《智能机器人设计》

机器人定位与导航

机器人定位与导航

定位

机器人确定自己实际位置的过程，称为定位。

导航

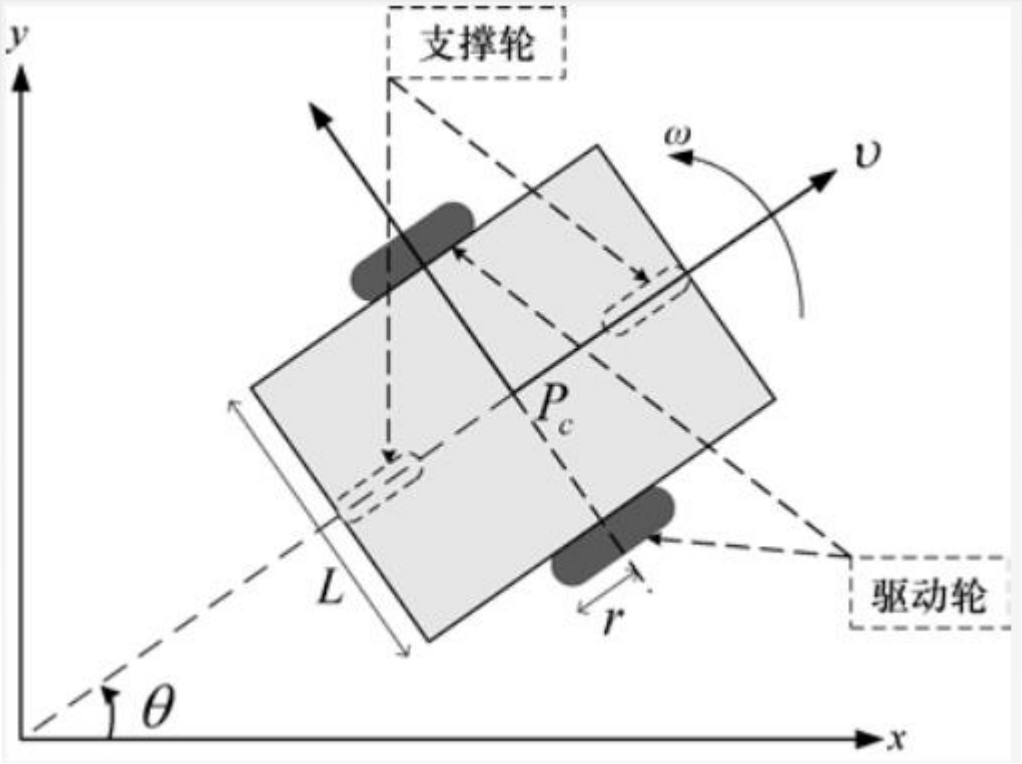
机器人被引导移动到指定位置的策略称为导航

目前世界范围内最常用的室外定位系统是全球定位系统（global positioning system,GPS),它依靠24颗卫星提供高精度无线电信号，它能在全球任何地方提供准确的地理位置。GPS自问世以来，就以其全天候、全球覆盖、方便灵活吸引了众多用户，但是在室内、隧道、水下等无线电信号无法到达的环境，GPS无法提供稳定的服务。下面介绍一种经典的定位方法——**航迹推算**

机器人定位技术—航迹推算

航迹推算（dead reckoning）是基于机器人当前的位姿，利用预测速度、移动方向、运行时间等信息对下一时刻位姿进行估算的方法。想要实现航迹推算，第一步需要构建机器人在移动过程中的数学模型。

右图为常见的两轮驱动小车俯视结构，驱动轮外半径为r，由两个独立的电机驱动，两轮间轴长度为L，设轴中心P与该小车的质心在地面的投影重合，前后配有两个支撑作用的万向轮保持车身平衡。图中全局坐标系为x0y，小车的前进方向与x轴正方向的夹角为θ，小车实时位置可由点P在坐标系中的位置表示，设列向量$q=[x\ y\ \theta]^T$。



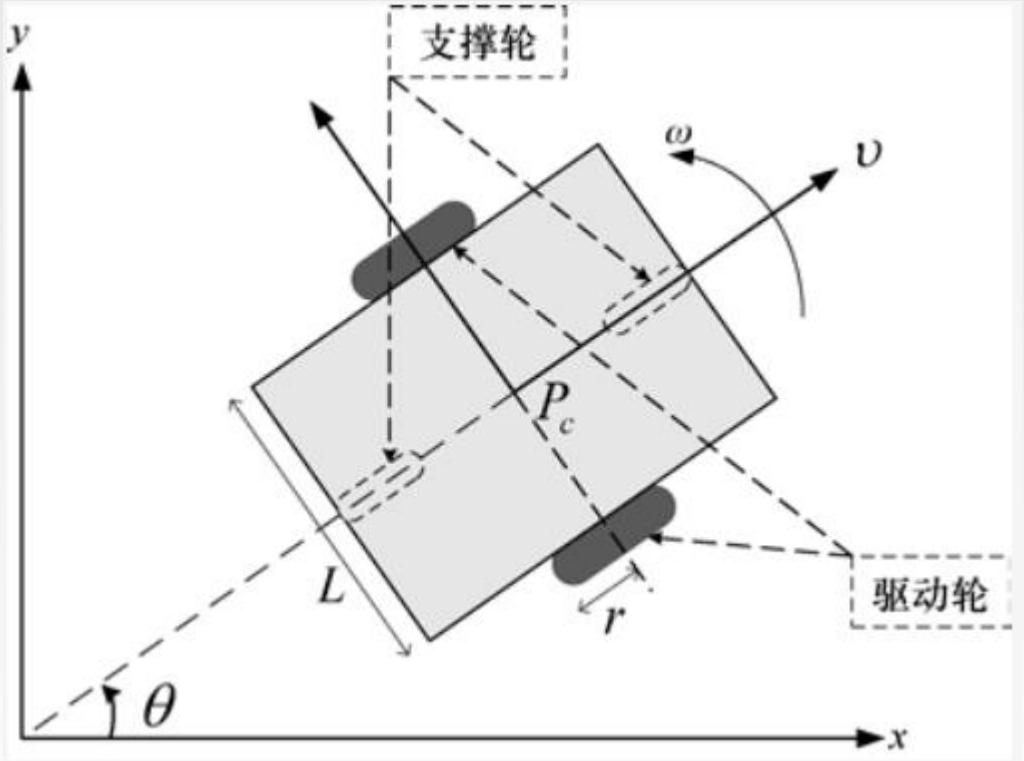
机器人定位技术—航迹推算

v为小车前进方向的线速度，可根据左右驱动轮线速度V_l、Vr计算得到：

$$v = \frac{v_l + v_r}{2}$$

这里的线速度V_l、Vr可以通过安装在驱动电机轴上的光电编码器测量得到。图中，w为偏离x轴正方向的角速度，可根据下面的式子计算得出：

$$\omega = \frac{v_r - v_l}{L}$$

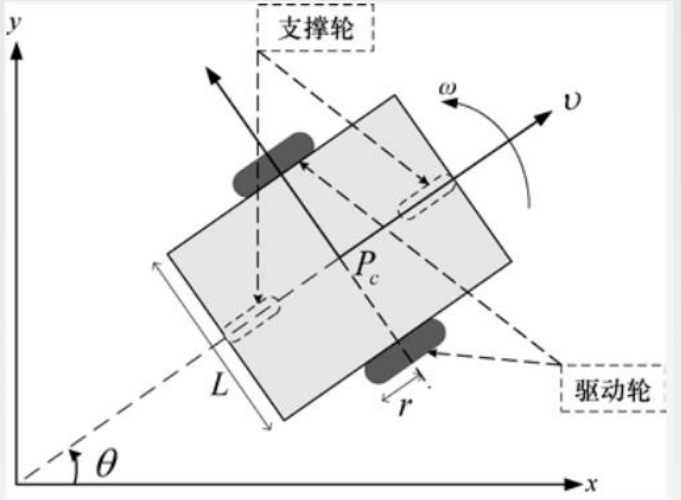


w也可以通过陀螺仪测量获得，由于实际情况下驱动轮和地面会产生一定的滑动，所以一般偏向于陀螺仪的测量值，或将两种数据互相融合后使用。

机器人定位技术—航迹推算

根据图中坐标，该系统的运动学模型可表示为：

$$\dot{q} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$



显然，上式为连续时间模型，如果选取一个很小的时间间隔T作为采样周期，k表示当前时刻，k+1表示下一时刻，将上式简单离散化后可以得到：

$$q_{k+1} = q_k + \begin{bmatrix} \cos\theta_k & 0 \\ \sin\theta_k & 0 \\ 0 & 1 \end{bmatrix} \Delta_k$$

其中$\Delta_K=[\Delta d_k \Delta \theta_k]$可以看作里程计的测量值，$\Delta d_k=T_s V_k$和$\Delta\theta=T_s w_k$分别为一个采样周期内机器人前进的距离和角度的变化量。通过式，只要知道机器人的初始位置信息，结合光电编码器、电子罗盘、陀螺仪等传感器的测量数据，就可以计算出机器人的实时位置信息。

注意该方法的准确度会随着时间的推移变得越来越差，原因是多种误差的存在，如驱动轮半径不准、行驶过程中的轮胎打滑、陀螺仪数据的漂移、电子罗盘受到电磁干扰等。

机器人定位技术—航迹推算

除了式前页所示的运动学模型，也可以给出机器人位姿的离散时间模型。设机器人在k时刻的初始位姿为：

$$\xi_k : \begin{bmatrix} \cos\theta_k & -\sin\theta_k & x_k \\ \sin\theta_k & \cos\theta_k & y_k \\ 0 & 0 & 1 \end{bmatrix}$$

因为采样周期T很小，所以在这段时间内位姿的变化可以看作先平移了△d，然后旋转了△θ，于是得到：

$$\xi_{k+1} : \begin{bmatrix} \cos\theta_k & -\sin\theta_k & x_k \\ \sin\theta_k & \cos\theta_k & y_k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \Delta d_k \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\Delta\theta_k) & -\sin(\Delta\theta_k) & 0 \\ \sin(\Delta\theta_k) & \cos(\Delta\theta_k) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_k+\Delta\theta_k) & -\sin(\theta_k+\Delta\theta_k) & x_k+\Delta d_k\cos(\Delta\theta_k) \\ \sin(\theta_k+\Delta\theta_k) & \cos(\theta_k+\Delta\theta_k) & y_k+\Delta d_k\sin(\Delta\theta_k) \\ 0 & 0 & 1 \end{bmatrix}$$

机器人定位技术—位姿估计

由于里程计存在误差，因此上节的利用式，进行航迹推算得到的实时位置是不准确的。我们可以利用估算的方法尽可能地改善计算结果，这里介绍一下卡尔曼滤波(Kalman filter)。在下式中增加里程计的随机测量噪声$w=[w_d w_\theta]^T$，可以得到非线性模型$q_{k+1}=f(qk.\triangle k,W)$：

$$q_{k+1}=q_k+\begin{bmatrix}\cos\theta_k & 0\\ \sin\theta_k & 0\\ 0 & 1\end{bmatrix}\Delta_k \Rightarrow f(q_k,\Delta_k,w)=\begin{bmatrix}x_k+(\Delta d_k+w_d)\cos\theta_k\\ y_k+(\Delta d_k+w_d)\sin\theta_k\\ \theta_k+(\Delta\theta_k+w_\theta)\end{bmatrix}$$

这里将里程计的噪声建模为$w=[w_d\ w_\theta]\sim N(0,W)$，即均值为零的高斯随机变量，并做局部线性近似，如下：

$$q_{k+1}=f(q_{k|k},\Delta_k,0)+F_q(q_k-q_{k|k})+F_w w_k$$

其中W为协方差矩阵 $W=\begin{bmatrix}\sigma_d^2 & 0\\ 0 & \sigma_\theta^2\end{bmatrix}$

$q_{k|k}$是当前时刻的最优估计值，$F_q=\partial f/\partial q$和$F_w=\partial f/\partial w$为雅可比矩阵，在每一个迭代过程中被重新赋值。

机器人定位技术—位姿估计

参照卡尔曼滤波的方程可以得到以下的预测方程：

$$\boldsymbol{q}_{k+1|k} = f\left(\boldsymbol{q}_{k|k}, \Delta_k, 0\right)$$
$$\boldsymbol{P}_{k+1|k} = \boldsymbol{F}_q \boldsymbol{P}_{k|k} \boldsymbol{F}_q^{\mathrm{T}} + \boldsymbol{F}_w \boldsymbol{W} \boldsymbol{F}_w^{\mathrm{T}}$$

这种方法称为扩展卡尔曼滤波（extended Kalman filter,EKF),具体推导过程这里不做展开介绍，值得注意的是，上式与标准线性卡尔曼滤波两个预测方程在形式上有所不同。首先，对于状态向量q的预测，仍然基于非线性模型式，因为当前时刻的噪声无法得知，所以假设w等于其均值0。其次，协方差矩阵P的预测方程中W左右增加了雅可比矩阵F_w和F_w^T。及我们只得到两个预测方程，因为没有外部传感器对机器人的位姿进行测量，所以无法对预测状态$q_{k+1|k}$进行修正，从而得到最优化的估计值。协方差矩阵P代表机器人预测位姿的不确定性，由于F_w W F_w^T这部分是正定的，这就表明不确定性P会不断增大。

机器人定位技术—卡尔曼滤波

卡尔曼滤波（Kalman filtering）是一种利用线性系统状态方程，通过系统输入输出观测数据，对系统状态进行最优估计的算法。由于观测数据中包括系统中的噪声和干扰的影响，所以最优估计也可看作是滤波过程。

数据滤波是去除噪声还原真实数据的一种数据处理技术，Kalman滤波在测量方差已知的情况下能够从一系列存在测量噪声的数据中，估计动态系统的状态。由于它便于计算机编程实现，并能够对现场采集的数据进行实时的更新和处理，Kalman滤波是目前应用最为广泛的滤波方法，在通信，导航，制导与控制等多领域得到了较好的应用。

简单来说，卡尔曼滤波器是一个最优化自回归数据处理算法。对于解决很大部分的问题，他是最优，效率最高甚至是最有用的。他的广泛应用已经超过30年，包括机器人导航，控制，传感器数据融合甚至在军事方面的雷达系统以及导弹追踪等等。近来更被应用于计算机图像处理，例如头脸识别，图像分割，图像边缘检测等等。

机器人定位技术—卡尔曼滤波

卡尔曼滤波说明：

假设条件

假设我们要研究的对象是一个房间的温度。根据你的经验判断，这个房间的温度是恒定的，也就是下一分钟的温度等于现在这一分钟的温度（假设我们用一分钟来做时间单位）。假设你对你的经验不是100%的相信，可能会有上下偏差几度。我们把这些偏差看成是高斯白噪声（White Gaussian Noise），也就是这些偏差跟前后时间是没有关系的而且符合高斯分布（Gaussian Distribution）。另外，我们在房间里放一个温度计，但是这个温度计也不准确的，测量值会比实际值偏差。我们也把这些偏差看成是高斯白噪声。

机器人定位技术—卡尔曼滤波

卡尔曼滤波说明：

数值估计

假如我们要估算k时刻的实际温度值。首先你要根据k-1时刻的温度值，来预测k时刻的温度。因为你相信温度是恒定的，所以你会得到k时刻的温度预测值是跟k-1时刻一样的，假设是23度，同时该值的高斯噪声的偏差是5度

（5是这样得到的：如果k-1时刻估算出的最优温度值的偏差是3，你对自己预测的不确定度是4度，他们平方相加再开方，就是5）。

然后，你从温度计那里得到了k时刻的温度值，假设是25度，同时该值的偏差是4度。

机器人定位技术—卡尔曼滤波

卡尔曼滤波说明：

数值计算

由于我们用于估算k时刻的实际温度有两个温度值，分别是23度和25度。究竟实际温度是多少呢？相信自己还是相信温度计呢？究竟相信谁多一点，我们可以用他们的协方差(covariance)来判断。因为Kg=5^2/(5^2+4^2)，所以Kg=0.61，我们可以估算出k时刻的实际温度值是：23+0.61*(25-23)=24.22度。可以看出，因为温度计的协方差(covariance)比较小（比较相信温度计），所以估算出的最优温度值偏向温度计的值。

机器人定位技术—卡尔曼滤波

卡尔曼滤波说明：

数值递归

现在我们已经得到k时刻的最优温度值了，下一步就是要进入k+1时刻，进行新的最优估算。到现在为止，好像还没看到什么自回归的东西出现。对了，在进入k+1时刻之前，我们还要算出k时刻那个最优值（24.22度）的偏差。

算法如下：((1-Kg)*5^2)^0.5=3.12。这里的5就是上面的k时刻你预测的那个23度温度值的偏差，得出的3.12就是进入k+1时刻以后k时刻估算出的最优温度值的偏差（对应于上面的3）。

就是这样，卡尔曼滤波器就不断的把协方差(covariance)递归，从而估算出最优的温度值。他运行的很快，而且它只保留了上一时刻的协方差(covariance)。上面的Kg，就是卡尔曼增益（Kalman Gain）。他可以随不同的时刻而改变他自己的值！

机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

对于卡尔曼滤波器，我们几乎可以下这么一个定论：只要是存在不确定信息的动态系统，卡尔曼滤波就可以对系统下一步要做什么做出有根据的推测。即便有噪声信息干扰，卡尔曼滤波通常也能很好的弄清楚究竟发生了什么，找出现象间不易察觉的相关性。因此卡尔曼滤波非常适合不断变化的系统，它的优点还有内存占用较小（只需保留前一个状态）、速度快，是实时问题和嵌入式系统的理想选择。

机器人定位技术—卡尔曼滤波

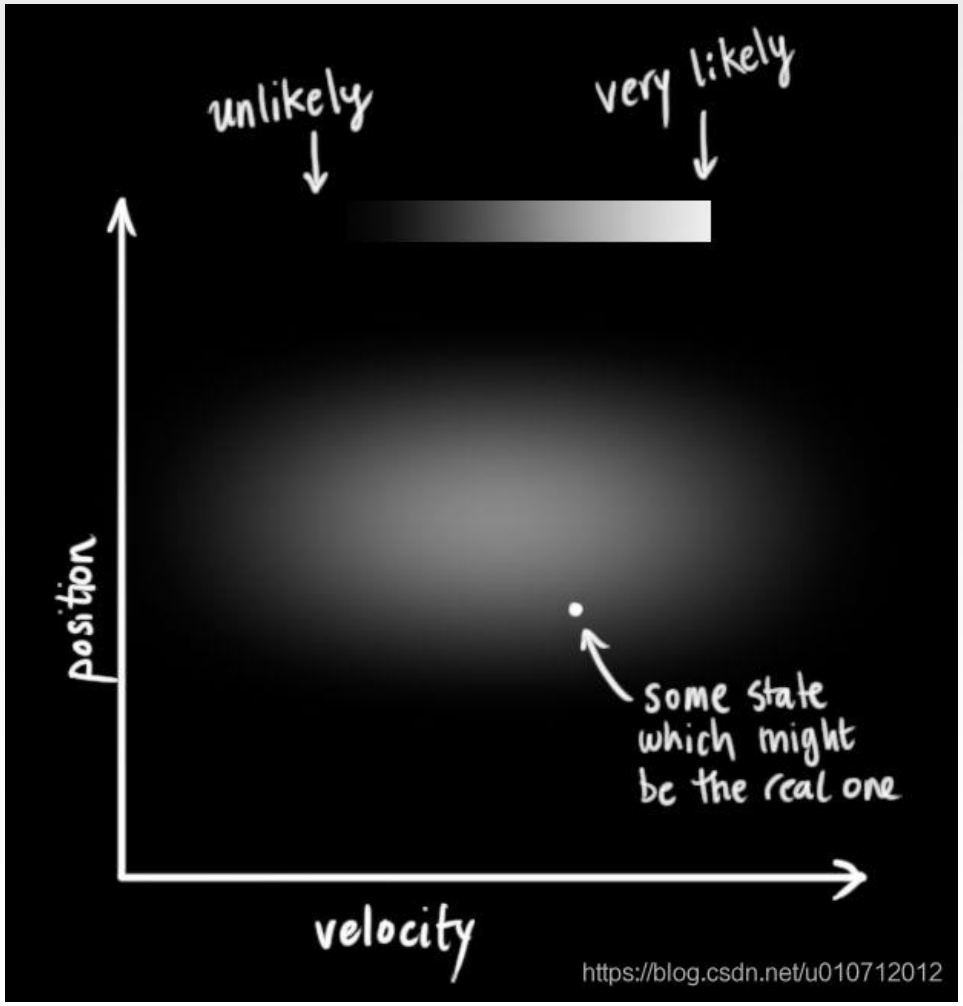
卡尔曼滤波讲解：

举个例子：在树林里四处溜达的小机器人，为了让它实现导航，机器人需要知道自己所处的位置。

我们定一个状态，它和速度、位置有关：

$$\vec{x} = \begin{bmatrix} p \\ v \end{bmatrix}$$

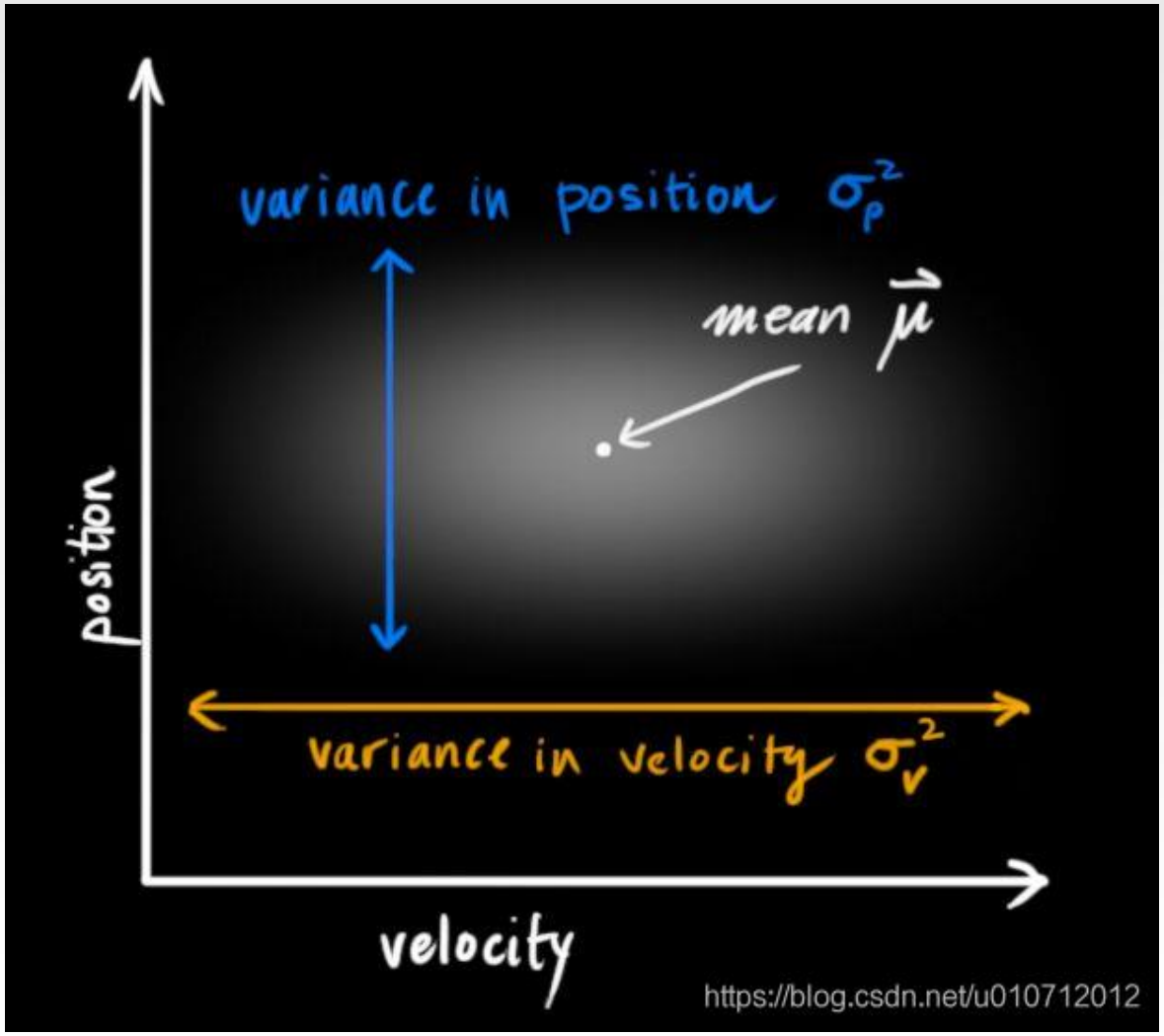
我们不知道它们的实际值是多少，但掌握着一些速度和位置的可能组合，其中某些组合的可能性更高：



机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

卡尔曼滤波假设两个变量（在我们的例子里是位置和速度）都应该是随机的，而且符合高斯分布。每个变量都有一个均值 μ ，它是随机分布的中心；有一个方差 σ^2 ，它衡量组合的不确定性。

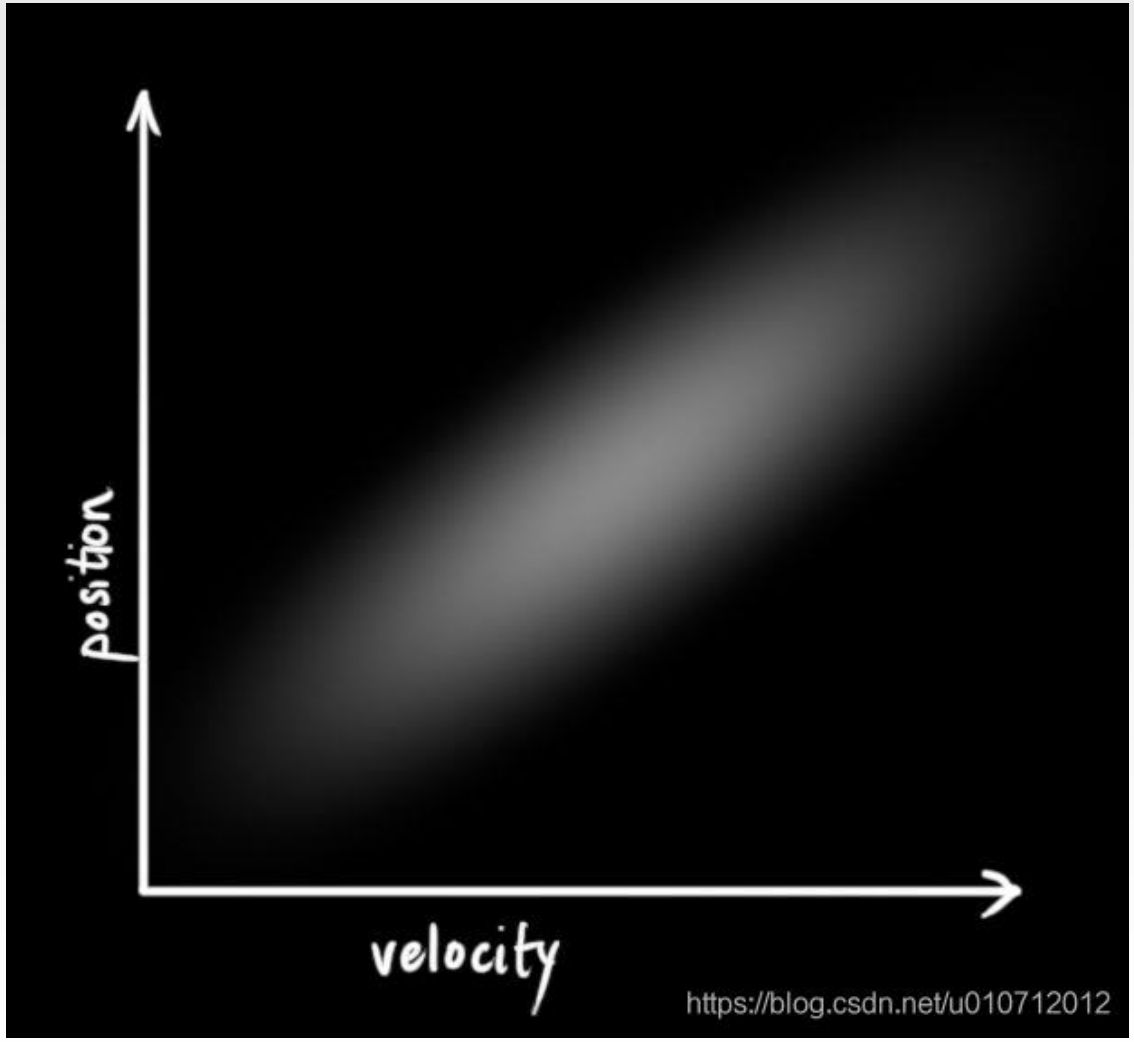


在上图中，位置和速度是不相关的，这意味着我们不能从一个变量推测另一个变量。

机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

如果位置和速度相关呢？如图所示，机器人前往特定位置的可能性取决于它拥有的速度。



我们可以理解成，如果基于旧位置估计新位置，我们会产生这两个结论：如果速度很快，机器人可能移动得更远，所以得到的位置会更远；如果速度很慢，机器人就走不了那么远。这种关系对目标跟踪来说非常重要，因为它提供了更多信息：一个可以衡量可能性的标准。这就是卡尔曼滤波的目标：从不确定信息中挤出尽可能多的信息！

机器人定位技术—卡尔曼滤波

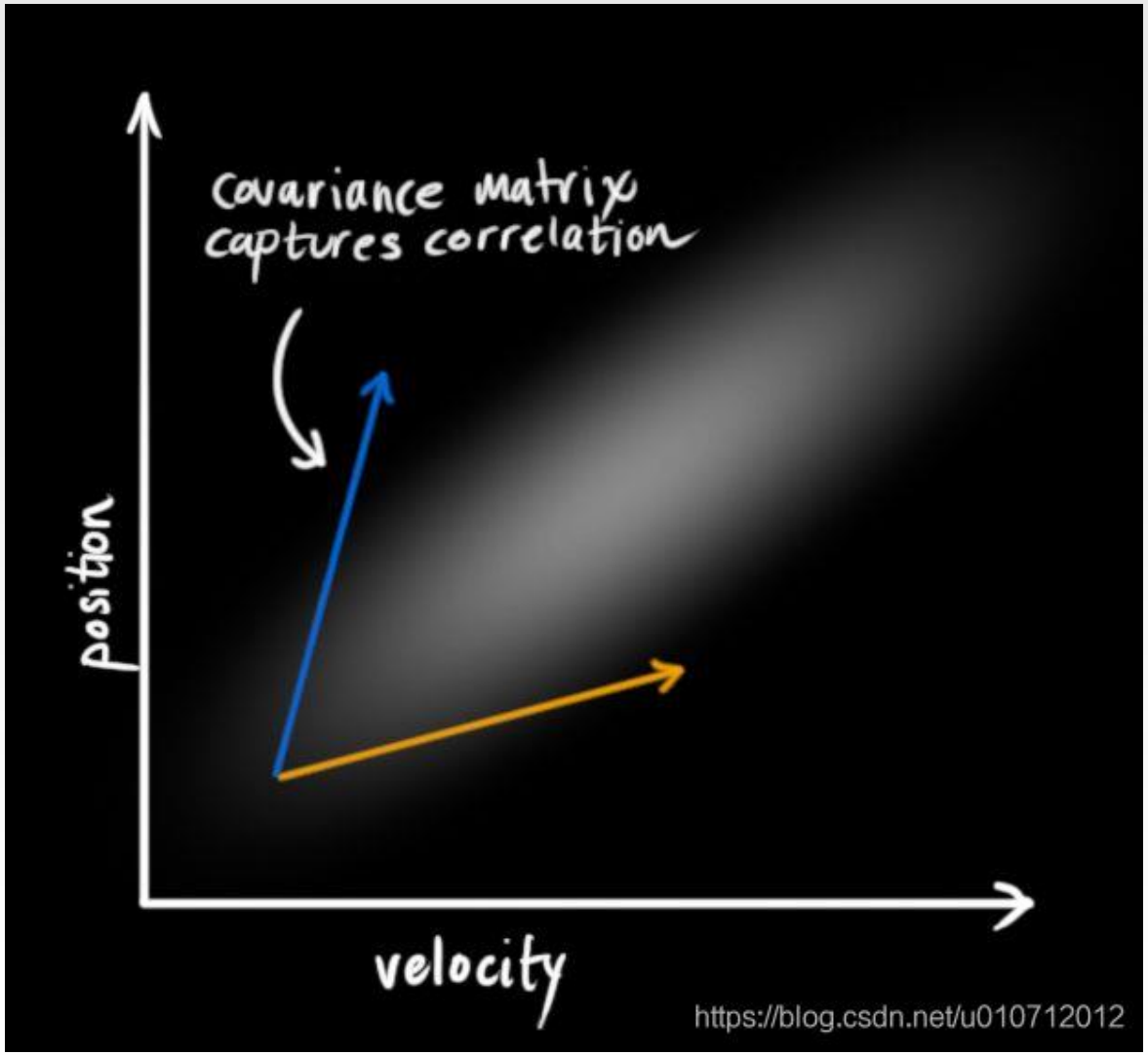
卡尔曼滤波讲解：

为了捕获这种相关性，我们用的是协方差矩阵。简而言之，矩阵的每个值是第i个变量和第j个变量之间的相关程度（由于矩阵是对称的，i 和 j 的位置可以随便交换）。我们用Σ(sigma)表示协方差矩阵，在这个例子中，就是Σ_{ij}

用矩阵描述问题为了把以上关于状态的信息建模为高斯分布（图中色块），我们还需要 k 时的两个信息：最佳估计$\hat{X}_k$(均值，也就是 μ ），协方差矩阵P_k

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

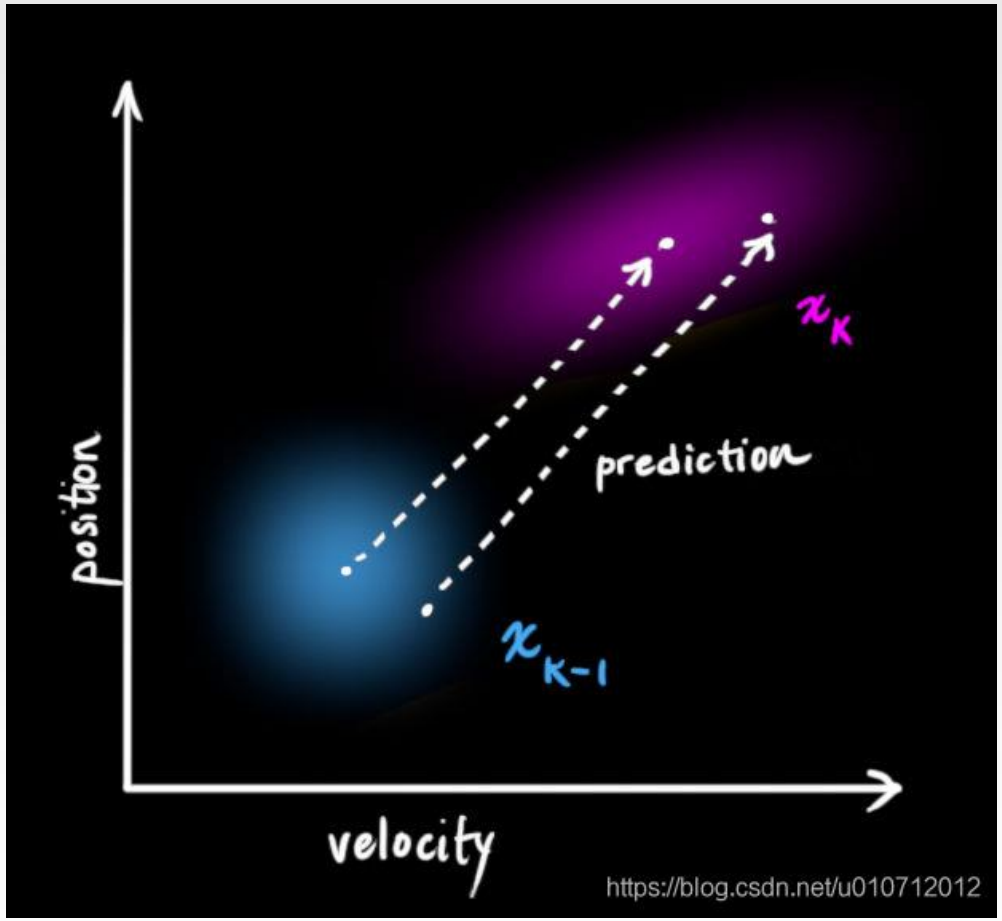
$$\mathbf{P}_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{vp} & \Sigma_{vv} \end{bmatrix}$$



机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

我们要通过查看当前状态（k-1时）来预测下一个状态（k时）。这里我们查看的状态不是真值，但预测函数无视真假，可以给出新分布：



机器人定位技术—卡尔曼滤波

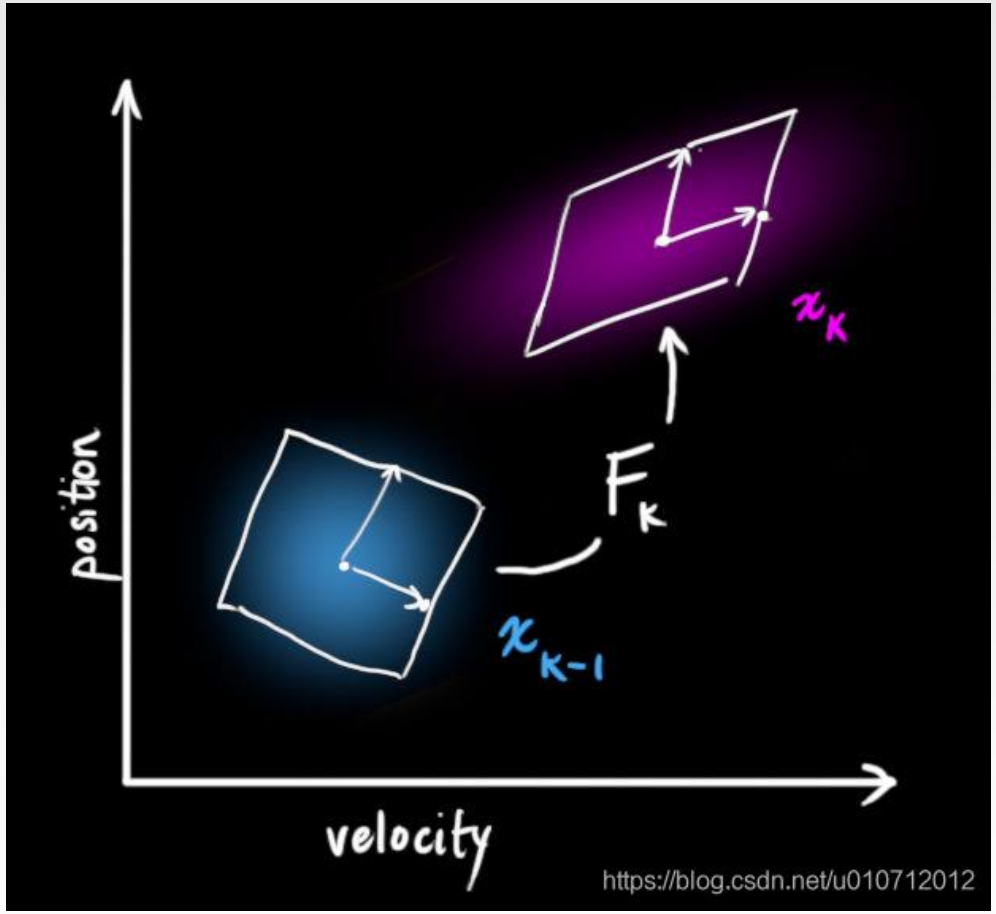
卡尔曼滤波讲解：

我们可以用矩阵F_k表示这个预测步骤：

它从原始预测中取每一点，并将其移动到新的预测位置。如果原始预测是正确的，系统就会移动到新位置。

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} \\ v_k &= v_{k-1} \end{aligned}$$

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



这是一个预测矩阵，它能给出机器人的下一个状态，但目前我们还不知道协方差矩阵的更新方法。这也是我们要引出下面这个等式的原因：如果我们将分布中的每个点乘以矩阵A，那么它的协方差矩阵会发生什么变化

$$\begin{aligned} Cov(x) &= \Sigma \\ Cov(\mathbf{A}x) &= \mathbf{A}\Sigma\mathbf{A}^T \end{aligned}$$

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

机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

外部影响：

除了速度和位置，外因也会对系统造成影响。比如模拟火车运动，除了列车自驾系统，列车操作员可能会手动调速。在我们的机器人示例中，导航软件也可以发出停止指令。对于这些信息，我们把它作为一个向量 u_k，纳入预测系统作为修正。

假设油门设置和控制命令是已知的，我们知道火车的预期加速度 a 。根据运动学基本定理，我们可得：

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} + \frac{1}{2} a \Delta t^2 \\ v_k &= v_{k-1} + a\Delta t \end{aligned}$$

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

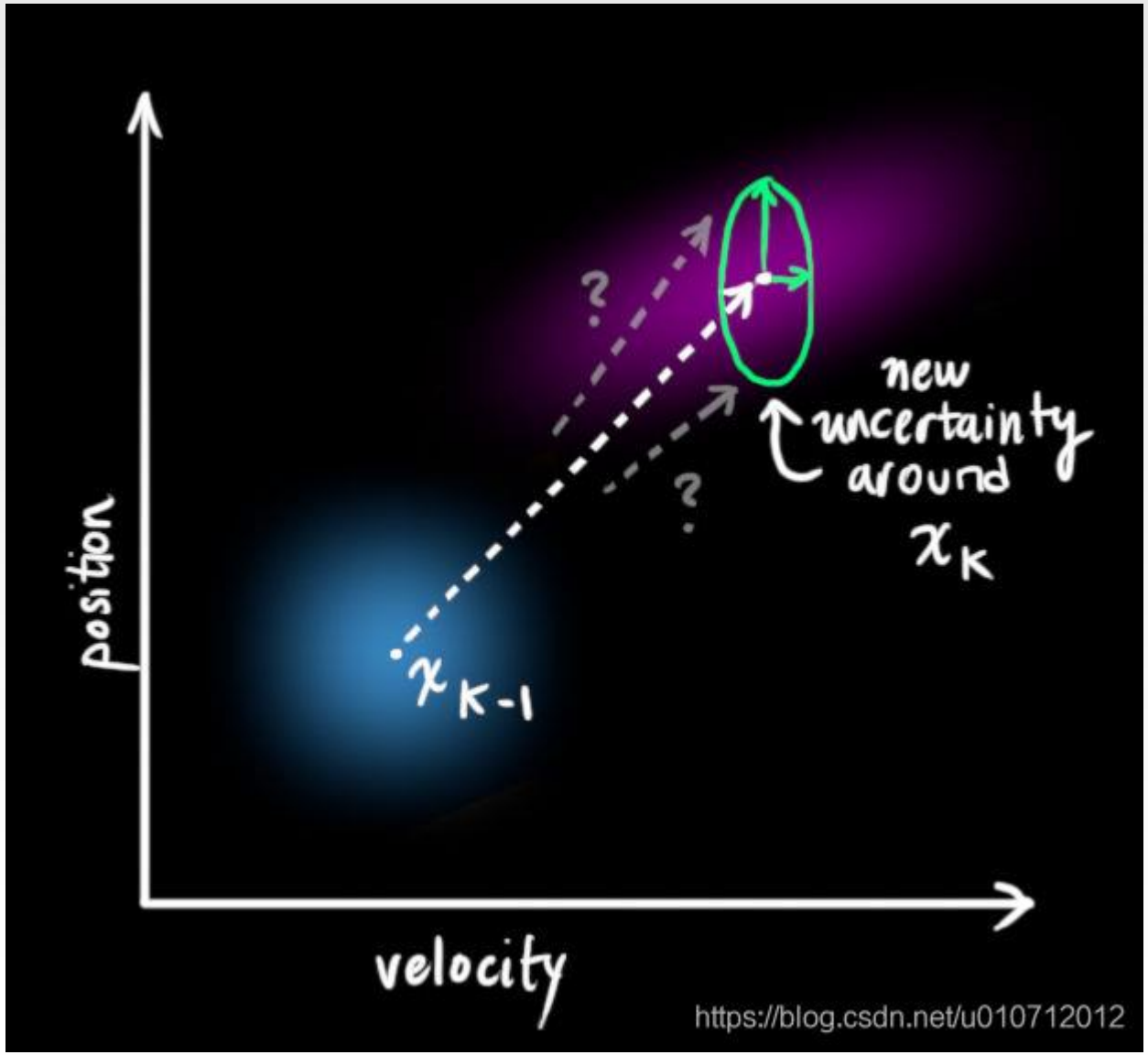
B_k是控制矩阵， $\vec{u}_k$是控制向量。如果外部环境异常简单，我们可以忽略这部分内容，但是如果添加了外部影响后，模型的准确率还是上不去，这又是为什么呢？

机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

外部不确定性：

我们跟踪轮式机器人时，它的轮胎可能会打滑，或者粗糙地面会降低它的移速。这些因素是难以掌握的，如果出现其中的任意一种情况，预测结果就难以保障。这要求我们在每个预测步骤后再加上一些新的不确定性，来模拟和“世界”相关的所有不确定性：

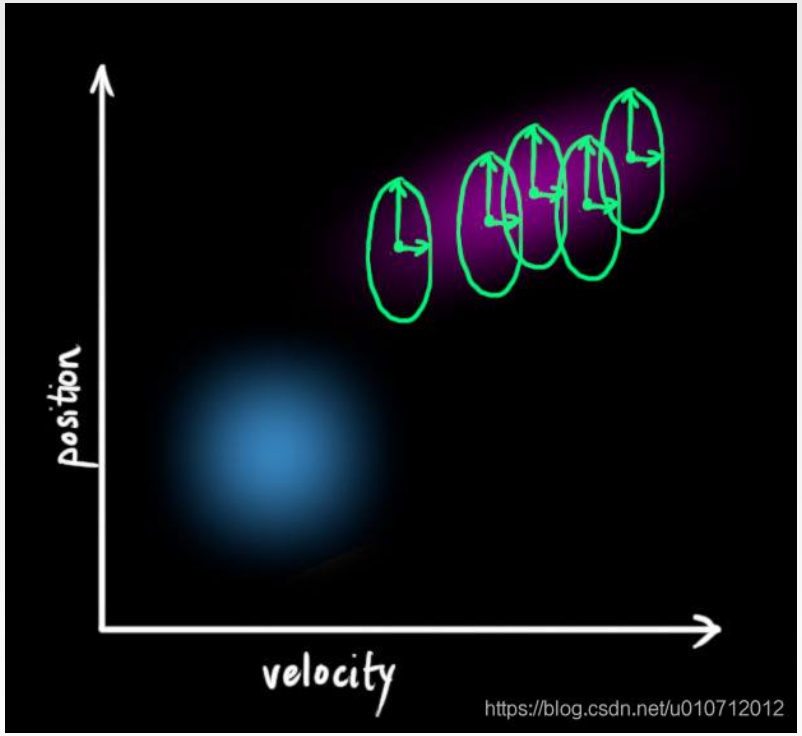


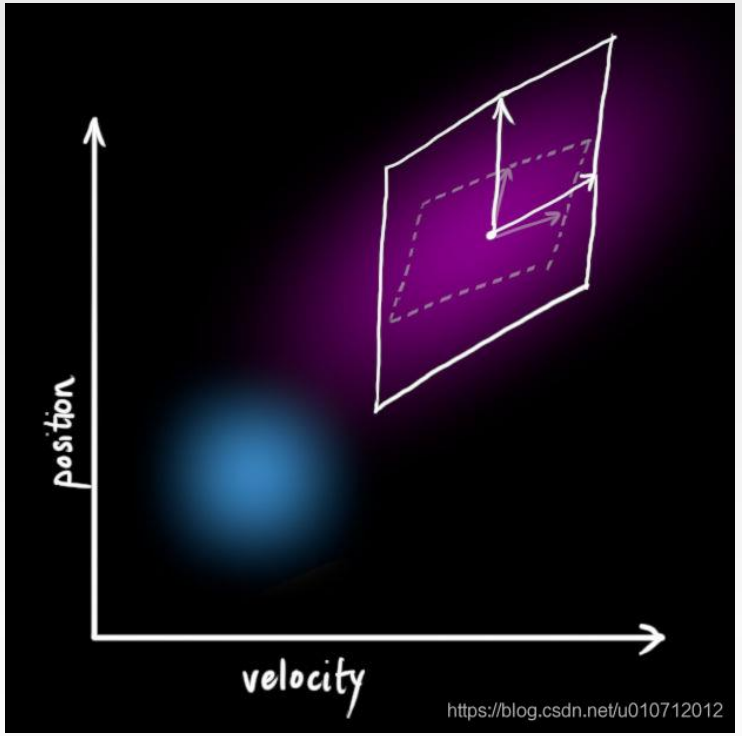
机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

外部不确定性：

如上图所示，加上外部不确定性后，$\hat{x}_{k-1}$ 的每个预测状态都可能会移动到另一点，也就是蓝色的高斯分布会移动到紫色高斯分布的位置，并且具有协方差Q_k换句话说，我们把这些不确定影响视为协方差Q_k的噪声。





这个紫色的高斯分布拥有和原分布相同的均值，但协方差不同。

机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

我们在原式上加入Q_k

$$\hat{\mathbf{x}}_k = \mathbf{F}_k\hat{\mathbf{x}}_{k-1} + \mathbf{B}_k\vec{\mathbf{u}}_k$$
$$\mathbf{P}_k = \mathbf{F}_k\mathbf{P}_{k-1}\mathbf{F}_k^T + \mathbf{Q}_k$$

新的**最佳估计**是基于**原最佳估计**和**已知外部影响**校正后得到的预测。

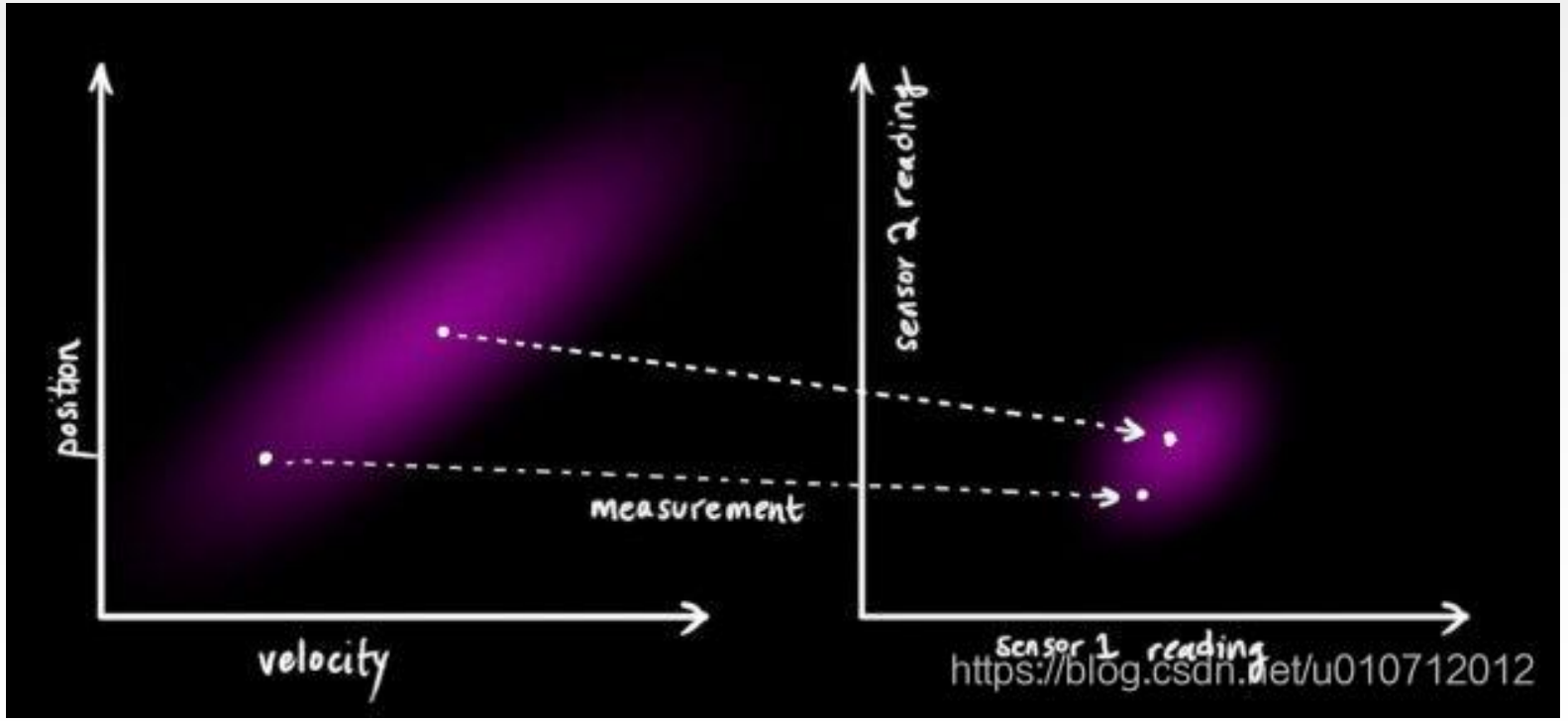
新的**不确定性**是基于**原不确定性**和**外部环境的不确定性**得到的预测。

机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

通过测量来细化估计值：

我们可能有好几个传感器，它们一起提供有关系统状态的信息。它可以读取位置，可以读取速度，通过它能告诉我们关于状态的间接信息——它是状态下产生的一组读数。

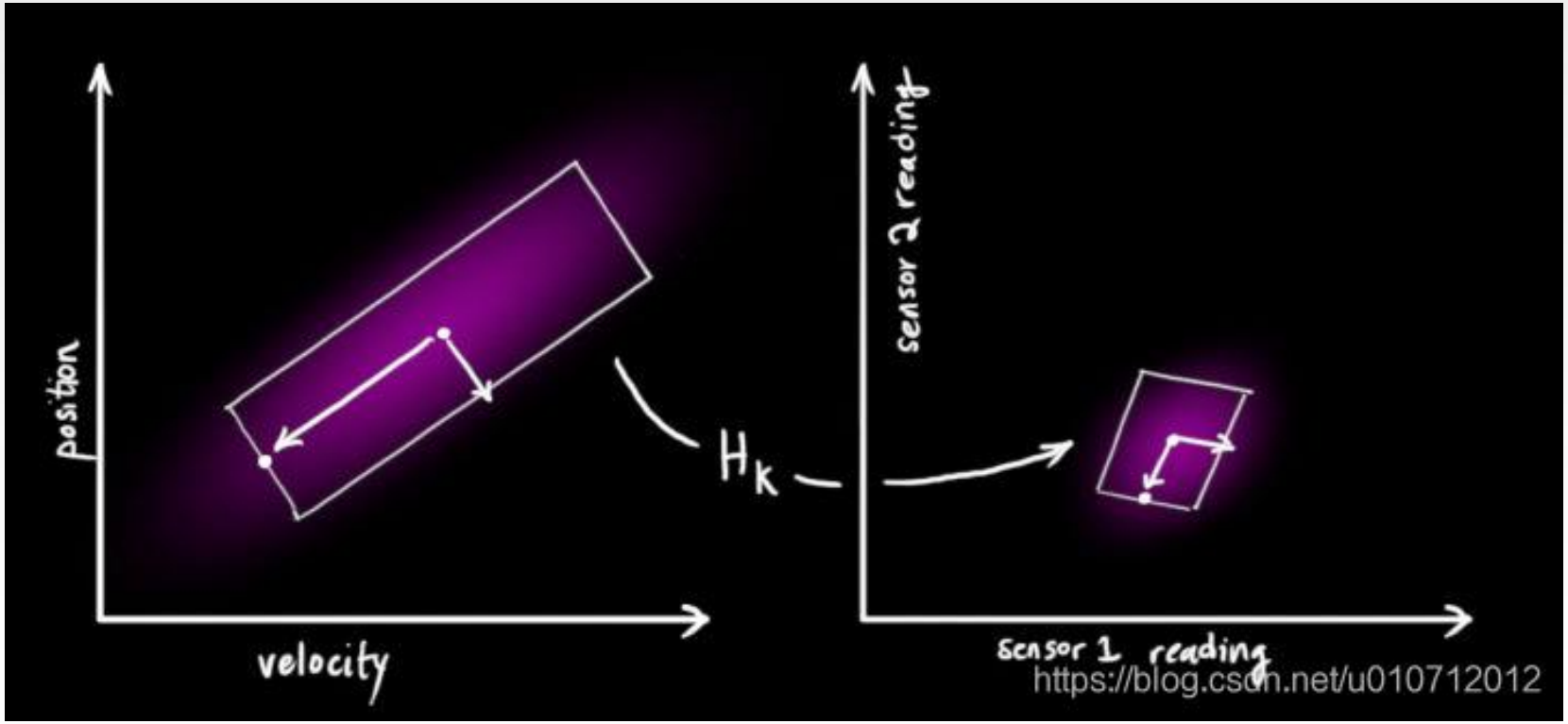


机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

通过测量来细化估计值：

读数的规模和状态的规模不一定相同，所以我们把传感器读数矩阵设为H_k



把这些分布转换为一般形式：

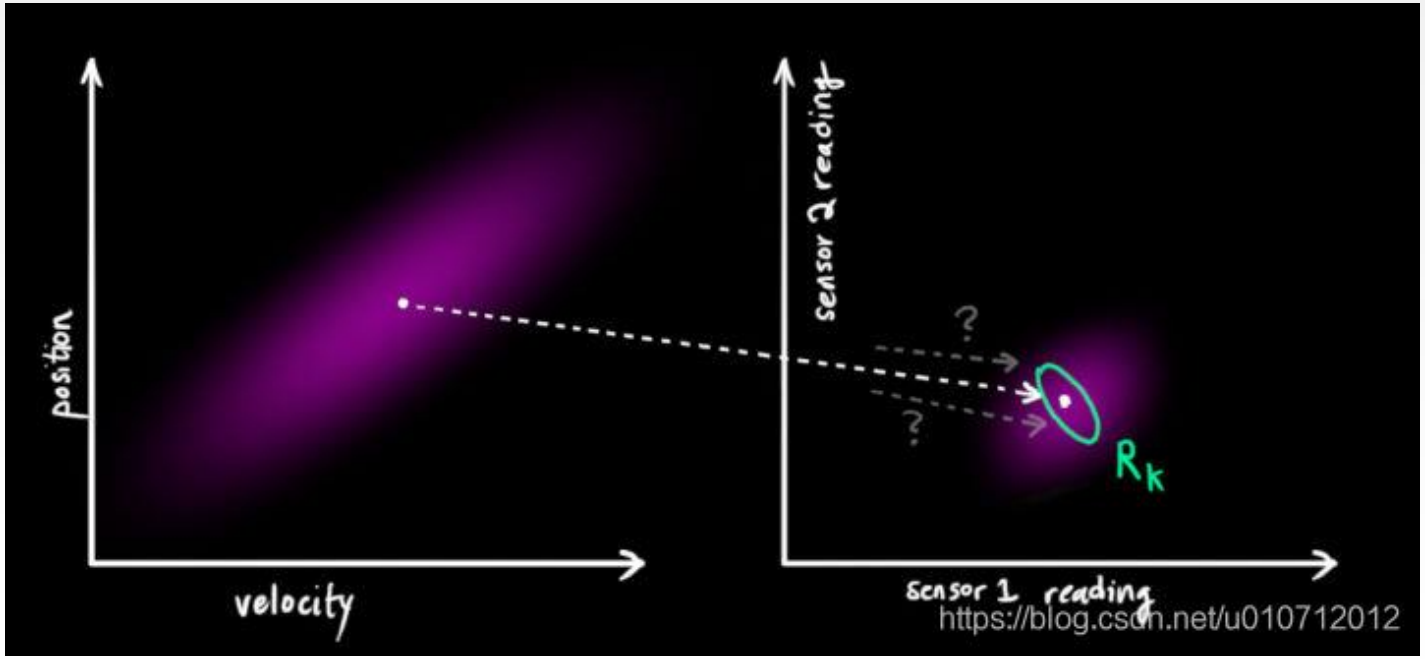
$$\vec{\mu}_{\text{expected}} = \mathbf{H}_k\hat{\mathbf{x}}_k$$
$$\mathbf{\Sigma}_{\text{expected}} = \mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T$$

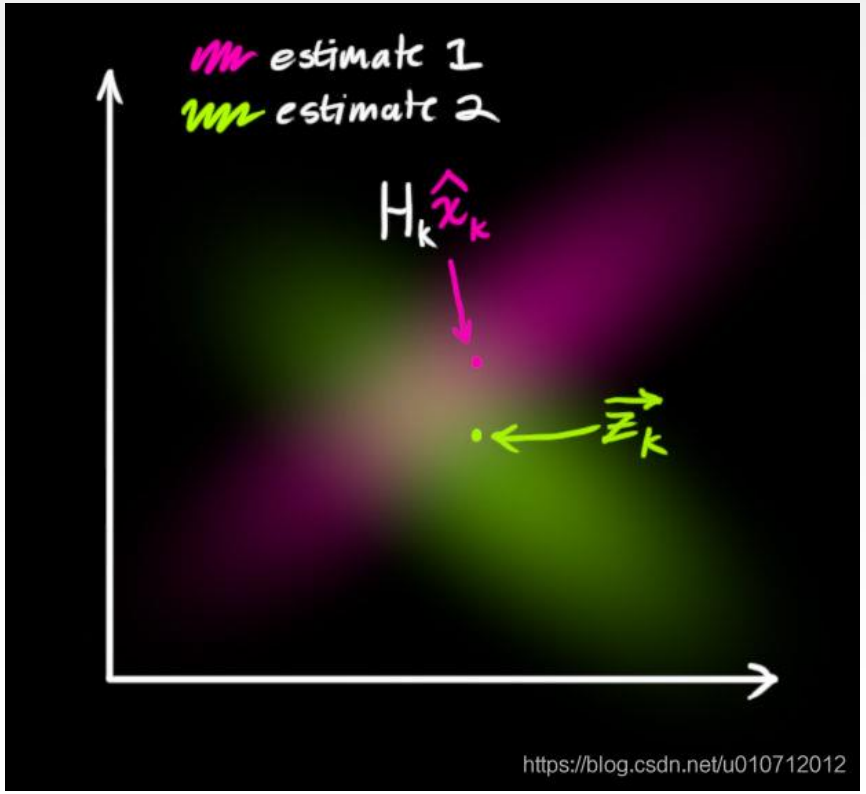
机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：　通过测量来细化估计值：

卡尔曼滤波的一大优点是擅长处理传感器噪声。换句话说，由于种种因素，传感器记录的信息其实是不准的，一个状态事实上可以产生多种读数。

我们将这种不确定性（即传感器噪声）的协方差设为R_k，读数的分布均值设为z_k。现在我们得到了两块高斯分布，一块围绕预测的均值，另一块围绕传感器读数。

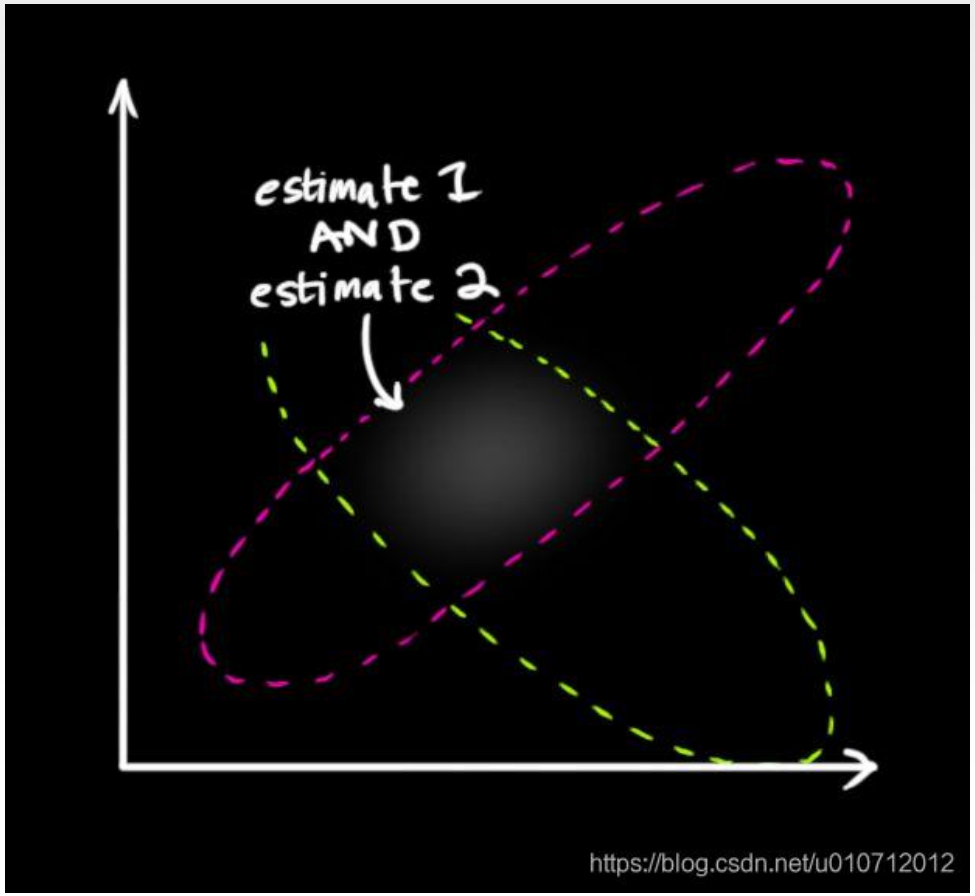




机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解： 通过测量来细化估计值：

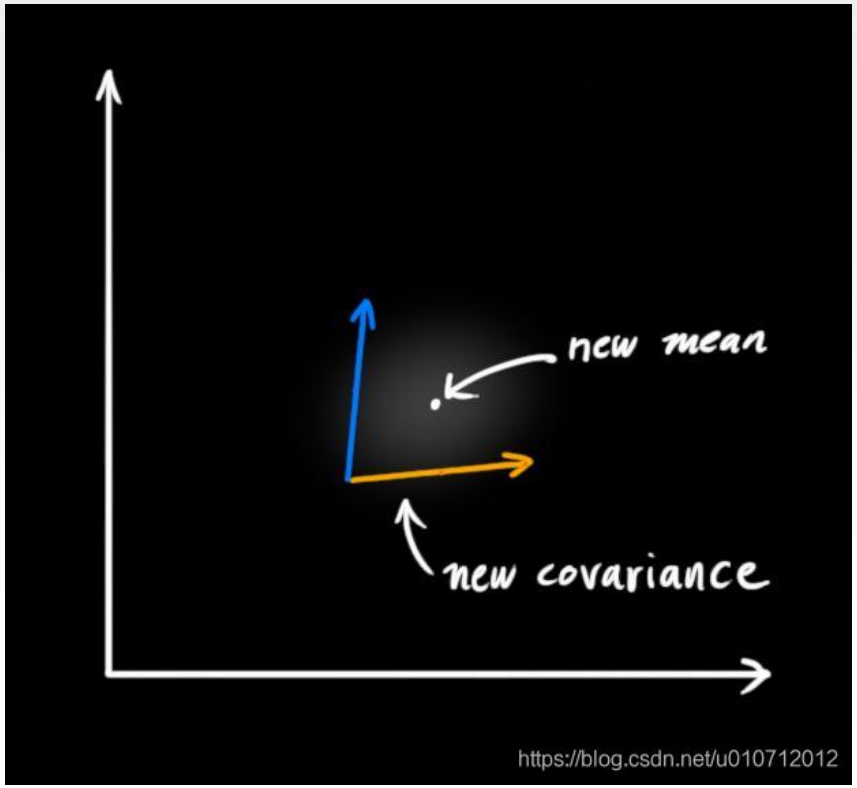
如果要生成靠谱预测，模型必须调和这两个信息。也就是说，对于任何可能的读数（z_1，z_2），这两种方法预测的状态都有可能是准的，也都有可能是不准的。重点是我们怎么找到这两个准确率。最简单的方法是两者相乘：



机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解： 通过测量来细化估计值：

两块高斯分布相乘后，我们可以得到它们的重叠部分，这也是会出现最佳估计的区域。换个角度看，它看起来也符合高斯分布：



事实证明，当你把两个高斯分布和它们各自的均值和协方差矩阵相乘时，你会得到一个拥有独立均值和协方差矩阵的新高斯分布。

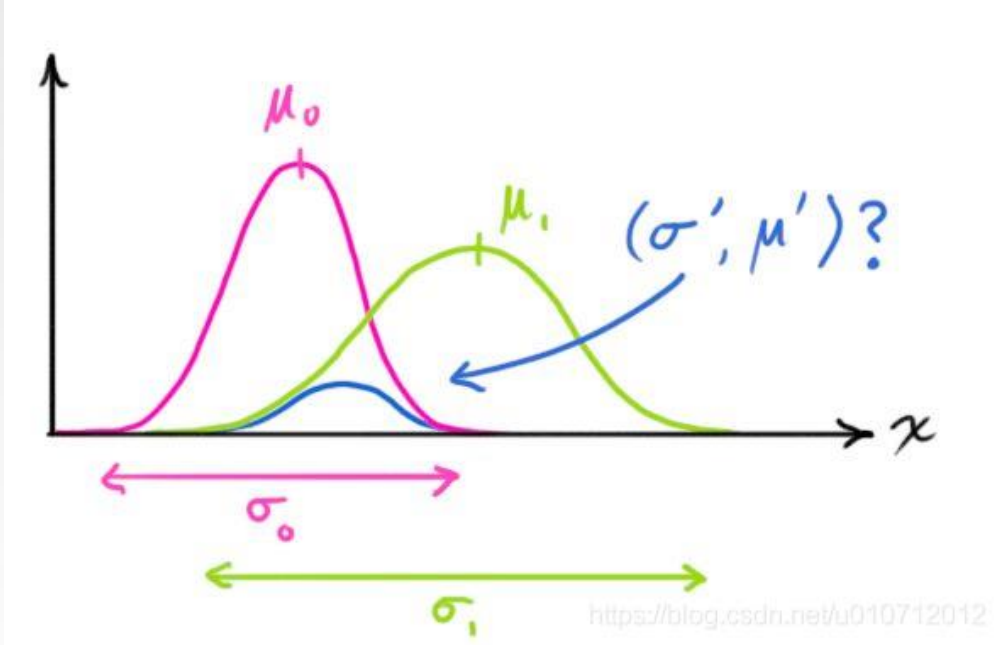
最后剩下的问题就不难解决了：我们必须有一个公式来从旧的参数中获取这些新参数！

机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：结合高斯

让我们从一维看起，设方差为 σ^2 ，均值为 μ ，一个标准一维高斯钟形曲线方程如下所示：

$$\mathcal{N}(x,\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



那么两条高斯曲线相乘呢？

$$\mathcal{N}(x,\mu_0,\sigma_0)\cdot\mathcal{N}(x,\mu_1,\sigma_1)\overset{?}{=}\mathcal{N}(x,\mu',\sigma')$$

把这个式子按照一维方程进行扩展，可得：

$$\mu'=\mu_0+\frac{\sigma_0^2(\mu_1-\mu_0)}{\sigma_0^2+\sigma_1^2}$$

$$\sigma'^2=\sigma_0^2-\frac{\sigma_0^4}{\sigma_0^2+\sigma_1^2}$$

我们用k简化一下：

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

$$\mu'=\mu_0+\mathbf{k}(\mu_1-\mu_0)$$

$$\sigma'^2=\sigma_0^2-\mathbf{k}\sigma_0^2$$

机器人定位技术—卡尔曼滤波

卡尔曼滤波讲解：

以上是一维的内容，如果是多维空间，把这个式子转成矩阵格式：其中K为卡尔曼增益。

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

$$\vec{\mu}' = \vec{\mu_0} + K(\vec{\mu_1} - \vec{\mu_0})$$

$$\Sigma' = \Sigma_0 - K\Sigma_0$$

截至目前，我们有用矩阵 $(\mu_0, \Sigma_0) = (H_k\hat{x}_k, H_k P_k H_k^T)$ 预测的分布

有用传感器读数 $(\mu_1, \Sigma_1) = (\vec{z}_k, R_k)$ 预测的分布 把它们代入矩阵等式中：

$$H_k\hat{x}_k' = H_k\hat{x}_k + K(\vec{z_k} - H_k\hat{x}_k)$$

$$H_k P_k' H_k^T = H_k P_k H_k^T - K H_k P_k H_k^T$$

卡尔曼增益就是

$$K = H_k P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}$$

考虑到 K 里还包含着一个H_k，我们再精简一下上式：

$$\hat{x}_k' = \hat{x}_k + K'(\vec{z_k} - H_k\hat{x}_k)$$

$$P_k' = P_k - K' H_k P_k$$

$$K' = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}$$

$\hat{x}_k'$ 是我们的最佳估计值，我们可以把它继续放进去做另一轮预测：

机器人学中的地图

如果想解决航迹推算过程中位姿不确定性发散增长的问题，最有效的方法是观察周围环境中具有固定位置的特征物体，利用这些物体的信息对航迹推算结果进行修正，这就是通常所说的引入路标的概念。举一个很简单的例子，

假设某人每走一步能够前进0.5m，从A点运动到B点走了100步，如果不考虑方向偏移，利用航迹推算应该前进50m。此时观察到起点A正好在1号路灯下方，终点B正好在2号路灯下方，1号路灯与2号路灯的距离间隔为45m，那么就可以利用路灯的信息(45m)对航迹推算结果(50m)进行修正，得到前进距离的最优化估计。

机器人在运动过程中使用的地图，本质上就是包含了大量路标位置信息的数据集。

机器人学中的地图—使用地图

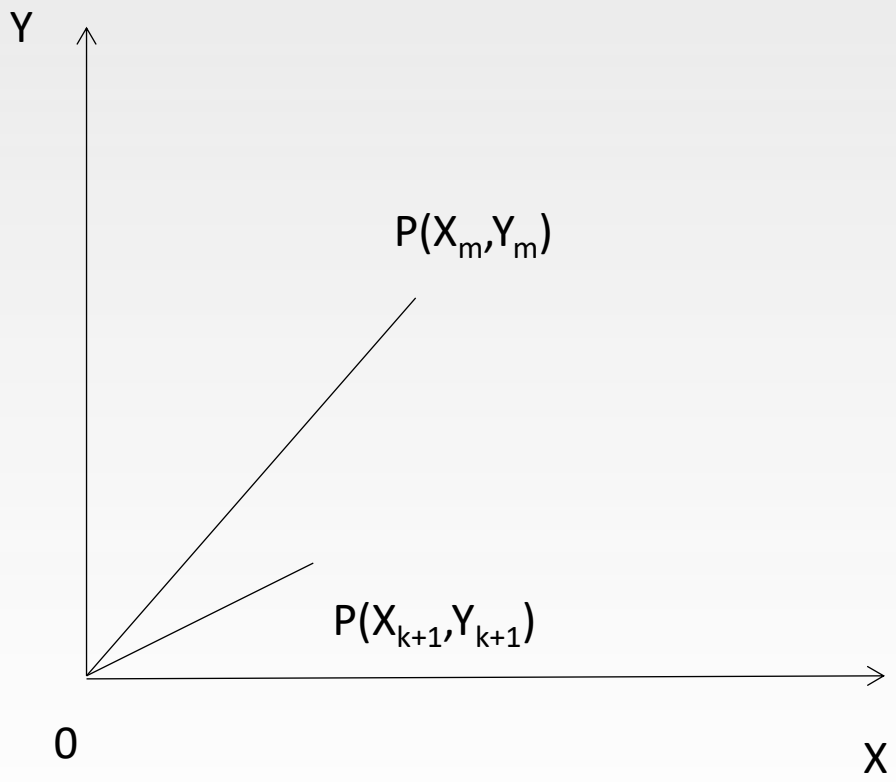
首先假设机器人配置有外部传感器用于检测路标，如三角测距激光雷达，那么测量得到的路标相对于机器人的位置信息可以表示为：

$$z_k = h(q_k, q_m, v_k)$$

其中$q_k \in R^3$是机器人的位置状态向量，$q_m \in R^2$是路标在全局坐标系中的位置信息，v是传感器的测量误差。注意，q_k是三维向量，因为该位置向量除了坐标信息x、y,还包括姿态θ（角度）信息，而q_m仅是地标的位置坐标信息。

机器人学中的地图—使用地图

下面讨论函数h的具体形式，通过激光雷达对路标的测量，得到与机器人之间的距离信息，以及方位信息：



$$z_{k+1}=\begin{bmatrix}\sqrt{(x_m-x_{k+1})^2+(y_m-y_{k+1})^2}\\ \arctan\dfrac{y_m-y_{k+1}}{x_m-x_{k+1}}-\theta_{k+1}\end{bmatrix}+\begin{bmatrix}v_r\\ v_\alpha\end{bmatrix}$$

其中$Z_{k+1}=[z_r\ \ z_a]^T$为系统状态空间模型在k+1时刻的输出向量，这里通过外部传感器测量得到，z_r是路标相对机器人位置的距离，z_a是相对机器人姿态的方位角。测量噪声向量为$v_k=[v_r\ \ v_a]$，满足高速随机分布v_k–N(0, V)，协方差矩阵V为对角矩阵：

$$V=\begin{bmatrix}\sigma_r^2 & 0\\ 0 & \sigma_\alpha^2\end{bmatrix}$$

机器人学中的地图—使用地图

由于z_{k+1}为非线性的输出方程，运用线性处理方法可以得到：

$$z_{k+1} = h\left(q_{k+1|k}, q_m, 0\right) + H_q\left(q_{k+1} - q_{k+1|k}\right) + H_v v$$

$H_q=\partial h/\partial q$和$H_v=\partial h/\partial v$为雅可比矩阵

定义一个测量误差 $\tilde{z}_{k+1} = z_{k+1} - h\left(q_{k|k}, q_m, 0\right) = H_q\left(q_k - q_{k|k}\right) + H_v v$

其代表传感器实际测量值与传感器预测值之间的差。卡尔曼滤波正是利用这个向量修正预测状态$q_{k+1|k}$与状态不确定性矩阵$P_{k+1|k}$.

这里还需要定义一个卡尔曼增益矩阵K_{K+1}：

$$K_{k+1} = P_{k+1|k} H_q^{\mathrm{T}}\left(H_q P_{k+1|k} H_q^{\mathrm{T}} + H_v V H_v^{\mathrm{T}}\right)^{-1}$$

修正上节的预测方程得到最优化估算值：

$$q_{k+1|k+1} = q_{k+1|k} + K_{k+1}\tilde{z}_{k+1}$$

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1} H_q P_{k+1|k}$$

从第二个式子可以看出，因为减号的关系，协方差矩阵更新之后是有可能减小的，这就意味着，状态量的不确定性也会变小，表明利用地标测量值对航迹推算预测状态的修正是必要的。

机器人学中的地图—例

首先假设机器人的最大线速度为0.5m/s，最大角速度为0.5rad/s，即在每个采样周期(0.1s)内最远前进0.05m，航向角最多改变约28.66°。里程计噪声设定为g=0.02m，ag=0.5°。得到式中的**协方差矩阵W**：

$$\boldsymbol{W}=\begin{bmatrix}\sigma_d^2 & 0\\ 0 & \sigma_\theta^2\end{bmatrix}$$

```
>> sigma_d=0.02; sigma_theta=0.5*pi/180;
>> W=diag([sigma_d sigma_theta].^2);
```

机器人学中的地图—例

状态向量的预测方程式可以使用下面的函数实现：

$$f(\boldsymbol{q}_k,\Delta_k,\boldsymbol{w})=\begin{bmatrix} x_k+(\Delta d_k+w_d)\cos\theta_k \\ y_k+(\Delta d_k+w_d)\sin\theta_k \\ \theta_k+(\Delta\theta_k+w_\theta) \end{bmatrix}$$

```
>> function [ state_next ] = state_step( state_k, delta_v, delta_theta)
>>       state  next=zeros(3,1);
>>       state_next(1)=state_k(1)+delta_v*cos(state_k(3));
>>       state_next(2)=state_k(2)+delta_v*sin(state_k(3));
>>       state_next(3)=state_k(3)+delta_theta;
>> end
```

输入state_k、delta_v和delta_theta分别为qr、△dk和△θk，该函数有以下3种用途：

①在只有里程计的情况下，估计每一个采样时刻的状态向量；

②模拟机器人的真实状态，此时里程计值可以设置为

delta_v(k)+normrnd(0,sigma_d),delta_theta(k)+normrmd(0,sigma_theta),

即加上满足正态分布的随机噪声；

③基于当前时刻的最优估计值，预测下一时刻的状态向量并等

待优化修正。

机器人学中的地图—例

预测协方差矩阵P由以下函数得到：

$$q_{k+1|k} = f\left(q_{k|k}, \Delta_k, 0\right)$$

$$P_{k+1|k} = F_q P_{k|k} F_q^{\mathrm{T}} + F_w W F_w^{\mathrm{T}}$$

```
>> function [ P_next ] = P_step( P_k, delat_v, state_3,W )
>>       F_q=[1  0  -delat_v*sin(state_3);
>>            0  1  delat_v*cos(state_3);
>>            0  0  1                  ];
>>       F_w=[cos(state_3)    0;
>>            sin(state_3)    0;
>>            0               1];
>>       P_next=F_q*P_k*F_q'+F_w*W*F_w';
>> end
```

其中Pk是当前时刻协方差矩阵的最优估计，state 3是当前时刻机器人角度θ的最优估计（“_3”表示向量q的第3个元素）。接着考虑输出方程中的雅可比矩阵H和Hy，设置地标：

机器人学中的地图—例

接着考虑输出方程中的雅可比矩阵H和Hy，设置地标：

```
>> Mark=[2;2];
```

机器人到地标距离的估计值为：

```
>> r_pre=((Mark(1)-state_opt_next(1))^2+(Mark(2)-state_opt_next(2))^2)^0.5;
```

这里的state_opt_next是用state_step()函数预估的状态qk+1|k，于是雅可比矩阵为：

```
>> H_q=[-(Mark(1)-state_opt_next(1))/r_pre  -(Mark(2)-state_opt_next(2))/r_pre  0;
    (Mark(2)-state_opt_next(2))/r_pre^2 -(Mark(1)-state_opt_next(1))/r_pre^ 2-1];
>> H_v=[1 0;0 1];
```

由于没有外部传感器，因此实际测量值可以模拟表示为：

```
>> z_real=[((Mark(1)-state_next_real(1))^2+(Mark(2)-state_next_real(2))^2)^0.5;
        atan2(Mark(2)-state_next_real(2),Mark(1)-state_next_real(1))-state_
        next_real(3)]+[normrnd(0,miu_r);normrnd(0,miu_alfa)];
```

这里的state_next_real是通过state_step()函数模拟得到的机器人真实状态，语句最后加上了满足正态分布的测量噪声。

机器人学中的地图—例

预估的测量值则为：

```
>> z_pre=[r_pre;
       atan2(Mark(2)-state_opt_next(2),Mark(1)-state_opt_next(1))-state_opt_next(3)];
```

于是可以得到卡尔曼增益矩阵：

```
>> K_next=P_opt_next*H_q'/(H_q*P_opt_next*H_q'+H_v*V*H_v');
```

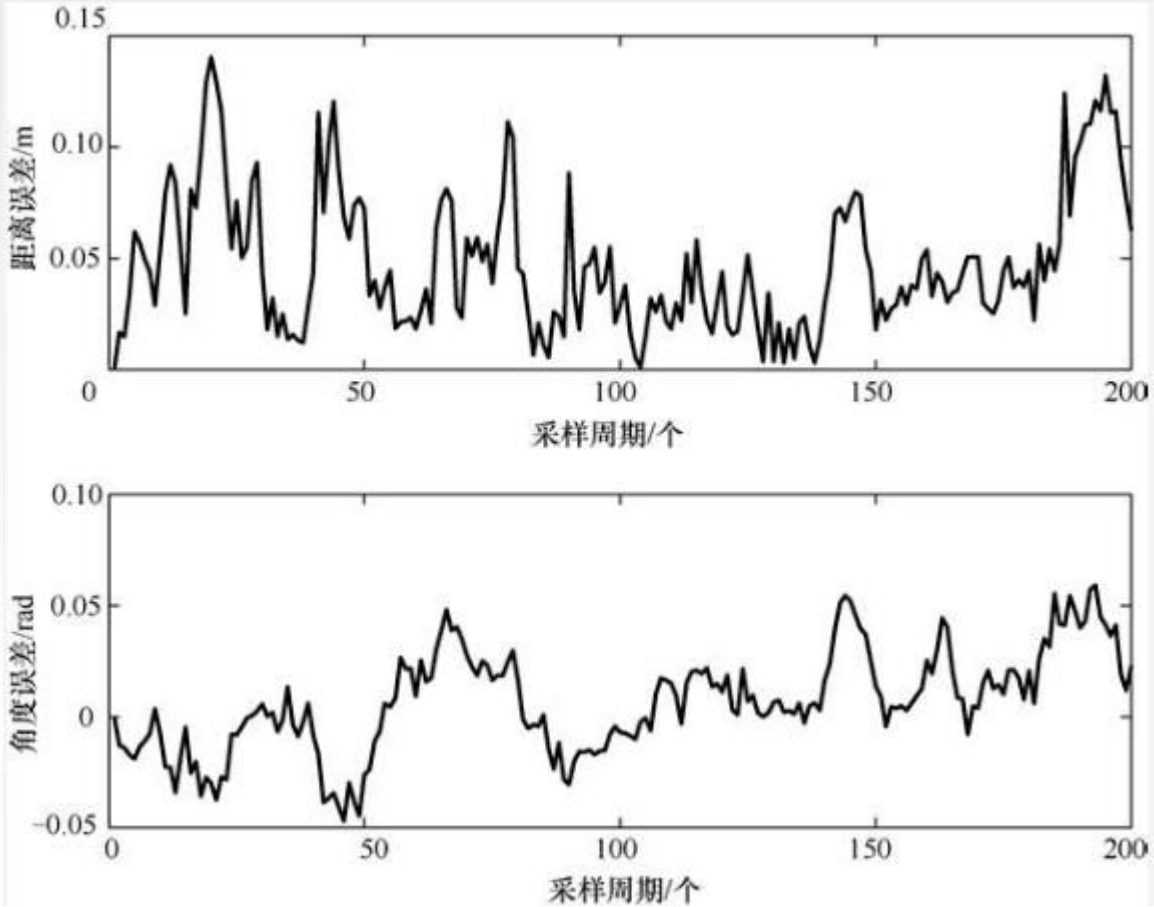
这里的P_opt_next是通过P_step()函数预估的$P_{k+1|k}$

最后，完成状态预估qk+1和协方差矩阵预估Pk+1k的优化修正：

```
>> state_next_opt=state_opt_next+K_next*(z_real-z_pre);
>> P_next_opt=P_opt_next-K_next*H_q*P_opt_next;
```

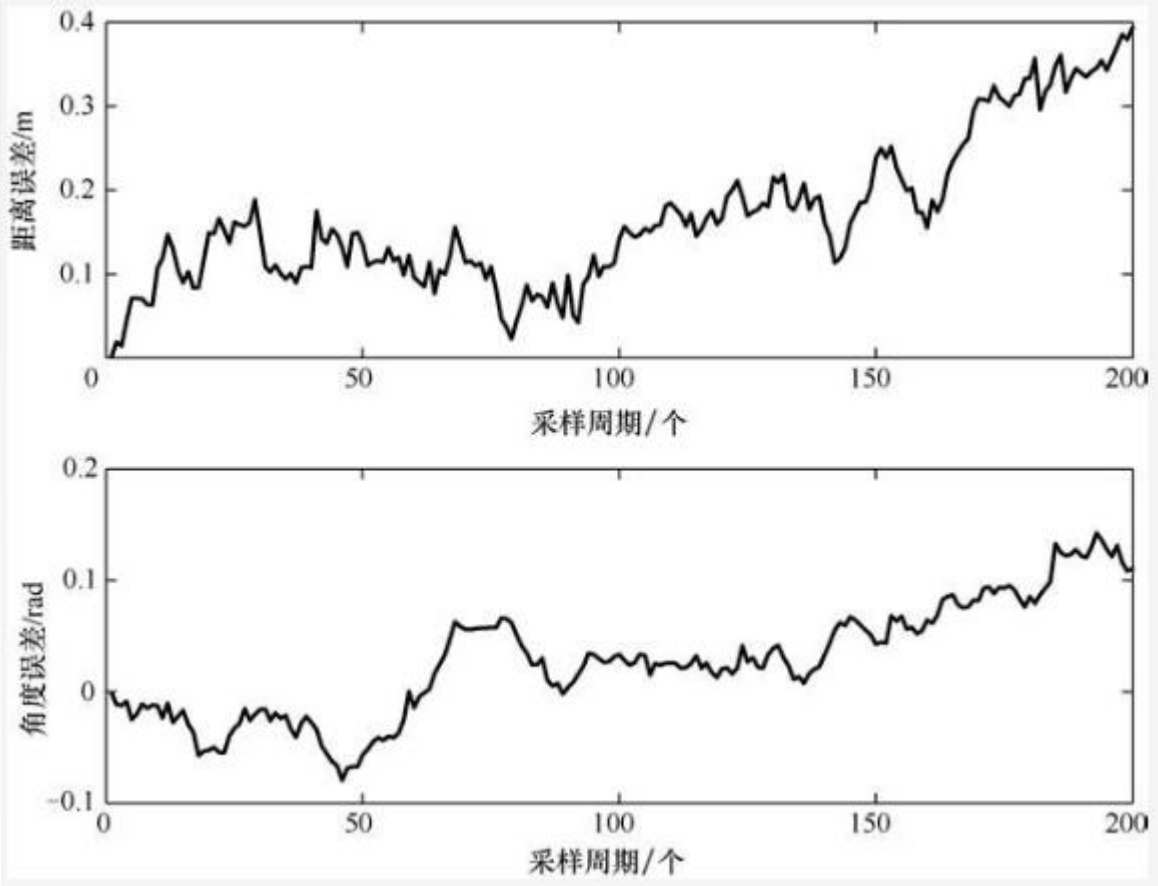
机器人学中的地图—例

在每个采样周期内，里程计数值设为不超过最大值的随机数，仿真程序运行200个周期后，最优估计运动轨迹与模拟真实路径之间的距离与航向角度误差如图所示。



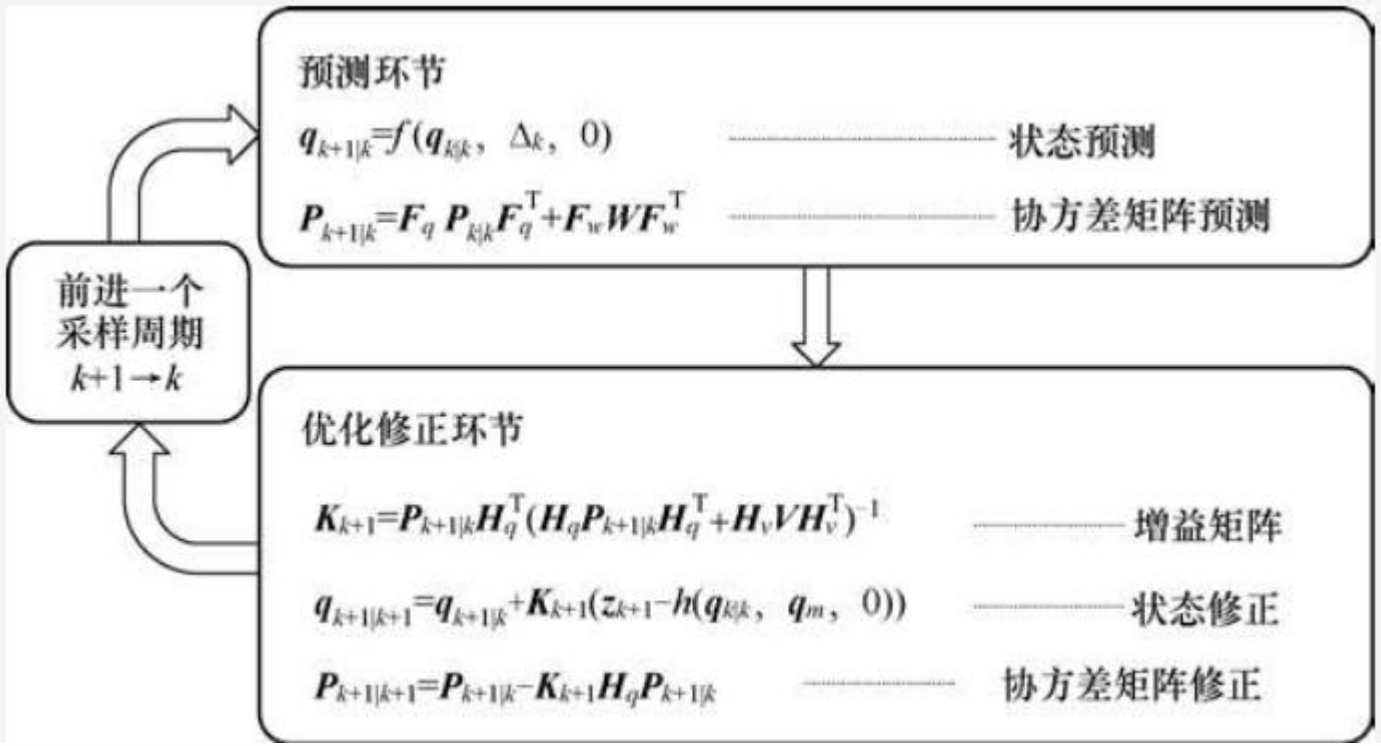
机器人学中的地图—例

在相同情况下，仅依靠航迹推算方法得到的仿真结果如图所示，可以看到，在没有用扩展卡尔曼滤波的情况下误差明显更大，而且呈现出发散扩大的趋势。



机器人学中的地图—使用地图

总结以上内容，容易发现扩展卡尔曼滤波（EKF）方法由“预测”和“优化修正”两部分构成，如图所示。这里的卡尔曼滤波增益矩阵K至关重要，它实现了地标测量值对状态向量和协方差矩阵的修正，这也正是EKF方法相比于纯航迹推算方法的优势所在。



机器人学中的地图—创建地图

在上节中我们假设地标的位置和身份信息都是已知的。在现实生活中，我们到陌生环境有导航地图可以直接使用。但机器人来到一个陌生的环境中，就需要在移动的过程中逐渐完成地图的绘制，下面考虑具体的方法与实施过程。

假设机器人配备外部传感器，能够测量地标相对于自身的距离和方位角。除此之外，假设环境中存在N个地标（或者叫作特征物体），并且机器人能够清楚地辨别每个地标的身份信息。从而估计地标的位置坐标，于是定义以下的状态向量：

$$\boldsymbol{M}=\left[\left(\boldsymbol{q}_m^1\right)^{\mathrm{T}} \quad \left(\boldsymbol{q}_m^2\right)^{\mathrm{T}} \quad \cdots \quad \left(\boldsymbol{q}_m^i\right)^{\mathrm{T}}\right]^{\mathrm{T}}$$

其中 $\boldsymbol{q}_m^i=\left[x_m^i \quad y_m^i\right]^{\mathrm{T}}$ 表示第个地标的位置向量，i=1, 2……. N。可以看到，每探测到一个新的地标，向量M的维度就加2，直到 $M\in\mathbf{R}^{2N}$，对应的协方差矩阵也变为 $P\in\mathbf{R}^{2N\times 2N}$。

机器人学中的地图—创建地图

1. 由于地标的位置是固定的，所以预测方程相对简单

$$\boldsymbol{M}_{k+1|k}=\boldsymbol{M}_{k|k}$$
$$\boldsymbol{P}_{k+1|k}=\boldsymbol{P}_{k|k}$$

2. 利用机器人的位姿和传感器的测量值计算地标的位置：

$$g(\boldsymbol{q},\boldsymbol{z})=\begin{bmatrix} x+z_r\cos(\theta+z_\alpha) \\ y+z_r\sin(\theta+z_\alpha) \end{bmatrix}$$

状态向量M的维度会随着地标的检测而不断增加，于是引入一个函数Y(•)，用来将现有的向量加长变成新的向量：

$$\begin{aligned} \boldsymbol{M}_{k|k}^{*} &= Y(\boldsymbol{M}_{k|k},\boldsymbol{z}_k,\boldsymbol{q}_{k|k}) \\ &= \begin{bmatrix} \boldsymbol{M}_{k|k} \\ g(\boldsymbol{q}_{k|k},\boldsymbol{z}_k) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{M}_{k|k} \\ x_{k|k}+z_r\cos(\theta_{k|k}+z_\alpha) \\ y_{k|k}+z_r\sin(\theta_{k|k}+z_\alpha) \end{bmatrix} \end{aligned}$$

机器人学中的地图—创建地图

新的地标位置坐标被加在向量的最后，该向量中地标的顺序就是它们被测量到的先后顺序。状态向量延长的同时，协方差矩阵也需要扩展，这里需要用到雅可比矩阵：

$$\boldsymbol{P}_{k|k}^{*}=\nabla Y_{\boldsymbol{M},z}\begin{bmatrix}\boldsymbol{P}_{k|k} & 0\\ 0 & \boldsymbol{V}\end{bmatrix}\nabla Y_{\boldsymbol{M},z}^{\mathrm{T}}$$

其中：

$$\nabla Y_{\boldsymbol{M},z}=\begin{bmatrix}\boldsymbol{I}^{n\times n} & 0^{n\times 2}\\ \dfrac{\partial g}{\partial \boldsymbol{M}} & \dfrac{\partial g}{\partial \boldsymbol{z}}\end{bmatrix}$$

这里，n表示向量M扩展前的维度。由于函数g(•)与扩展前的状态向量无关联，所以∂g/∂M=0，于是得到：

$$\boldsymbol{P}_{k|k}^{*}=\begin{bmatrix}\boldsymbol{P}_{k|k} & 0\\ 0 & \boldsymbol{G}_z\boldsymbol{V}\boldsymbol{G}_z^{\mathrm{T}}\end{bmatrix}$$

$$\boldsymbol{G}_z=\frac{\partial g}{\partial \boldsymbol{z}}=\begin{bmatrix}\cos\left(\theta_{k|k}+z_\alpha\right) & -z_r\sin\left(\theta_{k|k}+z_\alpha\right)\\ \sin\left(\theta_{k|k}+z_\alpha\right) & z_r\cos\left(\theta_{k|k}+z_\alpha\right)\end{bmatrix}$$

通过计算，可得到雅可比矩阵：

$$\boldsymbol{H}_{q_m}=\begin{bmatrix}\cdots 0\cdots & \dfrac{\partial h}{\partial \boldsymbol{q}_m^i} & \cdots 0\cdots\end{bmatrix}$$

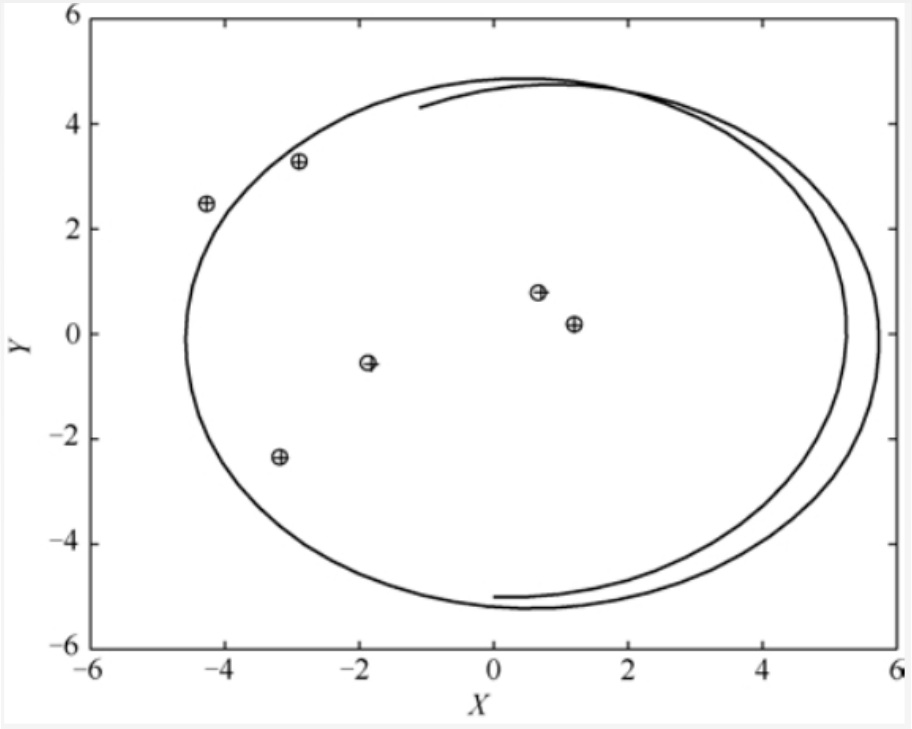
该矩阵中大部分元素为0，因为该信息只依赖当前检测到的某一个地标，与其他地标无关。

机器人学中的地图—创建地图

依照上诉方程式，

设仿真地图条件为：地图的大小设为“10”，随机生成6个地标，地标被限制在以原点为中心，边长是10m的正方形内。

地图构建仿真结果如图3所示。地标的真实位置用“+”号表示，估计位置用“0”表示，实线为机器人的行走路径。可以看出，虽然存在测量噪声，但是估计位置与真实位置之间的误差还是比较小的。



机器人导航技术

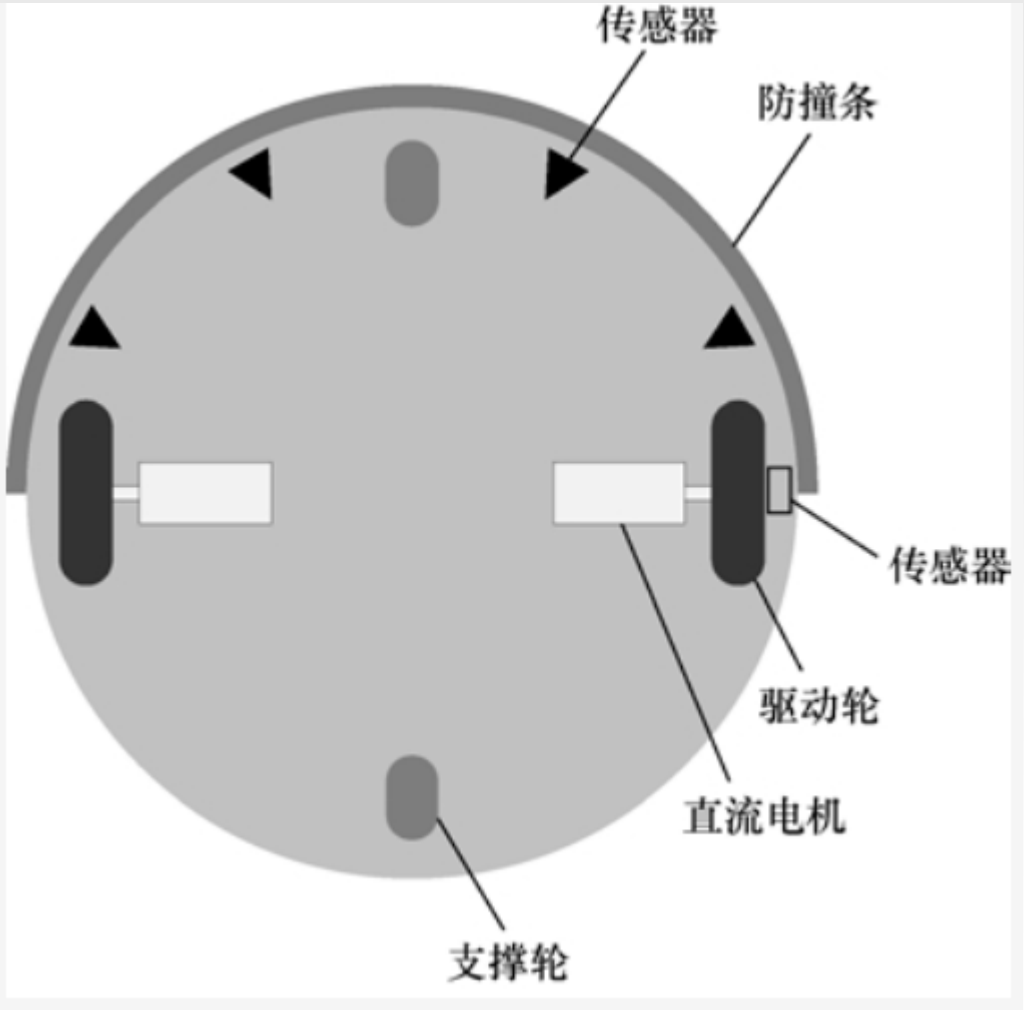
所谓导航，就是引导一个运载工具到达预定位置的方法。

生活中最常见的是汽车导航（软件），通过利用GPS卫星导航系统或者我国自行研制的北斗卫星导航系统，能够在终端硬件（如手机）上进行定位，并引导车辆在城市或者乡间道路上行驶。

在机器人所处的地图中同样需要规划出道路，这就需要用到一些数学方法，这个过程称为路径规划。反之，假如出现无法建立地图的情况，机器人还可以依靠自身感觉和反应移动，比如，借助传感器探测前方是否有障碍物，改变方向沿着随机路径行走，这种方式称为**反应式导航**。

机器人导航技术-反应式导航

在实际生活中，最常见的例子就是扫地机器人。右图是一种最常见的扫地机器人的简单结构示意图，左右两个驱动轮由直流电机提供动力，前后两个支撑轮仅负责维持机身平衡。分布在前端的传感器主要有两种：免碰撞传感器与碰撞传感器。免碰撞传感器一般由一对红外线发射管与接收管组成，用来探测前方一定距离内是否有障碍物。碰撞传感器一般由微动开关构成，并与防撞条相连接，当机器人与障碍物接触时作出响应。另外，在机器人侧面还安装有沿墙传感器，通常选用位置敏感检测器（position sensitive detector，PSD），检测机器人一侧与墙壁的距离。



机器人导航技术-反应式导航

扫地机器人的工作目标是尽可能地将行走路径覆盖整个可行区域。以市面上不配备全局传感器（如激光雷达)的扫地机器人为例，主要有以下几种行走方式。

反弹方式

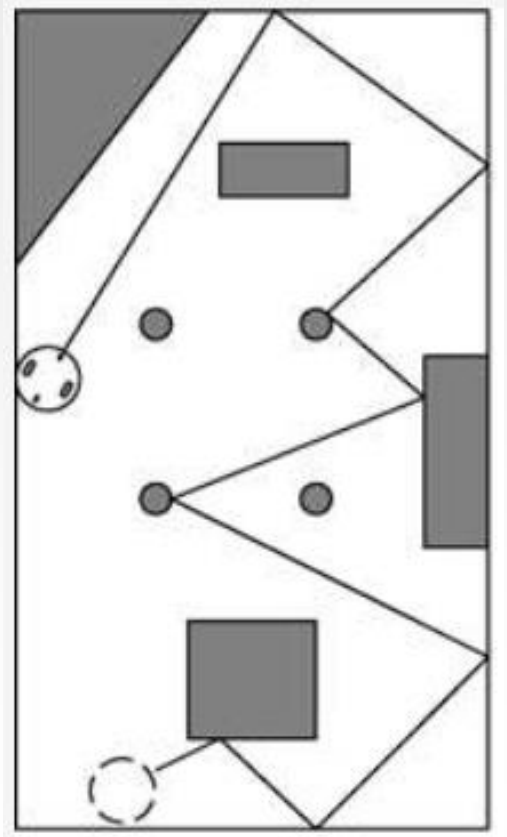
“弓”字形方式

沿墙方式

机器人导航技术-反应式导航

★反弹方式

机器人保持左右驱动电机转速相同，实现直线行走。当前端免碰撞或者碰撞传感器触发，都表示遇到障碍物，此时依靠左右轮速度差改变机器人前进角度，然后继续直线行走。

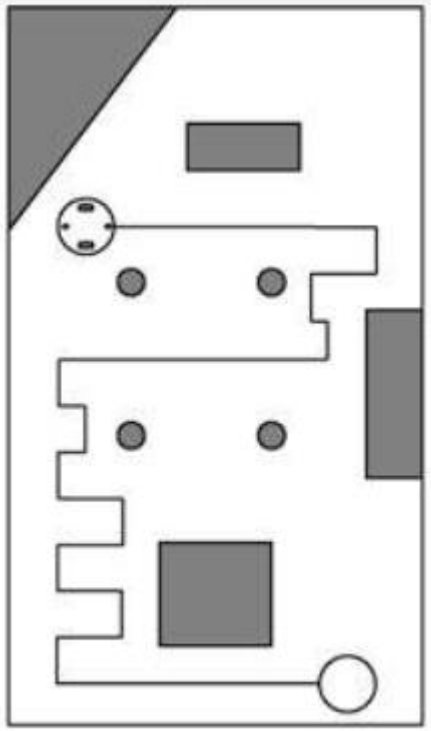


（a）反弹方式

这种类似光遇到镜面反射的方式，其优点是可以让机器人有更大的概率能够进入整个可行区域的各个角落，提高清扫覆盖率。而其缺点是不同的直线路径间容易出现呈多边形的空白漏扫区域。

★“弓”字形方式

首先，机器人保持直线行走，当遇到障碍物时，向左侧（或右侧）转动90°，前进一小段距离后，再次左转（或右转）90°，然后继续直行。当再次遇到障碍物时，重复以上动作，最终实现对空白地面的往返梳状清扫。该方式需要机器人配备陀螺仪之类的角度传感器，用来对转动的角度进行测量，以保证来回直线路径尽可能地平行。



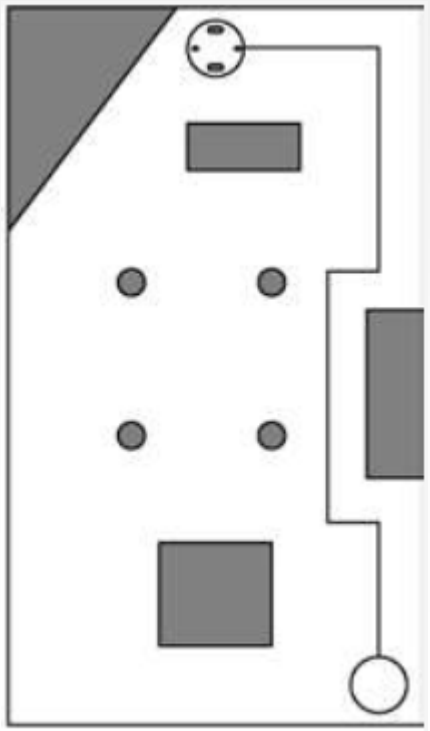
（b）“弓”字形方式

其优点是能够减少路径之间的遗漏区域，针对大面积的空白区域，清扫过程表现得比较规整，缺点是降低了机器人进入某些角落区域的概率。

机器人导航技术-反应式导航

★ 沿墙方式

此种行走方式一般作为以上两种方式的辅助方式，当机器人经过一段时间的反弹或“弓”字形清扫后，会试图沿着区域的外围进行清扫，此举的目的是对一些难以进入的角落或者墙边进行补漏，主要依靠机器人一侧的PSD传感器检测与墙面的距离并反馈，调节左右轮转速，使机器人在前进过程中与墙面保持平行。当然，此时机器人无法感知侧面的物体是墙体还是家具的底部边沿。



（c）沿墙方式

通过对以上三种方式的观察，可以发现扫地机器人下一时刻的导航动作完全依赖当前时刻传感器的检测信号，而且所获得的信号类型相对简单，一般为电压的数字量或者模拟量，相比于激光雷达等传感器的点云数据，其处理过程十分简单，通常如STM32F108系列的低端芯片完全能够胜任，这也体现出反应式导航的优势。

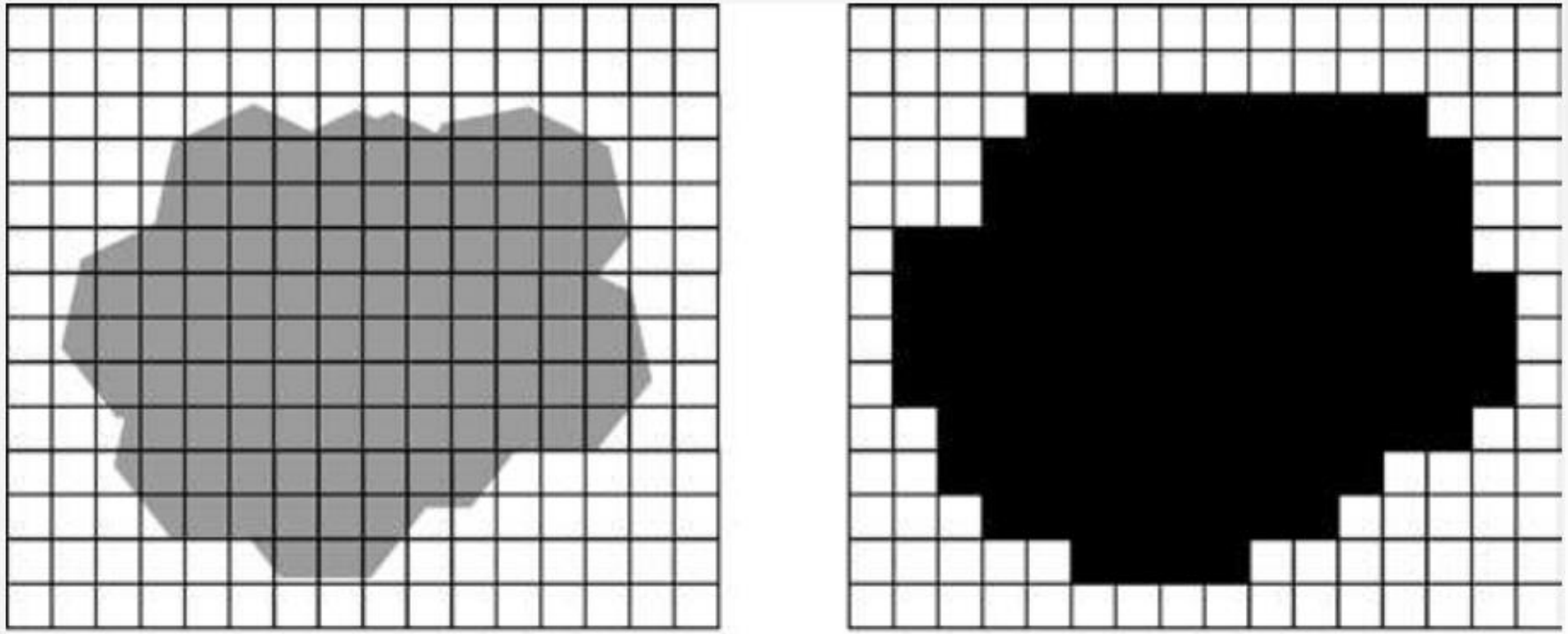
机器人导航技术-基于地图导航

前节中我们已经讨论了如何定位与建图，在例子中地标仅用一点的坐标表示，而实际的情况是地标通常为障碍物，并且在地图中占据了一定区域，如何表示可通行区域与障碍物区域成为新的问题。如果用一个多面体表示障碍物的大小，此种方法看似十分精确，但是会增大地图信息存储的压力，同时，在确定机器人与障碍物是否会碰撞时，计算更加复杂。

一种存储简单并且易于计算的方法是将整个地图区域分割成若干个大小相同的方格，每个方格标记为0或1，用0表示没有被障碍物占用，可以通行；用1表示被占用，无法通行。单元格的大小可以根据实际应用场景自行设定。当然，随着整个地图区域的扩大或者单元格面积的缩小，计算机需要的存储空间会增加。

机器人导航技术-基于地图导航

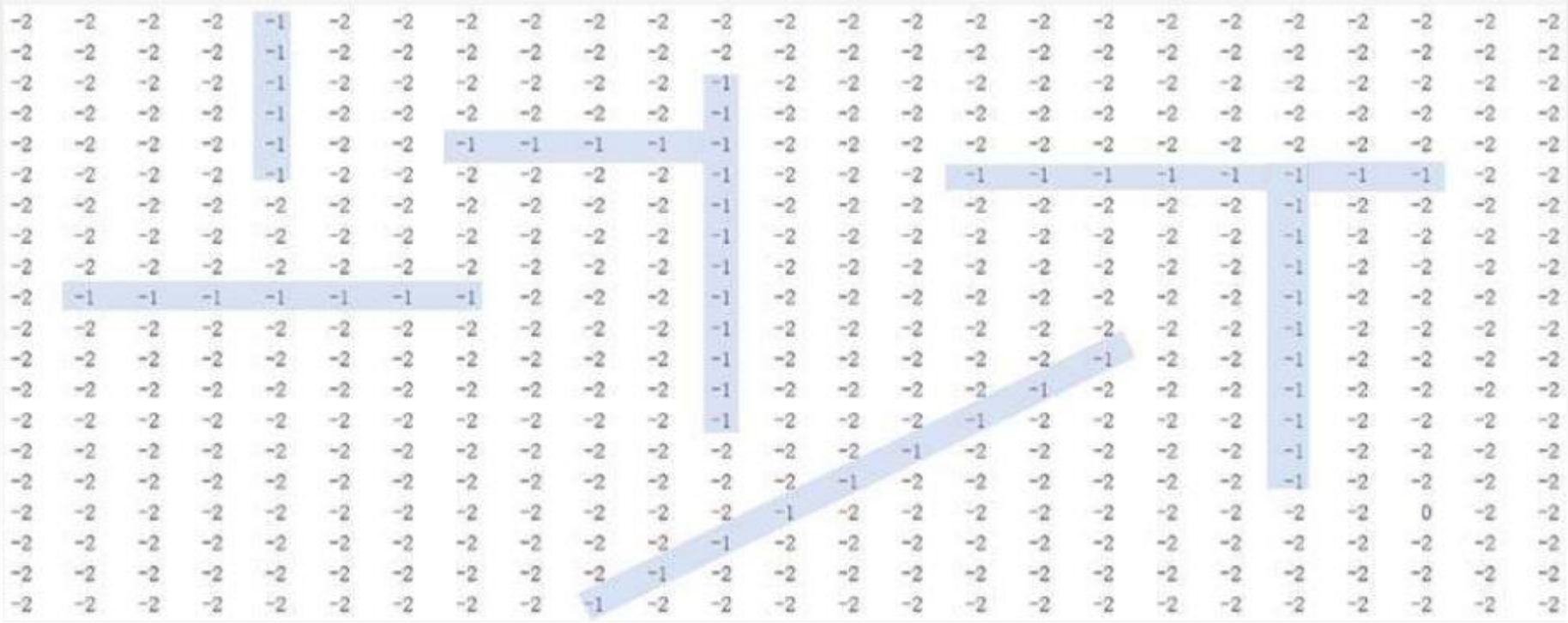
下图为一个多面体的障碍物在网格地图中的表示，占用网格填充为黑色，可以看到，即使有些网格面积没有被全部占满，我们依旧将其标记为1，如果需要精确表示障碍物的边缘，就需要减小网格面积，这时相应的存储空间也会增加，这里需要设计者自行寻找一个平衡点。



（a）障碍物在网格地图中的表示　　（b）将占用网格填充为黑色

机器人导航技术-基于地图导航

介绍一种基础的路径规划方法——梯度法。首先，假设地图的大小为20×25的矩阵，即横轴方向有25个网格，纵轴方向有20个网格，这里将矩阵初始化为：所有元素的初始值都为-2，然后将障碍物位置设为-1，终点位置设为0。



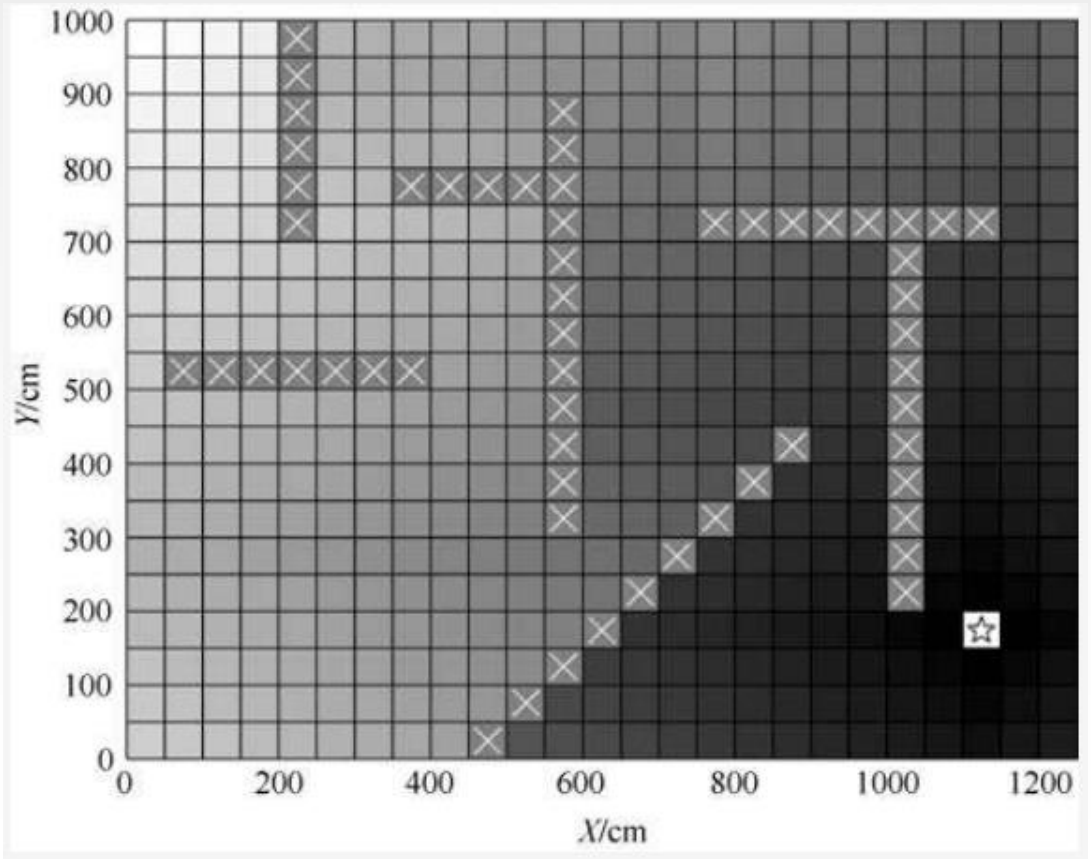
机器人导航技术-基于地图导航

以终点为中心，将外层位置上的元素值以1为单位递增，得到表示梯度关系的矩阵。

46	45	44	43	-1	33	32	31	30	29	28	27	26	25	24	25	24	23	22	21	20	19	18	17	18
45	44	43	42	-1	32	31	30	29	28	27	26	25	24	23	24	23	22	21	20	19	18	17	16	17
44	43	42	41	-1	33	32	31	30	29	28	-1	24	23	22	23	22	21	20	19	18	17	16	15	16
43	42	41	40	-1	34	33	32	31	30	29	-1	23	22	21	22	21	20	19	18	17	16	15	14	15
42	41	40	39	-1	35	34	-1	-1	-1	-1	-1	22	21	20	21	20	19	18	17	16	15	14	13	14
41	40	39	38	-1	36	35	34	33	32	31	-1	21	20	19	-1	-1	-1	-1	-1	-1	-1	-1	12	13
40	39	38	37	36	35	34	33	32	31	30	-1	20	19	18	17	16	15	14	13	-1	11	10	11	12
39	38	37	36	35	34	33	32	31	30	29	-1	19	18	17	16	15	14	13	12	-1	10	9	10	11
38	37	36	35	34	33	32	31	30	29	28	-1	18	17	16	15	14	13	12	11	-1	9	8	9	10
37	-1	-1	-1	-1	-1	-1	-1	29	28	27	-1	17	16	15	14	13	12	11	10	-1	8	7	8	9
36	35	34	33	32	31	30	29	28	27	26	-1	16	15	14	13	12	11	10	9	-1	7	6	7	8
35	34	33	32	31	30	29	28	27	26	25	-1	17	16	15	14	13	-1	9	8	-1	6	5	6	7
34	33	32	31	30	29	28	27	26	25	24	-1	18	17	16	15	-1	9	8	7	-1	5	4	5	6
33	32	31	30	29	28	27	26	25	24	23	-1	19	18	17	-1	9	8	7	6	-1	4	3	4	5
32	31	30	29	28	27	26	25	24	23	22	21	20	19	-1	9	8	7	6	5	-1	3	2	3	4
33	32	31	30	29	28	27	26	25	24	23	22	21	-1	9	8	7	6	5	4	-1	2	1	2	3
34	33	32	31	30	29	28	27	26	25	24	23	-1	9	8	7	6	5	4	3	2	1	0	1	2
35	34	33	32	31	30	29	28	27	26	25	-1	11	10	9	8	7	6	5	4	3	2	1	2	3
36	35	34	33	32	31	30	29	28	27	-1	13	12	11	10	9	8	7	6	5	4	3	2	3	4
37	36	35	34	33	32	31	30	29	-1	15	14	13	12	11	10	9	8	7	6	5	4	3	4	5

机器人导航技术-基于地图导航

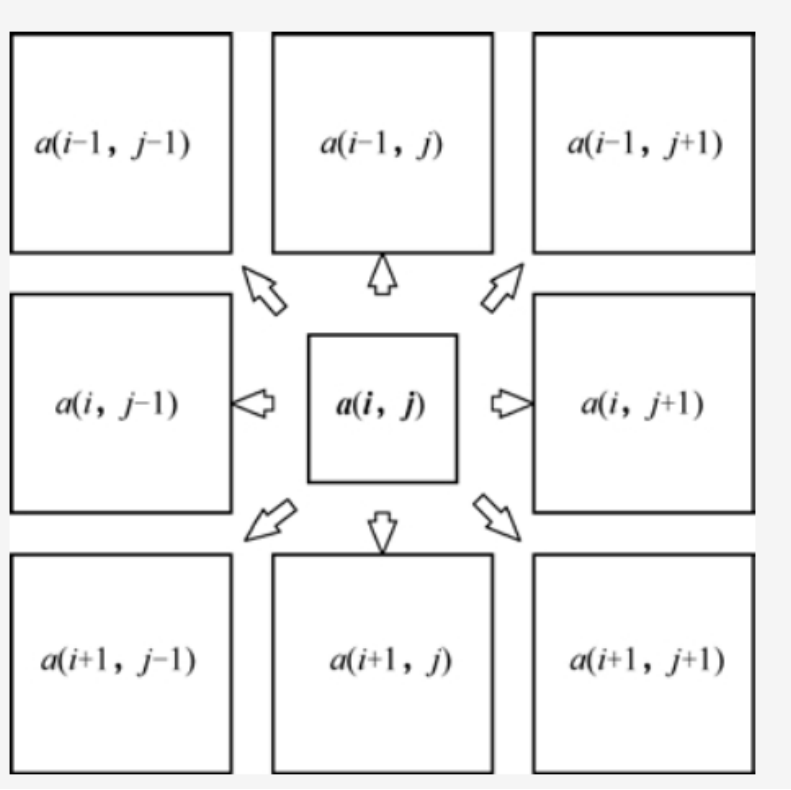
梯度化过程只在可以通行区域进行，读者可以自行思考完成该部分程序，结果如图3.15所示。下面画出与地图矩阵对应的网格图，由于矩阵中元素的最大值为a(1,1)=46,最小值为终点a(17,23)=0,将a(1,1)对应网格设为白色，a(17,23)对应网格设为黑色，其余每个网格填充成与其矩阵元素值大小对应的灰度值，同时障碍物用“×”表示，终点用“☆”表示，每个网格边长设为50cm,得到下图所示的结果。



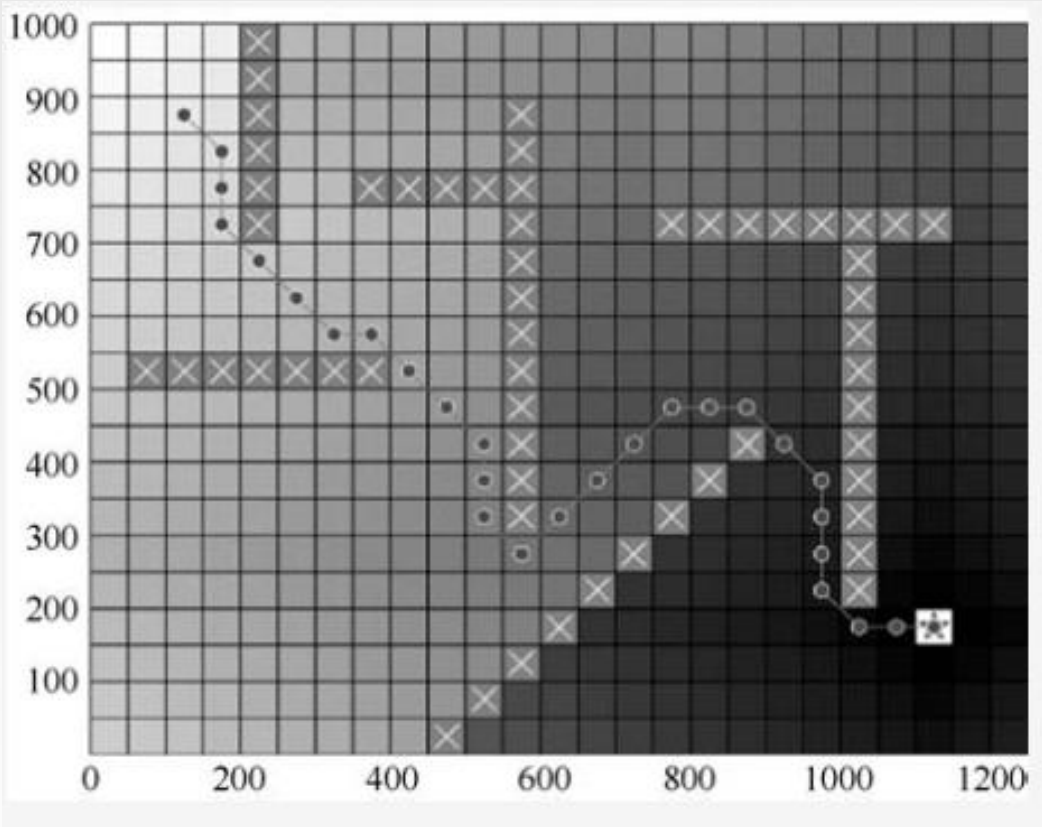
机器人导航技术-基于地图导航

我们设定图中矩阵的a(3.3)位置为起点，因为可行区域的梯度关系已经得到，此时要做的就是寻找一条道路，满足在前进过程中行经方格所对应的值始终是递减的。这种简单的路径规划策略很容易实现，只需依次搜索当前位置点a(i j)周围的8个方格，如下图所示。如果搜索到的方格为障碍物，则将其忽略，直到找到对应矩阵元素值最小的方格，然后将其作为新的位置点。

46	45	44	43	-1	33	32	31	30	29	28	27	26	25	24	25	24	23	22	21	20	19	18	17	18
45	44	43	42	-1	32	31	30	29	28	27	26	25	24	23	24	23	22	21	20	19	18	17	16	17
44	43	42	41	-1	33	32	31	30	29	28	-1	24	23	22	23	22	21	20	19	18	17	16	15	16
43	42	41	40	-1	34	33	32	31	30	29	-1	23	22	21	22	21	20	19	18	17	16	15	14	15
42	41	40	39	-1	35	34	-1	-1	-1	-1	-1	22	21	20	21	20	19	18	17	16	15	14	13	14
41	40	39	38	-1	36	35	34	33	32	31	-1	21	20	19	-1	-1	-1	-1	-1	-1	-1	-1	12	13
40	39	38	37	36	35	34	33	32	31	30	-1	20	19	18	17	16	15	14	13	-1	11	10	11	12
39	38	37	36	35	34	33	32	31	30	29	-1	19	18	17	16	15	14	13	12	-1	10	9	10	11
38	37	36	35	34	33	32	31	30	29	28	-1	18	17	16	15	14	13	12	11	-1	9	8	9	10
37	-1	-1	-1	-1	-1	-1	-1	29	28	27	-1	17	16	15	14	13	12	11	10	-1	8	7	8	9
36	35	34	33	32	31	30	29	28	27	26	-1	16	15	14	13	12	11	10	9	-1	7	6	7	8
35	34	33	32	31	30	29	28	27	26	25	-1	17	16	15	14	13	-1	9	8	-1	6	5	6	7
34	33	32	31	30	29	28	27	26	25	24	-1	18	17	16	15	-1	9	8	7	-1	5	4	5	6
33	32	31	30	29	28	27	26	25	24	23	-1	19	18	17	-1	9	8	7	6	-1	4	3	4	5
32	31	30	29	28	27	26	25	24	23	22	21	20	19	-1	9	8	7	6	5	-1	3	2	3	4
33	32	31	30	29	28	27	26	25	24	23	22	21	-1	9	8	7	6	5	4	-1	2	1	2	3
34	33	32	31	30	29	28	27	26	25	24	23	-1	9	8	7	6	5	4	3	2	1	0	1	2
35	34	33	32	31	30	29	28	27	26	25	-1	11	10	9	8	7	6	5	4	3	2	1	2	3
36	35	34	33	32	31	30	29	28	27	-1	13	12	11	10	9	8	7	6	5	4	3	2	3	4
37	36	35	34	33	32	31	30	29	-1	15	14	13	12	11	10	9	8	7	6	5	4	3	4	5



重复以上工作，得到的结果如图所示。可以看到，“●”表示的网格为规划后的路



上述导航算法充分利用了可行区域与占用区域的全局信息，除此之外，还有很多适用于机器人路径规划的算法，例如D*算法，将占用网格推广为成本地图，通过寻找一条路径使得运行的总成本最低，在之后的章节中将会有相关的介绍。

机器人导航技术-基于地图导航

课后目标

(1) 了解两轮驱动小车的运动学模型；

(2) 掌握航迹推算及位姿估计的具体方法；

(3) 掌握定位与建图的基础理论；

(4) 了解反应式导航和基于地图导航的区别。