待解决疑问：

自己总结待确定

* 第三、四讲有关旋转向量，旋转矩阵，欧拉角、李代数涉及较多公式，总结在书上
* Slam的重点是定位和建图。可通过姿态传感器或其他设备确定小车（相机姿态），通过？？设备确定位移，运用李代数或雅克比等算法更新姿态，使用相机确定外部环境物体在相机坐标系的位置，继而可求出外部环境物体在世界坐标系的位置，完成构图
* 单目相机一张图片只能确定空间点在成像平面的x、y坐标，还有z不能确定。只要确定z就可建立完整的坐标体系，然后就能确定空间点的位置、以及颜色，就可构建地图
* 善用机器学习的数学方法（svd，牛顿等），尝试和培养看英文文献能力
* 视觉里程计理解：初始化时，假如相机坐标系和世界坐标系重合，然后通过隔固定时间，通过比较相机移动前后两幅图像的特征点，从而获得相机移动的位姿以及特征点的位置。随着相机移动要不断选择合适的新特征点。
* 平面摄像头与深度摄像头（RGB-D、双目）通过2幅图求解位姿的相同与区别：
* 相同：可通过内参和成像平面坐标通过针孔成像原理获得特征点（路标）在相机坐标系下坐标。其中，特征点通过图像处理获得
* 不同：平面摄像头不能直接获得相对相机坐标系的深度，故只能先通过2D-2D算法求出归一化后特征点x、y坐标，在两幅图中三角测量求出深度。

而深度摄像头可通过一幅图直接获取相对相机的3D坐标，而从前一幅图可求出固定的特征点的世界坐标，所以每次移动，只需记住特征点世界坐标，就能在下一帧图像求出相机的位姿变换

* 位姿估计：就是通过在计算值附近使用调整位姿，使所有特征点在成像平面的投影方差最小。
* Bundle adjustable：通过使所有点最小方差的求解方法来估计位姿，可以通过前面的数学计算方法来计算初始值，可以减少迭代次数，更快收敛
* 看到的只是成像平面，其他的都是通过其算出来的
* cpu足够强大时，计算速度就迎刃而解了，精度也能更高
* R三个自由度（虽然有9个参数，但能找到6个约束方程），t三个自由度，所以式7.9只需六个方程就可解出，又因深度问题，无论如何有一个自由度算不出，少了一维，因为轨迹和地图扩大缩小任意倍后结果是一样的，5个方程可解出，但较难算，所以采用8个点,不管约束条件了
* 特征点法较成熟，运用较多，直接法偏学术性，不太成熟，学术研究较多
* 包含相机坐标系、像素坐标系、世界坐标系
* 单相机不能判断是物体动还是相机动，需加IMU，IMU擅长快速移动情况，相机适合长时间移动测位姿
* 关键帧是运动过程中保留下来的，参考帧是VO计算中作为参考的帧，当前帧就是最新帧
* 小范围调整位姿和空间点坐标，两者之间不会产生矛盾，可以看成两个独立变量调节
* 去均值化，去质心化，其实就是减去均值和质心，相当于均值为0点

1. 常用符号

* SO(n)（special orthogonal group）特殊正交群:n维空间的旋转矩阵，常用三维
* SE（n）（special Euclidean group）特殊欧式群：n维空间变换矩阵，常用3维
* a^：取反对称，P40,另外，Av=a，P65
* R：3维旋转矩阵，特殊正交群（SO(3)）
* T：4位非齐次变换矩阵,特殊欧式群（SE(3)）
* t：3维位移向量
* 世界坐标系下叫空间点（3D），成像平面上叫投影点（2D）
* <a,b>:表示a、b向量夹角

1. 运动方程：P23，根据传感器数据u和上一帧位姿xk-1得到当前位姿，当然还要考虑噪声wk

观测方程：可以通过已知位姿xk和路标点y得到路标点在相机图像上的投影

x：机器人（相机）位姿，不同时刻x的变化即机器人的运动

u：运动传感器（姿态传感器等）读数，也叫输入

w：噪声

z：观测数据，相机观察到的数据，空间点在相机成像平面的坐标，仅指x，y。

y：路标点，用于观测。空间中的点P，相对于世界坐标系坐标（x,y,z），要看成一个像素点对应的空间点。是给出的已知的确定的坐标点，如机器学习中的输出量，可认为是真实值。通过将其重投影到成像平面通过最小方差来减小误差

v: 噪声

k:是离散时间1……K

j：路标点1……N，通过对比2个不同时刻拍摄图像说明

深度：相对相机坐标系的z，深度较难取得

1. 常用库

* Opencv：计算机视觉
* Eigen：矩阵代数
* Ceres：来自谷歌的图优化，用于非线性优化
* g2o：图优化，也是用于非线性优化
* gtsam：因子图优化，注意与图优化的区别
* Dbow3：构建词典
* Pangolin：可视化库

<http://ttshun.com/2018/09/05/%E8%B7%A8%E5%B9%B3%E5%8F%B0%E5%8F%AF%E8%A7%86%E5%8C%96%E5%BA%93Pangolin/>

* 1

1. 问题：

* 第四讲sophus库操作

需先解压sophus文件，然后cmake，make；

源文件存在问题：

3rdparty/Sophus/sophus/so2.cpp:32:28: error: lvalue required as left operand of assignment

赋值语句中左边操作数需要时变量而不是表达式

unit\_complex\_.real() = 1.0;

^~~

/home/yiluzhang/work/slam\_14/3rdparty/Sophus/sophus/so2.cpp:33:28: error: lvalue required as left operand of assignment

unit\_complex\_.imag() = 0.0;

解决：

<https://blog.csdn.net/qq_40641575/article/details/81006349>

* ch6的g2o源码编译出错

解决

<https://www.cnblogs.com/xueyuanaichiyu/p/7921382.html>

1. 工具使用

* Ubuntu18.04系统SLAM环境配置（包括书上大多数，注意Ubuntu版本不同可能有不同问题）

<https://blog.csdn.net/qq_41814939/article/details/81556866>

* 查找Ubuntu软件包网站：<https://packages.ubuntu.com/>

查找是否包含无法定位的软件包以及正确的包名，解决“无法定位软件包”的问题。

https://blog.csdn.net/weixin\_42744670/article/details/82797586

* 安装点云库PCL(point cloud library)P99

<https://blog.csdn.net/wishchin/article/details/39754165>

sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl

sudo apt-get update

sudo apt-get install libpcl-all

错误：

http://ppa.launchpad.net/v-launchpad-jochen-sprickerhof-de/pcl/ubuntu bionic Release

404 Not Found [IP: 91.189.95.83 80]

解决：尝试过：

<https://www.linuxidc.com/Linux/2014-09/107055.htm>

最后发现是因为Ubuntu版本的差异，Ubuntu18.04使用以下命令即可：

sudo apt-get install libpcl-dev pcl-tools

* Ceres安装

按工具使用第一个网站步骤，18.04需将libcxsparse3.1.2改为libcxsparse3

因直接使用书上提供库，导致与18.04不兼容，耽误3小时左右

* g2o安装

libqglviewer-dev无法定位

解决：安装时出现提示，用libqglviewer-header取代，命令行libqglviewer后用tab键补齐，得一堆提示

libcholmod-dev

解决：命令行输入libcholmod+tab键补齐得libcholmod3

* kdevelop简单使用

<https://blog.csdn.net/qq_27806947/article/details/80119294>

1. 同时定位与地图构建SLAM(simultaneous localization and mapping)：搭载特定传感器的主体，在没有环境先验信息的情况下，于运动过程中建立环境的模型，同时估计自己的运动。目的是定位和地图构建。
2. 推荐书籍（p3）：概率机器人、计算机视觉中的多视图几何、机器人学中的状态估计。
3. 四个重要环节：视觉里程计、后端优化、回环检测、地图构建。P17
4. 网站

* 推荐书籍：P3
* 代码托管（P5）：<https://github.com/jiajunhua/gaoxiang12-slambook>
* Eigen学习：<http://eigen.tuxfamily.org/dox-devel/modules.html>
* 罗德里格斯公式（待看）：

https://en.wikipedia.org/wiki/Rodrigues%27\_rotation\_formula

* 万向锁（待看）：https://en.wikipedia.org/wiki/Gimbal\_lock
* Ceres官网

<http://www.ceres-solver.org/>

* TUM数据集，RGB-D数据集

<https://vision.in.tum.de/> P187

<https://vision.in.tum.de/data/datasets/rgbd-dataset/download>

* Ros网站：希望系统学习ros，并使用ros开发自己的slam程序

<http://wiki.ros.org/>

* 导航算法A\*

[https://en.wikipedia.org/wiki/A\*\_search\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm)

1. 常用模块

* 时间模块：ctime
* 代数库：eigen，具体见

1. 第1讲：p3推荐书籍、p5代码网站
2. 第2讲：slam简介
3. 第3讲：旋转时，将2个坐标系中心和向量起点都移到一起，那么旋转更直观。在变换前后旋转矩阵都以世界坐标系做参考，比较容易理解，记住P41-3.4。旋转的表示有旋转矩阵、旋转向量（旋转角乘单位旋转向量）、欧拉角、四元数。旋转矩阵与旋转向量之间变换P49。旋转向量与四元数之间变换P52。四元数旋转及其与旋转矩阵之间变换见P55。欧拉角可以看成先后绕固定或旋转后坐标轴旋转，那么很容易和旋转向量、旋转矩阵、四元数变换。具体换算过程还待实践验证理解。P41公式3.4，P60，注意坐标系的转换

问题：P57式3.39第四行？

1. 第4讲：

* sim（3）相似变换能够将尺度（深度信息）不确定性表示出来，相似变换不变形，是成比例的放缩，类似于相机的移动
* P72BCH（baker-campbell-hausdorff）近似
* P74李代数求导方法：
* 直接利用导数定义求导
* 左右扰动模型，能够得到扰动对结果的影响，

1. 第5讲：针孔模型、内参数（alpha、betal、f、cx、cy）、外参数（位姿）。把自己眼睛看成照相机，那么向前为z轴，向右（右手方向）为x轴，向下为y轴，得到的图片实际已经经过转换，直接把图片放到正前方即可。因为Z深度不明，故将其归一化，即使成像在前方f处，此时x、y的单位是f，经过乘一个系数alpha、beta变为像素坐标，再平移cx、cy将原点从中心移到左上角

* 畸变的两类型径向畸变、切向畸变，处理畸变的两种方法
* 常用：先去畸变，然后直接用
* 将去畸变与姿态估计结合起来
* 视差d（disparity）：P91，越远视差越小
* 深度摄像头分为
* 红外结构光（structured light）：根据结构光图案确定距离
* 飞行时间法（time-of-flight，ToF）：根据光速飞行时间确定距离

1. 第6讲：

* 状态估计的两种方法，早期的slam问题就是状态估计的问题，正是后端优化要解决的问题。根据运动方程和观测方程是否为线性判断是线性还是非线性系统，噪声是否为高斯判断是否为高斯系统。对于线性高斯系统（Linear gaussian ， LG），它的无偏最优估计使用KF（kalman filter），对于复杂非线性非高斯系统（NLNG），可使用EKF和非线性优化

1. 历史上很长一段时间，研究者们使用滤波器，尤其是扩展卡尔曼滤波器（EKF），只关心当前状态。知道21世纪早期
2. 近年来普遍使用非线性优化方法，使用所有时刻采集的数据进行状态估计，被认为优于滤波器，而成为主流方法。在计算资源允许的情况下

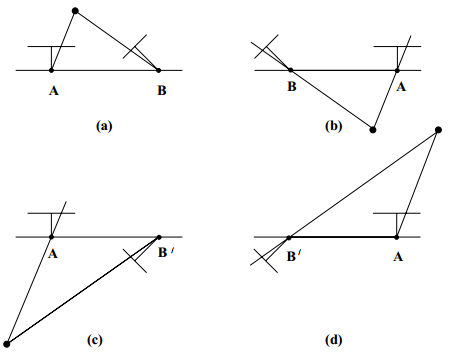
* 非线性最小二乘法，要么通过直接求导的方法（导数为0）求得，显然很多时候很难直接的到解析解，而且slam中通常是超定方程。故常用迭代的方法，以下后两种用的比较多
* 一阶和二阶梯度法：一阶（关于x的导数）是最速下降过于贪心，二阶（关于detalx的导数）要计算H矩阵，通常倾向于避免
* 高斯牛顿法（关于detalx的导数）：其中H（hessian）=JTJ是牛顿法中H的近似，可能是奇异的，增量稳定性较差，导致算法不收敛
* 列文伯格-马夸尔特（关于detalx的导数）：加了信赖区域
* 图优化：P121

问题：

* 解线性方程矩阵分解的方法：QR、cholesky
* Slam中矩阵往往是稀疏的，可以先消元，再分解，消元方法：schur、

1. 第7讲：

* 求解相机位姿根据两帧（参考帧-当前帧）图像匹配点的坐标形式可分为
* 2D-2D:初始化时用，面对尺度不确定性，需确定尺寸（使初始步t归一化或者令初始化所有特征点平均深度唯一，这种方法可以控制场景规模大小），可通过三角测量（三角化）得到像素点的三维坐标，然后就可用3D-2D方法
* 正常情况：对极几何。得到对极约束方程（p143），包含基础矩阵（fundamental matrix）F和本质矩阵（essential matrix）E，两者相差内参矩阵K。E的特征奇异值必定为两位相等的非零值和一个零值，E有5个自由度（6-1个尺度不确定性，最少需要5个点求解，但一般匹配点够用，没必要用5个点）。常用八点法（8对匹配点）求E，解包含一个自由度，但八点求出解不能满足旋转矩阵的约束，需要校正，然后再进行SVD分解求出t和R。结果有四个解，需要根据正深度选择正确解。如下图，只有a是正确的，先画出四种解位姿，然后过投影点和光心做直线，两个camera相交，通过两条线与基线的角度就可判断他们会不会相交。都是正深度为正确的



* 特征点在一个平面上（无人机常见、墙面）：单应矩阵（homography matrix）H。多了式7.16的约束，最后只需四对匹配点就能求出。然后分解求出R、t。此情形相对于普通情形，自由度降低，称为退化（degenerate），若继续用八点法将变成超定方程，受噪声影响更大。为了避免退化的影响，通常算出F和H，选择重投影小的。
* 纯旋转：无解，t=0，构建不了方程
* 3D-2D：pnp（perspective-n-point）。需要3对点计算，额外一堆点验算。最重要方法。PnP也可以在内参K未知情况下求解。有以下几种方法。通常先估计相机位姿，再用BA最小化重投影误差调整。
* 直接线性变换（direct linear transform，DLT）：需要6对点，R矩阵不满足约束，可使用QR分解从矩阵空间重新投影到流形
* P3P：余弦定理先求空间点在相机坐标系下坐标，求解用到吴消元法，结果有四个解，用验证点求最可能解。然后得到3D（世界坐标系）-3D（相机坐标系）坐标，使用ICP求解
* EPnP：用迭代方式对相机位姿进行优化，尽可能消除噪声的影响
* UPnP：用迭代方式对相机位姿进行优化，尽可能消除噪声的影响
* 非线性方法
* 3D-3D：迭代最近点（iterative closet point，ICP）：其主要研究激光slam这种没有匹配好的3D点，在视觉slam中这已经匹配好了。有以下两种方法
* SVD：P174步骤
* 非线性优化：相当于只需迭代改变位姿

问题：

* 特征提取与匹配实验，P138
* 本质矩阵奇异值，文献3
* 尺度等价性与尺度不确定性联系：因为解有一个自由度，所以若物体尺寸和距离拉近相同的倍数（x/z），结果不变
* SVD从E中分解出R、t
* 解决误匹配问题的随机采用一致性（random sample concensus，RANSACs）
* P146式7.16
* 从H矩阵分解R、t，数值分解和解析法，注意它和解线性方程的区别
* 解超定方程，如DLT超出6对点情况：最小二乘法、SVD等
* 文献3,48用QR分解将DLT结果校正
* P3P求解，文献45
* P3p用验证点求最可能解
* ICP中SVD最优结果证明50、51
* 匹配已知情况下，ICP非线性方法中可以得到解析解52-54

1. 第8讲：利用最小化光度误差的方法，适合场景明暗渐变，不形成局部图像梯度

* 只计算关键点，不计算描述子。使用光流法跟踪特征点
* 只计算关键点，不计算描述子。使用直接法计算特征点在下一时刻的位置
* 都不计算，直接计算相机运动

问题：

* 开源项目SVO56、LSD-SLAM57
* 梯度求解

1. 第9讲：设计前端
2. 第10讲:整个优化问题由许多观测方程组成，SfM（structure from motion）图像没有先后顺序，slam中有先后顺序。有以下优化方法

* 线性系统和EK：
* 非线性系统和EKF：
* BA和图优化：eij。所有i个位姿，所有j个路标点。不相关导数为0。P251公式。用到schur消元和边缘化。S矩阵能体现共视。可以使用鲁棒核函数huber、cauchy、tukey降低误匹配的影响

1. 第11讲

* 位姿图（pose graph）：只优化位姿。路标点数量远远大于位姿，这样可以减小优化量
* 滑动窗口法：丢弃一些历史数据
* 因子（factor）图优化：与图优化有些相似，但它可以选择哪些存储信息可以重用

问题

* Schur消元，边缘化68。求解xp方程
* Pose graph：78-80
* 滑动窗口：77
* 因子图优化：83-84
* QR、choleshy、schur怎么加速求解P288

1. 第12讲：回环检测

* 两幅图像相似性（similarity）统计方法
* 两幅图像相减
* bag-of-word：每位表示某word（slam中特征点）有无或者数量。使用kd-tree将特征点聚类加速查找，达到log级。只需将当前帧特征点与k层聚类中心比较（d次比较）来搜索特征属于哪个词，然后将图像向量相应位置1，最后得到图像的向量。还可在词典中加入相应词的权重w，根据TF（某特种在图像中频率，越大越好）-IDF（某特征在全部图中频率，越小越好，所以用倒数）（term frequency-inverse document frequency），P310式12.7、12.8、12.9。
* list-of-word：含有词的顺序
* 机器学习和深度学习可能可以更好的识别
* 准确率（precision）和召回率（recall）：两者相互矛盾，slam中准确率比较重要。假阳性是感知偏差，假阴性是感知变异

问题

* 查找方法P305：94-96fabmap、97chou-liu tree、树结构98
* 聚类方法：92K-means、层次聚类、k-means++93
* TF-IDF:100-101
* 传统机器学习方法：104-105

1. 第13讲：稀疏地图智能定位，稠密地图才能用于导航、避障和重建（给人看），语义地图用于交互

* slam:重视定位，建图没那么重要。单目可利用极线搜索和快匹配来匹配合适区域（P324SAD（sum of absolute difference）、SSD（sum of square distance）、NCC（normalized cross correlation）），精度和效率经常会冲突，使用深度滤波器来估计
* 3D重建：重视建图，不那么重视定位。常使用RGB-D重建，包括如下方法，定位使用ICP
* 直接拼接
* TSDF和fusion系列：P353

问题

* 109块匹配
* 56、112深度滤波器
* 考虑如何使用GPU并行化加速P342、P353
* TSDF（truncated signed distance function，截断符号距离函数）和fusion系列：P353，120-124

1. 第14讲：现在和未来方案介绍，注意阅读文中相关方向文献。未来方向如下

* VIO:128
* 语义slam：语义和slam可以相互帮助。135-138
* 基于线面特征的slam：154-156
* 动态场景下的slam：157-159
* 多机器人的slam：67、160、161

1. 旋转的四种常用方法：旋转矩阵、旋转向量、齐次变换矩阵、四元数
2. 李群（旋转矩阵和齐次变换矩阵）不满足加法，因为相加就不满足正交矩阵的约束了，李代数（三维向量和六维向量）可以相加，因为三维向量没有约束。且三维向量和旋转矩阵一一对应，六维向量和齐次变换矩阵一一对应。
3. 针孔模型：联系光的直线传播，无论从哪看，只能通过小孔看到对面。相对于外部环境来说镜头相当于一个针孔。
4. 光流估计仅估计了像素间的移动，但：

* 没有用到相机本身的几何结构
* 没有考虑相机的旋转和图像的缩放
* 直接法则考虑这些信息

1. P284人们倾向于把前端和后端分开，运行于两个独立线程之中，历史上称为跟踪和建图，建图部分主要指后端优化的内容，前端需要实时的响应，如每秒30帧，而优化可以慢悠悠的进行，只要优化完成返回前端即可，对速度要求低。
2. 应用场景：室内扫地机器人、移动机器人、野外自动驾驶汽车、无人机、虚拟现实、增强现实
3. 相机性质：近处物体运动快，远处物体运动慢。近处大，远处小。
4. 单目摄像头的优缺点：结构简单成本低。需通过2帧图像测深度，对于单张图像具有尺度（scale，物体的大小）不确定性
5. 双目的优缺点：可通过单帧图像测距离。深度量程和精度受双目基线和分辨率影响，视差需要计算量大，要做到实时需用GPU和FPGA，成本高。
6. 深度摄像头的优缺点：通过过红外结构光或time-of-flight(ToF)的原理测深度，可以节省计算资源。多数存在测量范围窄、噪声大、视野小、易受日光干扰、无法测量透射材质等诸多问题，主要用于室内，室外较难应用
7. 地图的构建：不同情况对地图有不同的需要，比如扫地机器人，只需二维网格地图，在方格中标注可不可以通过就可以。地图可分为：

* 度量地图（metric map）：二维度量地图是小格子（grid），三维地图是小方块（voxel），可以用占据、空闲、未知来标注各个属性，判断其中是否有物体。用于各种导航算法，如A\*、D\*（p22有网址）。
* 拓扑地图（topological map）：由节点和边构成，如A、B可以通行，将两者连接起来。在其上的导航和路径规划的方法还有待研究

1. 在视觉slam优化方法（P111）:高斯牛顿法和列文伯格-马夸尔特方法。最后得到的是线性方程求解增量。对于一系列的位姿变换，增量往往达到很高维，直接进行求逆计算量太大，往往使用线性方程的数值求解方法。不同领域使用的方法不一样，几乎不会直接求逆。slam中会用到QR、Cholesky等。视觉slam中这个矩阵往往有特定的稀疏形式，为实时求解提供了可能。利用稀疏形式的消元、分解，最后进行求解增量，会让求解效率大大提高
2. 曾产生疑问

* 深入学习cmake，怎么直接使用库文件？

解决：p29，第二讲习题，《cmake 实践》

* 测相机内参的方法
* 矩阵求导?P65
* 验证李代数？P67，看推荐网站相关英文推理过程
* 路标点怎么找？P74怎么用路标点、位姿做优化？

解决：路标点用两幅图的特征匹配来找相同空间点对应像素点作为路标点，将成像平面像素点作为实际值，通过八点法和三角测量求出初始值，用BA优化

* 第四讲sophus库操作实现步骤待考究，P76
* 文献21，P79，相似变换，单目时有用
* 怎么构建地图？存储数据？路径规划？
* 怎么减小噪声？怎么减小单目相机尺度不确定性，尺度漂移

解决：BA优化来减小噪声

三角测量来测尺度，回环检测或IMU来减小尺度不确定性及漂移

* 切向畸变校正数学原理
* 双目如何确实对应同一个点?

解决：特征匹配

* 状态估计时，待估计量和已知量，及他们之间关系。

解决：已知量：观测值（像素坐标u、v）、初始化后通过上两幅图像得到的特征点（路标点）在世界坐标系下的坐标

待估计量（随机给定初始值或通过第七章方法算出）：位姿、新特征点在当前相机坐标系下坐标

目的：最小化预测值（式5.9由位姿到像素坐标）与观测值之间的方差

方法：用牛顿方法、最小二乘法等（如式7.45、7.47），看能不能用EM算法

* 特征点只能知道特征点的位置，怎么确定所有点位置呢？

解决：第13讲，极线搜索和块匹配来稠密建图

* 构图是指所有的点，还是一些点？

解决（P196）:稀疏法可以快速求解相机位姿，适用实时性较高的场合；稠密法可以建立完整地图，需要用GPU

* orb原理
* P143，144，SVD（奇异解分解），奇异值
* 测试用相机和用传感器测出有什么不同

解决：用实验2查看相机计算位姿和mpu6050差别

* 归一化理解，尺度单位怎么来

解：归一化后可令3D变为深度为1上的2D,求出x/z，y/z。对于2D尺度是以第一次初始化平移距离为单位的

* DLT求解、QR分解
* 经典书：SLAM for dummies
* 计算出特征点位置、以及相机位姿后，怎么判断障碍物，怎么规划路径和移动

解决：13讲，本书仅详解了定位，还有导航（路径规划，知道全局后）、避障（判断障碍物、然后避障）、交互（人和地图互动）。具体方法有网格法、八叉树法、点云地图等。

* 纹理信息？梯度大的地方?如衣褶
* 对于相机来说，灰度值如何形成，和光照、材料、曝光的具体关系
* 用python或其他工具处理文件，得到想要的格式
* 式7.45怎么用来优化？

解决（待验证）：见书P164

* 光流法怎么用？求出相同点便可求出位姿？
* 卡尔曼滤波器？
* 各章最小二乘法的区别
* 雅克比矩阵的深入理解
* H、F阵理解
* Schur消元
* 卡尔曼滤波器原理
* 运动怎么和概率扯上关系？

解决：噪声（测量误差等等）带来，一般认为噪声为高斯分布，即为钟型，也就是高斯均值出现的概率最大，用回归来估计噪声参数

* 边缘化
* 零空间问题
* 熟悉g2o、ceres等库， 尝试ceres消元
* 怎样算大场景，怎样算更大场景？
* 视源科技限制：
* dynamic objects（动态物体）
* occulusion（闭塞？）
* motion blur（动态模糊）
* illumination change（光照变化）
* extremely dark（特别黑）
* low texture environment（不好的环境）
* 对边的理解
* 高斯分布方差怎么算
* 回环检测怎么优化
* 是否可以新建墙类等
* ros（robot operate system）：导航、通信、路径规划
* agv机器人调度算法

1. 4