Robótica - Qlearning

- Descripción general y aclaraciones:

Para esta práctica el primer paso ha sido implementar tres acciones dos para girar y una de andar en línea recta, para ello hemos “reciclado” parte del código de la práctica anterior para el movimiento por posición. Se han definido los siguientes elementos:

-MAX_SPEED: define la velocidad de movimiento, es la mitad que la del código de ejemplo.

-ALPHA & GAMMA: las variables del refuerzo, las dejamos ambas en los valores indicados por defecto a 0.5.

-MAX_ITER: establece el número de iteraciones del entrenamiento, lo establecemos en 500. A partir de entonces las acciones serán 100% seleccionadas de la matriz Q. Mientras se entrena las acciones pueden ser aleatorias o no, cuantas más iteraciones de entrenamiento menor es la probabilidad de aleatoriedad.

-TIME_STEP: al igual que en prácticas anteriores lo establecemos en 24.

-MIN_INFRA: valor mínimo de los sensores inferiores del robot, lo establecemos a 500, al igual que el valor proporcionado por defecto, ya que tras comprobarlo nunca se supera si está sobre la línea negra.

-MAX_INFRA: de igual forma tras comprobar cuánto miden los sensores sobre zonas blancas el valor proporcionado es más que suficiente, lo dejamos en 750.

-MIN_INFRA_CHOCAR: valor para evitar chocar, en este caso para los sensores frontales infrarrojos, lo establecemos a 450. (No usamos sensores ultrasónicos para evitar chocarnos ya que, si bien funcionan con el mapa básico sin tener en cuenta las líneas negras, en cuanto conseguimos que el robot se comporte como debe siguiendo la línea los sensores ultrasónicos provocan que este se desvíe cuando no debería al detectar sólo a partir de ciertas distancias.)

-Radio_rueda, radio_entre_ruedas, incr, incr_giro e incr_esquivar: son/se calculan siguiendo las mismas fórmulas proporcionadas para la práctica anterior de odometría. Estas son datos del robot y las de incr_X sirven para calcular la posición de destino del robot en base a lo que miden los sensores de las ruedas.

-Andar_X: es la distancia que recorrerá el robot cada vez que tenga que desplazarse en línea recta, tras probar con diferentes valores el que mejores resultados ha mostrado es 20mm.

-Angulo_de_giro: ángulo que el robot girará a izquierda o derecha cuando corresponda, al igual que en el caso de la distancia tras probar con varios valores un pequeño nos da mejores resultados, en este caso 20 grados.

-mirarSensores: función que guarda las mediciones actuales de los sensores en otras variables para tener el estado anterior y después actualiza los datos con la medición actual. Por

último, consultando los valores de los sensores identificamos en qué estado se encuentra el robot (en la línea, abandonandola por la derecha o por la izquierda).

-maximizar(est): consulta la matriz y determina cual es la mejor acción en base al estado pasado como parámetro.

-siguienteAcción(): elige una acción a realizar esta puede ser aleatoria o la mejor dada por la función anterior, la probabilidad de que esta sea aleatoria decrece cuanto más se entrena. Una vez terminado el entrenamiento todas las acciones dependen de la matriz Q dando comienzo a la etapa de explotación.

-aprender(): función que calcula el refuerzo y lo aplica a la matriz, para el cálculo del refuerzo miramos los sensores tras realizar un movimiento, esto nos indica en las variables x_prev el estado de las mediciones antes de realizar la acción y en las variables sin "_prev" el estado después de realizarla, con esto calculamos el valor de refuerzo. Si antes de la acción un sensor estaba en zona blanca y ahora está en línea sumaremos 1 al refuerzo, si antes estaba en línea y sigue en línea también reforzamos ya que es lo que debe hacer. Si por el contrario se ha salido o no ha entrado restamos al refuerzo 0.25. Estos valores son para los sensores principales, los sensores frontolaterales suman y restan la misma cantidad, 0.25, esta es menor ya que estos sensores los consideramos "menos importantes" desde el punto de vista que al girar, por ejemplo, serán los primeros en detectar la zona blanca y podrían ayudar a rectificar la trayectoria más rápido, pero aunque no lo hagan el robot seguiría en su mayoría dentro de la línea. También se habían intentado otras formas de calcular este refuerzo como haciendo medias sobre el valor resultante de este proceso o unificando varias de las actuales condiciones de los if-else en una sola, entre otras. Todas han sido descartadas por dar resultados dispares haciendo que en ocasiones el robot se quedase parado girando sobre sí mismo al llegar a la etapa de explotación. Una vez calculado el refuerzo simplemente lo aplicamos actualizando los valores de la matriz Q siguiendo la fórmula.

-main(): contiene tres partes diferenciadas dentro del bucle, la primera es la que se calcula la siguiente acción a realizar haciendo uso de dicha función, a continuación calculamos el movimiento mirando las posiciones de los encoders y sumando el incremento correspondiente a la acción a realizar. En el segundo trozo de se realiza el movimiento comprobando si los encoders llegan a la posición marcada tras dos veces sin que el robot se mueva consideramos que ha llegado a su destino, en esta práctica no tenemos en cuenta el error ya que este es muy bajo y dado que no necesitamos una precisión exacta no lo consideramos necesario. Por último, en el tercer trozo, aplicamos la función de aprender si estamos en fase de exploración, actualizamos los sensores y verificamos no chocarnos con nada con una adaptación del código de ejemplo del khepera a movimiento discreto. Para saber si está en la fase de planificar o en la de moverse (ya que para planificar debe haber terminado el movimiento) usamos la variable "moviéndose".

- Resultados:

Los resultados son buenos, tras terminar el entrenamiento y encontrar la línea el robot avanza por ella sin problemas y sin perderse o salirse en ningún momento. La única situación destacable es cuando se "atasca" rebotando entre dos puntos de la línea avanzando y retrocediendo durante un rato, al cabo de unos segundos vuelve a recorrer la línea normalmente. En cuanto al objeto rojo, si lo introducimos en el mapa una vez empezada la etapa de explotación tampoco supone un problema, si el robot se va a chocar simplemente se da la vuelta y recorre la línea en sentido contrario.