

Probabilidad Móvil.

Z = observación

U = acción

X = estado.

PROBABILIDAD
DE TRANSICIÓN
DE ESTADO

La ley probabilística que caracteriza la evolución de un estado está dada por una distribución:

$$P(X_t | X_0:t-1; Z_1:t-1; U_1:t)$$

Ahora bien, si el estado x es un estado completo entonces es un resumen suficiente de todo lo sucedido en los pasos anteriores. Por ende si conocemos el estado X_{t-1} solo importa la acción u_t .

$$\Rightarrow P(X_t | X_0:t-1; Z_1:t-1; U_1:t) = P(X_t | X_{t-1}, u_t)$$

A su vez uno podría querer generar un modelo para el proceso a partir del cual se obtienen las observaciones z . Si X_t es un estado completo tenemos independencia condicional.

PROBABILIDAD
DE
OBSERVACIÓN

$$P(Z_t | X_0:t; Z_1:t-1; U_1:t) \stackrel{?}{=} P(Z_t | X_t)$$

$\Rightarrow X_t$ es suficiente para predecir la observación, potencialmente ruidosa, Z_t .

\Rightarrow El conocimiento de cualquier otra variable es irrelevante si X_t es un estado completo.

BELIEF:

$$bel(X_t) = P(X_t | Z_1:t; U_1:t)$$

Probabilidad a posteriori del estado.

\Rightarrow Distribución de probabilidad sobre el estado X_t en el tiempo t condicionado a todas las observaciones anteriores $Z_1:t$ y todas las acciones anteriores $U_1:t$.

Se observa que el belief es obtenido una vez que se ha incorporado la observación z_t . En ciertas ocasiones es conveniente calcularlo antes de incorporar la observación z_t y justo luego de incorporar la acción u_t .

$$\bar{bel}(x_t) = P(x_t / z_{1:t-1}; u_{1:t})$$

Predicción: $\bar{bel}(x_t)$

Corrección: $\bar{bel}(x_t)$

Filtro de Bayes

Este algoritmo se encarga de calcular el belief a partir de la observación y de la acción.

El filtro es del tipo recursivo, es decir, el belief $bel(x_t)$ en tiempo t se calcula a partir del belief $bel(x_{t-1})$ en tiempo $t-1$.

Algoritmo Filtro-Bayes ($bel(x_{t-1})$, u_t , z_t)

for all x_t do

Predicción $\bar{bel}(x_t) = \int P(x_t / u_t, x_{t-1}) bel(x_{t-1}) dx$

Corrección $bel(x_t) = \prod P(z_t / x_t) \bar{bel}(x_t)$

end for

return $bel(x_t)$.

Para computar recursivamente el belief a posteriori, el algoritmo requiere un belief inicial $bel(x_0)$ en tiempo $t=0$. como condición de contorno. Si uno conoce el valor de x_0 con certeza, $bel(x_0)$ debería ser inicializado como una distribución con un único punto que concentre toda la masa de probabilidad en el correcto valor de x_0 .

y asignando cero de probabilidad en todas las obs restantes. Si no se conociera el valor inicial x_0 , $b(x_0)$ podría inicializarse usando una distribución uniforme sobre el dominio de x_0 .

Derivación matemática

Recordemos: $P(x_t | z_t, \mathcal{Z}) = \frac{P(\mathcal{Z} | x_t, z_t) P(x_t | z_t)}{P(\mathcal{Z} | z_t)}$

Buscamos calcular la distribución a posteriori $P(x_t | z_{1:t}; u_{1:t})$ a partir de la distribución a posteriori un tiempo anterior $t-1$, $P(x_{t-1} | z_{1:t-1}; u_{1:t-1})$.

$$\begin{aligned} P(x_t | z_{1:t}; u_{1:t}) &= \frac{P(z_t | x_t, z_{1:t-1}, u_{1:t}) P(x_t | z_{1:t-1}, u_{1:t})}{P(z_t | z_{1:t-1}, u_{1:t})} \\ &= \eta P(x_t | x_t; z_{1:t-1}, u_{1:t}) P(x_t | z_{1:t-1}, u_{1:t}) \end{aligned}$$

Si conocemos el estado x_t y estamos interesados en calcular la observación z_t , ninguna observación pasada ni acción nos provee de información adicional. Esto es:

$$P(z_t | x_t; z_{1:t-1}, u_{1:t}) = P(z_t | x_t)$$

Simplificamos así:

$$P(x_t | z_{1:t}, u_{1:t}) = \eta P(z_t | x_t) P(x_t | z_{1:t-1}, u_{1:t})$$

Por ende:

$$\bar{bel}(x_t) = \eta P(z_t/x_t) \bar{bel}(x_t).$$

Si expandimos el término $\bar{bel}(x_t)$:

$$\bar{bel}(x_t) = P(x_t/z_{1:t-1}; u_{1:t})$$

$$= \int P(x_t/x_{t-1}, z_{1:t-1}, u_{1:t}) P(x_{t-1}/z_{1:t-1}, u_{1:t}) dx_{t-1}$$

Si nuevamente, asumimos que el estado es completo podemos independizarnos de observaciones y acciones anteriores. En particular para $t-1$:

$$P(x_t/x_{t-1}, z_{1:t-1}, u_{1:t}) = P(x_t/x_{t-1}, u_t)$$

Así:

$$\bar{bel}(x_t) = \int P(x_t/x_{t-1}, u_t) P(x_{t-1}/z_{1:t-1}, u_{1:t}) dx_{t-1}$$

Obligando el control u_t :

$$\bar{bel}(x_t) = \int P(x_t/x_{t-1}, u_t) P(x_{t-1}/z_{1:t-1}, u_{1:t-1}) dx_{t-1}.$$

Finalmente:

$$\bar{bel}(x_t) = \eta P(z_t/x_t) \int P(x_t/x_{t-1}, u_t) P(x_{t-1}/z_{1:t-1}, u_{1:t-1}) dx_{t-1}$$

$$\boxed{\bar{bel}(x_t) = \eta P(z_t/x_t) \int P(x_t/x_{t-1}, u_t) \bar{bel}(x_{t-1}) dx_{t-1}.}$$

Sustitución de Parkou

$$P(z_t | x_{0:t}; z_{1:t-1}, u_{1:t}) = P(z_t | x_t)$$

$$P(x_t | x_{s:t-1}, z_{s:t-1}, u_{s:t}) = P(x_t | x_{t-1}, u_t).$$

→ Si conozco el estudio en un cierto tiempo t, me independizo de todas las observaciones y acciones pasadas.

Suposiciones: •) Punto estático •) Punto independiente.
•) Modelo perfecto, sin errores de aproximación

Filtros Gaussianos

Para gaussianas tenemos:

$$P(x) \sim N(\mu, \sigma^2)$$

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}\right\}$$

$$P(x) \sim N(\mu, \Sigma)$$

$$P(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right\} \quad | \text{ Multivariate }$$

-) Los beliefs están representados por distribuciones normales multivariadas.

FILTRO DE KALMAN

El filtro de Kalman representa beliefs a partir de la representación de movimientos.: en el tiempo t , el belief está representado por la media μ_t y la matriz de covarianza Σ_t .

PROPIEDADES.

- 1) La probabilidad del cambio de estado $P(x_t/x_{t-1}, u_t)$ debe ser una función lineal del siguiente tipo:

$$x_t = A_t x_{t-1} + \beta_t u_t + \epsilon_t$$

Donde:

A_t : matriz de $n \times n$ que describe como el estado evoluciona de $t-1$ a t sin control ni ruido.

β_t : matriz de $n \times l$ que describe como el control u_t cambia el estado de $t-1$ a t

ϵ_t, η_t = Variables aleatorias que representan el ruido de proceso y medición que se asumen independientes y normalmente distribuidas con covarianzas Ω_t y η_t respectivamente.

$$P(x_t/x_{t-1}, u_t) = \det(2\pi \Omega_t)^{-1/2} \exp \left\{ -\frac{1}{2} (x_t - A_t x_{t-1} - \beta_t u_t)^T \cdot \Omega_t^{-1} (x_t - A_t x_{t-1} - \beta_t u_t) \right\}$$

2) La probabilidad de observación $P(z_t/x_t)$, también debe ser una función lineal.

$$z_t = C_t x_t + s_t$$

Donde:

C_t : Matriz de $k \times n$ que describe como aparecer el estado x_t a una observación z_t .

s_t : idem a ϵ_t .

$$P(z_t/x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1} (z_t - C_t x_t)\right\}$$

3) El belief inicial $bel(x_0)$ debe estar normalmente distribuido. Su media y matriz de covarianza son p_0 y \bar{z}_{10}

$$bel(x_0) = p(x_0) = \det(2\pi \bar{z}_{10})^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x_0 - p_0)^T \bar{z}_0^{-1} (x_0 - p_0)\right\}.$$

⇒ Estas tres propiedades son suficientes para asegurar que el posteriori $bel(x_t)$ es siempre gaussiano, para cualquier tiempo t .

Algoritmo.

Algoritmo Filtro-Kalman ($p_{t-1}, \bar{z}_{t-1}, u_t, z_t$)

$$\bar{p}_t = A_t p_{t-1} + B_t u_t$$

$$\bar{z}_t = A_t \bar{z}_{t-1} A_t^T + R_t$$

$$K_t = \bar{z}_t C_t^T (C_t \bar{z}_t C_t^T + Q_t)^{-1}$$

$$p_t = \bar{p}_t + K_t (z_t - C_t \bar{p}_t)$$

$$\bar{z}_t = (I - K_t C_t) \bar{z}_{t-1}$$

return p_t, \bar{z}_t

El filtro de Kalman representa el belief $\bar{bel}(x_t)$ en tiempo t a partir de la media μ_t y la matriz de covarianza \bar{Z}_t . La entrada del filtro es el belief en tiempo $t-1$, representado por $\bar{\mu}_{t-1}$ y \bar{Z}_{t-1} . Para actualizar estos parámetros, el filtro de Kalman requiere el control u_t y la observación z_t . La salida es el belief en tiempo t , representado por $\bar{\mu}_t$ y \bar{Z}_t .

La predicción del belief $\bar{\mu}_t$ y \bar{Z}_t es calculado representando el belief $\bar{bel}(x_t)$ un tiempo después, pero antes de incorporar la observación z_t . El belief se obtiene al incorporar la acción u_t . La ganancia k_t es conocida como ganancia de Kalman.

Derivación matemática

PARTIE 1: PREDICCIÓN

El belief $\bar{bel}(x_t)$ es calculado a partir del belief un tiempo antes $\bar{bel}(x_{t-1})$.

$$\bar{bel}(x_t) = \underbrace{\int P(x_t | x_{t-1}, u_t)}_{\bar{bel}(x_{t-1})} dx_{t-1}$$

$$NN(x_t; A_t x_{t-1} + B_t u_t; \bar{\mu}_t) \quad NN(x_{t-1}; \bar{\mu}_{t-1}, \bar{Z}_{t-1})$$

$$\begin{aligned} \bar{bel}(x_t) &= \eta \int \exp \left\{ -\frac{1}{2} (x_t - A_t x_{t-1} - B_t u_t)^T \bar{P}_t^{-1} (x_t - A_t x_{t-1} - B_t u_t) \right\} \\ &\quad \cdot \exp \left\{ -\frac{1}{2} (x_{t-1} - \bar{\mu}_{t-1})^T \bar{Z}_{t-1}^{-1} (x_{t-1} - \bar{\mu}_{t-1}) \right\} dx_{t-1} \end{aligned}$$

Así:

$$\begin{aligned} \bar{bel}(x_t) &= \int \bar{\mu}_t = A_t \bar{\mu}_{t-1} + B_t u_t \\ &\quad \bar{Z}_t = A_t \bar{Z}_{t-1} A_t^T + \bar{P}_t \end{aligned}$$

PARTIE 2: ACTUALIZACIÓN DE LA OBSERVACIÓN

$$bel(x_t) = \underbrace{\eta}_{\sim N(z_t; C_t x_t; Q_t)} \underbrace{bel(x_t)}_{\sim N(x_t; \bar{\mu}_t; \bar{\Sigma}_t)}$$

$$\sim N(z_t; C_t x_t; Q_t) \sim N(x_t; \bar{\mu}_t; \bar{\Sigma}_t)$$

$$bel(x_t) = \eta \exp \left\{ -\frac{1}{2} (z_t - C_t x_t)^T Q_t^{-1} (z_t - C_t x_t) \right\} \cdot \\ \cdot \exp \left\{ -\frac{1}{2} (x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1} (x_t - \bar{\mu}_t) \right\}$$

Entonces:

$$bel(x_t) = \begin{cases} \bar{\mu}_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\ \bar{\Sigma}_t = (I - K_t C_t) \bar{\Sigma}_t \end{cases}$$

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

RESUMEN:

-) Los parámetros describen el belief del estado del sistema
-) MUY EFICIENTE: polinómico en la dimensión de la medición K y la dimensión del estado n :
 $O(K^{2,346} + n^2)$
-) Óptimo para sistemas gaussianos lineales
-) La mayoría de los sistemas robóticos son no-lineales
-) Solo puede modelar beliefs unimodales.

FILTRO DE KALMAN EXTENDIDO (EKF).

La mayoría de los problemas reales de robótica incluyen funciones no lineales por lo que el filtro de Kalman no podría utilizarse.

El filtro de Kalman extendido asume que la probabilidad del próximo estado y las probabilidades de las observaciones están gobernadas por funciones no lineales g y h respectivamente:

$$x_t = g(u_t, x_{t-1}) + \epsilon_t$$

$$z_t = h(x_t) + \delta_t$$

•) Funciones no lineales llevan a distribuciones no gaussianas

⇒ El filtro de Kalman ya no se puede utilizar.

⇒ Linearización local.

La función g reemplaza a las matrices A_t y B_t y la función h reemplaza a la matriz C_t . Desafortunadamente, con funciones g y h arbitrarias, el belief ya no es gaussiano. Realizar la actualización del belief es usualmente imposible para funciones g y h no lineales en el sentido de que el filtro de Bayes no posee una solución cerrada.

El filtro extendido de Kalman (EKF) calcula una aproximación al verdadero belief. Representa dicha aproximación a través de una gaussiana. En particular, el belief $bel(x_t)$ en tiempo t es representado por la media μ_t y la covarianza Σ_t . Entonces, el EKF hereda del filtro de Kalman la representación básica del belief, pero difiere en que el belief es solamente una aproximación

g no es exacto como en el caso de los filtros de Kalman.

LINEALIZACIÓN Vía EXPANSIÓN DE TAYLOR

EKF utiliza un método llamado expansión de Taylor de primer orden. La expansión de Taylor construye una aproximación lineal a la función g a partir de valores y pendientes de g 's. La pendiente está dada por la derivada parcial:

$$g'(x_{t-1}, u_t) = \frac{\partial g(x_{t-1}, u_t)}{\partial x_{t-1}}$$

PREDICCIÓN:

$$g(x_{t-1}, u_t) \approx g(u_t, p_{t-1}) + \frac{\partial g(u_t, p_{t-1})}{\partial x_{t-1}} (x_{t-1} - p_{t-1})$$

$$g(x_{t-1}, u_t) \approx g(u_t, p_{t-1}) + g_t (x_{t-1} - p_{t-1})$$

Corrección:

$$h(x_t) \approx h(\bar{p}_t) + h'(\bar{p}_t) (x_t - \bar{p}_t)$$

FuncióN
LINEAL

$$h'(\bar{p}_t) = \frac{\partial h(\bar{p}_t)}{\partial x_t}$$

$$h(x_t) \approx h(\bar{p}_t) + H_t (x_t - \bar{p}_t)$$

Donde g_t y H_t son matrices Jacobianas:

-) Son la orientación del plano tangente a una función vectorial en un punto dado.
-) Es la generalización del gradiente.

ALGORITMO

$\bar{\mu}_t = g(\bar{x}_t, \bar{\mu}_{t-1})$ $\bar{\Sigma}_t = G_t \bar{\Sigma}_{t-1} G_t^T + R_t$ $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$ $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$ $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$ return μ_t, Σ_t .	KF $\bar{\mu}_t = A\bar{\mu}_{t-1} + B\bar{x}_{t-1}$ $\bar{\Sigma}_t = A\bar{\Sigma}_{t-1}A^T + R$ $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$ $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$ $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
---	---

PESIMISMO:

- Solución ad-hoc para tratar las no-línealidades
- Linealiza localmente en cada paso.
- Funciona bien para no linearidades suavizadas

UNSCENTED KALMAN FILTER (UKF)

UNSCENTED TRANSFORP. Pasos

- 1) Calcular un conjunto de puntos sigma
- 2) A cada punto sigma se le asocia un peso
- 3) Se transforman cada punto con la transformación no-lineal
- 4) Se calcula una gaussiana a partir de los puntos pesados.

Aclaración: Se evita linealizar alrededor de la media
como en el caso de Taylor (EKF).

Sean: $X^{[i]} = \text{puntos sigma}$; $w^{[i]} = \text{pesos}$

Se tiene que cumplir que:

$$\sum_i w^{[i]} = 1$$

o) No hay una única

solución a $X^{[i]}, w^{[i]}$

$$\mu = \sum_i w^{[i]} X^{[i]}$$

$$\tilde{\Sigma} = \sum_i w^{[i]} (X^{[i]} - \mu)(X^{[i]} - \mu)^T$$

Puntos sigma: elección

1: dimensiona $X^{[0]} = \mu$ Primer punto: la media
lidad

$$X^{[i]} = \mu + (1/(n+\lambda) \tilde{\Sigma})^T_i \quad \text{Para } i=1, \dots, n$$

$$X^{[i]} = \mu - (1/(n+\lambda) \tilde{\Sigma})_{i-n}^T \quad \text{Para } i=n+1, \dots, 2n$$

vector columna.

El número de puntos sigma depende de la dimensión del sistema.
La fórmula general es $2N+1$, donde N es la dimensión.

λ es el factor de escala el cual nos dice avan alejado de la media
deberíamos elegir nuestros puntos sigma. (se recomienda $\lambda=3n$).

Puntos sigma: pesos

$$w_{mu}^{[0]} = \frac{\lambda}{n+\lambda} ; \quad w_c^{[0]} = w_{mu}^{[0]} + (1-\alpha^2/\beta)$$

$$w_{mu}^{[i]} = w_c^{[i]} = \frac{1}{2(n+\lambda)} \quad \text{Para } i=1, \dots, 2n$$

w_c : cálculo para covarianza

w_{mu} : cálculo para media

λ, α y β : parámetros

Recoverar la gaussiana: Calcular una gaussiana de los puntos transformados y sus pesos

Predicción

$$\hookrightarrow \mu' = \sum_{i=0}^{2n} w_{\mu i}^{(i)} g(x^{(i)}) \quad \text{Corrección: } g(x^{(i)}) \rightarrow h(x^{(i)}) \\ \mu' \rightarrow \hat{x} ; R_t \rightarrow Q_t$$

$$\hookrightarrow S_i^1 = \sum_{i=0}^{2n} w_{\mu i}^{(i)} (g(x^{(i)}) - \mu') (g(x^{(i)}) - \mu')^T + R_t$$

Parámetros: son libres (no hay solución única).

Comúnmente:

$$K \geq 0$$

$$\alpha \in (0, 1]$$

$$\lambda = \alpha^2 (n + K) - n$$

$$\beta = 2.0 \quad \text{Valor óptimo para gaussinas}$$

Dicen que tan

lejos están los puntos de μ

RESUMEN

- U_t es una alternativa de linearización
- U_t es una mejor aproximación que la expansión de Taylor
- U_t usa la propagación de puntos sigma
- Existen parámetros libres
- UKF usa U_t en los pasos de predicción y corrección

UKF VS EKF

-) Resultados iguales para modelos lineales
-) UKF aproxima mejor que EKF para modelos no lineales
-) No se necesita calcular jacobianos
-) Tiene orden de complejidad
-) UKF es ligeramente más lento que EKF.
-) UKF también está restringido a distribuciones gaussianas.

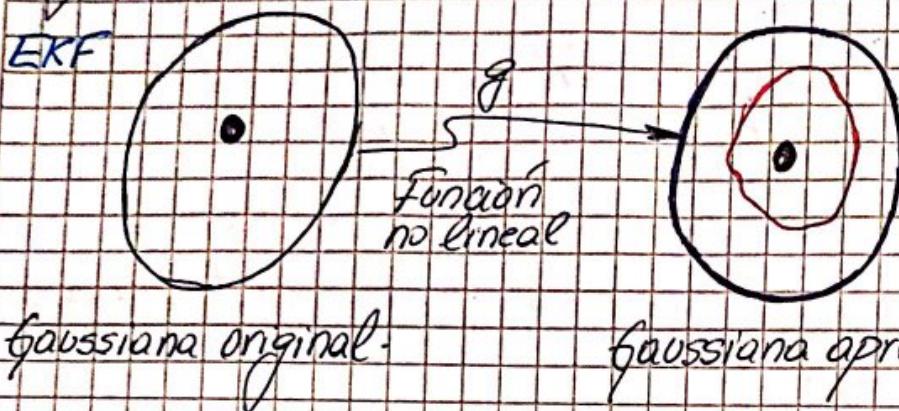
Hasta ahora:

KF: es una herramienta para predecir valores haciendo uso de un conjunto de ecuaciones matemáticas bajo las hipótesis de que nuestra data se encuentra en la forma de una distribución gaussiana y de que aplicaríamos ecuaciones lineales a dicha distribución gaussiana.

EKF: en el mundo real, tenemos ecuaciones no lineales, debido a que podemos estar prediciendo en una dirección mientras que nuestro sensor está tomando lectura en alguna otra dirección, por lo que involucra ángulos y funciones seno y coseno las cuales son no lineales. Entonces EKF se ayuda con las series de Taylor para aproximar linealmente una función no lineal alrededor de la media de la gaussiana y luego predice los valores.

UKF: ¿Por qué? Performance

Para EKF se toma tan solo un punto para aproximar una nueva función lineal a partir de una función no lineal: la media de la gaussiana.



■ Gaussiana luego de pasar por g .

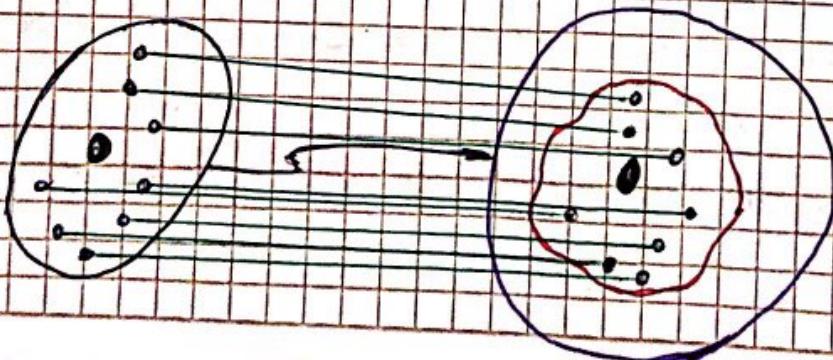
■ Mejor aproximación a la gaussiana

→ Tenemos un solo punto para aproximar la gaussiana

En UKF entra en juego el concepto de puntos sigma. Tomaremos algunos puntos de la gaussiana original y los maparemos luego de pasar los puntos a través de alguna función no lineal y luego calcularemos la media y covarianza de la gaussiana transformada.

Puede resultar difícil transformar la complejidad de la distribución de un estado a través de una función no lineal pero resulta muy sencillo transformar algunos puntos individuales, estos son los puntos sigma. Dichos puntos sigma son representativos de la distribución completa.

UKF



FILTRO DE PARCÍCULAS

El filtro de Kalman y sus variantes solo pueden ser utilizados para modelizar distribuciones gaussianas. Ahora bien los filtros de partículas representan eficientemente distribuciones no-gaussianas y su principio básico radica en:

-) Conjunto de estado hipótesis (partículas = muestras)
-) Supervivencia del más apto.

⇒ Busco poder representar distribuciones arbitrarias (no solo gaussianas).

El filtro de partículas es una implementación alternativa al filtro de Bayes pero es del tipo no paramétrico. La idea clave de este filtro es representar el a posteriori $p(x_t)$ por medio de un conjunto de muestras de estado obtenidas de dicha a posteriori. En lugar de representar la distribución de una forma paramétrica (una distribución normal), los filtros de partículas representan a la distribución por un conjunto de muestras obtenidas de dicha distribución. Si bien dicha representación es aproximada, también es no paramétrica, por lo que puede representar un espacio mucho más amplio de distribuciones que, por ejemplo, los filtros gaussianos.

Una de las ideas principales es utilizar estas múltiples muestras para representar las distribuciones arbitrarias y se cumple que cuanta más partículas hay en un intervalo, mayor es la probabilidad

dad de ese intervalo.

Tenemos entonces un conjunto de muestras pesadas las cuales representan la distribución posterior.

$$\mathcal{X} = \{(x^{[j]}, w^{[j]})\}_{j=1 \dots J}$$

Estados hipótesis

Peso de importancia

$$P(x) = \sum_{j=1}^J w^{[j]} \delta_{x^{[j]}}(x).$$

•) Cada partícula es una instantaneación concreta del estado en tiempo t , es decir, una hipótesis de lo que el verdadero estado podría ser en tiempo t .

•) Cada partícula es un posible estado en el cual se puede encontrar el sistema.

•) Cuanto más poblado se encuentra una subregión del espacio de estados por muestras, entonces, es más probable que el verdadero estado caiga dentro de dicho régión

→ los conjuntos de partículas pueden usarse para aproximar funciones

↳ Cuanto más muestras mayor será la aproximación.

⇒ La pregunta es: ¿Cómo obtengo dichas muestras?

PRINCIPIO DE MUESTREO POR IMPORTANCIA

Tenemos una distribución f , llamada objetivo (target), de la cual queremos obtener muestras pero no podemos generar las de una forma cerrada. Para lograr dichas muestras utilizamos una distribución g , llamada propuesta (proposal), de la cual si podemos obtener muestras como por ejemplo, una gaussiana.

Entonces generamos las muestras de la distribución g y realizamos una corrección teniendo en cuenta las diferencias entre g y f usando un peso de importancia : $w = f/g$

Hasta ahora tenemos : acerca del filtro de partículas.

-) Variante del filtro de Bayes recursivo
-) Aproximación no paramétrica
-) Se modelan las distribuciones a partir de muestras
-) Predicción: muestras de la propuesta
-) Corrección: Pesar las muestras: objetivo/propuesta

PREDICCIÓN: la propuesta suele ser en base al modelo de movimiento (odometría). \rightarrow PROPUESTA = ODOMETRÍA.

CORRECCIÓN: se realiza en base a las observaciones, es decir, al modelo de medición

Algoritmo

Consta de 3 pasos:

- 1.- Muestrear partículas usando la distribución propuesta
- 2.- Calcular los pesos de importancia.

$$w = \frac{\text{target}}{\text{propuesta}}$$

- 3.- Repuesteo: reemplazar muestras poco probables por otras más probables

↳ Muestreo con reemplazo.

↳ Supervivencia de las muestras más apropiadas, es decir, de las muestras más pesadas.

1. Algoritmo Filtro-Partículas (X_{t-1}, u_t, z_t)

1. $\bar{X}_t = X_t = \emptyset$.

2. For $j=1$ to J do

3. sample $X_t^{(j)} \sim \pi(x_t)$

4. $w_t^{(j)} = P(X_t^{(j)}) / \pi(x_t^{(j)})$

5. $\bar{X}_t = \bar{X}_t + (X_t^{(j)}, w_t^{(j)})$

6. end for

7. For $j=1$ to J do

8. draw i with probability $\alpha w_t^{(j)}$

9. add $X_t^{(i)}$ to X_t

10. end for

11. return \bar{X}_t

0. Inputs: conjunto de muestras anterior: X_{t-1} (belief anterior)

Acción: u_t ; observación: z_t

1. Inicializo los dos nuevos conjuntos de muestras en cero.

X_t : conjunto de predicción

\bar{X}_t : conjunto de corrección (belief final = ourPost).

2. Itero sobre todas las muestras del belief anterior.

3. Obtengo una muestra de la distribución propuesta

4. Aplico la corrección calculando el peso de importancia

5. Agrego la muestra al conjunto de predicción. \bar{X}_t

7-10. - Remuestreo o repuesteo con importancia

Se incorporan los pesos de importancia al proceso de remuestreo.

lo que genera que la distribución de las partículas cambie.

ANTES: $bel(X_t)$

DESPUES: $bel(\bar{X}_t) = \eta P(Z_t | X_t^{(i,j)}) bel(X_t)$.

) Cada partícula es una pose potencial del robot y el conjunto de partículas approxima la distribución a posteriori de la pose.

) El remuestreo asegura que las partículas quedan en áreas apropiadas del espacio de estados ya que hay un número finito de partículas

REMUESTREO.

•) Reemplaza muestras poco probables por otras más probables

•) Supervivencia de las más apropiadas.

•) El truco es evitar que muchas muestras ocupen estados poco factibles

•) Necesario debido a que contamos con un número limitado de muestras

.) Para lidiar con errores de localización:

- 1.- Agregar aleatoriamente un número finito de muestras
- 2.- Insertar muestras aleatorias de manera proporcional al proporcional del likelihood de las partículas.

RESUMEN: Filtro de partículas (FP):

.) Son filtros bayesianos recursivos y no paramétricos

↳ No tenemos una distribución paramétrica para describir la distribución a posteriori si no muestras aleatorias.

.) Obtenemos las muestras de una distribución propuesta para avanzar al próximo estado y luego procedemos al proceso de corrección para tener en cuenta que la distribución propuesta no es la distribución objetivo que queremos aproximar.

.) El paso final es el paso de renuestreo

.) Representan la densidad de probabilidad a posteriori a través de un conjunto de muestras pesadas.

.) Pueden modelar distribuciones no gaussianas

RESUMEN: LOCALIZACIÓN CON FILTRO DE PARTÍCULAS

.) Utiliza partículas para representar la pose del robot ($x; y; \theta$)

.) Distribución propuesta: modelo de movimiento

.) Las muestras se pesan según el likelihood de las observaciones

Esta técnica se llama localización Monte-Carlo

- .) En el reencuestreo se toman nuevas partículas con una probabilidad proporcional al likelihood de las observaciones.

MAPEO

Motivación:

- .) Aprender mapas es uno de los problemas fundamentales de la robótica móvil.
- .) Los mapas permiten localización, hacer tareas de manera eficiente, etc.
- .) Los sistemas robóticos complejos se basan en mapas para estas tareas.
- .) Es una representación del entorno sobre el cual voy a trabajar.

FEATURES

- .) Representación compacta, con solo las coordenadas x e y ya puedo identificar la ubicación de un landmark.
- .) Observaciones múltiples de los features mejoran la estimación de la localización de los landmarks (EKF).

MÁS DE GRILLA:

La idea es poder representar la información de si hay algo o no para cada posible sitio en el entorno. Para ello:

- .) Discretizamos al mundo en celdas y cada celda se corresponde con una pequeña área del entorno.
- .) La estructura de grilla es rígida.
- .) Cada celda se asume que puede estar ocupada o vacía.

Donde ocupada = obstáculo y vacía = espacio libre

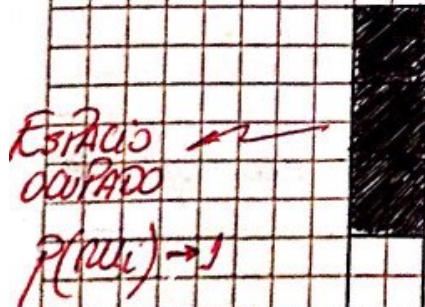
↳ Cuando hablamos de ocupada o vacía hacen referencia a toda el área abarcada por la celda
⇒ Completamente ocupada o completamente vacía

-) Es un modelo no-paramétrico de cada celda, guarda la información de si se encuentra ocupado o vacío
-) Grandes mapas requieren considerable memoria de almacenamiento (desventaja)
-) No depende de la detección de features (ventajas)

⇒ FEATURES: solo especifican la forma del entorno en las localizaciones específicas, es decir, la localización de objetos contenidos en el mapa

⇒ Volumétricos: no solo dan información de los objetos comprendidos en el entorno sino también acerca de la ausencia de objetos (espacios libres)

Sustitución 1: el área que corresponde a una celda está o bien completamente libre o completamente ocupada.



•) Cada celda es una variable aleatoria binaria que modela la ocupación

Sustitución 2: las celdas (U_{AS}) son independientes unas de otras

Probabilidad de Ocupación (Celda = Una binaria \rightarrow modela ocupación)

-) Celda ocupada: $P(\mu_i) = 1$
-) Celda no ocupada: $P(\mu_i) = 0$
-) Celda sin información: $P(\mu_i) = 0,5$
-) El entorno se asume estático.

\hookrightarrow Los obstáculos no se mueven. Si una celda está ocupada luego siempre estará ocupada (lo mismo sucede si está vacía)

REPRESENTACIÓN

La distribución de probabilidad del mapa está dada por el producto de las distribuciones de probabilidad de las celdas individuales.

$$\text{Mapa} \leftarrow P(\mu) = \prod_i P(\mu_i) \rightarrow \text{celdas}$$

independencia de celdas

\hookrightarrow Ej: Saber el estado de μ_1 no me

μ_i = Variable aleatoria binaria

ayuda nada para saber μ_2 o
algunas otras celdas

μ = Multivariable.



Estado:
1 de
4 dimensiones

Mapa
(Vector 2×2)

4 celdas independientes

← Pareo con Poses concoidas.

Estructuración de un parámetro a partir de datos

Dados los datos de un sensor: $z_{s,t}$ y las poses $x_{s,t}$ del sensor, estimar el mapa:

$$P(u_i / z_{s,t}; x_{s,t}) = \prod_j P(u_{ij} / z_{s,t}; x_{s,t})$$

⇒ Filtro de Bayes binario
(para un estado estático).

No realiza procedimientos de \Leftarrow El mundo es estático por lo que no necesita hacer una predicción (el mundo no cambia).
de corrección.

•) No tenemos acciones u pero si tenemos observaciones z .

Entonces buscamos resolver: $P(u_i / z_{s,t}; x_{s,t})$

Aplicamos Bayes:

$$P(u_i / z_{s,t}; x_{s,t}) = \frac{P(z_s / u_i; z_{s,t-1}; x_{s,t}) P(u_i / z_{s,t-1}; x_{s,t})}{P(z_s / z_{s,t-1}; x_{s,t})}$$

Si conozco el estado anterior no me interesan las observaciones anteriores ni las poses anteriores:

$$P(z_s / u_i; z_{s,t-1}; x_{s,t}) = P(z_s / u_i; x_t)$$

Luego se las mediciones hasta $t-1$ ($z_{1:t-1}$) pero conozco las poses hasta t (x_t). Entonces dado que no realice observación de la pose x_t puedo no tenerla en cuenta y así:

$$P(m_i | z_{1:t-1}, x_{1:t}) = P(m_i | z_{1:t-1}; x_{1:t-1})$$

Queda así:

$$\frac{P(m_i | z_{1:t}, x_{1:t})}{P(z_t | m_i, x_t)} = \frac{P(z_t | m_i, x_t) P(m_i | z_{1:t-1}, x_{1:t-1})}{P(z_t | z_{1:t-1}, x_{1:t})}$$

Ahora quito el primer término del numerador y aplico Bayes nuevamente:

$$P(z_t | m_i, x_t) = \frac{P(m_i | z_t, x_t) P(z_t | x_t)}{P(m_i | x_t)}$$

Lo coloco en la original.

$$P(m_i | z_{1:t}, x_{1:t}) = \frac{P(m_i | z_t, x_t) P(z_t | x_t) P(m_i | z_{1:t-1}, x_{1:t-1})}{P(m_i | x_t) P(z_t | z_{1:t-1}, x_{1:t})}$$

Si sé donde el robot estuvo (x_t) pero no tengo observación (z_t) entonces no me sirve saber dicha pose: $P(m_i | x_t) = P(m_i)$

$$P(m_i | z_{1:t}, x_{1:t}) = \frac{P(m_i | z_t, x_t) P(z_t | x_t) P(m_i | z_{1:t-1}, x_{1:t-1})}{P(m_i) P(z_t | z_{1:t-1}, x_{1:t})}$$

Supongamos que era un filtro bayesiano, es decir, los estados posibles, ocupado o vacío. Entonces el procedimiento para el evento opuesto (\bar{n}_{ui}) es el mismo:

$$P(\bar{n}_{ui} | z_{1:t}, x_{1:t}) = \frac{P(n_{ui} | z_t, x_t) P(z_t | x_t) P(\bar{n}_{ui} | z_{1:t-1}, x_{1:t-1})}{P(n_{ui}) P(z_t | z_{1:t-1}, x_{1:t})}$$

Ahora tomemos el cociente entre los terminos:

$$\frac{P(n_{ui} | z_{1:t}, x_{1:t})}{P(\bar{n}_{ui} | z_{1:t}, x_{1:t})} = \frac{P(n_{ui} | z_t, x_t) P(z_t | x_t) P(\bar{n}_{ui} | z_{1:t-1}, x_{1:t-1})}{P(n_{ui}) P(z_t | z_{1:t-1}, x_{1:t-1})}$$

$$\frac{P(\bar{n}_{ui} | z_{1:t}, x_{1:t})}{P(n_{ui} | z_{1:t}, x_{1:t})} = \frac{P(\bar{n}_{ui} | z_t, x_t) P(z_t | x_t) P(n_{ui} | z_{1:t-1}, x_{1:t-1})}{P(\bar{n}_{ui}) P(z_t | z_{1:t-1}, x_{1:t})}$$

$$P(n_{ui} | z_{1:t}, x_{1:t}) = P(n_{ui} | z_t, x_t) P(n_{ui} | z_{1:t-1}, x_{1:t-1}) P(\bar{n}_{ui})$$

$$P(\bar{n}_{ui} | z_{1:t}, x_{1:t}) = P(\bar{n}_{ui} | z_t, x_t) P(\bar{n}_{ui} | z_{1:t-1}, x_{1:t-1}) P(n_{ui})$$

$$P(n_{ui} | z_{1:t}, x_{1:t}) = P(n_{ui} | z_t, x_t) P(n_{ui} | z_{1:t-1}, x_{1:t-1}) (1 - P(n_{ui})) =$$

$$P(\bar{n}_{ui} | z_{1:t}, x_{1:t}) = P(\bar{n}_{ui} | z_t, x_t) P(\bar{n}_{ui} | z_{1:t-1}, x_{1:t-1}) P(n_{ui})$$

$$= P(n_{ui} | z_t, x_t) \cdot P(n_{ui} | z_{1:t-1}, x_{1:t-1}) \cdot (1 - P(n_{ui}))$$

$$\underbrace{1 - P(n_{ui} | z_t, x_t)}_{\text{solo utiliza la pose de observación actual}} \cdot \underbrace{P(n_{ui} | z_{1:t-1}, x_{1:t-1})}_{\text{usa todas las poses y observaciones anteriores.}} \cdot \underbrace{P(n_{ui})}_{\text{como es el mapa en general}}$$

solo utiliza la pose de observación actual

usa todas las poses y observaciones anteriores.

como es el mapa en general
No hay pose no observación

\Rightarrow término actual

\Rightarrow término recursivo

\Rightarrow prior

$$P(x) = \frac{1}{1 + \exp(-l(x))}$$

Notación Log odds: $l(x) = \log\left(\frac{P(x)}{1 - P(x)}\right)$

Aplíco dicha notación a mi ecuación:

$$l(\text{nu}_i / z_{1:t}, x_{1:t}) = l(\text{nu}_i / z_t, x_t) + l(\text{nu}_i / z_{0:t-1}, x_{1:t-1}) + l(\text{nu}_i)$$

$\log \left[\frac{P(\text{nu}_i / z_{1:t}, x_{1:t})}{1 - P(\text{nu}_i / z_{1:t}, x_{1:t})} \right]$

inverse sensor model

termino recursivo prior

Se suele escribir como:

$$\left\{ l_{t,i} = \text{inv.-sensor-model}(\text{nu}_i, z_t, x_t) + l_{t-1,i} - l_0 \right\}.$$

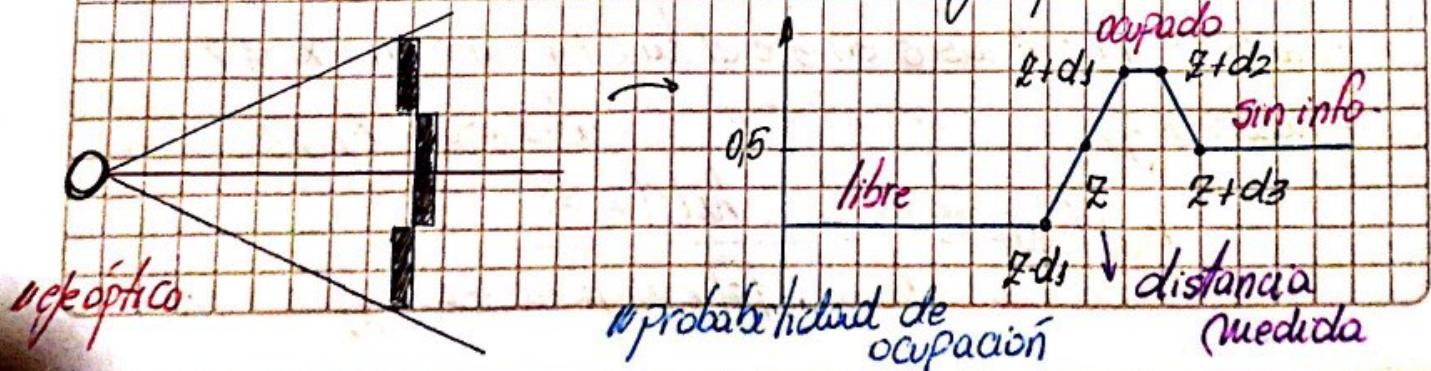
\Rightarrow MAPAS DE GRILLA DE OCUPACIÓN

- Originalmente hecho para ultrasonidos muy ruidosos.
-) También llamado "mapeo con poses conocidas".

Modelo Inverso de Sensor para Sonar

$$l(\text{nu}_i / z_t, x_t) = \text{inv.-sensor-model}(\text{nu}_i, z_t, x_t).$$

Volvemos a considerar solo las celdas sobre el eje óptico



- El mapa de máximos likelihood se obtiene redondeando las probabilidades de cada celda a 0 ó 1.

Para un mapa

Prec
Prior
Pree

RESUMEN: Mapeo de grilla

- Los mapas de grilla son un modelo muy usado para representar el entorno.
- Las grillas de ocupación discretizan el espacio en celdas independientes
- Cada celda es una variable aleatoria binaria (ocupado/vacio)
- Estimamos el estado de cada celda usando un filtro de Bayes binario (de estado estacionario).
- Esto permite un algoritmo eficiente para el mapeo con posos conocidas.
- El modelo "log odds" es rápido de calcular (computo sencillo).

Alejandra: modelo de conteo.

Para cada celda contar.

- $hits(x; y)$: #Casos en que el haz terminó en $(x; y)$
- $misses(x; y)$: #Casos en que el haz no pasó por $(x; y)$

$$Bel((x; y)) = \frac{hits(x; y)}{hits(x; y) + misses(x; y)}$$

\Rightarrow te interesa saber: $P(\text{reflexión}(x, y))$

-) El cálculo del mapa más probable se reduce a contar qué tan a menudo una celda refleja una medición (hits) y qué tan a menudo la celda es atravesada por el haz (misses).

Grilla de ocupación vs conteo

-) Conteo \rightarrow qué tan a menudo una celda refleja un haz.
-) Ocupación \rightarrow celda ocupada por un objeto o no.
-) Aunque una celda puede estar ocupada por un objeto, la probabilidad de reflexión de ese objeto puede ser baja.

Resumen:

-) Los mapas de probabilidad de reflexión son una representación alternativa.
-) La idea principal del modelo de sensores calcular para cada celda la probabilidad de que refleje el haz del sensor.
-) Dado el modelo del sensor, contar la frecuencia con que el haz intercepta o atraviesa una celda da un modelo de mapas nómico likelihood.

SLAIA: LOCALIZACIÓN Y PAREO SIMULTÁNEO.

Este método hace referencia a estimar la pose del robot y el mapa del entorno al mismo tiempo. Es una técnica difícil debido a que se necesita un mapa para la localización y a su vez una buena estimación de la pose es necesaria para hacer un mapeo.

Localización: inferir pose dado un mapa. (estimuar la trayectoria)

PARTE: inferir un mapa dadas las poses. (modelar el entorno).

Start: aprender un cuadro y localizar el robot en el cuadro simultáneamente.

→ Por lo general si en un problema de localización o con grupos estanques resolvemos de fondo el problema porque se necesita mucha información tanto para un caso como para el otro y no se suele contar con ella.

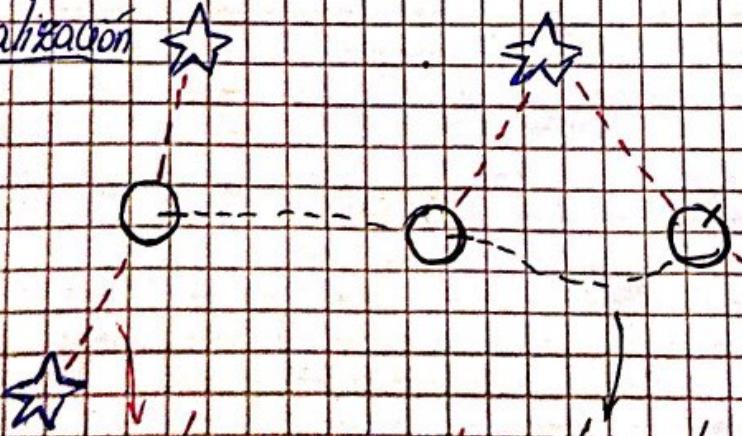
⇒ Localización y ruapeo suelen interactuar entre ellos, más aún cuando no hay un modelo dado

Landscape: es el algo que el robot puede observar y reconocer

Ej.: Un árbol podría ser uno Landmark.

Ejemplo de

Localización



Observaciones de trayectoria
los landmarks del robot

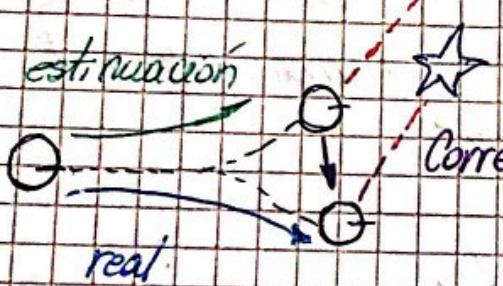
Si el robot puede estimar la
pose de los landmarks, luego
puede estimar su posición
relativa a dichos
landmarks.

★ Landmarks

Localización es saber cuáno es el entorno, es decir, saber cuáno es el mapa pero no sabemos donde se encuentra el robot. Entonces en otras palabras, sabemos las posiciones de los landmarks (conocemos las estrellas) y luego queremos estimar donde estuvo el robot.

→ Utiliza pasos de corrección para hay variaciones respecto de la trayectoria verdadera

Ej: El robot esta midiendo que el landmark se encuentra a una distancia de 1m, pero de acuerdo con el mapa debería estar a tan solo 15cm. → hay falta de coincidencia.



La cantidad de corrección va a depender de la precisión del sensor y de la precisión de la ejecución del movimiento

El robot circula a través del entorno, realiza los comandos de control (acciones), estima donde el robot probablemente está, luego hace una observación, ve landmarks en el entorno y así puede corregir su pose.

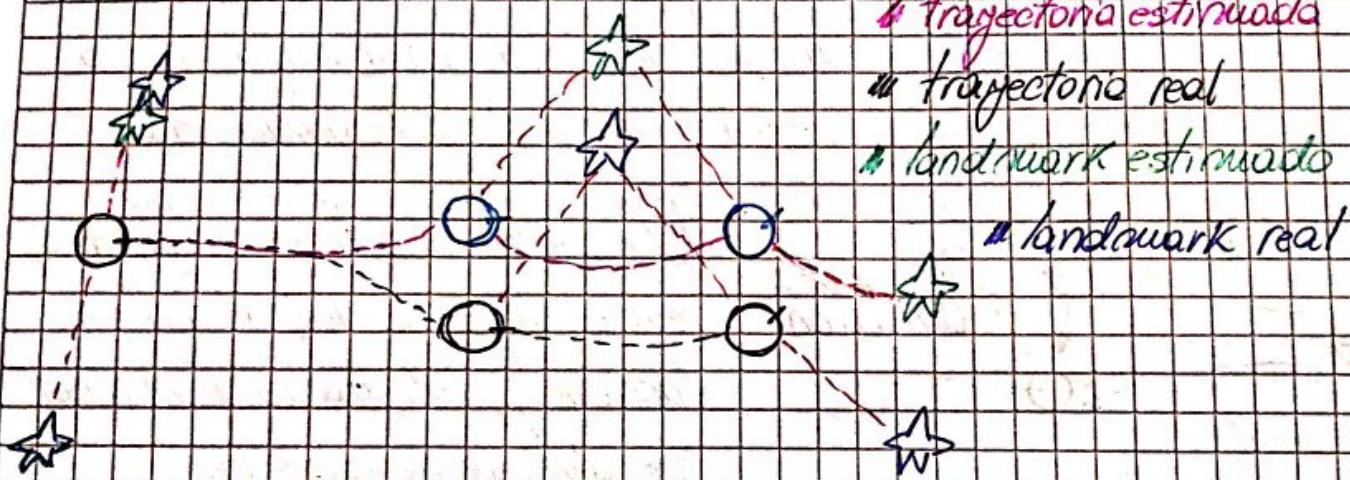
Ejemplo de mapa

En este caso lo que busco es estimar los landmarks sabiendo las poses del robot. Entonces suponemos que de alguna manera sabemos para cada tiempo donde estuvo. Luego el robot realiza observaciones moviéndose por el entorno, creando así un mapa marcando donde cada feature se encuentra (estrella).

 landmark
 verdadero
 landmark
 observado

Veamos entonces que la estimación del estado del mundo no es perfecta. La posición de los landmarks no son realmente donde se encuentran. Es un resultado que se atribuye al sensor ruidoso. Sabiendo la incertezza del sensor se puede armar un entorno alrededor del landmark.

Ejemplo de SLAM



→ SLAM es un problema del tipo "el huevo y la gallina"

- ✓ ↳ Un mapa es requerido para localización
- ✓ ↳ La estimación de pose es requerido para hacer un mapa

→ Hay dependencia entre el modelo del entorno y cuan bien puede el robot localizarse a si mismo

localización



Mapeo.

⇒ NAVEGACIÓN AUTÓNOMA.

\Rightarrow El camino del robot y el mapa son ambos desconocidos

6) El error en la ubicación del robot se correlaciona con errores en el mapa.

SLAM BASADO EN FEATURES ($\text{feature} = \text{landmark}$).

DADOS:

.) Señales de control de robot

$$U_{1:K} = \{u_1, u_2, \dots, u_K\}$$

.) Observaciones relativos

$$Z_{1:K} = \{z_1, z_2, \dots, z_K\}$$

RUMBO

.) Un mapa de features

$$M = \{m_1, m_2, \dots, m_n\}$$

.) El recorrido del robot

$$X_{1:K} = \{x_1, x_2, \dots, x_K\}$$

Cuento con:

.) poses de robot absolutas.

.) posiciones del landmarks absolutas.

.) solo mediciones relativas de landmarks

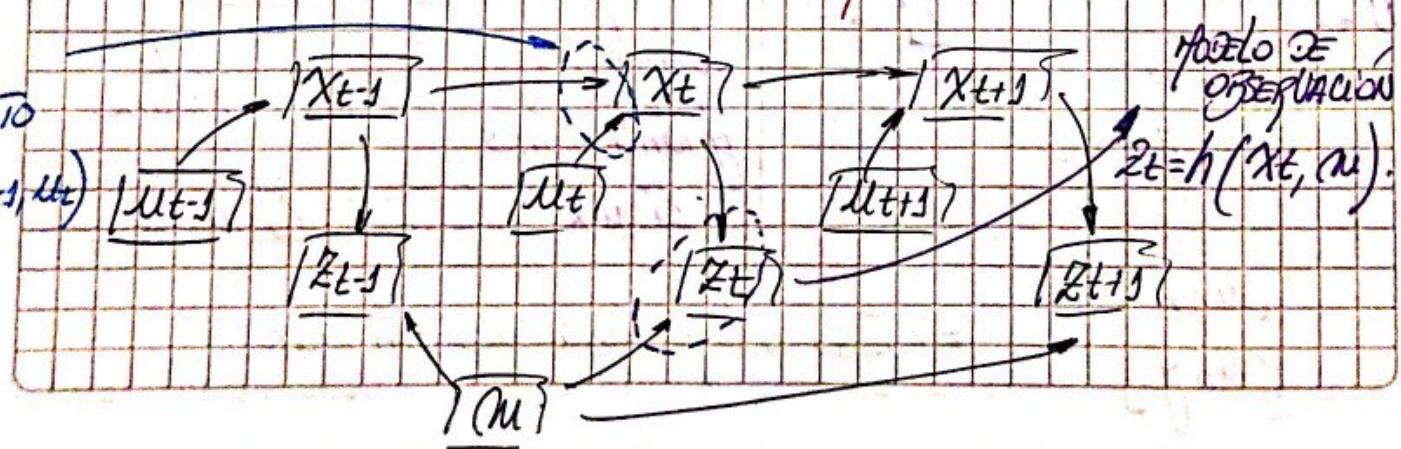
Full SLAM: $P(x_{0:t}, m / z_{1:t}, u_{1:t})$

Estimua el trayecto completo y el mapa.

Online SLAM: $P(x_t, m / z_{1:t}, u_{1:t})$

Estimua la pose mas reciente y el mapa.

MODELO
DE
MOVIMIENTO
 $x_t = f(x_{t-1}, u_t)$



Modelo de movimiento: que dice como el robot se mueve

Modelo de observación: que dice como debo interpretar mis observaciones

SLAM EKF

Es la aplicación del filtro de Kalman EKF a SLAM, en donde queremos estimar tanto la pose del robot $(x, y; \theta)$ como la posición de landmarks en el entorno asumiendo correspondencias conocidas.

Pose Robot : $(x, y; \theta)$. Landmarks : (m_x, m_y) .

Para un plano en 2D, el espacio de estados es:

$$x_t = (x, y, \theta, m_{1x}, m_{1y}, \dots, m_{nx}, m_{ny})^T$$

Pose Robot

Landmark 1

\Rightarrow Los landmarks extienden el estado

Resulta entonces que para un mapa con n landmarks tenemos una distribución gaussiana de dimensión $(3+2n)$. El belief se encuentra representado por:

$$\mu = \begin{pmatrix} x \\ y \\ \theta \\ m_{1x} \\ m_{1y} \\ \vdots \\ m_{nx} \\ m_{ny} \end{pmatrix}$$
$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{x1} & \cdots & \Sigma_{xn} \\ \Sigma_{1x} & \Sigma_{11} & & \Sigma_{1n} \\ \vdots & & \ddots & \\ \Sigma_{nx} & \Sigma_{n1} & \cdots & \Sigma_{nn} \end{pmatrix}$$

Matriz de Cov de la pose del robot

Correlación entre poses y landmarks

Correlación entre poses y landmarks

Incerteza de los landmarks

\Rightarrow Esto es lo que buscamos estimar

Ciclo de filtro

1) Predicción de estado (colorimetría).

Tenemos el control o acción y estimamos la nueva posición del robot dado el control.

2) Predicción de la medición.

Evaluamos la función H en la medida predicha. Esto nos dice lo que esperamos observar.

3) Medición

Tenemos la medida/observación

4) Asociación de datos

Determinar a qué landmark le adjunto la observación.

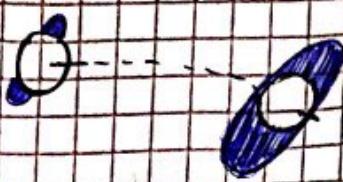
Necesitamos computar la diferencia entre la observación estimada y la observación obtenida.

5) Actualización

Paso de corrección

6) Integración de nuevos landmarks.

1) Predicción de Estado: El robot se mueve y vemos representada su incertidumbre \Rightarrow predecimos el siguiente estado



El comando de movimiento solo cambia la pose del robot, no cambia la posición de los landmarks. Esto es asumiendo que el robot no modifica el entorno.

\Rightarrow El robot solo modifica el estado y no el entorno

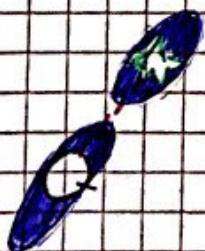
Entonces se actualiza: \bullet (actualizo)

$$\begin{pmatrix} x_R \\ \vdots \\ x_n \end{pmatrix} \quad \left(\begin{array}{cc} z_{1xx} & z_{1xnu} \\ z_{1nxu} & z_{1nuu} \end{array} \right)$$

• Complejidad computacional: $O(n)$

\Rightarrow lineal con la cantidad de landmarks.

2) Predictión de medición: toma el estado actualizado y ve qué tipo de observación debería obtener



\Rightarrow Transformación de forma global a local

$$\hat{z}_k = h(\hat{x}_k)$$

3) Medición: landmark en punto (x, y) , z_k

\hookrightarrow medición real

4) Asociación de Datos: Asocia la predicción prebucha \hat{x}_k^i con la observación \hat{z}_k^j
→ Diferencia entre ambas

5) Actualización: Se modifican todas las matrices, se ejecutan las expresiones típicas del filtro de Kalman

.) complejidad computacional: $O(n^2)$
→ Cuadrática con el número de landmarks.

6) NUEVOS LANDMARKS: Se aumenta el estado

$$\begin{pmatrix} x \\ u_1 \\ \vdots \\ u_n \\ u_{n+1} \end{pmatrix} \quad \begin{pmatrix} \hat{x}_{xx} & \hat{x}_{xu_{n+1}} \\ \hat{x}_{u_{n+1}x} & \hat{x}_{u_{n+1}u_{n+1}} \end{pmatrix}$$

CIERRE DE LAZO

Hace referencia a reconocer un área ya mapeada, en general, después de una vuelta de exploración (el robot "cierra el lazo").

⇒ El robot vuelve a un área donde ya ha estado y reconoce los features que ya ha visto.

⇒ Es una especie de paso de asociación de datos. bjp.

.) altos niveles de ambigüedad

.) tener cuidado con entornos simétricos

Lo difícil saber si está revisando o si es nuevo.

→ Al cerrar un lazo las incertezas "colapsan", es decir, se achican considerablemente (ya sea un cierre correcto o no).

↳ gracias a las correlaciones entre poses y landmarks

↳ Revisitar áreas ya recorridas hace que la incertezas en la estimación de la pose del robot y los landmarks se reduzca.

→ Permite obtener mejores ruinas (más precisas)

→ Cerrar cada el lazo lleva a la divergencia del filtro.

Complejidad: $n = \# \text{ landmarks}$.

.) Costo por paso: $O(n^2)$.

.) Costo total para hacer un ruina con n landmarks: $O(n^3)$

.) Consumo de memoria: $O(n^2)$

.) Problema: computacionalmente intractable para ruinas grandes.

Resumen

.) Convergencia para el caso lineal gaussiano.

.) Puede divergir si las no linearidades son grandes

.) Solo puede lidiar con un solo lazo.

.) Exitoso para entornos de mediana escala.

FASI SLAM: basado en features

FILTRO DE PARCÍCULAS (repaso).

-) Técnica de estimación no paramétrica, es decir, que no cuenta con una forma paramétrica (ej: distribución gaussiana) para describir una distribución de probabilidad.
-) Representan belief con muestras aleatorias a las cuales se le asignan un peso. La cantidad de muestras que caen dentro de una región es proporcional a la probabilidad de dicha región.
-) Pueden modelar distribuciones arbitrarias.
-) Procedimiento de 3 pasos (principio de muestreo con importancia)
 - i) Muestrear una nueva generación de partículas (muestrear de la distribución propuesta)
 - ii) Asignar un peso de importancia a cada partícula
 - iii) Repermuestrear: redistribuye las muestras, \therefore reemplaza muestras poco probables por otras más probables. (survivencia de las más apropiadas)
-) Conjunto de muestras pesadas: $X = \{x^{[i]}, w^{[i]}\}_{i=1, \dots, N}$

↳ Pensar a las muestras como una hipótesis acerca del estado.

Para localización: $x = (x; \theta)$

Para slay:

Dimension alta

Estado: $x = (x_1:t, \underbrace{w_{1,x}, w_{1,y}, \dots, w_{1,x}, w_{1,y}}_{\text{Poses}}, \underbrace{\theta_1, \dots, \theta_M}_{\text{landmarks}})^T$

) Queremos estimar la pose del robot junto con la ubicación de los landmarks (mapa completo).

→ Problema: El número de partículas necesarias para representar el belief crece exponencialmente con la dimensión del espacio de estados.

↳ Los filtros de partículas son efectivos en espacios de estados de dimensiones pequeñas.

↳ La idea es ver si existen dependencias entre ciertas dimensiones del espacio de estados

⇒ Si hay dependencias: resuelvo más eficientemente

vii

Si hay dependencia entre las variables: $x_1:t, w_1, \dots, w_M$?

Si sabemos las poses del robot, entonces podemos mappear fácilmente:

$$x_1:t, \underbrace{w_1, \dots, w_M}_{\text{Map}}$$

La idea entonces es utilizar el filtro de partículas solo para representar las poses del robot y luego para cada muestra computar el mapa de landmarks individualmente.

⇒ Mapeo con poses conocidas!

→ La idea es decir: la partícula sigue esta trayectoria y el entorno se ve de la siguiente manera y así sucesivamente con los siguientes partículas.

↳ Rao-Blackwellizierung

Paoz Blackwellizacin

Es una factorización para aprovechar la dependencia entre variables.

$$P(a,b) = P(a) P(b|a) \cdot \text{Prob. condicional}$$

Si $P(b|a)$ puede calcularse en forma cerrada, representar solo $P(a)$ con muestras y calcular $P(b|a)$ para cada muestra.

En nuestro caso:

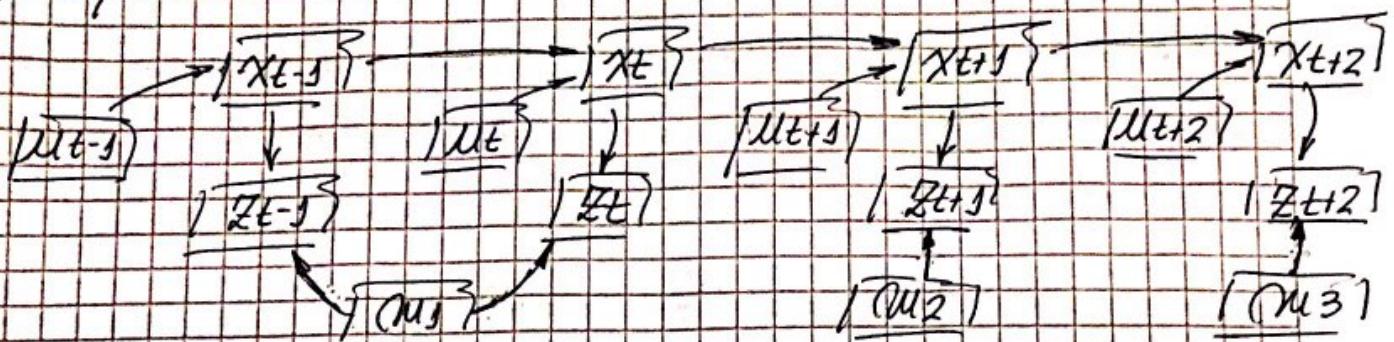
- a: trayectoria del robot
- b: mapa

⇒ Necesito que las cuivestras solo cubran las regiones de a y no las de a y b.

→ Discrepui las dimensiones.

Así:

GRÁFICO AMPLIADO:



⇒ los m_i están descondicionados, es decir, los landmarks son condicionalmente independientes dada las poses o trayectoria del robot.

Luego:

$$P(x_{0:t}, \theta_{1:T} | z_{0:t}, m_{1:t}) = P(x_{0:t} | z_{0:t}, m_{1:t}) \prod_{i=1}^T P(m_i | x_{0:t}, z_{i:t})$$

Problema de
localización

Posterior de la
trayectoria del
robot

Posiciones de
landmarks condicio-
nalmente independientes.

P_T

→ El segundo término no puede calcularse eficientemente.

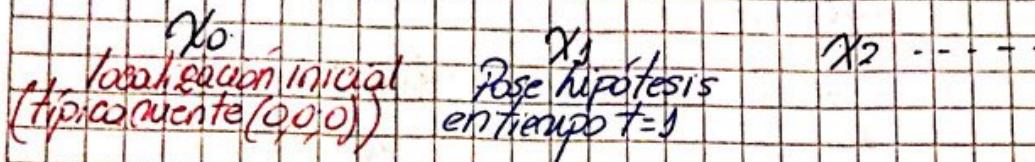
→ Puedo usar filtro de partículas.

FASISLAT:

- Filtro de partículas Rao-Blackwellizado basado en landmarks.
- Cada landmark se representa con un EKF de 2×2 .
- Cada partícula lleva consigo M EKFs.

Para modelar la trayectoria del robot:

- .) Rrepresentación basada en muestras para $P(x_{0:t} | z_{1:t}, u_{1:t})$
- .) Cada muestra es una hipótesis.



- .) Muestras pasadas de poses no son tenidas en cuenta
- .) No es necesario mantener las poses pasadas en el conjunto de muestras.

Partícula 1 $(x_1; \theta; \text{Landmark}_1, \dots, \text{Landmark}_M)$

Partícula 2 $(x_1; \theta; \text{Landmark}_1, \dots, \text{Landmark}_M)$

⋮

Partícula N $(x_1; \theta; \text{Landmark}_1, \dots, \text{Landmark}_M)$

Secuencia:

Tengo cada partícula con los poses (trayectorias) γ los grupos estandares de cada muestra individual, es decir, los landmarks (EKF's 2×2). Entonces:

- 1) El robot se mueve de acuerdo al modelo de movimiento de odometría
- 2) Tengo en cuenta la información que me brinda mi sensor para actualizar el belief.

⇒ obtengo la medición del landmark.

3) Tomo las observaciones realizadas y las traduzco en pesos de importancia

Lo lo puedo pensar como: que tan bien la partícula estima el mapa dadas las observaciones realizadas en el paso anterior

⇒ cuanto mejor sea la estimación del mapa respecto de la observación, entonces mayor peso de importancia (mayor likelihood también).

4) Se actualiza el mapa de cada muestra

⇒ obtengo el nuevo belief actualizado

Complejidad

Actualizar partículas (robot)
basándose en el control $O(N)$

Incorporar una observación en
los filtros de Kalman (dado la
asociación de datos) $O(N)$

Renuestrear el conjunto de
partículas $O(NM)$

$$N = \# \text{partículas}$$

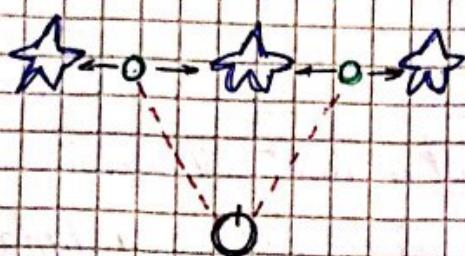
$$M = \# \text{landmarks}$$

$$O(NM)$$

$$\xrightarrow{\text{Mejorado}} O(N \log M)$$

Asociación de Datos: Problema

¿Qué observación pertenece a qué landmark?



• landmark • observación

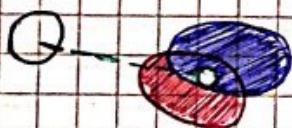
⇒ Se ve que hay más de una posible asociación de datos

↳ Depende de la pose del robot

Muti-hipótesis: Cada partícula puede hacer una asociación de datos distinta: las partículas que hacen una buena asociación de datos son más tendenciosas a sobrevivir en el proceso de renuestro.

⇒ El error de la pose del robot se elimina de las decisiones de asociación de datos.

Asociación por Partícula



•) La observación fue generada por el landmark rojo o el azul?

$$P(\text{observación}/\text{azul}) = 0,3$$

Criterios para la asociación:

$$P(\text{observación}/\text{rojo}) = 0,7$$

1) Elegir el más probable. → rojo.

•) Si la probabilidad es muy baja, generar un landmark nuevo.

2) Elegir una asociación aleatoria pesada por el likelihood de observaciones

→ Belief multi-modal ⇒ gran ventaja frente a EKF

Resumen

•) Usa el tro de partículas para modelar el belief.

•) Factoriza la posterior de SLAM en problemas de estimación de baja dimensión.

•) Modela solo la trayectoria del robot a través del muestreo
↳ cada partícula contiene una trayectoria distinta

⇒ Computa los landmarks dadas las trayectorias de las partículas

⇒ Asociación de datos por partícula

↳ No hay incertezza en la pose del robot para la asociación de datos por partícula. \textcircled{A}

•) Complejidad (implementación simple): $O(NM)$.

•) Ventajas sobre método EKF clásico (especialmente con no linearidades).

$\textcircled{1}$

•) Robusto en ambigüedades para asociación de datos

Fast SLAM: basado en grillas

Usa los conceptos de Fast SLAM para construir un mapa de grilla

- Al igual que con landmarks, el mapa depende de las poses del robot al adquirir datos
- Si las poses son conocidas, el mapeo basado en grillas es trivial ("mapeo con poses conocidas").

→ ¿Puede resolverse el SLAM si no hay landmarks predefinidas disponibles?

RAO - Blackwellización

Factorización de lo posterior:

$$P(x_0:t, \omega | z_{1:t}, u_{1:t}) = P(x_0:t | z_{1:t}, u_{1:t}) \underbrace{P(\omega | x_{1:t}, z_{1:t})}_{\substack{\text{poses} \\ \text{mapa} \\ \text{observaciones}}} \underbrace{\text{trayectoria posterior}}_{\substack{\text{ocasiones} \\ \text{(filtro de partículas)}}} \underbrace{\text{Posterior del mapa}}_{\substack{\text{(dada la trayectoria)} \\ \text{Localización)}}$$

.) Cada partícula representa una trayectoria posible del robot. A su vez, cuantifica su propia mapa y lo actualiza según un "mapeo con poses conocidas"

↳ Cada una sobrevive con una probabilidad proporcional al likelihood de las observaciones relativas a su propio mapa.

Limiaciones

•) Cada cuadro es muy grande al usar cuadros de grilla

•) Cada particula contiene su propio cuadro, por lo que el numero de particulas debe ser bajo.

→ Solución: calcular mejores distribuciones propuestas

Idea: mejorar la estimación de la pose antes de aplicar el filtro de partículas.

⇒ Uso muchos de escaneo (SCAN MATCHING).

SCAN MATCHING

La idea es alinear el escaneo en tiempo t con el escaneo en $t-1$, es decir, ajustar localmente las poses para que el escaneo se ajuste bien.

⇒ se busca hacer coincidir o cuadrar el cuadro construido hasta el momento con el escaneo del entorno.

Corrección de pose con scan matching:

Maximizar el likelihood de la pose y cuadro en el tiempo t relativos a la pose y cuadro en el tiempo $t-1$:

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}} \left\{ P(z_t / x_t, u_{t-1}) \cdot P(x_t / u_{t-1}; \hat{x}_{t-1}) \right\}$$

medición actual

cuadro construido hasta
el momento

acción del robot

⇒ La incertidumbre se reduce porque tenemos en cuenta la observación así como también la información de la odometría.
↳ Mejora la odometría.

FASTSLAM con odometría (mejorada)

- .) Scan-matching brinda una corrección de pose localmente consistente.
- .) Pre-corregir secuencias cortas de odometría usando scan-matching y usarla como entrada a FASTSLAM.
- .) Se necesitan menos partículas, ya que el error de entrada es menor.

⇒ Scan matching se usa para transformar secuencias de mediciones láser en mediciones de odometría

⇒ Al basarse en grillas pocas cuanear entornos más grandes en "horizonte real"

Lo ideal es usar una mejor distribución propuesta, una que considere la observación más reciente:

$$x_t^{(k)} \sim P(x_t | z_{1:t}, x_{1:t-1}, u_{1:t}, z_{1:t})$$

⇒ Considero la observación actual en la distribución propuesta del filtro de partículas para hacer el scan matching al realizar la nueva generación de partículas y luego tener en cuenta la observación al momento de pesar la muestra

- v) Muestreo más preciso.
- v) Obtengo un mejor mapa
- v) Necesito menos partículas

Distribución propuesta de posa

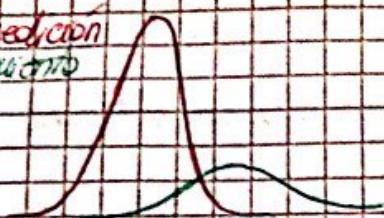
• Modelo de medición "Modelo de movimiento"

$$P(x_t^{(i)} | x_{t-1}^{(i)}, m_t^{(i)}, z_t, u_t) = \frac{P(z_t | x_t^{(i)}, m_t^{(i)}) P(x_t^{(i)} | x_{t-1}^{(i)}, u_t)}{\int P(z_t | x_t^{(i)}, m_t^{(i)}) P(x_t^{(i)} | x_{t-1}^{(i)}, u_t) dx_t}$$

probabilidad de la pose
dada la info. recolectada

normalización

• medición
movimiento



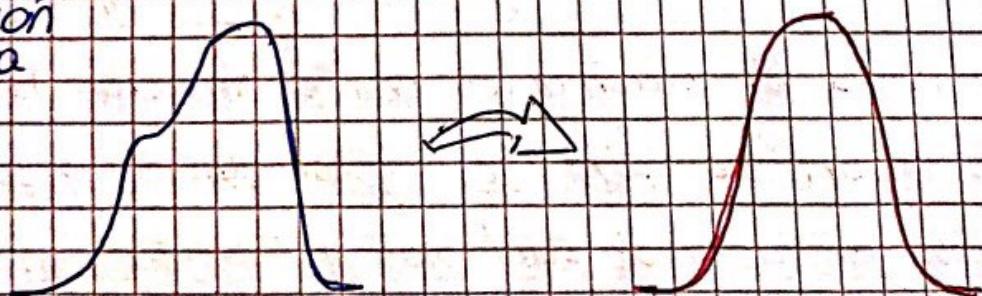
→ Para losers $P(z_t | x_t^{(i)}, m_t^{(i)})$ es muy estrecha y por ende domina en el producto
↳ estimación bien

- El modelo de medición nos da una estimación local, porque su forma de pico, es decir, limita el área de integración **local**
- El modelo de movimiento tiene una distribución chata pero nos limita de manera global

Aproximamos la ecuación con una gaussiana:

$$P(x_t^{(i)} | x_{t-1}^{(i)}, m_t^{(i)}, z_t, u_t) \sim N(\mu^{(i)}, \sigma^{(i)})$$

• Distribución verdadera



→ Utilizo la aproximación gaussiana para la distribución propuesta

Táso a poso:

- 1) Asumimos que la distribución verdadera luce como la azul ~~■~~ (no es gaussiana)
 - 2) Necesito encontrar el área abarcada por la distribución, porque no sabemos en qué parte se encuentra el robot.
 - 3) Tenemos una buena estimación gracias al modelo de odometría, entonces aplicamos scan matching
 - ⇒ si obtenemos entonces el máximo reportado por el scan matching
 - 4) Muestreagamos puntos alrededor de dicho máximo, los cuales tendrán un peso de acuerdo a la altura de la distribución.
 - 5) La distribución verdadera no se encuentra lejos de ser una gaussiana ~~■~~. Toco la gaussiana y obtengo muestras de ella y uso eso como mi distribución propuesta
 - ⇒ Crea la próxima generación de muestras,
 - ... ⇒ obtuve mi distribución propuesta para el filtro de partículas.

⇒ Antes tenía solo el modelo de movimiento (odometría) pero ahora incorporo la observación más reciente. Para ello toco dicha observación y la integro en la distribución propuesta.

Viene del producto entre el modelo de movimiento y el modelo de medición

→ Hay sobre un área pequeña (local) que me interesa donde esta distribución se encuentra por encima de cero. Por ende solo quiero integrar en dicha área.

↳ Necesito entonces una forma que sea adecuada para el muestreo. Aproximemos así, a través de una distribución gaussiana y la utilizaremos como nuestra distribución propuesta.

⇒ Puedo muestrear eficientemente porque es una gaussiana y tiene en cuenta la observación y el modelo de odometría.

→ La distribución propuesta mejorada se adapta a la estructura del entorno.

REMUVESTREO.

- .) Remuestrear de la distribución propuesta mejorada reduce los efectos del remuestreo.
- .) Remuestrear en cada paso limita la "memoria" del filtro
 -) Se pierde diversidad

→ Objetivo: reducir el número de acciones de remuestreo

⇒ REMUESTREO SELECTIVO.

Si bien el remuestreo es peligroso debido a que se pueden perder muestras que son importantes (problema de agotamiento de partículas) también es necesario para lograr la convergencia. La idea del remuestreo solo tiene sentido si los pesos de importancia de las partículas difieren significativamente.

→ La pregunta es: ¿Cuándo renuestro?

Número de Partículas efectivas

Medida de que tan bien se approxima la distribución objetivo con muestras tomadas de la distribución propuesta:

- Describe "la varianza en los pesos de las partículas"

$$N_{eff} = \frac{1}{\sum_i (w_t^{(i)})^2}$$

Si las partículas fueron normalizadas, es decir, sus pesos fueron normalizados:

$$\sum_{i=1}^n w_t^{(i)} = 1 \Rightarrow N_{eff} \in [1, n]$$

⇒ Es máxima para pesos iguales y en ese caso, la approximación de muestras es parecida al objetivo.

⇒ Si nuestra approximación es parecida al objetivo, luego no necesitamos renuestrear

⇒ La respuesta es: Solo se renuestreo cuando N_{eff} cae por debajo de un umbral, generalmente $N/2$.

Si $N_{eff} > N/2 \rightarrow$ no renuestreo.

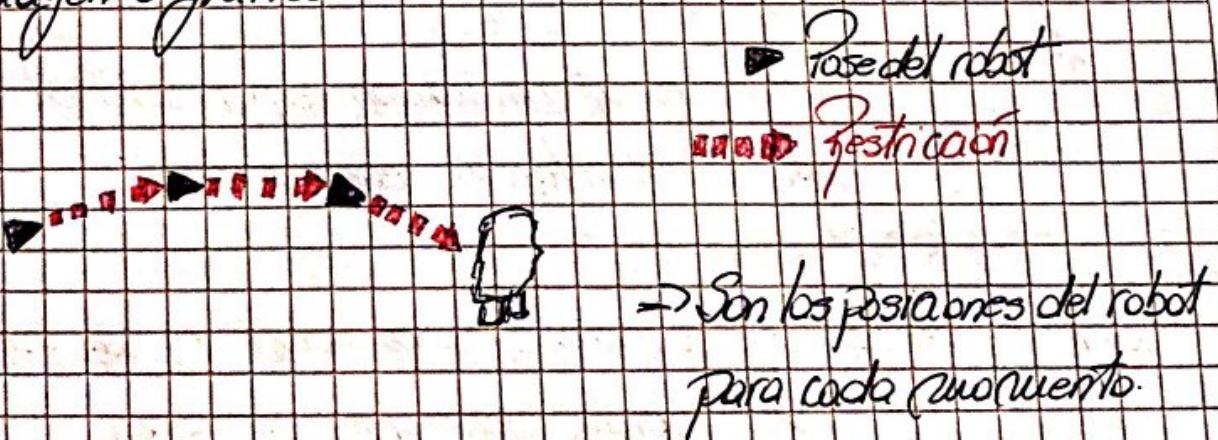
Si $N_{eff} < N/2 \rightarrow$ renuestreo.

Conclusiones

- .) El método de Fastslam también se puede aplicar a mapas de grillas
- .) Se incorpora la observación más reciente en la distribución propuesta para obtener una versión mejorada y más eficiente.
- .) Simular el scan-matching aplicado a cada partícula
- .) La cantidad de partículas y pasos de resuestro pueden ser reducidos significativamente.

Slam: basado en grafos

Tenemos un robot que se desplaza por el entorno o ambiente y para cada tiempo o luego de haber viajado una cierta distancia crea la siguiente imagen o gráfico:



Vemos entonces las poses del robot para cada movimiento, los cuales se conectan por medio de restricciones y dichas restricciones tienen incertezas. (son ruidosas).

Cada vez que el robot revisita un área conocida o un área del entorno donde ya ha estado se generan restricciones entre poses no sucesivas.

-) Las restricciones pueden provenir de:
 - i) modelo de observación
 - ii) observaciones (cierre de lazo).



IDEAS

- v) Usar un grafo para representar el problema.
- v) Cada nodo en el grafo corresponde a una pose del robot durante el mapeo ►
- v) Cada arco entre nodos corresponde a una restricción espacial entre ellos ►

⇒ Solución basada en grafos: Armar el grafo y encontrar la configuración de nodos que minimiza el error introducido por las restricciones.

EL GRÁFO:

Consiste de n nodos individuales $X = X_1:n$ donde cada X_i es una transformación 2D o 3D, es decir, cada X_i es la pose del robot en el tiempo t_i .

Una restricción o arco existe entre dos nodos X_i y X_j si:

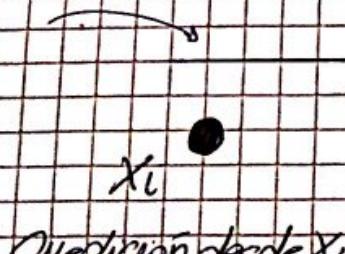
1) Proximidad de odometría \Rightarrow el robot se mueve de x_i a x_{i+1}



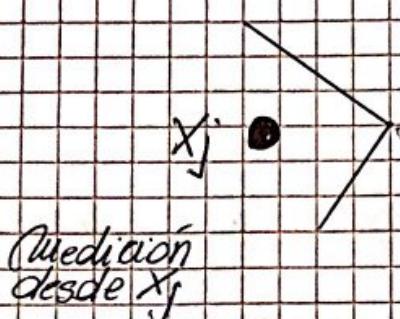
El arco representa la
medición de odometría

2) Proximidad de observaciones \Rightarrow el robot observa la misma región
del entorno desde x_i y desde x_j

Pared



(Medición desde x_i)



(Medición
desde x_j)

• Son muy similares
 x_i y x_j

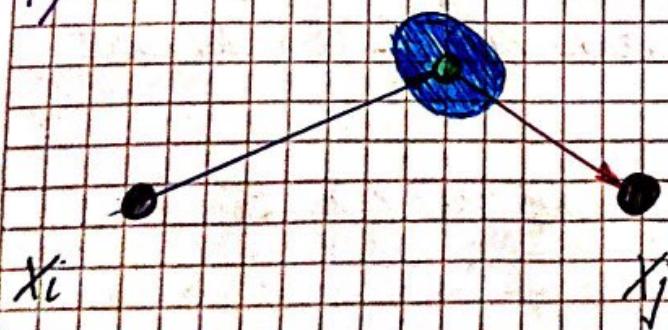
\rightarrow La pregunta es
cómo relacionarlos
y qué tan lejos está
 x_j de x_i

Construir una medición virtual de la posición de x_j visto desde x_i

El arco representa la posición de x_j vista
desde x_i basada en la observación.



Grafo de Pose



Observación de x_j desde x_i

Error: $e_{ij}(x_i, x_j)$

Incerteza en la observación

$$\text{Objetivo: } \hat{x}^* = \underset{x}{\operatorname{arg\,min}} \sum_{ij} e_{ij}^T S_{ij} e_{ij}$$

$$S_{ij} = \tilde{Z}_{ij}^{-1}$$

Ahora bien: $\tilde{e}_{ij}^T S_{ij} \tilde{e}_{ij}$ coincide con el exponente de la distribución gaussiana

\Rightarrow Minimizar ese término es minimizar nuestra distribución gaussiana

Concepto: Una vez armado el grafo, determinaremos el mapa más probable corrigiendo los nodos y así luego podremos armar el mapa basado en poses conocidas

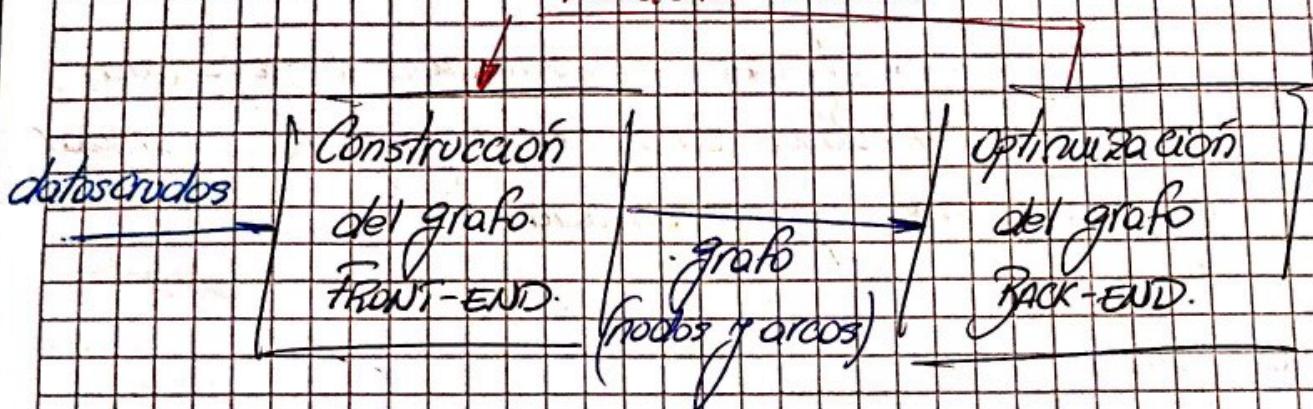
Estructura completa

•) Interacción entre 2 componentes: front-end y back-end.
La diferencia entre estos conceptos radica en que el front-end toma como input los datos crusos obtenidos a partir del sensor y la configuración actual del mapa. Lo que hace entonces es tomar la observación actual y tratar de relacionarla con otras observaciones que el robot ha visto hasta el momento. Esto es, la componente que se encarga de construir el grafo e intenta identificar las restricciones (arcos), es decir, donde se encuentra el robot actualmente relativo a una pose anterior. Entonces el front-end se encarga de encontrar estos arcos y formular el grafo. Una vez formulado el grafo se pasa al back-end, el cual es un bloque de optimización el cual busca optimizar el grafo, es decir, intenta encontrar nuevas configuraciones de

nodos para que el error, introducido por las restricciones, sea comunicado. Luego el back-end reporta la posición de los nodos al front-end formando así un lazo.

→ El front-end intenta encontrar restricciones y el back-end utiliza dichas restricciones para construir el cuadro y dicho cuadro es reportado al front-end y así sucesivamente.

Posiciones de nodos



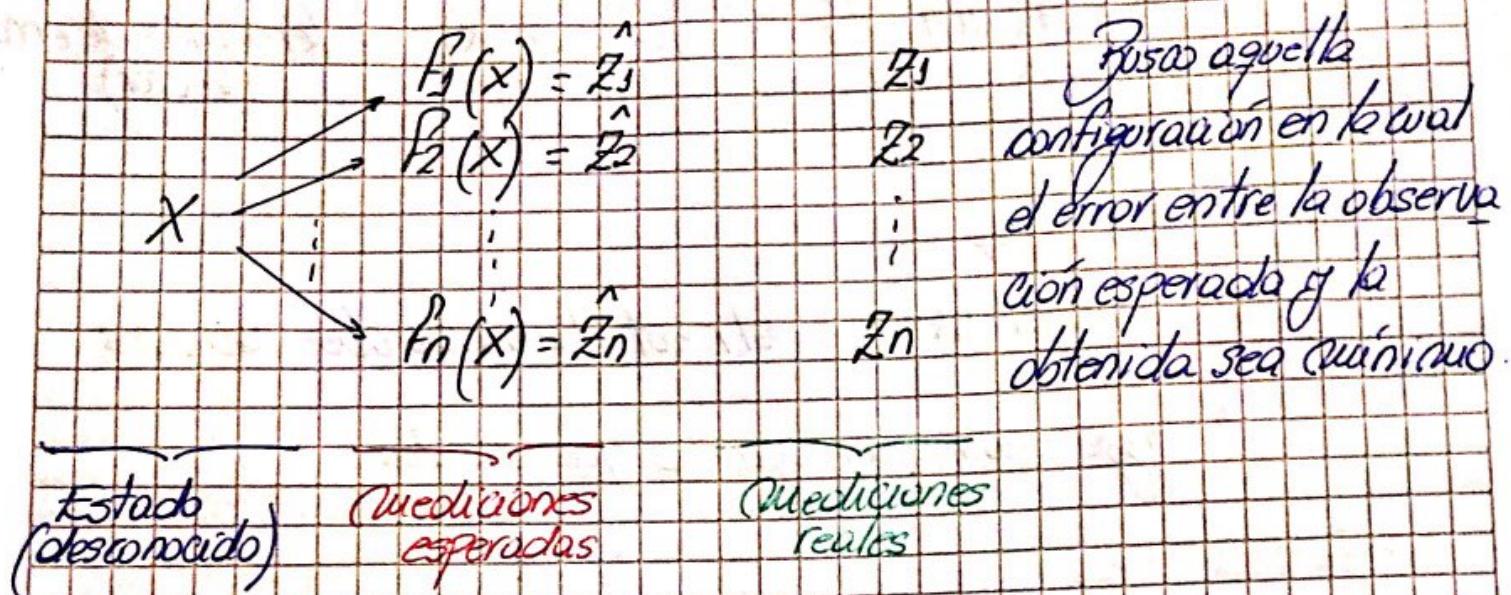
Cuadrados mínimos

-) Método para calcular soluciones para sistemas sobre determinados, es decir, aquellas sistemas en las cuales tenemos más ecuaciones que incógnitas.
-) Minimiza la suma de los errores al cuadrado del sistema.

Dado un sistema descripto por un conjunto de n funciones de observación $\{f_i(x)\}_{i=1:n}$ y sean:

-) X el vector de estados
-) Z la medición del estado X
-) $\hat{Z}_i = f_i(x)$ la función que mapea X a la medición esperada \hat{Z}_i

Todas n mediciones ruidosas $Z_1:n$ del estado X se busca estimar el estado X que mejor explica dichas mediciones $Z_1:n$.



función de ERROR:

El error e_i es la diferencia entre la medición esperada y la medida.

vector: $e_i(x) = Z_i - P_i(x)$

Asumimos que todas las observaciones son independientes y que el error tiene media cero y se encuentra distribuido normalmente. Luego se trata de un error gaussiano con matriz de información S_i . El error cuadrado de una medición depende sólo del estado y es un escalar:

$$e_i(x) = e_i(x)^T S_i e_i(x)$$

OBJETIVO: encontrar el estado x^* que minimiza el error dadas las mediciones:

$$x^* = \underset{x}{\operatorname{arg\,min}} \hat{f}(x) \rightarrow \text{error global (escalar)}$$

$$x^* = \underset{x}{\operatorname{arg\,min}} \sum_i e_i(x) \rightarrow \text{terminos cuadrados (escalar)}$$

$$x^* = \underset{x}{\operatorname{arg\,min}} \sum_i e_i^T(x) \cdot D_i \cdot e_i(x) \rightarrow \text{terminos de error (vector)}$$

→ Aplicado a SLAM:

Busco estimar las poses del robot dadas las observaciones

↳ Tengo nuevas observaciones que estoy dando

TÉCNICAS DE MAPPING EN 3D

Posición

- .) Los mapas 2D se han usado exitosamente para tareas tales como navegación y localización.
- .) Los robots se mueven en un mundo 3D.
- .) La evasión robusta de obstáculos y el planeamiento de trayectorias requieren modelos en 3D.

Representaciones

Nubes de puntos

Pro:

- .) No discretizan los datos
- .) El área a mapear no está limitada

Contra:

- .) Uso de memoria no acotado
- .) No hay representación directa de espacio libre o desocupado.

Malla de Vóxels 3D

Pro:

-) Representación volumétrica
-) Tiempo de acceso constante
-) Actualización probabilística

Contra:

-) Requerimiento de memoria: el mapa completo debe alojarse en memoria.

-) El tamaño del mapa debe saberse.
-) Errores de discretización

Mapa de elevación (Máps 2.5D)

-) Grilla 2D que almacena un altura (elevación) estimada para cada celda
-) En general la incertezza aumenta con la distancia (medida).
-) Kriguan para estimar la elevación.

Pro:

-) Eficiente en cuanto a memoria
-) Tiempo de acceso constante
-) Estimación estadística de la altura
-) Representación 2.5D

Contra:

-) No hay distinción entre espacio libre y desconocido.
-) No contempla superficies verticales.
-) Solo un nivel es representado.

Mapa de elevación extendido

•) Identificar:

- / Celdas que corresponden a estructuras verticales
- / Celdas que contienen espacios vacíos

- .) Chequear si la varianza de la altura de todos los mediciones de una celda es alta.
- .) Chequear si el espacio libre es mayor que la altura del robot.
("gap cell").

Mapas multi-nivel (MHS)

- .) Cada celda 2D almacena varios paneles con valores

✓ la media de la altura μ

✓ la varianza de la altura σ^2

✓ Un valor de profundidad d

⇒ Un panel puede no tener profundidad (objetos planos: ej: piso)

⇒ Una celda puede tener muchos paneles (espacios verticales, ej: puentes).

Pro:

- .) Puede representar múltiples superficies por celda

Contra:

- .) No representa áreas desconocidas

.) No es una representación volumétrica sino una discretización en la dimensión vertical

- .) La localización rugosa
MHS no es fácil de implementar

OCTREES

- .) Estructura de datos basada en árboles
- .) Subdivisión recursiva del espacio en octantes
- .) Asignación de volumen a medida que se necesita
- .) Grilla "3D inteligente"

- Pro:
- .) Modelo completo 3D
 - .) Probabilístico
 - .) Resolución variable.
 - .) Eficiente en memoria

- Contra:
- .) La implementación es más compleja (memoria, actualizaciones, etc.).

Ocrojaps

- .) Basado en octrees.
- .) Representación probabilística y volumétrica de ocupación incluyendo espacios desconocidos:
 - ✓ Ocupación modelada como un filtro de Bayes binario recursivo.
 - ✓ Actualización eficiente usando notación log-odds.
 - ✓ Reporte de valores (clamping) asegura actualización.
- .) Soporta multi-resolución
- .) Eficiente en memoria
 - ✓ Archivo de mapa más compacto.
- .) Comprensión de mapa sin pérdidas.

Algoritmo ICP: Iterative Closest Point.

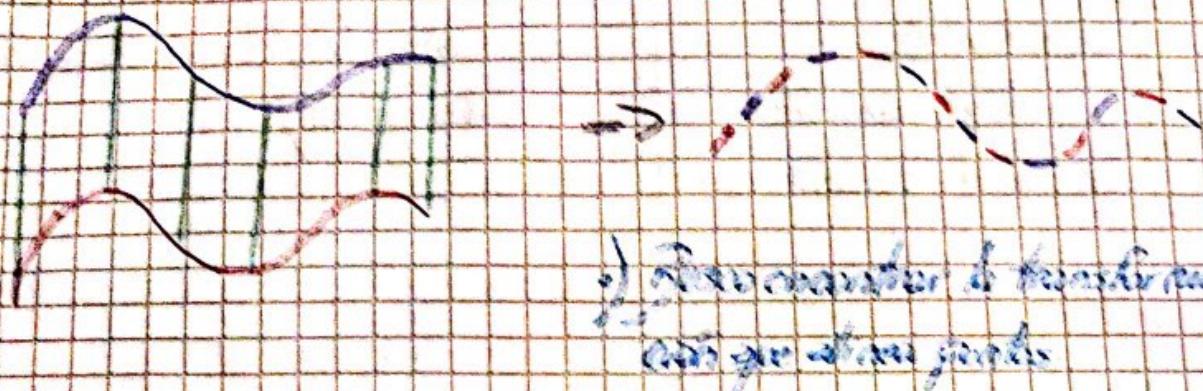
Dados dos conjuntos de puntos que se corresponden:

$$X = \{x_1, \dots, x_N\}$$

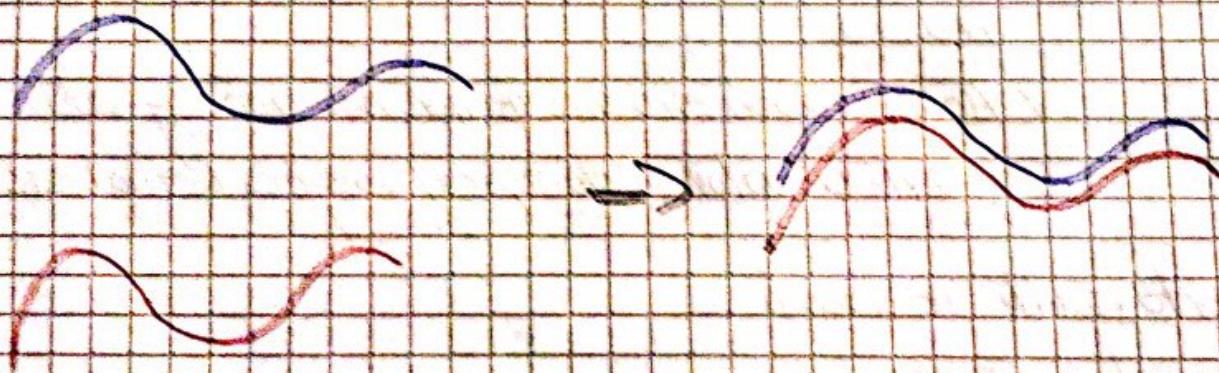
$$P = \{p_1, \dots, p_M\}$$

El objetivo es encontrar la traslación T y rotación R que minimiza la suma de los errores al cuadrado.

Si las correspondencias correctas son conocidas, la rotación y traslación correcta pueden ser estimadas en líneas cerradas



Si las correspondencias correctas no son conocidas, el procedimiento es imposible de terminar si la rotación y traslación relativa optima es un solo paso.



Método

- Idea: iterar para encontrar alineamientos
- Paseada de puntos cercanos
- Converge si las posiciones iniciales están lo suficiente "cerca"

Tareas:

- 1) Determinar los puntos que se corresponden
- 2) Calcular rotación R y traslación t por descomposición en valores singulares

- 3) Aplicar R y t a los puntos del conjuntos a ser registrados
- 4) Calcular el error $E(R, t)$
- 5) Si el error decrece y $E > \text{umbral}$: repetir pasos anteriores
SINO: Finalizar y devolver la alineación final.

VARIANZAS DE ICP

- 1) Subconjunto de puntos (de uno o ambos conjuntos de puntos).

Selección puntos de entrada

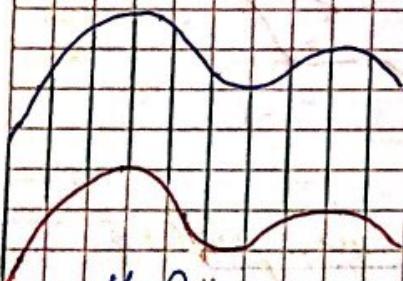
✓ Usar todos los puntos

✓ Submuestreo uniforme

✓ Muestreo aleatorio

✓ Muestreo basado en características (Features) #

✓ Muestreo de espacio normal



Uniforme.

- ④ • Encontrar puntos importantes
- Disminuye número de correspondencias a encontrar
- más eficiente y más preciso
- requiere pre-procesamiento.

- 2) Peso de correspondencias.

✓ Seleccionar subconjunto de puntos de cada conjunto

✓ Tochar los puntos seleccionados de ambos conjuntos

✓ Pesar los pares correspondientes

✓ Determinar la transformación que minimiza la función de error

3) Asociación de datos

•) Tiene mayor incidencia en convergencia y velocidad

•) Métodos de muestreo:

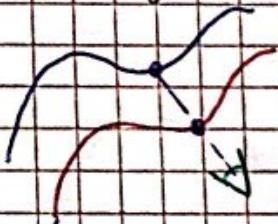
✓ Punto más cercano: estable pero convergencia lenta
y requiere pre-procesamiento.

✓ Normal shooting: converge mejor para estructuras simples
(proyectar en la dirección normal) pero peor para estructuras complejas

✓ Punto compatible más cercano:

Cocompatibilidad basada en: normales, colores, curvaturas,
derivadas de orden mayor
o otras características locales

✓ Basado en proyección: Proyectar puntos según el punto
de vista



4) Rechazo de ciertos puntos (outliers).

•) Puntos correspondientes con una distancia punto a punto
muy mayor que cierto umbral

•) Rechazo de pares que no son consistentes con los pares vecinos

Algoritmo resultante con variantes

- 1) Muestrear puntos
- 2) Determinar los puntos que se corresponden
- 3) Pesar/rechazar pares
- 4) Calcular R y t por DVS.
- 5) Aplicar R y t a todos los puntos
- 6) Calcular $E(R, t)$.
- 7) Si $E(R, t) >$ umbral \rightarrow Repetir.
 $E(R, t) <$ umbral \rightarrow Finalizar y devolver resultado de la alineación.

PUNTOS

-) ICP es un buen algoritmo para calcular desplazamientos entre escaneos.
-) El mayor problema es determinar la asociación de datos
-) La convergencia depende de los puntos (muchos/dos)
-) ICP no siempre converge.

PLANEAMIENTO DE TRAYECTORIAS.

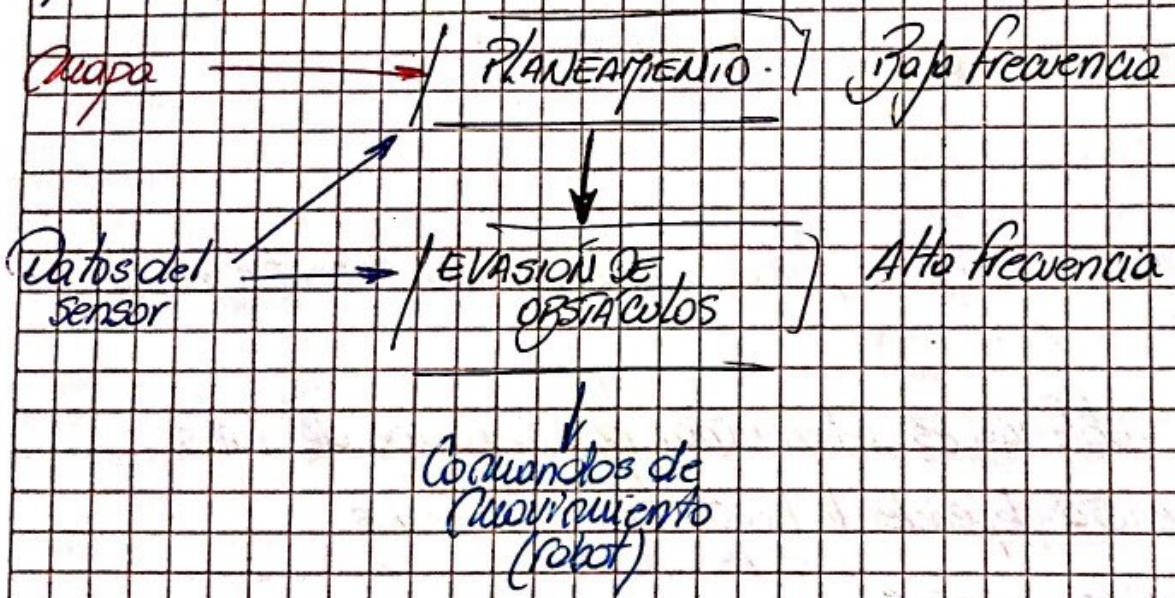
OBJETIVOS

-) Trajetorias sin colisiones
-) El robot debe llegar a destino lo mas rapido posible.

⇒ Busco calcular el camino optimo tomando en cuenta las incertezas en las acciones

⇒ Debo generar acciones rapidamente en caso de encontrar objetos inesperados

ARQUITECTURA DE DOS CAPAS



MÉTODO DE VENTANA Dinámica (DWA)

Es un método para la evasión de obstáculos en donde se busca determinar trayectorias libres de colisiones usando operaciones geométricas.

⇒ Se usan comandos de movimiento (τ, ν).

- .) restringido al espacio (x, y)
- .) No en el espacio de velocidades

↓

Los comandos se eligen según una función heurística de navegación, la cual trata de minimizar el tiempo de viaje "llegando rápido en la dirección correcta".

Funciónde
navegación

$$NF = \alpha \text{ Inf} + \beta \text{ nf} + \gamma \text{ Ainf} + \delta \text{ goal} \leftarrow \text{Cercanía al objetivo}$$

(minimizar la velocidad considera el costo de alcanzar el objetivo)

Es un método con:

- .) reacción rápida
- .) bajos requerimientos computacionales
- .) produce trayectorias libres de colisiones
- .) las trayectorias resultantes son en general sub-optimas.
- .) Puede no llegarse al objetivo debido a minimos locales

Limitación: DWA tiene problemas para llegar al objetivo.

PROBLEMA DE PLANEAMIENTO: Formulación

Dados:

- .) Una pose inicial del robot
- .) Una pose objetivo deseada
- .) Una descripción geométrica del robot
- .) Una representación geométrica del entorno

→ Busco encontrar un camino que lleva gradualmente al robot desde la pose inicial hasta la pose objetivo deseada sin tocar ningún obstáculo.

Espacio de Configuración

Las configuraciones del robot q es una especificación de las posiciones de todos los puntos del robot relativos a un sistema de coordenadas fijo.

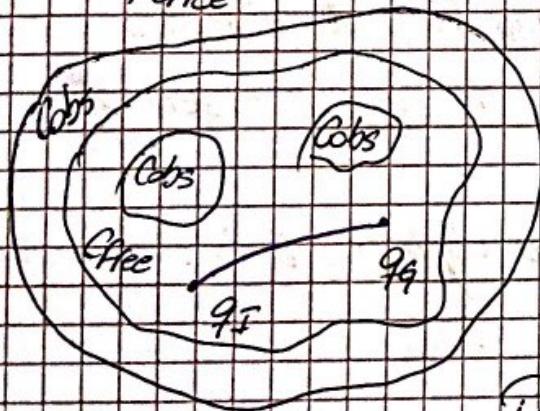
Siendo $W = \mathbb{R}^n$ el espacio de trabajo, $O \in W$ el conjunto de obstáculos, $A(q)$ el robot en configuración $q \in C$.

$$C_{\text{free}} = \{q \in C / A(q) \cap O = \emptyset\}$$

$$C_{\text{obs}} = C \setminus C_{\text{free}}$$

q_I = configuración inicial

q_F = configuración final



Planeamiento implica entonces encontrar un camino continuo:

$$\tau: [0, 1] \rightarrow C_{\text{free}}$$

$$\text{con } \tau(0) = q_I; \tau(1) = q_F$$

Todo esto se hace un planeamiento con el robot como un punto en el espacio C .

(el espacio C debe discretizarse).

Búsqueda

El problema de búsqueda es encontrar una secuencia de acciones (un camino) que lleva al estado deseado (objetivo)

No-informada: no hay información más allá de la definición del problema.

Variantes dadas por como se expanden los nodos

Información adicional: información adicional a través de heurísticas
Capacidad de decir si un nodo es mejor candidato que otro (Algoritmo: A*)

La performance de los algoritmos se mide a través de:

- ✓) Completitud: encuentra una solución si existe?
- ✓) Optimidad: la solución es la mejor en términos de costos?
- ✓) Complejidad temporal: cuánto tiempo para llegar a la solución?
- ✓) " de memoria: cuánto se requiere?

PLANEAMIENTO 5-D

Es una alternativa a la estructura de dos capas donde se plantea en el espacio de configuración completo (x, j, θ, r, w) usando A* considerando restricciones kinemáticas del robot.

La idea es buscar en el espacio de búsqueda discretizado (x, j, θ, r, w) , pero esto tiene el problema de que dicho espacio es demasiado grande para ser explorado dentro de las restricciones de tiempo. Busco entonces restringir el espacio de estados.

Algoritmo:

- 1) Actualizar el mapa de grilla según los datos del sensor
 - ✓) Grilla de ocupación 2-D
 - ✓) Convolución (filtra el mapa).
 - ✓) Se agregan obstáculos detectados
 - ✓) Liberar celdas no ocupadas

2) Usar A* para encontrar la trayectoria en el espacio (x, y) usando el mapa de grilla actualizado

✓) A* encuentra el camino más corto (de menor costo).

3) Determinar un espacio de configuración restringido en 5D basado en el paso 2.

✓) La proyección del camino en 5D sobre el espacio (x, y) está cerca del óptimo en 2D

\Rightarrow busco construir un espacio de búsqueda restringido basado en el camino 2D.

4) Encontrar la trayectoria planeando en el espacio restringido $(x, y, \theta, \dot{x}, \dot{y})$

✓) Usar A* en el espacio restringido en 5D para encontrar la secuencia de comandos para llegar al objetivo intermedio

5-D vs DWA

✓) DWA muchas veces tiene problemas para entrar en lugares estrechos

✓) El método 5D resulta en movimientos mucho más rápidos al pasar por lugares estrechos

✓) 5D es más cercano al óptimo.