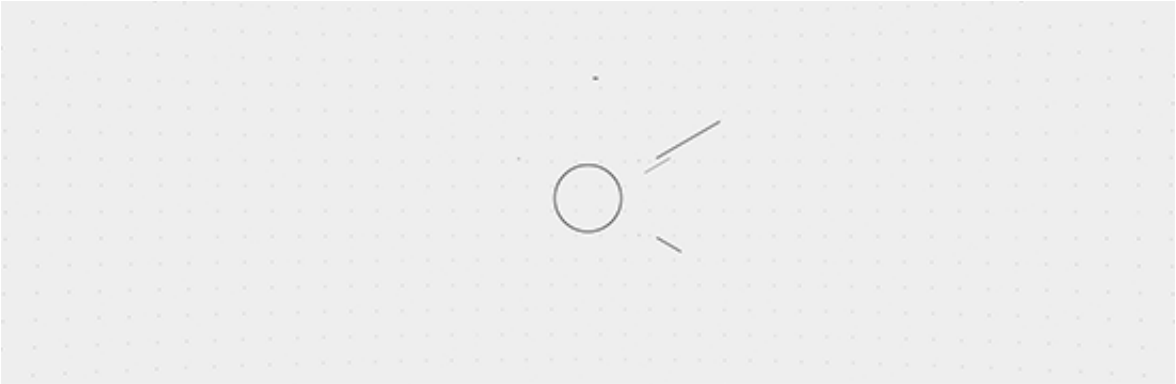
一文图解卡尔曼滤波(Kalman Filter)



1 背景

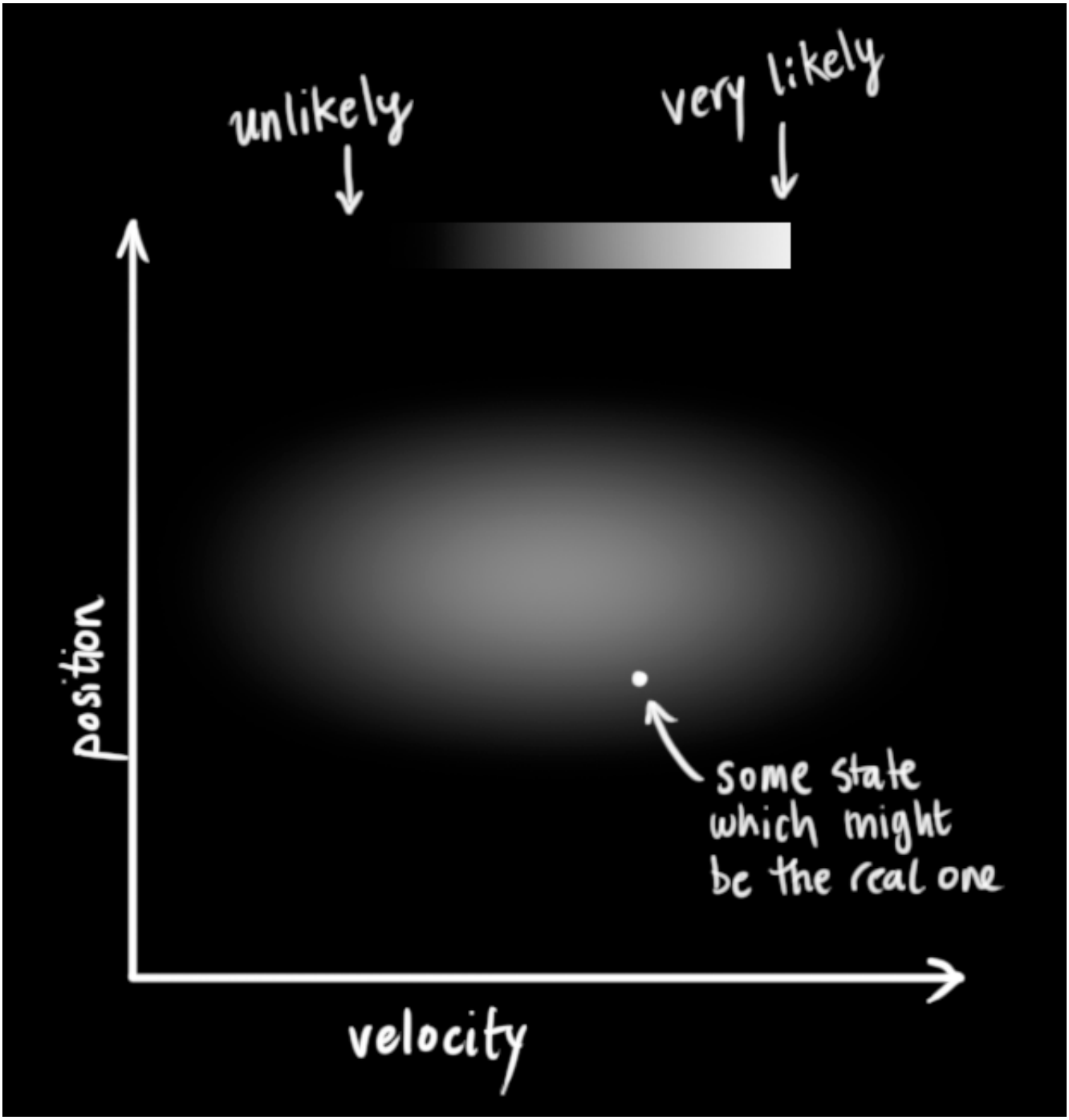
“一位专业课的教授给我们上课的时候，曾谈到：filtering is weighting（滤波即加权）。**滤波的作用就是给不同的信号分量不同的权重。**最简单的loss pass filter， 就是直接把低频的信号给1权重，而给高频部分0权重。对于更复杂的滤波，比如维纳滤波，则要根据信号的统计知识来设计权重。
从统计信号处理的角度，降噪可以看成滤波的一种。降噪的目的在于突出信号本身而抑制噪声影响。从这个角度，降噪就是给信号一个高的权重而给噪声一个低的权重。维纳滤波就是一个典型的降噪滤波器。”
Kalman Filter 算法，是一种**递推预测滤波算法**，算法中涉及到滤波，也涉及到对下一时刻数据的预测。Kalman Filter 由一系列递归数学公式描述。它提供了一种高效可计算的方法来估计过程的状态，并使估计均方误差最小。卡尔曼滤波器应用广泛且功能强大：它可以估计信号的过去和当前状态，甚至能估计将来的状态，即使并不知道模型的确切性质。
Kalman Filter 也可以被认为是一种数据融合算法（Data fusion algorithm），已有50多年的历史，是当今使用最重要和最常见的数据融合算法之一。Kalman Filter 的巨大成功归功于其小的计算需求，优雅的递归属性以及作为具有高斯误差统计的一维线性系统的最优估计器的状态。
Kalman Filter 只能减小均值为0的测量噪声带来的影响。只要噪声期望为0，那么不管方差多大，只要迭代次数足够多，那效果都很好。反之，噪声期望

不为0，那么估计值就还是与实际值有偏差[3]。

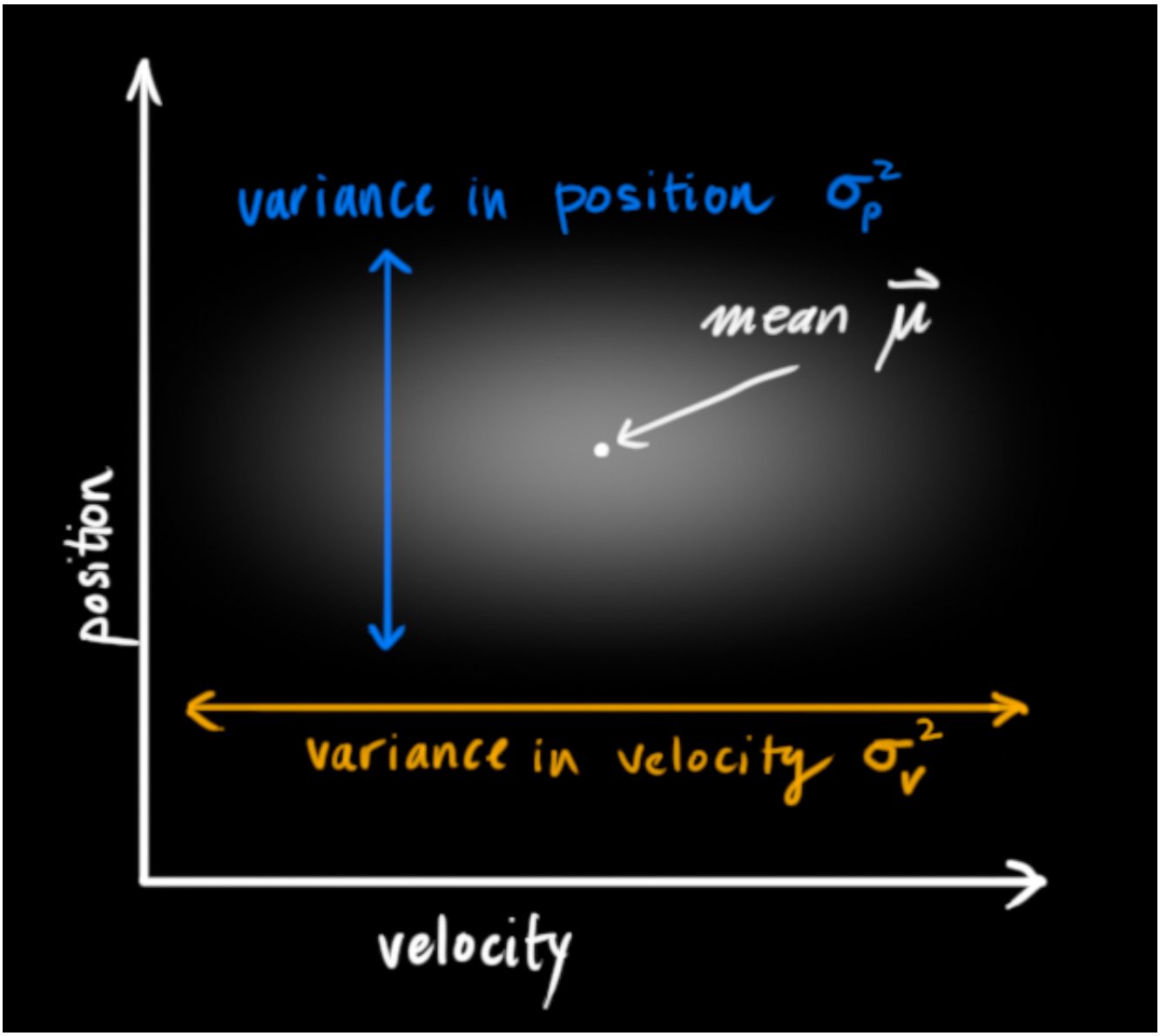
我们会有一个疑问：**卡尔曼滤波到底是如何解决实际问题的呢?**

我们以机器人为例介绍卡尔曼滤波的原理，我们的任务是要预测机器人的**状态**，包括位置与速度，即可表示为：

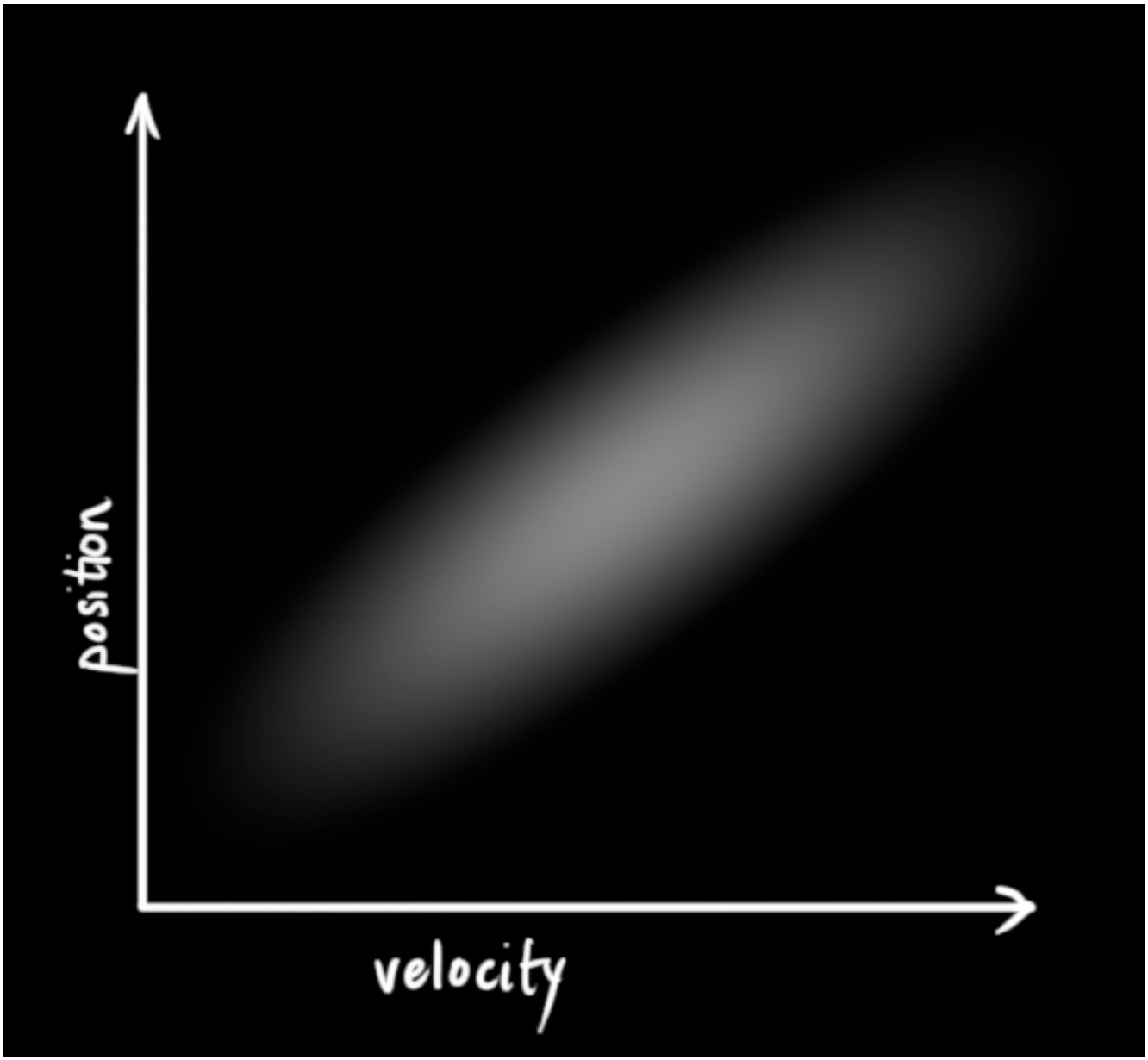
某个时刻，我们不知道真实的位置与速度到底是多少，二者存在一个所有可能性的组合，大致如下图所示。



卡尔曼滤波假设状态所有的变量都是随机的且都服从高斯分布，每个变量都有其对应的均值以及方差（它代表了不确定性）。

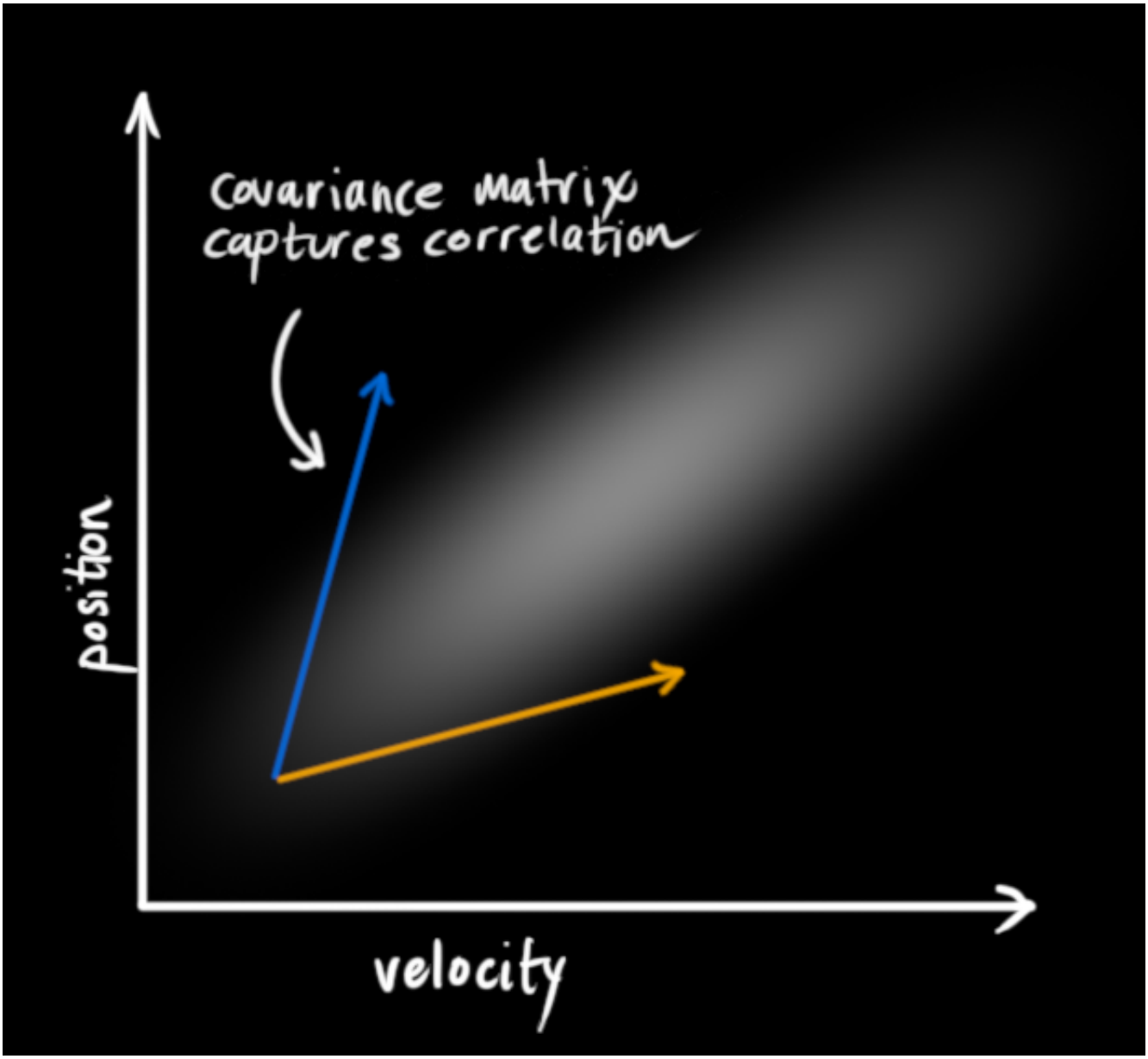


在上图中，位置和速度是不相关（**uncorrelated**）的，这意味着某个变量的状态不会告诉你其他变量的状态是怎样的。即，我们虽然知道现在的速度，但无法从现在的速度推测出现在的位置。但实际上并非如此，我们知道速度和位置是有关系的（**correlated**），这样一来二者之间的组合关系变成了如下图所示的情况。



这种情况是很容易发生的，例如，如果速度很快，我们可能会走得更远，所以我们的位置会更大。如果我们走得很慢，我们就不会走得太远。
这种状态变量之间的关系很重要，因为它可以为我们提供**更多信息**：One measurement tells us something about what the others could be。这就是卡尔曼滤波器的目标，我们希望从不确定的测量中尽可能多地获取信息！
这种状态量的相关性可以由协方差矩阵表示。简而言之，矩阵的每个元素是第i个状态变量和第j个状态变量之间的相关度。（显然地可以知道协方差矩阵是对称的，这意味着交换i和j都没关系）。协方差矩阵通常标记为“ ”，因

此我们将它们的元素称为“”。



2 状态预测

问题的矩阵形式表示

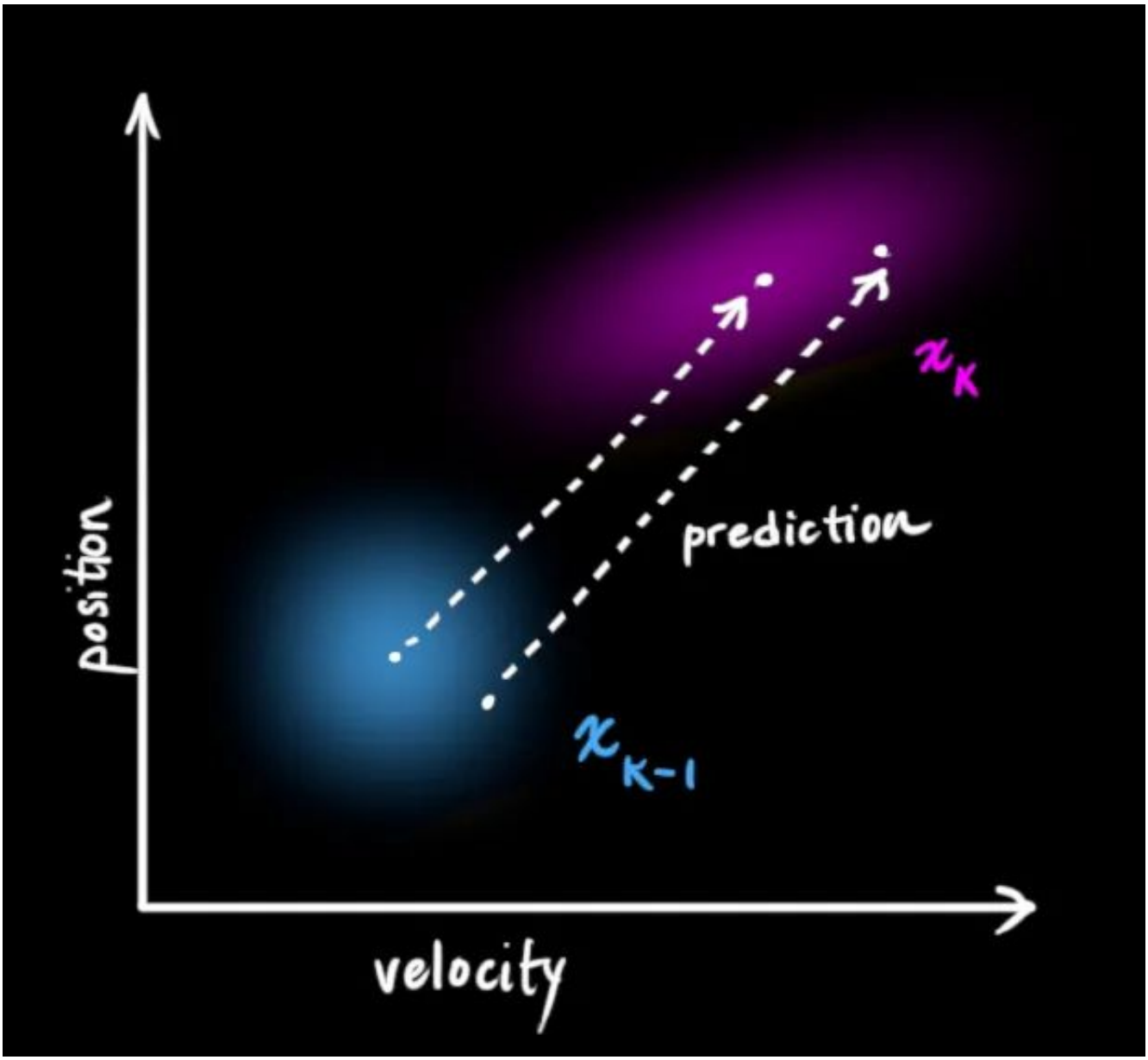
我们把状态建模成高斯分布（Gaussian blob，由于二维高斯分布长得像一个个小泡泡，之所以长这个样子，可参考链接[**2**]）。我们需要求解/估计在时间时刻的两个信息：1. 最优估计以及它的协方差矩阵，我们可以写成下面矩

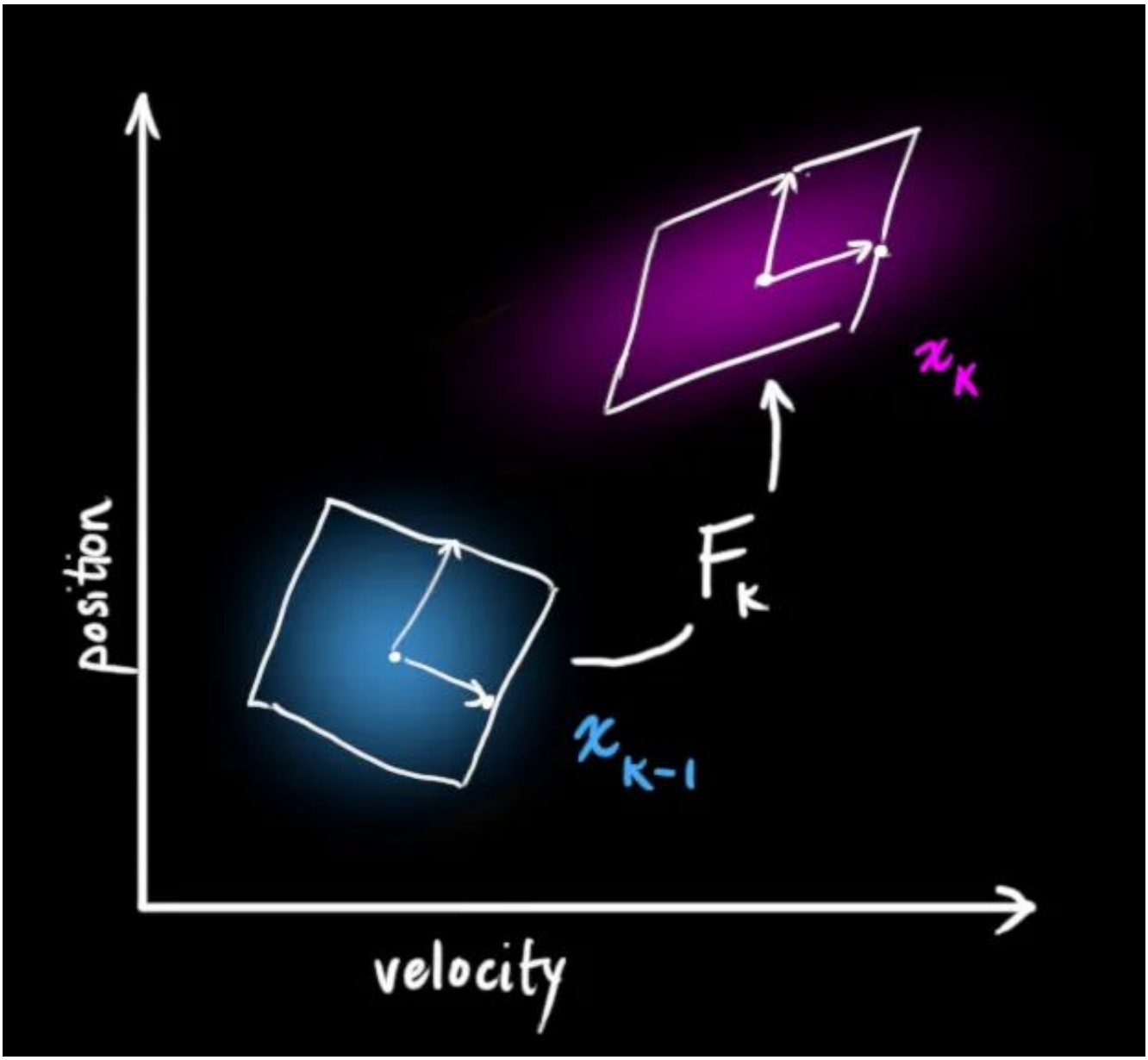
阵形式：

$$\begin{aligned}\hat{\mathbf{x}}_k &= \begin{bmatrix}\text{position}\\ \text{velocity}\end{bmatrix}\\ \mathbf{P}_k &= \begin{bmatrix}\Sigma_{pp} & \Sigma_{pv}\\ \Sigma_{vp} & \Sigma_{vv}\end{bmatrix}\end{aligned} \tag{1}$$

当然，这里我们仅使用位置和速度，但是请记住状态可以包含任意数量的变量，并且可以表示所需的任何变量。

接下来，我们需要某种方式来查看当前状态（时刻）并**预测**在时刻处的状态。请记住，我们不知道哪个状态是“真实”状态，但是这里提到的**预测**（prediction）并不在乎这些。


position
velocity
prediction
x_K
x_{K-1}



那么问题来了，应该如何使用上述矩阵来预测下一时刻的位置和速度呢？为了阐述这个过程，我们使用了一个非常基础的运动学公式（初中物理中就学过）进行描述：

$$\hat{\mathbf{x}}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} \tag{2}$$

$$= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \tag{3}$$

现在我们有了一个**预测矩阵**或者叫做**状态转移矩阵**，该矩阵可以帮助我们计算下一个时刻的状态。但我们仍然不知道如何更新状态的协方差矩阵，其实过程也是很简单，如果我们将分布中的每个点乘以矩阵，那么其协方差矩阵

会发生什么?

$$\begin{aligned}\hat{\mathbf{x}}_k &= \mathbf{F}_k\hat{\mathbf{x}}_{k-1} \\ \mathbf{P}_k &= \mathbf{F_k}\mathbf{P}_{k-1}\mathbf{F}_k^T\end{aligned} \tag{5}$$

不过我们并没有考虑到所有的影响因素。可能有一些与状态本身无关的变化——如外界因素可能正在影响系统。

例如，我们用状态对列车的运动进行建模，如果列车长加大油门，火车就加速。同样，在我们的机器人示例中，导航系统软件可能会发出使车轮转动或停止的命令。如果我们很明确地知道这些因素，我们可以将其放在一起构成一个向量，我们可以对这个量进行某些“处理”，然后将其添加到我们的预测中对状态进行更正。

假设我们知道由于油门设置或控制命令而产生的预期加速度。根据基本运动学原理，我们可以得到下式：

$$\begin{aligned}\hat{\mathbf{x}}_k &= \mathbf{F}_k\hat{\mathbf{x}}_{k-1} + \begin{bmatrix}\frac{\Delta t^2}{2} \\ \Delta t\end{bmatrix} a \\ &= \mathbf{F}_k\hat{\mathbf{x}}_{k-1} + \mathbf{B}_k\vec{\mathbf{u}}_k\end{aligned} \tag{6}$$

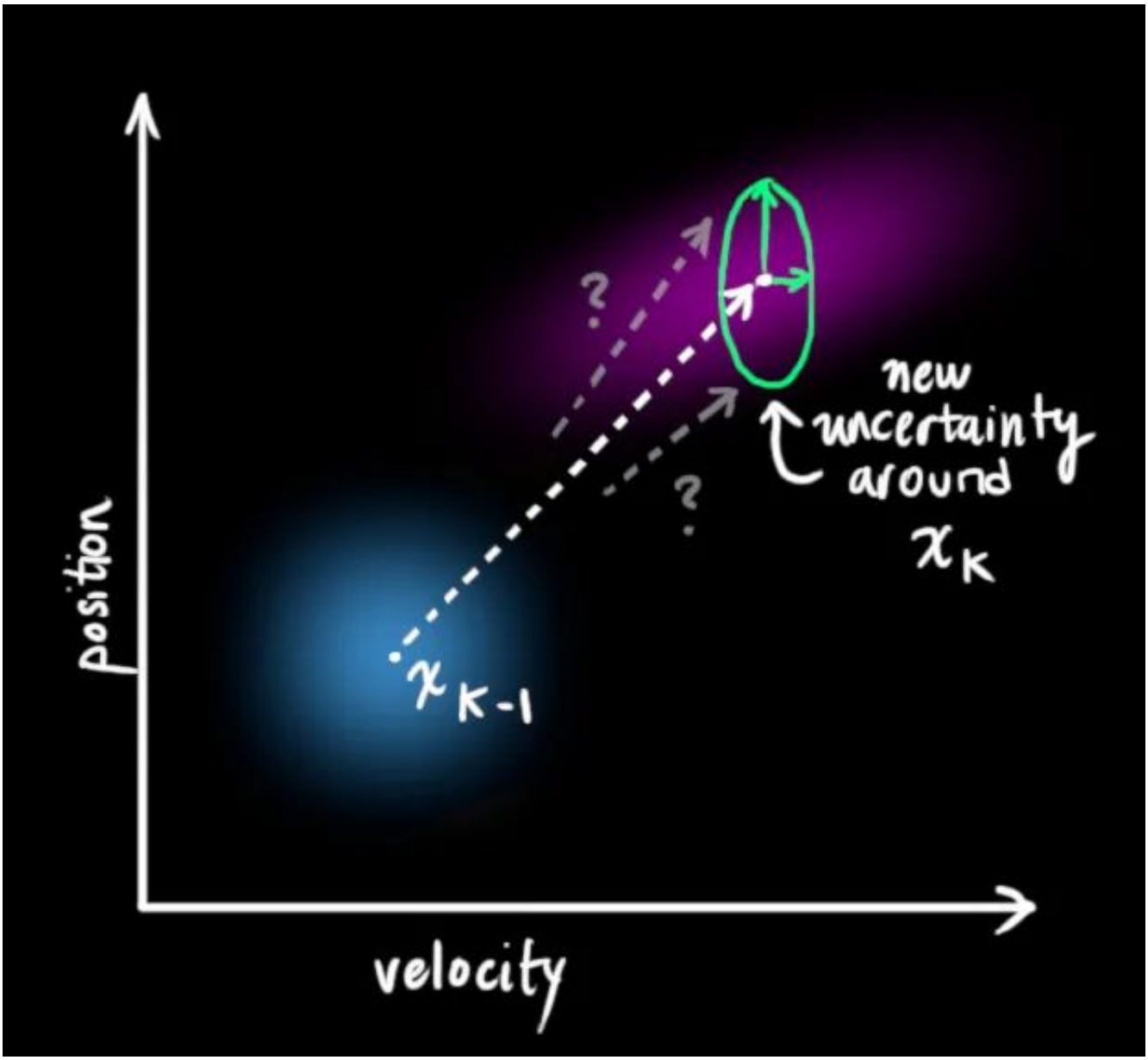
其中被称为**控制矩阵**，被称为**控制向量**。（注意：对于没有外部影响的简单系统，可以忽略这个控制项）。

如果我们的预测并不是100%准确模型，这会发生什么呢?

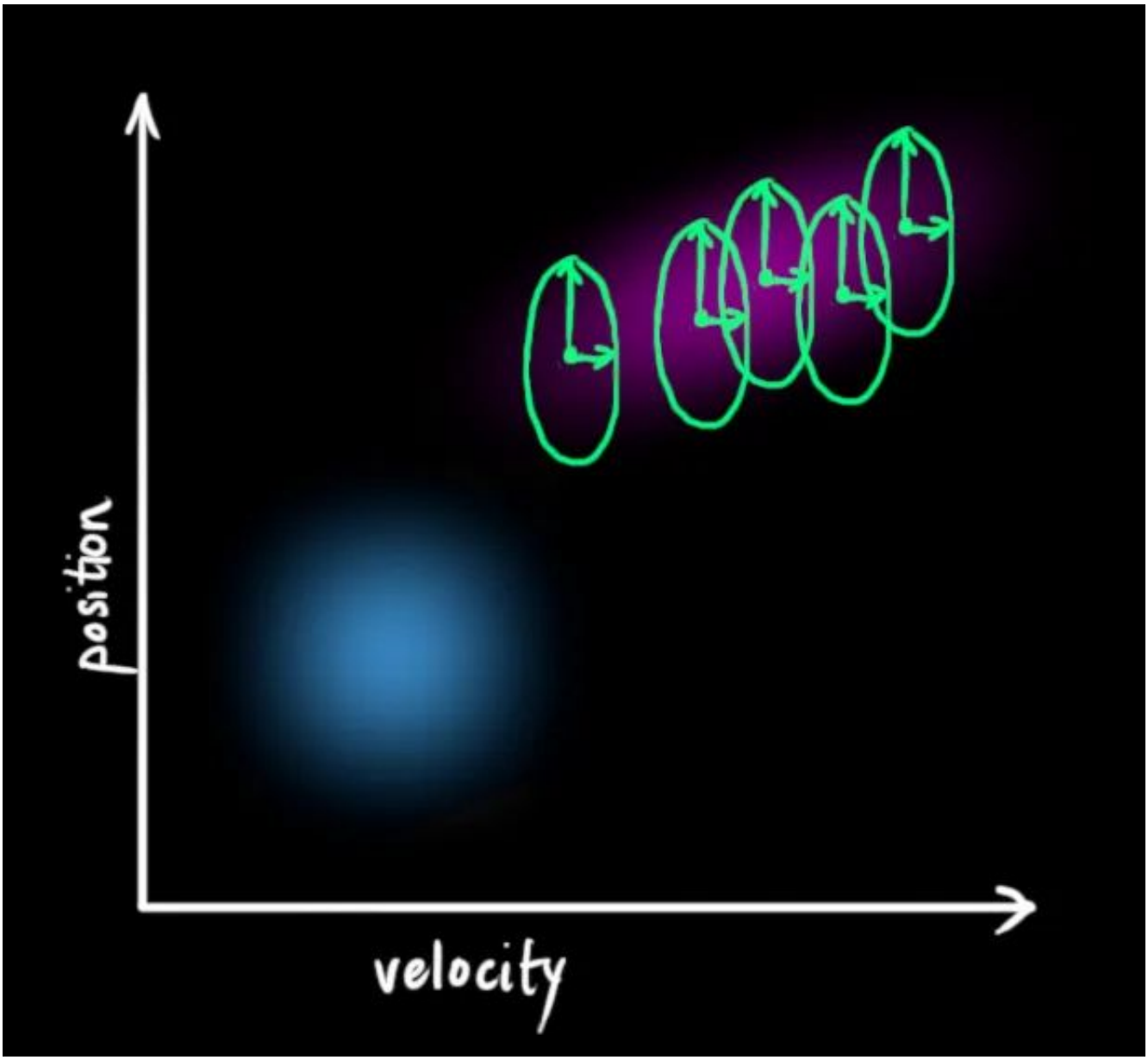
如果状态仅仅依赖其自身的属性进行演进，那一切都会很好。如果状态受到外部因素进行演进，我们只要知道这些外部因素是什么，那么一切仍然很好。

但在实际使用中，我们有时不知道的这些外部因素到底是如何被建模的。例如，我们要跟踪四轴飞行器，它可能会随风摇晃；如果我们跟踪的是轮式机器人，则车轮可能会打滑，或者因地面颠簸导致其减速。我们无法跟踪这些外部因素，如果发生任何这些情况，我们的预测可能会出错，因为我们并没有考虑这些因素。

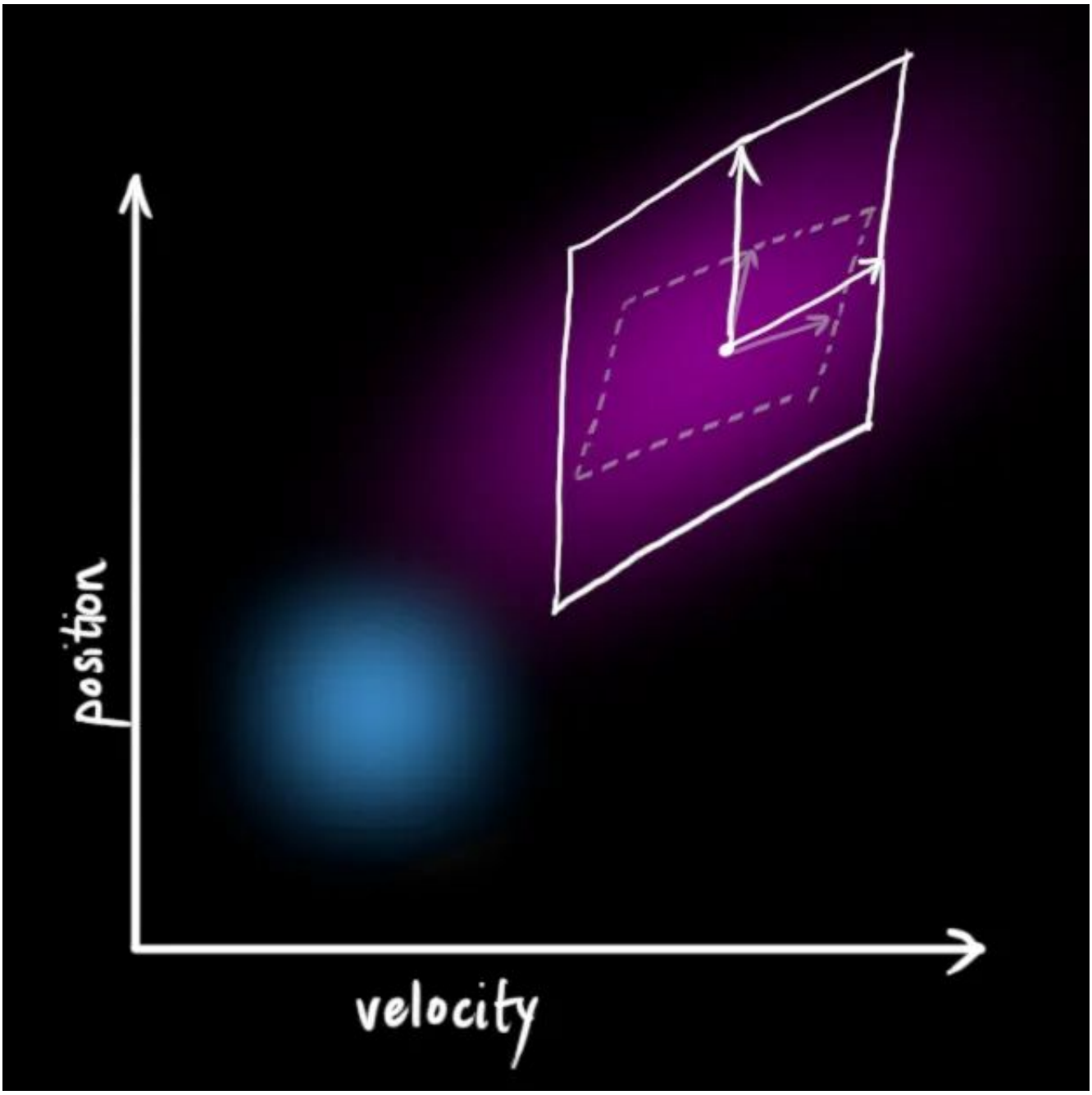
通过在每个预测步骤之后添加一些新的不确定性，我们可以对与“世界”相关的不确定性进行建模（如我们无法跟踪的事物）：



这样一来，由于新增的不确定性**原始估计中的每个状态都可能迁移到多个状态**。因为我们非常喜欢用高斯分布进行建模，此处也不例外。我们可以说的每个点都移动到具有协方差的高斯分布内的某个位置，如下图所示：



这将产生一个新的高斯分布，其协方差不同（但均值相同）：



所以呢，我们在状态量的协方差中增加额外的协方差，所以**预测阶段**完整的状态转移方程为：

换句话说：**新的最佳估计是根据先前的最佳估计做出的预测**，再加上对**已知外部影响**的校正。

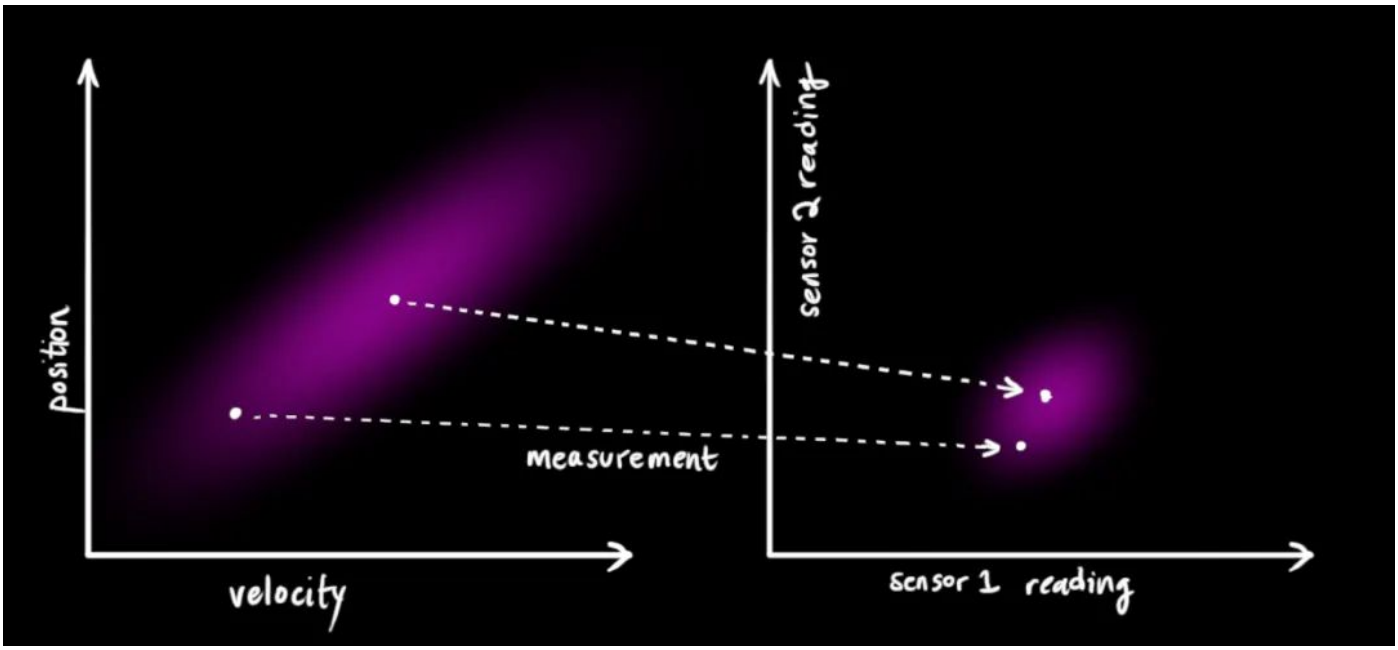
新的不确定度是根据先前的不确定度做出的预测，再加上**来自环境额外的不确定度**。

上述过程描绘了状态预测过程，那么当我们从传感器中获取一些测量数据时会发生什么呢?

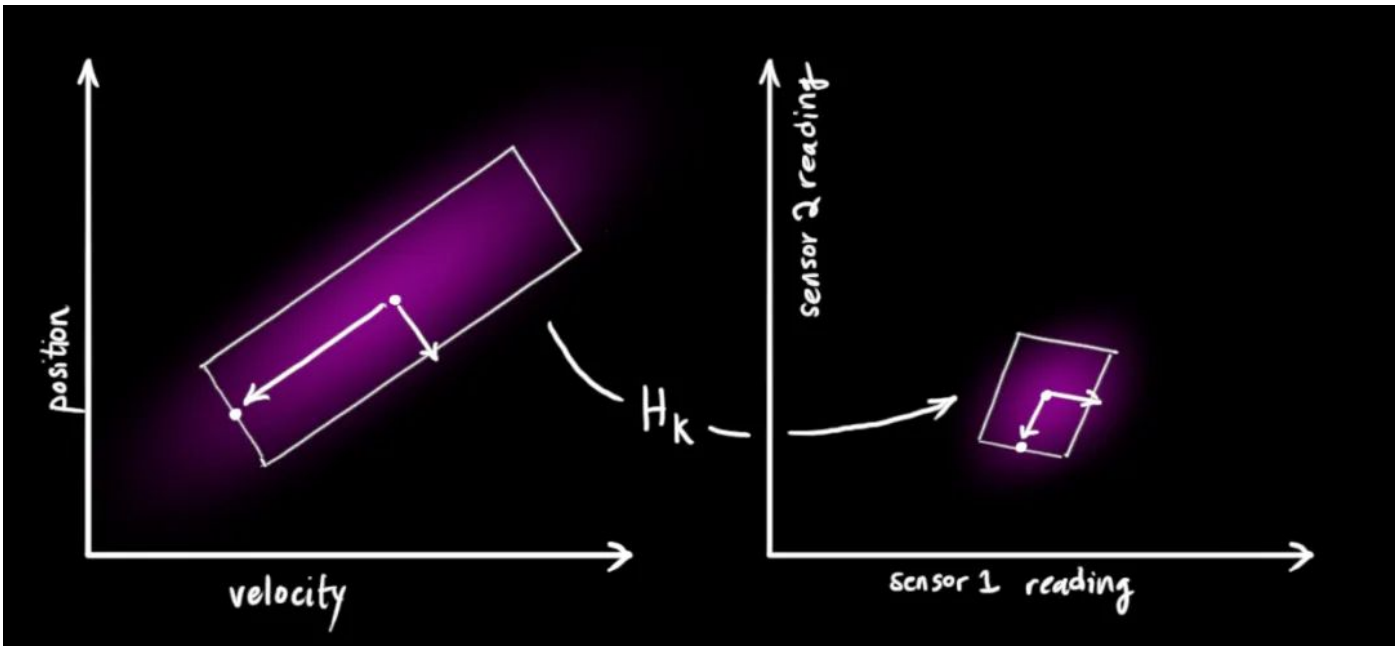
3 状态更新

利用测量进一步修正状态

假设我们有几个传感器，这些传感器可以向我们提供有关系统状态的信息。就目前而言，测量什么量都无关紧要，也许一个读取位置，另一个读取速度。每个传感器都告诉我们有关状态的一些间接信息（换句话说，传感器在状态下运作并产生一组测量读数）。

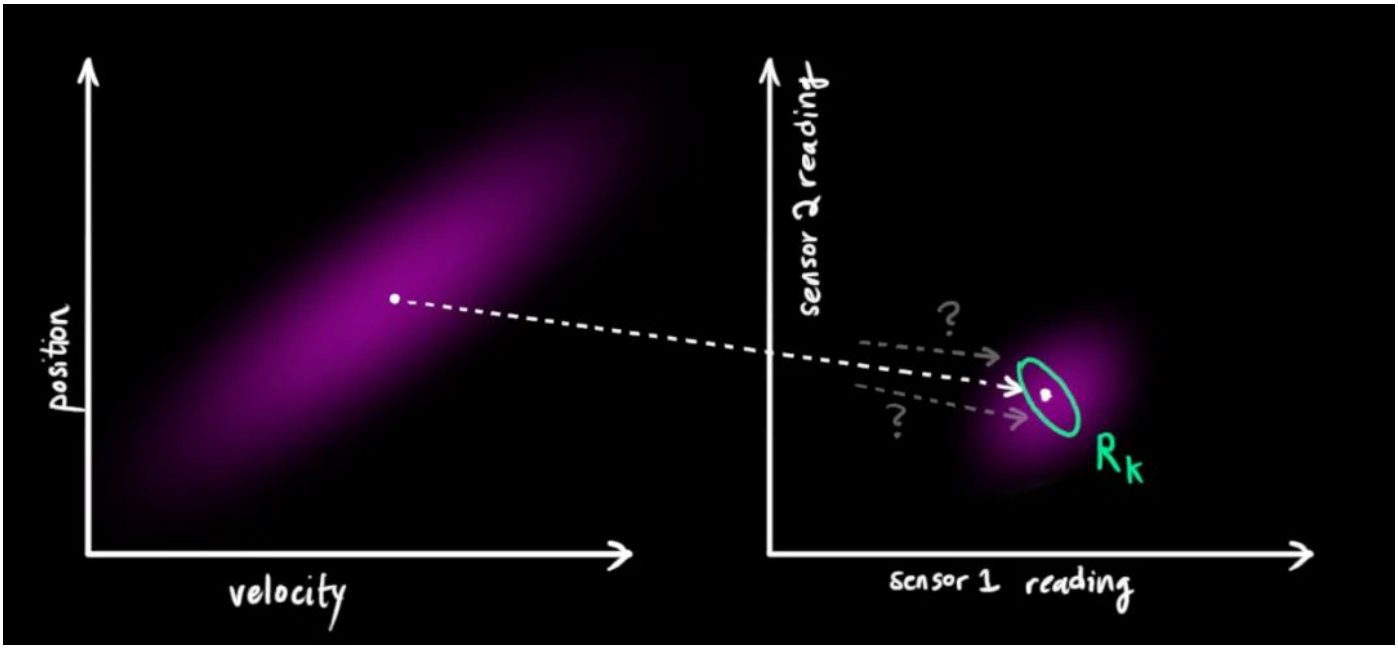


请注意，测量的单位可能与状态量的单位不同。我们使用矩阵对传感器的测量进行建模。



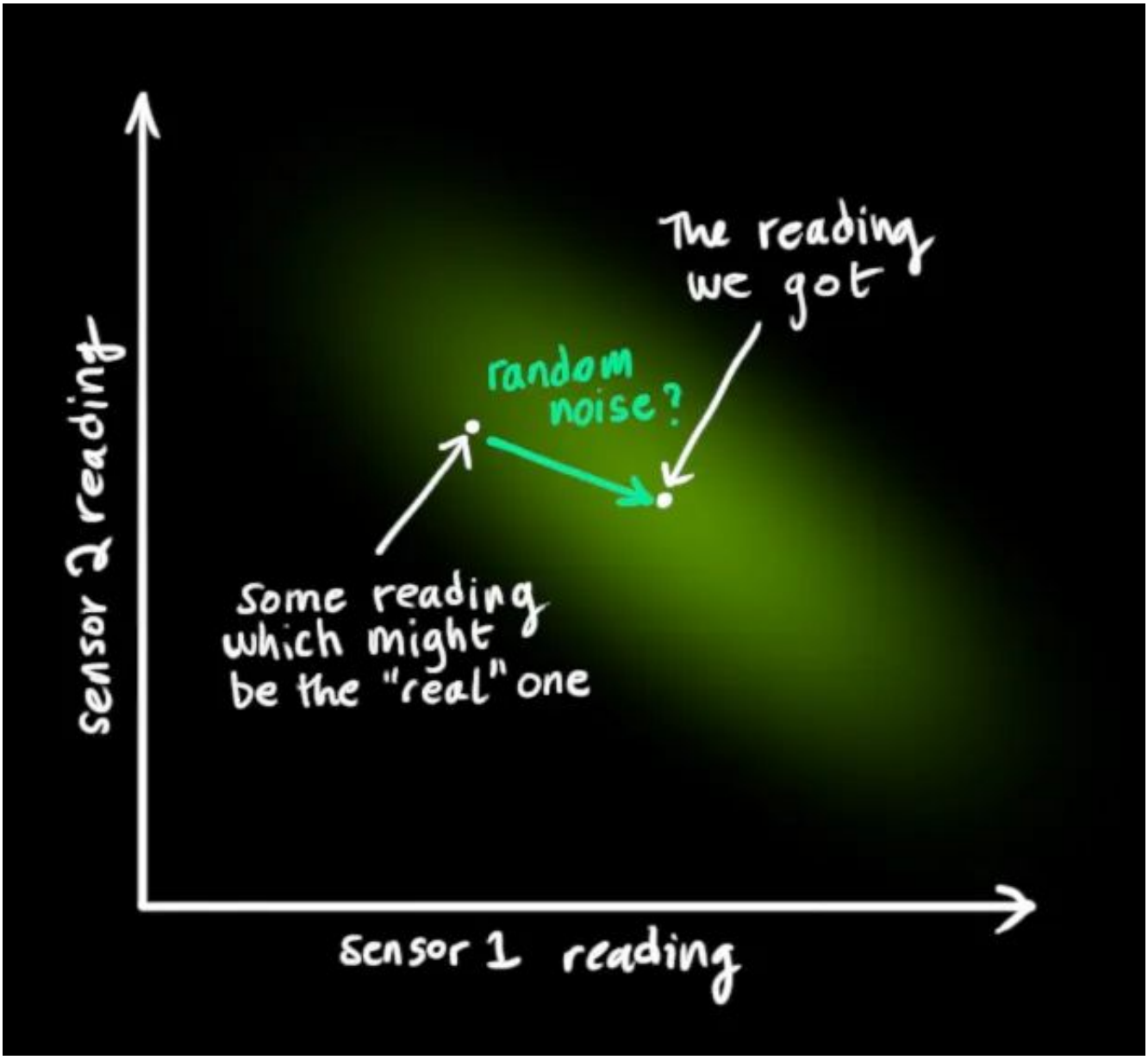
$$\begin{aligned}\vec{\mu}_{\text{expected}} &= \mathbf{H}_k\hat{\mathbf{x}}_k \\ \mathbf{\Sigma}_{\text{expected}} &= \mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T\end{aligned} \tag{8}$$

卡尔曼滤波器的伟大之处就在于它能够处理传感器噪声。换句话说，传感器本身的测量是不准确的，且原始估计中的每个状态都可能导致一定范围的传感器读数，而卡尔曼滤波能够在这些不确定性存在的情况下找到最优的状态。

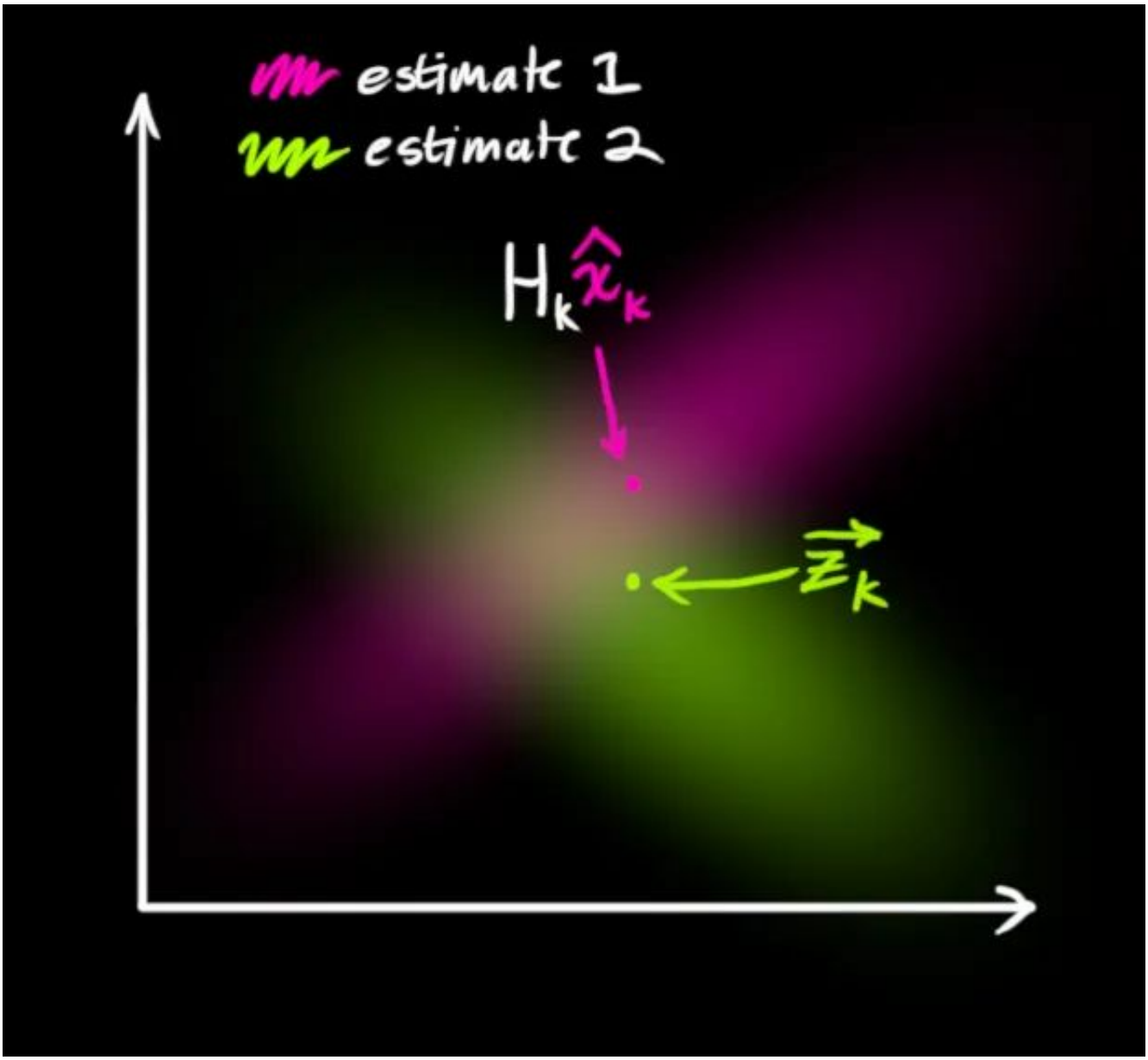


根据传感器的读数，我们会猜测系统正处于某个特定状态。但是由于不确定

性的存在，**某些状态比其他状态更可能产生我们看到的读数**：



我们将这种不确定性（如传感器噪声）的**协方差**表示为，**读数的分布均值**等于我们观察到**传感器的读数**，我们将其表示为
这样一来，我们有了两个高斯分布：**一个围绕通过状态转移预测的平均值，另一个围绕实际传感器读数。**

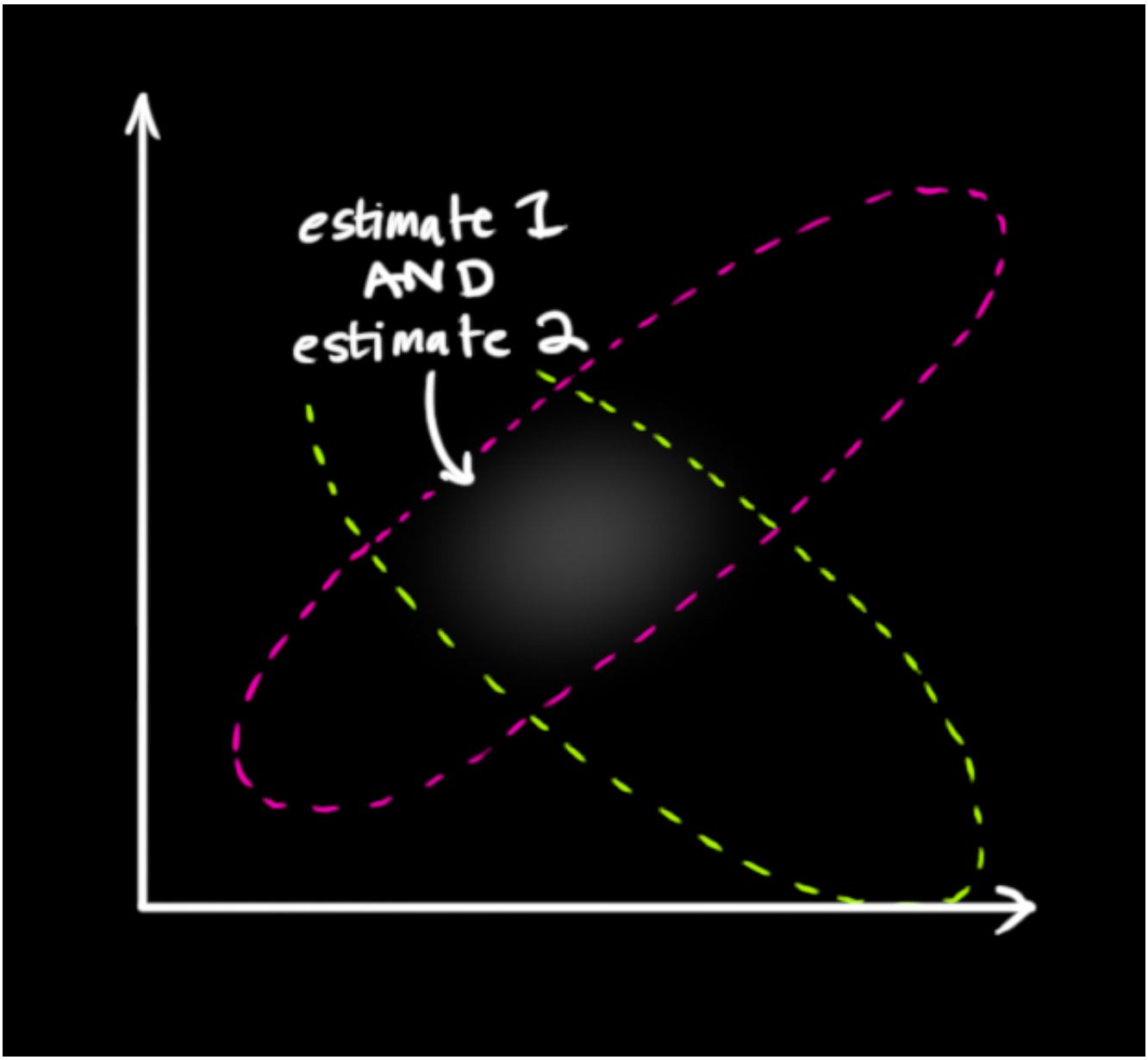


因此，我们需要将基于预测状态（**粉红色**）的推测读数与基于实际观察到的传感器读数（**绿色**）进行融合。

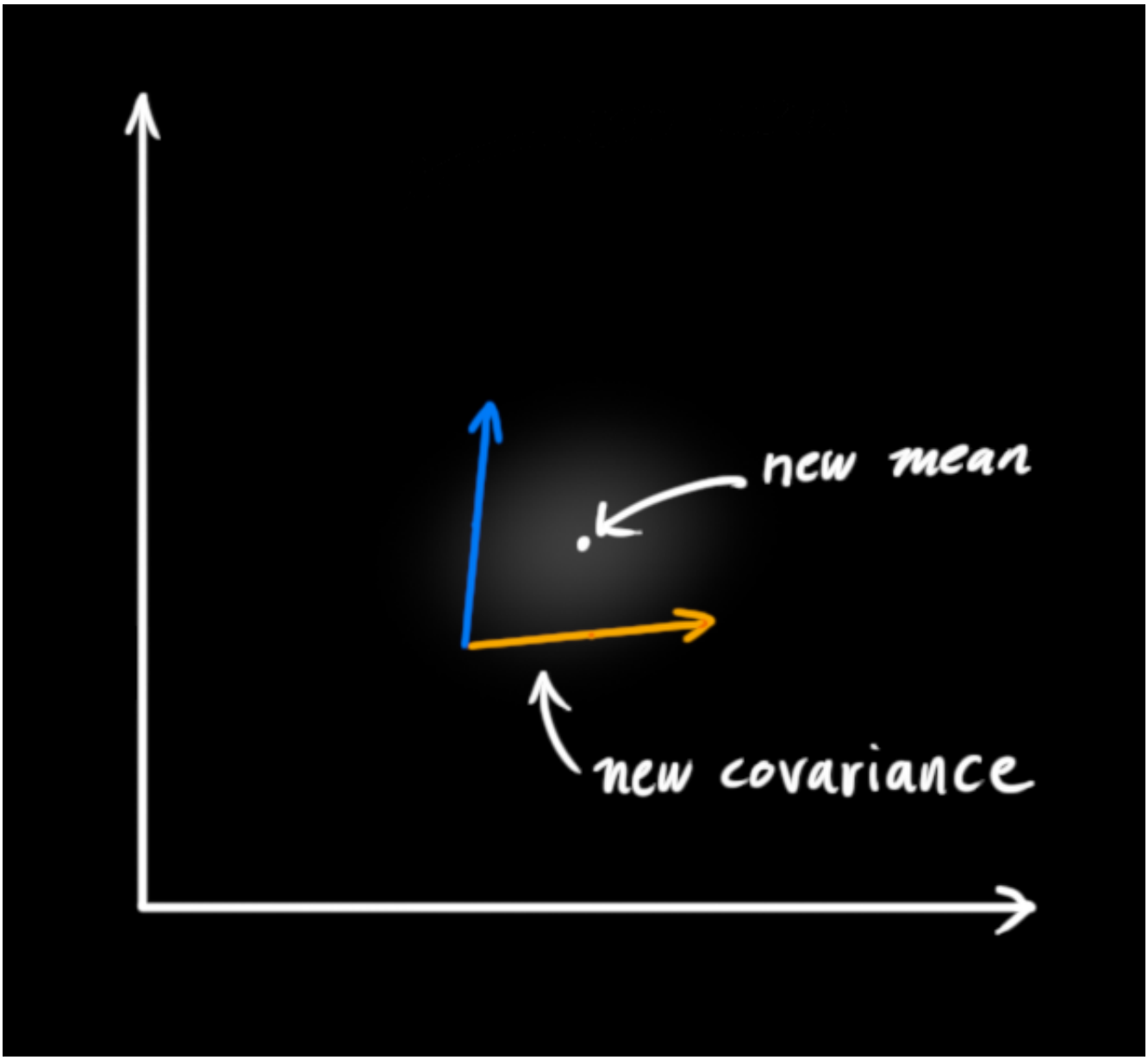
那么融合后**最有可能的新状态**是什么？对于任何可能的读数，我们都有两个相关的概率：（1）我们的传感器读数是的测量值的概率，以及（2）先前估计值的概率认为是我们应该看到的读数。

如果我们有两个概率，并且想知道两个概率都为真的机会，则将它们相乘。

因此，我们对两个高斯分布进行了相乘处理：



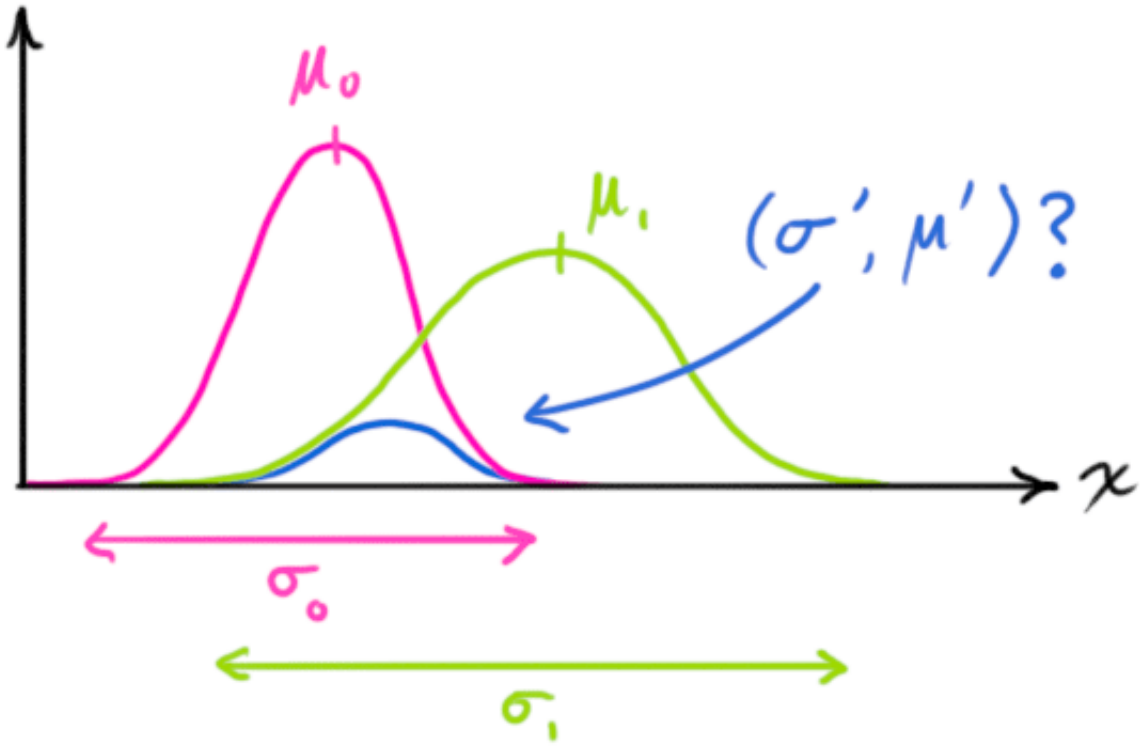
两个概率分布相乘得到的就是上图中的重叠部分。而且重叠部分的概率分布会比我们之前的任何一个估计值/读数都精确得多，这个分布的均值就是两种估计最有可能配置（得到的状态）。



事实证明，**两个独立的高斯分布相乘之后会得到一个新的具有其均值和协方差矩阵的高斯分布**！下面开始推公式。

首先考虑一维高斯情况：一个均值为，方差为的高斯分布的形式为：

我们想知道将两个高斯曲线相乘会发生什么。下图中的蓝色曲线表示两个高斯总体的（未归一化）交集：



将公式(**9**)代入公式(**10**)，我们可以得到新的高斯分布的均值和方差如下所示：

$$\begin{aligned}\mu' &= \mu_0 + \frac{\sigma_0^2(\mu_1-\mu_0)}{\sigma_0^2+\sigma_1^2} \\ \sigma'^2 &= \sigma_0^2 - \frac{\sigma_0^4}{\sigma_0^2+\sigma_1^2}\end{aligned} \tag{11}$$

$$\mathbf{k} = \frac{\sigma_0^2}{\sigma_0^2+\sigma_1^2} \tag{12}$$

$$\begin{aligned}\mu' &= \mu_0 + \mathbf{k}(\mu_1-\mu_0) \\ \sigma'^2 &= \sigma_0^2 - \mathbf{k}\sigma_0^2\end{aligned} \tag{13}$$

这样一来，公式的形式就简单多了！我们顺势将公式(**12**)和(**13**)的矩阵形式写在下面：

$$\mathbf{K} = \Sigma_0(\Sigma_0+\Sigma_1)^{-1} \tag{14}$$

$$\begin{aligned}\vec{\mu}' &= \vec{\mu_0} + \mathbf{K}(\vec{\mu_1}-\vec{\mu_0}) \\ \Sigma' &= \Sigma_0 - \mathbf{K}\Sigma_0\end{aligned} \tag{15}$$

其中表示新高斯分布的协方差矩阵，是每个维度的均值，就是大名鼎鼎

的“卡尔曼增益”（**Kalman gain**）。

公式汇总

我们有两个高斯分布，一个是我们预测的观测，另外一个是实际的观测(传感器读数)，我们将这两个高斯分布代入公式(15)中就可以得到二者的重叠区域：

然后我们将公式(**16**)与公式(**17**)中的去除，同时将后面的去除，我们可以得到最终的化简形式的更新方程：

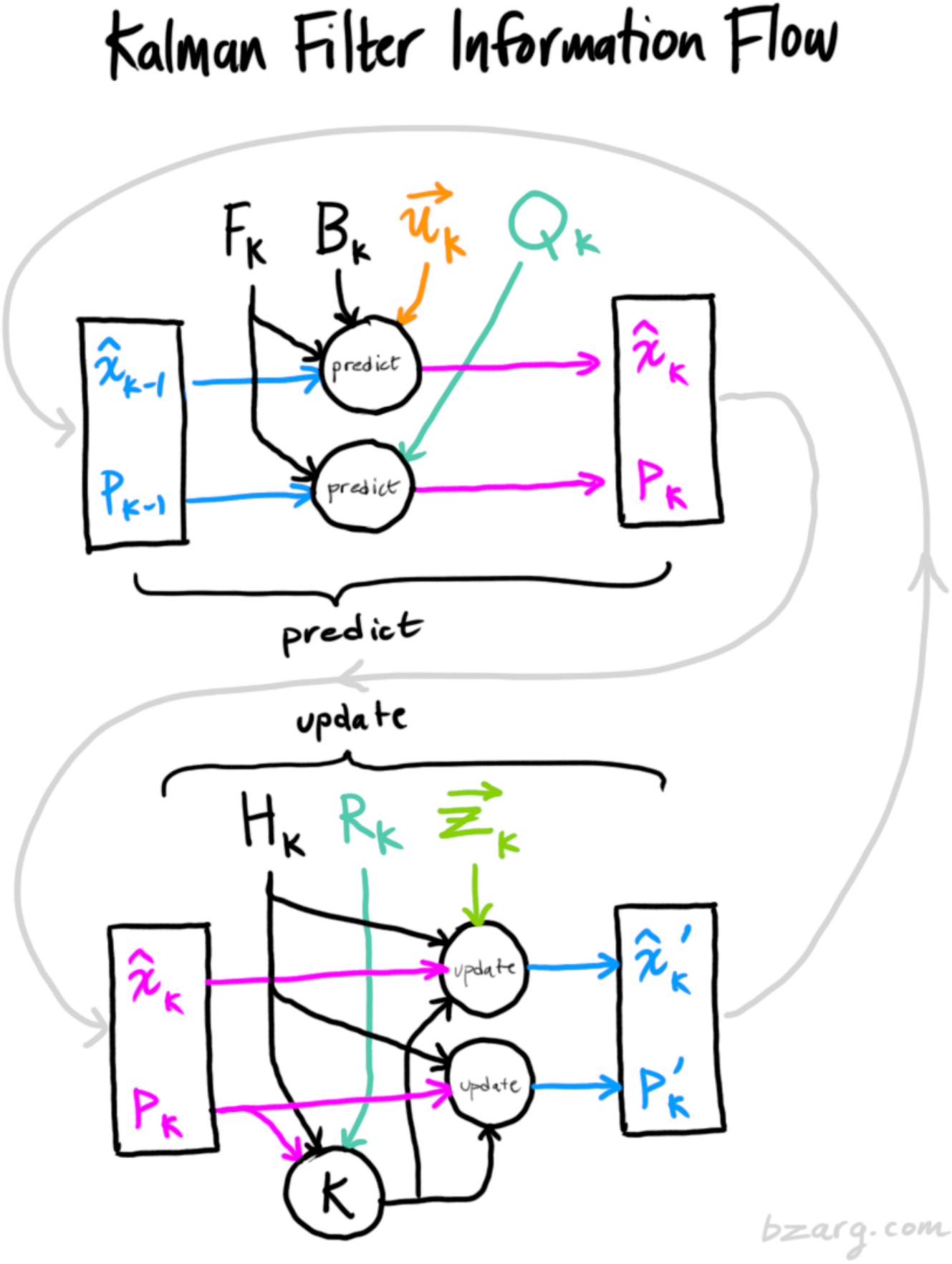
$$\hat{\mathbf{x}}_k' = \hat{\mathbf{x}}_k + \mathbf{K}'(\vec{\mathbf{z}}_k - \mathbf{H}_k\hat{\mathbf{x}}_k) \tag{18}$$

$$\mathbf{P}_k' = \mathbf{P}_k - \mathbf{K}'\mathbf{H}_k\mathbf{P}_k$$

$$\mathbf{K}' = \mathbf{P}_k\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{19}$$

4 图说

大功告成，就是更新后的最优状态！接下来我们可以继续进行预测，然后更新，重复上述过程。下图给出卡尔曼滤波信息流：


Kalman Filter Information Flow
F_k
B_k
$\vec{u}_k$
Q_k
$\hat{x}_{k-1}$
P_{k-1}
predict
predict
$\hat{x}_k$
P_k
predict
update
H_k
R_k
$\vec{z}_k$
$\hat{x}_k$
P_k
update
update
$\hat{x}'_k$
P'_k
K
bzarg.com

5 总结

在上述所有数学公式中，你需要实现的只是公式(**7**)(**18**)和(**19**)。
或者，如果你忘记了这些，可以从等式(**4**)和(**15**)重新推导所有内容。
这将使你能够准确地对任何线性系统建模。对于非线性系统，我们使用**扩展卡尔曼滤波器**，该滤波器通过简单地线性化预测和测量值的均值进行工作。

参考资料

[**1**]: How a Kalman filter works, in pictures, 图解卡尔曼滤波是如何工作的：
http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/#mathybits
[**2**]: A geometric interpretation of the covariance matrix, 协方差矩阵的几何解释:
https://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix
[**3**]: Kalman Filter 卡尔曼滤波:
https://sikasjc.github.io/2018/05/08/kalman_filter

推荐阅读

- 正式开赛！ACCV 2020国际细粒度网络图像识别竞赛——是时候展现真正的技术了！
- 消除Aliasing！加州大学&英伟达提出深度学习下采样新思路：自适应低通滤波器层
- 即插即用的涨点神器，南航开源AFF：注意力特征融合

ACCV 2020国际细粒度网络图像识别竞赛正式开赛！


ACCV 2020
国际细粒度网络图像识别竞赛
2020.10.9 – 2020.11.30
国际赛事
大型数据集开放下载
专业评审阵容
评审嘉宾
魏秀参
南京理工大学
计算机科学与工程学院教授
沈春华
阿德莱德大学
计算机科学学院教授
姚亚洲
南京理工大学
计算机科学与工程学院教授
王文冠
瑞士苏黎 Luc Van Gool
计算机视觉实验室 博士后
杨健
南京理工大学
计算机科学与工程学院教授
吴建鑫
南京大学
计算机科学与技术系教授

添加极市小助手微信（**ID : cvmart2**），备注：**姓名-学校/公司-研究方向-城市**（如：小极-北大-目标检测-深圳），即可申请加入**极市目标检测/图像分割/工业检测/人脸/医学影像/3D/SLAM/自动驾驶/超分辨率/姿态估计/ReID/GAN/图像增强/OCR/视频理解等技术交流群：每月大咖直播分享、真实项目需求对接、求职内推、算法竞赛、干货资讯汇总、与 10000+来自港科大、北大、清华、中科院、CMU、腾讯、百度等名校名企视觉开发者互动交流~**