

Robots móviles

SLAM (Mapeado y
Localización
Simultáneos)

Índice

- Introducción al problema del SLAM
- SLAM basado en EKF
- SLAM con filtros de partículas

Mapeado y localización dependen el uno del otro

- Para localizarse hace falta tener un mapa
- Para construir un mapa necesitamos saber dónde estamos en cada momento

SLAM (Simultaneous Localization And Mapping)

Tenemos que resolver los dos problemas **simultáneamente**

- El mapa (aunque sea parcial) nos ayudará a localizarnos
- La localización "provisional" nos permite seguir construyendo el mapa

Aplicaciones del SLAM

En muchos casos el usuario no tiene un mapa en formato adecuado o sería muy tedioso crear un mapa teleoperando al robot

Roomba 980 Robot Vacuum Cleans a Whole Level of Your ...



Speaker notes

El Roomba 980 no es el único robot de limpieza que usa SLAM, por ejemplo el LG Hombot crea un mapa de características visuales del techo con una cámara que apunta hacia arriba <https://www.youtube.com/watch?v=UANWyiDf3hA>

Más aplicaciones del SLAM

Vehículos autónomos

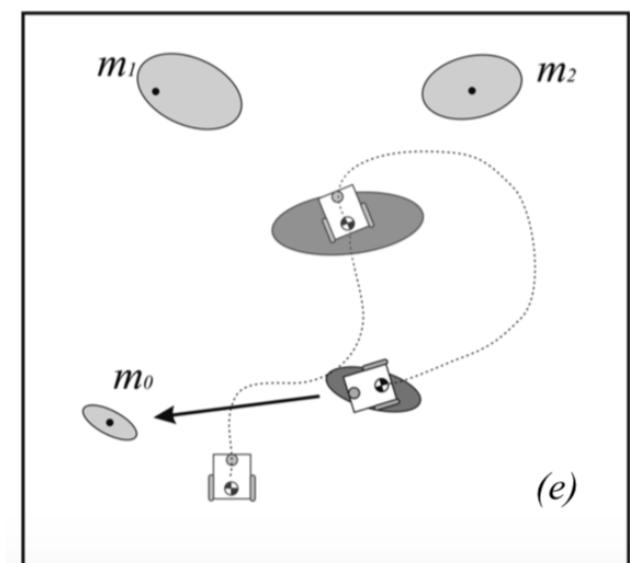
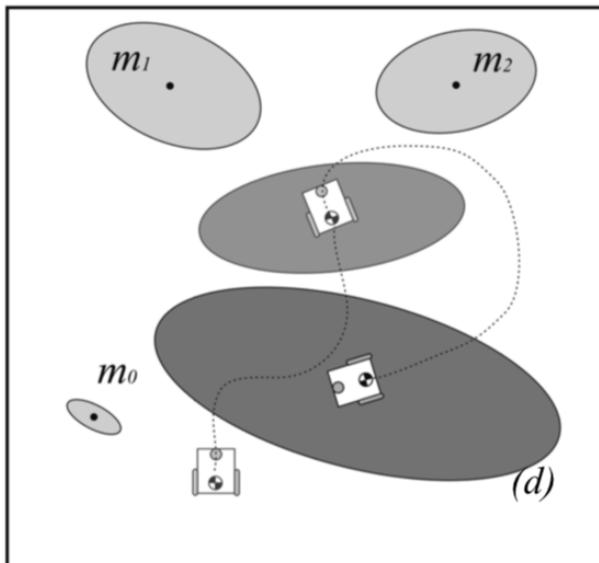
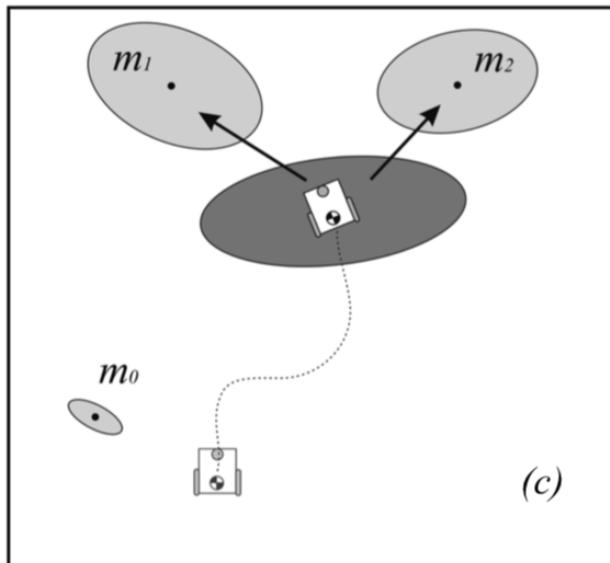
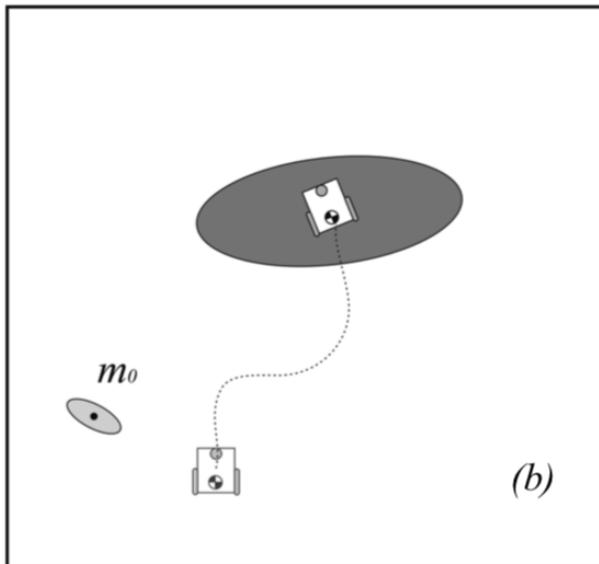
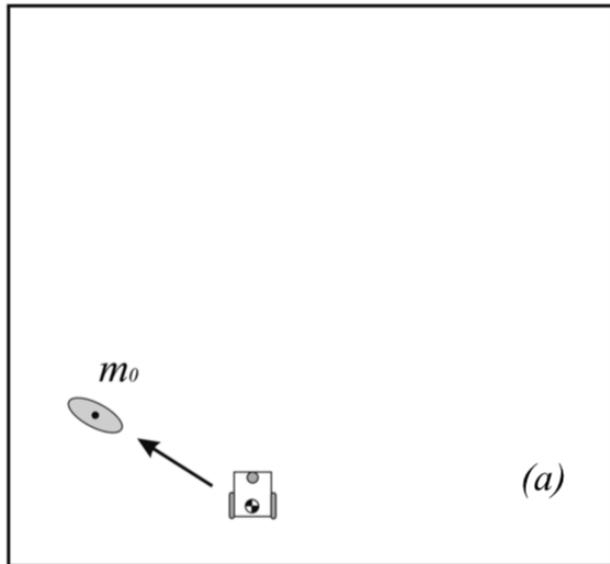
MapperSampleVideo.mp4



Visual Inertial State Estimation at 20m/s on ...



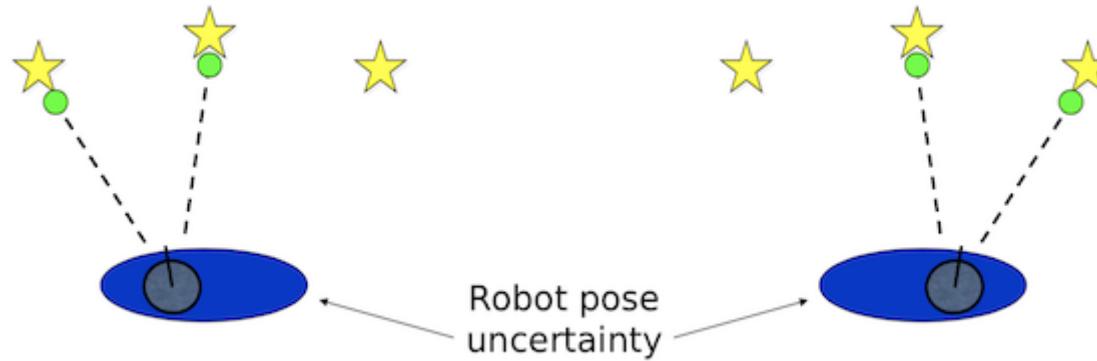
La incertidumbre en el SLAM



Speaker notes

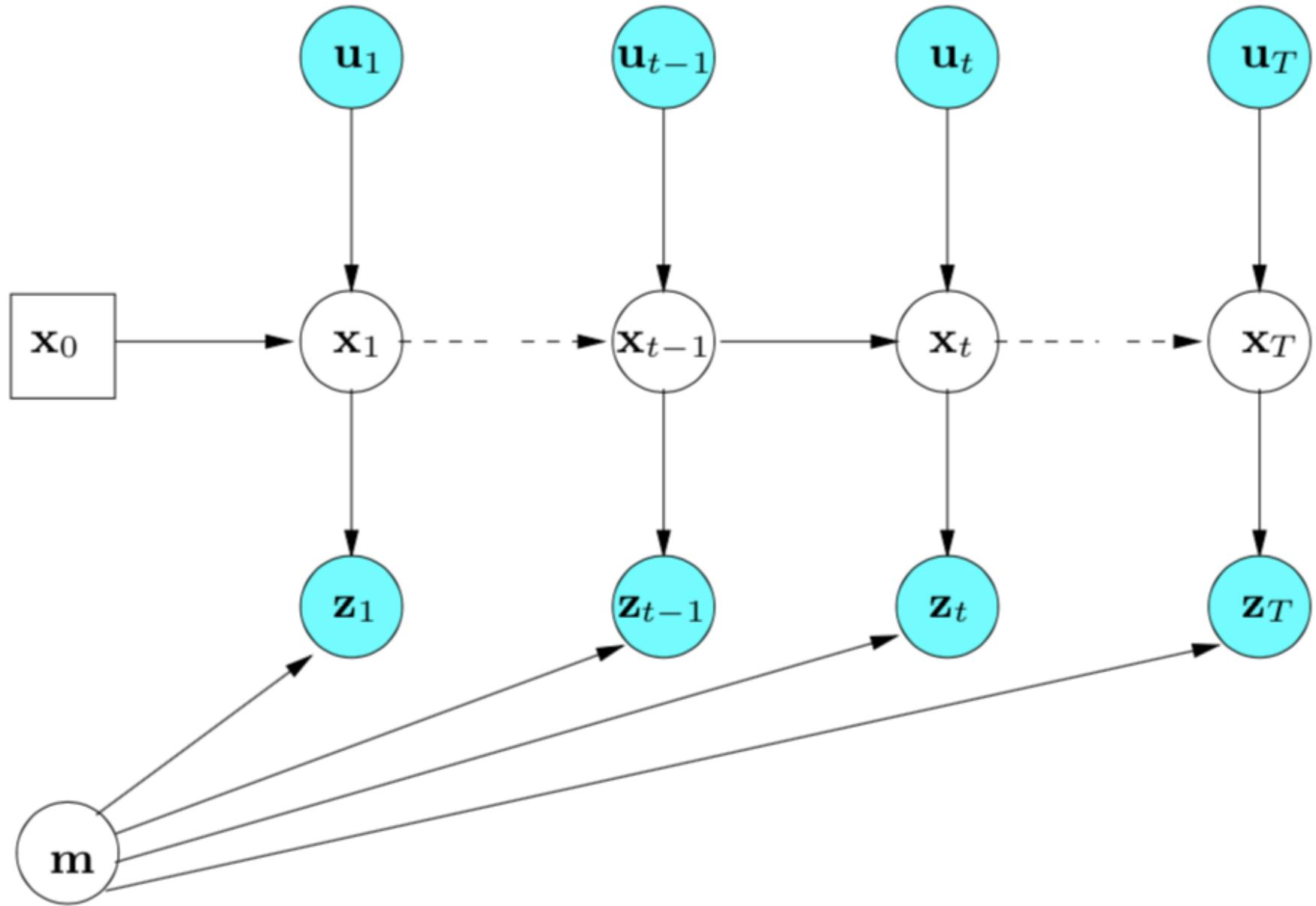
- (a) Al comienzo, la incertidumbre en la posición del robot es 0. Desde aquí el robot observa un *landmark* cuya posición tendrá una incertidumbre asociada al modelo de error del sensor
- (b) Conforme el robot se va moviendo, la incertidumbre en su posición crece según el modelo de error de la odometría.
- (c) Cuando el robot observa nuevos *landmarks* la incertidumbre de su posición es mayor que en (a) ya que es una combinación de la incertidumbre en la localización más la del sensor. Es decir, *la incertidumbre del mapa y del robot están correlacionadas*.
- (e) Cuando el robot observa *landmarks* con una posición más precisa (por ejemplo porque los ha observado al comienzo), la incertidumbre en su propia localización *decrece*. Además el mapa entero se actualiza y la incertidumbre de todos los *landmarks* vistos previamente también decrece. Esto se conoce en SLAM como **cerrar el ciclo** (*closing the loop*)

El problema de la asociación de datos



- En general la asociación entre observaciones y landmarks es desconocida, a causa de la incertidumbre en la posición y las medidas
- Escoger la asociación equivocada puede tener consecuencias catastróficas

SLAM como una red bayesiana



- x_1, \dots, x_t son las posiciones del robot
- m es el mapa, típicamente formado por *landmarks*
- z_1, \dots, z_t son las detecciones de *landmarks* hechas por los sensores
- u_1, \dots, u_t son los *movimientos* hechos por el robot

Dos variantes del problema:

- **Online SLAM**: solo nos interesa la posición actual del robot (también llamado *filtering*)

$$p(x_t, m | z_{1:t}, u_{1:t})$$

- **Full SLAM**: nos interesa toda la trayectoria (también llamado *smoothing*)

$$p(x_{1:t}, m | z_{1:t}, u_{1:t})$$

Algunos algoritmos

- *Online SLAM* (estimar (x_t, m))
 - EKF SLAM
 - Filtros de partículas "Rao-Blackwellizados" (explícitamente solo mantienen x_t)
- *Full SLAM* (estimar $(x_{1:t}, m)$)
 - Graph SLAM
 - Filtros de partículas "Rao-Blackwellizados" (podemos recuperar $x_{1:t-1}$)

Algoritmos para SLAM

EKF SLAM

El mismo algoritmo EKF que usábamos para localización, pero ahora en el estado tendremos, además de la **posición** del robot, también la de los **landmarks**

$$\mu_t = (x_t, m), \Sigma_t = \begin{bmatrix} \Sigma_{RR} & \Sigma_{RM} \\ \Sigma_{MR} & \Sigma_{MM} \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} x \\ y \\ \theta \\ m_{1,x} \\ m_{1,y} \\ \vdots \\ m_{n,x} \\ m_{n,y} \end{bmatrix}}_{\mu} \quad \underbrace{\begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \dots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \dots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \dots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\ \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \dots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\ \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \dots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} & \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \dots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\ \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta} & \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \dots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} \end{bmatrix}}_{\Sigma}$$

Speaker notes

Dimensionalidad: En el caso de un mapa bidimensional con N landmarks puntuales, tendremos una dimensión de 3 + 2N (x, y, θ) robot + (x, y) para cada landmark

La matriz de covarianza se divide en 4 partes:

- la parte amarilla representa la covarianza en las posiciones del robot, sin tener en cuenta el mapa
- la Azul la de los landmarks
- las partes verdes representan la correlación entre la posición del robot y los landmarks

Recordad el EKF para localización

Extended_Kalman_Filter(μ_{t-1} , Σ_{t-1} , u_t , z_t):

1. **Predicción**

$$2. \bar{\mu}_t = g(u_t, \mu_{t-1})$$

$$3. \bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + Q_t$$

4. **Corrección**

$$5. K_t = \frac{\bar{\Sigma}_t H_t^T}{(H_t \bar{\Sigma}_t H_t^T)}$$

$$6. \mu_t = \bar{\mu}_t + K_t (Z_t - h(\bar{\mu}_t))$$

$$7. \Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$$

Donde H y G son los *jacobianos*

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t} \quad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$

Las fórmulas son prácticamente iguales, cambia que **tenemos**
además la parte de los *landmarks*

Predicción

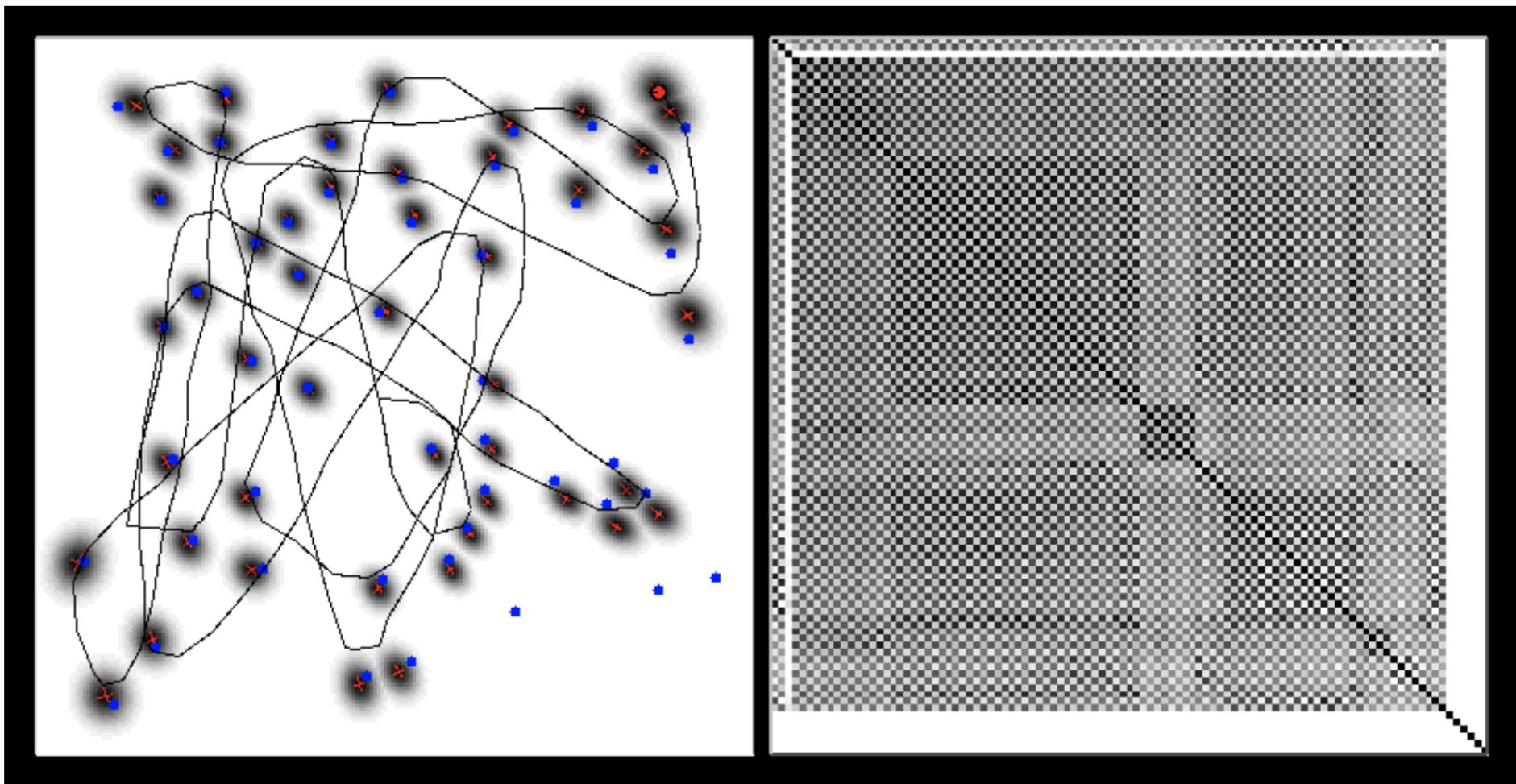
Solo se actualiza la parte en la que intervienen las coordenadas del robot (ya que se supone el mapa estático).

Siendo ingeniosos, la actualización se puede implementar con coste *lineal* con el número de *landmarks*.

$$\begin{bmatrix}
 x \\
 y \\
 \theta \\
 m_{1,x} \\
 m_{1,y} \\
 \vdots \\
 m_{n,x} \\
 m_{n,y}
 \end{bmatrix}_{\mu} \quad \begin{bmatrix}
 \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\
 \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\
 \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \\
 \hline
 \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} \\
 \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} \\
 \vdots & \vdots & \vdots \\
 \hline
 \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} \\
 \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta}
 \end{bmatrix}_{\Sigma} \quad \begin{bmatrix}
 \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \dots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\
 \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \dots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\
 \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \dots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\
 \hline
 \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \dots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\
 \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \dots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\
 \vdots & \vdots & \ddots & \vdots & \vdots \\
 \hline
 \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \dots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\
 \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \dots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}}
 \end{bmatrix}_{\Sigma}$$

Corrección

- La ganancia \mathbf{K} implica **toda la matriz de covarianza**, y eso hace que cambien todas las correlaciones
- Es decir, el EKF **mantiene de forma explícita las correlaciones entre la posición del robot y la de los landmarks** (y también de los landmarks entre sí)
- El coste es $O(n^2)$



Cierre del ciclo

como se mantienen explícitamente las correlaciones, al **cerrar el ciclo** se actualizan todas "automáticamente", reduciendo la incertidumbre de todos los *landmarks*.

[Matlab] Conventional EKF SLAM Loop Closing



Demo de EKF

Código en Octave/Matlab disponible en
https://github.com/ottocol/openslam_bailey-slam (cambiado para
que funcione en Octave, **original de Tim Bailey**, http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm)

Victoria Park

Un experimento clásico (http://www-personal.acfr.usyd.edu.au/nebot/victoria_park.htm)



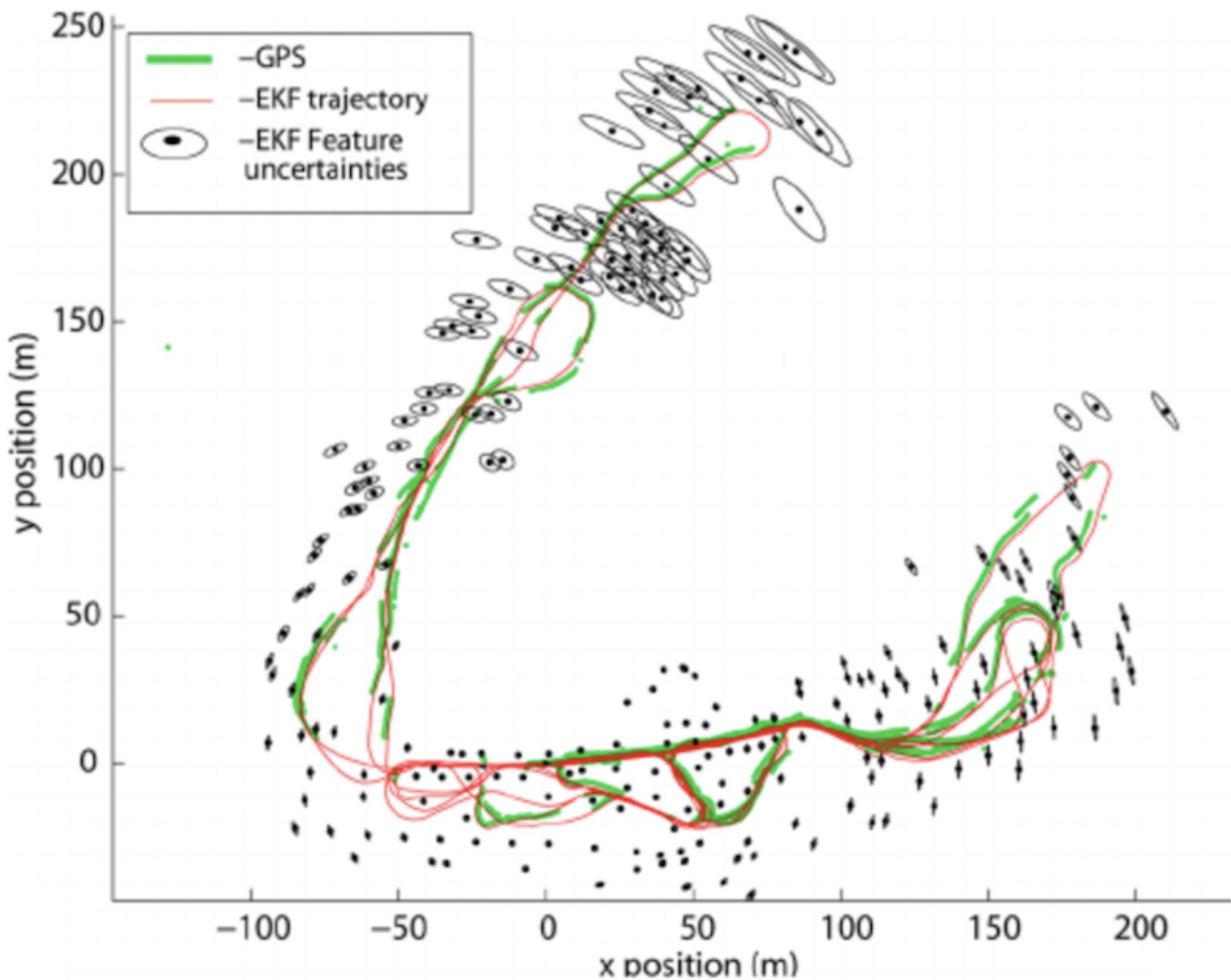
- El vehículo usa *dead reckoning* (*encoders* en las ruedas, *IMUs*) + laser + GPS para *ground truth*
- *Landmarks*: los troncos de los árboles, detectados con el laser

Speaker notes

La explicación detallada de cómo se llevó a cabo el experimento y los algoritmos probados está en http://www-personal.acfr.usyd.edu.au/nebot/publications/slam/IJRR_slam.htm

Resultado del EKF en Victoria Park

El resultado es razonablemente preciso, y el algoritmo es aplicable porque no hay un gran número de *landmarks*



Resumiendo: limitaciones del EKF SLAM

El coste es cuadrático en el número de **landmarks**, lo que implica que no es posible aplicarlo en mapas que tengan más de unos cientos de éstos.

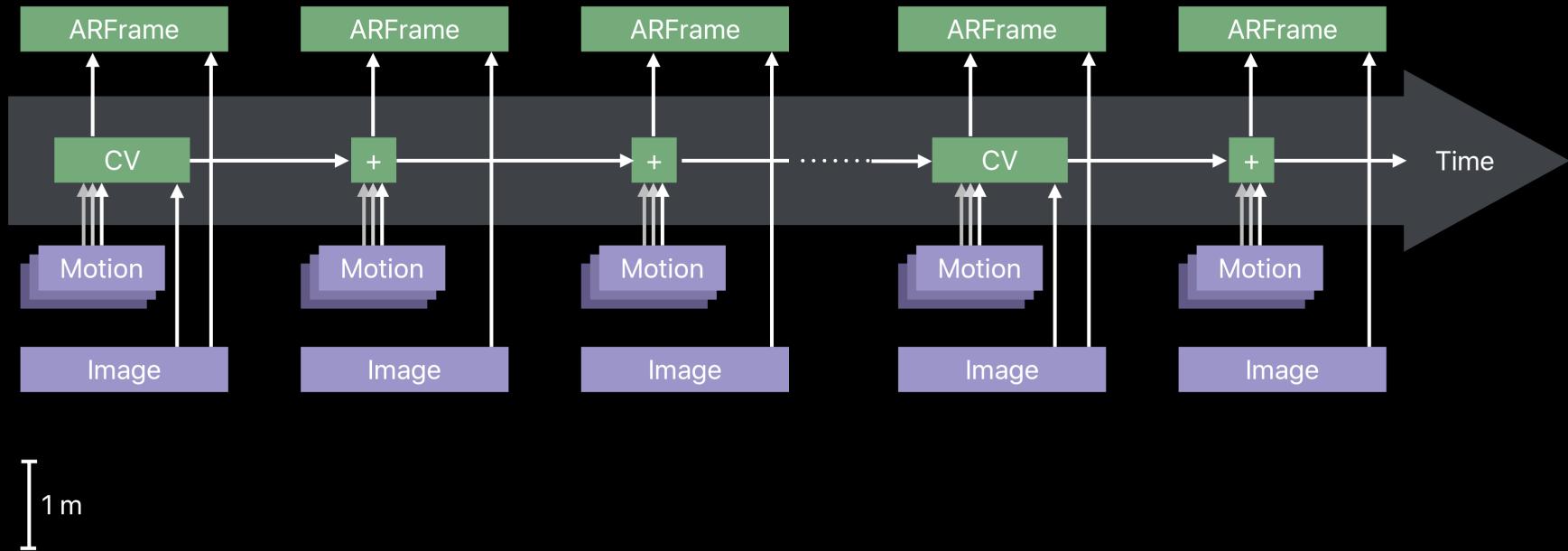
Aplicabilidad del EKF SLAM

Apropiado para aplicaciones de **realidad aumentada** (AR) donde el rango de movimiento y por tanto el mapa es pequeño.

<https://developer.apple.com/videos/play/wwdc2018/610/>

Behind the Scenes—Visual Inertial Odometry

Motion data and computer vision

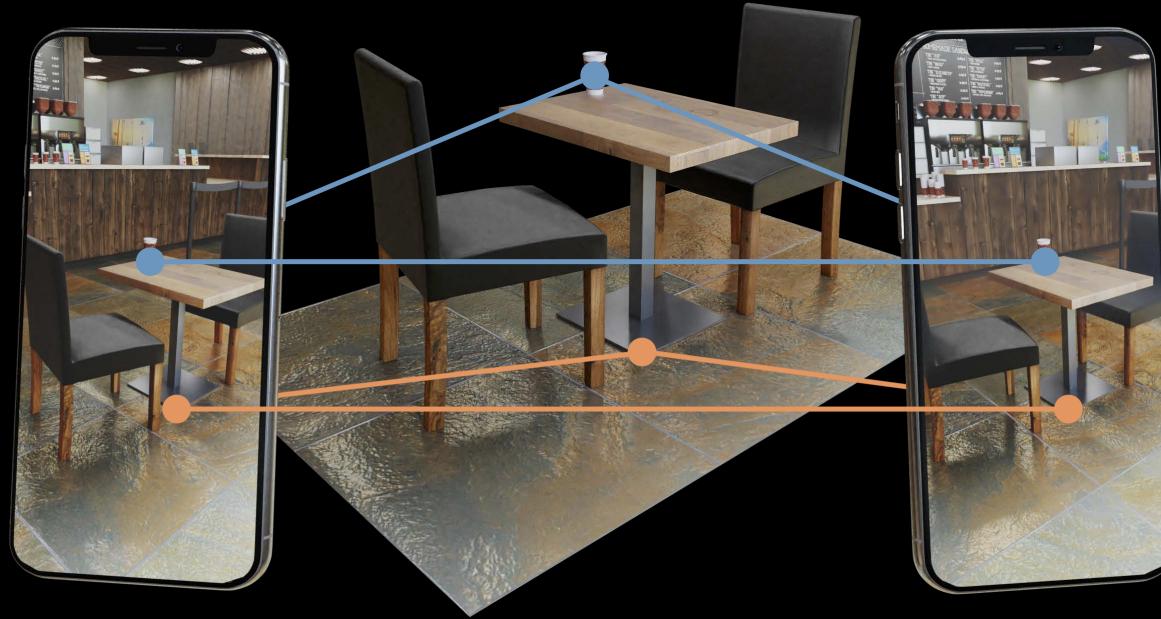


ARKit de Apple, el equivalente de Google es ARCore

Aplicabilidad del EKF SLAM (II)

Behind the Scenes—Visual Inertial Odometry

Triangulation creates World Map



keypoint matching + triangulación = nube de puntos dispersa

Algoritmos para SLAM

**SLAM con filtros de
partículas**

Recordando filtros de partículas para localización

- Cada partícula es una hipótesis de pose (x, y, θ) del robot
- **Sampling**: del modelo de movimiento (*predicción*)
- **Importance sampling** (*corrección*)
 - Asignar peso con el modelo del sensor
 - Remuestrear basándose en ese peso

¿Podríamos aplicar la misma idea a SLAM?

Por desgracia ¡la dimensionalidad del problema del SLAM es mucho mayor!. **De 3 dimensiones pasamos fácilmente a cientos** (más exactamente a $2N+3$, con N =número de *landmarks*)

Necesitaríamos un número enorme de partículas para cubrir adecuadamente un espacio tan grande

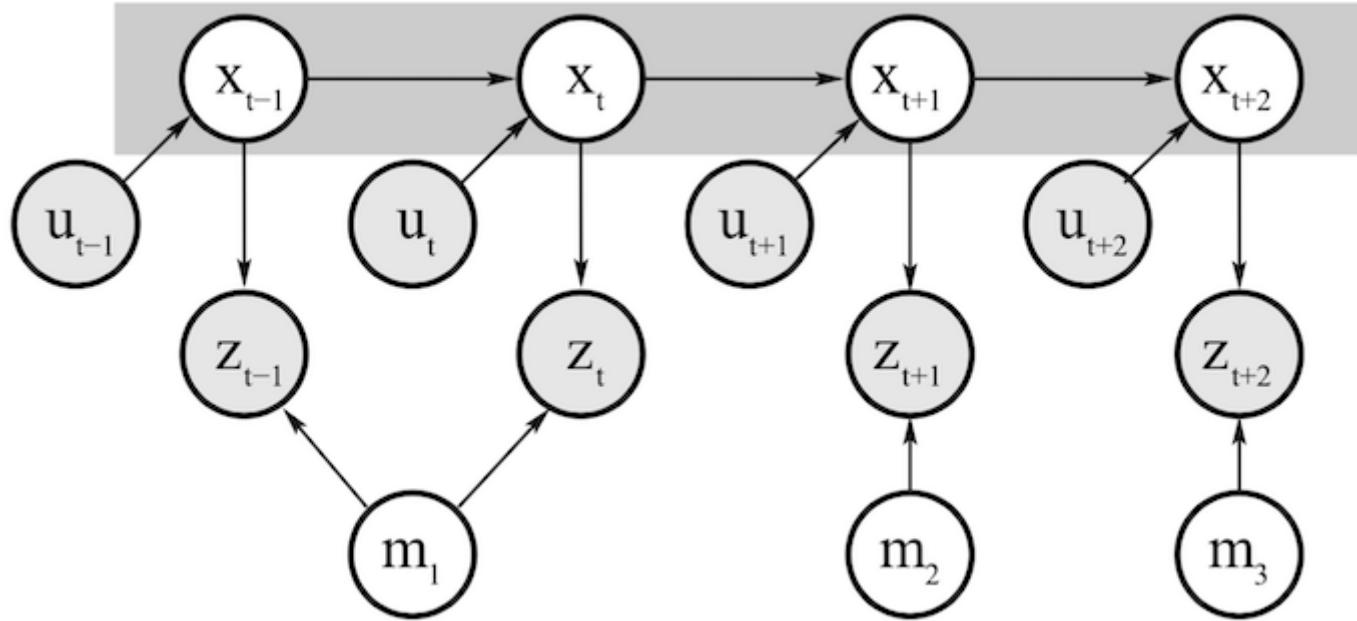
Vamos a intentar aplicar algún "truco" para **"reducir" la dimensionalidad**. La clave va a estar en separar el problema en varias partes (cada una con mucho menos dimensiones)

Factorización del SLAM

Por ciertas propiedades básicas de la probabilidad condicional, podemos hacer

$$\begin{aligned} P(x_{1:t}, l_{1:m} | z_{1:t}, u_{0,t-1}) &= \\ P(x_{1:t} | z_{1:t}, u_{0,t-1})P(l_{1:m} | x_{1:t}, z_{1:t}) \end{aligned}$$

Sobre esto podemos aplicar un filtro de partículas "Rao-Blackwellizado": aplicar partículas sobre una parte del problema y un método analítico (por ejemplo un EKF) sobre la otra



Suponiendo conocidas las poses del robot, $x_{1:t}$, las posiciones de los *landmarks* son independientes entre sí (p.ej. tener más información sobre la posición de m_1 no nos da más datos sobre m_2). Formalmente esto se conoce como *d-separación* en redes bayesianas.

FastSLAM

Idea clave: podemos *reducir la dimensionalidad* del problema separando la estimación de las probabilidades de cada *landmark*

$$\begin{aligned} P(x_{1:t}, l_{1:m} | z_{1:t}, u_{0,t-1}) &= \\ P(x_{1:t} | z_{1:t}, u_{0,t-1}) P(l_{1:m} | x_{1:t}, z_{1:t}) &= \\ P(x_{1:t} | z_{1:t}, u_{0,t-1}) \prod_{i=1}^M P(l_i | x_{1:t}, z_{1:t}) \\ (\text{al ser los } l_i \text{ independientes}) \end{aligned}$$

$$p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^M p(l_i \mid x_{1:t}, z_{1:t})$$



Filtro de N partículas



M EKFs, uno por landmark

- Cada partícula representa una posible trayectoria del robot (explícitamente solo guardamos la última posición)
- Para cada partícula mantenemos M EKFs de tamaño 2x2 que se actualizan por separado

Particle #1	x, y, θ	Landmark 1	Landmark 2	...	Landmark M
Particle #2	x, y, θ	Landmark 1	Landmark 2	...	Landmark M
:					
Particle N	x, y, θ	Landmark 1	Landmark 2	...	Landmark M

Algoritmo FastSLAM

- Do the following M times:
 - **Retrieval.** Retrieve a pose $x_{t-1}^{[k]}$ from the particle set Y_{t-1} .
 - **Prediction.** Sample a new pose $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$.
 - **Measurement update.** For each observed feature z_t^i identify the correspondence j for the measurement z_t^i , and incorporate the measurement z_t^i into the corresponding EKF, by updating the mean $\mu_{j,t}^{[k]}$ and covariance $\Sigma_{j,t}^{[k]}$.
 - **Importance weight.** Calculate the importance weight $w^{[k]}$ for the new particle.
- **Resampling.** Sample, with replacement, M particles, where each particle is sampled with a probability proportional to $w^{[k]}$.

FastSLAM y "cerrar el ciclo"

- Ya vimos que el EKF mantiene **explícitamente** las correlaciones entre las poses de los *landmarks* y la del robot. Eso permite reducir la incertidumbre en todas ellas cuando se *cierra el ciclo*.
- En FastSLAM las correlaciones se mantienen a través de las partículas, que son distintas hipótesis sobre la trayectoria. Cuando se cierra el ciclo se reduce la incertidumbre descartando las que no "cuadran" con las medidas de los sensores.

FastSLAM Loop Closure

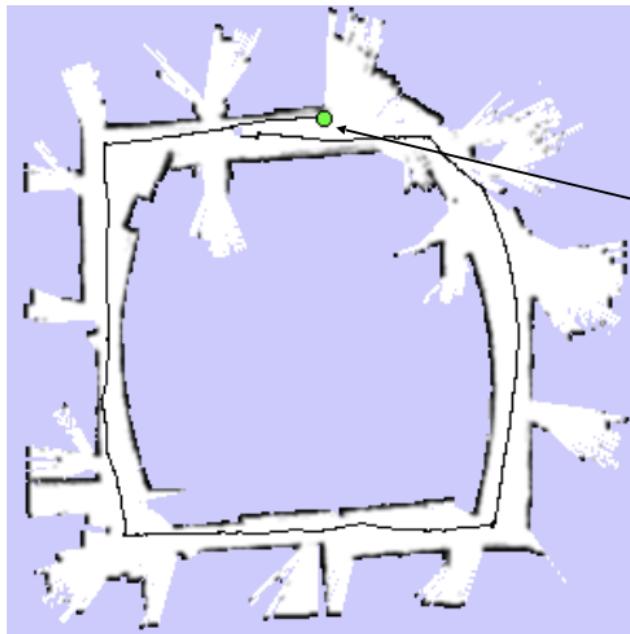


Problema: como solo podemos mantener un número limitado, se "perderán" muchas trayectorias posibles, si hay pocas partículas al final acaban teniendo todas una "historia común" en algún punto (no podemos cerrar ciclos muy largos).

FastSLAM para rejillas de ocupación

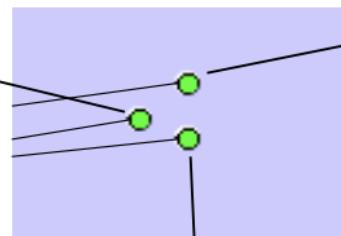
La extensión del algoritmo es bastante directa:

1. Cada **partícula** tiene un **mapa asociado** (una rejilla)

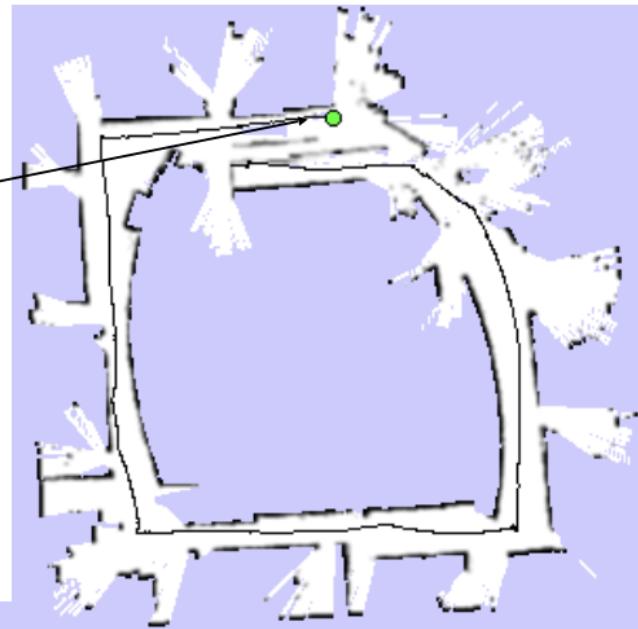


mapa de la partícula 1

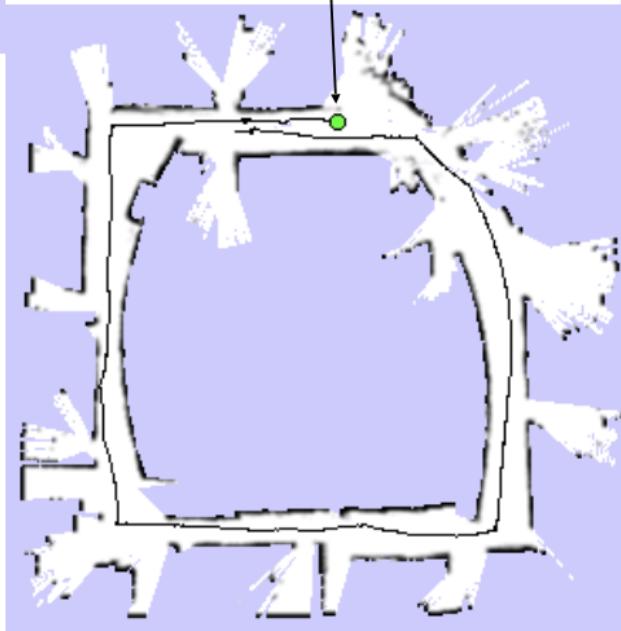
3 partículas



mapa de la partícula 2



mapa de la partícula 3



FastSLAM para rejillas de ocupación (II)

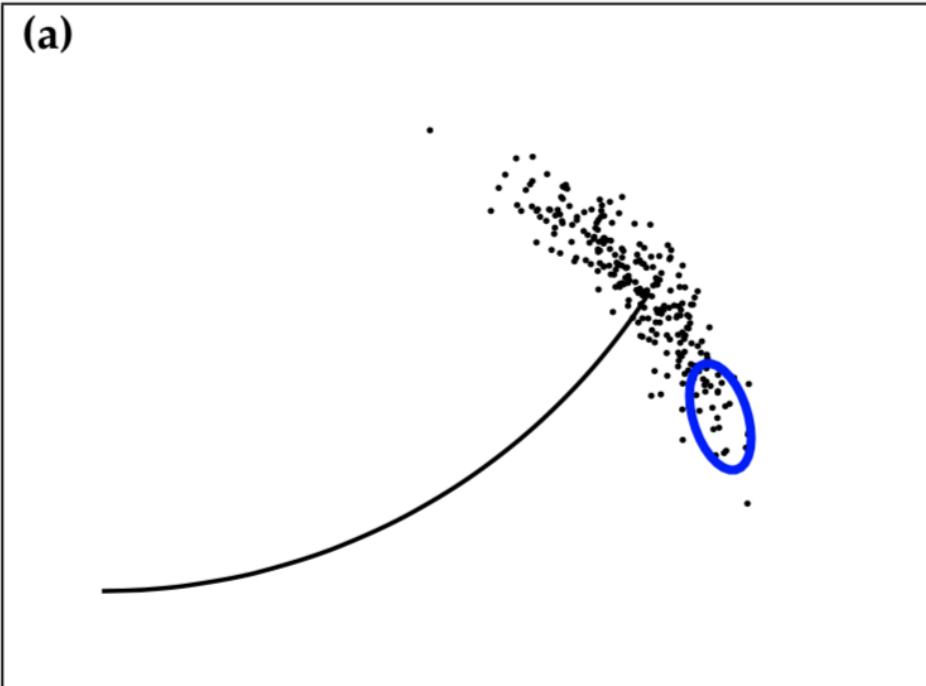
2. Como una **partícula** supone una **posición** del robot, para actualizar el mapa podemos aplicar el algoritmo de **mapeado con posición conocida** que ya visteis en el tema anterior

GMapping en ROS

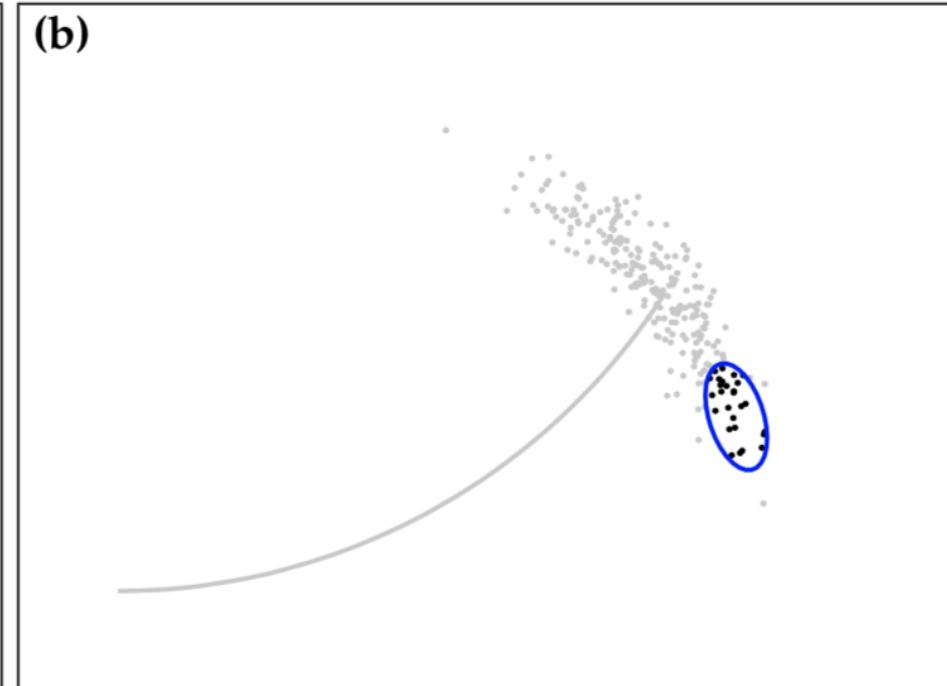
El algoritmo `gmapping` integrado en ROS es de tipo fastSLAM para rejillas de ocupación.

Mejora (incluída también en FastSLAM 2.0): en el *sampling* las muestras no se obtienen solo de la función de movimiento sino que tienen en cuenta las observaciones. Así nos quedamos solo con las que concuerdan con la situación actual.

(a)



(b)



Speaker notes

- El algoritmo de *gmapping* se describe en detalle en "Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard: Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters, IEEE Transactions on Robotics, Volume 23, pages 34-46, 2007" ([pdf](#))
- En "An Evaluation of 2D SLAM Techniques Available in Robot Operating System, 11th IEEE Int. Symp. on Safety, Security, and Rescue Robotics (SSRR 2013)" ([pdf](#)) podéis ver una comparación experimental entre diversos algoritmos de SLAM implementados en ROS.

Resultado ejemplo



- 30 particles
- 250x250m²
- 1.088 miles
(odometry)
- 20cm resolution
during scan
matching
- 30cm resolution in
final map

Bibliografía

- Capítulo 10. Probabilistic Robotics. MIT Press. Thrun, Burgard, Fox.
- Sección 5.8. Introduction to Autonomous Mobile robots. Roland Siegwart and Illah R. Nourbakhsh