Universidad Rey Juan Carlos

Práctica 1: Dinámicas en Pybullet

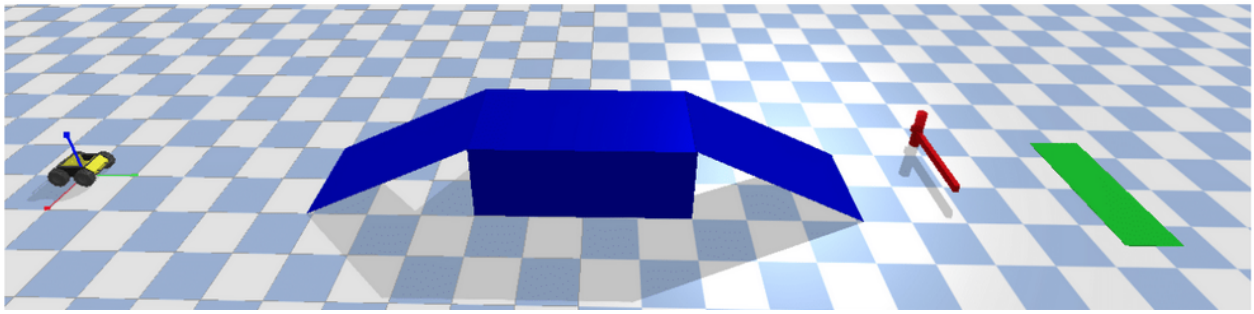
Modelado y Simulación de Robots

En nuestra primera práctica, vamos a diseñar y construir un entorno interactivo dentro del simulador pybullet. Este entorno estará diseñado específicamente para probar y observar las capacidades dinámicas de un robot, el Husky, que deberá operar de manera completamente autónoma, sin necesidad de ser controlado por nosotros directamente.
Los principales objetivos de esta actividad son:

- Manejo de pybullet
- La creación y definición de partes del escenario utilizando el formato URDF
- Implementación de comportamientos dinámicos en un robot
- Evaluación y análisis del desempeño del robot dentro de este entorno dinámico que hemos configurado

Fase 1: Modelado del Escenario

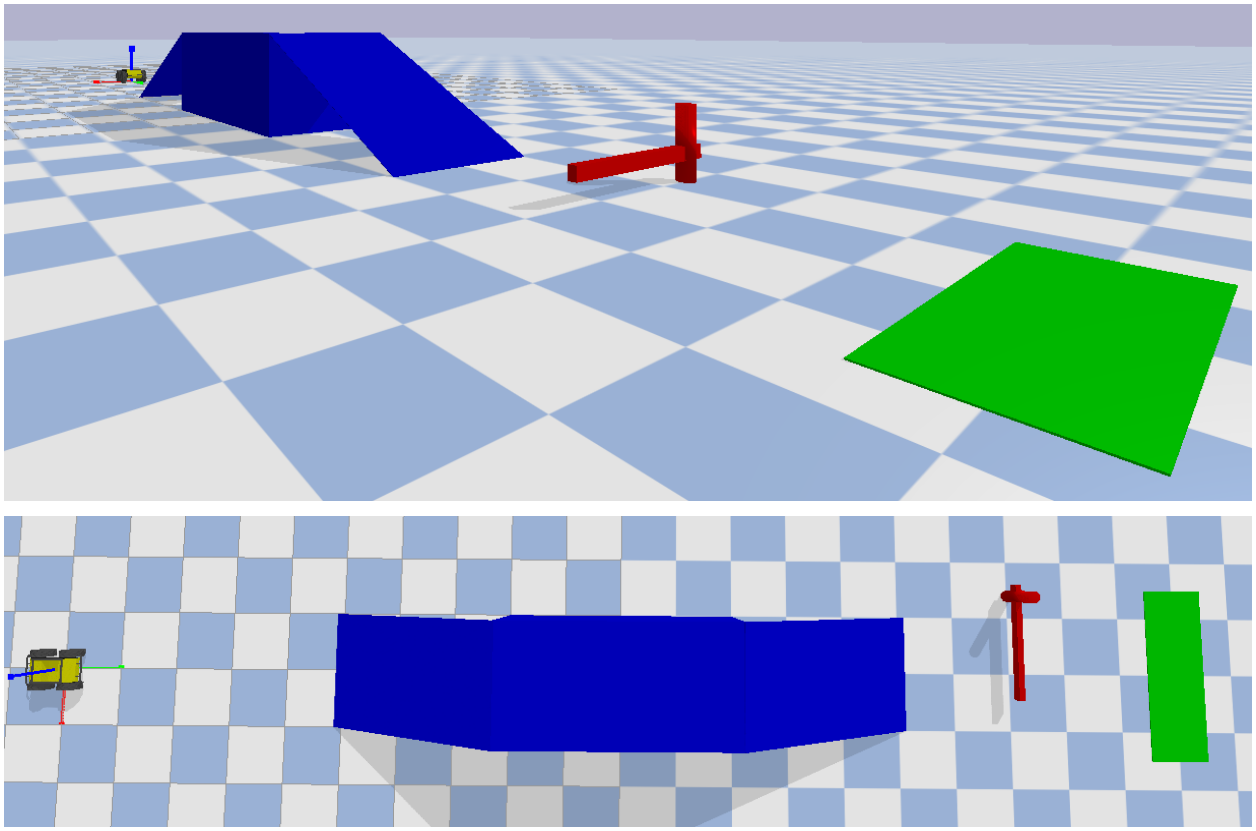
Genera el siguiente escenario 3D modelado en diferentes URDFs. Utiliza los joints y links que consideres oportunos.



El escenario debe contener:

- El robot husky (incluido dentro de los datos de pybullet, "husky/husky.urdf")
- Una estructura que provea una rampa de subida y una de bajada (elemento azul). La altura máxima de este bloque es de 1.5 metros.
- Una estructura que simula una barrera compuesta de un cilindro estático a modo de poste y una barrera móvil (peso 5 Kg), que únicamente permite el giro sobre su eje Z (elemento rojo). La barrera móvil debe estar a la altura del parachoques delantero del Husky.
- Una estructura plana de muy poca altura, que define el final del recorrido (elemento verde)

A continuación tienes más capturas del escenario para que lo repliques con la máxima exactitud posible. El husky se sitúa en el origen de coordenadas. Ten en cuenta que cada cuadrado del suelo se corresponde a 1x1 metros/unidades. Por tanto la rampa empieza en 5 y termina en 15, la barrera está en 17 y la llegada en 20 (aproximadamente).



- En esta primera fase debes implementar un comportamiento autónomo sencillo que haga recorrer al husky el escenario y llegar hasta el elemento verde que define el fin de la evaluación de nuestro test.
- Para ello debes comandar velocidades a cada joint que permita mover el propio robot.
- Para esta fase todos los joint referentes a las ruedas del husky deben tener la misma configuración de velocidad y fuerza motor.
- No debes teleoperar el robot, solo ajustar las velocidades.
- En esta fase NO configures la inercia de la barrera, ni coeficiente de rozamiento de ningún elemento.
- El tiempo máximo para recorrer el escenario completo es de **30 segundos.**

Fase 2: Obtención de métricas

Después de haber configurado el escenario en la anterior fase, procederemos a recopilar datos cuantitativos sobre el comportamiento del robot, con un enfoque particular en su velocidad a lo largo del recorrido completo. Para realizar este análisis de manera precisa, es necesario que recojas y registres cierta información cada **0.01 metros** que el robot avance.

tiempo, posicion_robot[Y], velocidad_robot[Y], velocidad_ruedas, fuerza_ruedas

Cuando el robot husky esté sobre el fin del escenario de evaluación (elemento verde), la ejecución debe pararse, los datos deben guardarse en un fichero en el formato CSV y salir de pybullet.

Se recomienda consultar el manual en línea de pybullet, para obtener información sobre ciertas funciones (getBaseVelocity, *getBasePositionAndOrientation*, …)

Importante: Para esta nueva fase y las siguientes se ha de establecer el modo realtime del simulador, mediante el siguiente comando y configurar el bucle de control adecuadamente.

p.setRealTimeSimulation(1)

Vamos a crear una nueva aplicación en Python, denominada **plot_results.py,** que estará diseñada para procesar datos almacenados en los archivos CSV. Esta aplicación tendrá la funcionalidad de generar un gráfico que exhiba de manera clara y precisa la variación de la velocidad del robot en función de su posición. Específicamente, la posición del robot se mostrará en el eje X, mientras que su velocidad se representará en el eje Y. Es crucial asegurarse de que el gráfico se presente de manera adecuada, incluyendo las etiquetas correspondientes para cada eje y especificando las unidades de medida utilizadas.

Analiza las gráficas resultantes y asegúrate que tienen sentido.

Se recomienda el uso de numpy para el tratamiento de los datos y matplotlib para la visualización de la gráfica.

- Numpy: https://numpy.org/doc/stable/user/quickstart.html
- Matplotlib: https://matplotlib.org/stable/tutorials/introductory/usage.html

Fase 3: Evaluación de métricas

Esta tercera fase tiene como objetivo evaluar el comportamiento del husky en diferentes configuraciones del escenario.

Escenario 3.1: Asignación velocidades y fuerzas

Durante esta fase, nuestro objetivo es llevar a cabo un análisis cuantitativo de varias dinámicas del robot a lo largo de su trayectoria. Para ello establece la velocidad de las ruedas a 11 m/s y la fuerza a 25 N. Ten en cuenta que para todos los escenarios de la Fase3 debes utilizar la misma velocidad y fuerzas, por lo que está permitido cambiar estos parámetros si fuera necesario. Asegúrate de que, con los ajustes realizados, el husky alcance una velocidad aproximada de 2 m/s en el primer tramo del recorrido, antes de llegar a la rampa.

Obtiene el análisis de la velocidad del robot en función de su posición y ejecuta plot_results.py con la configuración anterior asegurándote que el husky termina el recorrido correctamente.

Analiza las gráficas resultantes y asegúrate que tienen sentido.

Escenario 3.2: Asignación velocidades y fuerzas + fricción

Ahora añade la siguiente configuración de fricción a las ruedas del husky:

- lateralFriction=0.93
- spinningFriction=0.005
- rollingFriction=0.003

Obtiene el análisis de la velocidad del robot en función de su posición y ejecuta plot_results.py con la configuración anterior asegurándote que el husky termina el recorrido correctamente. Se debe mostrar en el mismo plot ambos escenarios: Escenario 3.1 como el Escenario 3.2.

Analiza las gráficas resultantes y asegúrate que tienen sentido.

Escenario 3.3: Asignación velocidades y fuerzas + fricción + Inercia

Añade el modelo de inercia rotacional correcto a la barrera.

Ejecuta plot_results.py con la configuración anterior asegurándote que el husky termina el recorrido correctamente. Se debe mostrar en el mismo plot tanto el Escenario 3.1 como el Escenario 3.2 como el Escenario 3.3. Asegúrate que el plot tenga las leyendas correctas para cada escenario.

Analiza las gráficas resultantes y asegúrate que tienen sentido.

Fase 4: Controlador dinámico del Robot

Para esta Fase 4 utilizaremos la configuración establecida en la Fase 3 con la fricción y modelo de inercia. El objetivo de esta última fase es implementar un controlador dinámico de velocidad y fuerza sobre el husky que permite mantener una velocidad constante durante todo el recorrido de 2 m/s (aproximadamente).

El controlador dinámico **NO** podrá tener en cuenta la posición del robot ni de los obstáculos.
El controlador **SI** podrá tener en cuenta la velocidad actual del robot y/o su inclinación como entrada, y podrá actuar sobre las **velocidades y fuerzas de las ruedas**. Si se considera oportuno se podrá comandar diferentes velocidades y fuerzas a cada rueda o par de ruedas.

En esta fase se tiene total libertad para implementar el controlador que se considere oportuno teniendo en cuenta las restricciones anteriores.

Cuando obtengas los datos de la velocidad en función de la posición de esta fase, utiliza plot_results.py para ver el correcto funcionamiento del controlador y compararlo con el plot obtenido en el Escenario 3.3. Se debe mostrar únicamente 2 escenarios en este plot:

- Asignación velocidades y fuerzas + Fricción + Inercia (Escenario 3.3)
- Controlador dinámico (Escenario 4)

Analiza las gráficas resultantes y asegúrate que tienen sentido.