Robótica Móvil

Localización

Taihú Pire

Laboratorio de Robótica

CONICET

UNIVERSIDAD NACIONAL DE ROSARIO

CONTINGERE HOMINEM COGITANTEM

C I F A S I S

Temario para estas slides

- Bayes filter
- kalman filter
- Particle filter

Localización

Localización

Es la habilidad que posee una máquina (robot) para localizarse en el espacio.

Estimación de estado

- Estimar el **estado** x de un sistema dada las **observaciones** z y **comandos de control** u
- Objetivo:

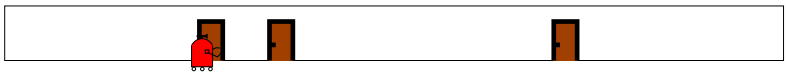
$$p\left(x_t | z_{1:t}, u_{1:t}\right) \tag{1}$$

Estimación de estado recursiva: Se trata de estimar x_t el estado actual utilizando también el estado inmediatamente anterior x_{t-1}.

Filtro de Bayes Recursivo

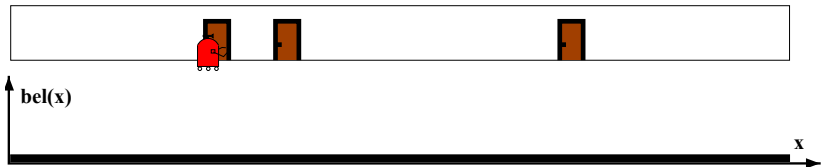
Ejemplo

- Dado un robot en un mundo de una dimensión, sin conocimiento de en donde se encuentra
- El robot se puede mover hacia delante o atrás
- Supongamos además que hay tres puertas (landmarks), el robot puede detectar si se encuentra al lado de una puerta o no.



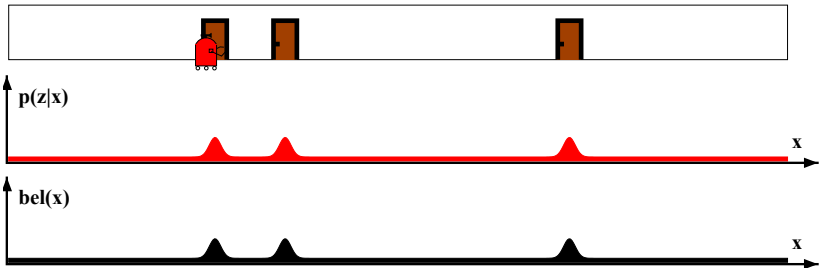
Filtro de Bayes Recursivo: Posición inicial

Como en un principio el robot desconoce cual es su posición entonces es igualmente posible que se encuentre en cualquier punto del mundo (belief). Esto lo podemos representar matemáticamente diciendo que la función de distribución de probabilidad del robot es uniforme sobre el mundo en que se encuentra.



Filtro de Bayes Recursivo: Medición

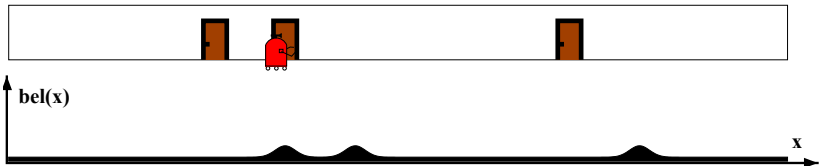
Si el robot sensa que se encuentra al lado de una puerta entonces la creencia de su ubicación se se ve alterada de la siguiente manera:



Esta nueva función representa otra distribución de probabilidades llamada Posterior belief.
La función Posterior belief es la mejor representación de la posición del robot actualmente. Cada "loma" representa la evaluación de su posición con respecto a una puerta.
La probabilidad $p(\mathbf{z}|x)$ se lee: "dado que sabemos dónde se encuentra el robot, ¿Cúal es la probabilidad de que observar la puerta?"

Filtro de Bayes Recursivo: Movimiento

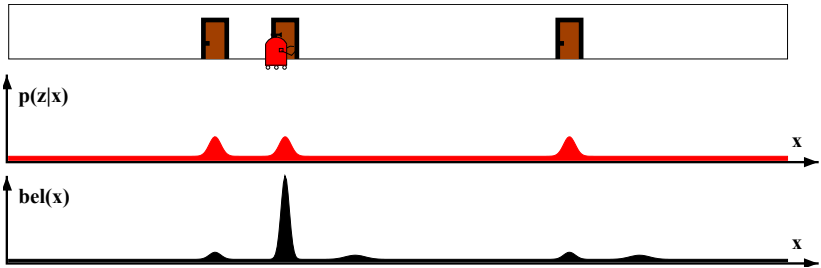
Si **el robot se mueve hacia la derecha** la creencia es cambiada de acuerdo al movimiento. Así mismo como el movimiento del robot es inexacto, al trasladarse su incertidumbre crece, dicho de otra manera, las lomas son aplanadas. Este aplanamiento matemáticamente se lleva a cabo por medio de la operación de convolución entre la función Posterior belief y la función que describe la distancia recorrida.



La operación de convolución mide la superposición mientras se desliza una funcion sobre otra.

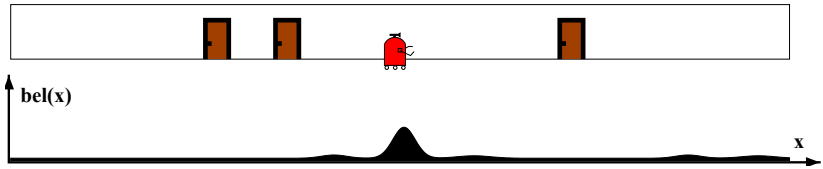
Filtro de Bayes Recursivo: Segunda medición

Supongamos que el robot luego de haberse movido sensa nuevamente que se encuentre al lado de una puerta entonces, como antes, la probabilidad se incrementara por un cierto factor la función de probabilidad donde haya una puerta.



Filtro de Bayes Recursivo: Segundo movimiento

Supongamos que el robot se mueve nuevamente...



Estimación de estado

- Estimar el estado x de un systema dada las observaciones z y comandos de control u
- Objetivo:

$$p\left(x_t | z_{1:t}, u_{1:t}\right) \tag{2}$$

Estimación de estado recursiva: Se trata de estimar x_t el estado actual utilizando también el estado inmediatamente anterior x_{t-1}.

Estimación de estado

Estamos interesados en la creencia (*belief*) del sistema sobre dónde está en el tiempo t:

$$bel(x_t) = p\left(x_t | z_{1:t}, u_{1:t}\right) \quad \text{(ecuación de estimación de estado o definición de distribución de probabilidad)}$$

La ecuación se puede leer como: "¿dónde estoy ahora x_t?, dada todas las observaciones $z_{1:t}$ y todos los comandos de control $u_{1:t}$"

Ahora vamos a simplificar la distribución utilizando la Regla de Bayes:

Derivación del Filtro de Bayes Recursivo

$$bel(x_t) = p\left(x_t | z_{1:t}, u_{1:t}\right)$$

Derivación del Filtro de Bayes Recursivo

$$
\begin{aligned}
bel(x_t) &= p\left(x_t | z_{1:t}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t, z_{1:t-1}, u_{1:t}\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \qquad \text{donde } \eta \text{ es una constante de normalización}
\end{aligned}
$$

Regla de Bayes

Derivación del Filtro de Bayes Recursivo

$$
\begin{aligned}
bel(x_t) &= p\left(x_t | z_{1:t}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t, z_{1:t-1}, u_{1:t}\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \quad \text{donde } \eta \text{ es una constante de normalización} \\
&= \eta\, p\left(z_t | x_t\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right)
\end{aligned}
$$

Markov assumption. Lo que mido ahora z_t solo depende del estado actual x_t y es independiente de las mediciones anteriores $z_{1:t-1}$ y de todos los comandos $u_{1:t}$

Derivación del Filtro de Bayes Recursivo

$$
\begin{aligned}
bel(x_t) &= p\left(x_t | z_{1:t}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t, z_{1:t-1}, u_{1:t}\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \quad \text{donde } \eta \text{ es una constante de normalización} \\
&= \eta\, p\left(z_t | x_t\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, z_{1:t-1}, u_{1:t}\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t}\right) dx_{t-1}
\end{aligned}
$$

Ley de probabilidad total. Es decir, integramos sobre todos los estados anteriores posibles. Se puede intepretar como "¿Dónde estámos ahora, dado el estado anterior x_{t-1} multiplicado porque tan probable es que se alcance dicho estado x_{t-1}."

Derivación del Filtro de Bayes Recursivo

$$
\begin{aligned}
bel(x_t) &= p\left(x_t | z_{1:t}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t, z_{1:t-1}, u_{1:t}\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \quad \text{donde } \eta \text{ es una constante de normalización} \\
&= \eta\, p\left(z_t | x_t\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, z_{1:t-1}, u_{1:t}\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t}\right) dx_{t-1} \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, u_t\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t}\right) dx_{t-1}
\end{aligned}
$$

Markov assumption. Si x_{t-1} es dado, entonces son irrelevantes todas las mediciones $z_{1:t-1}$ y comandos u_{t-1}. Nos Importa u_t porque nos dice cómo evoluciona de x_{t-1} a x_t

Derivación del Filtro de Bayes Recursivo

$$
\begin{aligned}
bel(x_t) &= p\left(x_t | z_{1:t}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t, z_{1:t-1}, u_{1:t}\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \quad \text{donde } \eta \text{ es una constante de normalización} \\
&= \eta\, p\left(z_t | x_t\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, z_{1:t-1}, u_{1:t}\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t}\right) dx_{t-1} \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, u_t\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t}\right) dx_{t-1} \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, u_t\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t-1}\right) dx_{t-1}
\end{aligned}
$$

Independence assumption. Asumimos que un comando futuro u_t no nos ayuda a saber nuestro estado actual x_t.

Derivación del Filtro de Bayes Recursivo

$$
\begin{aligned}
bel(x_t) &= p\left(x_t | z_{1:t}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t, z_{1:t-1}, u_{1:t}\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \quad \text{donde } \eta \text{ es una constante de normalización} \\
&= \eta\, p\left(z_t | x_t\right) p\left(x_t | z_{1:t-1}, u_{1:t}\right) \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, z_{1:t-1}, u_{1:t}\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t}\right) dx_{t-1} \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, u_t\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t}\right) dx_{t-1} \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, u_t\right) p\left(x_{t-1} | z_{1:t-1}, u_{1:t-1}\right) dx_{t-1} \\
&= \eta\, p\left(z_t | x_t\right) \int p\left(x_t | x_{t-1}, u_t\right) bel(x_{t-1}) dx_{t-1}
\end{aligned}
$$

Término recursivo!

Paso de Predicción y Paso de Corrección

- El Filtro de Bayes puede ser escrito como un proceso de dos pasos

$$bel(x_t) = \eta\, p\,(z_t|x_t) \int p\,(x_t|x_{t-1}, u_t)\, bel(x_{t-1}) dx_{t-1}$$

- Paso de Predicción

$$\overline{bel}(x_t) = \int p\,(x_t|x_{t-1}, u_t)\, bel(x_{t-1}) dx_{t-1}$$

- Paso de Corrección

$$bel(x_t) = \eta\, p\,(z_t|x_t)\, \overline{bel}(x_t)$$

Motion model y observation model

- Paso de Predicción

$$\overline{bel}(x_t) = \int \underbrace{p\left(x_t | x_{t-1}, u_t\right)}_{\text{Motion model}} bel(x_{t-1}) dx_{t-1}$$

- Paso de Corrección

$$bel(x_t) = \eta \underbrace{p\left(z_t | x_t\right)}_{\text{Observation model (Measurement model o Sensor model)}} \overline{bel}(x_t)$$

Diferentes implementaciones

- El Filtro de Bayes es un **framework** estimación de estado recursiva
- Hay diferentes implementaciones
- Diferentes Propiedades:
  - Modelos lineales vs no-lineales para el modelo de movimiento y observación
  - Utilización de distribuciones de probabilidad solamente Gausianas?
  - Filtros paramétricos y no-paramétricos (representar la distribución de probabilidad de manera paramétrica o no-paramétrica)
  - ...

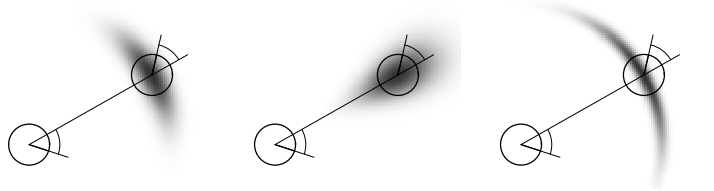
Filtros populares

- Kalman Filter
  - Utiliza distribuciones de probabilidad Gaussianas
  - Solo funciona con modelo de movimiento y observación lineales
- Extended Kalman Filter (EKF)
  - Utiliza distribuciones de probabilidad Gaussianas
  - **Linealiza** los modelos de movimiento y observación no lineales por medio de aproximación de Taylor
- Particle filter
  - No-paramétrico
  - Modelos de distribuciones probabilísticas arbitrarias, es decir no todo es Gaussiano (etapa de sampleo de muestras)

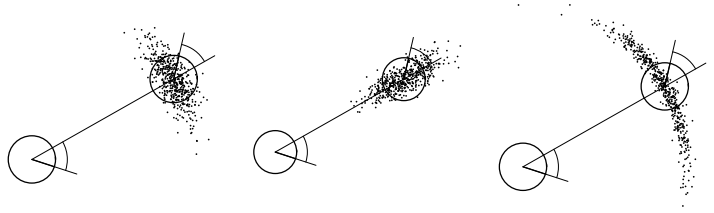
Motion Model

$$\overline{bel}(x_t) = \int \underbrace{p\left(x_t | x_{t-1}, u_t\right)}_{\text{Motion model}} bel(x_{t-1}) dx_{t-1}$$

Ejemplo: Movimiento basado en odometría



Covarianza del motion model



Muestras del motion model

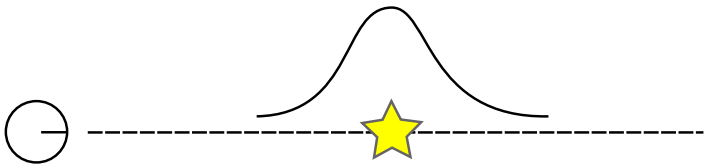
Observation model

$$bel(x_t) = \eta \underbrace{p\,(z_t|x_t)}\, \overline{bel}(x_t)$$

Observation model (Measurement model o Sensor model)

Ejemplo: Observation Model simple con Ruido Gaussiano

- Sensor de rango estimando la distancia a un objeto cercano
- Ruido Gaussiano en la lectura del sensor

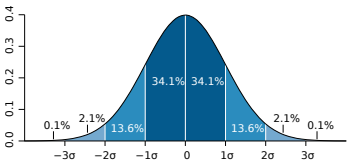


Kalman Filter

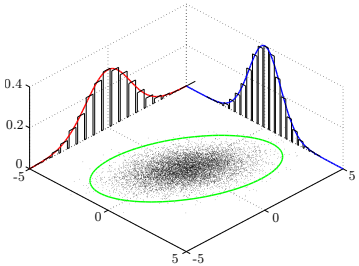
- Es un Filtro de Bayes
- Todo es Gaussiano

$$p(x) = \det(2\pi\Sigma)^{\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

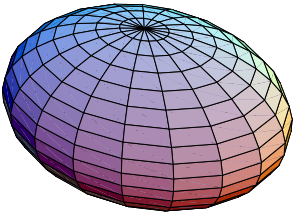
- Soluciones optimas para modelos lineales y distribuciones Gaussianas.



Distribución Gaussiana en 1D



Distribución Gaussiana en 2D



Distribución Gaussiana en 3D

Suposiciones de Kalman Filter

- Ruido y distribuciones Gaussianas
- Modelos de movimiento y observación lineales

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$
$$z_t = C_t x_t + \delta_t$$

A_t Matriz $(n \times n)$ que describe como el estado evoluciona de $t-1$ a t sin comandos de control o ruido.

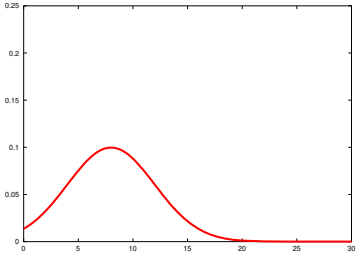
B_t Matriz $(n \times l)$ que describe cómo el control u_t cambia el estado de $t-1$ a t.

C_t Matriz $(k \times n)$ que describe cómo mapear el estado x_t a una observación z_t.

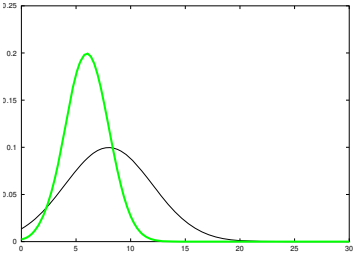
ϵ_t Variable aleatoria representando el ruido gaussiano del **modelo de movimiento**, se asume independiente y con distribución normal con media cero y covarianza R_t.

δ_t Variable aleatoria representando el ruido gaussiano del **modelo de observación**, se asume independiente y con distribución normal con media cero y covarianza Q_t.

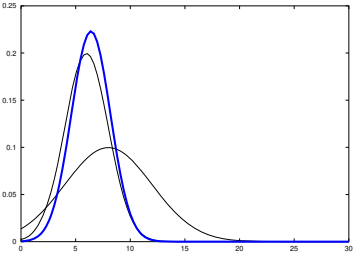
1D Kalman Filter



Predicción Inicial

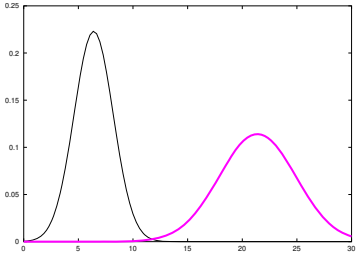


Medición

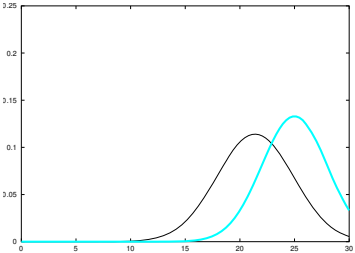


Corrección

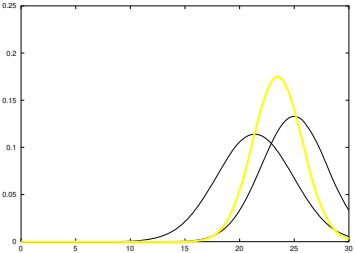
1D Kalman Filter



Predicción por Movimiento



Segunda Medición



Segunda Corrección

Suposiciones de Kalman Filter

- Ruido y distribuciones Gaussianas
- Modelos de movimiento y observación lineales

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$
$$z_t = C_t x_t + \delta_t$$

¿Qué pasa si este estas suposiciones no pasan?

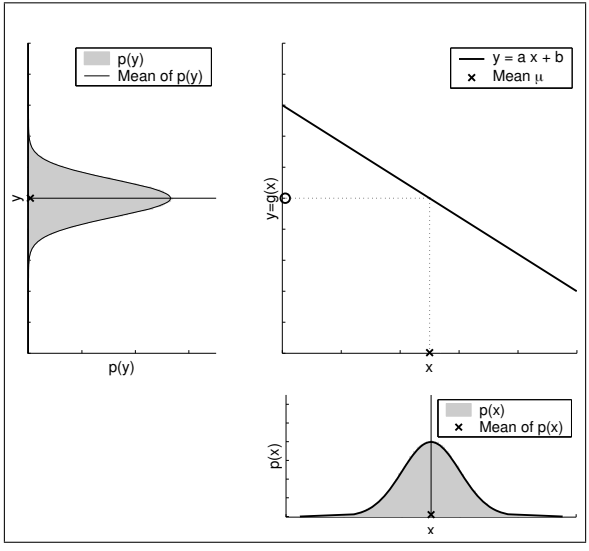
Sistemas dinámicos no Non-lineales

- Los problemas reales en general emplean funciones no-lineales

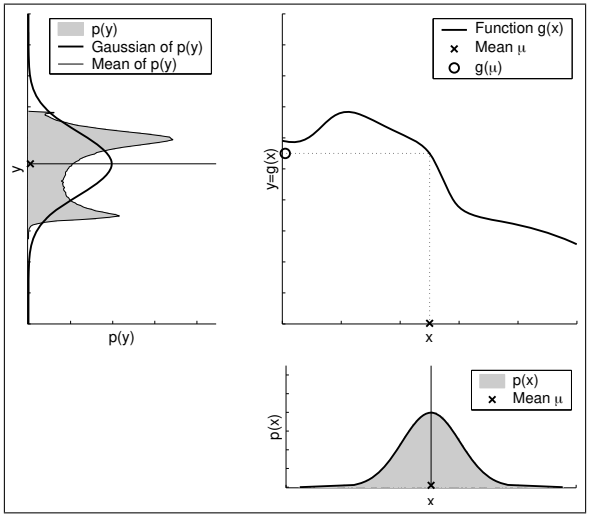
$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$
$$z_t = C_t x_t + \epsilon_t$$

$$x_t = g\left(u_t, x_{t-1}\right) + \epsilon_t$$
$$z_t = h\left(x_t\right) + \delta_t$$

Suposición de Linealidad



Función No-Lineal



Distribuciones No-Gaussianas

- Las funciones no-lineales llevan a distribuciones no Gaussianas
- Entonces, no podemos aplicar Kalman Filter!

Linearización en EKF: Expansión de Taylor de primer orden

- Predicción:

$$g\left(u_t, x_{t-1}\right) \approx g\left(u_t, \mu_{t-1}\right) + \underbrace{\frac{\partial g\left(u_t, \mu_{t-1}\right)}{\partial x_{t-1}}}_{G_t}\left(x_{t-1} - \mu_{t-1}\right)$$

- Corrección:

$$h\left(x_t\right) \approx h\left(\overline{\mu}_t\right) + \underbrace{\frac{\partial h\left(\overline{\mu}_t\right)}{\partial x_t}}_{H_t}\left(x_t - \overline{\mu}_t\right)$$

G_t y H_t se denominan Jacobianos.

Recordando Jacobianos

- En general es una matriz no-cuadrada $m \times n$
- Dada una función vectorial

$$g(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{bmatrix}$$

- El Jacobiano está definido como

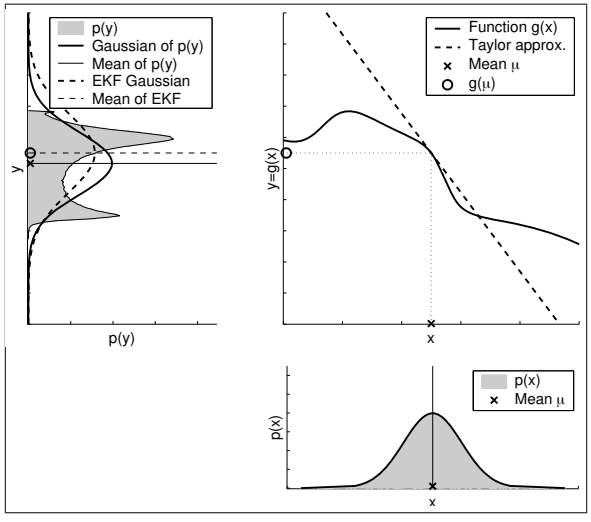
$$G_x = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{bmatrix}$$

Recordando Jacobianos

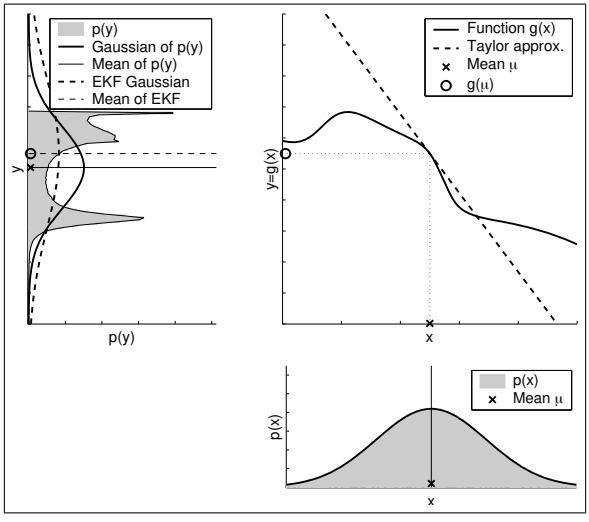
agregar imagenes de jacobianos 3D, plano tangencial

- El Jacobiano es la orientación del plano tangente a la función vectorial en un punto dado
- El Jacobiano es la generalización del gradiente de una función escalar

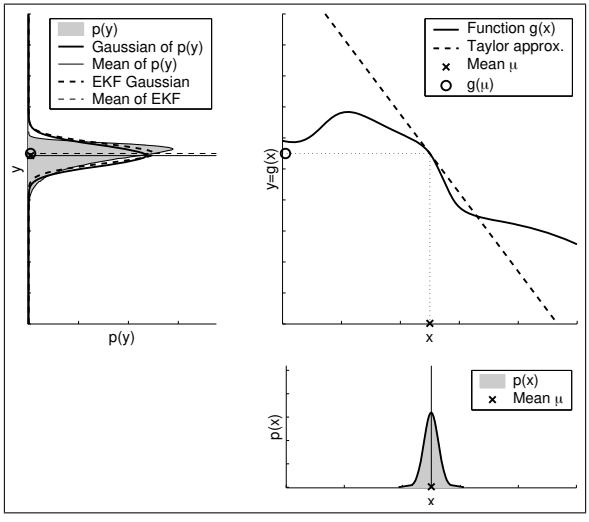
Linearización en EKF



Calidad de la Linearización en EKF



Calidad de la Linearización en EKF



Modelo de movimiento linearizado

- Linearizar el modelo esta dado por:

$$p(x_t|u_t, x_{t-1}) \approx \det(2\pi R_t)^{\frac{1}{2}} \exp(-\frac{1}{2}(x_t - g\,(u_t - \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1}))^\top R_t{}^{-1}(x_t - \underbrace{g\,(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})}_{\text{modelo linearizado}}))$$

- R_t describe el ruido del movimiento

Modelo de observación linearizado

- Linearizar el modelo esta dado por:

$$p(z_t|x_t) \approx \det(2\pi Q_t)^{\frac{1}{2}} \exp(-\frac{1}{2}(z_t - h(\overline{\mu_t}) - H_t(x_t - \overline{\mu_t}))^\top {Q_t}^{-1}(z_t - \underbrace{h(\overline{\mu_t}) - H_t(x_t - \overline{\mu_t})}_{\text{modelo linearizado}}))$$

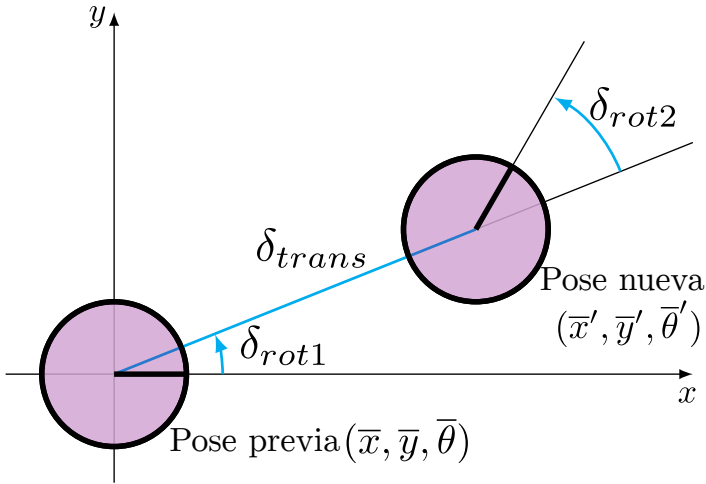
- Q_t describe el ruido de la medición

Algoritmo de Filtro de Kalman Extendido

1: **procedure** ExtendedKalmanFilter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
2: $\quad \overline{\mu}_t = g\left(u_t, \mu_{t-1}\right)$
3: $\quad \overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^\top + R_t$

4: $\quad K_t = \overline{\Sigma}_t H_t^\top \left(H_t \overline{\Sigma}_t H_t + Q_t\right)^{-1}$
5: $\quad \mu_t = \overline{\mu}_t + K_t(z_t - h\left(\overline{\mu}_t\right))$
6: $\quad \Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
7: $\quad$ **return** μ_t, Σ_t
8: **end procedure**

Odometry as controls



$$u = (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$$

Motion model

- Odometría de modelo de movimiento

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \underbrace{\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \delta_{trans} \cos(\theta + \delta_{rot_1}) \\ \delta_{trans} \sin(\theta + \delta_{rot_1}) \\ \delta_{rot_1} + \delta_{rot_2} \end{bmatrix}}_{g(u_t, x_{t-1})} + \mathcal{N}(0, R_t)$$

- Linearización

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$$

Jacobianos del modelo de movimiento

Calculamos el Jacobiano con respecto al estado

$$G_t = \frac{\delta g\left(u_t, \mu_{t-1}\right)}{\delta x_{t-1}} = \begin{bmatrix} \frac{\delta x'}{\delta \mu_{t-1,x}} & \frac{\delta x'}{\delta \mu_{t-1,y}} & \frac{\delta x'}{\delta \mu_{t-1,\theta}} \\ \frac{\delta y'}{\delta \mu_{t-1,x}} & \frac{\delta y'}{\delta \mu_{t-1,y}} & \frac{\delta y'}{\delta \mu_{t-1,\theta}} \\ \frac{\delta \theta'}{\delta \mu_{t-1,x}} & \frac{\delta \theta'}{\delta \mu_{t-1,y}} & \frac{\delta \theta'}{\delta \mu_{t-1,\theta}} \end{bmatrix}$$

$$G_t = \begin{bmatrix} 1 & 0 & -\delta_{trans}\sin(\theta + \delta_{rot_1}) \\ 0 & 1 & \delta_{trans}\cos(\theta + \delta_{rot_1}) \\ 0 & 0 & 1 \end{bmatrix}$$

Jacobianos del modelo de movimiento

Calculamos el Jacobiano con respecto al comando de control

$$Vt = \frac{\delta g\left(u_t, \mu_{t-1}\right)}{\delta u_t} = \begin{bmatrix} \frac{\delta x'}{\delta u_{t,\delta_{rot_1}}} & \frac{\delta x'}{\delta u_{t,\delta_{trans}}} & \frac{\delta x'}{\delta u_{t,\delta_{rot_2}}} \\ \frac{\delta y'}{\delta u_{t,\delta_{rot_1}}} & \frac{\delta y'}{\delta u_{t,\delta_{trans}}} & \frac{\delta y'}{\delta u_{t,\delta_{rot_2}}} \\ \frac{\delta \theta'}{\delta u_{t,\delta_{rot_1}}} & \frac{\delta \theta'}{\delta u_{t,\delta_{trans}}} & \frac{\delta \theta'}{\delta u_{t,\delta_{rot_2}}} \end{bmatrix}$$

$$V_t = \begin{bmatrix} -\delta_{trans}\sin(\theta + \delta_{rot_1}) & \cos(\theta + \delta_{rot_1}) & 0 \\ \delta_{trans}\cos(\theta + \delta_{rot_1}) & \sin(\theta + \delta_{rot_1}) & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Modelo de observación

- Modelo de distancia y orientación (range-bearing model)

$$z_t^i = \begin{bmatrix} r_t^i \\ \phi_t^i \end{bmatrix} = \begin{bmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \end{bmatrix} + \mathcal{N}(0, Q_t)$$

- Linearización

$$h(x_t, m) \approx h(\overline{\mu}_t) + H_t^i (x_t - \overline{\mu}_t)$$

Jacobianos del modelo de observación

Calculamos el Jacobiano con respecto al estado

$$H_t = \frac{\delta h\left(\overline{\mu}_t, m\right)}{\delta x_t} = \begin{bmatrix} \frac{\delta r_t^i}{\delta \overline{\mu}_{t,x}} & \frac{\delta r_t^i}{\delta \overline{\mu}_{t,y}} & \frac{\delta r_t^i}{\delta \overline{\mu}_{t,\theta}} \\ \frac{\delta \phi_t^i}{\delta \overline{\mu}_{t,x}} & \frac{\delta \phi_t^i}{\delta \overline{\mu}_{t,y}} & \frac{\delta \phi_t^i}{\delta \overline{\mu}_{t,\theta}} \end{bmatrix}$$

$$H_t^i = \begin{bmatrix} -\frac{m_{j,x} - \overline{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{j,y} - \overline{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{j,y} - \overline{\mu}_{t,y}}{q} & -\frac{m_{j,x} - \overline{\mu}_{t,x}}{q} & -1 \end{bmatrix}$$

donde

$$q = (m_{j,x} - \overline{\mu}_{t,x})^2 + (m_{j,y} - \overline{\mu}_{t,y})^2$$

Algoritmo de Filtro de Kalman Extendido

1: ExtendedKalmanFilter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t,\ c_t,\ m$)

2: $\theta = \mu_{t-1,\theta}$

3: $G_t = \begin{bmatrix} 1 & 0 & -\delta_{trans}\sin(\theta + \delta_{rot_1}) \\ 0 & 1 & \delta_{trans}\cos(\theta + \delta_{rot_1}) \\ 0 & 0 & 1 \end{bmatrix}$

4: $V_t = \begin{bmatrix} -\delta_{trans}\sin(\theta + \delta_{rot_1}) & \cos(\theta + \delta_{rot_1}) & 0 \\ \delta_{trans}\cos(\theta + \delta_{rot_1}) & \sin(\theta + \delta_{rot_1}) & 0 \\ 1 & 0 & 1 \end{bmatrix}$

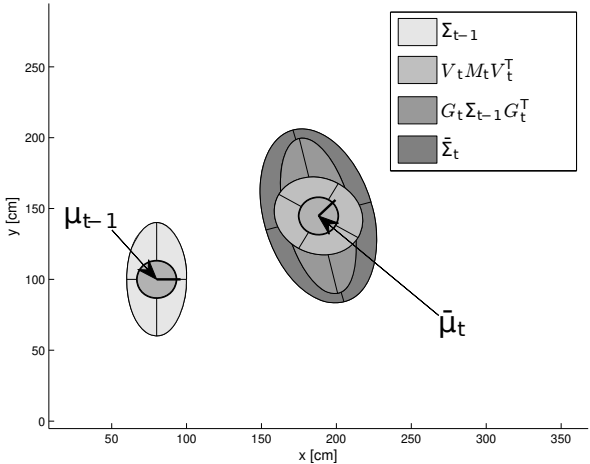
5: $M_t = \begin{bmatrix} \alpha_1\delta^2_{rot_1} + \alpha_2\delta^2_{trans} & 0 & 0 \\ 0 & \alpha_3\delta^2_{trans} + \alpha_4\left(\delta^2_{rot_1} + \delta^2_{rot_2}\right) & 0 \\ 0 & 0 & \alpha_1\delta^2_{rot_2} + \alpha_2\delta^2_{trans} \end{bmatrix}$

6: $\overline{\mu}_t = \mu_{t-1} + \begin{bmatrix} \delta_{trans}\cos(\theta + \delta_{rot_1}) \\ \delta_{trans}\sin(\theta + \delta_{rot_1}) \\ \delta_{rot_1} + \delta_{rot_2} \end{bmatrix}$

7: $\overline{\Sigma}_t = G_t\Sigma_{t-1}G_t^\top + V_tM_tV_t^\top$

M_t es una covarianza que modela el ruido que hay en los comandos de control. $V_tM_tV_t^\top$ hace referencia a la incertidumbre que es agregada debido al ruido de los comandos de control.

Prediction step



Predicción de Pose

Algoritmo de Filtro de Kalman Extendido

1: $Q_t = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}$

2: **for** t **do**odos los features observados $z_t^i = (r_t^i, \phi_t^i)^\top$

3: $\quad j = c_t^i$

4: $\quad q = (m_{j,x} - \overline{\mu}_{t,x})^2 + (m_{j,y} - \overline{\mu}_{t,y})^2$

5: $\quad \hat{z}_t^i = \begin{bmatrix} \sqrt{q} \\ \text{atan2}(m_{j,y} - \overline{\mu}_{t,y}, m_{j,x} - \overline{\mu}_{t,x}) - \overline{\mu}_{t,\theta} \end{bmatrix}$

6: $\quad H_t^i = \begin{bmatrix} -\frac{m_{j,x} - \overline{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{j,y} - \overline{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{j,y} - \overline{\mu}_{t,y}}{q} & -\frac{m_{j,x} - \overline{\mu}_{t,x}}{q} & -1 \end{bmatrix}$

7: $\quad S_t^i = H_t^i \overline{\Sigma}_t H_t^{i\top} + Q_t$

8: $\quad K_t^i = \overline{\Sigma}_t H_t^{i\top} S_t^{i-1}$

9: $\quad \overline{\mu}_t = \overline{\mu}_t + K_t^i (z_t - \hat{z}_t^i)$

10: $\quad \overline{\Sigma}_t = (I - K_t^i H_t^i) \overline{\Sigma}_t$

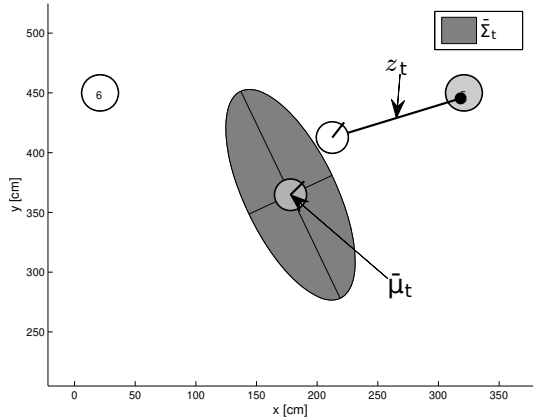
11: **end for**

12: $\mu_t = \overline{\mu}_t$

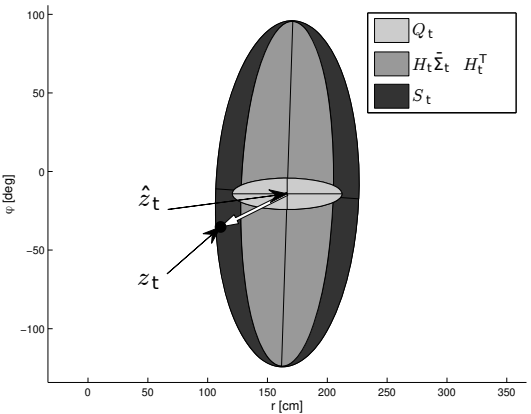
13: $\Sigma_t = \overline{\Sigma}_t$

14: **return** μ_t, Σ_t

Predicción de Medición



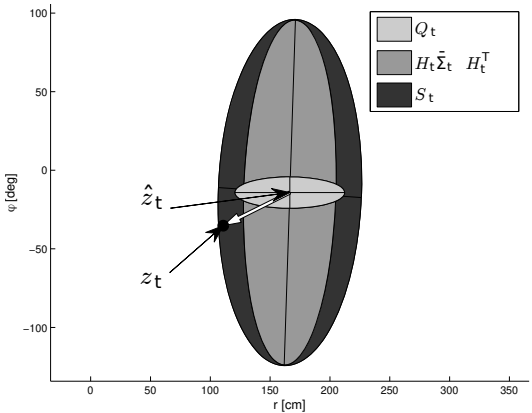
Predicción de Pose



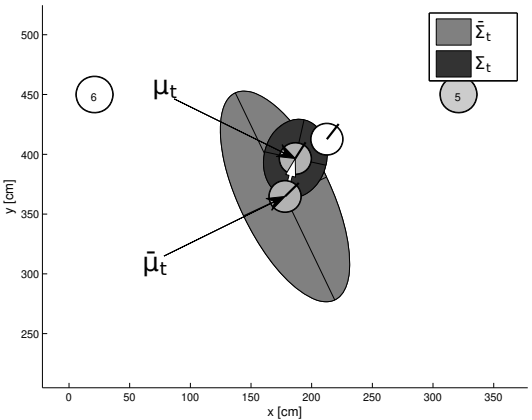
Predicción de Medición

- El ground-truth del robot y la medición están indicados por el círculo blanco y la línea en negrita, respectivamente.
- La flecha blanca indica la **innovación**, diferencia entre la observación y predicción de la medición.

Correction step

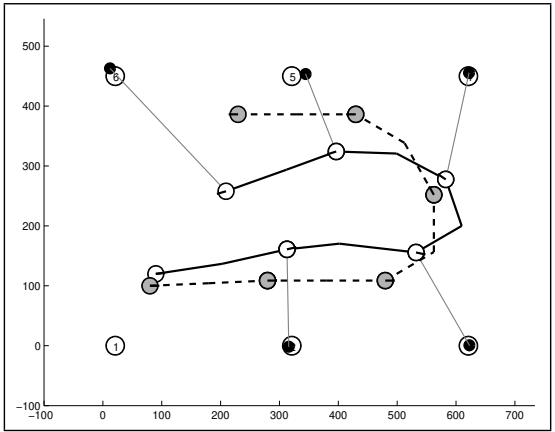


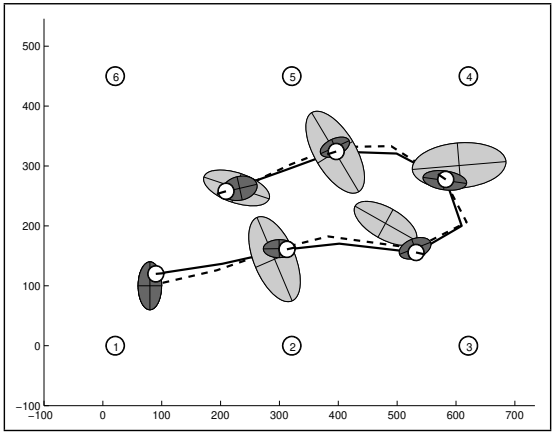
Predicción de Medición



Corrección de Medición

Ejemplo: EKF Localization 2D





La Línea sólida es el Ground-truth. En el primer gráfico la línea punteada es la odometría y en el segundo gráfico es la estimación dada por el EKF.

Resumen de EKF

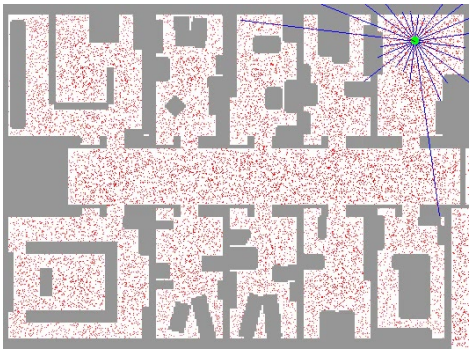
- Es una extensión del Filtro de Kalman
- Una forma de trabajar con no-lineridades
- Realiza linearizaciones locales
- Funciona bien en la práctica para casos moderadamente no lineales
- Grandes incertidumbres conllevan un incremento del error

Extended Kalman Filter

https://automaticaddison.com/extended-kalman-filter-ekf-with-python-code-example/
https://github.com/shangzhouye/EKF-SLAM-on-Turtlebot3
https://github.com/ser94mor/sensor-fusion
https://github.com/AtsushiSakai/PythonRobotics
Puede ser que el tp sea en python no mas y el tp final hacerlo en ROS2.
https://github.com/debbynirwan/mcl

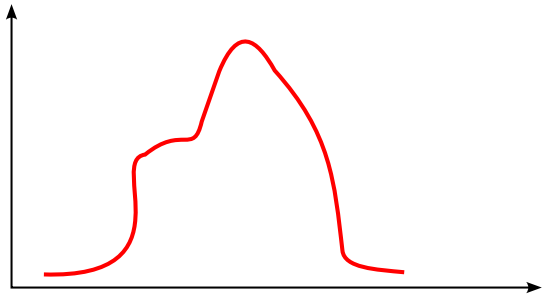
Filtro de Partículas (Particle Filter)

- Con EKF estamos restringidos a distribuciones Gaussianas.
- Cuando usamos EKF obtenemos una Distibuión Gaussiana que describe dónde se encuentra el robot.
- En Particle Filter utilizamos partículas o hipótesis que describen dónde podría estar el robot.
- En vez de tener una forma paramétrica como es EKF, que describimos la distribución de probabilidad con los parámetros media μ y covarianza Σ. Partible Filter utiliza muestras no-paramétricas como hipótesis sobre dónde el robot podría estar.



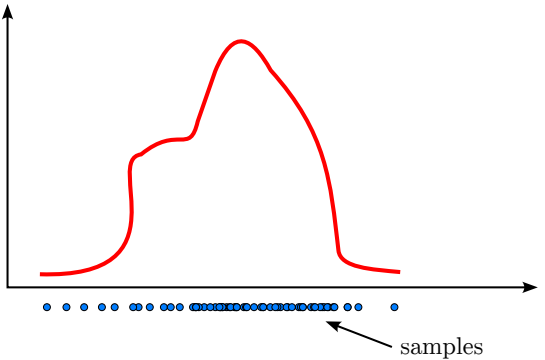
Aproximación de una Función

- Objetivo: Poder estimar cualquier **distribución de probabilidad arbitraria**.



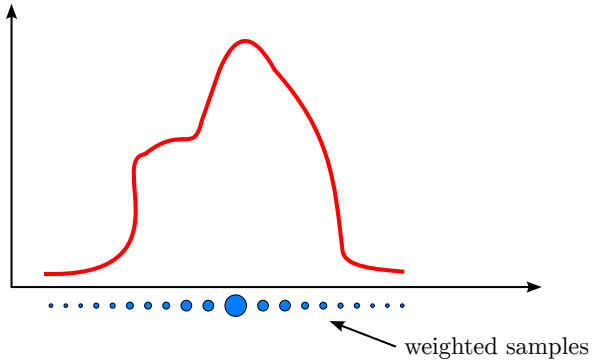
Utilizando Muestras (Partículas)

- **Múltiples muestras** para representar una distribución de probabilidad arbitraría
- Las muestras son están más agrupadas en algunas áreas en otras menos. La cantidad de partículas por unidad de área describe que tan probable es que el robot se encuentre en esa área.
- Cada muestra está acumulando un poco de "masa de probabilidad".
- Las muestra puede ser vista como una aproximación a la función de densidad de probabilidad (pdf).
- Para obtener la pdf, hay que integrar sobre una cierta área de manera de obtener la probabilidad matemática de que el robot se encuentre en dicha área.



Utilizando Muestras con Peso

- **Múltiples muestras con peso** para representar una distribución de probabilidad arbitraría
- Es posible reducir el número de muestras que necesitamos, si le agregamos pesos a cada muestra
- Mientras más peso tiene una muestra, más masa de probabilidad hay en esa región
- Los pesos de todas las partículas juntas deben sumar 1.
- Al inicio, podríamos agregarle a cada muestra un peso uniforme. Por ejemplo, si tenemos n muestras, entonces cada muestra tiene peso $\frac{1}{n}$



Filtro de Partículas (Particle Filter)

- Notar que es una aproximación de la pdf
- Es importante tener un número de muestras suficientes para poder representar la fdp adecuadamente.

Material para Particle Filter

- Información extraída de Vídeo de Cyrill Stachniss https://youtu.be/MsYlueVDLI0
- https://rse-lab.cs.washington.edu/projects/mcl/

TODO

UTILIZAR LAS SLIDES DEL SEMINARIO

Material

Vídeos para armar estas slides:

- Cyrill Stachniss Bayes: https://youtu.be/0lKHFJpaZvE
- Cyrill Stachniss KF y EKF: https://youtu.be/E-6paM_Iwfc
- Chebrolu EKF Localization: https://youtu.be/PiCC-SxWlH8
- Cyrill Stachniss Particle Filter: https://youtu.be/MsYlueVDLI0
- Slides de ECI 2012
- Seminario de curso: CS373 Udacity Programming a Robotic Car
- https://www.ipb.uni-bonn.de/html/teaching/photo12-2021/2021-pho2-14-ekf.pptx.pdf

Bibliografía

Capítulo 3 y 7 de: [1]
Shon and Lindsen: Manipulating the Multivariate Gaussian Density

[1] Sebastian Thrun, Wolfram Burgard y Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. isbn: 0262201623.