

Robots Móviles

Tema 3. Localización y mapeado de un robot móvil
Parte II. Localización con filtro de partículas

Sergio Orts Escolano
Otto Colomina Pardo

Índice

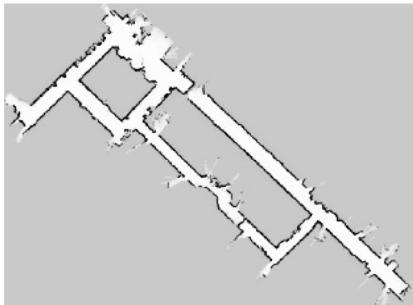
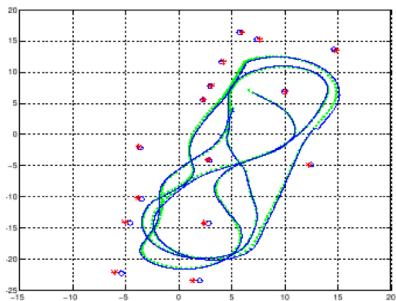
Localización
Filtro de partículas

Introducción

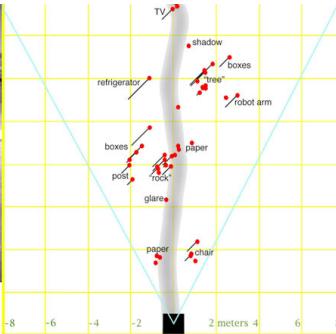
- Uno de los problemas fundamentales en robótica móvil, junto con la construcción de mapas
- Subproblemas
 - Local vs. global - kidnapped robot
 - Entorno: estático vs. dinámico
- Se nos pueden dar dos casos:
 - Partimos de una posición desconocida por el robot
 - Sabemos la posición inicial, pero el error acumulado es demasiado elevado

Necesitamos un mapa

- Necesitamos un mapa para poder comparar con él lo que perciben los sensores
 - Tipos de mapas más comunes:
 - Basados en características (landmarks)
 - Rejillas de ocupación

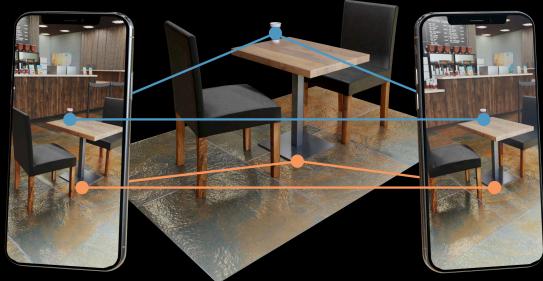


Mapas basados en características



Victoria Park, Sydney, un experimento clásico en localización y mapeado
http://www-personal.acfr.usyd.edu.au/nebot/victoria_park.htm

Behind the Scenes—Visual Inertial Odometry
Triangulation creates World Map

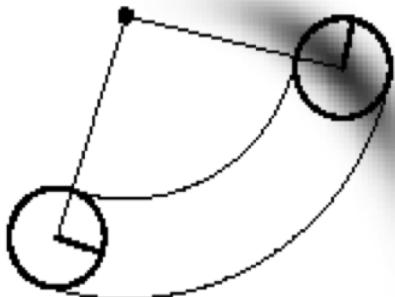


<https://developer.apple.com/videos/play/wwdc2018/610/>

Modelo de movimiento

- Nos dice la posición del robot en el instante t dada la posición en $t-1$ y el comando de control

$$P(x_t|x_{t-1}, u_t)$$

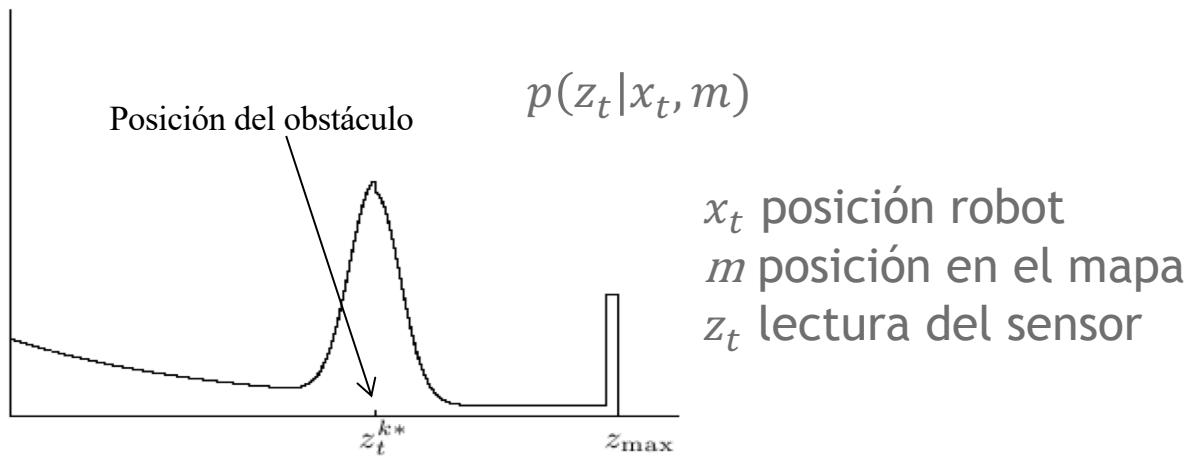


Como curiosidad: implementación en ROS (paquete AMCL)

https://github.com/ros-planning/navigation/blob/melodic-devel/amcl/src/amcl/sensors/amcl_odom.cpp

Modelo del sensor

- Modela el comportamiento del sensor: si sabemos dónde está el obstáculo, ¿cuál será la lectura devuelta por el sensor?



Como curiosidad: implementación en ROS (paquete AMCL)

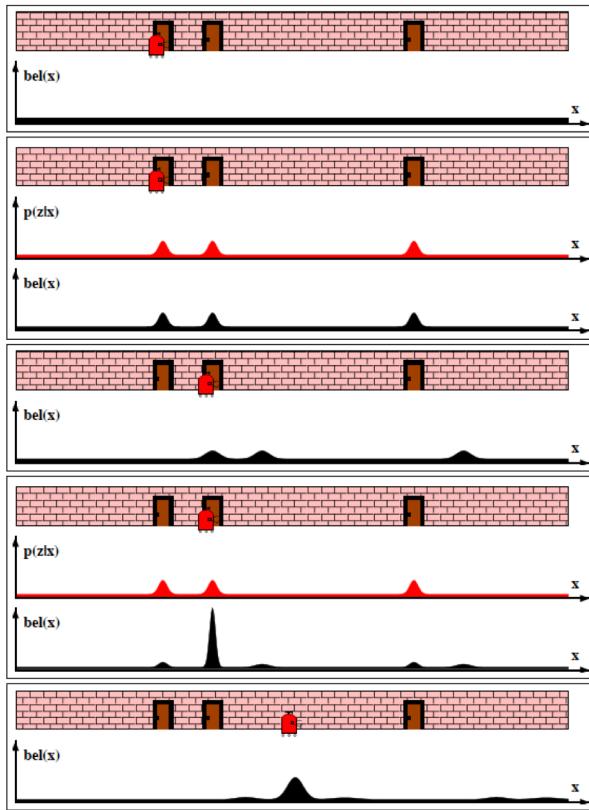
https://github.com/ros-planning/navigation/blob/melodic-devel/amcl/src/amcl/sensors/amcl_laser.cpp

Algoritmo de localización de Markov

- Aplicación directa del filtro de Bayes
- Transforma una creencia probabilística en el instante t-1 en creencia para el instante t
- $\text{bel}(x_0)$ representa el conocimiento inicial de la pose del robot
 - Si la pose inicial (x') es conocida: $\text{bel}(x_0) = 1$ si $x_0=x'$, 0 en otro caso
 - Si no, $\text{bel}(x_0) = 1 / |X|$ (distribución uniforme)

```
1:   Algorithm Markov_localization( $\text{bel}(x_{t-1})$ ,  $u_t$ ,  $z_t$ ,  $m$ ):  
2:     for all  $x_t$  do  
3:        $\overline{\text{bel}}(x_t) = \int p(x_t | u_t, x_{t-1}, m) \text{bel}(x_{t-1}) dx$   
4:        $\text{bel}(x_t) = \eta p(z_t | x_t, m) \overline{\text{bel}}(x_t)$   
5:     endfor  
6:     return  $\text{bel}(x_t)$ 
```

Markov: Ejemplo



Localización de Markov: propiedades

- La localización de Markov es independiente de la representación subyacente del espacio
- Existen numerosos algoritmos prácticos del método de Markov que utilizan diferentes representaciones:
 - Filtro de partículas
 - Histogramas
 - EKF (Extended Kalman Filter)

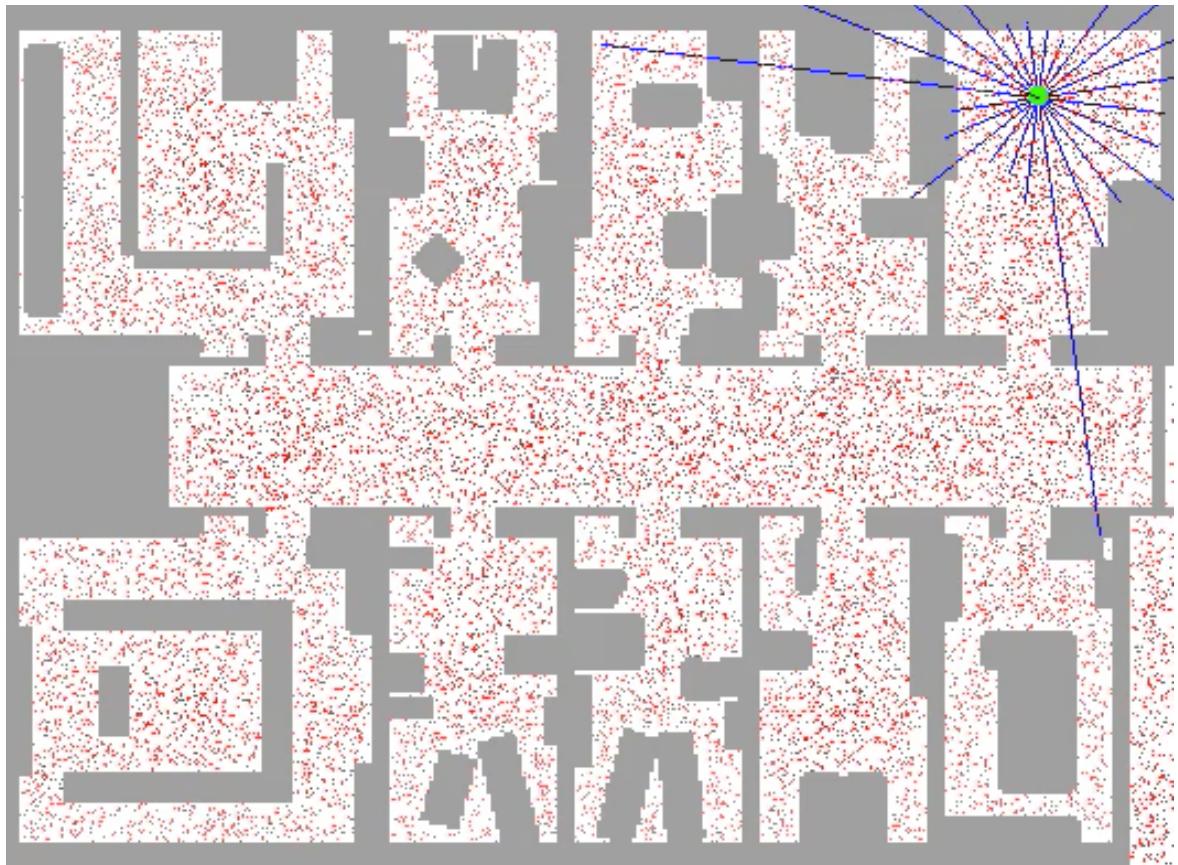
Índice

Localización
Filtro de partículas

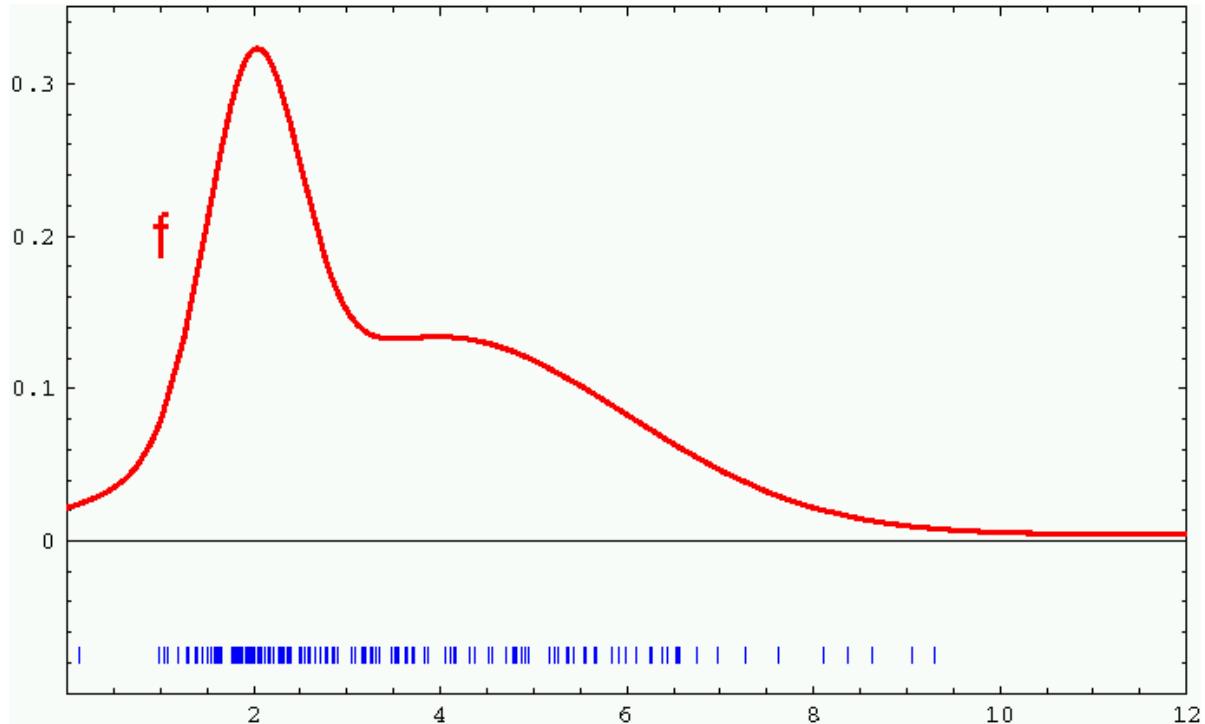
Filtros de partículas

- Mantiene un conjunto de hipótesis sobre la posición o “partículas”
- No calcula las probabilidades de una manera analítica, sino que las muestrea
- Supervivencia de las partículas que mejor se adaptan
- Orígenes
 - Condensation (inicialmente aplicado a visión)
 - Algoritmos tipo Monte Carlo

Ejemplo

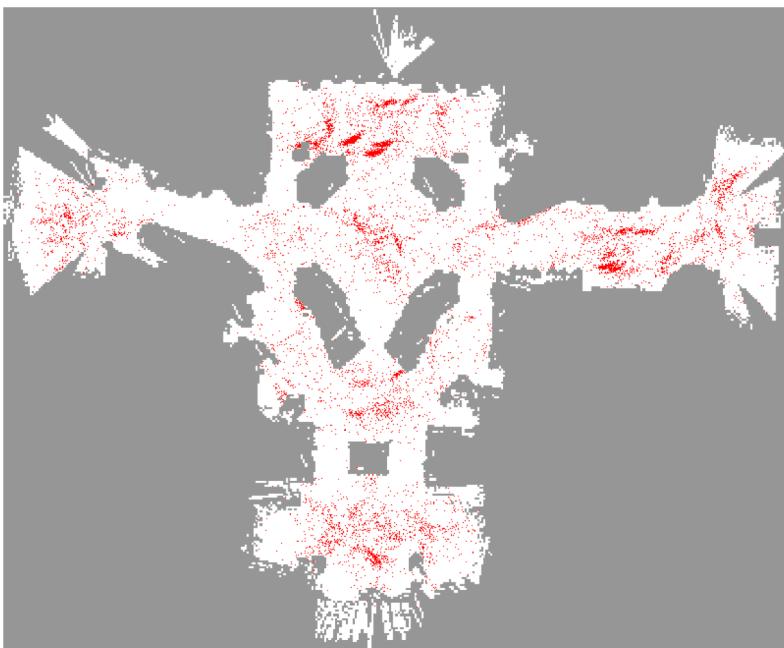


Representación de la densidad basada en muestras



Representación de la densidad basada en muestras (II)

- En nuestro caso cada muestra será una combinación de x , y , ángulo
- Al igual que en el caso 1D queremos concentrar las muestras en las zonas de alta probabilidad



Muestreando el modelo de movimiento

1. Algoritmo sample motion model

$$u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle, x = \langle x, y, \theta \rangle$$

$$\hat{\delta}_{rot1} = \delta_{rot1} + \text{sample}(\alpha_1 | \delta_{rot1} | + \alpha_2 \delta_{trans})$$

$$\hat{\delta}_{trans} = \delta_{trans} + \text{sample}(\alpha_3 | \delta_{trans} | + \alpha_4 (|\delta_{rot1}| + |\delta_{rot2}|))$$

$$\hat{\delta}_{rot2} = \delta_{rot2} + \text{sample}(\alpha_5 | \delta_{rot2} | + \alpha_6 \delta_{trans})$$

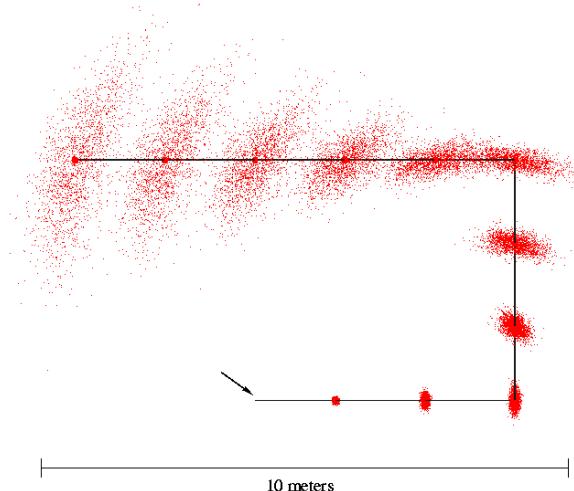
$$x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$$

$$y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$$

$$\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$$

sample normal distribution

8. devuelve $\langle x', y', \theta' \rangle$



Lo vamos a aplicar aquí

Algorithm Markov_localization($bel(x_{t-1})$, u_t , z_t , m):

for all x_t do

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

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

endfor

return $bel(x_t)$

Algoritmo (de momento, faltan cosas)

```
1:   Algorithm Markov_localization( $bel(x_{t-1})$ ,  $u_t$ ,  $z_t$ ,  $m$ ):  
2:     for all  $x_t$  do  
3:        $\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}, m) bel(x_{t-1}) dx$   
4:        $bel(x_t) = \eta p(z_t | x_t, m) \overline{bel}(x_t)$   
5:     endfor  
6:   return  $bel(x_t)$ 
```

Algorithm Particle_filter (X_{t-1} , u_t , z_t)

$$\hat{X}_t = X_t = \emptyset$$

for $m = 1$ to M do

sample $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$

... (falta la parte del sensor)

$$\hat{X}_t = \hat{X}_t + x_t^{[m]}$$

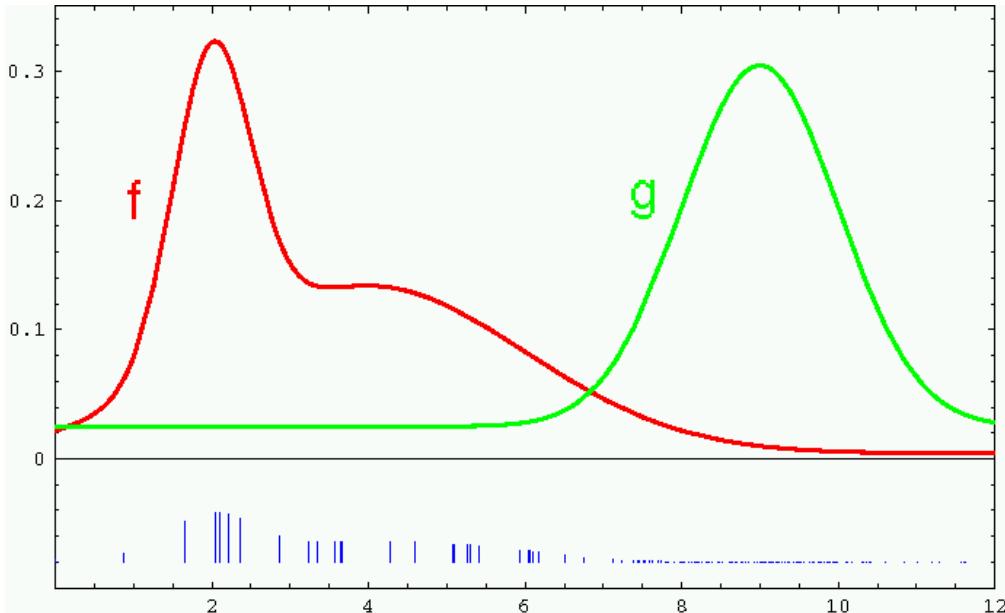
endfor

Problema

- Queremos muestrear $\text{bel}(x_t)$, pero no es una función gaussiana y no sabemos cómo muestreárla
- Necesitamos alguna técnica para **muestrear funciones arbitrarias**

Importance sampling

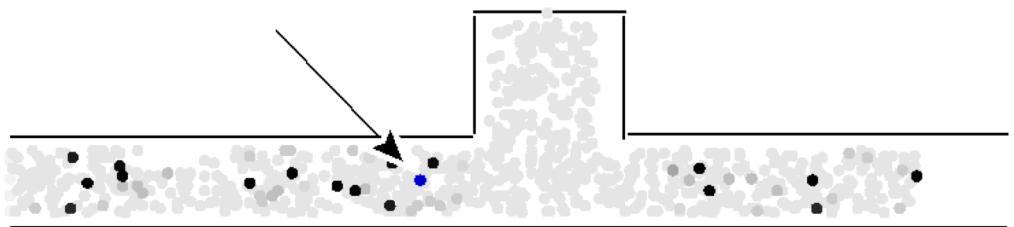
- Técnica para muestrear una función f suponiendo que sabemos muestrear una función g y conocemos el ratio f/g
- Muestreamos g y asignamos a cada muestra un valor de importancia igual a f/g



La importancia está relacionada con la probabilidad de que el robot se encuentre en esa posición, dadas las lecturas de los sensores

$$S = \{\langle s^{[i]}, w^{[i]} \rangle \mid i = 1, \dots, N\}$$

↑ ↑
State hypothesis Importance weight

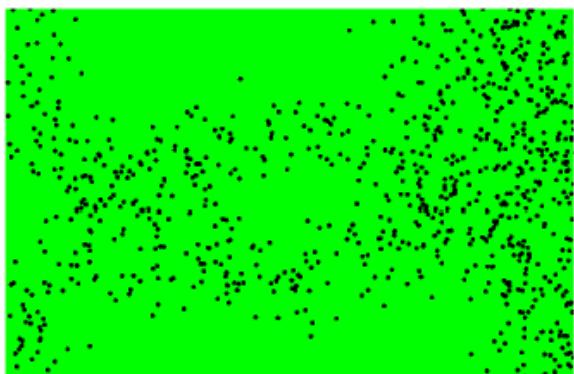


Resampling

- Objetivo: concentrar las muestras en las zonas de alta probabilidad de f
- Partiendo de las M muestras actuales, elegimos M muestras al azar, con probabilidad proporcional a la importancia
 - Se elige *con reemplazo*, es decir, la misma muestra se puede seleccionar varias veces
 - Como resultado, de las muestras de mayor importancia habrá varias “copias”, y las de menor se perderán
- Este paso es una especie de implementación probabilística de la “*supervivencia de los más aptos*”

Importancia del proceso de resampling

- Podríamos implementar el filtro de partículas sin resampling, solo manteniendo la importancia de cada partícula y actualizándola en cada paso, pero
 - Tenemos un número finito de partículas
 - Estaríamos malgastando recursos computacionales en las zonas de baja densidad y no explorando lo suficiente las de alta



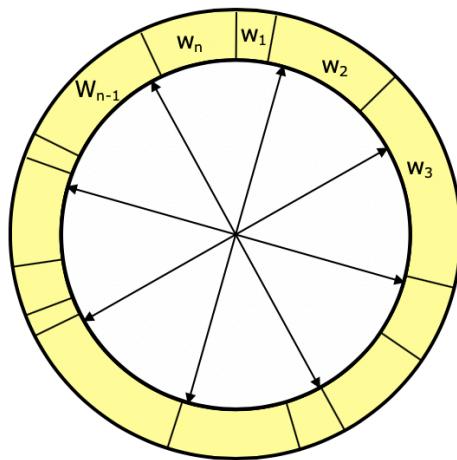
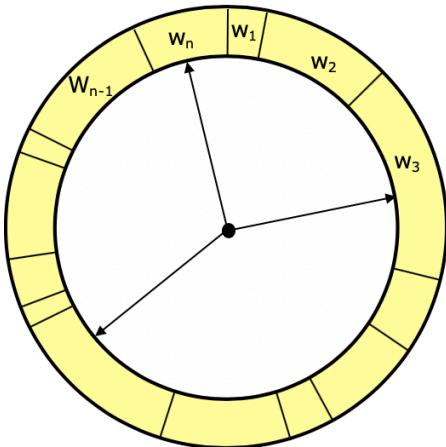
Muestras con pesos



Supervivencia muestras con mayor probabilidad

Implementación del resampling

– Alternativas



- Sup. $\sum w_i = 1$
- Generamos n valores al azar entre $[0, 1]$
- Seleccionamos la muestra donde “caiga” cada valor
- Se conoce como “selección por ruleta”

- Generamos 1 valor inicial al azar
- Vamos sumándole una cantidad fija

Algoritmo completo

- Tenemos M partículas. En cada instante de tiempo:
- Para cada partícula:
 - Aplicar modelo de movimiento
 - Calcular “peso” de la partícula con el modelo del sensor
- remuestreo
 - Seleccionar M partículas con probabilidad proporcional al peso

Algoritmo

Algorithm Particle_filter (X_{t-1}, u_t, z_t)

$$\hat{X}_t = X_t = \emptyset$$

for $m = 1$ to M do

$$\text{sample } x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$$

$$w_t^{[m]} = p(z_t | x_t^{[m]})$$

$$\hat{X}_t = \hat{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$$

endfor

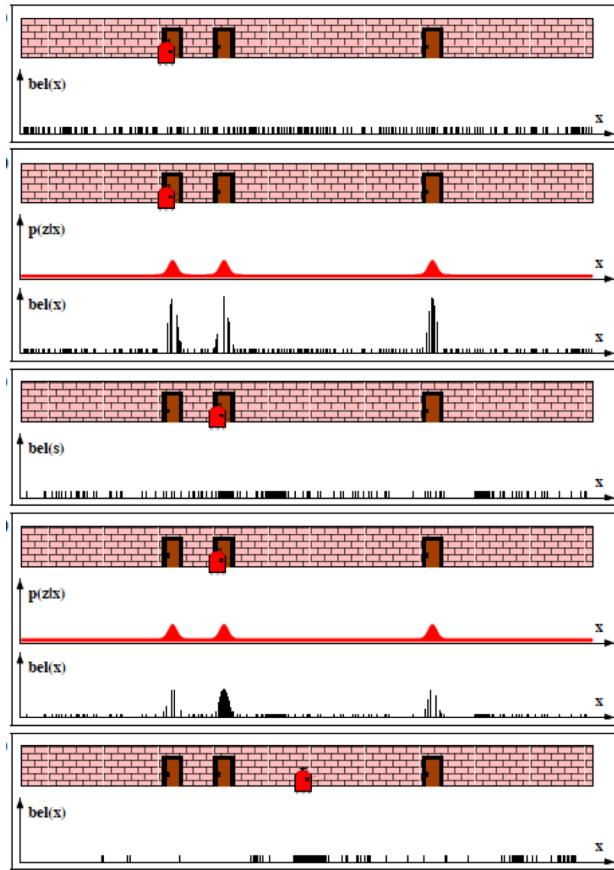
for $m = 1$ to M do (*importance sampling*)

draw i with probability $\propto w_t^{[m]}$

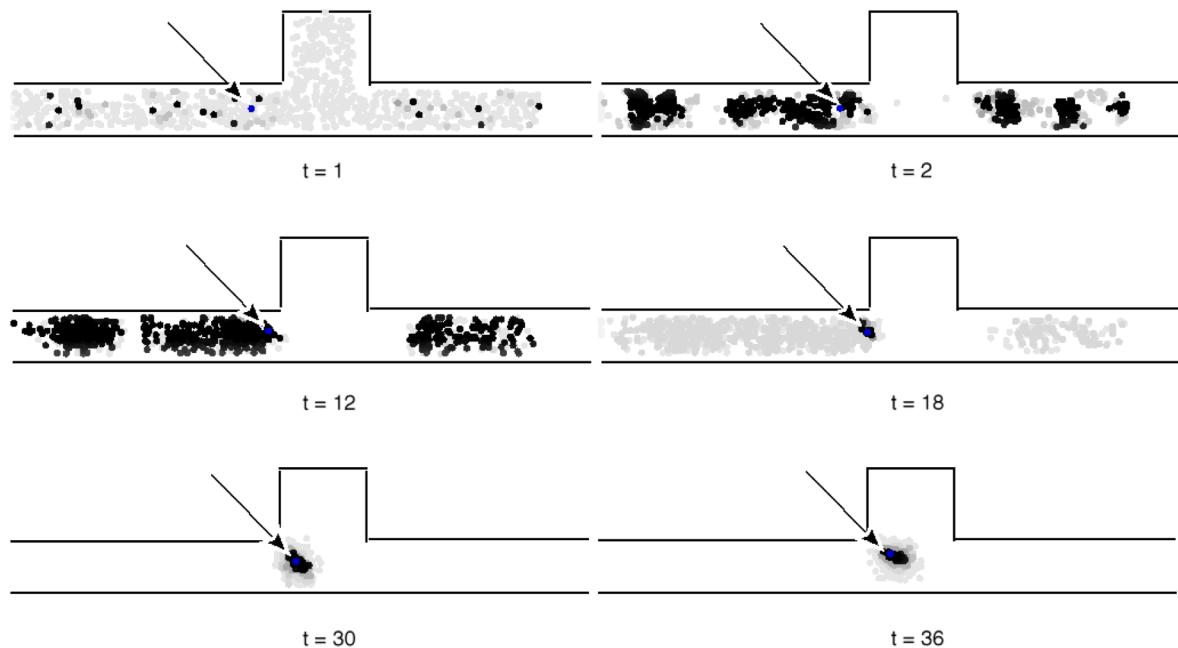
add $x_t^{[i]}$ to X_t .

endfor

Ejemplo unidimensional

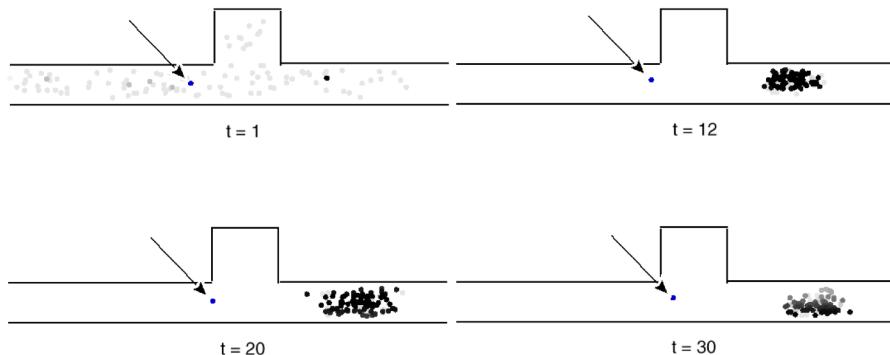


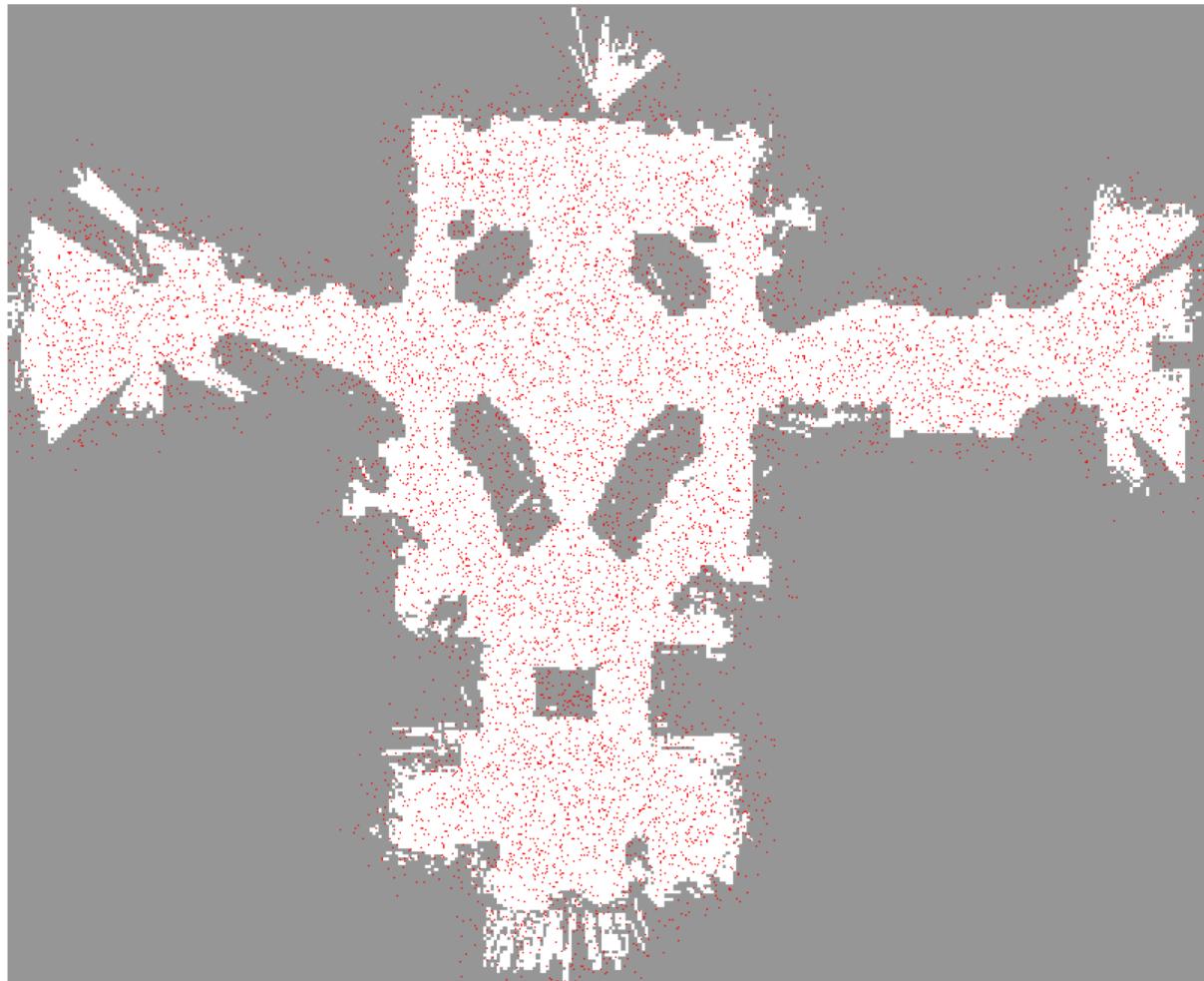
Ejemplo $\langle x, y, \Theta \rangle$

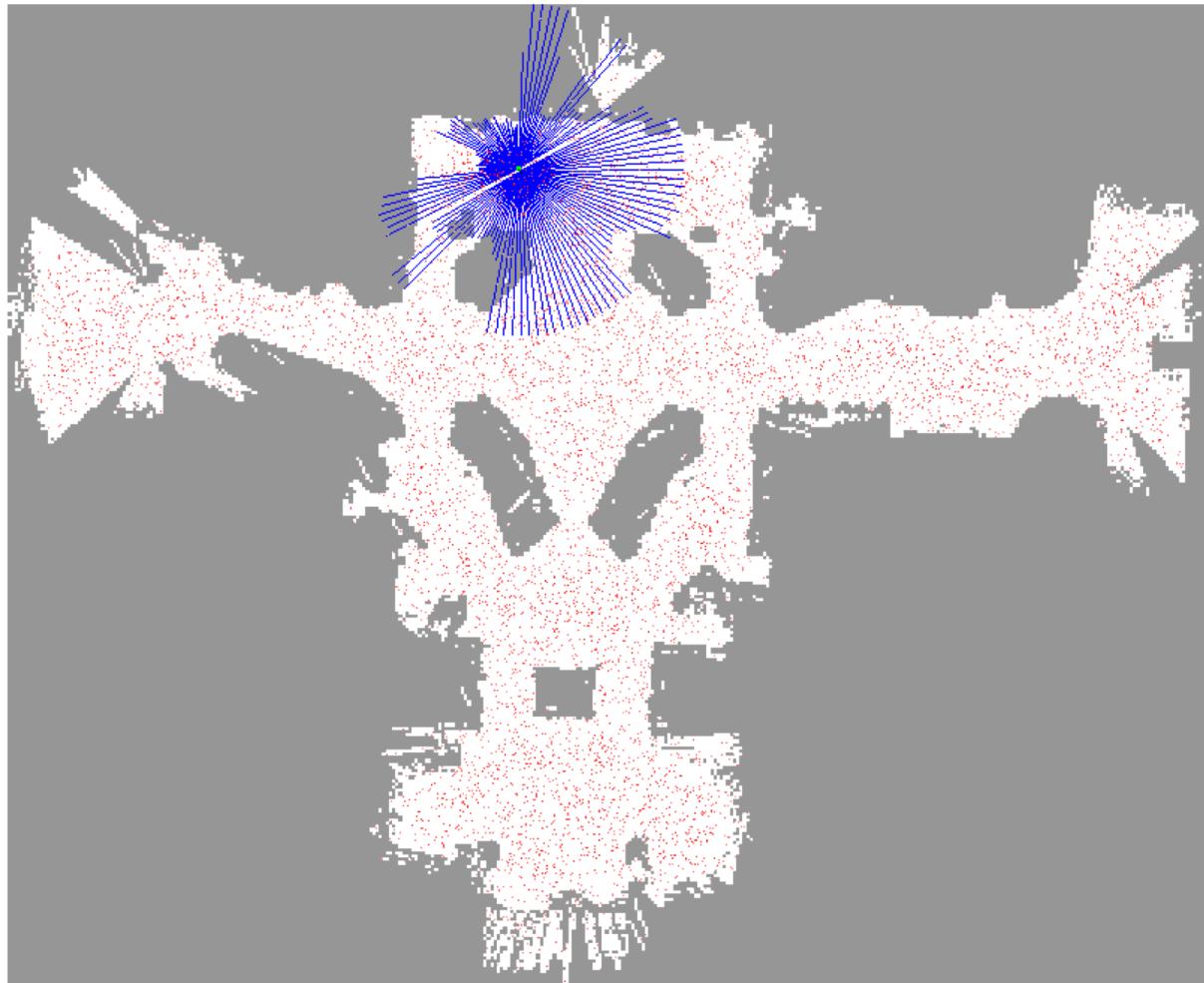


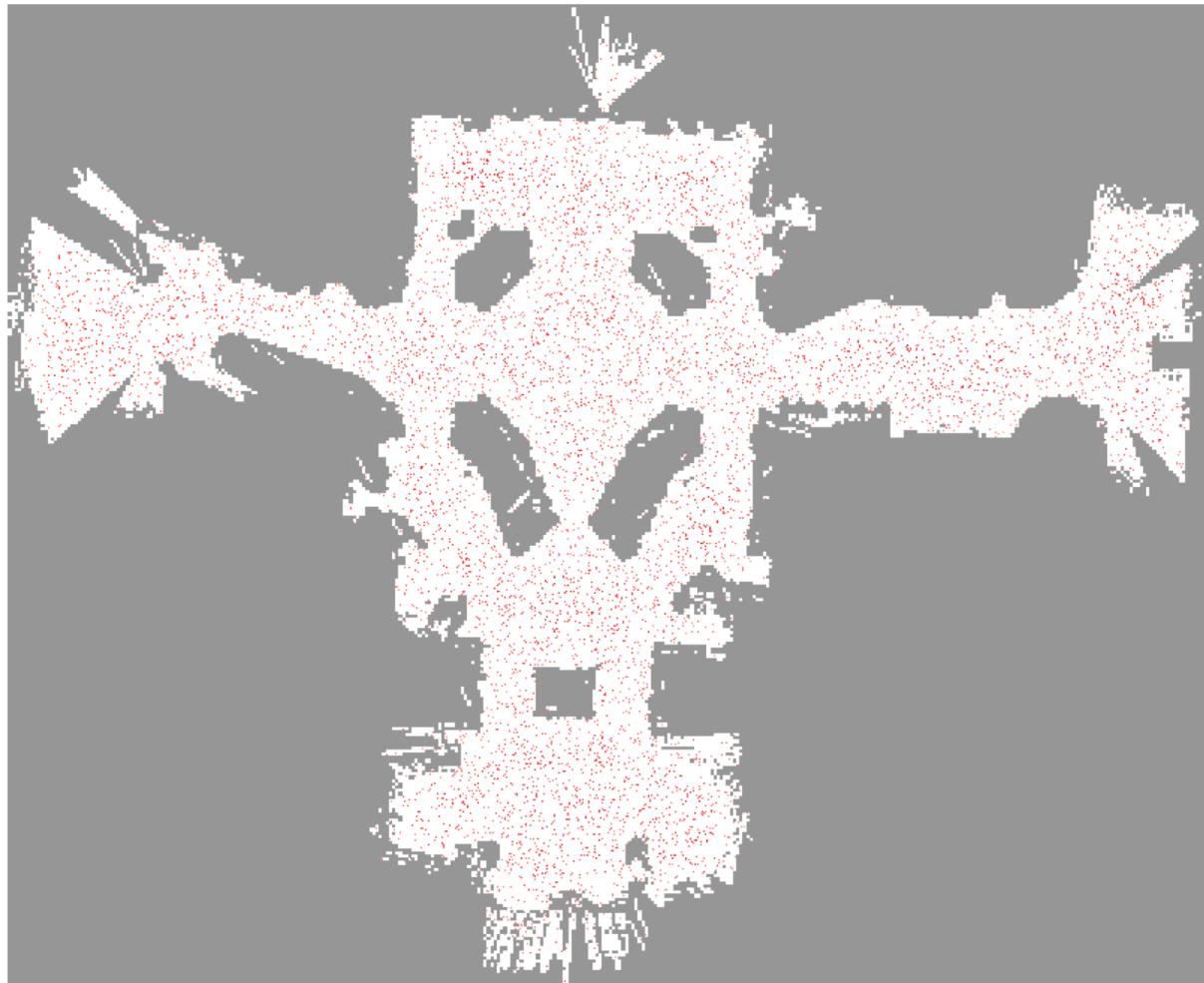
Posibles problemas

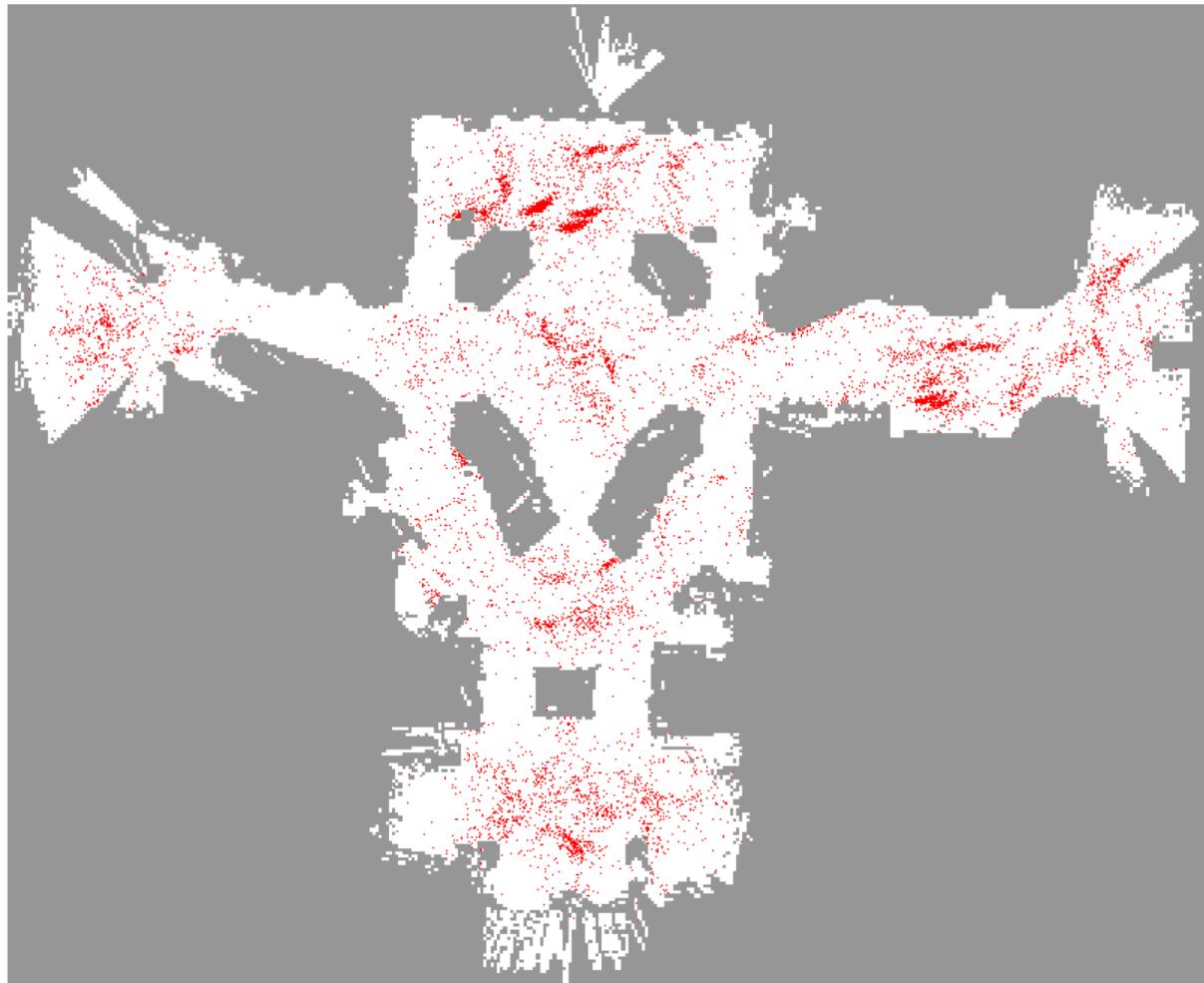
- Ajustar número de partículas:
 - Pocas: nunca localizaremos al robot
 - Muchas: consume mucho tiempo
- Ajustar varianzas y diversos parámetros

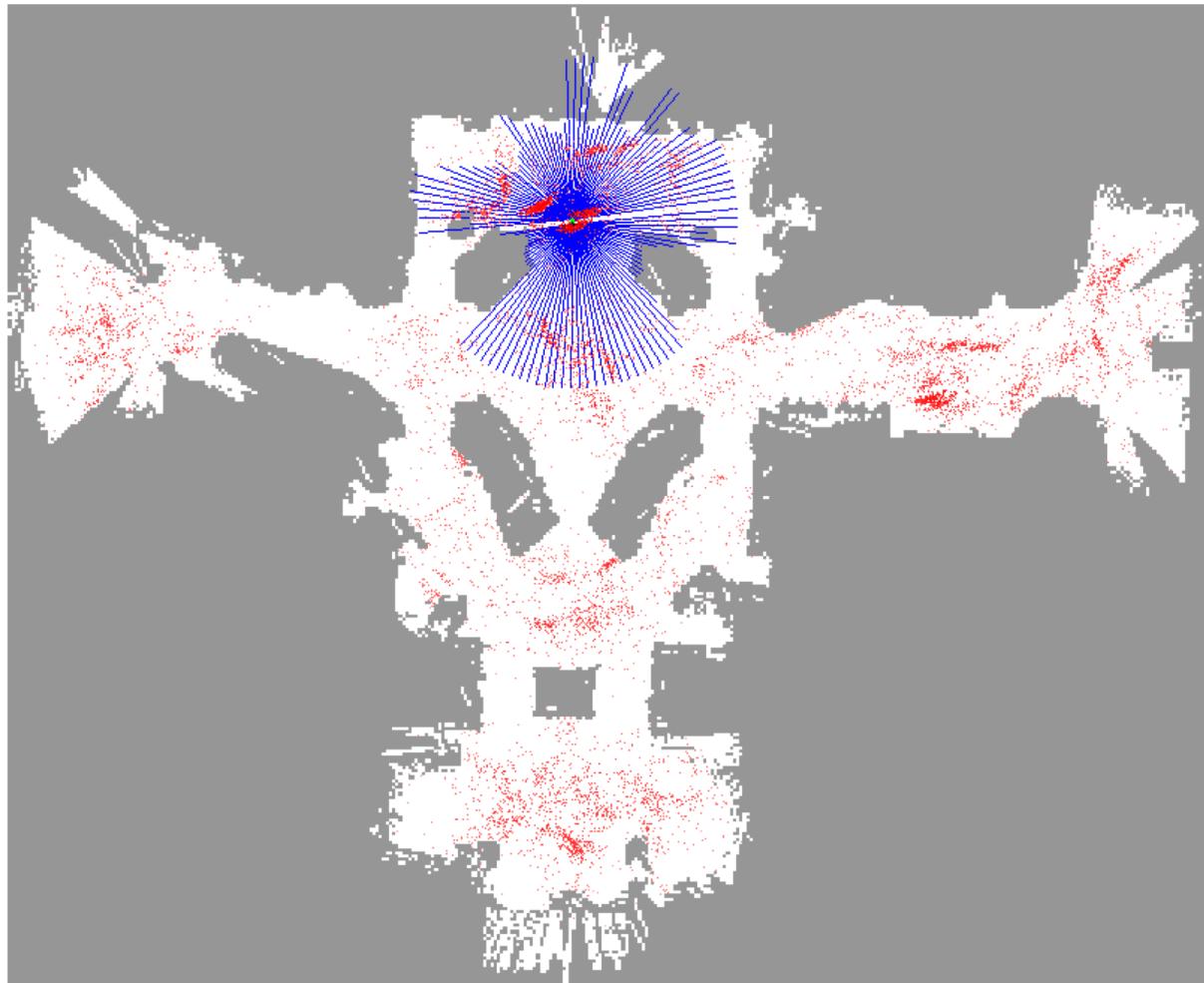


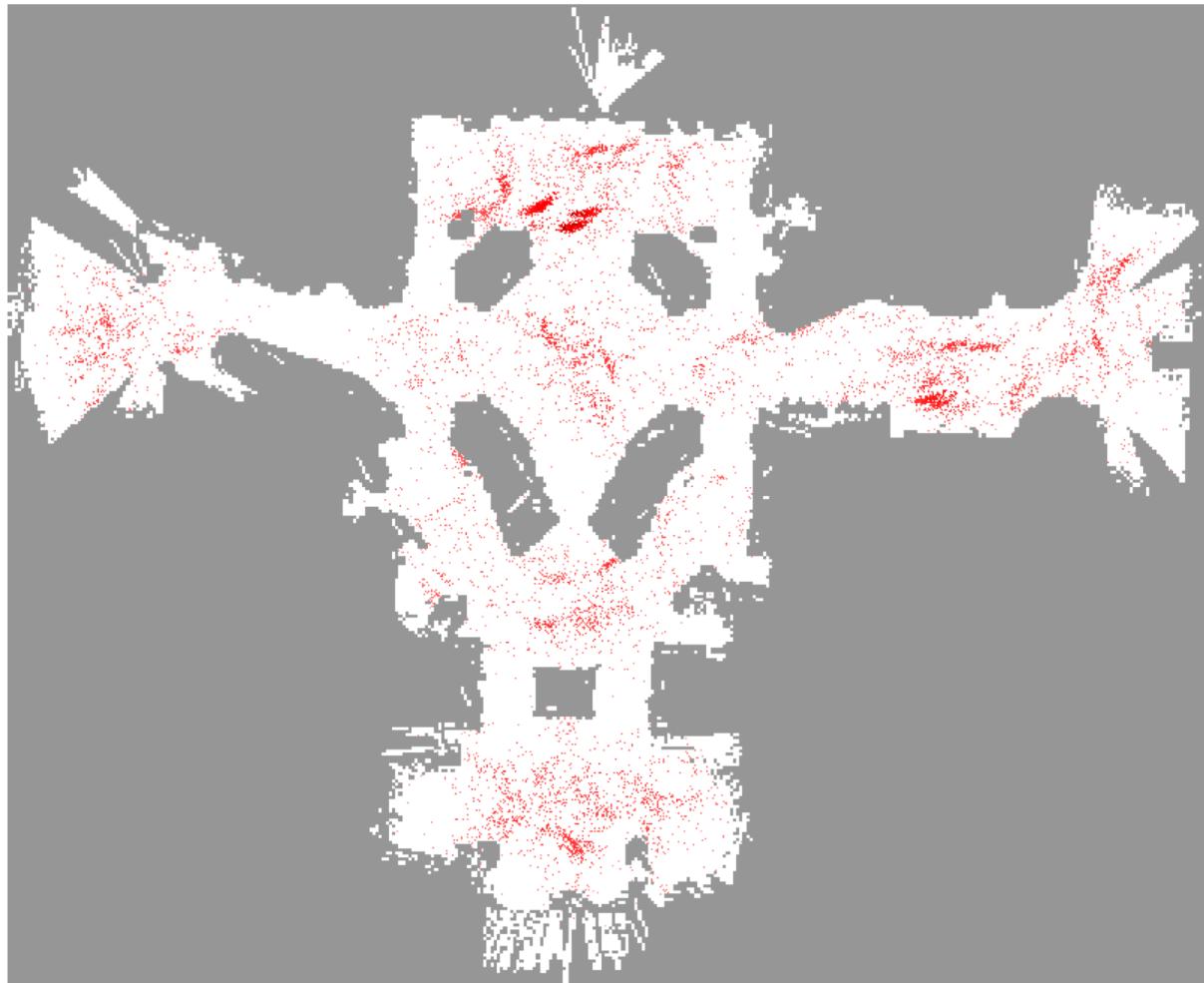


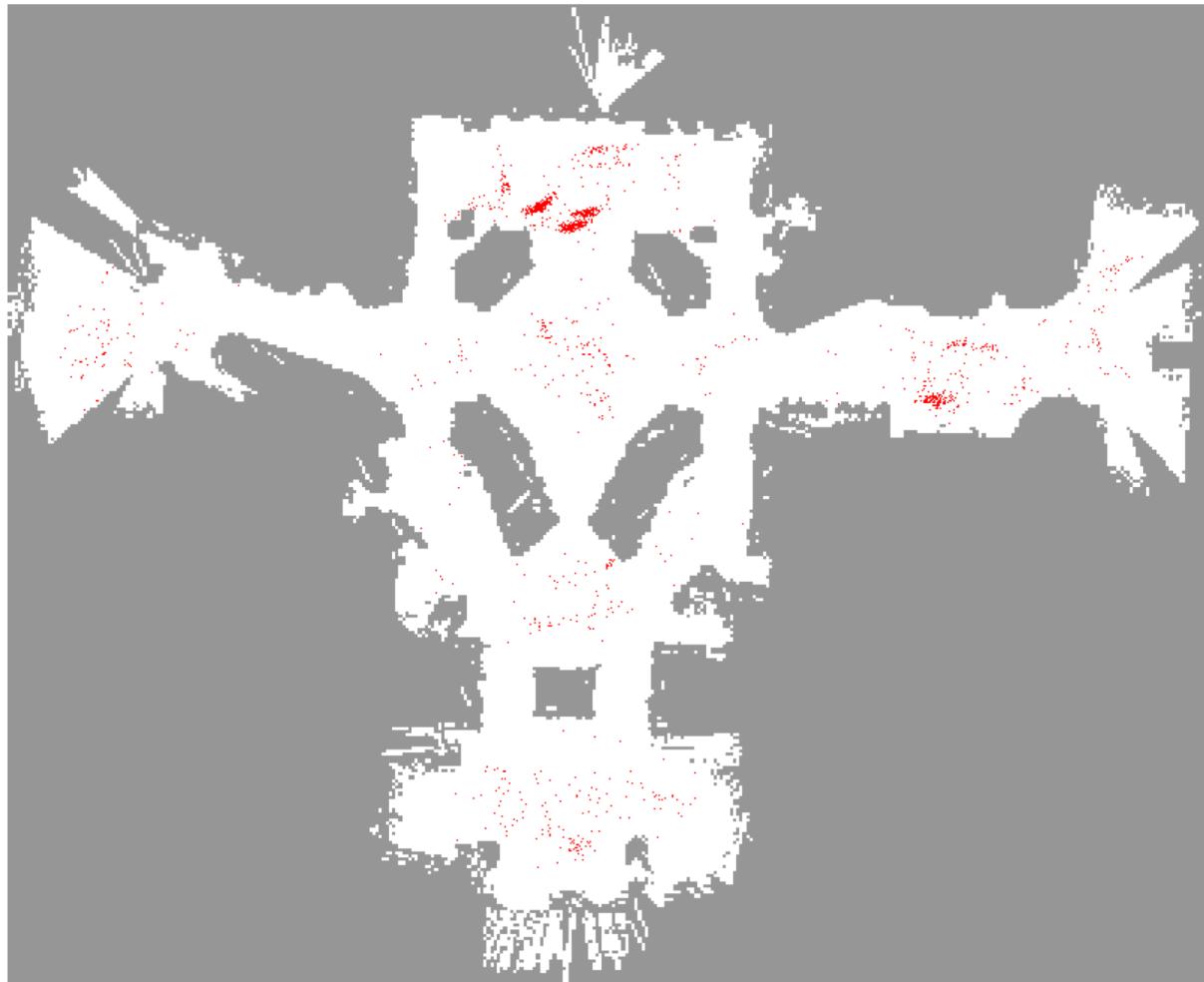




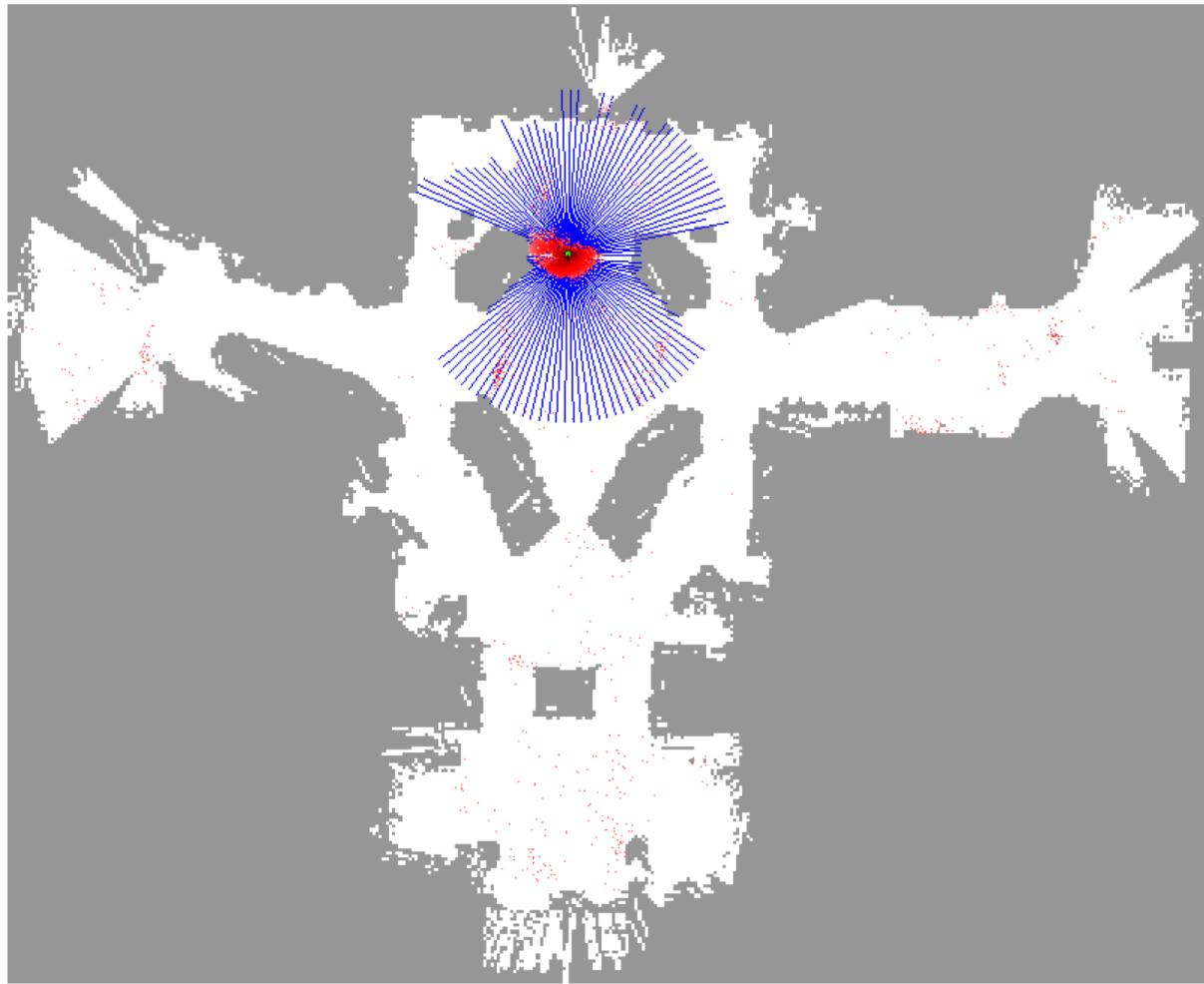


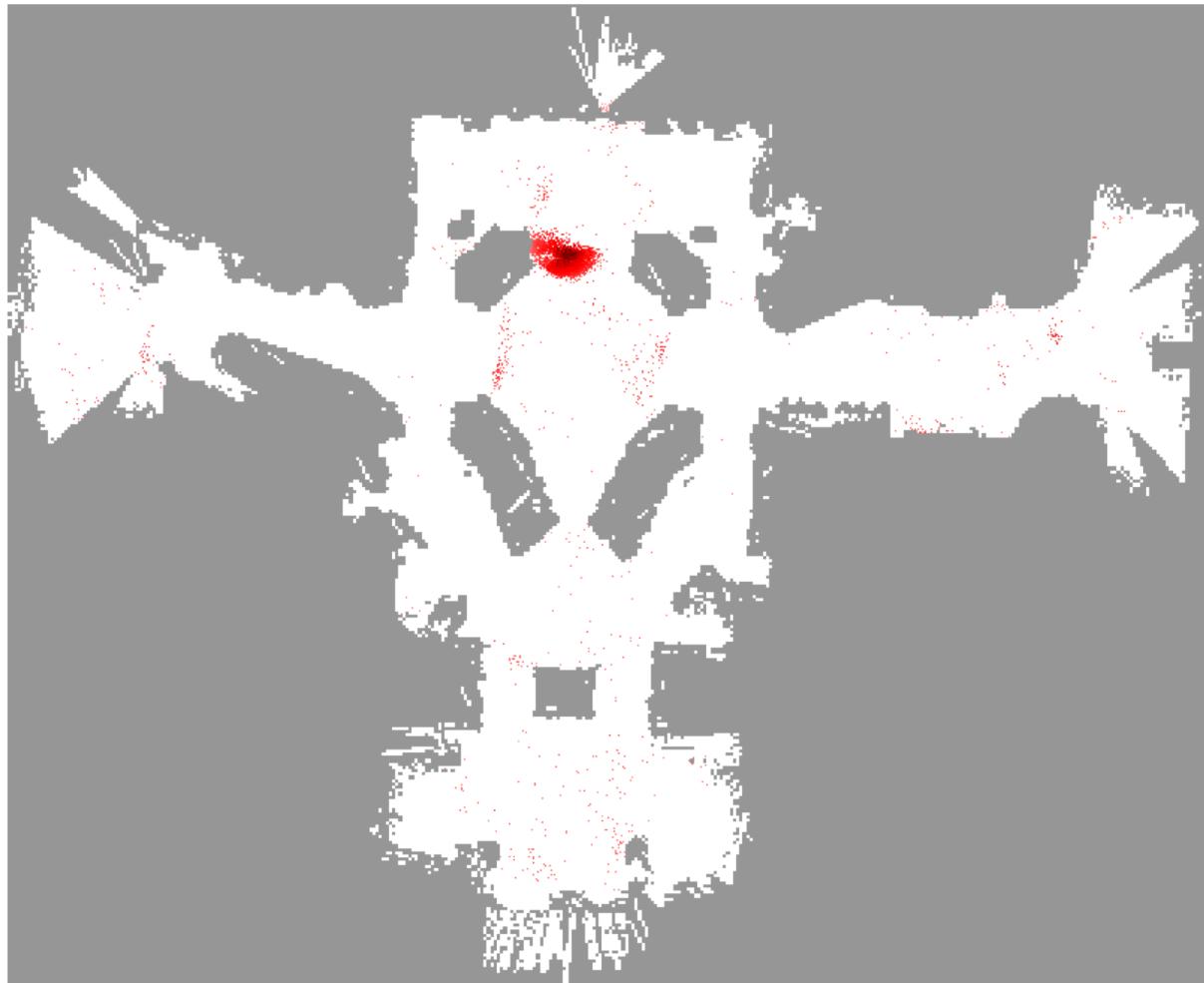


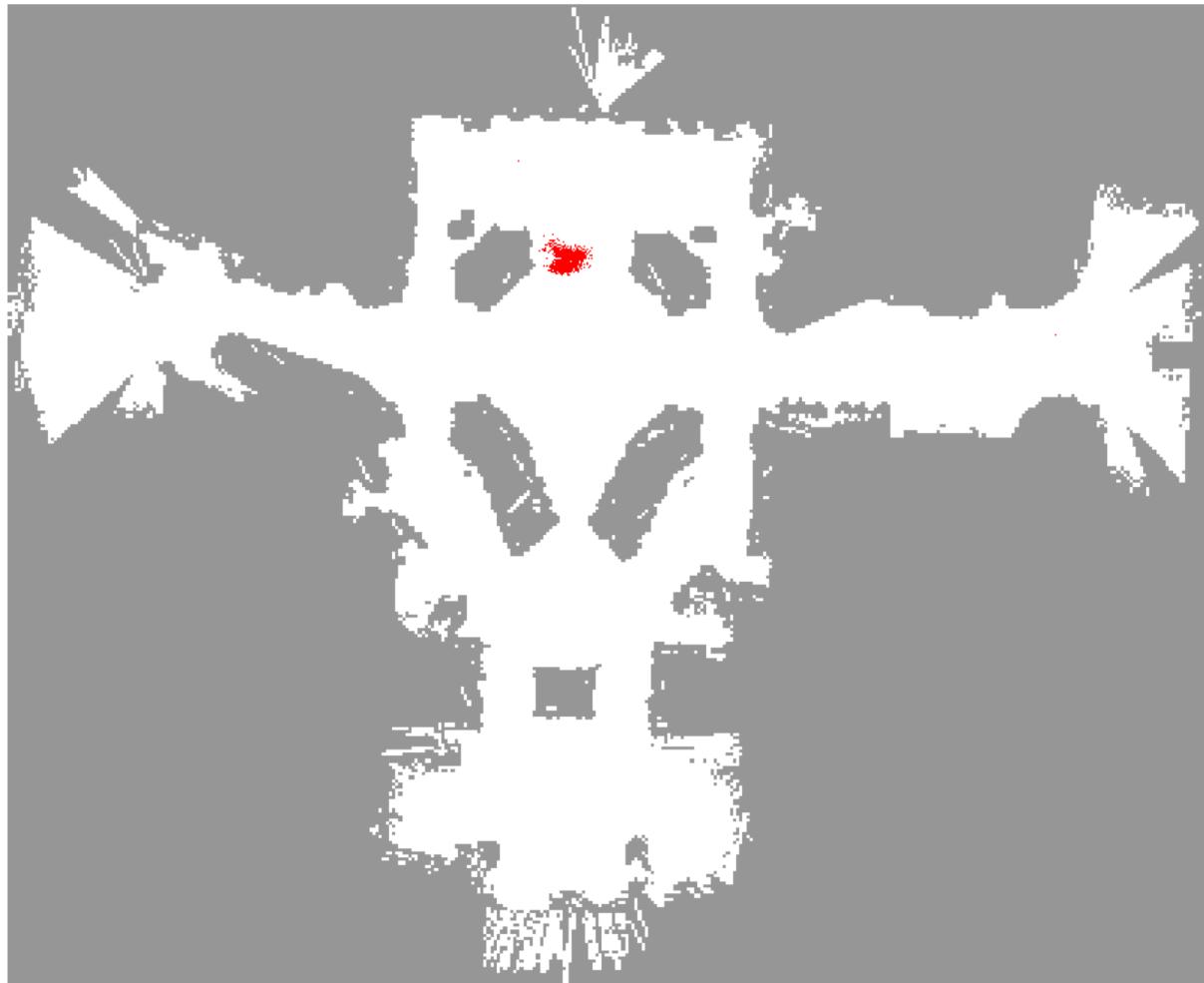


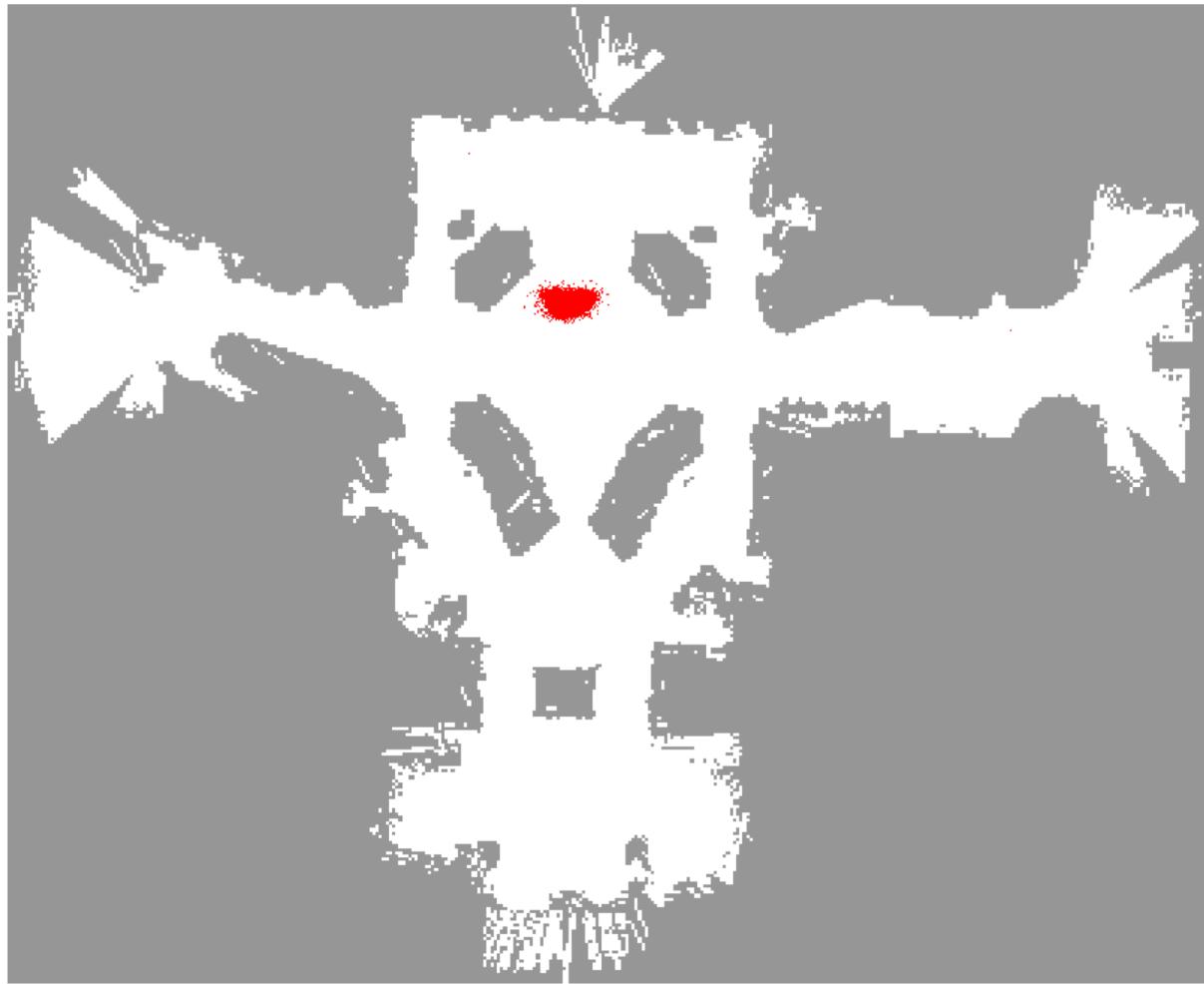


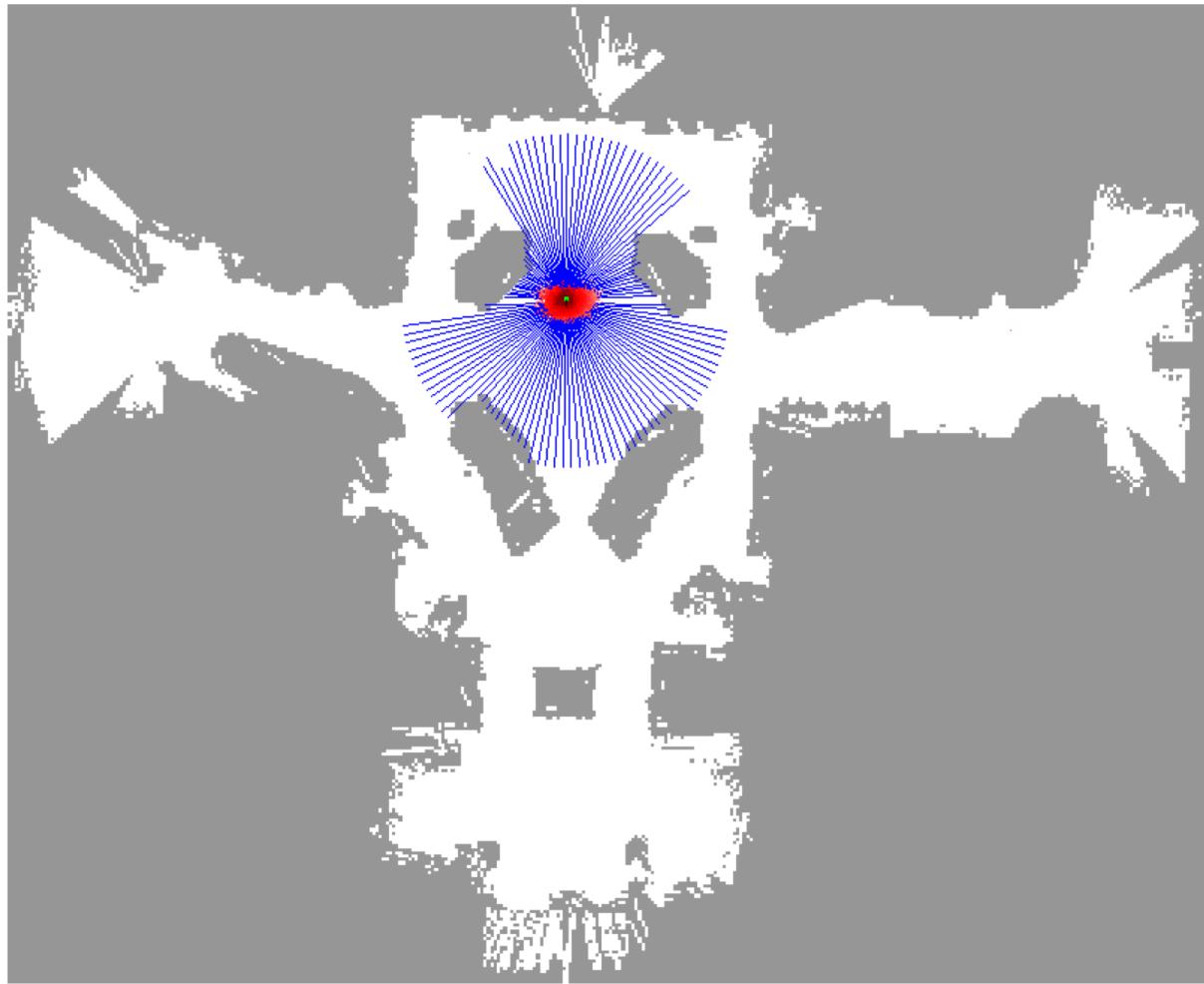


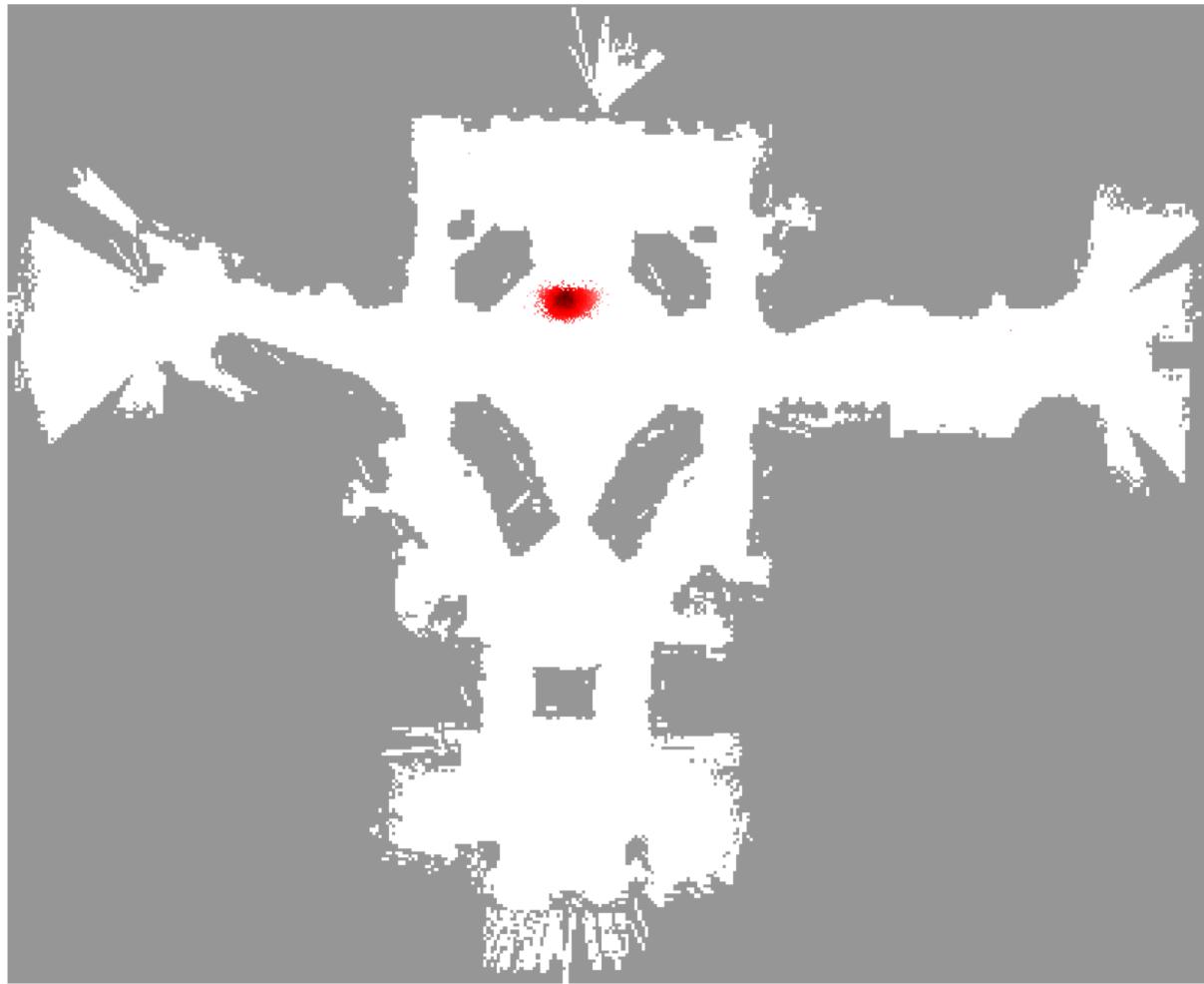


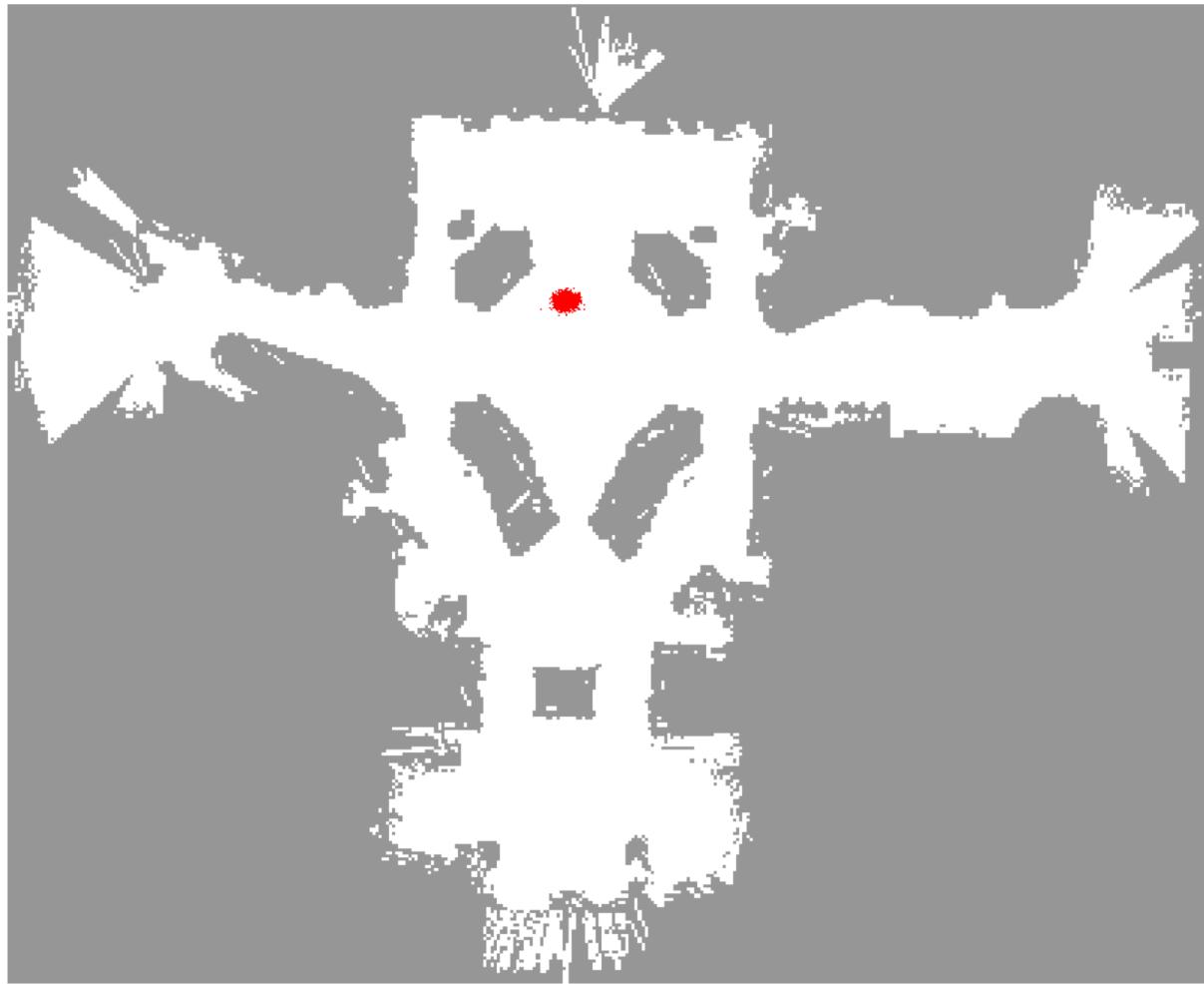


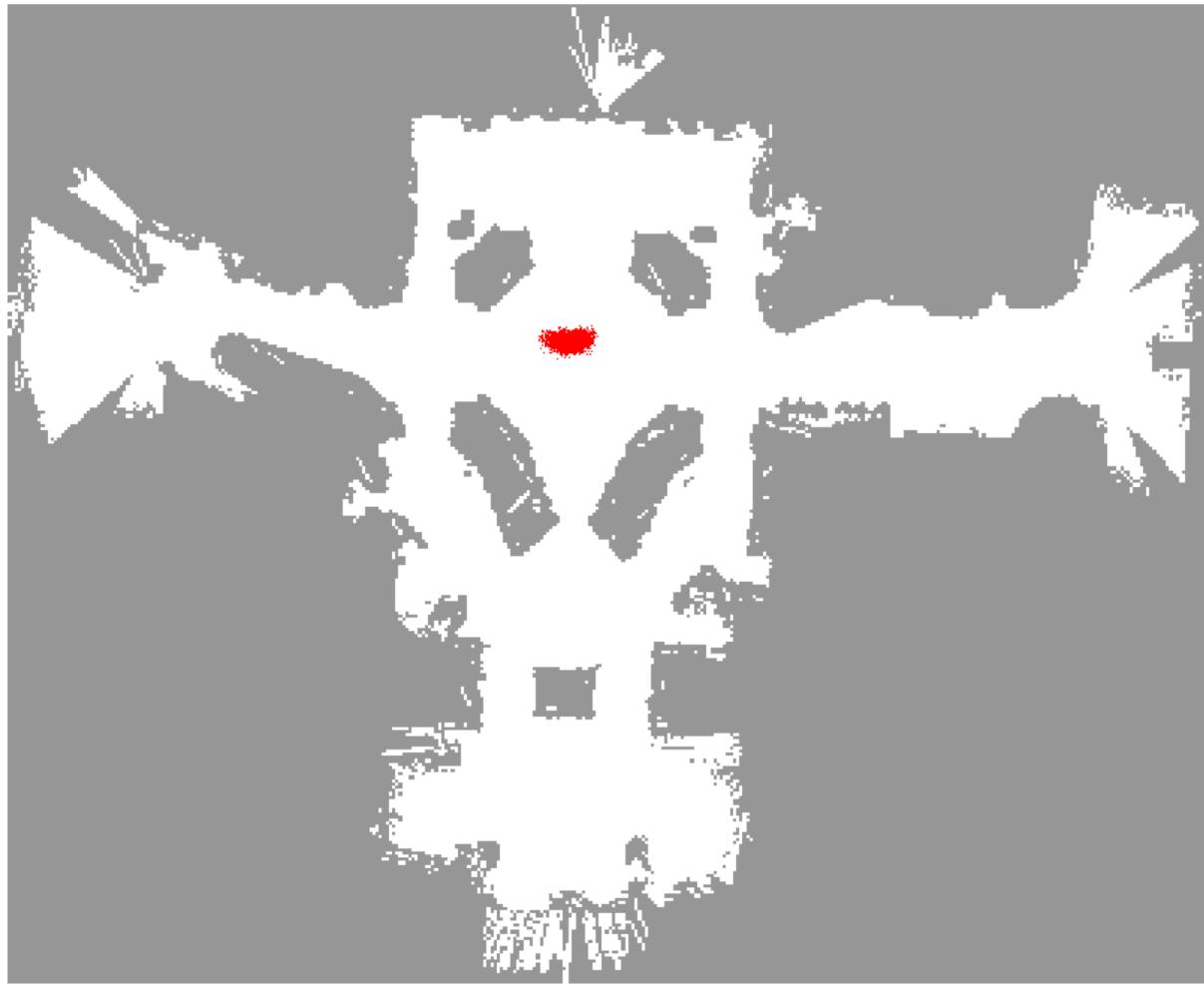


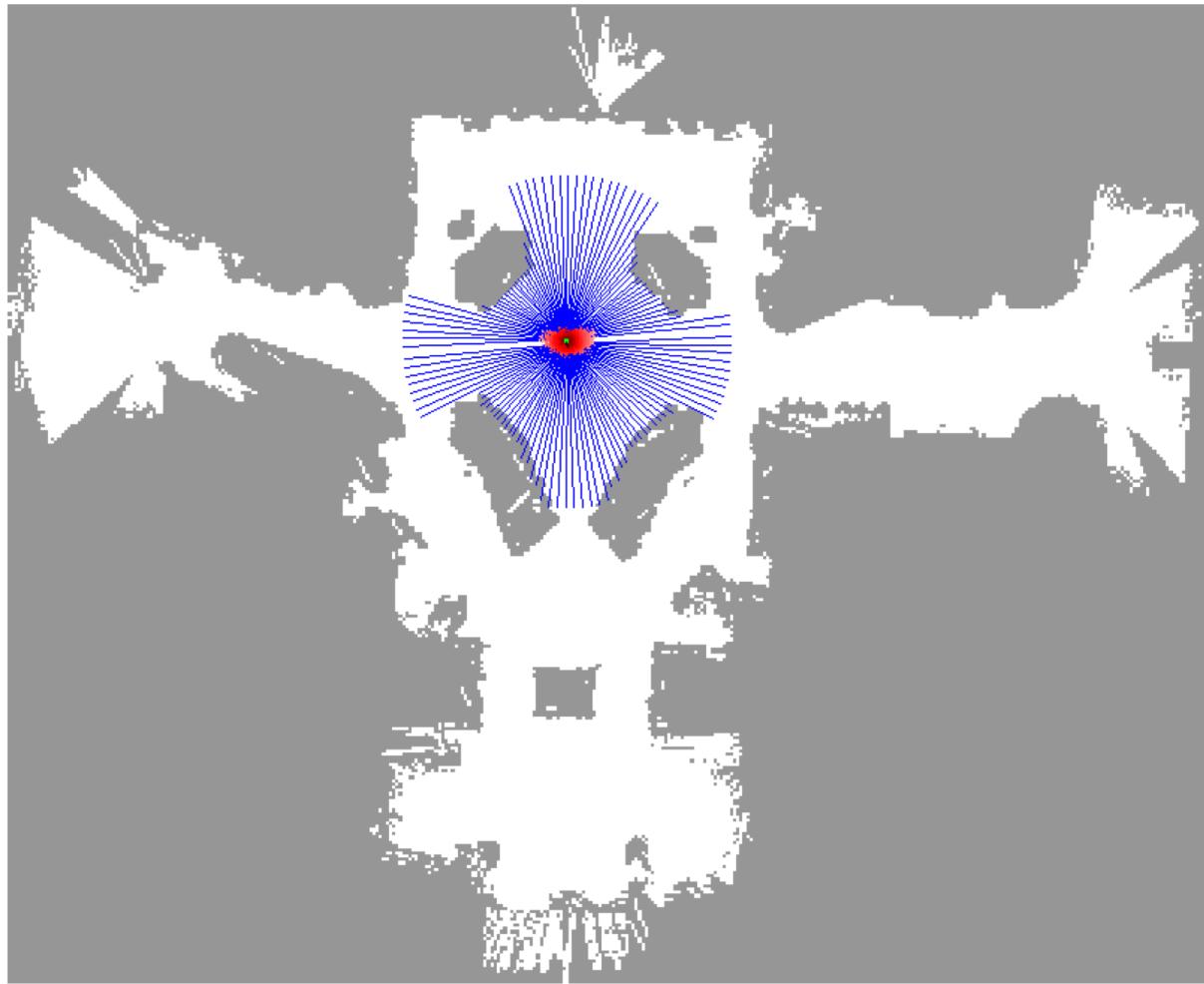








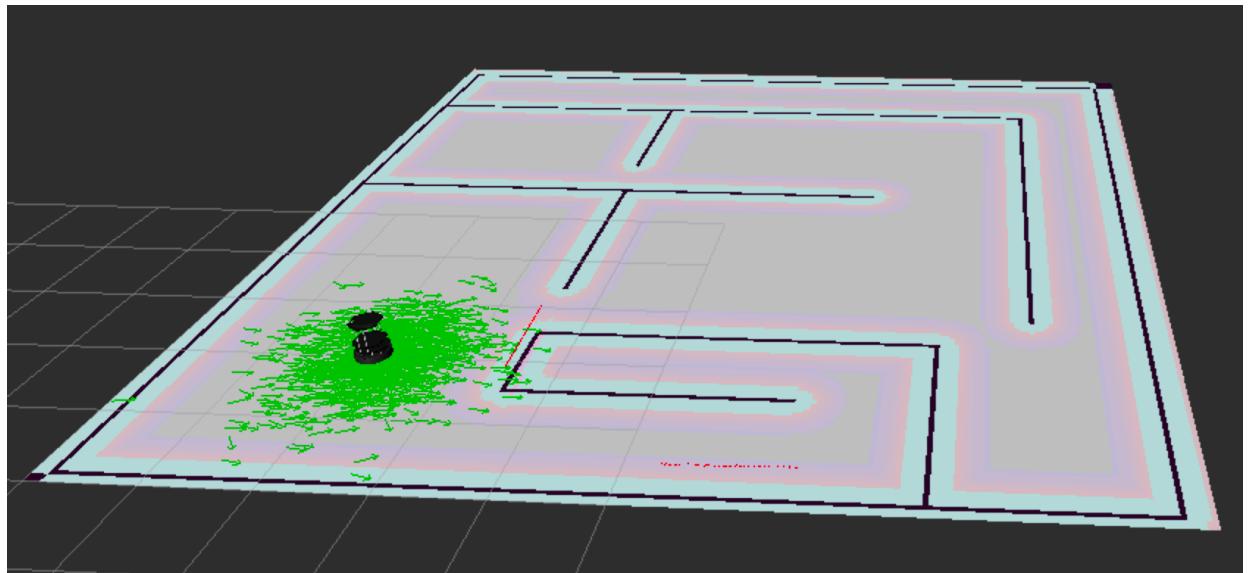




- ¿Cómo podemos afrontar errores de localización?
 - Por ejemplo, el secuestro del robot y su ubicación en una nueva posición.
- Insertar un número aleatorio de partículas (poses aleatorias)
 - Esta estrategia cumple con el principio de que un robot se puede “teletransportar” a cualquier punto en el mapa
 - Podemos aumentar el número de partículas aleatorias, cuando la probabilidad media de las observaciones es baja.

Filtros de partículas en ROS

- Paquete AMCL (<http://wiki.ros.org/amcl>) : implementa un filtro de partículas para localización $\langle x, y, \theta \rangle$
- Lo más sencillo es probar una demo en el simulador
`roslaunch turtlebot_stage turtlebot_in_stage.launch`



Resumen

- Los filtros de partículas son una implementación de filtros bayesianos
- Representan la probabilidad a posteriori mediante un conjunto de pesos
- Pueden modelar distribuciones arbitrarias
- Los pesos se computan de acuerdo a la diferencia entre la observación propuesta y la objetivo.
- En el contexto de la localización, las partículas se desplazan siguiendo el modelo de movimiento
- En el proceso de “resampling” nuevas partículas son muestreadas de acuerdo a la probabilidad de la partícula

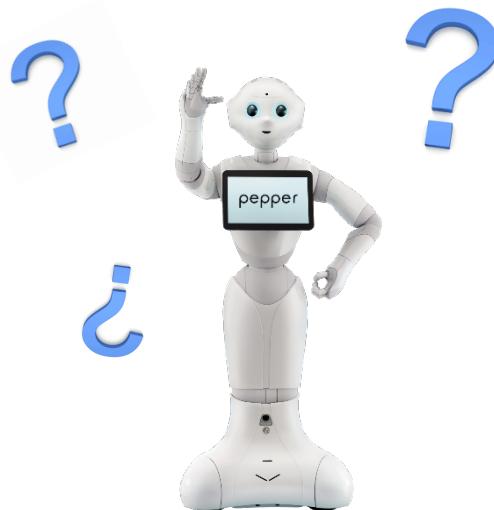
Bibliografía

Teoría

- Chapter 5. Introduction to Autonomous Mobile robots. Roland Siegwart and Illah R. Nourbakhsh. A Bradford Book.
- Chapter 7 & 8. Probabilistic Robotics. MIT Press. Thrun, Burgard, Fox.

Robots Móviles

Localización bayesiana



Sergio Orts Escolano
Otto Colomina Pardo

sorts @ ua.es

