PRÁCTICA 4. LOCALIZACIÓN MEDIANTE FILTROS DE PARTÍCULAS CON PLAYER/STAGE

El objetivo de esta práctica es aprender a manejar las herramientas que proporciona Player para localizar un robot dentro de un mapa de entorno utilizando métodos probabilísticos que permitan mantener más de una hipótesis sobre su posición.

Concretamente se va a trabajar con los siguientes drivers, interfaces y utilidades de Player:

- **Driver "amcl"**
  Se trata de un driver virtual que implementa el Método de Localización de Monte Carlo Adaptativo (Adaptive Monte Carlo Localization - AMCL).
  Conceptualmente, este driver mantiene un filtro de partículas sobre el conjunto de posibles posiciones del robot, y actualiza estas partículas utilizando los datos de odometría y láser. El driver también requiere un mapa a priori del entorno con el cual comparar los datos de los sensores. El filtro es adaptativo ya que ajusta dinámicamente el número de partículas: cuando la posición del robot tiene gran incertidumbre el número de partículas es elevado, y éste se reduce a medida que el robot se va localizando en el entorno.

- **Interfaz "localize"**
  Este interfaz proporciona información sobre la posición del robot cuando ésta no es determinística (como sucede con el interfaz `position2d`), sino que se compone de múltiples hipótesis. Por lo tanto, es el interfaz apropiado para acceder al driver `amcl`. Otro driver que también admite el interfaz `localize` es `fakelocalize`.
  Hay que destacar que el driver `amcl`, además del interfaz `localize` que permite acceder a todas las hipótesis de posición del robot, también admite el interfaz `position2d` que proporciona acceso a la posición (hipótesis) más probable.

- **Utilidad "playernav"**
  Es una utilidad (en definitiva, una aplicación cliente que se conecta al servidor `player` para visualizar datos y enviar comandos a los diferentes dispositivos, igual que `playerv`) que proporciona un interfaz gráfico para facilitar el control de dispositivos tipo `localize` y tipo `planner`. Permite inicializar las hipótesis sobre la posición de varios robots simplemente colocándolas sobre el mapa con el ratón, así como indicar posiciones de destino con el mismo método.

4.1. Aplicación básica a desarrollar

La aplicación a desarrollar permitirá comprobar el funcionamiento del Método de Localización de Montecarlo Adaptativo (AMCL) para localizar un robot dentro de un entorno de simulación como el mostrado en la figura 1. Este algoritmo se encuentra programado en el driver virtual `amcl`.

Una particularidad del driver `amcl` es que sólo permite inicializar las partículas según una distribución gaussiana, de la cual hay que indicar su media (posición inicial estimada/aproximada del robot) y su varianza (grado de incertidumbre sobre la posición anterior). Por lo tanto, resulta apropiado para resolver problemas de localización local (*tracking*) en los que la posición inicial se conoce de forma aproximada. Sin embargo, para resolver la localización global del robot dentro del mapa la única solución es utilizar una gaussiana centrada en el mapa y con una varianza muy grande que permita que haya partículas distribuidas por todo el entorno. Esta no es una solución óptima (ya que las partículas no presentan una distribución aleatoria uniforme), pero permite resolver problemas de localización global.

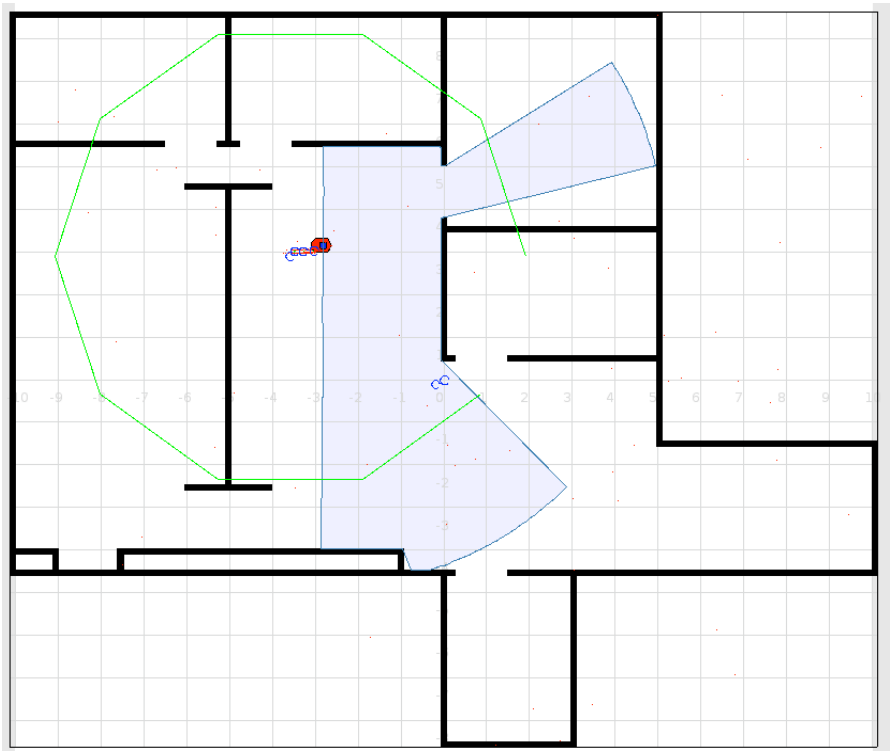

Figura 1

4.2. Fases de desarrollo

El proyecto se realizará siguiendo las siguientes fases:

4.2.1. Configuración de Stage (practica4.world)

El bitmap del mapa a utilizar, mostrado en la figura 1, se encuentra entre los ejemplos proporcionados en la propia distribución de player utilizando una escala de 0.025 m/píxel (`/home/alumno/Stage-3.2.2-Source/worlds/bitmaps/autolab.png`).

El robot a simular debe ser un robot `pioneer2dx` con un `laser sick` incorporado (anidado) y con error de odometría no nulo (puede utilizarse el valor por defecto del modelo `position`).

4.2.2. Configuración de Player (practica4.cfg)

La figura 2 muestra todos los drivers necesarios y sus correspondientes interfaces de acceso.

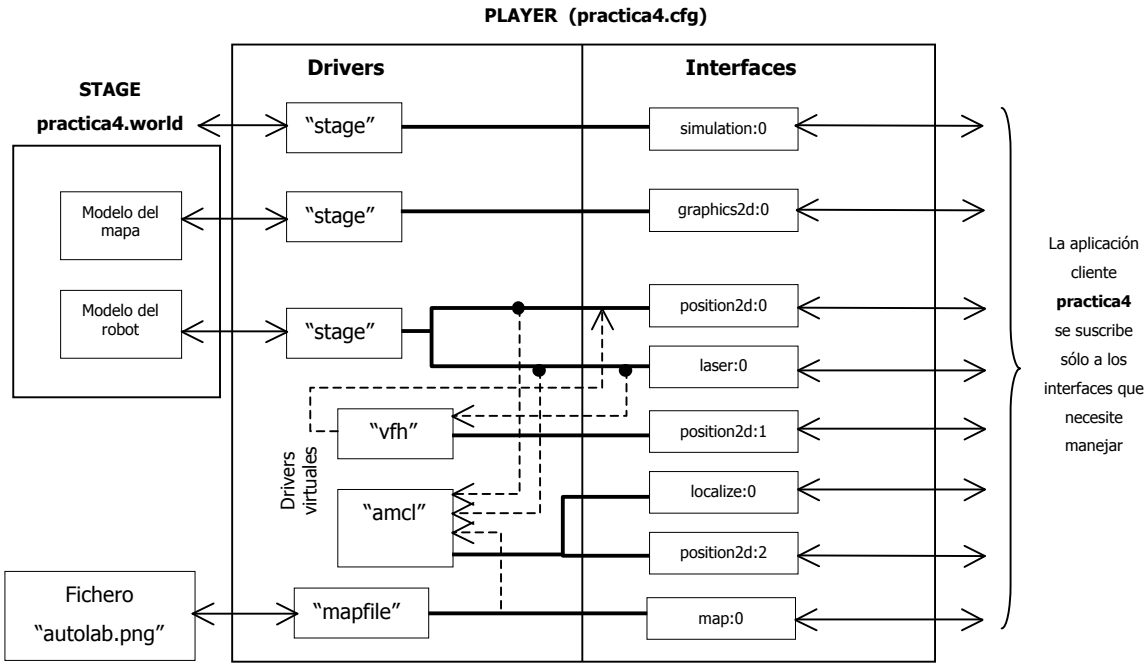

Figura 2

Al modelo del mapa de stage se le vincula el interfaz de acceso `graphics2d:0`, para permitir a la aplicación cliente dibujar sobre él (para visualizar las partículas, la posición más probable, etc.)

Para acceder al mapa a través del interfaz map:0 se utiliza el driver mapfile, que accede directamente al fichero que contiene el mapa. El driver virtual `amcl` requiere conocer el mapa de entorno.

Por otro lado, el modelo del robot presenta los interfaces `position2d:0` y `laser:0` para controlar la base robótica y obtener datos del láser respectivamente.

Se va a incorporar el driver virtual `vfh` para que el robot pueda navegar de forma segura (evitando obstáculos) hacia las posiciones de destino indicadas a través del interfaz `position2d:1`. Este driver necesita conocer los datos del láser (a través de `laser:0`) y enviar comandos a la base robótica (a través de `position2d:0`).

El driver virtual `amcl` necesita conocer el mapa (`map:0`), los movimientos realizados (`position2d:0`) y los datos del láser (`laser:0`) para actualizar convenientemente el filtro de partículas, cuyos datos se ofrecen a través del interfaz `localize:0`. Este driver admite también el interfaz de tipo `position2d` para ofrecer sólo la posición más probable (en este caso se le da la dirección `position2d:2`).

El driver `amcl` admite varias opciones de configuración que pueden indicarse mediante sus correspondientes pares opción/valor dentro de su sección `driver()` del fichero *.cfg. Entre estas opciones, que pueden consultarse en el manual de Player, cabe destacar las siguientes, para las cuales se indica también su valor por defecto:

- `init_pose [0 0 0]`. Estimación inicial de la posición del robot en [m m rad]
- `init_pose_var [1 1 2pi]`. Varianzas para caracterizar la incertidumbre inicial de la estimación anterior
- `pf_min_samples 100`. Mínimo número de partículas a mantener en el filtro
- `pf_max_samples 10000`. Máximo número de partículas a mantener en el filtro

4.2.3. Programación de la aplicación cliente (practica4.c)

La figura 3 muestra un organigrama básico de la aplicación a desarrollar:

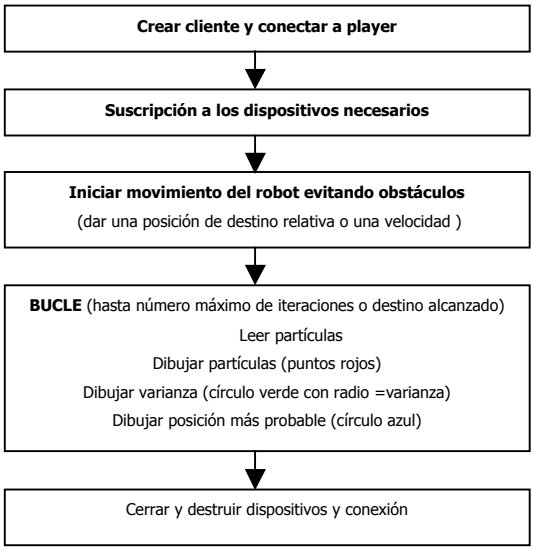

Figura 3

4.2.4. Realización de pruebas

- Compruebe el funcionamiento del sistema de localización cuando la hipótesis inicial del filtro se coloca aproximadamente en la posición real del robot.

- Compruebe el funcionamiento del sistema de localización cuando la hipótesis inicial se coloca en una posición errónea.

- Compruebe la capacidad del sistema para resolver el problema de la localización global. Para ello, inicialice el filtro siempre con posición inicial [0 0 0] y una varianza elevada que cubra todo el mapa de entorno. Realice múltiples pruebas con el robot en diferentes posiciones iniciales para comprobar la convergencia del filtro.

4.3. Ampliación: incorporación de un planificador de rutas mediante el driver "wavefront" y el interfaz tipo "planner"

Para completar la aplicación de navegación, se propone incorporar al sistema de localización desarrollado anteriormente un planificador de rutas que permita alcanzar con éxito una posición de destino indicada de forma absoluta respecto al sistema de referencia global. Una solución es utilizar el driver "`wavefront`" a través de un interfaz de tipo "`planner`".

El driver `wavefront` funciona del siguiente modo: tras recibir una nueva posición de destino a través del interfaz `planner`, se utiliza el mapa de entorno (debe estar disponible a través de un interfaz tipo `map`) y la posición actual del robot (la más probable, proporcionada a través de un interfaz `position2d`) para calcular un camino hacia el destino formado por un conjunto de destinos intermedios. Estos puntos intermedios se pasan, secuencialmente, al dispositivo `position2d` subyacente, que debe ser capaz de realizar una navegación local segura hacia dicho punto (el driver `vfh` es un buen candidato para este dispositivo).

Para incorporar el planificador de rutas al fichero de configuración de player disponible hasta el momento, debe tenerse en cuenta que el driver `wavefront` proporciona un interfaz de tipo `planner`, y requiere tres dispositivos:

a. Un dispositivo `position2d` de entrada, con información sobre la posición actual del robot (en este caso procedente del driver `amcl`).
b. Un dispositivo `position2d` de salida al cual enviarle las posiciones intermedias de destino (en este caso asociado al driver `vfh` que realiza la navegación local).
c. Un dispositivo tipo `map` con el mapa de entorno.

Para diferenciar qué dispositivo `position2d` se utiliza como entrada y cuál se utiliza como salida, deben utilizarse en su dirección las palabras clave `input` y `output` respectivamente.

Los pares opción/valor admitidos por el driver `wavefront` pueden consultarse en la página web del proyecto.

Para comprobar el funcionamiento del planificador de rutas sería necesario añadir a la aplicación cliente la parte de código que pide al usuario la posición de destino a través del teclado y envía dicho destino al dispositivo `planner`. Sin embargo, para reducir la duración de

la práctica, se propone comprobar su funcionamiento directamente mediante la utilidad `playernav` que se describe en el siguiente apartado.

4.4. Ampliación: manejo de la utilidad playernav

`Playernav` es una aplicación cliente incluida como utilidad en el proyecto Player/Stage. Básicamente, `playernav` proporciona control sobre dispositivos tipo `localize` y tipo `planner`, proporcionando además un interfaz gráfico independiente de Stage. Permite inicializar las hipótesis de localización (partículas) de varios robots simplemente colocándolas sobre el mapa, establecer posiciones de destino para cada robot del mismo modo, y observar la ruta planificada así como el movimiento del robot hacia los puntos de destino intermedios.

Para ejecutar `playernav` debe escribirse el siguiente comando:

```
$ playernav [-fps <dumprate>] [-zoom <zoom>] [-aa {0|1}] [-map <map_idx>] <host:port>
                                    [<host:port>...]
```

con las siguientes opciones:

```
-fps <dumprate> : when requested, dump screenshots at this rate in Hz (default: 5Hz)
-zoom <zoom> : initial level of zoom in the display (default: 1)
-aa {0|1} : whether to use anti-aliased canvas for display; the anti-aliased canvas
looks nicer but may require more processor cycles (default: 1)
-map <map_idx> : the index of the map to be requested and displayed (default: 0)
```

`Playernav` se conecta a player en cada combinación `host:port` indicada en línea de comandos (es decir, se puede conectar a varios robots, cada uno con su propio player). Para cada una de ellas, trata de suscribirse a los siguientes dispositivos: `localize:0` y `planner:0`. Además, trata de suscribirse al dispositivo `map:<map_idx>` del primer servidor (ya que se supone que el mapa de entorno es igual para todos los robots). Si la suscripción al mapa falla, `playernav` finaliza. Si el resto de suscripciones fallan, se envía un aviso pero `playernav` se sigue ejecutando.

Al arrancar `playernav`, aparece una ventana mostrando el mapa. Mediante las barras de la derecha y la parte inferior es posible mover el mapa, mientras que la de la izquierda modifica el zoom. Se muestra un icono representando cada robot, y al pasar el ratón sobre cada uno de ellos es posible ver su dirección (`host:port`).

Si no se realiza ninguna acción, la hipótesis inicial sobre la posición de cada robot (partículas iniciales) es la indicada en el correspondiente fichero `*.cfg`. Sin embargo, es posible modificar dichas hipótesis del siguiente modo: utilizando el botón izquierdo del ratón, arrastrar el robot correspondiente hacia su posición inicial estimada; pulsando de nuevo con el botón izquierdo se establece la orientación deseada para dicha posición inicial.

Para indicar una posición de destino, arrastrar el robot correspondiente con el botón derecho del ratón hasta la posición de destino deseada, y pulsando una segunda vez con el botón derecho se establece la orientación en el destino. A continuación se muestra el camino con sus correspondientes puntos intermedios. Seguidamente el robot comenzará a recorrer el camino, y los puntos irán desapareciendo a medida que se van alcanzando.

Las hipótesis de localización y destinos pueden cambiarse en cualquier momento, incluso aunque el robot esté en movimiento.

Por último, para parar un robot, pulsar sobre él con el botón intermedio del ratón. Volver a pulsarlo para que reinicie el movimiento.

Como ampliación de la práctica, compara el funcionamiento del sistema de localización de la aplicación cliente desarrollada y de `playernav`, y comprueba el funcionamiento del planificador de rutas.