Universidad de San Andrés, Depto. de Ingeniería

Práctica 3 - Filtros de Partículas

I-402 - Principios de la Robótica Autónoma

Prof. Dr. Ignacio Mas, Tadeo Casiraghi y Bautista Chasco

2 de abril de 2025

Fecha límite de entrega: 18/04/25, 23:59 hs.
Modo de entrega: Enviar por el Aula Virtual del Campus todo el código comentado y los gráficos (.jpg ó .pdf), todo en una sola carpeta comprimida.

1. Filtro de Partículas

En este ejercicio se implementará un filtro de partículas basándose en una estructura de código provista por la cátedra.

1.1. Notas preliminares

La estructura provista contiene los modelos de movimiento y de medición, para que el esfuerzo del desarrollo sea en el filtro propiamente dicho. El archivo comprimido que se provee incluye las carpetas:

- **data** contiene la descripción del mundo y las lecturas del sensor
- **code** contiene la estructura del filtro de partículas con funciones para ser completadas
- **plots** guarda los gráficos generados como resultado

Para ejecutar el filtro, correr en la terminal `python particle_filter_framework.py sensor_data.dat world.dat N`, donde `N` es el número de partículas.
Nota: Para que el algoritmo corra sin errores, antes hay que completar las funciones `resample_particles, get_mean_position` y `measurement_prob_range`.

La estructura de software incluye la generación de gráficos de los resultados. Luego de ejecutarse el filtro, se genera una figura de *matplotlib* para su visualización.
Algunos consejos adicionales:

- Desactivar la visualización para acelerar la ejecución del algoritmo comentando la función `get_mean_position`.
- Para leer los datos del mundo y del sensor, se usan diccionarios. Las funciones `read_sensor_data` y `read_world_data` leen los datos de los archivos correspondientes y arman diccionarios con *timestamps* como keys primarios. Para acceder a los datos del sensor en el diccionario, puede usarse:

  ```
  data_dict[timestamp,'sensor']['id']
  data_dict[timestamp,'sensor']['range']
  data_dict[timestamp,'sensor']['bearing']
  ```

  Para la información de odometría, el diccionario puede accederse como:

  ```
  data_dict[timestamp,'odom']['r1']
  data_dict[timestamp,'odom']['t']
  data_dict[timestamp,'odom']['r2']
  ```

  Para acceder a la pocisión de las *landmarks* en `world_dict`, puede usarse:

  ```
  world_dict[id]
  ```

1.2. Ejercicio

Un filtro de partículas consiste principalmente de tres pasos:

1. Muestrear nuevas partículas usando el modelo de movimiento.
2. Calcular el peso de las nuevas partículas usando el modelo de medición.
3. Calcular el nuevo *belief* muestreando las partículas de manera proporcional a sus pesos con reemplazo.

El modelo de movimiento (1) está implementado en la estructura de código provista. En este ejercicio se deben implementar (2) y (3).

- Completar la función `measurement_prob_range`. Esta función implementa el paso de actualización del filtro de partículas, usando un sensor de distancia. Toma como entrada *landmarks* (`world_dict`) y observaciones de *landmarks* (*id* –identificación– y valor de distancia medida). La salida es un peso para cada partícula. En vez de calcular la probabilidad, es suficiente calcular el *likelihood p(z |x,l)*. El desvío estandard de la medición es $\sigma_r = 0,2$.
- Completar la función `resample_particles` implementando el Muestreo Estocástico Universal. La función toma como entrada un conjunto de partículas y sus correspondientes pesos y debe devolver otro conjunto de partículas.

1.3. Visualización

Se desea elegir una sola hipótesis del conjunto de partículas del filtro para representar la posición del robot. Implementar la función `get_mean_position` que se usa para graficar la pose del robot.