Aprendizaje por Refuerzo en Robótica
Departamento de Ingeniería Informática
Instituto Tecnológico de Buenos Aires

Trabajo Práctico 2

Objetivo

Familiarizarse con el uso de los sensores y actuadores del robot e-puck.

1. Introducción

El objetivo es implementar controladores simples que utilicen varios de los sensores y actuadores del robot e-puck: los sensores de distancia, los acelerómetros, los sensores de luz, los emisores y receptores de sonido, los LEDs, la cámara y el sensor de piso. Asimismo, aplicar un método de odometría sencillo, según lo visto en las clases teóricas, y estudiar la simulación de la física de obstáculos.

1. Para empezar, crear un nuevo proyecto llamado "arrtp2" seleccionando "New Project Directory..." en el menú "Wizard".

2. Copiar $WEBOTS_HOME/projects/robots/e-puck/worlds/e-puck.wbt al directorio worlds del proyecto (donde WEBOTS_HOME corresponde al directorio donde fue instalado el Webots)

2. Física de obstáculos y odometría básica

1. Copiar el directorio del controlador $WEBOTS_HOME/projects/robots/e-puck/controllers/e-puck al directorio controllers del proyecto. Asignarlo como controlador del robot en el árbol de escena (atributo controller del nodo DifferentialWheels).

2. Analizar el método de odometría implementado en el código fuente del controlador. Identificar la función se computa la distancia cubierta por ambas ruedas y el cambio de orientación del robot y explicar cómo funciona. De ser necesario, modificar el código para imprimir los valores por salida estándar.

NOTA: Incluir en el informe de laboratorio las modificaciones realizadas al código fuente y la discusión de resultados.

3. Modificar el controlador para que el robot no se mueva (velocidad nula) manteniendo las impresiones por salida estándar. Acercar objetos del ambiente al robot para observar cómo varía la lectura de los sensores de proximidad a distintas distancias.

Luego, ubicar un objeto a una distancia media y alterar sus propiedades físicas en el árbol de escena para probar si varían los valores leídos por los sensores. Indicar cuál es el parámetro físico del objeto que altera la medición y cómo.

NOTA: Para mayor información sobre los sensores de distancia, referirse a la sección 3.16 (DistanceSensor) del Webots Reference Manual.

NOTA: Haciendo doble clic sobre el e-puck se puede ver en una ventana aparte el detalle de las mediciones de todos los sensores a cada instante.

4. Modificar el código fuente una vez más para adaptar el método de odometría implementado al visto en las clases teóricas. Computar la posición del robot después de cada lectura de los encoders de las ruedas e imprimir los valores por salida estándar.

NOTA: Para mayor información sobre los encoders de las ruedas, referirse a la sección 3.13 (DifferentialWheels) del Webots Reference Manual.

3. Acelerómetros

1. Hacer una copia del controlador anterior llamada e-puck_accelerometer y asignarlo al robot simulado.

2. Modificar el controlador para generar una muestra de 1000 mediciones con las lecturas de los 3 acelerómetros y las velocidades de ambas ruedas. (Puede imprimirse a un archivo en un formato de tipo CSV o a salida estándar para copiar y pegar.) Los datos deben generarse en un contexto en el que el robot gire, es decir, que no se mueva solamente a velocidad constante. Guardar este controlador en un archivo aparte.

3. Graficar la aceleración del robot en las 3 direcciones y la velocidad de las ruedas en función del tiempo. (Pueden utilizarse planillas de cálculo o utilitarios como Matlab y GNU Octave.) Explicar los resultados obtenidos.

NOTA: Para mayor información sobre los acelerómetros, referirse a la sección 3.1 (Accelerometer) del Webots Reference Manual.

4. Sensores de luz

1. Copiar al proyecto el controlador de $WEBOTS_HOME/projects/samples/devices/controllers/light_sensor y asignarlo al robot simulado.

2. Fijar la intensidad de los puntos de iluminación (nodos PointLight) a 0.12. Crear un nuevo punto de iluminación en una de las esquinas del ambiente con intensidad 4.

3. Analizar en el código fuente cómo se computan las velocidades de las ruedas dependiendo de la proximidad a la fuente de luz. Modificar el método de cálculo de las velocidades tomando en cuenta los 4 sensores de luz delanteros del robot (ls6, ls7, ls0, ls1). Considerar el máximos valor medido por el sensor de luz (4095).

NOTA: Para mayor información sobre los sensores de luz, referirse a la sección 3.30 (LightSensor) del Webots Reference Manual.

4. Probar la reacción del robot frente a distintas intensidades de las fuentes de luz y distintas ubicaciones. Indicar cómo se podrían relacionar los sensores de luz con las ruedas para que el robot gire alrededor de la luz.

5. Sonido y LEDs

NOTA: Para esta sección, es necesario cambiar la escena (world) por la del siguiente archivo: $WEBOTS_HOME/projects/robots/e-puck/worlds/e-puck_sound.wbt.

5. Copiar al proyecto el controlador de $WEBOTS_HOME/projects/robots/e-puck/controllers/e-puck_sound y asignarlo al robot simulado.

6. Duplicar en el árbol de la escena el nodo del e-puck para que hayan dos robots en el ambiente. Asignarles roles de "speaker" y "microphone" utilizando el primer argumento pasado al controlador (atributo controllerArgs del nodo DifferentialWheels).

7. Analizar el código fuente del controlador. El robot de rol SPEAKER sólo debe emitir un sonido mientras se mueve. El robot de rol MICROPHONE debe recibir una señal por cada micrófono y analizar sus componentes para determinar si es un sonido o solamente ruido (puede ser comparando la suma de los valores de los componentes con un umbral). El e-puck posee 3 micrófonos: uno a la izquierda (mic1), otro a la derecha (mic0) y otro atrás (mic2). Agregar al controlador una salida que indique si se recibe sonido para cada micrófono.

NOTA: Para mayor información sobre los emisores y receptores de sonido, referirse a las secciones 3.18 (Emitter) y 3.35 (Receiver) del Webots Reference Manual.

8. Modificar el controlador para que se encienda el LED más cercano al micrófono que recibe mayor señal sonora (derecha: led2, izquierda: led6, atrás: led4).

NOTA: Para mayor información sobre los LEDs, referirse a la sección 3.28 (LED) del Webots Reference Manual.

6. Cámara

1. Copiar al proyecto el controlador de $WEBOTS_HOME/projects/samples/devices/controllers/camera y asignarlo al robot simulado.

2. Cambiar los atributos de emissiveColor y specularColor del obstáculo con forma de cubo a rojo y eliminar su textura. El objetivo es que, valiéndose de la cámara, el robot pueda "sacarle una foto" al objeto rojo (guardar la imagen capturada en un archivo) cada vez que lo detecte.

3. Implementar la función take_snapshot(WbDevice camera, int width, int height, char* filename) que dada la cámara del robot, las dimensiones y el nombre de archivo de la "foto instantánea", guarde la imagen capturada en formato PNG.

NOTA: Para mayor información sobre la cámara, referirse a la sección 3.5 (Camera) del Webots Reference Manual.

4. Utilizar dicha función en el ciclo principal de la simulación para capturar imágenes cada vez que se detecta un objeto rojo. Esto se puede determinar en base a los valores de las variables red, green, blue (líneas 63 a 71 del código original) con una condición como la siguiente:
   red >= 4000000 && green < 2000000 && blue < 2000000
   Los umbrales pueden variar dependiendo del ambiente. Imprimir por salida estándar que se ha detectado un objeto rojo cada vez que se cumpla la condición. Agregar también una pequeña pausa (20 ms) cuando se detecta el objeto y explicar para qué sirve.

5. Visualizar las imágenes capturadas y discutir los resultados.

7. Sensor de piso

1. Copiar el ambiente de $WEBOTS_HOME/projects/robots/e-puck/worlds/e-puck_line/e-puck_line.wbt y el controlador del mismo nombre al proyecto local.

2. Analizar el controlador. Notar que el sensor de piso del robot e-puck se modela como un DistanceSensor. Probar la simulación ubicando al robot fuera y dentro del camino marcado o con obstáculos en medio del camino. Explicar la máquina de estados en la que se basa el controlador, detallando estados y transiciones.

3. Un caso particular se da cuando el robot se encuentra avanzando en forma perpendicular a la línea del camino. Corregir la máquina de estados según corresponda para que el robot pueda seguir el camino correctamente, incluso en este caso.