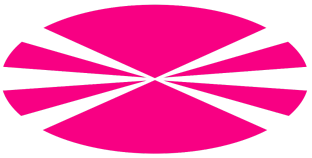

UNIVERSIDADE
DA CORUÑA

Robótica

Memoria práctica 1

Behavior-based robotics (BBR)

Óscar Alejandro Manteiga Seoane
Antonio Vila Leis

Q8 2022/2023

Comportamiento e implementación

Introducción

El comportamiento deseado para el robot EV3 tendría que satisfacer cuatro tareas básicas para que en conjunto pudiesen llegar a encontrarse con una fuente de luz y pararse delante de ella.

Antes de comentar las tareas, mencionar que existen variables globales que se utilizarán a lo largo del programa: la distancia máxima de detección, la distancia a la que se sigue la pared, la velocidad por defecto, los ángulos de giro y esquina y los umbrales de detección de luz.

Se definen los semáforos sem12, sem23 y sem34, necesarios para sincronizar las tareas y evitar condiciones de carrera y bloqueos. En estas arquitecturas donde se utilizan semáforos, se suele establecer un orden de prioridad entre las tareas que compiten por el acceso a los recursos compartidos. En nuestro caso se utilizan tres semáforos (los ya mencionados) para coordinar la ejecución de las tareas que compiten por el acceso a los sensores y motores del robot EV3. En este caso, el orden de prioridad se establece por el orden en que se adquieren los semáforos. Por ejemplo, la tarea escape adquiere primero el semáforo sem12, lo que le da prioridad sobre la tarea light y follow_wall, que deben adquirir el semáforo sem23 y sem34 respectivamente antes de poder ejecutar su código crítico. De esta manera, se evita que se produzcan situaciones de bloqueo o inanición en la ejecución del código.

Ir hacia la pared (go_to_wall)

En nuestro caso, el comportamiento del robot para buscar una pared es muy sencillo. Se establece una distancia máxima (MAX_DISTANCE) de 30 cm y, mientras el sensor de ultrasonidos devuelva un valor mayor, se irá hacia delante a una velocidad preestablecida (DEFAULT_SPEED). Cuando ese valor sea menor a la distancia máxima, se irá reduciendo la velocidad con la distancia. De esta forma hacemos que el sensor nos proporcione datos más precisos y que otras tareas decidan entrar o no.

Seguir la pared (follow_wall)

El comportamiento de seguimiento de paredes es un poco más complejo. En un primer acercamiento lo desarrollamos de forma que el robot avanzaba hacia delante unos segundos, se paraba y giraba para comprobar que seguía teniendo la pared a su lado, y repetía el proceso. Este comportamiento era muy lento y utilizaba demasiado el giroscopio, por lo que al final almacenábamos pequeños errores de medición que hacían que no fuese muy efectivo.

Para evitar esto, cambiamos el comportamiento tras ver en clase de teoría el comportamiento de seguir paredes del Rug Warrior. En este caso lo que se hacía una vez detectada la pared es recorrerla haciendo arcos, evitando tener que pararse y girar 90º como en el anterior comportamiento.

Ir hacia la luz (light)

Este apartado fue el más difícil de todos, ya que no podíamos probar el comportamiento en el simulador y fuimos a ciegas durante la mayor parte del tiempo. Es por este motivo que nuestro robot hace algo muy sencillo, pero que resulta efectivo en las pruebas reales. Establecemos unos valores predeterminados por constantes para los valores de luminosidad ambiental (LIGHT_FOUND = 40 y LIGHT_DETECTED = 20), que serán la guía para el EV3.

Una vez detecta la luz, seguirá en línea recta mientras no la pierda. El problema en este caso es que el sensor nos da muchas lecturas de 0 o valores muy bajos de vez en cuando, por lo que el robot hace el amago de seguir recto haciendo el movimiento en arco del seguimiento de paredes. Esto no es un problema, porque al final se encuentra con la luz más de cerca y, cuando el valor supera al de luz detectada, para todas las tareas.

Escapar (escape)

El comportamiento para escapar de situaciones donde nos quedamos atascados es muy sencillo. Utilizamos el sensor de ultrasonidos y el de toque para detectar colisiones frontales o situaciones en donde estamos excesivamente cerca de la pared. Si esto sucede se retrocederá un momento y se girará 90º para evitar ese obstáculo.

Elementos a mejorar

Arquitectura subsumida

Nuestro código no utiliza una arquitectura subsumida estándar, ya que en lugar de bloquear las variables de acceso concurrente, bloqueamos las funciones completas. Esto puede causar que si se añaden más tareas el programa deje de funcionar y existan interbloqueos, que la adición de nuevas tareas requiera cambiar mucha parte del código, y que el rendimiento no sea el más adecuado por la no correcta concurrencia de las tareas. No pudimos arreglarlo por falta de tiempo, pero sería tan sencillo como mantener los comportamientos de las tareas usando los semáforos para bloquear y desbloquear variables globales (una por tarea) que permitan ejecutarse o no a las tareas.

Tarea de seguimiento de luz

La tarea de seguimiento de luz no hace un seguimiento activo, si no uno más pasivo. Esto es así por falta de tiempo en pruebas reales y por los extraños valores medidos por el sensor (muchas veces nos daba 0 de luminosidad delante de las fuentes de luz). Se podría mejorar haciendo que se siga de forma activa y no esperando que el seguimiento de paredes nos acerque lo suficiente como para detectar la luz y para la ejecución.

Tarea de buscar una pared

La tarea de buscar o ir hacia una pared la realizamos de forma que circule hacia delante hasta encontrarse una. Otra forma que podría llegar a ser más eficiente es que al iniciarse el programa, gire para buscar alguna. No es un fallo o problema como tal, pero sí digno de mención.