Proyecto de Robótica Inteligente y Sistemas Autónomos: *Robobo Explorador*

Autores

Alejandro González Norat

José J. Rodríguez Salgado

[Universidade A Coruña]

ENERO 21, 2022

Resumen

En el presente documento se describe la implementación de un robot explorador que se basa en una arquitectura subsumida. El robot explora un mundo donde puede encontrar obstáculos y donde va en busca de un objetivo. Una vez detectado el objetivo se dirige hacia él aunque continúa evitando obstáculos. El diseño propuesto se ha probado en laberintos donde el robot es capaz de llegar al objetivo final.

1. Diseño del software

La solución cuenta con varios componentes que se organizan de acuerdo a su propósito y nivel de abstracción. Todo el código fuente se ubica dentro del directorio *src* de la solución que se entregó

- **Módulo *constants*:** Aquí se definen todas las constantes y configuraciones de la solución. Desde la dirección IP para conectarse al Robobo pasando por las configuraciones de movimientos y de sensores.

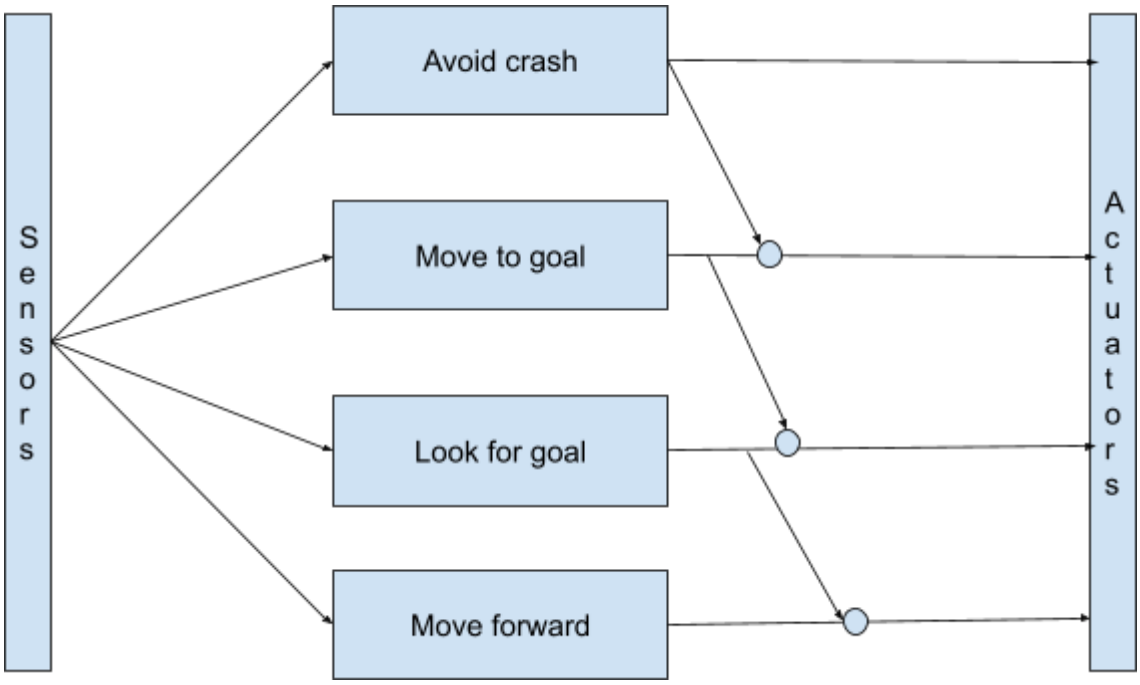
- **Módulo *robobo*:** El objetivo de este módulo es facilitar la interacción directa con el Robobo. Hay funciones de movimiento, relacionadas con la visión, con las emociones, etc. No necesariamente se utilizan todas las funciones en la solución final. Con este módulo, las órdenes al Robobo son más expresivas, entendibles y fáciles de usar. Por ejemplo, si se quiere girar a la derecha, en lugar de usar el método *moveWheels* del Robobo, se puede simplemente usar el método *move_right* del módulo **robobo.movement.simple_movements**
- **Módulo *utils*:** Aquí se definen varias funciones que se usan en varios lugares de la solución como las de determinar si un obstáculo está cerca.
- **Módulo *subsumed_architecture:*** Donde se define toda la implementación de la arquitectura cognitiva que rige el comportamiento del Robobo**.** Cuenta con los submódulos **behaviors,** donde se encuentran las definiciones de los diferentes comportamientos que puede ejecutar el robot, y **states** donde se definen los estados por los que pasa la máquina de estados de la arquitectura. Además se tienen los archivos **augmented_fsm** donde se define la máquina de estados, y **transitions** donde se definen las transiciones de dicha máquina

La solución cuenta además con los archivos **main** que es el que se debe ejecutar para iniciar los movimientos del Robobo, y los archivos **returning** y **step_logging** que contienen la implementación de un intento de que el Robobo regresara sobre sus pasos y de los que se habla más adelante

2. Definición teórica de la arquitectura

Se trata de una arquitectura subsumida con cuatro comportamientos esenciales. El más prioritario es el de **evitar obstáculos**, luego el de **moverse hacia el objetivo**, luego el de **buscar el objetivo**, y por último el de **moverse hacia adelante.**

El siguiente diagrama muestra la arquitectura descrita:



A no ser que sea subsumido por un comportamiento más prioritario, cada comportamiento se continúa ejecutando, de manera que varios comportamientos se ejecutan de forma paralela.

3. Implementación de la arquitectura

En la solución, se define un estado (**state**) como una clase que implementa un determinado comportamiento (**behavior**). Además los estados cuentan con acceso a una información que se comparte entre todos los estados del autómata. Esta información puede ser, por ejemplo, en qué posición se encuentra el objetivo, cuántos grados había girado la cámara cuando detectó el objetivo, etc.

Las transiciones entre estados se definen en el archivo **transitions** y son independientes del código de los estados y de la máquina de estados misma.

El estado final del autómata es cuando alcanza el objetivo. Al acercarse lo suficiente al objetivo, el Robobo se detiene y termina el programa.

Se intentó implementar un algoritmo alternativo para regresar al punto inicial una vez que se alcanzara el objetivo. Se trató de realizar esta tarea volviendo sobre los propios pasos, para eso se usó el código de los archivos **step_logging** y **returning.** En el primero de ellos se crearon las funcionalidades para que cada paso que se dió quedara salvado y así poder regresar. En el archivo **returning** se implementó el regreso siguiendo los pasos salvados. En la gran mayoría de los entornos en los que se probó la solución, el regreso no se realizó

con éxito por lo que se decidió no incluirlo en la solución final. No obstante, se adjunta un video en la entrega donde se logró un resultado satisfactorio. Además se puede consultar el código en los archivos antes mencionados.

4. Resultados

El objetivo fue crear un robot que explorara un mundo hasta encontrar un objeto de un determinado color. Entre los escenarios de los que se dispone en el simulador, uno de los más retadores para este tipo de robot es el laberinto. Pero debe entenderse que el robot no está programado para resolver un laberinto, sino para explorar un entorno de manera más libre. Es por eso que no se incluye un seguimiento de paredes, por ejemplo.

El robot es capaz de encontrar el objetivo en todos los diferentes laberintos que se presentan en el simulador, además de resolver escenarios en el mundo real, como se pudo comprobar en los laboratorios con un Robobo real.

Como ya se mencionó, la implementación de un algoritmo complementario de forma que permitiera regresar sobre sus pasos al Robobo no fue del todo satisfactoria y se decidió excluirla de la entrega final, aunque el código implementado se puede consultar.

El código resultante es fácilmente extensible. Agregar un nuevo comportamiento consiste solamente en definir dicho comportamiento, los estados que lo implementen y las transiciones de esos estados, sin necesidad de modificar el código de la máquina de estados ni otras partes del software.

Todo el código de la solución es público y está disponible en Github. En el archivo **README.md** se encuentran las instrucciones para utilizar el programa.

Conclusiones

En el presente trabajo se implementó una arquitectura subsumida para permitir que un robot fuera capaz de explorar un mundo en busca de un objetivo. Las simulaciones tanto virtuales como con el Robobo real, evidenciaron que la propuesta cumple con este propósito, mientras que la versión híbrida donde se implementa un algoritmo para regresar al punto inicial no es satisfactoria. El código presentado se pensó para que fuera fácilmente extensible, modificable y reusable.