Robótica - Arquitecturas Subsumidas

- Descripción general y aclaraciones:

Al tener que implementar cuatro comportamientos bien diferenciados hemos decidido hacer cuatro funciones principales y tres auxiliares:

-behabiour01: Buscar pared, el robot anda en línea recta buscando una pared este comportamiento se activa cuando ninguno de los otros está activado y cuando además los sensores frontales y laterales del robot no detectan una pared cercana.

-behabiour02: Seguir pared, cuando el robot está posicionado con una pared a su izquierda haciendo uso de los sensores laterales izquierdos tanto ultrasónicos (para no alejarse demasiado) como infrarrojos (para mantenerse siguiendo la pared y no acercarse demasiado) sigue la pared y toma curvas a la izquierda.

-behabiour03: En esta función se controla la cámara, haciendo uso de una función auxiliar “process_image” que nos indica los niveles de color verde en tres zonas de la imagen (izquierda, medio, derecha) esta función compara los valores y cuando se detecta el objeto toma el control para ir hacia el, también en esta función si el objeto está cerca se le indica al robot que se pare y se usa también para reactivarlo si se ha quedado atascado.

-behabiour04: Esquivar, cuando los sensores infrarrojos detectan que nos acercamos demasiado a un objeto/pared, nos alejan de este priorizando girar a la derecha, esta función sirve también para hacer los giros a la derecha ya que cuando llegamos a una esquina siguiendo una pared por la izquierda sino nos chocaremos. En este comportamiento también está el de separarse de la pared de la izquierda si nos hemos acercado demasiado en el comportamiento 2.

-init_devices: función que se encarga de inicializar el robot y los sensores.

-change: función para bloquear y cambiar el valor de las variables globales compartidas por todos los hilos y donde se lee y escribe qué comportamiento toma el control.

-process_image: función que recibe la imagen de la cámara, hace una copia, la analiza y devuelve cuatro valores (tres de cantidad de color y uno con una media de color en una región de la imagen que se usara para “desatascar” al robot) que después se procesarán en behaviour03.

Cada uno de los behaviours está en un hilo de ejecución distinto, estos se ejecutan de forma continua hasta que le indiquemos al robot que pare o se detecte el objeto verde en la cámara muy cerca. Para el control de los distintos comportamientos hemos optado por una variable a la que todos los hilos tienen acceso de lectura escritura y un “lock” para controlar tanto el acceso a esta como a las variables de actualización de la velocidad de los motores.

En un bucle principal se actualizan una serie de variables globales que serán los valores de los sensores cada vez que sincronizamos el robot (con un time step de 24). Los distintos hilos consultarán estas variables y actuarán en consecuencia cambiando dos variables LEFT y RIGHT que marcan la velocidad con la que giran las respectivas ruedas del robot. Estas variables solo se leen en el bucle principal donde se actualizan las velocidades del robot.

Para  evitar que el robot se quede atascado en ciertas situaciones se ha implementado, en el hilo que usa la cámara, que cuando la imagen no cambia en 3 pasadas consecutivas (el timeStep de la cámara es mayor que el del robot para evitar sobrecargar el rendimiento) se muevan ligeramente y a la mínima velocidad las ruedas hacia la derecha, moviendo una ligeramente más que la otra, de esta forma otros comportamientos podrán empezar a funcionar otra vez al recibir nuevas lecturas de los sensores.

Para evitar que tarde demasiado el procesamiento de la imagen esta no se procesa completa solo se procesa una ventana central de toda la altura de la imagen y, eso si, todo el ancho. El ancho se parte en 3 zonas para controlar el giro del robot, dos laterales a izquierda y derecha y una central que en nuestro caso es más pequeña que indica al robot ir en línea recta, las zonas laterales son mayores ya que hemos comprobado que por la distorsión del ojo de pez de la cámara funciona mejor de esta forma.
Cada uno de los píxeles de este recorte de la imagen se mira si se corresponde a color verde, se mira con un cierto margen de error ya que debido a la iluminación los valores no son exactos, los márgenes utilizados son <40 para el azul, < 50 para el rojo y >170 para el verde. El robot girará en la dirección donde el valor sea más alto, siempre que dicho valor sea mayor a un determinado porcentaje de la ventana correspondiente. Esto lo hacemos porque debido a la iluminación saltaban falsos positivos.
Para que cada proceso sepa el nivel de prioridad que tiene y este sea fácil de modificar si es necesario en futuro, se decidió por hacer una implementación en la cual se le pasa por parámetro la prioridad a este a la hora de crear el hilo que lo va a implementar.
La forma que decidimos implementar para que cada proceso actuase cuando su nivel de prioridad lo permitiese es la siguiente, creamos una función auxiliar llamada change, la cual es la encargada de gestionar los niveles y utilizar los datos de la función que tiene la prioridad de actuar en cada momento.

Cuando un comportamiento no cumple con las condiciones de activación libera la variable de control para que otro de menor nivel pueda iniciarse.

- Problemas conocidos:

-Cuando el robot se pega mucho a la pared a veces no le da tiempo a girar y los sensores “atraviesan” la pared, por lo que en vez de medir un valor muy alto (infrarrojos) pasan a no medir nada, como si no hubiese pared. Esto evita que el robot pueda separarse en ciertas situaciones. Se ha implementado una opción dentro del comportamiento 4 “esquivar” que activa la marcha atrás del robot a velocidad “-1” para cuando esto ocurre con los sensores frontales el funcionamiento no se vea interrumpido.

-Cuando llega a la caja, hay veces que tarda en parar, esto se debe al ángulo de visión que tiene en la foto que está procesando de la caja, si el ángulo no es muy bueno este puede tardar en detectar y parar mal o no parar, es un fallo poco frecuente.

-Cuando el robot se para cerca de la caja algunas veces se queda con el último comportamiento activado pese a no estar funcionando ya. Esto provoca que se quede girando en círculos cerca de la caja o sobre sí mismo. Solo ocurre algunas veces y creemos que se debe a la “desconexión” del robot y el controlador.

-Algunas veces se separa demasiado de la pared y no consigue regresar a ella activándose el comportamiento de descubrir pared nuevamente.

-Algunas veces cuando se termina una pared y debe hacer el giro a la izquierda para continuar pegado se separa demasiado debido al sensor ultrasónico y no es capaz de regresar. Algunas veces otro de los comportamientos se activa y, ocasionalmente (en el mundo 3 casi al final), se queda girando en círculos, si se reinicia la simulación puede volver a funcionar con normalidad.

Hemos realizado 6 simulaciones en cada uno de los mapas para comprobar estos errores y los resultados son:

-En el mundo uno, las seis veces ha completado el recorrido con éxito a excepción de un par de veces que se ha quedado girando al lado de la caja al desconectar el controlador.

-En el mundo dos, de las seis veces en una se alejó demasiado de la pared en un par de ocasiones pero consiguió volver a ella sin inconvenientes, en otra ocasión tras completar el recorrido con éxito no se paró junto a la caja ya que se acercó a ella con un mal ángulo de cámara.

-En el mundo tres, nuevamente en un par de ocasiones se alejó demasiado de la pared consiguiendo volver a ella tras dar un par de vueltas sobre sí mismo, en otras 3 ocasiones y casi al final del recorrido (la segunda parte de la pared con huecos) no fue capaz de regresar a esta.