Laboratorio de Robots Móviles

Práctica No. 2

Comportamientos Reactivos Usando Máquinas de Estados

Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE100821

Objetivo:

- Familiarizar al alumno con el middleware ROS.
- Familiarizar al alumno con los comportamientos reactivos.

Duración: Tres semanas

1. Sí no tiene instalado python en su computadora, instálelo de la siguiente forma:

$sudo apt-get update

$sudo apt-get install python2.7

2.- Descargue de la página: https://biorobotics.fi-p.unam.mx/courses/robots-moviles/

el archivo denominado PYCLIPS.zip, el cual contiene el sistema para usar el sistema basado en reglas CLIPS.

Descomprimir el archivo en /home/<usuario> y colocarse en su directorio con $ cd ~/PYCLIPS

En este directorio hay dos directorios: instPy y pyclips.

Para instalar el software de instPy colocarse su directorio y ejecutar los siguientes comandos:

$sudo python setup.py build

$sudo python setup.py install

Para instalar el software de pyclips colocarse su directorio y ejecutar los siguientes comandos:

$sudo python setup.py build

$sudo python setup.py install

3.- Descargue de la página: https://biorobotics.fi-p.unam.mx/courses/robots-moviles/ en la pestaña Material de apoyo el archivo denominado MobileRobotSimulator1.zip el cual se encuentra en la pestaña Simulador Final. El cual contiene el sistema gráfico que se utilizará para simular el robot móvil usando ROS, código en C++, Python y CLIPS. Descomprimir el archivo en /home/<usuario>, antes de hacer esto deberá asegurarse de que no exista algún folder llamado catkin_ws, si existiera, cambiar su nombre, posteriormente colocarse en el directorio *catkin_ws con:*

$ cd ~/catkin_ws

También puede clonarlo desde el repositorio de github : https://github.com/dieg4231/MobileRobotSimulator

3.1.- Compilar con:

$ catkin_make

Si marca errores de compilación por una ruta no existente, se corrige con el siguiente comando:

$ sudo rm build/ devel/ -R

Volver a compilar con **catkin_make**

3.2.- Si la compilación fue exitosa dar permisos de ejecución al archivo start_ros.sh e install.sh, si se necesitase, con:

$ chmod +x start_ros.sh

$ chmod +x install.sh

Y ejecute el script install.sh con el comando:

$ ./install.sh

3.3.- Iniciar el sistema con

$ ./start_ros.sh

Aquí se abrirán 7 terminales X las cuales contienen los siguientes nodos:

1. El nodo *roscore* se encarga de establecer la interconexión entre los diferentes modulos.

2. El nodo *simulator_node* contiene el GUI del sistema, en donde se introducen parametros para la ejecución de los algoritmos y se muestra la operación del robot.

3. El nodo *light_node* contiene la simulación de una fuente luminosa.

4. El nodo *laser_node* contiene la simulación de un sensor laser, éste se puede configurar en el GUI del sistema indicando el número de sensores, el inicio de éstos a partir del centro del robot y su rango.

5. El nodo *base_node* contiene la simulación de la base del robot móvil.

6. El nodo *motion_planner_node* contiene los diferentes algoritmos que ejecuta el robot móvil.

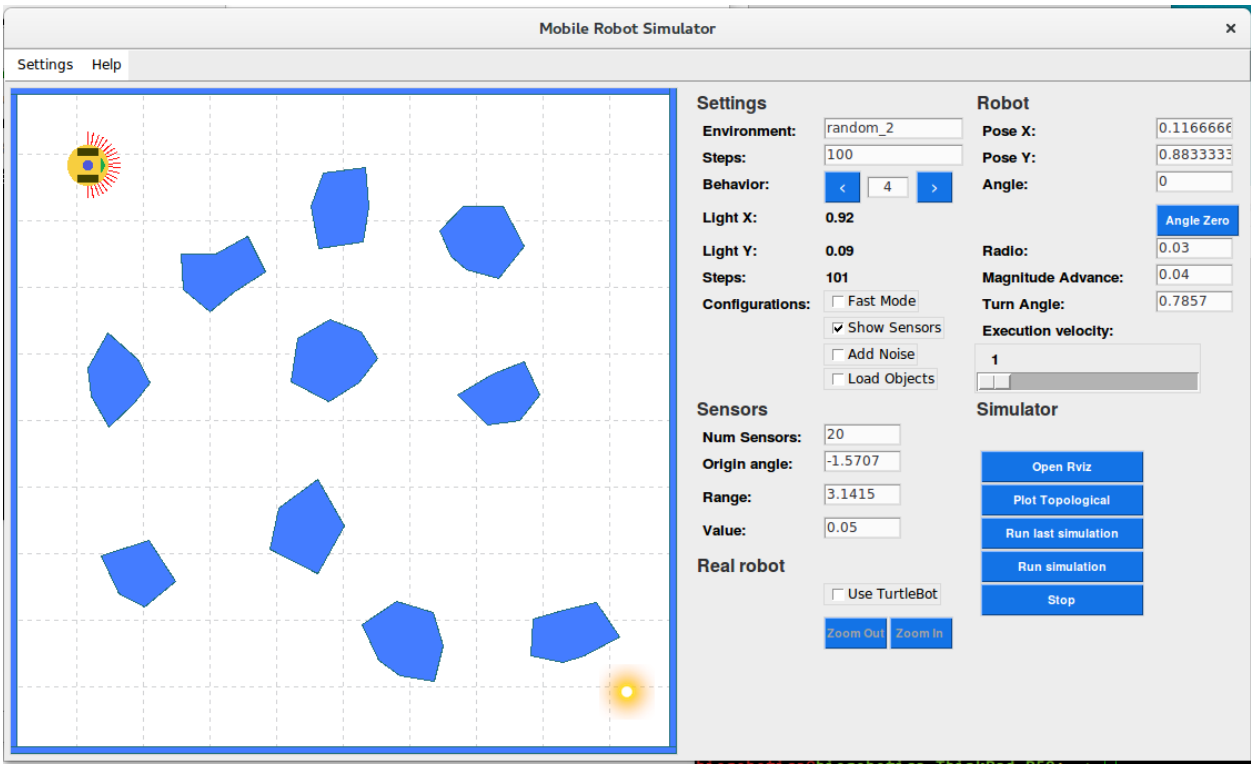
7. El nodo *ros_pyclips_node* contiene las reglas del sistema experto CLIPS para controlar la operación del robot.

O también puede iniciar el sistema en una sola ventana de terminal con los siguientes comandos dentro de la carpeta catkin_ws:

$ source devel/setup.bash

$roslaunch simulator simulator.launch

En la interfaz gráfica seleccione el comportamiento deseado del robot en Behavior: con un 1 se selecciona una función que solamente se dirige a una fuente luminosa; con un 2 selecciona el algoritmo de una máquina de estados que se dirige a una fuente luminosa; con 3 selecciona un algoritmo el algoritmo de una máquina de estados que evade obstáculos; con un 4 selecciona un algoritmo que se dirige a una fuente luminosa evadiendo obstáculos; con un 5 se selecciona el comportamiento que se dirige a una fuente luminosa evadiendo obstáculos usando lógica de primer orden;



Con el botón izquierdo del mouse seleccione la posición inicial del robot y con el derecho la final, después observe el comportamiento del robot. Seleccionando de nuevo el botón derecho del mouse el robot tomará su última posición como origen y el destino la posición del botón derecho. Las coordenadas (0,0) del mapa se encuentran en el lado inferior izquierdo de la figura. Familiarícese con el funcionamiento de la interfaz gráfica con diferentes configuraciones y comportamientos ejecutados por el nodo *motion_planner_ROS*.

La interfaz gráfica muestra el resultado de la simulación del código en:

~/catkin_ws/src/simulator/src/motion_planner/motion_planner_node.cpp

Entienda el funcionamiento del código motion_planner_node.cpp así como el de las máquinas de estados,

que se encuentran en:

~/catkin_ws/src/simulator/src/state_machines

En el apéndice A se muestra el mapa simbólico en donde el robot navega, los objetos se representan con polígonos. Este tipo de archivos tienen terminación *.wrl y se encuentran en el directorio, en sus respectivos directorios:

~/catkin_ws/src/simulator/src/data

4. Modifique el código de ~/catkin_ws/src/simulator/src/state_machines/user_sm.h para que el robot tenga un comportamiento que se dirija a una fuente luminosa y que cuando encuentre un obstáculo lo rodee, utilizando el algoritmo Bug 0. Pruebe primero su algoritmo con el medio ambiente bug, después con cualquiera de los ambientes denominados randoms. Pruebe su algoritmo ya sea sin ruido o con ruido en el movimiento y sensado del robot, para agregar ruido a la simulación seleccione el check box **Add Noise .**

Para compilar su código hacer lo siguiente:

Colocarse en el directorio ~/catkin_ws con :

$ cd ~/catkin_ws

Compilar de nuevo el proyecto con:

$ catkin_make

aquí solo se compilarán los últimos archivos modificados.

Seleccione la ejecución de su código con el comportamiento 8 en la interfaz gráfica.

5. Repita el inciso cuatro con los algoritmos Bug 1 y Bug 2.

APENDICE A

Mapa Simbólico room.wrl

En este apéndice se muestra las siguientes directivas de la representación de los mapas simbólicos:

1. Se pueden incluir comentarios o comentar una línea si se coloca un ";" al principio del renglón.

;* File: room.wrl *

2. "dimensions" indica las dimensiones del medio ambiente, las cuales están indicadas en el sistema métrico decimal. En el siguiente ejemplo se presentan las dimensiones del medio ambiente "room" de 1m x 1m:

(dimensions room 1.000 1.000)

3. "polygon" indica los vértices de un polígono, se indica si es de tipo obstáculo o pared, después viene el nombre del obstáculo y a continuación los vértices. Los vértices de los polígonos se indican con las coordenadas Xi y Yi, ordenados hacía el sentido horario de las manecillas del reloj:

(polygon obstacle obs1 .40 .55 .60 .55 .60 .35 .40 .35)

(polygon wall wall1 0.0 0.0 0.0 1.0 0.01 1.0 0.01 0.0)

room.wrl

```
; ***************************************************************
; * File: room.wrl                                              *
; * Definition of the forbidden and allowed areas in the Robot's *
; * world. These areas are derivated from the objects in the     *
; * world.                                                      *
;***************************************************************
( dimensions room 1.000 1.000 )
```

(polygon obstacle obs1 .40 .55 .60 .55 .60 .35 .40 .35)

(polygon wall wall1 0.0 0.0 0.0 1.0 0.01 1.0 0.01 0.0)

(polygon wall wall2 0.0 0.99 0.0 1.0 1.0 1.0 1.0 0.99)

(polygon wall wall2 0.99 1.0 1.0 1.0 1.0 0.0 0.99 0.0)

(polygon wall wall2 1.0 0.01 1.0 0.0 0.0 0.0 0.0 0.01)