Homework2

一.作为图像描述的关键点，图像特征点需具备哪些性质？请给出fast，ORB和sift特征点概念及检测原理。

**特征点：**图像当中具有代表性的部分

• 可重复性: 在不同图像中能重现

• 可区别性: 不同的点有不同的表达

• 高效: 特征点数量应远小于像素的数量

• 本地: 特征仅与一小片图像区域相关

特征点的信息

• 位置、大小、方向、评分等——关键点(Key point)

• 特征点周围的图像信息——描述子（Descriptor）

1. FAST角点

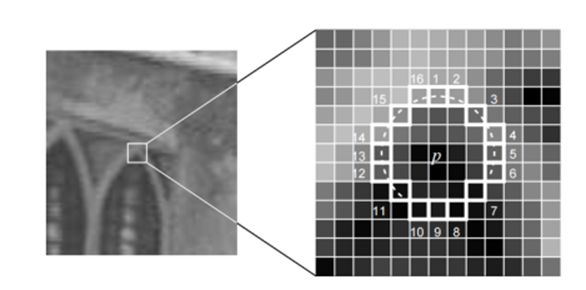
FAST 是一种角点，主要检测局部像素灰度变化明显的地方，以速度快著称。它的思想是：如果一个像素与它邻域的像素差别较大（过亮或过暗） , 那它更可能是角点。相比于其他角点检测算法， FAST 只需比较像素亮度的大小，十分快捷。它的检测过程如下：

1. 在图像中选取像素 p，假设它的亮度为 Ip。

2. 设置一个阈值 T (比如 Ip 的 20%)。

3. 以像素 p 为中心, 选取半径为 3 的圆上的 16 个像素点。

4. 假如选取的圆上，有连续的 N 个点的亮度大于 Ip + T 或小于 Ip − T，那么像素 p可以被认为是特征点 (N 通常取 12，即为 FAST-12。其它常用的 N 取值为 9 和 11，他们分别被称为 FAST-9， FAST-11)。

5. 循环以上四步，对每一个像素执行相同的操作。

以p为中心，以3为半径的圆上16个像素点，有N个点符合判据，则p为特征点，N取12,11或9

特征：实时性高，原始角点比较集中，进行非极大值抑制，避免角点过于集中。连续N个点的灰度有明显差异，约束较为单一。

其次， FAST 角点不具有方向信息。而且，由于它固定取半径为 3 的圆，存在尺度问题：远处看着像是角点的地方，接近后看可能就不是角点了。针对 FAST 角点不具有方向性和尺度的弱点， ORB 添加了尺度和旋转的描述。

（二） ORB特征点：

ORB 特征亦由关键点和描述子两部分组成。它的关键点称为“Oriented FAST”，是一种改进的 FAST 角点，什么是 FAST 角点我们将在下文介绍。它的描述子称为 BRIEF（Binary Robust Independent Elementary Features）。因此，提取 ORB 特征分为两个步骤：

1. FAST 角点提取：找出图像中的” 角点”。相较于原版的 FAST, ORB 中计算了特征点的主方向，为后续的 BRIEF 描述子增加了旋转不变特性。

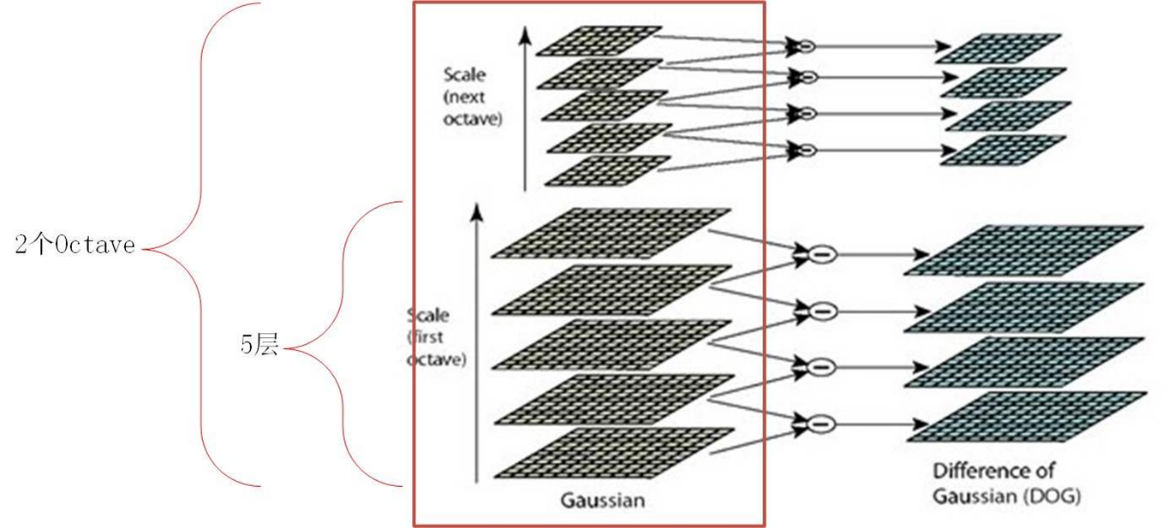
2. BRIEF 描述子：对前一步提取出特征点的周围图像区域进行描述。下面我们介绍 BRIEF。

由于FAST 角点不具有方向信息。而且，由于它固定取半径为 3 的圆，存在尺度问题：远处看着像是角点的地方，接近后看可能就不是角点了。针对 FAST 角点不具有方向性和尺度的弱点， ORB 添加了尺度和旋转的描述。尺度不变性由构建图像金字塔‹，并在金字塔的每一层上检测角点来实现。而特征的旋转是由灰度质心法（Intensity Centroid）实现的。

BRIEF 描述子

在提取 Oriented FAST 关键点后，我们对每个点计算其描述子。 ORB 使用改进的BRIEF 特征描述。我们先来讲 BRIEF 是什么。‹金字塔是指对图像进行不同层次的降采样，以获得不同分辨率的图像。

BRIEF 是一种二进制描述子，它的描述向量由许多个 0 和 1 组成，这里的 0 和 1 编码了关键点附近两个像素（比如说 p 和 q）的大小关系：如果 p 比 q 大，则取 1，反之就取 0。如果我们取了 128 个这样的 p; q，最后就得到 128 维由 0， 1 组成的向量。那么， p和 q 如何选取呢？在作者原始的论文中给出了若干种挑选方法，大体上都是按照某种概率分布，随机地挑选 p 和 q 的位置，读者可以阅读 BRIEF 论文或 OpenCV 源码以查看它的具体实现 。 BRIEF 使用了随机选点的比较，速度非常快，而且由于使用了二进制表达，存储起来也十分方便，适用于实时的图像匹配。原始的 BRIEF 描述子不具有旋转不变性的，因此在图像发生旋转时容易丢失。而 ORB 在 FAST 特征点提取阶段计算了关键点的方向，所以可以利用方向信息，计算了旋转之后的“Steer BRIEF”特征，使 ORB 的描述子具有较好的旋转不变性。

尺度上引入图像金字塔，在FAST基础上计算旋转，添加描述：BRIEF

（三） SIFT特征点：

SIFT，即尺度不变特征变换（Scale-invariant feature transform，SIFT），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。

1 特征提取

1.1 构建尺度空间

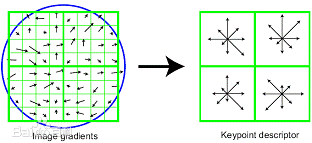
1.2 选取特征点

   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

1.3 去除不好的特征点

2 特征描述子

        用直方图统计特征点领域内的像素梯度方向，取直方图bin值最大的以及超过最大bin值80%的那些方向做为特征点的主方向。论文中是45°为一个柱状图，并且使用了高斯函数平滑曲线。再将特征点领域内的图像旋转，使特征方向与X轴方向平行，保证旋转不变性。选取特征点周围16\*16区域，划分为4\*4=16块，求取每一块在8个方向上的梯度，这样就得到了4\*4\*8=128维的SIFT描述子。



3 描述子特点

        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。

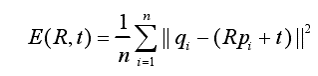
        旋转不变性：取描述子的时候讲特征点周围旋转到了X轴方向；

        尺寸无关性：构建尺度空间的时候，构建了不同尺度缩放下的图片，在求取特征点的时候，在多种尺度空间下进行了检测；

        亮度变化抗性：将描述子中各个维度的归一化，可以减小亮度变化的影响。

二.在视觉里程计中，有多种求解相机运动的方法，请简述ICP法的原理，并给出DLT法和ICP法的不同之处。

1．ICP算法的基本原理是：分别在带匹配的目标点云P和源点云Q中，按照一定的约束条件，找到最邻近点（pi，qi），然后计算出最优匹配参数R和t，使得误差函数最小。误差函数为E（R，t）为：



其中n为最邻近点对的个数，pi为目标点云 P 中的一点，qi 为源点云 Q 中与pi对应的最近点，R 为旋转矩阵，t为平移向量。

ICP算法步骤：

（1）在目标点云P中取点集pi∈P；

（2）找出源点云Q中的对应点集qi∈Q，使得||qi-pi||=min；

（3）计算旋转矩阵R和平移矩阵t，使得误差函数最小；

（4）对pi使用上一步求得的旋转矩阵R和平移矩阵t进行旋转和平移变换，的到新的对应点集pi’={pi’=Rpi+t,pi∈P}；

（5）计算pi’与对应点集qi的平均距离；

https://img-blog.csdn.net/20180511160030251

（6）如果d小于某一给定的阈值或者大于预设的最大迭代次数，则停止迭代计算。否则返回第2步，直到满足收敛条件为止。

ICP算法关键点：

（1）原始点集的采集 均匀采样、随机采样和法矢采样

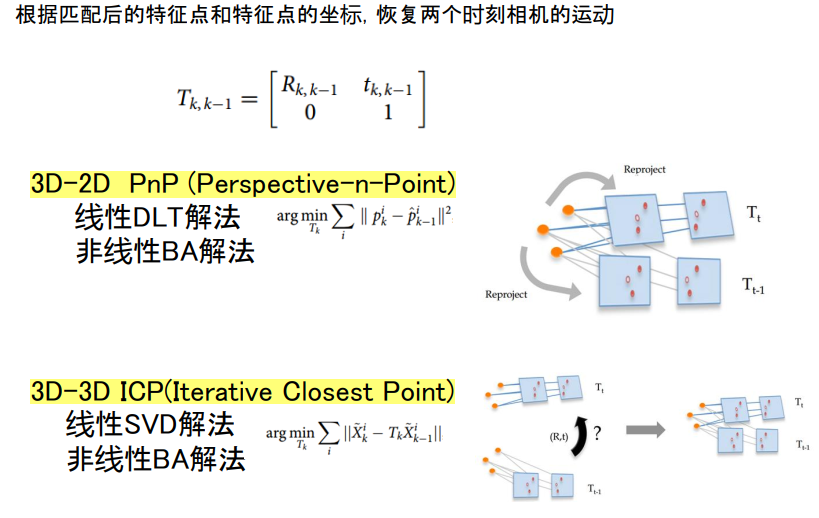
（2）确定对应点集 点到点、点到投影、点到面

（3）计算变化矩阵 四元数法、SVD奇异值分解法

2、DLT法叫做直接线性变换，ICP法叫做迭代最近点

总的来说DLT法是通过几何关系通过线性代数根据直接求解相机位姿，

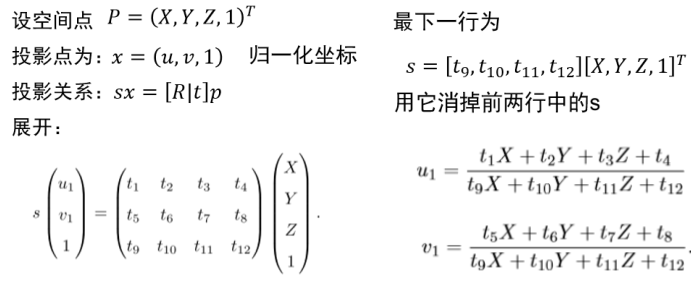
而ICP法是一种通过误差与优化的最小二乘法的优化问题。

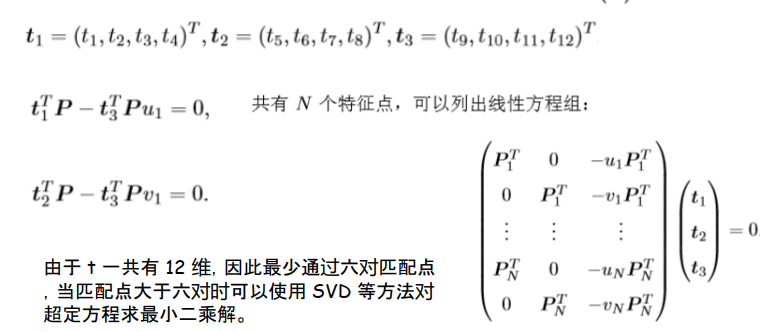


一个是3D-2D的一种解法，一个是3D-3D的一种解法

DLT直接线性解法：3D-2D

PnP已知3D点的空间位置和相机上的投影点，求相机的旋转和平移（外参）





三.阅读课程QQ群文件《视觉SLAM十四讲》第12章内容，简述词袋模型原理，运行配套代码，实现字典生成和基于词袋模型的图像相似度计算，在linux环境中给出运行结果并截图。

**词袋模型：**

将所有词语装进一个袋子里，不考虑其词法和语序的问题，即每个词语都是独立的。例如下面2个例句，就可以构成一个词袋，袋子里包括Jane、wants、to、go、Shenzhen、Bob、Shanghai。假设建立一个数组（或词典）用于映射匹配

[Jane, wants, to, go, Shenzhen, Bob, Shanghai]

那么下面两个例句就可以用以下两个向量表示，对应的下标与映射数组的下标相匹配，其值为该词语出现的次数：

Jane wants to go to Shenzhen.

Bob wants to go to Shanghai.

词袋

[1,1,2,1,1,0,0]

[0,1,2,1,0,1,1]

这两个词频向量就是词袋模型，可以很明显的看到语序关系已经完全丢失。类似一个集合的概念，其实除了词袋模型外还有词向量模型word2vec常用于NLP领域，基础的底层采用基于CBOW和Skip-Gram算法，这里就不展开赘述了。其实词袋模型不仅仅用于NLP领域，而且可以在SLAM的回环检测中大显身手。

SLAM 主体（前端、后端）主要的目的在于估计相机运动，而回环检测模块，无论是目标上还是方法上，都与前面讲的内容相差较大，所以通常被认为是一个独立的模块。我们将介绍主流视觉 SLAM中检测回环的方式：词袋模型，并通过 DBoW 库上的程序实验，使读者得到更加直观的理解

词袋，也就是 Bag-of-Words（BoW），目的是用“图像上有哪几种特征”来描述一个图像。例如，如果某个照片，我们说里面有一个人、一辆车；而另一张则有两个人、一只狗。根据这样的描述，可以度量这两个图像的相似性。再具体一些，我们要做以下几件事：

1. 确定“人、车、狗”等概念——对应于 BoW 中的“单词”（Word），许多单词放在一起，组成了“字典”（Dictionary）

2. 确定一张图像中，出现了哪些在字典中定义的概念——我们用单词出现的情况（或直方图）描述整张图像。这就把一个图像转换成了一个向量的描述。

3. 比较上一步中的描述的相似程度

**字典：**

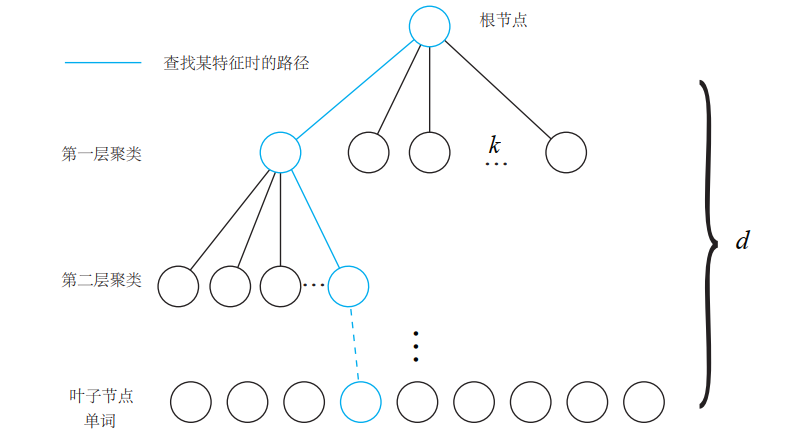
字典由很多单词组成，而每一个单词代表了一个概念。一个单词与一个单独的特征点不同，它不是从单个图像上提取出来的，而是某一类特征的组合。所以，字典生成问题类似于一个聚类（Clustering）问题。

但是字典里的单词太多了，不好管理怎么办，我们可以使用一种 k 叉树来表达字典。它的思路很简单，类似于层次聚类，是 kmeans 的直接扩展。假定我们有 N 个特征点，希望构建一个深度为 d，每次分叉为 k 的树，那么做法如下

1. 在根节点，用 k-means 把所有样本聚成 k 类（实际中为保证聚类均匀性会使用k-means++）。这样得到了第一层。

2. 对第一层的每个节点，把属于该节点的样本再聚成 k 类，得到下一层。

3. 依此类推，最后得到叶子层。叶子层即为所谓的 Words。



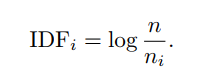
实际上，最终我们仍在叶子层构建了单词，而树结构中的中间节点仅供快速查找时使用。这样一个 k 分支，深度为 d 的树，可以容纳 kd 个单词。另一方面，在查找某个给定特征对应的单词时，只需将它与每个中间结点的聚类中心比较（一共 d 次），即可找到最后的单词，保证了对数级别的查找效率

**相似度计算：**

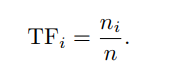
有了字典之后，给定任意特征 fi，只要在字典树中逐层查找，最后都能找到与之对应的单词 wj——当字典足够大时，我们可以认为 fi 和wj 来自同一类物体。

不过我们希望对单词的区分性或重要性加以评估，给它们不同的权值以起到更好的效果。常用的一种做法称为 TF-IDF（Term Frequency– Inverse DocumentFrequency） [100, 101]，或译频率-逆文档频率›。 TF 部分的思想是，某单词在一个图像中经常出现，它的区分度就高。另一方面， IDF 的思想是，某单词在字典中出现的频率越低，则分类图像时区分度越高。

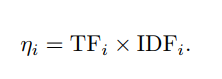
在词袋模型中，在建立字典时可以考虑 IDF 部分。我们统计某个叶子节点 wi 中的特征数量相对于所有特征数量的比例，作为 IDF 部分。假设所有特征数量为 n， wi 数量为ni，那么该单词的 IDF 为：



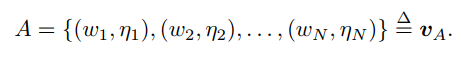
另一方面， TF 部分则是指某个特征在单个图像中出现的频率。假设图像 A 中，单词wi 出现了 ni 次，而一共出现的单词次数为 n，那么 TF 为：



于是 wi 的权重等于 TF 乘 IDF 之积：



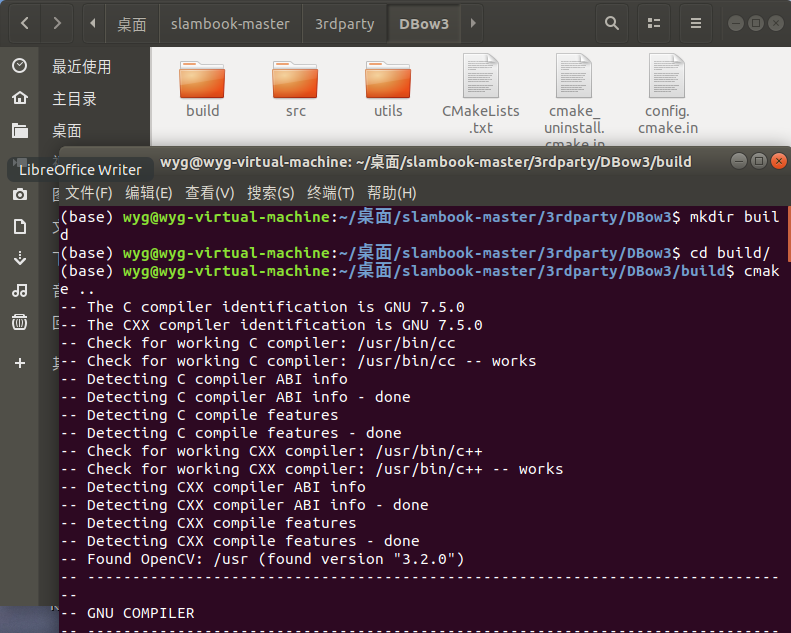
考虑权重以后，对于某个图像 A，它的特征点可对应到许多个单词，组成它的 Bag-ofWords：



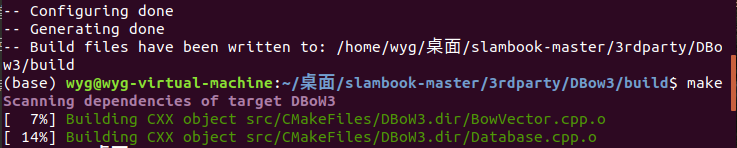
由于相似的特征可能落到同一个类中，因此实际的 vA 中会存在大量的零。无论如何，通过词袋，我们用单个向量 vA 描述了一个图像 A。这个向量 vA 是一个稀疏的向量，它的非零部分指示出图像 A 中含有哪些单词，而这些部分的值为 TF-IDF 的值。

Ubuntu18.04下验证实验：

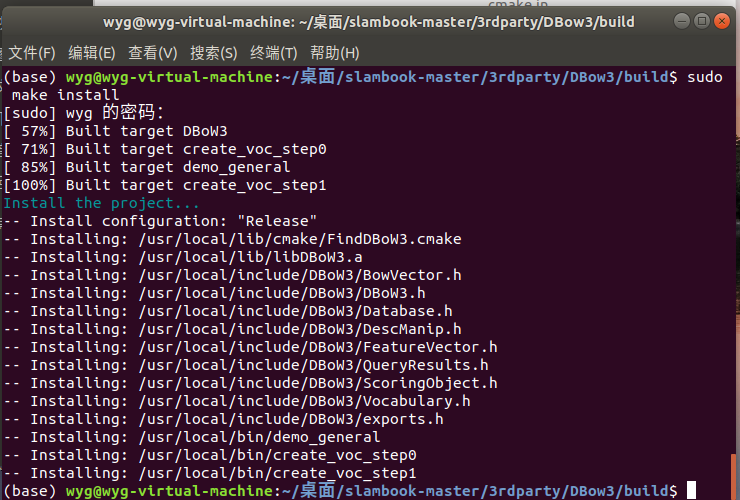
安装BOW3



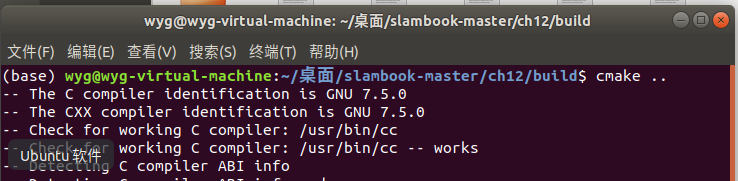
编译

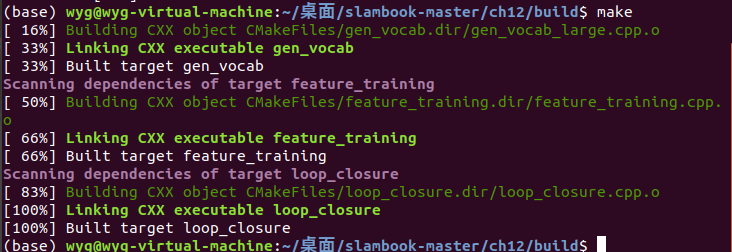


DBOW3安装完成

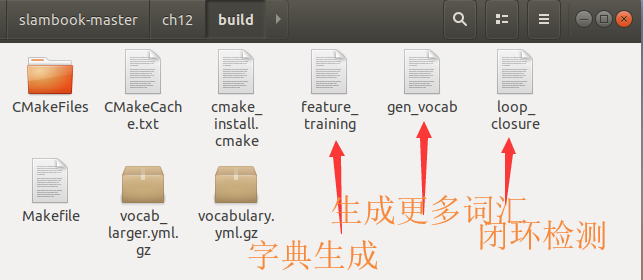


编译第十二章内容

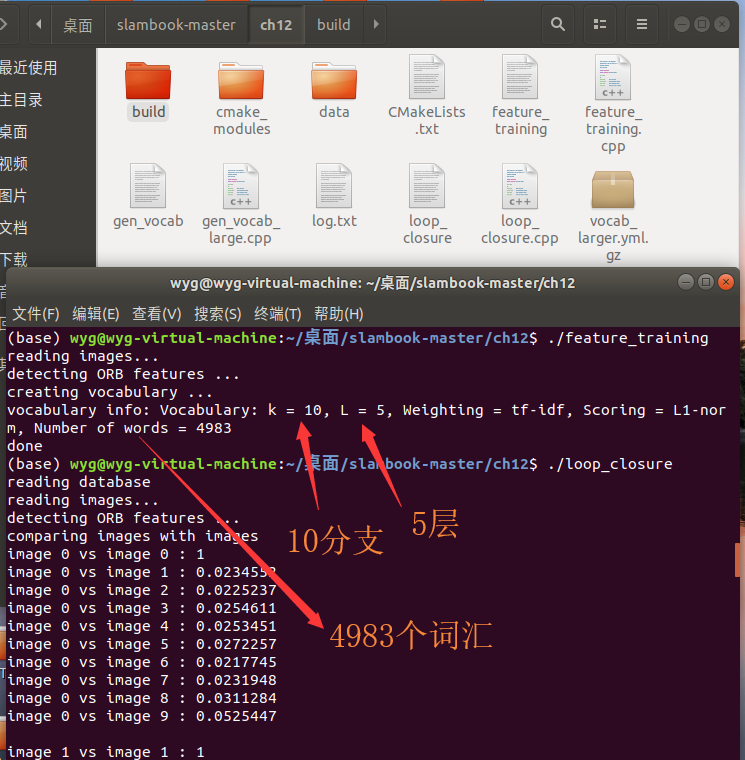




生成可执行文件

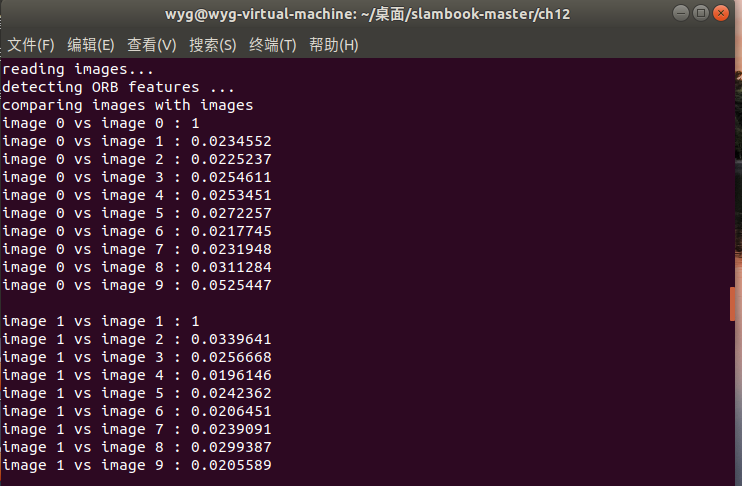


不过发现文件夹的位置不对，在当前目录下没有data文件夹，所以把这三个可执行文件放在上一级文件夹之中，并依次运行验证。

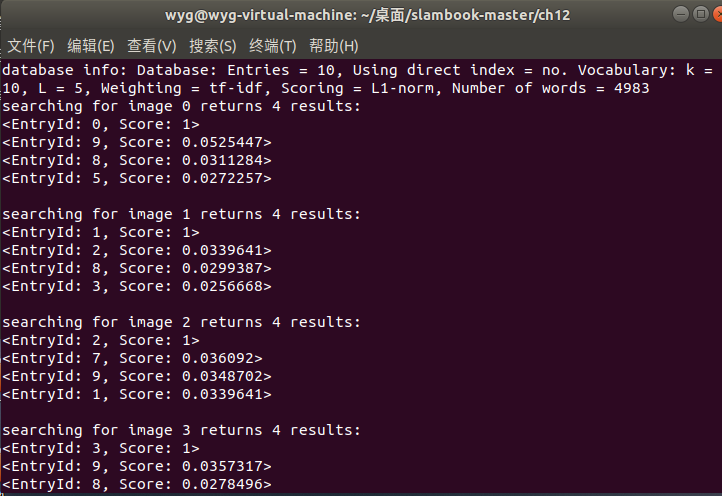


在feature\_training.cpp中，我们开始训练字典，先读入图片，送到向量容器中，对十张目标图像提取 ORB 特征并存放至 vector容器中，然后调用 DBoW3 的字典生成接口即可。在 DBoW3::Vocabulary 对象的构造函数中，我们能够指定树的分叉数量以及深度，不过这里使用了默认构造函数，也就是 k =10; d = 5。这是一个小规模的字典，最大能容纳 10000 个单词。对于图像特征，我们亦使用默认参数，即每张图像 500 个特征点。最后我们把字典存储为一个压缩文件。我们看到输出的字典的信息：分支数量 k 为 10，深度 L 为 5‹，单词数量为 4983，没有充满最大容量。

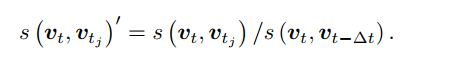
剩下的Weighting是权重， Scoring指的是评分，但评分是如何计算的呢？他们和相似度又有什么关系？



可以看出任意两张图片之间都被比较了相似度，一共C210 =45个比较结果，在数据库查询时， DBoW 对上面的分数进行排序，给出最相似的结果：（其实就是对相似度进行了筛选）



对任意两个图像，我们都能给出一个相似性评分，但是只利用这个分值的绝对大小，并不一定有很好的帮助。譬如说，有些环境的外观本来就很相似，像办公室往往有很多同款式的桌椅；另一些环境则各个地方都有很大的不同。考虑到这种情况，我们会取一个先验相似度 s (vt; vt−∆t)，它表示某时刻关键帧图像与上一时刻的关键帧的相似性。然后，其他的分值都参照这个值进行归一化：



站在这个角度上，我们说：如果当前帧与之前某关键帧的相似度，超过当前帧与上一个关键帧相似度的 3 倍，就认为可能存在回环。这个步骤避免了引入绝对的相似性阈值，使得算法能够适应更多的环境。

四.为什么说移动机器人的定位问题是一个状态估计问题？调研状态估计优化算法中(1)卡尔曼滤波，(2)扩展卡尔曼滤波，(3)粒子滤波和(4)图优化中 任一个方法的原理和计算实例，并给出自己的理解。

1. 为什么说移动机器人的定位问题是一个状态估计问题？

本质上可以把定位理解为一个估计问题，将传感器测量好的数据来估计位置，也可以定义为一个优化问题，通过多种观测数据之间的约束关系，对位置进行优化。对于机器人来说，最重要的传感器应该是摄像机了，获取了丰富的图像信息，除此之外还有激光雷达、红外传感器、超声波传感器等等，这么多传感器如何协同工作，而且不同时刻的观测数据是不是都可以用来总结出一个最合理的位置信息，我觉得离不开状态估计的工作。

早期的 SLAM 问题是一个状态估计问题——正是后端优化要解决的东西。在最早提出 SLAM 的一系列论文中，当时的人们称它为“空间状态不确定性的估计”（Spatial Uncertainty） [4, 11]。虽然有一些晦涩，但也确实反映出了 SLAM 问题的本质： 对运动主体自身和周围环境空间不确定性的估计。为了解决 SLAM，我们需要状态估计理论，把定位和建图的不确定性表达出来，然后采用滤波器或非线性优化，去估计状态的均值和不确定性（方差）。主流视觉 SLAM 使用以图优化（Graph Optimization）为代表的优化技术进行状态估计。

2.卡尔曼滤波理解：

通俗解释

对动态事物的感知我们往往有两种方法，一种是预测，一种是观测。但是无论是预测还是观测均避免不了误差，比如天气情况可以用一组非线性偏微分方程算出来，也可以用各个基站测量出来，但是均有误差。再比如导弹的运行轨迹我们可以预测也可以通过雷达观测，但是也均有误差。那么自然的问题就是如何在预测和观测给出一个最优的估计值。卡尔曼告诉了我们答案。答案很简单就是从两个方面观察得到，一是静态，如果预测的误差大观测的误差小我们有理由相信最优解更加靠近观测反之亦然；二是动态的角度，如果观测的误差大我们有理由相信下一时刻观测的贡献要小一些。我称之为动态的最小二乘，这便是卡尔曼算法。对未来时刻用卡氏算法我们称之为预测，对当下的结果用卡氏算法我们称之为滤波，对过去的结果用卡氏算法我们称之为平滑。

先验概率和后验概率

先验概率是指根据以往经验和分析得出的概率，比如说全概率公式 ，它往往作为“由因求果”问题中的“因”出现。比如说我们人有一个常识,比如扔一枚硬币,我们都知道概率是1/2,而且无数次重复实验也表明是这个数,这是一种我们人的常识,也是我们在不知道任何情况下必然会说出的一个值.而所谓的先验概率是我们人在未知条件下对事件发生可能性猜测的数学表示。

后验概率是指在得到“结果”的信息后重新修正的概率，是“执果寻因”问题中的“因” ，如贝叶斯公式后验概率是基于新的信息，修正原来的先验概率后所获得的更接近实际情况的概率估计。后验概率 : 事情已经发生，要求这件事情发生的原因是由某个因素引起的可能性的大小。

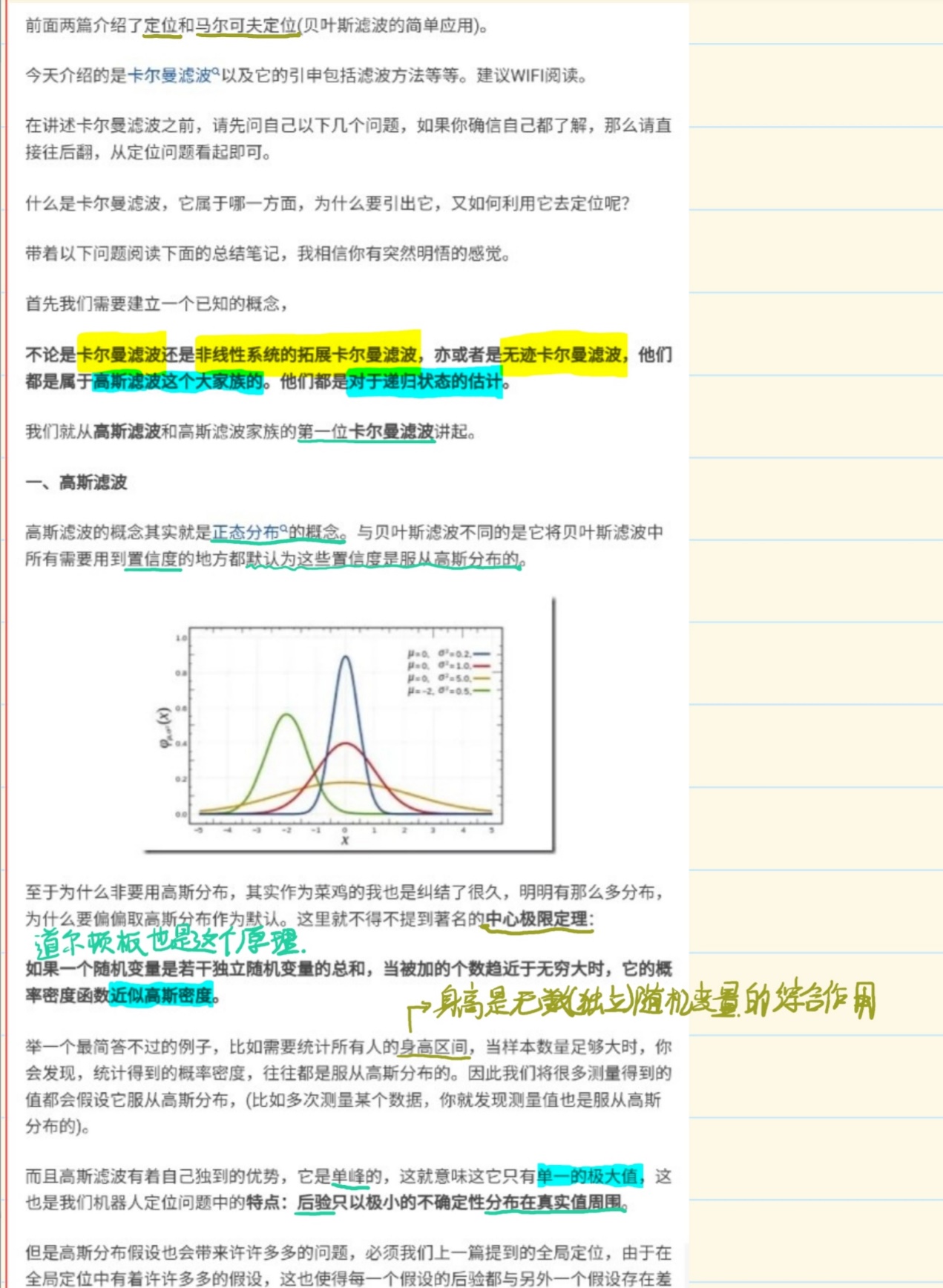
先验概率和后验概率是相对的。如果以后还有新的信息引入，更新了现在所谓的后验概率，得到了新的概率值，那么这个新的概率值被称为后验概率。

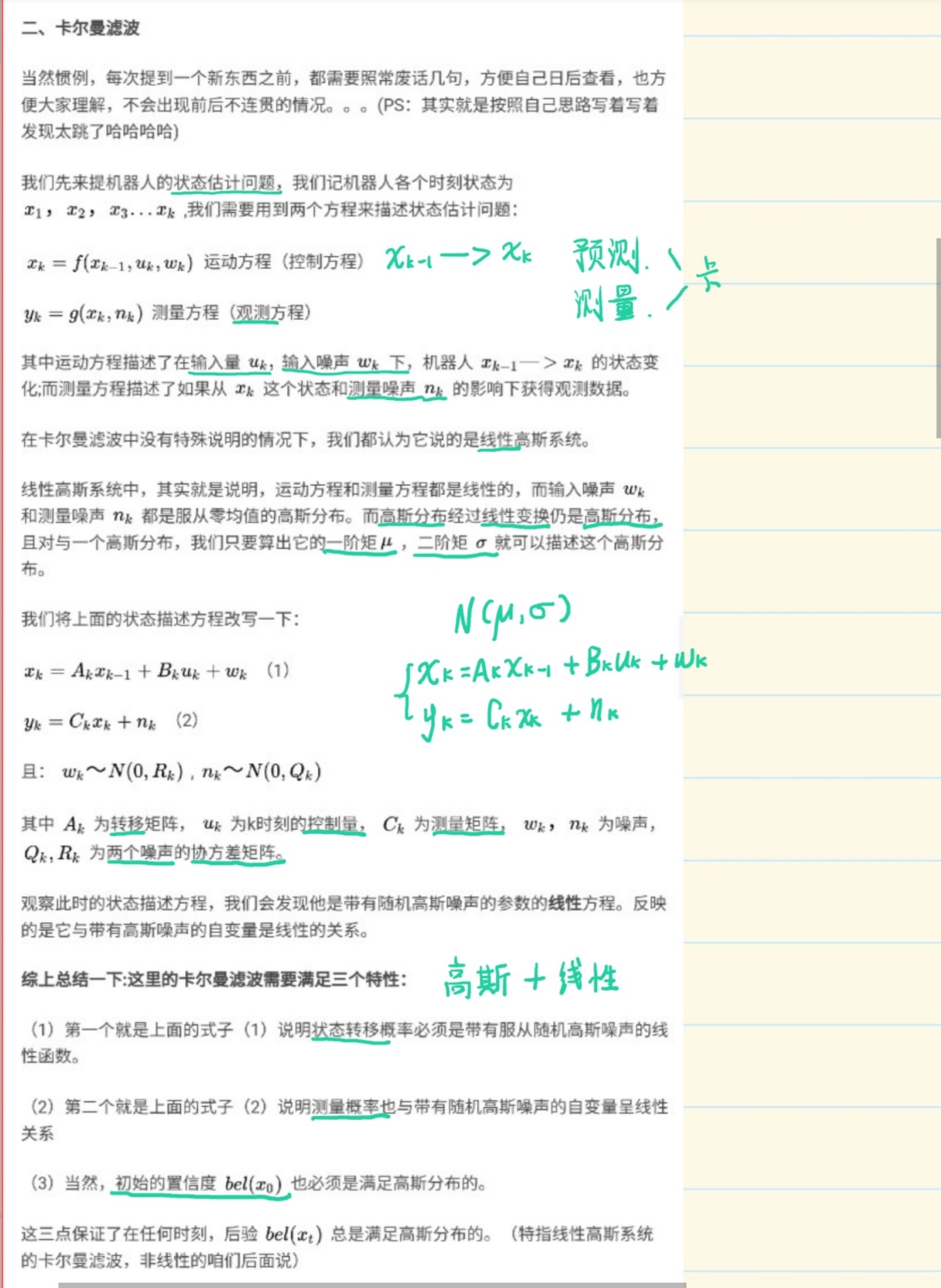
协方差

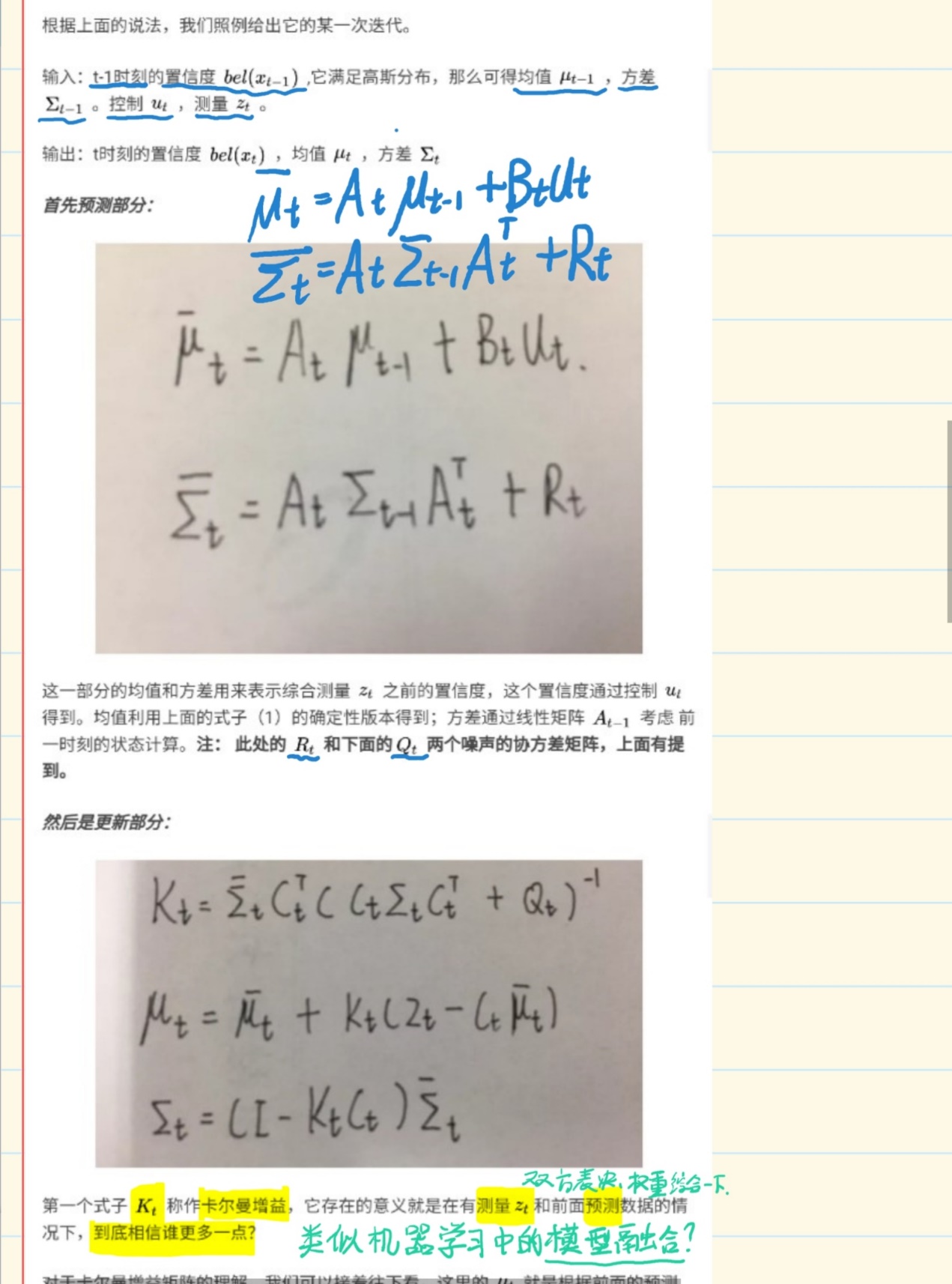
协方差是对两个随机变量X,Y联合分布线性相关程度的一种度量。两个随机变量越线性相关，协方差越大，完全线性无关，协方差为零。

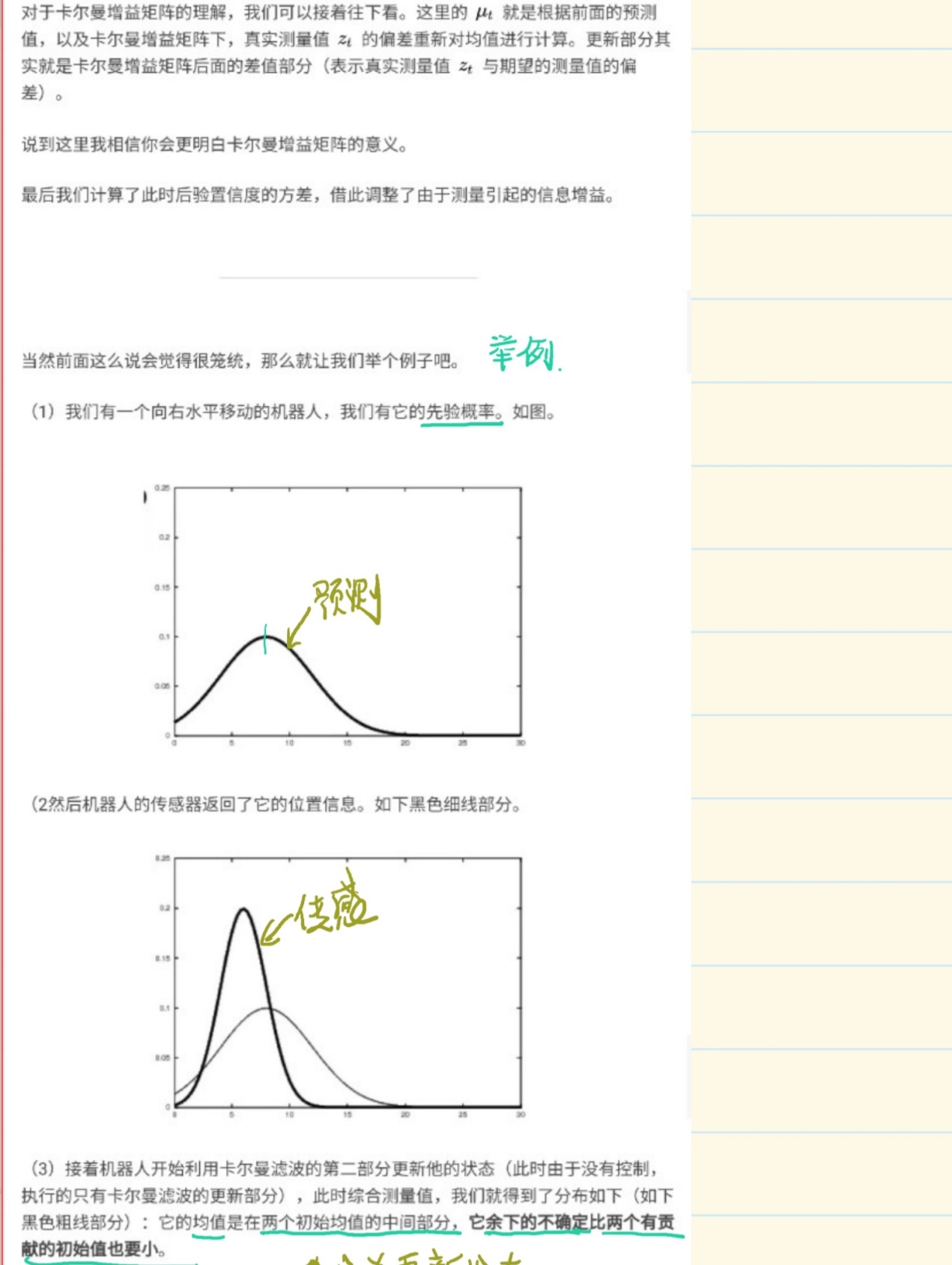
3.原理与计算实例：

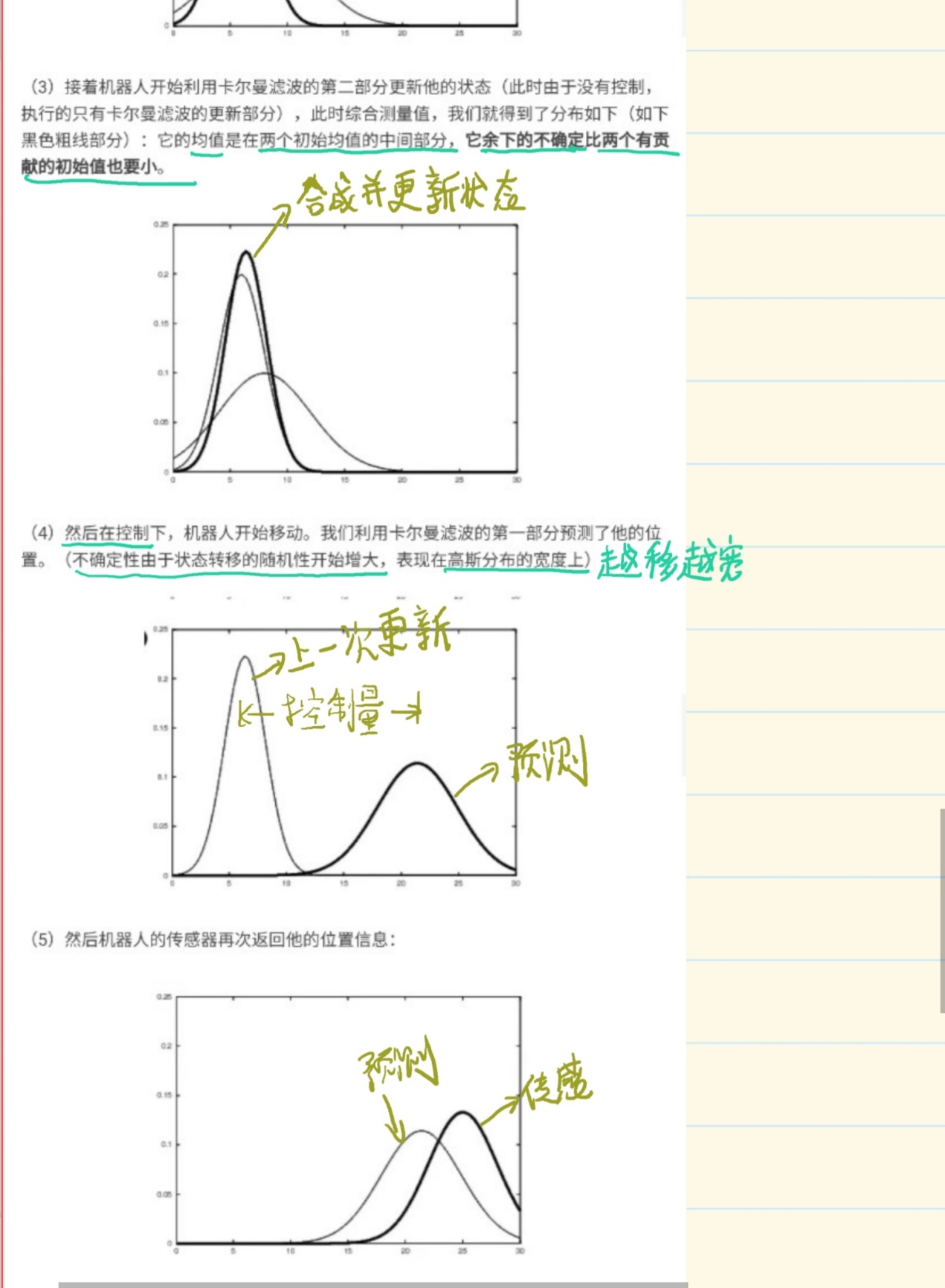
**（笔记）**

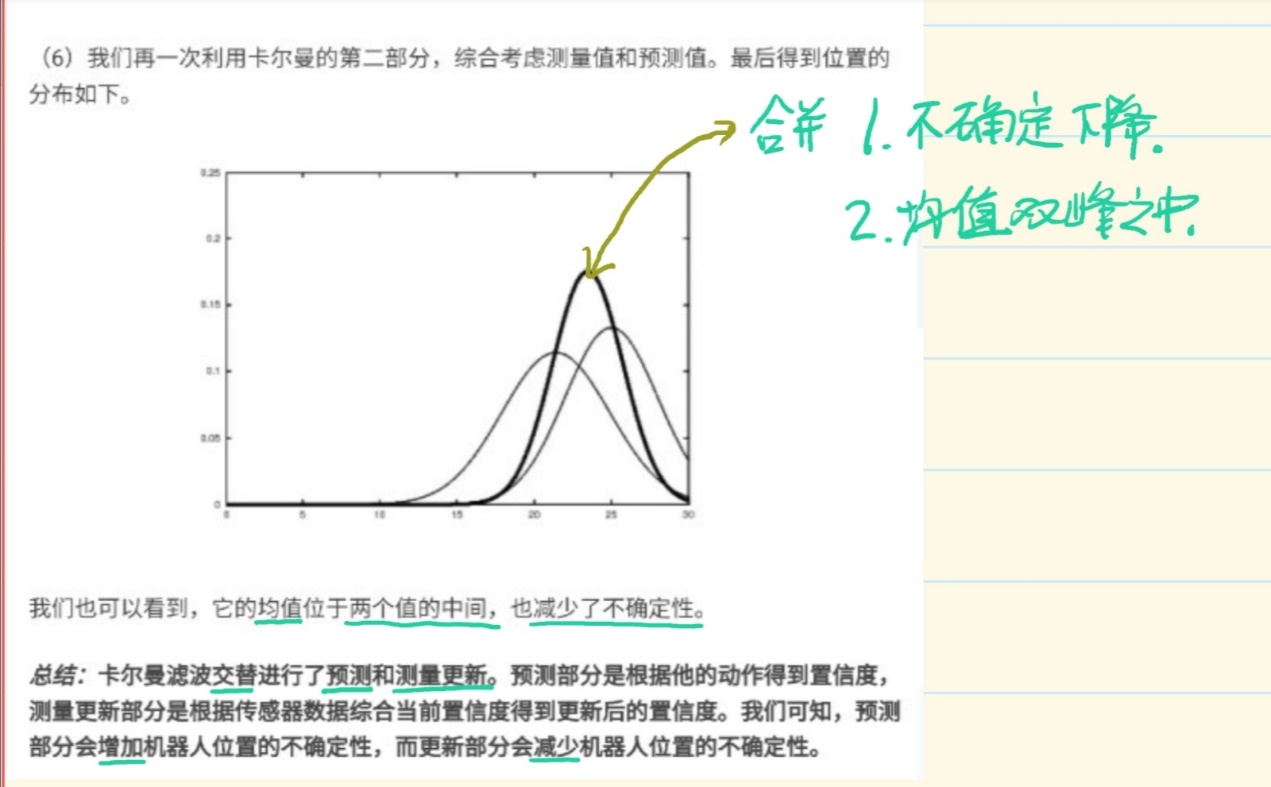












**（WIKI百科）卡尔曼滤波**（Kalman filter）是一种高效率的[递归滤波器](https://zh.wikipedia.org/wiki/%E9%80%92%E5%BD%92%E6%BB%A4%E6%B3%A2%E5%99%A8)（[自回归](https://zh.wikipedia.org/wiki/%E8%87%AA%E8%BF%B4%E6%AD%B8%E6%A8%A1%E5%9E%8B)滤波器），它能够从一系列的不完全及包含[杂讯](https://zh.wikipedia.org/wiki/%E9%9B%9C%E8%A8%8A_(%E9%80%9A%E8%A8%8A%E5%AD%B8))的[测量](https://zh.wikipedia.org/wiki/%E6%B5%8B%E9%87%8F)中，估计[动态系统](https://zh.wikipedia.org/wiki/%E5%8A%A8%E6%80%81%E7%B3%BB%E7%BB%9F)的状态。卡尔曼滤波会根据各测量量在不同时间下的值，考虑各时间下的[联合分布](https://zh.wikipedia.org/wiki/%E8%81%94%E5%90%88%E5%88%86%E5%B8%83)，再产生对未知变数的估计，因此会比只以单一测量量为基础的估计方式要准。卡尔曼滤波得名自主要贡献者之一的[鲁道夫·卡尔曼](https://zh.wikipedia.org/wiki/%E9%B2%81%E9%81%93%E5%A4%AB%C2%B7%E5%8D%A1%E5%B0%94%E6%9B%BC)。

卡尔曼滤波在技术领域有许多的应用。常见的有飞机及太空船的导引、导航及控制（英语：[guidance, navigation, and control](https://en.wikipedia.org/wiki/guidance,_navigation,_and_control)）[[1]](https://zh.wikipedia.org/#cite_note-ZarchanMusoff2000-1)。卡尔曼滤波也广为使用在[时间序列](https://zh.wikipedia.org/wiki/%E6%99%82%E9%96%93%E5%BA%8F%E5%88%97)的分析中，例如[信号处理](https://zh.wikipedia.org/wiki/%E4%BF%A1%E5%8F%B7%E5%A4%84%E7%90%86)及[计量经济学](https://zh.wikipedia.org/wiki/%E8%AE%A1%E9%87%8F%E7%BB%8F%E6%B5%8E%E5%AD%A6)中。卡尔曼滤波也是[机器人](https://zh.wikipedia.org/wiki/%E6%A9%9F%E5%99%A8%E4%BA%BA)运动规划及控制的重要主题之一，有时也包括在轨迹最佳化（英语：[trajectory optimization](https://en.wikipedia.org/wiki/trajectory_optimization)）。卡尔曼滤波也用在中轴神经系统运动控制的建模中。因为从给与运动命令到收到感觉神经的回授之间有时间差，使用卡尔曼滤波有助于建立符合实际的系统，估计运动系统的目前状态，并且更新命令[[2]](https://zh.wikipedia.org/#cite_note-2)。

卡尔曼滤波的演算法是二步骤的程序。在估计步骤中，卡尔曼滤波会产生有关目前状态的估计，其中也包括不确定性。只要观察到下一个量测（其中一定含有某种程度的误差，包括随机杂讯）。会透过[加权平均](https://zh.wikipedia.org/wiki/%E5%8A%A0%E6%AC%8A%E5%B9%B3%E5%9D%87)来更新估计值，而确定性越高的量测加权比重也越高。演算法是迭代的，可以在[实时控制系统](https://zh.wikipedia.org/wiki/%E5%AE%9E%E6%97%B6%E8%AE%A1%E7%AE%97)中执行，只需要目前的输入量测、以往的计算值以及其不确定性矩阵，不需要其他以往的资讯。

使用卡尔曼滤波不用假设误差是[正态分布](https://zh.wikipedia.org/wiki/%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83)[[3]](https://zh.wikipedia.org/#cite_note-3)，不过若所有的误差都是正态分布，卡尔曼滤波可以得到正确的条件机率估计。

也发展了一些扩展或是广义的卡尔曼滤波，例如运作在非线性糸统的扩展卡尔曼滤波（英语：[extended Kalman filter](https://en.wikipedia.org/wiki/extended_Kalman_filter)）及无迹卡尔曼滤波（unscented Kalman filter）。底层的模型类似[隐马尔可夫模型](https://zh.wikipedia.org/wiki/%E9%9A%90%E9%A9%AC%E5%B0%94%E5%8F%AF%E5%A4%AB%E6%A8%A1%E5%9E%8B)，不过[潜在变量](https://zh.wikipedia.org/wiki/%E6%BD%9C%E5%9C%A8%E5%8F%98%E9%87%8F)（英语：[latent variable](https://en.wikipedia.org/wiki/latent_variable)）的状态空间是连续的，而且所有潜在变量及可观测变数都是正态分布

**应用实例**

卡尔曼滤波的一个典型实例是从一组有限的，包含噪声的，通过对物体位置的观察序列（可能有偏差）预测出物体的位置的[坐标](https://zh.wikipedia.org/wiki/%E5%9D%90%E6%A0%87)及[速度](https://zh.wikipedia.org/wiki/%E9%80%9F%E5%BA%A6)。在很多工程应用（如[雷达](https://zh.wikipedia.org/wiki/%E9%9B%B7%E8%BE%BE)、[电脑视觉](https://zh.wikipedia.org/wiki/%E9%9B%BB%E8%85%A6%E8%A6%96%E8%A6%BA)）中都可以找到它的身影。同时，卡尔曼滤波也是[控制理论](https://zh.wikipedia.org/wiki/%E6%8E%A7%E5%88%B6%E7%90%86%E8%AE%BA)以及控制[系统工程](https://zh.wikipedia.org/wiki/%E7%B3%BB%E7%BB%9F%E5%B7%A5%E7%A8%8B)中的一个重要课题。

例如，对于雷达来说，人们感兴趣的是其能够跟踪目标。但目标的位置、速度、加速度的测量值往往在任何时候都有噪声。卡尔曼滤波利用目标的动态信息，设法去掉噪声的影响，得到一个关于目标位置的好的[估计](https://zh.wikipedia.org/w/index.php?title=%E4%BC%B0%E8%AE%A1&action=edit&redlink=1)。这个估计可以是对当前目标位置的估计（滤波），也可以是对于将来位置的估计（预测），也可以是对过去位置的估计（插值或平滑）。

**命名**

这种滤波方法以它的发明者[鲁道夫.E.卡尔曼](https://zh.wikipedia.org/wiki/%E9%B2%81%E9%81%93%E5%A4%AB%C2%B7%E5%8D%A1%E5%B0%94%E6%9B%BC)（Rudolph E. Kalman）命名，但是根据文献可知实际上Peter Swerling在更早之前就提出了一种类似的算法。

[斯坦利·施密特](https://zh.wikipedia.org/w/index.php?title=%E6%96%AF%E5%9D%A6%E5%88%A9%C2%B7%E6%96%BD%E5%AF%86%E7%89%B9&action=edit&redlink=1)（Stanley Schmidt）首次实现了卡尔曼滤波器。卡尔曼在NASA[埃姆斯](https://zh.wikipedia.org/wiki/%E5%9F%83%E5%A7%86%E6%96%AF)研究中心访问时，发现他的方法对于解决[阿波罗计划](https://zh.wikipedia.org/wiki/%E9%98%BF%E6%B3%A2%E7%BD%97%E8%AE%A1%E5%88%92)的轨道预测很有用，后来阿波罗飞船的导航电脑便使用了这种滤波器。关于这种滤波器的论文由Swerling（1958）、Kalman (1960)与Kalman and Bucy（1961）发表。

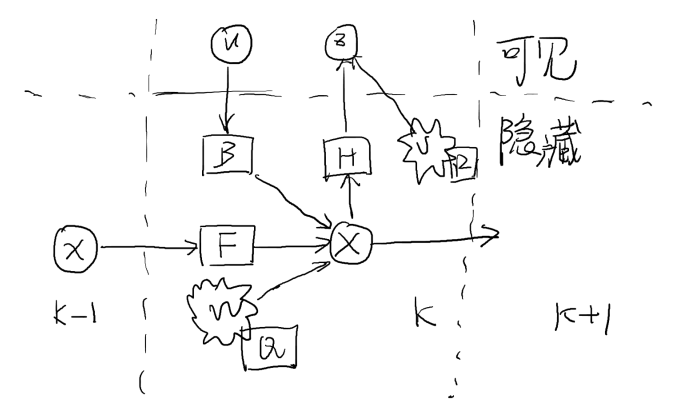
目前，卡尔曼滤波已经有很多不同的实现。卡尔曼最初提出的形式现在一般称为简单卡尔曼滤波器。除此以外，还有施密特扩展滤波器、信息滤波器以及很多Bierman, Thornton开发的平方根滤波器的变种。也许最常见的卡尔曼滤波器是[锁相环](https://zh.wikipedia.org/wiki/%E9%94%81%E7%9B%B8%E7%8E%AF)，它在收音机、计算机和几乎任何视频或通讯设备中广泛存在。

 以下的讨论需要[线性代数](https://zh.wikipedia.org/wiki/%E7%BA%BF%E6%80%A7%E4%BB%A3%E6%95%B0)以及[概率论](https://zh.wikipedia.org/wiki/%E6%A6%82%E7%8E%87%E8%AE%BA)的一般知识。

**基本动态系统模型**

卡尔曼滤波建立在[线性代数](https://zh.wikipedia.org/wiki/%E7%BA%BF%E6%80%A7%E4%BB%A3%E6%95%B0)和[隐马尔可夫模型](https://zh.wikipedia.org/wiki/%E9%9A%90%E9%A9%AC%E5%B0%94%E5%8F%AF%E5%A4%AB%E6%A8%A1%E5%9E%8B)（hidden Markov model）上。其基本动态系统可以用一个[马尔可夫链](https://zh.wikipedia.org/wiki/%E9%A9%AC%E5%B0%94%E5%8F%AF%E5%A4%AB%E9%93%BE)表示，该马尔可夫链建立在一个被高斯[噪声](https://zh.wikipedia.org/wiki/%E5%99%AA%E5%A3%B0)（即正态分布的噪声）干扰的[线性算子](https://zh.wikipedia.org/wiki/%E7%BA%BF%E6%80%A7%E7%AE%97%E5%AD%90)上的。系统的[状态](https://zh.wikipedia.org/wiki/%E7%8A%B6%E6%80%81%E7%A9%BA%E9%97%B4)可以用一个元素为实数的[向量](https://zh.wikipedia.org/wiki/%E5%90%91%E9%87%8F)表示。随着[离散时间](https://zh.wikipedia.org/wiki/%E7%A6%BB%E6%95%A3%E6%97%B6%E9%97%B4%E4%BF%A1%E5%8F%B7)的每一个增加，这个[线性算子](https://zh.wikipedia.org/wiki/%E7%BA%BF%E6%80%A7%E7%AE%97%E5%AD%90)就会作用在当前[状态](https://zh.wikipedia.org/wiki/%E7%8A%B6%E6%80%81)上，产生一个新的状态，并也会带入一些噪声，同时系统的一些已知的控制器的控制信息也会被加入。同时，另一个受噪声干扰的线性算子产生出这些隐含状态的可见输出。

为了从一系列有噪声的观察数据中用卡尔曼滤波器估计出被观察过程的内部状态，必须把这个过程在卡尔曼滤波的框架下建立模型。也就是说对于每一步*k*，定义[矩阵](https://zh.wikipedia.org/wiki/%E7%9F%A9%E9%98%B5)**F***k*, **H***k*, **Q***k*, **R***k*，有时也需要定义**B***k*，如下。



卡尔曼滤波器的模型。圆圈代表[向量](https://zh.wikipedia.org/wiki/%E5%90%91%E9%87%8F)，方块代表[矩阵](https://zh.wikipedia.org/wiki/%E7%9F%A9%E9%98%B5)，星号代表[高斯噪声](https://zh.wikipedia.org/wiki/%E9%AB%98%E6%96%AF%E5%99%AA%E5%A3%B0)，其[协方差矩阵](https://zh.wikipedia.org/wiki/%E5%8D%8F%E6%96%B9%E5%B7%AE%E7%9F%A9%E9%98%B5)在右下方标出。

卡尔曼滤波模型假设*k*时刻的真实状态是从（*k* − 1）时刻的状态演化而来，符合下式：



其中

* **F***k*是作用在**x***k*−1上的状态变换模型（/矩阵/矢量）。
* **B***k*是作用在控制器向量**u***k*上的输入－控制模型。
* **w***k*是过程噪声，并假定其符合均值为零，[协方差矩阵](https://zh.wikipedia.org/wiki/%E5%8D%8F%E6%96%B9%E5%B7%AE%E7%9F%A9%E9%98%B5)为**Q***k*的[多元正态分布](https://zh.wikipedia.org/wiki/%E5%A4%9A%E5%85%83%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83)。

时刻*k*，对真实状态**x***k*的一个测量**z***k*满足下式：



其中**H***k*是观测模型，它把真实状态空间映射成观测空间，**v***k*是观测噪声，其均值为零，协方差矩阵为**R***k*,且服从[正态分布](https://zh.wikipedia.org/wiki/%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83)。

初始状态以及每一时刻的噪声{**x**0, **w**1, ..., **w***k*, **v**1 ... **v***k*}都认为是互相[独立](https://zh.wikipedia.org/w/index.php?title=%E7%BB%9F%E8%AE%A1%E5%AD%A6%E7%8B%AC%E7%AB%8B&action=edit&redlink=1)的。

实际上，很多真实世界的动态系统都并不确切的符合这个模型；但是由于卡尔曼滤波器被设计在有噪声的情况下工作，一个近似的符合已经可以使这个滤波器非常有用了。更多其它更复杂的卡尔曼滤波器的变种，在下边讨论中有描述。

**卡尔曼滤波器**

卡尔曼滤波是一种[递归](https://zh.wikipedia.org/wiki/%E9%80%92%E5%BD%92)的估计，即只要获知上一时刻状态的估计值以及当前状态的观测值就可以计算出当前状态的估计值，因此不需要记录观测或者估计的历史信息。卡尔曼滤波器与大多数滤波器不同之处，在于它是一种纯粹的[时域](https://zh.wikipedia.org/wiki/%E6%99%82%E5%9F%9F)滤波器，它不需要像[低通滤波器](https://zh.wikipedia.org/wiki/%E4%BD%8E%E9%80%9A%E6%BB%A4%E6%B3%A2%E5%99%A8)等[频域](https://zh.wikipedia.org/wiki/%E9%A2%91%E5%9F%9F)滤波器那样，需要在频域设计再转换到时域实现。

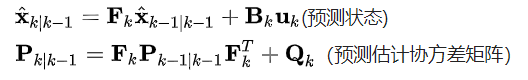
卡尔曼滤波器的状态由以下两个变量表示：

，在时刻*k*的状态的估计；

，后验估计误差协方差矩阵，度量估计值的精确程度。

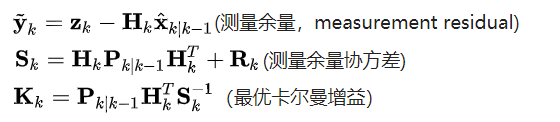
卡尔曼滤波器的操作包括两个阶段：**预测**与**更新**。在预测阶段，滤波器使用上一状态的估计，做出对当前状态的估计。在更新阶段，滤波器利用对当前状态的观测值优化在预测阶段获得的预测值，以获得一个更精确的新估计值。

**预测**

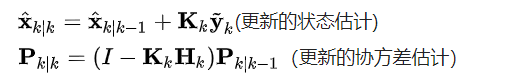
可参考：<http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf>

可参考：<http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf>

**更新**



然后用它们来更新滤波器变量**x**与**P**：



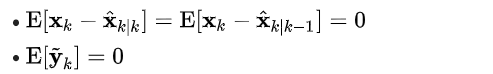
使用上述公式计算仅在最优卡尔曼增益的时候有效。使用其他增益的话，公式要复杂一些，请参见[推导](https://zh.wikipedia.org/wiki/%E5%8D%A1%E5%B0%94%E6%9B%BC%E6%BB%A4%E6%B3%A2#%E6%8E%A8%E5%AF%BC)。

**不变量（Invariant）**

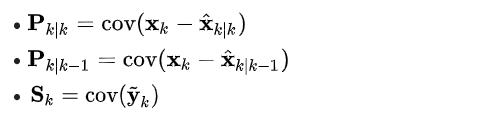
如果模型准确，而且

与

的值准确的反映了最初状态的分布，那么以下不变量就保持不变：所有估计的误差均值为零



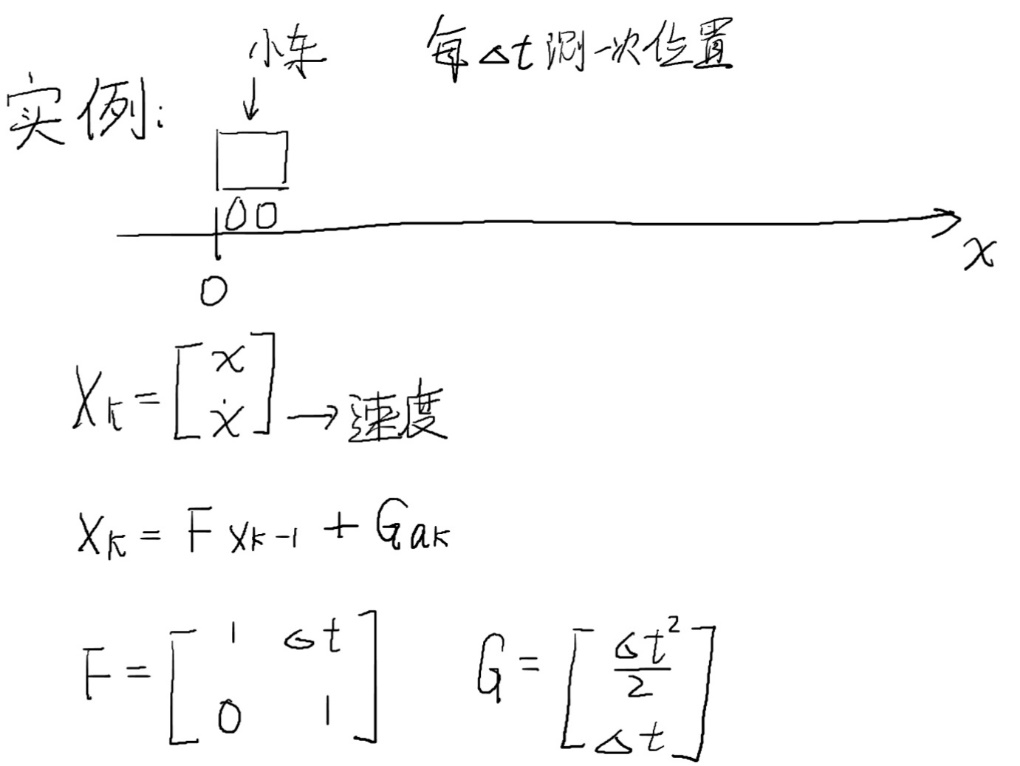
且[协方差矩阵](https://zh.wikipedia.org/wiki/%E5%8D%8F%E6%96%B9%E5%B7%AE%E7%9F%A9%E9%98%B5)准确的反映了估计的协方差：



请注意，其中表示a的期望值, 。

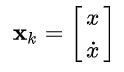
**实例**

考虑在无摩擦的、无限长的直轨道上的一辆车。该车最初停在位置0处，但时不时受到随机的冲击。每隔*Δt*秒即测量车的位置，但是这个测量是非精确的；想建立一个关于其位置以及[速度](https://zh.wikipedia.org/wiki/%E9%80%9F%E5%BA%A6)的模型。来看如何推导出这个模型以及如何从这个模型得到卡尔曼滤波器。



因为车上无动力，所以可以忽略掉**B***k*和**u***k*。由于**F**、**H**、**R**和**Q**是常数，所以时间下标可以去掉。

车的位置以及速度（或者更加一般的，一个粒子的运动状态）可以被线性状态空间描述如下：

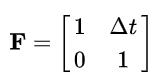


其中是速度，也就是位置对于时间的导数。

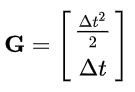
假设在（*k* − 1）时刻与*k*时刻之间，车受到*ak*的加速度，其符合均值为0，标准差为*σa*的[正态分布](https://zh.wikipedia.org/wiki/%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83)。根据[牛顿运动定律](https://zh.wikipedia.org/wiki/%E7%89%9B%E9%A1%BF%E8%BF%90%E5%8A%A8%E5%AE%9A%E5%BE%8B)，可以推出



其中



且



可以发现



（因为*σa*是一个标量）。

在每一时刻，对其位置进行测量，测量受到噪声干扰。假设噪声服从正态分布，均值为0，标准差为*σ*z。



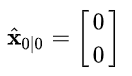
其中



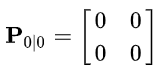
且



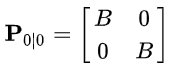
如果知道足够精确的车最初的位置，那么可以初始化



并且，若让滤波器知道确切的初始位置，可给出一个协方差矩阵：



如果不确切的知道最初的位置与速度，那么协方差矩阵可以初始化为一个对角线元素是*B*的矩阵，*B*取一个合适的比较大的数。



此时，与使用模型中已有信息相比，滤波器更倾向于使用初次测量值的信息。

**推导**

**推导后验协方差矩阵**

按照上边的定义，从误差协方差开始推导如下：



代入



再代入 



 与**Z**k



整理误差向量，得

因为测量误差**v***k*与其他项是非相关的，因此有



利用[协方差矩阵](https://zh.wikipedia.org/wiki/%E5%8D%8F%E6%96%B9%E5%B7%AE%E7%9F%A9%E9%98%B5)的性质，此式可以写作



使用不变量**P***k*|*k*-1以及**R***k*的定义这一项可以写作 ：



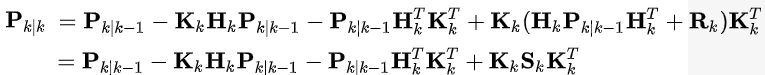
这一公式对于任何卡尔曼增益**K***k*都成立。如果**K***k*是最优卡尔曼增益，则可以进一步简化，请见下文。

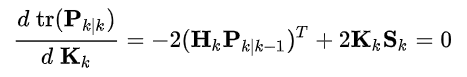
**最优卡尔曼增益的推导**

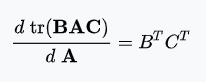
卡尔曼滤波器是一个[最小均方误差](https://zh.wikipedia.org/wiki/%E6%9C%80%E5%B0%8F%E5%9D%87%E6%96%B9%E8%AF%AF%E5%B7%AE)估计器，后验状态误差估计（英文：*a posteriori*state estimate）是



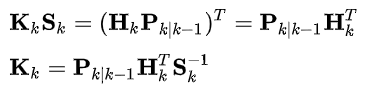
最小化这个矢量幅度平方的期望值，，这等同于最小化后验估计协方差矩阵**P***k*|*k*的[迹](https://zh.wikipedia.org/wiki/%E8%BF%B9%E6%95%B0)（trace）。将上面方程中的项展开、抵消，得到：

当[矩阵导数](https://zh.wikipedia.org/w/index.php?title=%E7%9F%A9%E9%98%B5%E5%AF%BC%E6%95%B0&action=edit&redlink=1)是0的时候得到**P***k*|*k*的[迹](https://zh.wikipedia.org/wiki/%E8%BF%B9%E6%95%B0)（trace）的最小值：

此处须用到一个常用的式子，如下：



从这个方程解出卡尔曼增益**K***k*：



这个增益称为*最优卡尔曼增益*，在使用时得到[最小均方误差](https://zh.wikipedia.org/wiki/%E6%9C%80%E5%B0%8F%E5%9D%87%E6%96%B9%E8%AF%AF%E5%B7%AE)。

**后验误差协方差公式的化简**

在卡尔曼增益等于上面导出的最优值时，计算后验协方差的公式可以进行简化。在卡尔曼增益公式两侧的右边都乘以**S***k***K***kT*得到



根据上面后验误差协方差展开公式，



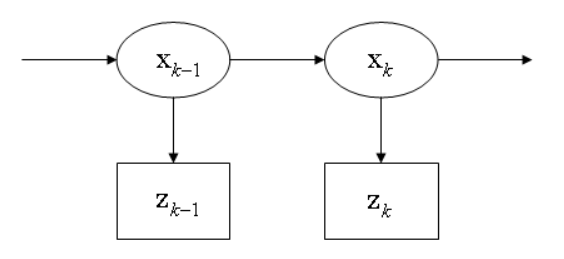
最后两项可以抵消，得到



这个公式的计算比较简单，所以实际中总是使用这个公式，但是需注意这公式仅在使用最优卡尔曼增益的时候它才成立。如果算术精度总是很低而导致[数值稳定性](https://zh.wikipedia.org/wiki/%E6%95%B0%E5%80%BC%E7%A8%B3%E5%AE%9A%E6%80%A7)出现问题，或者特意使用非最优卡尔曼增益，那么就不能使用这个简化；必须使用上面导出的后验误差协方差公式。

**与递归贝叶斯估计之间的关系**

假设真正的状态是无法观察的[马尔可夫过程](https://zh.wikipedia.org/wiki/%E9%A9%AC%E5%B0%94%E5%8F%AF%E5%A4%AB%E8%BF%87%E7%A8%8B)，测量结果是从隐性马尔可夫模型观察到的状态。



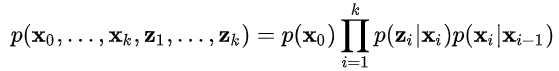
根据马尔可夫假设，真正的状态仅受最近一个状态影响而与其它以前状态无关。



与此类似，在时刻*k*测量只与当前状态有关而与其它状态无关。



根据这些假设，隐性马尔可夫模型所有状态的概率分布可以简化为：



然而，当卡尔曼滤波器用来估计状态***x***时，感兴趣的机率分布，是基于目前为止所有个测量值来得到的当前状态之机率分布



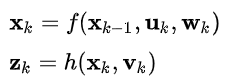
**信息滤波器**

**非线性滤波器**

基本卡尔曼滤波器（The basic Kalman filter）是限制在线性的假设之下。然而，大部份非平凡的（non-trivial）的系统都是非线性系统。其中的“非线性性质”（non-linearity）可能是伴随存在过程模型（process model）中或观测模型（observation model）中，或者两者兼有之。

**扩展卡尔曼滤波器**

在扩展卡尔曼滤波器（Extended Kalman Filter，简称EKF）中状态转换和观测模型不需要是状态的线性函数，可替换为（可微的）函数。

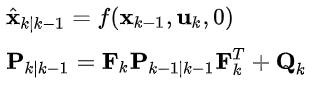


函数*f*可以用来从过去的估计值中计算预测的状态，相似的，函数*h*可以用来以预测的状态计算预测的测量值。然而*f*和*h*不能直接的应用在协方差中，取而代之的是计算偏导矩阵（[Jacobian](https://zh.wikipedia.org/wiki/%E9%9B%85%E5%8F%AF%E6%AF%94%E7%9F%A9%E9%98%B5)）。

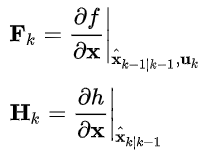
在每一步中使用当前的估计状态计算Jacobian矩阵，这几个矩阵可以用在卡尔曼滤波器的方程中。这个过程，实质上将非线性的函数在当前估计值处线性化了。

这样一来，卡尔曼滤波器的等式为：

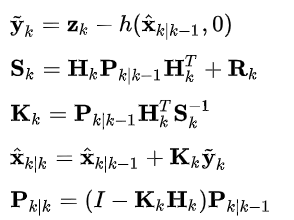
**预测**



**使用Jacobians矩阵更新模型**



**更新**



**预测**

如同扩展卡尔曼滤波器（EKF）一样, UKF的预测过程可以独立于UKF的更新过程之外，与一个线性的（或者确实是扩展卡尔曼滤波器的）更新过程合并来使用；或者，UKF的预测过程与更新过程在上述中地位互换亦可。

**应用**

[自动驾驶仪](https://zh.wikipedia.org/wiki/%E8%87%AA%E5%8A%A8%E9%A9%BE%E9%A9%B6%E4%BB%AA)

[动态定位系统](https://zh.wikipedia.org/w/index.php?title=%E5%8A%A8%E6%80%81%E5%AE%9A%E4%BD%8D%E7%B3%BB%E7%BB%9F&action=edit&redlink=1)（英语：[Dynamic positioning](https://en.wikipedia.org/wiki/Dynamic_positioning)）

[经济学](https://zh.wikipedia.org/wiki/%E7%BB%8F%E6%B5%8E%E5%AD%A6)，特别是[宏观经济学](https://zh.wikipedia.org/wiki/%E5%AE%8F%E8%A7%82%E7%BB%8F%E6%B5%8E%E5%AD%A6)，[时间序列](https://zh.wikipedia.org/wiki/%E6%97%B6%E9%97%B4%E5%BA%8F%E5%88%97)模型，以及[计量经济学](https://zh.wikipedia.org/wiki/%E8%AE%A1%E9%87%8F%E7%BB%8F%E6%B5%8E%E5%AD%A6)

[惯性导航系统](https://zh.wikipedia.org/wiki/%E6%83%AF%E6%80%A7%E5%AF%BC%E8%88%AA%E7%B3%BB%E7%BB%9F)

[雷达跟踪器](https://zh.wikipedia.org/w/index.php?title=%E9%9B%B7%E8%BE%BE%E8%B7%9F%E8%B8%AA%E5%99%A8&action=edit&redlink=1)（英语：[Radar tracker](https://en.wikipedia.org/wiki/Radar_tracker)）

[卫星导航系统](https://zh.wikipedia.org/wiki/%E5%8D%AB%E6%98%9F%E5%AF%BC%E8%88%AA%E7%B3%BB%E7%BB%9F)

**外部链接**

* [An Introduction to the Kalman Filter](http://www.cs.unc.edu/~tracker/media/pdf/SIGGRAPH2001_CoursePack_08.pdf), SIGGRAPH 2001 Course, Greg Welch and Gary Bishop
* [Kalman filtering chapter](https://web.archive.org/web/20060420154727/http:/www.cs.unc.edu/~welch/media/pdf/maybeck_ch1.pdf) from *Stochastic Models, Estimation*, by Peter Maybeck
* [Kalman Filter](http://www.cs.unc.edu/~welch/kalman/) webpage, with lots of links
* [Kalman Filtering](https://web.archive.org/web/20130623214223/http:/www.innovatia.com/software/papers/kalman.htm)
* [The unscented Kalman filter for nonlinear estimation](https://ieeexplore.ieee.org/document/882463)
* [*Kalman Filters*, thorough introduction to several types, together with applications to*Robot Localization*](http://www.negenborn.net/kal_loc/)

**参考文献**

* Gelb A., editor. Applied optimal estimation. MIT Press, 1974.
* Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems, *Transactions of the ASME - Journal of Basic Engineering* Vol. 82: pp. 35-45 (1960)
* Kalman, R. E., Bucy R. S., New Results in Linear Filtering and Prediction Theory, *Transactions of the ASME - Journal of Basic Engineering* Vol. 83: pp. 95-107 (1961)
* [JU97] Julier, Simon J. and Jeffery K. Uhlmann. A New Extension of the Kalman Filter to nonlinear Systems. In The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing,Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management II, SPIE, 1997.
* Harvey, A.C. Forecasting, Structural Time Series Models and the Kalman Filter. Cambridge University Press, Cambridge, 1989.

1. [**^**](https://zh.wikipedia.org/#cite_ref-ZarchanMusoff2000_1-0) Paul Zarchan; Howard Musoff. [**Fundamentals of Kalman Filtering: A Practical Approach**](https://books.google.com/books?id=AQxRAAAAMAAJ). American Institute of Aeronautics and Astronautics, Incorporated. 2000. [**ISBN 978-1-56347-455-2**](https://zh.wikipedia.org/wiki/Special:%E7%BD%91%E7%BB%9C%E4%B9%A6%E6%BA%90/978-1-56347-455-2).
2. [**^**](https://zh.wikipedia.org/#cite_ref-2) Wolpert, Daniel; Ghahramani, Zoubin. Computational principles of movement neuroscience. Nature Neuroscience. 2000, **3**: 1212–7. [**PMID 11127840**](https://www.ncbi.nlm.nih.gov/pubmed/11127840). [**doi:10.1038/81497**](https://dx.doi.org/10.1038%2F81497).
3. [**^**](https://zh.wikipedia.org/#cite_ref-3) Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering. 1960, **82**: 35. [**doi:10.1115/1.3662552**](https://dx.doi.org/10.1115%2F1.3662552).