可以得到很多张cost map；将这些cost map堆叠起来，得到了cost volume，我们将之称为DSI。

### winner take all（赢者通吃）



对于DSI，比如固定y（相当与将一个立方体切一刀），那么就得到了横坐标为x，纵坐标为d的一张图，图上的内容是灰度值的差异。根据灰度值最小的点，选取相应的视差作为这个x坐标的视差。

这种从所有视差中选取最适宜的视差的思想就称为赢者通吃。

## 18.4 cost aggregation

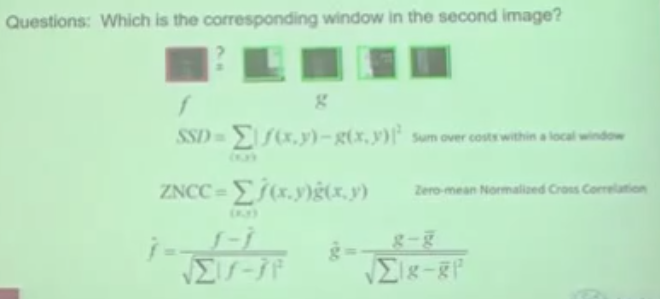
在纹理不清晰的区域，P2中的对应点向前后移动一定的距离，两个点之间的灰度差值可能变化不大，所以会得到一整个竖条颜色都相近的情况，那么就无法通过这种方法选取视差值了。

因此考虑通过比较目标像素周围的一个local window在两张图片中的对应关系，来减少这种匹配的模糊性。这种方法据称为cost aggregation。

通过CA，同样可以得到一条坐标与视差的关系图，同样通过赢者通吃的方法选取最适合的视差。

### 如何衡量两个local window的差异？

SSD，ZNCC。



## 18.5 CA的缺点

使用CA有一个默认的假设，那就是两张图片中window的深度相同，这就要求两个window都在平面上，但是这种假设当window较大时往往难以成立。

## 18.6 local methods

通过一个local window得到两幅图的像素点的对应关系，进而求得视差的方法称为local methods。

local window本质就是一个滤波器，其内权值的不同取法，会有不同的效果，也适应不同的情形。

## 18.7 global method

## 18.8 dynamic programming

# 19 optical flow（光流）

how to estimate pixel motion from image H to image I。

光流解决的问题是如何求解图像间像素的移动。

但其实要解决的根本问题还是pixel correspondence的问题。

**光流与立体匹配的区别：**光流中并没有对极约束，并没有计算E或F；而且研究的对象也可能是一个动态的场景（相机不动，场景内的物体在动）。

**光流基本假设：**

**color（brightness） constancy**：颜色恒常性。就是说同一个三维点在两张图片中的颜色应该相同。

用数学语言描述光度一致性假设

经过相同的时间间隔，像素点在x1，y1方向运动x（t），y（t）到达新位置x2、y2，新位置与旧位置的灰度值相同。也即：

**small motion**：图片间物体的运动很小。如果物体运动很大的话，就难以求解了。

用数学语言描述小位移假设：

经过相同时间间隔dt，像素点运动了dx、dy，但是因为位移很小，所以认为像素点未移动。也即：

上式被称为optical basic equation（光流基本方程）。

其中，lx，ly意味图片梯度，即像素沿着x、y方向灰度值变化的梯度。

lt为时间梯度，即相邻两帧对应位置上灰度值的差异。