

Trabajo Fin de Máster

APRENDIZAJE REFORZADO PARA AGARRE Y MANIPULACIÓN DE ROBOTS

REINFORCEMENT LEARNING FOR GRASPING AND HADLING IN ROBOTS

Autor/es

Daniel Cubel Gálvez

Director/es

Rubén Martínez Cantín

Máster en Ingeniería Industrial

Escuela de Ingeniería y Arquitectura

2020

ÍNDICE DE CONTENIDOS

[1. OBJETO 3](#_Toc45385947)

[2. ALCANCE 4](#_Toc45385948)

[3. MARCO TEÓRICO 5](#_Toc45385949)

[3.1. APRENDIZAJE POR REFUERZO 5](#_Toc45385950)

[3.1.1. INTRODUCCIÓN 5](#_Toc45385951)

[3.1.2. PROCESOS DE DECISIÓN DE MARKOV 5](#_Toc45385952)

[3.1.3. BÚSQUEDA DE POLÍTICA 5](#_Toc45385953)

[3.2. OPTIMIZACIÓN BAYESIANA 5](#_Toc45385954)

[4. HERRAMIENTAS TECNOLÓGICAS 5](#_Toc45385955)

[4.1. COPPELIASIM 5](#_Toc45385956)

[4.1.1. INTRODUCCIÓN 5](#_Toc45385957)

[4.1.2. FUNCIONALIDAD DEL SIMULADOR 5](#_Toc45385958)

[4.1.3. CONTROL DE LA SIMULACIÓN 7](#_Toc45385959)

[4.2. PYREP 7](#_Toc45385960)

[4.3. RLBENCH 7](#_Toc45385961)

[4.4. PYTHON Y ANACONDA 7](#_Toc45385962)

[4.5. BAYESOPT 7](#_Toc45385963)

[5. DEFINICIÓN DEL PROBLEMA 7](#_Toc45385964)

[5.1. ENTORNO ROBÓTICO 7](#_Toc45385965)

[5.2. TAREAS 7](#_Toc45385966)

[6. SOLUCIÓN ADOPTADA 7](#_Toc45385967)

[6.1. DEFINICIÓN DE LAS TAREAS 7](#_Toc45385968)

[7. SIMULACIÓN Y RESULTADOS 7](#_Toc45385969)

[8. CONCLUSIONES 7](#_Toc45385970)

# OBJETO

El objeto de este Trabajo Fin de Máster (TFM) es el diseño e implementación de una biblioteca de tareas robóticas de manipulación en un simulador 3D como banco de pruebas para algoritmos de aprendizaje por refuerzo. Esta tarea incluye la definición y parametrización de las tareas a realizar, tanto de las acciones y políticas disponibles, como de la recompensas asociadas. El banco de pruebas diseñado se utiliza para evaluar las características y estudiar las mejoras de una librería de aprendizaje por refuerzo basada en optimización bayesiana.

La justificación de este trabajo se centra sobre todo en que el problemas de las tareas robóticas en ambientes controlados, en los que se conoce la forma de los objetos que se quiere manipular, su posición y orientación, y la posición y orientación del elemento terminal está prácticamente resuelto. Por ejemplo, se puede observar en el uso de los robots en las líneas de producción. Sin embargo, en ambientes no controlados pueden aparecer problemas que antes no existían, ya que a priori no se conocen las características de los objetos que se encuentran en el entorno del robot.

El aprendizaje por refuerzo supone una revolución en muchos campos de la investigación y de la tecnología, como por ejemplo en este trabajo tareas robóticas de manipulación, pero también puede ser aplicado a una amplia variedad de problemas, como pueden ser sistemas de recomendación personalizado, diseño de estrategias financieras, juegos… en donde otros métodos fallan ya sea por la falta de estructura del entorno, la complejidad del espacio de soluciones posibles o el gran volumen de datos.

En nuestro caso, el aprendizaje por refuerzo nos va a servir para resolver una serie de tareas en las que el robot aprenderá una política óptima, en lugar de resolver las tareas de forma tradicional.

# ALCANCE

Las actividades desarrolladas para la realización de este trabajo fin de máster han sido:

1. Estudios previos.
   1. Revisión bibliográfica y estado del arte del aprendizaje por refuerzo.
   2. Revisión bibliográfica y estado del arte de la optimización bayesiana.
2. Familiarización con las herramientas tecnológicas.
   1. Lenguaje de programación Python.
   2. Sistema operativo Ubuntu (Linux).
   3. Plataforma de simulación de robots CoppeliaSim.
   4. PyRep, que funciona por encima de CoppeliaSim, para controlar el simulador con python.
   5. Benchmark y entorno de aprendizaje RLBench, que funciona por encima de PyRep y CoppeliaSim.
   6. Librería de optimización bayesiana BayesOpt.
   7. Control de versiones con Git y Github
3. Diseño de tareas fácilmente reproducibles para los algoritmos de aprendizaje por refuerzo.
4. Diseño de políticas para resolver las tareas planteadas.
5. Escribir el código de interfaz entre el simulador y la librería de aprendizaje.
6. Conclusiones.

# MARCO TEÓRICO

## APRENDIZAJE POR REFUERZO

### INTRODUCCIÓN

### PROCESOS DE DECISIÓN DE MARKOV

### BÚSQUEDA DE POLÍTICA

## OPTIMIZACIÓN BAYESIANA

# HERRAMIENTAS TECNOLÓGICAS

## COPPELIASIM

### INTRODUCCIÓN

CoppeliaSim es una plataforma de simulación robótica versátil y escalable desarrollada por Coppelia Robotics. Es libre y de código abierto mientras se utilice sin fines comerciales.

En este trabajo vamos a utilizar el simulador CoppeliaSim para construir las escenas y modelos correspondientes para cada una de las tareas diseñadas y para simular las tareas. Por lo tanto, es necesario conocer la funcionalidad de CoppeliaSim para construir las escenas y de que forma se pueden controlar las simulaciones.

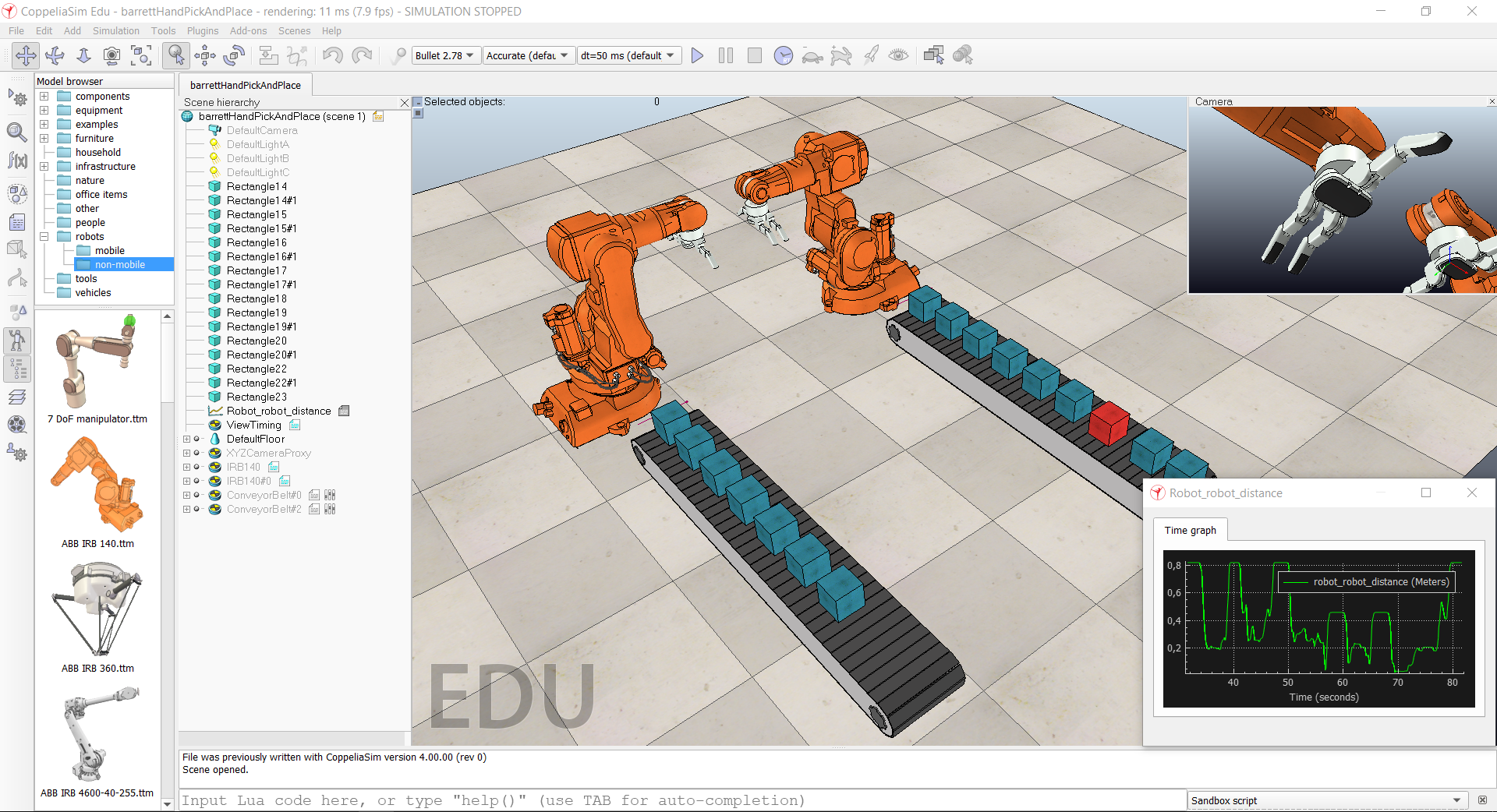


Fig. 4.1 – Interfaz de CoppeliaSim con una escena de ejemplo cargada.

### FUNCIONALIDAD DEL SIMULADOR

La funcionalidad de CoppeliaSim está relacionada tanto con los objetos de la escena como con los módulos de calculo que tiene disponibles. Dado que solo vamos a necesitar los objetos de la escena, los módulos de cálculo no se van a explicar.

#### Objetos de la escena

Una escena o modelo de CoppeliaSim contiene varios objetos de la escena u objetos elementales que se pueden ensamblar según una jerarquía de árbol. Los objetos disponibles en CoppeliaSim son los siguientes:

* Articulaciones (joints): las articulaciones son elementos que permiten enlazar dos o más objetos de la escena con uno a tres grados de libertad según el tipo de articulación (prismática, rotacional, tornillo, esférica). Tienen distintos modos de operación (modo de fuerza/momento, modo de cinemática inversa, etc.).
* Formas (shapes): las formas son mallas triangulares utilizadas para la simulación y visualización de solidos rígidos. Otros objetos de la escena o los módulos de cálculo dependen en gran medida de las formas (p. ej. los sensores de proximidad, el modulo de dinámica o el modulo de cálculo de distancia malla-malla).



Fig. 4.2 – Sensor de proximidad

* Sensores de proximidad (proximity sensor): los sensores de proximidad son elementos que llevan a cabo un cálculo de la distancia mínima exacto a la parte de una forma que se encuentra dentro de un volumen de detección configurable, en lugar de realizar una detección basada en rayos. Como resultado se obtiene una operación mas continua y, por tanto, permite una simulación más realista.
* Sensores de visión (visión sensors): los sensores de visión son elementos que permiten extraer información compleja de la imagen (colores, tamaño de los objetos, mapas de profundidad, etc.)
* Sensores de fuerza (force sensors): los sensores de fuerza son elementos que representan enlaces rígidos entre formas, que pueden registrar fuerzas y momentos aplicados y que pueden romperse cuando sobrepasa una condición determinada.
* Gráficos (graphs): los gráficos son elementos que pueden registrar una gran variedad de flujo de datos predefinidos o personalizados. Los flujos de datos pueden representarse directamente (gráfica de un tipo de dato respecto al tiempo) o combinarse con otros para mostrar un gráfico XY o curvas 3D.
* Cámaras (cameras): las cámaras son elementos que permiten la visualización de la escena cuando están asociadas a una ventana.
* Luces (lights): las luces son elementos que iluminan la escena o elementos de la escena y que influyen directamente en las cámaras o en los sensores de visión.
* Trayectorias (paths): las trayectorias son elementos que permiten definir movimientos complejos en el espacio (p. ej. sucesión de traslaciones, rotaciones y pausas, combinadas de forma libre). Las trayectorias pueden utilizarse para guiar el soplete de un robot de soldadura a lo largo de una trayectoria predefinida o para permitir los movimientos de una cinta transportadora. (como diferencio entre paths y trajectories en español?)
* Dummies: los dummies son sistemas de referencia, que pueden tener varias funciones y que son utilizados junto a otros objetos de la escena, por lo que pueden ser vistos como elementos de ayuda.

### CONTROL DE LA SIMULACIÓN

CoppeliaSim permite controlar la simulación con distintas técnicas, que pueden utilizarse de simultáneamente y también simbióticamente. Las técnicas de control disponibles son:

* Scripts incrustados (embedded scripts): los scripts incrustados son scripts que forman parte de una escena o de un modelo, y que se guardan y se cargan al mismo tiempo de la escena o el modelo. El lenguaje de programación es Lua. En una escena cualquiera, hay un script principal (main script) que se encarga de la funcionalidad general (p. ej. llamar a las funciones que se encargan de la cinemática o la dinámica). El script principal también se encarga de llamar a los scripts secundarios (child scripts) en forma de cascada con respecto a la jerarquía de la escena. Los scripts secundarios están unidos a objetos de la escena y se encargan de una parte concreta de la simulación.
* Add-ons: al igual que los scripts incrustados, los add-ons funcionan mediante scripts de Lua. Se pueden usar como funciones independientes (p. ej. para programar importadores/exportadores) o como código ejecutado de forma convencional (p. ej. como método ligero de customización del simulador).
* Plug-ins: los plug-ins son utilizados como una herramienta de customización. Pueden registrar comandos de Lua personalizados, permitiendo la ejecución de funciones callback desde un script incrustado. También pueden extender la funcionalidad de un modelo u objeto particular.
* Clientes de API remota:
* Nodos de ROS: CoppeliaSim implementa un nodo de ROS con un plugín que permite a ROS llamar a los comandos de CoppeliaSim desde los servicios ROS, o transmitir datos desde publicadores/suscriptores ROS. Los publicadores/ suscriptores pueden habilitarse con una llamada de servicio y también directamente desde CoppeliaSim, a través de un comando de un script incrustado.

## PYREP

## RLBENCH

## PYTHON Y ANACONDA

## BAYESOPT

## GIT Y GITHUB

# DEFINICIÓN DEL PROBLEMA

## ENTORNO ROBÓTICO

Como se configura el entorno del robot: robot con mesa.

## TAREAS

En que consiste cada tarea y como se define en el simulador.

# SOLUCIÓN ADOPTADA

## DEFINICIÓN DE LAS TAREAS

Para cada tarea, como se define la política y la recompensa.

# SIMULACIÓN Y RESULTADOS

# CONCLUSIONES