CENTRO DE INVESTIGACIÓN
Y DE ESTUDIOS AVANZADOS DEL
INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE MECATRÓNICA

Evaluación de enfoques de Aprendizaje Reforzado para la optimización del caminado de un robot humanoide

Tesis que presenta:
Talia Deli Peredo Barón.

Para obtener el grado de:
Maestra en Ciencias.

En la especialidad de:
Ingeniería Eléctrica.

Directores de Tesis:
Dr. Carlos Alberto Cruz Villar
Dr. Jaime Álvarez Gallegos.

Ciudad de México Febrero, 2023

Resumen

Una de las principles líneas de investigación en la robótica son los robots humanoides. Entre los humanoides más representativos se encuentra el robot NAO, usado como plataforma para la investigación y la educación [1]. Recientemente, el desarrollo de las técnicas de control libres de modelo ha permitido abordar problemas de control cuyos modelos dinámicos implican una gran cantidad de recursos computacionales para su solución, como es el caso del robot NAO.

En la presente tesis se aplican técnicas de Aprendizaje Reforzado (RL) bajo distintos enfoques para optimizar el avance de un robot humanoide NAO V5. Se programan los elementos necesarios para crear el entorno en el que se realiza la comunicación entre el robot, migrando de Choregraphe al IDLE de Python para reducir el consumo de recursos gráficos; y el programa para la implementación de las técnicas de Aprendizaje Reforzado, usando Python como lenguaje de programación. Como resultado del proceso de comunicación, se tiene un tiempo de muestreo de 100 ms.

A partir de esto, en este trabajo se pretenden optimizar dos criterios principales, usando RL, para el caminado dinámico del robot NAO: el desplazamiento y la velocidad de avance. Para el primero, el problema se aborda con la intención de un seguimiento de trayectoria; por otro lado, para la optimización de la velocidad de avance, el problema se aborda —desde una perspectiva paramétrica y cinemática— variando la *rigidez* de las articulaciones, su velocidad máxima de movimiento y el tiempo de espera entre la ejecución de comandos.

En ambos casos, se logra un mejor desempeño del criterio de optimización en comparación con el caminado predeterminado del robot, y se presentan los resultados experimentales obtenidos.

Abstract

One of the main lines of research in robotics are humanoid robots. Among the most representative humanoid robots is the NAO robot, that is used as a research and educational platform [1]. Recently, the development of model free control techniques has allowed boarding control problems which dynamical models uses a big amount of computational resources for it´s solution, this is the NAO robot case.

In this thesis work, Reinforcement Learning (RL) techniques, under different approaches, for the optimization of the NAO's walking are applied. The necessary elements are programmed to create the environment in which the communication between the robot takes place, migrating from Choregraphe to Python's IDLE to reduce the consumption of graphic resources; and the program for the implementation of Reinforcement Learning techniques, using Python as programming language. A sampling time of 100 ms. is obtained as a result of the communication process.

From this, in this work we intend to optimize two main criteria, using RL, for the dynamic walking of the NAO robot: the displacement and the forward speed.

For the first one, the problem is approached with the intention of trajectory tracking; on the other hand, for the forward speed optimization, the problem is approached - from a parametric and kinematic perspective- by varying the stiffness of the joints, their maximum speed of movement and the waiting time between the execution of commands.

In both cases, a better performance of the optimization criterion is achieved compared to the robot's default walking, and the experimental results obtained are presented.

Contenido

Capítulo 1

Introducción

1.1. Robots humanoides

Una de las principales líneas de investigación en la robótica son los robots bípedos y humanoides. Éstos han resultado de especial interés en la industria por su gran adaptabilidad debido a su similitud con el ser humano, tanto en anatomía como en capacidad de movimiento. A partir del surgimiento de estos robots se han realizado numerosas investigaciones en el desarrollo de habilidades como mantenerse en equilibrio, caminar, manipular otros objetos o, incluso, realizar trabajos colaborativos; las cuales se busca proporcionar a dichos robots [2], [3], [4] y [5].

La empresa Honda, como referente en este tema, desarrolla entre 1986 y 1993 una serie de robots humanoides experimentales que posteriormente dan lugar a la serie P, desarrollada de 1993 a 1997. Éstos humanoides poseen la habilidad de caminar, subir y bajar escaleras, girar y mantener el equilibrio en un pie y en superficies irregulares. En 2011, el robot ASIMO (Advanced Step in Innovative Mobility), sucesor de la serie P, mejora estas habilidades al correr más rápido hacia adelante y en reversa, y saltar en una o dos piernas. [6]

Recientemente, la empresa Boston Dynamics presentó su robot ATLAS, cuyas habilidades incluyen la capacidad de saltar entre superficies, saltar obstáculos, realizar saltos inversos y caminar sobre superficies irregulares.[7]

En la Tabla 1.1 se presentan, de forma resumida, las habilidades desarrolladas en los robots antes mencionados.

Honda P series (1993-1997)	ASIMO (2011)	ATLAS (2017)
Caminar	Caminar	Caminar
Subir y bajar escaleras	Subir y bajar escaleras	Subir y bajar escaleras
Girar	Correr hacia adelante y en reversa	Correr
Mantener el equilibrio en un pie y en superficies irregulares	Saltar en una o dos piernas de manera continua	Saltar (de una superficie a otra, obstáculos y saltos inversos)
	Mantener el equilibrio en superficies irregulares	Caminar sobre superficies irregulares
		

Tabla 1.1: Evolución de habilidades en humanoides

Softbank Robotics desarrolla el robot NAO, un robot humanoide programable, elegido en 2018 como la plataforma para el concurso internacional de robótica "RoboCup". Razón que propició su uso como plataforma para la investigación y la educación.[NAO Robocup]

Este trabajo de tesis se realiza haciendo uso de esta plataforma en su modelo V5 con 25 grados de libertad (GDL), en el Capítulo 3 se proporcionan las especificaciones técnicas e información respecto a la comunicación con el robot.

1.2. Enfoques de control

Se denominan *tareas de control* [8] a aquellos problemas de control donde deben tomarse decisiones o llevarse a cabo algun comportamiento.

Se identifican dos enfoques principales para abordar y resolver un problema de control:

- Desde el área de ingeniería de control se tienen las técnicas basadas en el modelo.
- Desde el área de la Inteligencia Artificial (IA) se cuenta con las técnicas de aprendizaje reforzado.

El robot NAO tiene un total de 25 GDL, cuyo control implica un vector de coordenadas generalizadas de la forma:

$$q(t) = \begin{bmatrix} q_1 & q_2 & q_3 & \dots & q_{25} \end{bmatrix}^T$$

Esto implica tener una matriz de inercia asociada de dimensión 25×25. Además, nótese que el robot cuenta con cinco cadenas cinemáticas abiertas y marcos móviles, por lo que su modelado dinámico no es tan simple. Sin mencionar el coste computacional, en términos de tiempo y recursos, que implicaría obtener la solución para las acciones de control.

Debido a la complejidad del modelo dinámico del robot y la dificultad que representa su control utilizando los métodos de control tradicionales, estudios recientes han utilizado el aprendizaje reforzado para controlar el robot bípedo [9], [10].

En esta tesis se toma el enfoque del área de la IA, utilizando técnicas de Aprendizaje Reforzado (Reinforcement Learning) para el control del robot.

1.3. Estado del arte

En la década de los 70's, Kato et. al.[2] establecen que la condición de balanceo estático se reduce a la proyección del centro de masa (COM) dentro del área formada por la planta de los pies de un robot bípedo, y al desplazamiento que cumple con este criterio se le conoce como *marcha estática*.

Años después, Vukobratović y Borova introducen el concepto del punto de momento cero (ZMP), como un indicador de equilibrio dinámico en la locomoción bípeda [3].

Múltiples investigaciones han abordado el equilibrio de robots bípedos usando algoritmos basados en COM o ZMP [4]; sin embargo, para mantener el equilibrio al caminar sobre una pendiente, en [11] proponen un Deep Deterministic Policy Gradient (DDPG) cuya velocidad de entrenamiento se mejora por medio de actores paralelos y de reproducción de experiencia priorizada (Prioritized Experience Replay o PER), demostrando su funcionamiento en simulaciones tanto de estados iniciales conocidos como desconocidos.

Aprovechando las capacidades de aprendizaje de las redes neuronales, en [12] proponen un controlador autónomo híbrido capaz de equilibrar un robot bípedo después de recibir un empuje, el cual combina dos sistemas eficientes jerárquicamente: en el nivel inferior, un controlador de estado completo, con 16 estados (8 para posiciones y 8 para velocidades), codificado en un microcontrolador, y en el nivel superior, un controlador de aprendizaje reforzado de baja velocidad; este último es entrenado mediante aprendizaje por refuerzo profundo, lo cual permite que el robot siga aprendiendo después de la implementación, siendo capaz de adaptarse a nuevos entornos.

En la tesis [5], usando el enfoque del modelo del sistema, se aborda el criterio de balanceo semi-estático y se utiliza el método SQP (Sequential Quadratic Programming) como técnica de optimización para encontrar posturas que estén en equilibrio estático con el fin de analizar y simular una tarea de trabajo colaborativo que contempla dos robots humanoides distintos: un BIOLOID GP y un robot NAO.

En cuanto a la optimización de la marcha de un robot bípedo, en [13] se propone un algoritmo de control robusto, el cual aprovecha el pie del robot que permite el rango de presencia de ZMP (Zero Moment Point), a través de la técnica de QP (Quadratic Programming). Se considera el rango máximo cinemático y la velocidad máxima del pie en el proceso de optimización al establecer las restricciones de desigualdad, optimizando las fuerzas de reacción con la superficie de contacto, así como la posición y el tiempo del paso. Con esto se logra mayor robustez en la marcha ante perturbaciones fuertes.

Con el fin de mantener la seguridad, [14] presenta un motor de simulación y entrenamiento (VISTA) basado en datos que utiliza recompensas escasas para el aprendizaje de políticas de control de vehículos autónomos de extremo a extremo, el cual aprovecha la información recolectada por los humanos, y los modelos entrenados —además de aprovechar el aprendizaje por refuerzo— pueden ser implementados directamente en el mundo real.

Para aquellos casos en que es necesaria un exploración segura y eficiente, las técnicas de exploración tradicionales no son especialmete útiles, para estos casos, García y Fernández introducen el algoritmo PI-SRL, el cual mejora de forma segura comportamientos subóptimos pero robustos para tareas de control con acciones y estados continuos, el cual aprende de forma eficiente de la experiencia obtenida del entorno, considerando una función de riesgo definida como una función de paso binario [15].

Recientemente, haciendo uso de Safe Reinforcement Learning (Aprendizaje Reforzado Seguro), García y Shafie proponen un algoritmo que incrementa la velocidad de aprendi-

zaje y reduce el número de caídas durante el proceso, permitiendo al robot NAO caminar más rápido que con el caminado predefinido. Este algortimo parte de una política de línea de base segura para aprender una mejor, siguiendo un enfoque basado en casos y, a diferencia de su predecesor, utiliza una función de riesgo monótona en continuo crecimiento [16].

1.4. Planteamiento del problema

El control de robots humanoides bajo el enfoque de control clásico, es decir, por medio de métodos basados en el modelo, puede resultar una tarea de suma complejidad, dependiendo del modelo dinámico del robot y de los actuadores a controlar.

El robot Nao tiene una rutina de caminado preprogramada, que si bien es funcional, se encuentra limitada por la velocidad que alcanza sin perder estabilidad. Para tareas que requieren una mayor velocidad de avance esta rutina de caminado no resulta viable. Para optimizar su velocidad de avance mediante un enfoque de control clásico, es decir, por medio de métodos basados en el modelo, es necesario trabajar con el modelo dinámico del sistema, que involucra matrices cuadradas de dimensión 25, convirtiendo el problema de control en una tarea compleja y cuya solución requiere una gran cantidad de recursos computacionales. Por esta razón se busca una alternativa para resolver el problema de control en el área del control basado en datos, dentro de la que se cuenta con las técnicas de Aprendizaje Reforzado.

Existen trabajos que se han dedicado a la optimización de la marcha por medio de Aprendizaje Reforzado [16]; sin embargo, no presentan una metodología para el diseño de la función de recompensa y hasta el momento se considera un proceso heurístico e intuitivo. En la mayoría de los trabajos de investigación que se encontraron hasta el momento de la publicación de la presente tesis, los modelos propuestos se entrenan y validan únicamente mediante simulación. Esto debido a que la implementación física no resulta un proceso trivial.

Con base en lo anterior, surge la motivación de proponer y evaluar el desempeño de distintos enfoques para la optimización de aspectos involucrados en el caminado de un robot humanoide, aplicando las técnicas de Aprendizaje Reforzado.

1.5. Objetivos

Evaluar dos enfoques distintos para la optimización del caminado de un robot humanoide.

1.5.1. Objetivos específicos

Para lograr el objetivo descrito anteriormente, a continuación se presentan los objetivos específicos divididos por enfoques:

- Para el enfoque 1:
  - Obtener la caracterización del caminado o comportamiento base.
  - Crear el ambiente personalizado para la implementación de las técnicas de Aprendizaje reforzado.
  - Proponer la función de recompensa para lograr un seguimiento de trayectoria.
  - Proponer la función de recompensa para la maximización del avance en función de la distancia y velocidad.
  - Elegir el agente
  - Realizar el entrenamiento del agente.
  - Evaluar los resultados del enfoque.
- Para el enfoque 2:
  - Definir las posturas clave.
  - Crear el ambiente personalizado para la implementación de las técnicas de Aprendizaje reforzado.
  - Proponer la función de recompensa para la maximización del avance en función de la distancia y velocidad.
  - Elegir el agente.
  - Realizar el entrenamiento del agente.
  - Evaluar los resultados del enfoque.

1.6. Organización de la tesis

Esta tesis consta de 6 capítulos, cuya organización y contenido se describen a continuación:

El Capítulo 1 presenta una introducción a los robots humanoides, se describen los enfoques principales para abordar tareas de control, se realiza una revisión del estado del arte en relación a robots humanoides y su control para desarrollo de habilidades, se realiza el planteamiento del problema identificado y, finalmente, se presenta el objetivo general del presente trabajo.

El Capítulo 2 presenta la descripción formal del Aprendizaje Reforzado (Reinforcement Learning), sus principales elementos, los principales algoritmos empleados en Reinforcement Learning (RL) y los casos en los que cada uno de ellos es válido. Finalmente,

se aborda el Deep Reinforcement Learning (DRL), área que surge de integrar redes neuronales con el RL.

El Capítulo 3 menciona las especificaciones del robot NAO, sus marcos de referencia, articulaciones y se presenta una sección dedicada a los efectos e importancia de su parámetro de "Rigidez". Posteriormente, se presenta la interfaz del robot, las librerías necesarias para la comunicación con el robot y para la creación del programa de RL. Después, se describe la creación del ambiente virtual como solución a la incompatibilidad entre los requerimentos de ambas librerías. Finalmente, se detalla el proceso completo de comunicación bidireccional generado para la implementación de RL en el robot NAO.

El Capítulo 4 introduce los enfoques abordados para la implementación de RL, desde los requerimentos previos para cada caso —como lo es el base behavior— y la definición de posturas para el ciclo de marcha, para el primer y segundo enfoque respectivamente; a continuación, se describe a detalle el ambiente generado para su implementación y el programa para el intercambio de datos con el robot, para cada uno de los casos.

El Capítulo 5 presenta los resultados obtenidos de la implementación en el robot NAO de cada uno de los enfoques descritos en el Capítulo 4, así como las condiciones de entrenamiento bajo las cuales se llega a dichos resultados. Se realiza un análisis y una comparación de los resultados obtenidos.

El Capítulo 6 presenta las conclusiones de este trabajo de tesis y la propuesta de trabajo futuro.

Capítulo 2

Aprendizaje Reforzado (Reinforcement Learning)

El presente capítulo reúne conceptos generales —descritos ampliamente con anterioridad y sobre los cuales hay numerosos materiales bibliográficos disponibles para su consulta— y los presenta al lector para ponerlo en contexto. Con esto, se porporciona al lector un panorama general del Aprendizaje Reforzado (Reinforcement Learning), también conocido como Aprendizaje por Refuerzo. Se describen sus principales elementos y se explica, paso a paso, el proceso mediante el cual ocurre la interacción entre estos elementos. Posteriormente, se proporciona la definición de la interacción entre sus principales elementos dentro de un marco formal. Enseguida, se abordan los principales algoritmos usados en la solución de problemas de Reinforcement Learning (RL) y, finalmente, se presenta una breve introducción al Aprendizaje Reforzado Profundo (Deep Reinforcement Learning).

2.1. Reinforcement Learning (RL)

El aprendizaje automático (Machine Learning) es un área de la IA que permite a las máquinas aprender a través del procesamiento de datos. Entre las técnicas que forman parte del Machine Learning (ML) se encuentran [17], [18]:

- Aprendizaje supervisado:
  En este caso, el sistema aprende a partir de un conjunto de datos de entrenamiento previamente etiquetados, tanto a su entrada como a su salida, por una persona que "supervisa" el proceso. El objetivo de esta técnica es que el sistema/modelo generalice y extrapole su aprendizaje y sea capaz de clasificar de forma adecuada datos que no forman parte del entrenamiento.

- Aprendizaje no supervisado:
  En este caso, el sistema aprende a partir de un conjunto de datos de entrenamiento etiquetados solo a su entrada. El objetivo de esta técnica es que el sistema aprenda a encontrar estructuras ocultas en conjuntos de datos de entrada.

- Aprendizaje semi-supervisado:
  Como la combinación de los dos anteriores, el sistema aprende a partir de un conjunto de datos de entrenamiento, etiquetados solo a su entrada o tanto a su entrada

como a su salida. El objetivo de esta técnica es que el sistema aprenda a estimar funciones para ambos tipos de datos.

- Aprendizaje por refuerzo o Reinforcement Learning (RL):
  En este caso, el sistema aprende a través de la experiencia, la cual adquiere mediante la interacción con su ambiente. El sistema descubre qué acciones le otorgan una mayor recompensa por medio de prueba y error. En esta técnica se tiene un aprendizaje orientado a objetivos, se busca que el sistema aprenda a elegir las mejores acciones para maximizar una señal de recompensa.

Se observa que cada una de las técnicas posee objetivos distintos para referirse al aprendizaje, y en lo posterior, la técnica de interés será RL. Bajo este contexto, se define por "aprendizaje" al hecho de lograr un objetivo por medio de la maximización de la recompensa obtenida por una política propuesta.

Para esto, el agente elige las acciones que le han otorgado una recompensa alta durante la experimentación, esto se conoce como *explotación*. Para encontrar las acciones que le proporcionan las mejores recompensas es necesario llevar a cabo un proceso de *exploración*, probando acciones diferentes a las que se han probado anteriormente en un estado dado del sistema, con esto se obtienen nuevos valores de recompensa que permiten encontrar las acciones más convenientes para realizar una mejor elección de la acción que permita maximizar la recompensa. Esto provoca un dilema al respecto, pues ni la exploración ni la explotación se pueden realizar de forma exclusiva si se desean obtener los mejores resultados, por lo tanto, el agente debe explorar probando nuevas acciones y favorecer de forma progresiva las acciones que le proporcionen mejores resultados [17].

2.2. Elementos principales

Un sistema de RL tiene dos componentes básicos: el *ambiente*, formado por todo aquello con lo que el agente interactúa; y el *agente*, que es el programa encargado de la toma de decisiones, tiene un objetivo explícito y es capaz de interactuar con su ambiente mediante acciones que modifiquen el estado del ambiente [17].

Además del agente y del entorno, se pueden identificar cuatro subelementos principales de un sistema de aprendizaje por refuerzo:

- Política: Es una función que mapea estados a acciones. Puede ser una función simple o ser todo un proceso de búsqueda. Por sí sola es suficiente para determinar el comportamiento del agente y suele ser estocástica, especificando probabilidades para cada acción.

- Señal de recompensa: Define cuáles son los eventos buenos y malos para el agente en cada paso del tiempo de forma inmediata. Es también la base principal para modificar el agente en función de maximizar la recompensa obtenida. Pueden ser funciones estocásticas del estado, del entorno y de las acciones tomadas.

- Función de valor: Una función de valor específica lo que es bueno a largo plazo. El valor de un estado es la cantidad total de recompensa que un agente puede esperar acumular en el futuro, a partir de ese estado, considerando los estados que probablemente seguirán y las recompensas disponibles en esos estados. Se considera

que el componente más importante de casi todos los algoritmos de aprendizaje por refuerzo es un método para estimar valores de manera eficiente.

- Modelo del ambiente: Es una representación del ambiente, la cual permite hacer inferencias sobre cómo se comportará el entorno, son usados para la planificación. Los métodos que usan modelos y planificación para resolver problemas de RL se denominan métodos basados en modelos, a diferencia de los métodos libres de modelo, que son aprendices a prueba y error [17].

El algoritmo que describe la interacción entre los elementos principales en un problema de RL es el siguiente:

1. El agente produce una acción (A_t), por medio de la cual, interactúa con el ambiente.
2. La acción generada se aplica al ambiente y provoca una transición de estado (S_t).
3. Según los resultados de aplicar la acción, el agente recibe una recompensa (R_t).
4. En función de la recompensa obtenida, el agente entenderá si la acción propuesta fue "buena" o "mala".
5. El agente preferirá aquellas acciones que le otorgan una mayor recompensa, o bien, probará otras acciones, aprendiendo a prueba y error [18].

Este proceso se ilustra en la Figura 2.1.

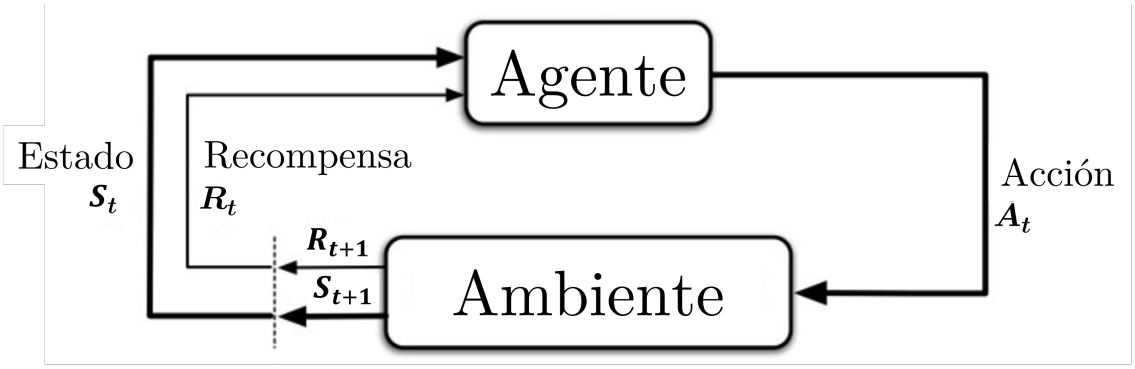


Figura 2.1: RL en lazo cerrado. Imagen tomada de [19]

2.3. Fundamento matemático

En esta sección se describe el fundamento matemático para tratar un problema de RL como un Proceso de Decisión de Markov (MDP), tomando como referencia el contenido en [19] y remitiendo al lector a esta fuente para un desarrollo más detallado.

El aprendizaje reforzado utiliza el marco formal de un proceso estocástico para definir la interacción entre un agente y su entorno en términos de estados, acciones y recompensas.

En adelante, se asume que la transición al siguiente estado solo depende del estado actual y de la acción actual, esto se conoce como la propiedad de Markov e implica que el estado actual y la acción en el tiempo t contienen información suficiente para determinar por completo la probabilidad de transición para el estado $t+1$. Esto es

$$s_{t+1} \sim P(s_{t+1}|s_t, a_t) \tag{2.1}$$

Donde:

$s_t \in S$ es el estado en el instante t, S es el espacio de estados[1]
$a_t \in A$ es la acción en el instante t, A es el espacio de acciones

Y se definen:

$P(s_{t+1}|s_t, a_t)$ es la función de transición de estado del ambiente
$R(s_t, a_t, s_{t+1})$ es la función de recompensa del ambiente

Con esto, es posible plantear un problema de RL como un Proceso de Decisión de Markov (MDP), el cual es un marco matemático que modela la toma de decisiones secuenciales.

Se asume que el agente no puede acceder a $P(s_{t+1}|s_t, a_t)$ o a $R(s_t, a_t, s_{t+1})$ y que solo puede obtener información respecto a estas funciones a través de los estados (s_t), las acciones (a_t) y las recompensas obtenidas (r_t) en el instante t , es decir la tupla (s_t, a_t, r_t).

Para formalizar el concepto de un objetivo, el cual el agente maximiza, definimos el *retorno*[2] $R(t)$ por medio de la trayectoria de un episodio, $\tau = (s_0, a_0, r_0), , \ldots, (s_T, a_T, r_T)$, donde $t = 0$ corresponde al instante inicial y $t = T$ es la duración máxima del episodio.

$$R(\tau) = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^T r_T = \sum_{t=0}^{T} \gamma^t r_t \tag{2.2}$$

La ecuación (2.2) define el retorno como la suma ponderada de las recompensas (r_t) en una trayectoria a lo largo del tiempo, donde $\gamma \in [0, 1]$ es el factor de descuento.

El objetivo $J(\tau)$ se puede definir como la expectativa de los retornos sobre muchas trayectorias, tal como se muestra en la ecuación (2.3):

$$J(\tau) = \mathbb{E}_{\tau\sim\pi}[R(\tau)] = \mathbb{E}_{\tau}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{2.3}$$

Así, el objetivo $J(\tau)$ es el retorno promediado sobre varios episodios.

[1] En este contexto, el espacio de estados está conformado por el conjunto de estados posibles en los que el sistema se puede encontrar. No se refiere al espacio de variables de estado del modelado tradicional.

Si $\gamma = 0$, solo se considera la recompensa incial, entonces:

$$R(\tau)_{\gamma=0} = r_0 \tag{2.4}$$

Si $\gamma = 1$, las recompensas de cada paso se ponderan por igual, entonces

$$R(\tau)_{\gamma=1} = \sum_{t=0}^{T} r_t \tag{2.5}$$

Se define una política, π, como la función que mapea estados a acciones: $a \sim \pi(s)$. Éstas suelen ser estocásticas. La probabilidad de una acción a a partir de un estado s se escribe como $\pi(a|s)$.

Por último, en la ecuación (2.6) se define la función de valor V^π, la cual es una medida del retorno esperado estando en el estado s, bajo la premisa de que el agente continúa actuando según su política actual π.

$$V^\pi(s) = \mathbb{E}_{s_0=s,\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{2.6}$$

2.4. Principales algoritmos de solución

Una clasificación general de los distintos tipos de algoritmos empleados para RL se muestra en la Figura 2.2. Estos algoritmos se dividen en aquellos libres de modelo (tal es el caso de este trabajo) y los que están basados en el modelo. Dentro de los algoritmos libres de modelo se encuentran algunos algoritmos de uso común: DDPG, TD3 y SAC. Es con estos últimos con los que se realizan los entrenamientos.

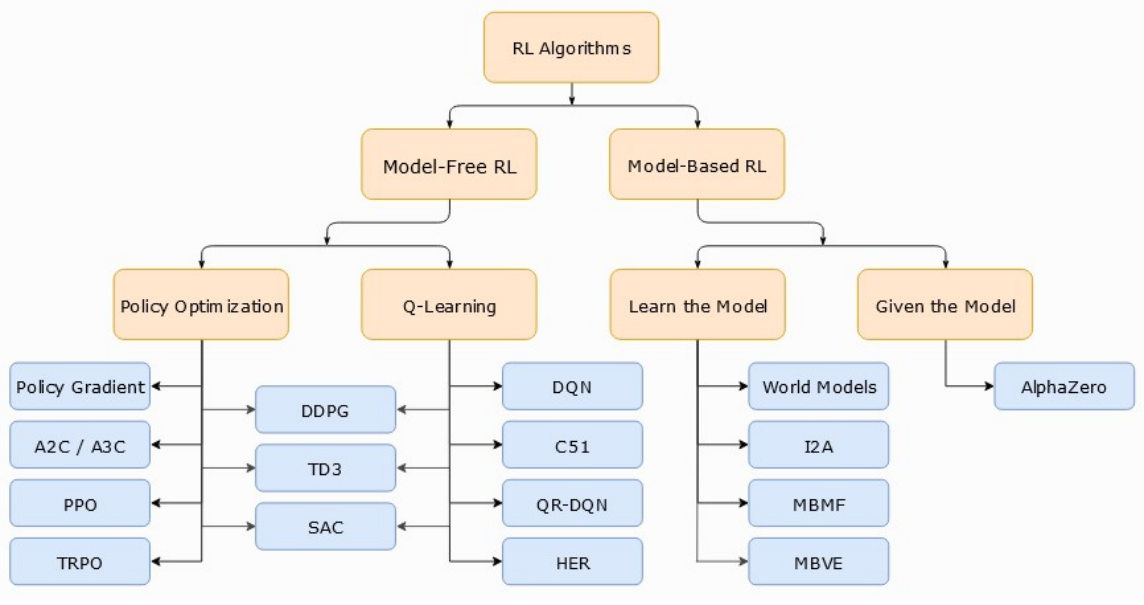


Figura 2.2: Clasificación de algoritmos para RL. Imagen tomada de [20]

Además de los principales criterios para la elección de un algoritmo mostrados en la Figura 2.2, es necesario tener en cuenta el tipo de espacio al que pertenecen las acciones y observaciones del agente.

En [21] se presenta una descripción a detalle de los algoritmos junto con algunas de las características más importantes para la selección de algoritmo.

2.5. Aprendizaje Reforzado Profundo (Deep Reinforcement Learning)

El Aprendizaje por Refuerzo Profundo, mejor conocido como "Deep Reinforcement Learning" (DRL), es un subcampo del Aprendizaje Automático (Machine Learning) que utiliza modelos de aprendizaje profundo (es decir, redes neuronales) en tareas de aprendizaje por refuerzo (RL).

Las redes neuronales profundas son, en muchos casos, los modelos paramétricos de Machine Learning más precisos para determinadas tareas. Tienen muchas capas (de ahí el término "deep"), lo que induce al modelo a aprender representaciones en capas de los datos de entrada, lo cual es una forma de composicionalidad donde un dato complejo se representa como la combinación de componentes más elementales [8].

Capítulo 3

Robot Nao y plataforma de comunicación

En el presente capítulo se proprciona al lector información acerca del robot y sus características, haciendo hincapié en que únicamente se aborda con el fin de poner en contexto al lector, se remite a la documentación oficial del robot en caso de requerir más información. Posteriormente, se describe el conjunto de elementos (Software, plataforma, librerías, etc.) involucrados en el proceso de comunicación y se explica paso a paso la forma en que éste se realiza.

3.1. Robot NAO

NAO (Figura 3.1) es un robot humanoide programable y capaz de interactuar de forma natural con todo tipo de público. Desarrollado en 2008 por la empresa Softbank Robotics (Antes Aldebaran); este robot escucha, ve, habla y se relaciona con el medio según se le haya programado.

Sus especificaciones técnicas son:

- Cuenta con 25 grados de libertad.
- El robot tiene una altura de 574 mm, una profundidad de 311 mm y un ancho de 275 mm.
- Tiene un peso de 5.4 Kg.
- Cuenta con dos cámaras 2D para identificar objetos en el campo visual.
- Cuenta con 4 micrófonos y 2 altavoces.
- Posee 9 sensores táctiles, 2 sensores de ultrasonidos y 8 sensores de presión.
- Cuenta con un acelerómetro y un giroscopio.
- CPU ATOM Z530 1.6 GHz.
- 1 GB RAM y 2 GB Flash memory.

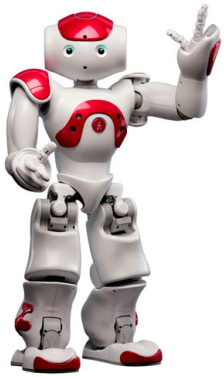


Figura 3.1: Robot NAO

3.2. Marcos de referencia

Para conocer la posición del robot y de sus articulaciones, es necesario establecer un sistema de referencia. Para el caso del robot NAO existen tres marcos de referencia principales (Figura 3.2):

- **Frame Torso**: Este marco toma como referencia el torso del robot, así, cuando este se desplaza, modifica su posición o gira, cambiando de orientación, el marco se mueve con él.
- **Frame Robot**: Este marco se ubica entre ambos pies con un eje x que apunta siempre hacia adelante, de tal forma que, conforme gira el robot, el marco gira sobre un eje z.
- **Frame World**: Este marco tiene un origen fijo que no cambia con los desplazamientos o rotaciones del robot.

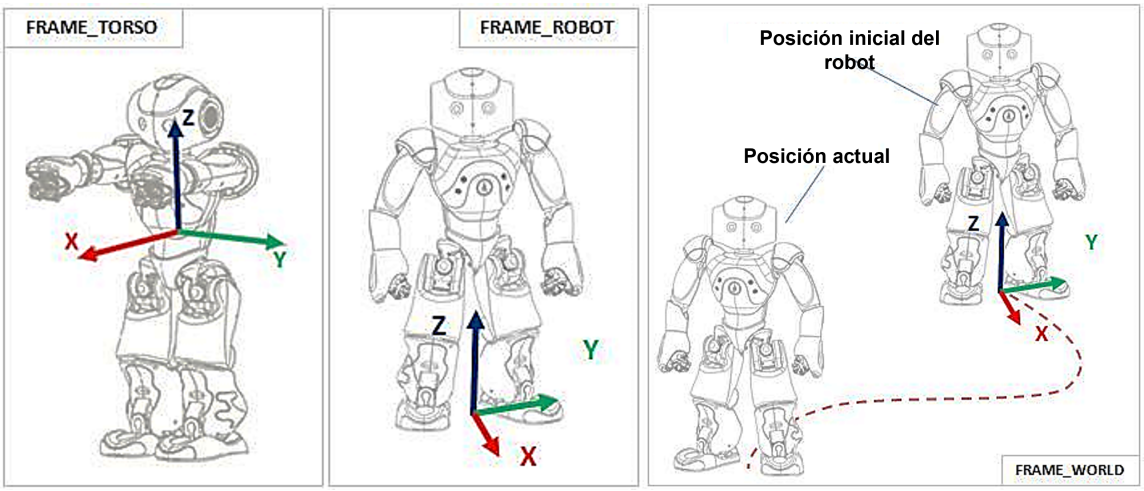


Figura 3.2: Marcos de referencia del NAO

3.3. Articulaciones

Se define una articulación en el robot como la unión entre dos componentes, o partes, de su cuerpo.

Se coloca un marco en cada articulación, de tal forma que las rotaciones Roll (WX), Pitch (WY) y Yaw (WZ), se realizan con respecto a los ejes X, Y y Z, respectivamente

Cuando se realizan rotaciones, se toma como referencia fija la parte del cuerpo más cercana al torso, y la más alejada de él como aquella sobre la que se aplica la rotación.

La Figura 3.3 muestra el nombre de cada una de las articulaciones, la ubicación y su o sus correspondientes actuadores.

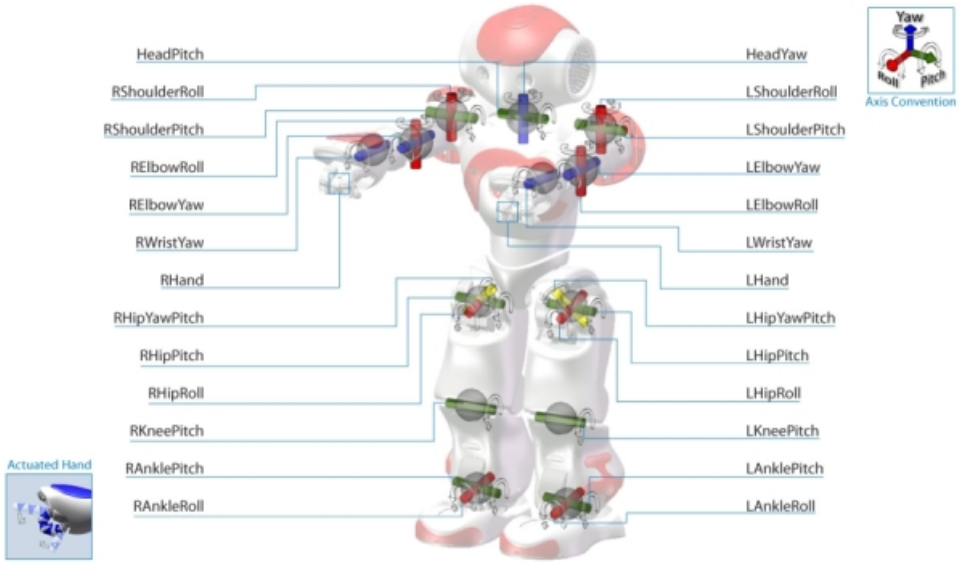


Figura 3.3: Articulaciones y actuadores del NAO. Imagen tomada de [90]

El rango de movimiento de cada articulación se puede consultar en el Anexo 1.[14]

3.4. Rigidez

El parámetro de rigidez, "stiffness" en adelante, se refiere a la dureza de la articulación y corresponde a una limitación del torque en los motores, el cual va de los 8.5–74 mN·m, dependiendo de la articulación que se esté manejando [22], y está dado en términos de porcentaje [23].

Este parámetro puede tomar valores entre [0-1], donde un 0 corresponde a 0 % del torque permitido, en este caso, el controlador de la articulación no tiene potencia de torque y la articulación no es capaz de efectuar movimientos; mientras que un valor de 1 se refiere al 100 % del torque permitido para alcanzar una posición determinada, lo cual implica máxima corriente.

En términos de corriente, este porcentaje se aplica directamente a la corriente máxima. El uso prolongado del robot con un valor de stiffness = 1, provoca que los motores se calienten.

Visualmente, la modificación de este parámetro provoca movimientos más "suaves" o "agresivos".

3.5. Comunicación RL-NAO

El fabricante del robot NAO proporciona un software llamado Choregraphe para la programación del robot que incluye una versión virtual del robot; sin embargo, en este robot virtual sus articulaciones se consideran siempre rígidas y no se ve afectado por la gravedad [24]. A su vez, no es posible modificar en el robot virtual la fricción existente entre los pies del robot y la superficie de contacto con la que ha de interactuar. Por estas razones, se opta por utilizarlo únicamente para simulación y realizar la comunicación con el robot real fuera de esta interfaz, usando Python como lenguaje de programación.

El software principal que se ejecuta en el robot y se encarga de su control, se conoce como NAOqi. La programación del NAO se realiza bajo el marco de programación de NAOqi. Este marco permite el acceso a los distintos módulos (movimiento, audio, etc.), la comunicación y el intercambio de información; la programación en las plataformas Windows, Linux o Mac; y presenta una Interfaz de Programación de Aplicaciones (API) idéntica para C++ y para Python [25].

En este trabajo se utiliza la plataforma Windows, en la cual se requiere la versión 2.7 de Python para la instalación y ejecución de NAOqi. Se programa directamente en el entorno de desarrollo integrado de Python, en adelante IDLE [26], dentro del ambiente principal del sistema, lo correspondiente a la comunicación directa con el robot.

Para la parte correspondiente a RL, se hace uso de la librería de código abierto llamada "Gym". Gym permite el desarrollo y la comparación de algoritmos de RL, proporciona una API para la comunicación de estos algoritmos y entornos, de los cuales proporciona un conjunto estándar con la posibilidad de crear entornos personalizados. Soporta versiones de Python 3.7 o superior [27].

Debido a la incompatibilidad de las versiones soportadas por Gym y NAOqi se requiere la creación de un ambiente virtual para la ejecución de ambos sin que se presenten conflictos al instalar actualizaciones.

De forma general, un ambiente virtual es un entorno de ejecución aislado en el cual es posible instalar y actualizar paquetes y librerías sin interferrir con el comportamiento de otras aplicaciones en el mismo sistema [28].

Se crea un ambiente virtual llamado "rl_env", en el que se instala todo lo necesario para la implementación de algoritmos de RL, incluyendo una versión de Python 3.10. Para programar dentro de este ambiente virtual lo correspondiente a RL, se utiliza "Jupyter notebook", una interfaz en línea de código abierto similar al IDLE.

A continuación, se describe el proceso de comunicación entre el programa de RL y el robot NAO de la siguiente forma:

1. El programa en Jupyter genera datos en forma de vectores. Estos vectores de datos son escritos en un archivo .csv, el cual funciona como intermediario o intérprete entre el ambiente virtual y el ambiente principal del equipo de cómputo.

2. El programa en el IDLE, realiza la lectura del archivo, procesa los datos y envía los comandos e información necesaria al robot para su correspondiente ejecución.

3. El robot lleva a cabo la ejecución de los comandos recibidos.

4. El robot retroalimenta información desde sus sensores.

5. El programa en el IDLE, recibe la información retroalimentada por el robot en forma de datos, los cuales escribe en un archivo .csv.

6. El programa en Jupyter realiza la lectura y procesamiento de los datos contenidos en el archivo .csv.

Este proceso se realiza de forma constante para el cálculo de recompensas para las acciones propuestas por el agente de RL en función de los valores retroalimentados por el robot, vea la Figura 3.4.

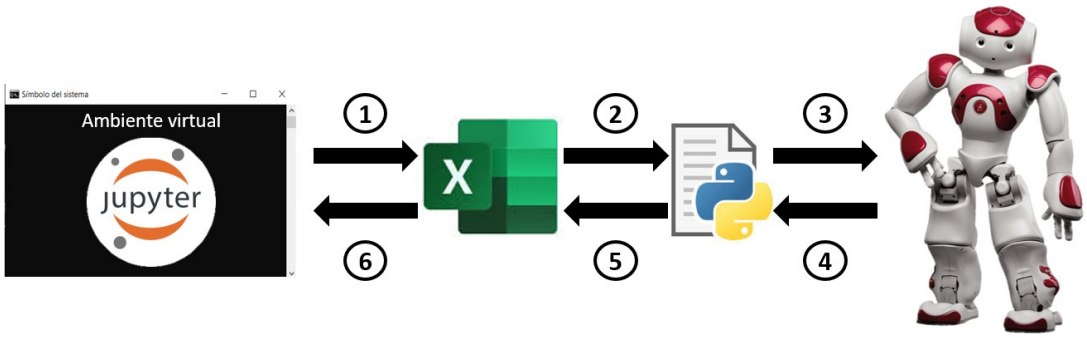


Figura 3.4: Proceso de comunicación

Debido a los pasos involucrados en el proceso de comunicación antes descrito y los tiempos de ejecución implicados en él, se tienen tiempos de muestreo alrededor de 100 ms. Este es un tiempo muy alto para controlar sistemas en general, considerando que, para el control de sistemas mecatrónicos los tiempos de muestreo suelen ser inferiores a los 10 ms.

En resumen, a lo largo de este capítulo se han dado a conocer las especificaciones del robot y se han presentado algunos de los principales conceptos con los que se ha de trabajar durante los capítulos siguientes. Se han dado a conocer los elementos que forman parte de la plataforma creada para establecer la comunicación entre RL y el robot, así como una descripción paso a paso de este proceso. Finalmente, el tiempo de muestreo resultante da lugar al enfoque 2, donde no se requiere una retroalimentación en cada paso y, por lo tanto, no es necesario un tiempo de muestreo constante.

Capítulo 4

Enfoques abordados en la implementación de RL.

En este capítulo se presentan los enfoques mediante los cuales se aborda la implementación de las técnicas de aprendizaje reforzado en el robot NAO. El primero de ellos, partiendo de una rutina de caminado como referencia, tiene como objetivo realizar un seguimiento de trayectoria; mientras que el segundo, ajusta el valor de rigidez del robot para un ciclo de marcha personalizado.

Se describe a detalle el ambiente personalizado creado para la ejecución e implementación de RL, así como el programa creado para el intercambio de datos con el robot NAO, en cada uno de los enfoques y sus casos.

4.1. Implementación de RL para la generación de movimientos articulares basado en un Comportamiento Base

En este enfoque se realiza la caracterización del caminado predeterminado que realiza el NAO mediante sus rutinas pre-programadas. En adelante, al hacer referencia a los datos obtenidos por esta caracterización, se empleará el término "Comportamiento Base".

Posteriormente, se propone un seguimiento de trayectoria para una pierna, tomando como valores deseados los datos contenidos en el Comportamiento Base. Lo que se busca con esto es, por medio de una función de recompensa multiobjetivo, maximizar la velocidad de avance partiendo del caminado de referencia.

A continuación, se aborda el programa necesario para la creación del ambiente para Reinforcement Learning, explicando a detalle sus elementos principales. Finalmente, se presentan las generalidades referentes a la creación del programa encargado de la comunicación bidireccional entre el programa de RL y el robot.

4.1.1. Obtención de datos del Comportamiento Base

Para obtener el Comportamiento Base, se ejecuta un caminado predeterminado de 1 metro mientras se realizan lecturas de las posiciones angulares de un conjunto definido de articulaciones y, con la información obtenida, se genera una base de datos.

Se definen en total 18 actuadores principales, los cuales se muestran en la Figura 4.1. Considerando la simetría del robot, se muestran solo las ubicaciones de los actuadores correspondientes al lado izquierdo para brazos y piernas.

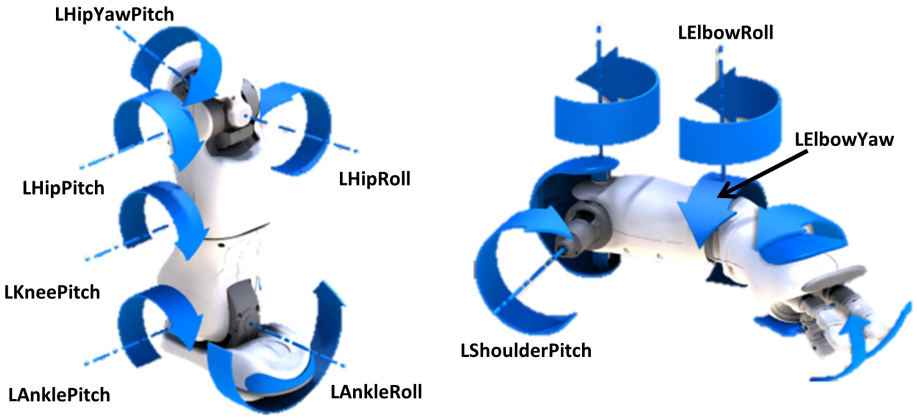


Figura 4.1: Actuadores en brazos y piernas

Se realiza un programa en el IDLE de Python, que se ejecuta en el ambiente principal, en el cual se crea un vector de nombres con cada una de las articulaciones de las cuales se desea obtener la lectura. El vector correspondiente se muestra en la siguiente ecuación:

$$
\begin{aligned}
names = (&\underbrace{\text{“}LHipYawPitch\text{”}, \text{“}RHipYawPitch\text{”}}_{Pelvisjoints}, \\
&\underbrace{\text{“}LHipRoll\text{”}, \text{“}LHipPitch\text{”}, \text{“}LKneePitch\text{”}, \text{“}LAnklePitch\text{”}, \text{“}LAnkleRoll\text{”}}_{LeftLegjoints}, \\
&\underbrace{\text{“}RHipRoll\text{”}, \text{“}RHipPitch\text{”}, \text{“}RKneePitch\text{”}, \text{“}RAnklePitch\text{”}, \text{“}RAnkleRoll\text{”}}_{RightLegjoints}, \\
&\underbrace{\text{“}LShoulderPitch\text{”}, \text{“}LElbowRoll\text{”}, \text{“}LElbowYaw\text{”}}_{LeftArmjoints}, \\
&\underbrace{\text{“}RShoulderPitch\text{”}, \text{“}RElbowRoll\text{”}, \text{“}RElbowYaw\text{”}}_{RightArmjoints})
\end{aligned}
$$

Para realizar las lecturas, se obtienen los valores de la posición angular correspondiente a cada articulación mediante el comando “getAngles()” con un tiempo de muestreo de 50 ms. Este comando requiere como argumento de entrada un vector de nombres de las articulaciones o actuadores de los que se desee conocer su valor, y nos regresa un vector de dimensión igual al número de elementos en el vector de nombres. Este nuevo vector se almacena dentro de una variable auxiliar y, posteriormente, se agrega a una lista que contiene todas las lecturas realizadas durante el caminado.

Finalmente, se exporta la lista con las lecturas a un archivo .csv. Se adjunta el archivo en formato .csv en el enlace: Comportamiento Base.

4.1.2. Seguimiento de trayectoria para una pierna

Se propone realizar un seguimiento de trayectoria para la pierna derecha, para lo cual, es necesario definir qué articulaciones serán consideradas en la tarea de seguimiento de trayectoria.

La pierna derecha está compuesta por 6 articulaciones, las cuales se muestran en la Figura 4.2, con su sentido de giro.

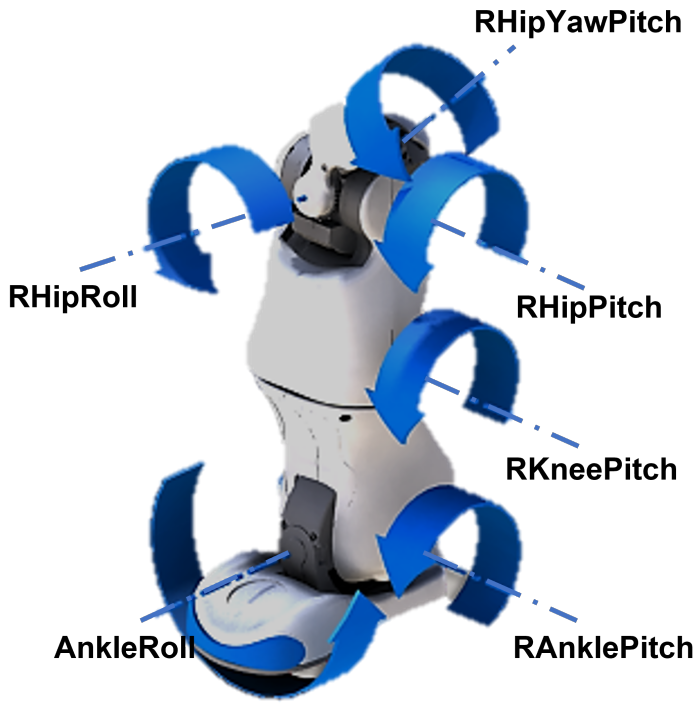


Figura 4.2: Pierna derecha

Debido al tipo de giro provocado por la articulación "RHipYawPitch", esta se descarta como parte de la tarea de seguimiento de trayectoria. Finalmente, se considera el siguiente vector de articulaciones involucradas en esta tarea:

$$RLeg_{joints} = \begin{bmatrix} RHipRoll & RHipPitch & RKneePitch & RAnklePitch & RAnkleRoll \end{bmatrix} \tag{4.1}$$

Definidas estas articulaciones, se extrae la información del Comportamiento Base para sus posiciones deseadas a lo largo del tiempo.

Una vez que se tiene esta selección, es posible realizar los programas correspondientes: el que genera nuestro ambiente de RL y el que se encarga de la comunicación con el robot.

Creación del ambiente para RL

Gym proporciona todo un catálogo de ambientes (en adelante "environment" o "ambiente" se utilizará para referirse a los ambientes en el contexto de la librería gym). Entiéndase por "ambiente" aquel programa donde se define y evoluciona el modelo de un sistema.

Para la tarea de seguimiento de trayectoria de la pierna derecha del Nao, es necesario crear un ambiente personalizado. En términos de programación, un ambiente es una clase formada por 4 funciones principales [29]:

- Función de inicialización de la clase (Init):
  Se declaran las variables necesarias a lo largo del programa. Se define el espacio de observación y el espacio de acciones, estos espacios establecen el formato para las acciones y observaciones, así como los límites en que son válidos y la dimensión de cada uno. Los tipos de espacios principales son:

  - *Discreto (Discrete)*:
    Se usa para espacios continuos de dimensión n y es un espacio acotado, en el que se pueden establecer límites superiores e inferiores entre los que las observaciones y las acciones son válidas.
  - *Caja (Box)*:
    Declara un espacio discreto donde las acciones y observaciones pueden tomar los valores $\{0, 1, 2, \cdots, n-1\}$.
  - *Diccionario (Dict)*:
    Define un diccionario compuesto por espacios simples (Discrete, o Box)[GymSpaces]

  Dentro de esta función se define también el estado inicial de nuestro problema de RL.

- Función de paso (Step):
  En esta función se declara la forma en que las acciones interactúan con el ambiente modificando su estado actual por medio de una acción que recibe como argumento. Se define la ecuación correspondiente a la función de recompensa y se realiza el proceso iterativo de ajuste de acuerdo al desempeño que muestre durante el entrenamiento.
  Se declaran y evalúan las condiciones de paro, se calcula la recompensa y se definen los elementos que la función va a regresar: observación, recompensa obtenida, valor booleano asociado a las condiciones de paro (en adelante "done") y, si existe, la información adicional de nuestro problema.
  Se establece el efecto de la ejecución de un paso en el número de pasos.

- Función de renderizado (Renderize):
  En esta función se genera una visualización del estado actual del sistema.
  En los ambientes que se han de crear en este capítulo no se hace uso de esta función, debido a que los datos se envían directamente al robot físico o a la versión virtual del robot existente en Choreographe.

- Función de reinicio: (Reset)
  Esta función resetea el ambiente a su estado inicial, al igual que el resto de las variables, cuando "done" adquiere un valor verdadero para iniciar un nuevo episodio.

Por último, retroalimenta la observación del ambiente en su estado inicial y la información adicional del problema.

Para el ambiente creado, los elementos principales de nuestro sistema RL son:

- Observaciones:
  Se definen como un espacio tipo *Box*, correspondientes al vector V^{Obs}.
  El vector de observación se describe en la ecuación (4.2):

$$V^{Obs} = \begin{bmatrix} v_{state} & v_{state_{real}} & v_{state_{n-1}} & v_{state_{n-2}} & v_{error} \end{bmatrix}^T \tag{4.2}$$

  Donde:

v_{state} =Vector de estado formado por el estado teórico del sistema (posiciones angulares teóricas de las articulaciones)
$v_{state_{real}}$ =Vector de estado real, formado por el estado real del sistema (posiciones angulares reales de las articulaciones)
$v_{state_{n-1}}$ =Vector de estado anterior $(n-1)$ del sistema (posiciones angulares reales de la muestra $(n-1)$)
$v_{state_{n-2}}$ =Vector de estado dos muestras antes $(n-2)$ del sistema (posiciones angulares reales de la muestra $(n-2)$)
v_{error} =Vector de error en la muestra actual (n) (diferencia entre las posiciones angulares reales y las posiciones angulares deseadas)

  V^{Obs} es acotada, término a término, por V_{low}^{Obs} y V_{high}^{Obs}, correspondientes a las matrices generadas por los vectores de límites inferiores v_{low}^{Obs} y superiores v_{high}^{Obs}, respectivamente. Así,

$$V_{low}^{Obs} = \begin{bmatrix} v_{low}^{Obs} & v_{low}^{Obs} & v_{low}^{Obs} & v_{low}^{Obs} & v_{low}^{Obs} \end{bmatrix}^T \tag{4.3}$$
$$V_{high}^{Obs} = \begin{bmatrix} v_{high}^{Obs} & v_{high}^{Obs} & v_{high}^{Obs} & v_{high}^{Obs} & v_{high}^{Obs} \end{bmatrix}^T \tag{4.4}$$

  Donde:

$$v_{low}^{Obs} = \begin{bmatrix} -0.790 & -1.536 & -0.092 & -1.186 & -0.768 \end{bmatrix}$$

$$v_{high}^{Obs} = \begin{bmatrix} 0.379 & -0.200 & 2.112 & 0.932 & 0.397 \end{bmatrix}$$

  Los elementos de los vectores v_{low}^{Obs} y v_{high}^{Obs} corresponden a los límites —modificados en algunos casos mediante una reducción de su rango de movimiento por seguridad del robot— del rango de movimiento de las articulaciones de la pierna derecha, vea la Figura A.6, que forman parte del vector $RLegjoint$, mostrado en la ecuación (4.1), en su equivalente en radianes.

- Acciones:
  Las acciones se definen como un espacio tipo *Box* y están dadas por el vector V^{Act}. Cada elemento que compone el vector se encuentra acotado por los límites deseados para la acción, que en este caso, corresponden a un rango de movimiento modificado para cada articulación de la ecuación (4.1), con el fin de evitar choques entre sus extremidades.
  Este vector se describe en la ecuación (4.5):

$$V^{Act} = \begin{bmatrix} \theta_{j1} & \theta_{j2} & \theta_{j3} & \theta_{j4} & \theta_{j5} \end{bmatrix} \tag{4.5}$$

  Donde:

  $\theta_{ji},\ i = 1, 2, \cdots, 5 =$ Posiciones angulares teóricas de la i-ésima articulación del sistema

  De esta forma, V_{low}^{Act} corresponde a los límites inferiores del rango de movimiento modificado, mientras que V_{high}^{Act} a los límites superiores del rango de movimiento modificado. Para ambos casos existen valores numéricos acotados.
  Así, los límites inferiores están dados por el vector:

$$V_{low}^{Act} = \begin{bmatrix} -0.14 & -0.733 & 0 & -0.44 & -0.76 \end{bmatrix}$$

  Y limites superiores dados por el vector:

$$V_{high}^{Act} = \begin{bmatrix} 0.018 & 0.43 & 1.0472 & 0.1 & 0.39 \end{bmatrix}$$

- Función de recompensa:

  La función de recompensa propuesta toma como referencia la función propuesta en [30], En esta función de referencia, penalizan las desviaciones dentro de un rango acotado de la posición deseada para el péndulo invertido.
  Con esto en consideración, para realizar el seguimientro de trayectoria se propone la función descrita en la ecuación (4.6):

$$\begin{aligned} r_i = & -10000 * E_{RHipRoll}^2 - 10000 * E_{RHipPitch}^2 \\ & - 10000 * E_{RKneePitch}^2 - 10000 * E_{RAnklePitch}^2 \\ & - 20000 * E_{RAnkleRoll}^2 \end{aligned} \tag{4.6}$$

  Donde:

$E_{RHipRoll} =$ Error en la posición angular de la articulación “RHipRoll”
$E_{RHipPitch} =$ Error en la posición angular de la articulación “RHipPitch”
$E_{RKneePitch} =$ Error en la posición angular de la articulación “RKneePitch”
$E_{RAnklePitch} =$ Error en la posición angular de la articulación “RAnklePitch”
$E_{RAnkleRoll} =$ Error en la posición angular de la articulación “RAnkleRoll”

De esta forma, la recompensa del i-ésimo paso está ligada directamente al error presente en cada articulación. En esta función se penaliza el cuadrado del error en todas las articulaciones, poniendo un peso mayor a la penalización del error en la articulación "RAnkleRoll", para evitar posiciones en el tobillo que puedan generar algún daño al motor.

- Condiciones de paro (*"done"*):
  - Condición de paro por duración máxima de episodios

  En este caso no es necesario imponer restricciones o condiciones de paro en caso de pérdida de equilibrio, puesto que únicamente es de interés el seguimiento de las trayectorias descritas por las posiciones de las articulaciones en el Comportamiento Base.

4.1.3. Seguimiento de trayectoria para dos piernas

Al tener el seguimiento de trayectoria para una pierna, se extiende al seguimiento de trayectoria para ambas piernas considerando el siguiente vector de articulaciones:

$$Legs_{joints} = \begin{bmatrix} LHipRoll \\ LHipPitch \\ LKneePitch \\ LAnklePitch \\ LAnkleRoll \\ RHipRoll \\ RHipPitch \\ RKneePitch \\ RAnklePitch \\ RAnkleRoll \end{bmatrix}^T \tag{4.7}$$

Creación del ambiente para RL

Para el programa elaborado, los elementos principales de nuestro sistema RL son:

- Observaciones:
  Se definen como un espacio tipo *Box* correspondiente al vector V^{Obs}.
  El vector de observación V^{Obs} está compuesto de la misma forma que el descrito en la ecuación (4.8):

$$V^{Obs} = \begin{bmatrix} v_{state} & v_{state_{real}} & v_{state_{n-1}} & v_{state_{n-2}} & v_{error} \end{bmatrix}^T \tag{4.8}$$

  Al igual en que en el caso del seguimiento para una pierna los elementos corresponden a posiciones angulares dentro de un rango de movimiento modificado para las articulaciones en cuestión .
  Se definen los límites inferiores, los cuales están dados por la matriz:

$$V^{Obs}_{low} = \begin{bmatrix} v^{Obs}_{low} & v^{Obs}_{low} & v^{Obs}_{low} & v^{Obs}_{low} & v^{Obs}_{low} \end{bmatrix}^T$$

Donde:

$$v_{low}^{Obs} = \begin{bmatrix} -0.379 & -1.536 & -0.092 & 1.189 & -0.397 & -0.790 & -1.536 & -0.092 & -1.186 & -0.768 \end{bmatrix}$$

Y los limites superiores, que están dados por la matriz:

$$V_{high}^{Obs} = \begin{bmatrix} v_{high}^{Obs} & v_{high}^{Obs} & v_{high}^{Obs} & v_{high}^{Obs} & v_{high}^{Obs} \end{bmatrix}^T$$

Donde:

$$v_{high}^{Obs} = \begin{bmatrix} 0.790 & -0.470 & 2.112 & 0.922 & 0.768 & 0.379 & -0.200 & 2.112 & 0.932 & 0.397 \end{bmatrix}$$

Al igual que en el caso de seguimiento para una pierna, los valores numéricos corresponden a los límites del rango de movimiento modificado para cada articulación, vea las Figuras A.5 y A.6, con el fin de evitar colisiones entre ellas durante el entrenamiento

- Acciones:
  Se definen como un espacio tipo *Box*, correspondiente al vector V^{Act}.
  Este vector se describe en la ecuación (4.9):

$$V^{Act} = \begin{bmatrix} \theta_{j1} & \theta_{j2} & \theta_{j3} & \theta_{j4} & \theta_{j5} & \theta_{j6} & \theta_{j7} & \theta_{j8} & \theta_{j9} & \theta_{j10} \end{bmatrix} \tag{4.9}$$

  Donde:

  $\theta_{ji},\ i = 1, 2, \cdots, 10:$ Posiciones angulares teóricas de la i-ésima articulación del sistema.

  Lon límites inferiores de estos valores, están dados por el vector:

$$V_{low}^{Act} = \begin{bmatrix} -0.018 & -0.733 & 0 & -0.44 & -0.39 & -0.14 & -0.733 & 0 & -0.44 & -0.76 \end{bmatrix}$$

  Y los limites superiores por el vector:

$$V_{high}^{Act} = \begin{bmatrix} 0.14 & 0.43 & 1.0472 & 0.1 & 0.76 & 0.018 & 0.43 & 1.0472 & 0.1 & 0.39 \end{bmatrix}$$

  En este caso, los límites corresponden a los extremos del rango de movimiento modificado de cada articulación que forma el vector.

- Función de recompensa:
  La función de recompensa para este caso, además de las consideraciones realizadas para el seguimiento de trayectoria de una pierna, se vale de las ideas presentes en la función de recompensa propuesta en [31]. En esta función de referencia se penalizan las desviaciones tanto laterales como verticales, considerando también estas desviaciones como condiciones de paro.
  La función de recompensa propuesta para realizar el seguimientro de trayectoria para ambas piernas busca maximizar la distancia recorrida con mayor prioridad que seguir la trayectoria de referencia y está dada por la ecuación (4.10):

$$\begin{aligned} r_i = & 100000 * D_x \\ & - 10000 * E^2_{RHipRoll} - 10000 * E^2_{RHipPitch} \\ & - 10000 * E^2_{RKneePitch} - 10000 * E^2_{RAnklePitch} \\ & - 16000 * E^2_{RAnkleRoll} \\ & - 10000 * E^2_{LHipRoll} - 10000 * E^2_{LHipPitch} \\ & - 10000 * E^2_{LKneePitch} - 10000 * E^2_{LAnklePitch} \\ & - 16000 * E^2_{LAnkleRoll} \\ & - 1000000000 * w_p \end{aligned} \tag{4.10}$$

  Donde:

$$\begin{aligned} D_x &= \text{La distancia recorrida en el eje } x \\ E_{RHipRoll} &= \text{Error en la posición angular de la articulación “RHipRoll”} \\ E_{RHipPitch} &= \text{Error en la posición angular de la articulación “RHipPitch”} \\ E_{RKneePitch} &= \text{Error en la posición angular de la articulación “RKneePitch”} \\ E_{RAnklePitch} &= \text{Error en la posición angular de la articulación “RAnklePitch”} \\ E_{RAnkleRoll} &= \text{Error en la posición angular de la articulación “RAnkleRoll”} \\ E_{LHipRoll} &= \text{Error en la posición angular de la articulación “RHipRoll”} \\ E_{LHipPitch} &= \text{Error en la posición angular de la articulación “RHipPitch”} \\ E_{LKneePitch} &= \text{Error en la posición angular de la articulación “RKneePitch”} \\ E_{LAnklePitch} &= \text{Error en la posición angular de la articulación “RAnklePitch”} \\ E_{LAnkleRoll} &= \text{Error en la posición angular de la articulación “RAnkleRoll”} \\ w_p &= \text{Penalización por condiciones de paro activadas} \end{aligned}$$

  Se plantea en esta forma para buscar que el agente aprenda, en primer lugar, a evitar activar las condiciones de paro debido a la penalización asociada a ellas. A la distancia recorrida en el eje x se le asigna una recompensa positiva, para que el agente busque maximizar este valor. Mientras que, al igual que para el seguimiento de trayectoria para una sola pierna, se penaliza el error cuadrado en todas las articulaciones, especialmente en la articulación AnkleRoll en cada pierna, pero a diferencia del caso anterior, se penaliza un 60 % más y no el doble como en la función de recompensa de la ecuación (4.6).

- Condiciones de paro (*done*):
  Debido al manejo de ambas piernas es necesario imponer condiciones de paro, con su respectiva penalización, para detectar la pérdida de equilibrio y desviaciones de la dirección de avance deseada.

  - Condición de paro por duración máxima de episodios
  - Condición de paro por desviación lateral ≥ 20 cm.
  - Condición de paro por desviación vertical ≥ 8 cm.
  - Condición de paro por rotación en $|WX| \geq 0.3$ rad
  - Condición de paro por rotación en $|WY| \geq 0.3$ rad

  Las condiciones de paro por desviaciones y por rotación están asociadas a la penalización en w_p.

4.2. Implementación de RL para el ajuste de la rigidez basado en la observación de tiempo y distancia recorrida en un ciclo de marcha

En este enfoque se realiza la definición de una serie de posturas, las cuales al ejecutarse de forma continua y cíclica, dan lugar a un ciclo de marcha.

A partir de esto, se propone el ajuste del valor del parámetro de rigidez, de la velocidad de movimiento de las articulaciones y del tiempo de espera entre envío de comandos con el fin de maximizar el tiempo que el robot se mantiene en pie y la distancia que recorre en dicho tiempo, por medio de la implementación de las técnicas de Reinforcement Learning. Se describe el programa realizado para la creación del ambiente para Reinforcement Learning, explicando sus elementos principales. Finalmente, se presentan las generalidades referentes a la creación del programa encargado de la comunicación entre el programa de RL y el robot.

4.2.1. Definición de posturas del robot para el ciclo de marcha y obtención de datos.

De forma general, un ciclo de marcha comprende el lapso entre dos acciones sucesivas que involucran la misma pierna sin que ambos pies estén separados del suelo al mismo tiempo.

Para una descripción más detallada de los conceptos relacionados a la marcha humana y criterios de equilibrio de la marcha bípeda, se remite al lector a [5].

En este caso se consideran posturas específicas definidas mediante desplazamientos de los efectores finales del robot por medio del comando *"SetPosition"*. A continuación se obtienen las posiciones angulares para cada articulación, se guardan en un vector de posiciones para las articulaciones involucradas y, posteriormente, estas posturas se ejecutan de manera secuencial, mientras no se presente una condición de paro, dando lugar al ciclo de marcha.

Bajo este enfoque, en (4.11) se presentan las articulaciones que componen el vector de articulaciones.

$$names = \begin{bmatrix} LShoulderPitch \\ LShoulderRoll \\ LElbowYaw \\ LElbowRoll \\ LHipYawPitch \\ LHipRoll \\ LHipPitch \\ LKneePitch \\ LAnklePitch \\ LAnkleRoll \\ RHipYawPitch \\ RHipRoll \\ RHipPitch \\ RKneePitch \\ RAnklePitch \\ RAnkleRoll \\ RShoulderPitch \\ RShoulderRoll \\ RElbowYaw \\ RElbowRoll \end{bmatrix}^T \tag{4.11}$$

Las posiciones que se definen son las siguientes:

- Subir y avanzar con pie derecho (Ecuación (4.12))
- Bajar y avanzar con pie derecho (Ecuación (4.13))
- Subir y avanzar con pie izquierdo (Ecuación (4.14))
- Bajar y avanzar con pie izquierdo (Ecuación (4.15))

En el vector (4.12), se presentan las posiciones de cada articulación para la primer postura:

$$\begin{aligned} RLeg_{UP_T} =& (1.830, 0.270, -1.498, -0.519, 4.300e-11, -0.003, -0.125, \\ & 0.742, -0.617, 0.003, 4.300 \times 10^{-11}, -0.003, -0.576, \\ & 1.345, -0.769, 0.003, 1.839, -0.269, 1.498, 0.515) \end{aligned} \tag{4.12}$$

En la ecuación (4.13), se presentan las posiciones angulares para la segunda postura:

$$\begin{aligned} RLeg_{EXTEND_T} =& (1.825, 0.269, -1.498, -0.523, -4.362 \times 10^{-11}, -0.003, \\ & 0.178, 0.436, -0.614, 0.003, -4.362 \times 10^{-11}, -0.003, -0.357, \\ & 0.887, -0.530, 0.003, 1.835, -0.270, 1.498, 0.517) \end{aligned} \tag{4.13}$$

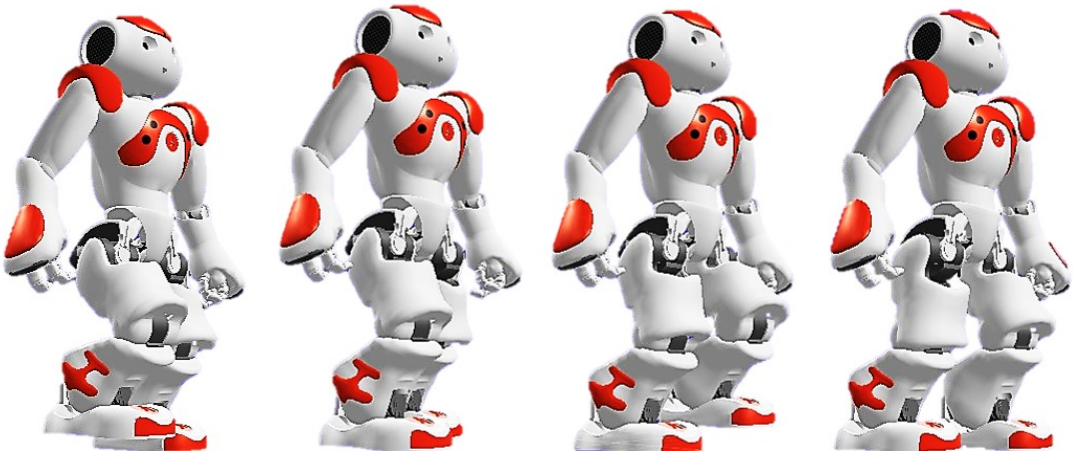
La ecuación (4.14) corresponde a las posiciones angulares para la tercera postura:

$$\begin{aligned} LLeg_{UP_T} =&(1.839, 0.269, -1.4981, -0.515, 1.564 \times 10^{-10}, -0.003, -0.568, \\ &1.329, -0.760, 0.003, 1.564 \times 10^{-10}, -0.003, -0.116, \\ &0.724, -0.607, 0.003, 1.830, -0.270, 1.498, 0.520) \end{aligned} \tag{4.14}$$

Finalmente, las posiciones angulares para la cuarta postura se presentan en (4.14):

$$\begin{aligned} LLeg_{EXTEND_T} =&(1.835, 0.270, -1.498, -0.517, 2.325 \times 10^{-11}, -0.003, \\ &-0.345, 0.864, -0.518, 0.003, 2.325 \times 10^{-11}, -0.003, 0.192, \\ &0.406, -0.598, 0.003, 1.825, -0.269, 1.498, 0.523) \end{aligned} \tag{4.15}$$

La Figura 4.3 ilustra las poses descritas anteriormente.



(a) Postura 1 (b) Postura 2 (c) Postura 3 (d) Postura 4

Figura 4.3: Posturas definidas para el ciclo de marcha

4.2.2. Creación del ambiente para RL

Bajo este enfoque, los elementos principales de nuestro sistema RL son el valor de rigidez, la velocidad máxima permitida a las articulaciones en función de su porcentaje [0-1] y el tiempo de espera entre envío de comandos acotado entre [0-1] segundos, estos serán tanto las acciones como las observaciones, pues con el fin de evitar el muestreo constante solo se tiene una realimentación final, por medio de la cual se realiza el cálculo de la recompensa.

- Observaciones:
  Se definen como un espacio tipo *Box* dado por el vector V^{Obs}. El vector de observación V^{Obs} se muestra en la ecuación (4.16)

$$V^{Obs} = \begin{bmatrix} k & Vj_{mx} & t_w \end{bmatrix} \tag{4.16}$$

  Donde:

$k =$ Valor del parámetro de rigidez
$Vj_{mx} =$ Velocidad para el movimiento de las articulaciones en función de su velocidad máxima
$t_w =$ Tiempo de espera entre ejecución de posturas, dado en segundos.

Lon límites inferiores están dados por el vector:

$$V_{low}^{Obs} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Y los limites superiores definidos por el vector:

$$V_{high}^{Obs} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix}^T$$

Estos límites corresponden a los extremos del rango de valores válido para cada componente del vector.

- Acciones:
  Las acciones en este caso son de la misma naturaleza que de las observaciones y se definen como un espacio tipo *Box*. El vector de acciones de la ecuación (4.17) está compuesto de la misma forma que el vector (4.16), así:

$$V^{Act} = \begin{bmatrix} k & Vj_{mx} & t_w \end{bmatrix} \tag{4.17}$$

  Los límites inferiores están dados por el vector:

$$V_{low}^{Act} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \end{bmatrix}$$

  Y los limites superiores están dados por el vector:

$$V_{high}^{Act} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix}$$

- Función de recompensa:

  La función de recompensa toma como referencia la función propuesta en [31]. Considerando las condiciones de desviaciones laterales y de rotaciones fuera de los límites como una "caída", dando por terminado un episodio y penalizando tal evento, con el fin de que el agente aprenda a evitarlo.

  La función de recompensa propuesta está dada por la ecuación (4.18):

$$\begin{aligned} r_i = & -10 * \left(\frac{1}{V_{vel}}\right) - 10 * \left(\frac{1}{D_x}\right) - 1000 * w_k - 100000 * w_{rx} \\ & - 100000 * w_{ry} - 100000 * w_h - 100000 * w_{lat} \end{aligned} \tag{4.18}$$

  Donde:

V_{vel} = Velocidad de avance en X
D_x = Distancia recorrida en el eje x
w_k = Penalización por condición asociada a valor de rigidez demasiado pequeño
w_{rx} = Penalización por condición asociada a rotación en x
w_{ry} = Penalización por condición asociada a rotación en y
w_h = Penalización por condición asociada a desviación vertical
w_{lat} = Penalización por condición asociada a desviación lateral

En esta función de recompensa propuesta se penalizan los inversos de V_{vel} y de Dx con el fin de que el agente busque maximizar estos valores a través de las acciones propuestas. Se penaliza también la elección de valores de rigidez inferiores a 0.3, ya que por debajo de este valor el robot no tiene suficiente torque en sus motores para mantenerse en pie o ejecutar movimientos a posiciones deseadas. Finalmente, las variables w_k, w_{rx}, w_{ry}, w_h y w_{lat}, asociadas a condiciones de paro, se castigan con mayor peso, puesto que indican desviaciones o desplazamientos equivalentes a la "caída" del robot.

- Condiciones de paro (*"done"*):
  Se definen las condiciones de paro para las cuales se considera que el sistema ha perdido el equilibrio o salido de los límites en que se desea que realice su movimiento:

  - Condición de paro por duración máxima de episodios.
  - Condición de paro por desviación lateral ≥ 15 cm.
  - Condición de paro por posición vertical del torso ≤ 30 cm.
  - Condición de paro por rotación en $|WX| \geq 0.15$ rad.
  - Condición de paro por rotación en $|WY| \geq 0.15$ rad.

  En el programa se definen w_k, w_{rx}, w_{ry}, w_h y w_{lat} para su condición de paro correspondiente.

Capítulo 5

Resultados

5.1. Implementación de RL para la generación de movimientos articulares basado en un base behavior

En este capítulo se presentan los resultados de la implementación de los enfoques descritos en el Capítulo 4, usando para ello el robot NAO con ayuda de un mecanismo tipo arnés para evitar caídas y daños al robot durante su entrenamiento. El polígono de soporte es el área formada por los puntos de contacto con el suelo. Si la proyección del Centro de masa (COM, por sus siglas en inglés) se encuentra dentro del polígono de estabilidad, se dice que la condición de equilibrio estático se cumple.[5]

El robot tiene por default un asistente de caída que detecta cuando la proyección del COM del robot en $z = 0$ se encuentra fuera de su polígono de soporte, provocando una reducción del torque en los motores para disminuir la velocidad de caída y amortiguar el impacto, prolongando así su tiempo de vida. Es necesario desactivar dicho asistente para permitir la exploración de posturas en el límite de estabilidad o no estables durante los entrenamientos realizados.

5.1.1. Seguimiento de trayectoria para una pierna

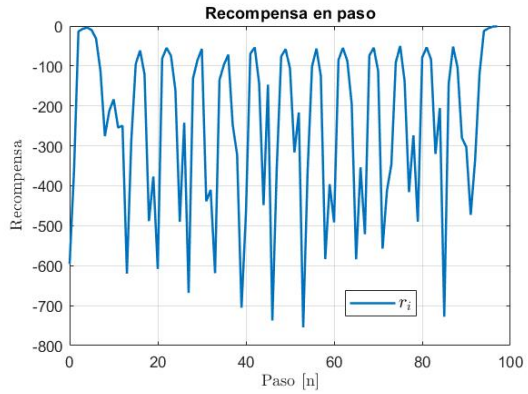
El entrenamiento se realiza utilizando un solo agente de tipo DDPG, con una red de política tipo "MlpPolicy", una arquitectura para las redes del actor y el crítico personalizada que consta de dos capas, con 500 y 400 neuronas, respectivamente.

Al personalizar una arquitectura de red, se agrega una capa lineal sobre la arquitectura definida para hacer coincidir las dimensiones de salida con la de las funciones de activación. Esta arquitectura de red, y las que se han de usar a lo largo de este capítulo, toma como referencia la arquitectura utilizada en [31], donde proponen una arquitectura de red de dos capas con 300 y 400 neuronas, respectivamente, para controlar un total de 6 articulaciones en ambas piernas de un humanoide que se entrena mediante RL, una razón de aprendizaje, que determina el tamaño del paso por iteración en la dirección de avance en la búsqueda de un mínimo,, de 1×10^{-3}, una función *eval_callback* que evalúa periódicamente el desempeño del agente durante el entrenamiento y guarda el mejor modelo. El resto de los parámetros del agente mantienen sus valores predeterminados, y se pueden consultar en la documentación del algoritmo DDPG [32].

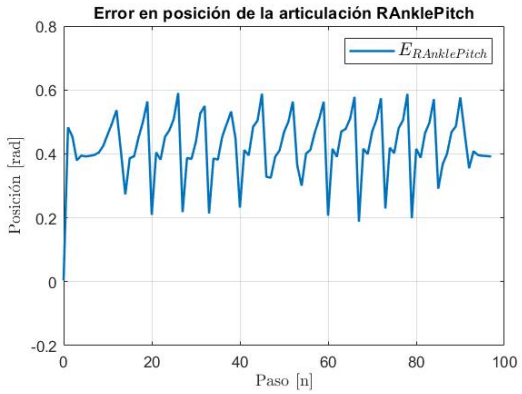
Después del entrenamiento del agente se cuenta con un buffer de experiencia de 1776 transiciones. El video se muestra en el siguiente enlace: Seguimiento de trayectoria para una pierna.

En la Figura 5.1 (a) se grafica la señal de recompensa obtenida en cada paso del tiempo. El valor acumulado de la recompensa obtenida a lo largo del episodio es de $-24,320.9$ y sus unidades son adimensionales.

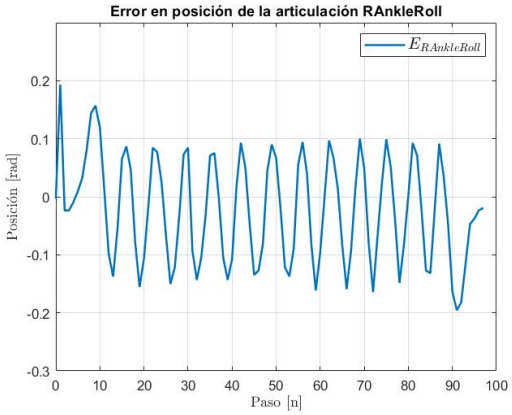
Las Figuras 5.1 (b), (c), (d), (e) y (f) muestran los errores en las pociciones de las articulaciones en cada paso durante la ejecución.



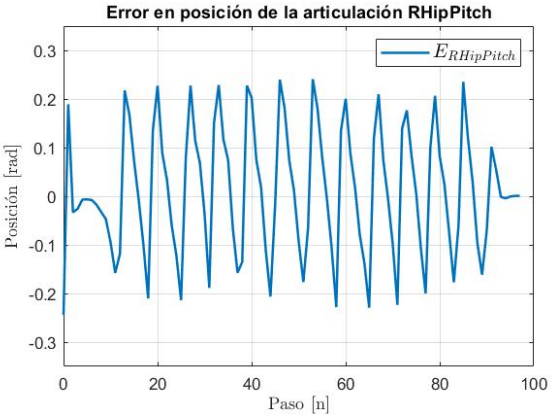
(a) Señal de recompensa en paso



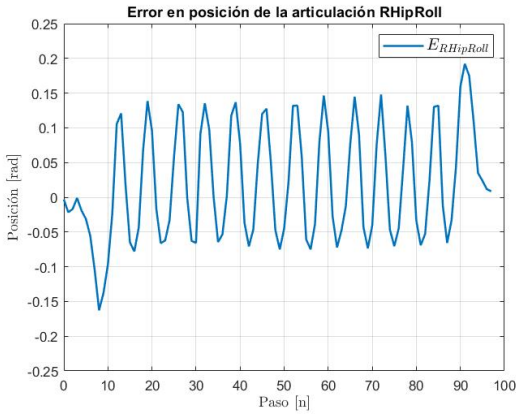
(b) Error en posición de RAnklePitch



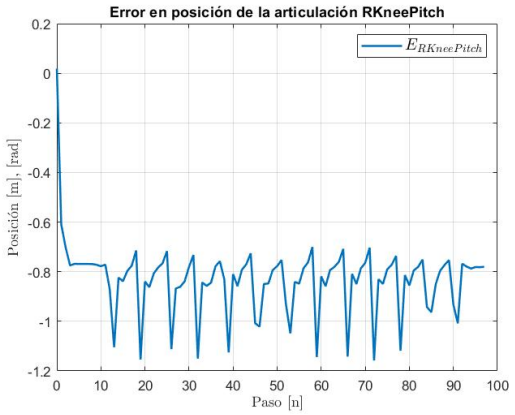
(c) Error en posición de RAnkleRoll



(d) Error en posición de RHipPitch



(e) Error en posición de RHipRoll



(f) Error en posición de RKneePitch

Figura 5.1: Gráficas de recompensa y errores em posiciones angulares del seguimiento de trayectoria para una pierna

En las gráficas de error se observa que los mayores errores se presentan en las articu-

laciones “RAnklEPitch” y “RKneePitch”. En el caso de esta última, el error se presenta hasta en casi -1.2 radianes y se mantiene entre $(-1.2, -0.7)$, esto coincide con ser la articulación que presenta calentamientos durante las pruebas experimentales y la responsable de que exista una desviación en WZ en el caminado en línea recta, pues es una falla recurrente en este modelo del robot. El error en el resto de las articulaciones se encuentra alrededor de (-0.2, 0.2) rad.

La Figura 5.2 muestra los extremos del seguimiento de trayectoria a través de fotogramas clave.

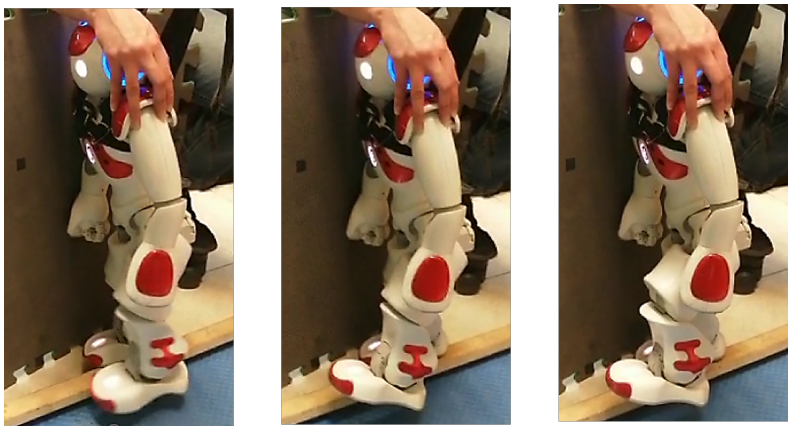


Figura 5.2: Fotogramas clave del seguimiento de trayectoria

En general, estos errores pueden disminuir continuando con el entrenamiento.

La comunicación asíncrona es un obstáculo a la hora de realizar entrenamientos, pues debido a esto, el entrenamiento mediante la plataforma de comunicación establecida no es estable, tiende a ser interrumpido e, incluso, en algunos casos llega a provocar que el servidor falle. Tal es el caso del comportamiento obtenido y presentado en el enlace: Seguimiento amplio. En la ejecución en cuestión, el servidor falla durante el entrenamiento antes de guardar la política y el modelo. Debido a que el entrenamiento prueba acciones estocásticas, la reproducibilidad de los resultados no se puede asegurar.

La Figura 5.3 muestra los extremos del seguimiento de trayectoria de dicha ejecución a través de fotogramas clave.

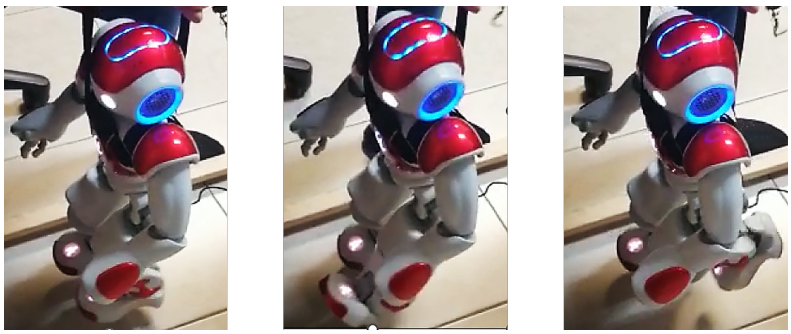


Figura 5.3: Fotogramas clave del seguimiento de trayectoria

5.1.2. Seguimiento de trayectoria para dos piernas

El entrenamiento se realiza utilizando la misma configuración para el agente que en la sección anterior, debido a que la arquitectura en [31] está pensada para controlar las piernas del humanoide en su modelo virtual, y en este caso también se trabaja con las piernas del humanoide. Tomando en cuenta que en este caso se maneja un número mayor de articulaciones que en la referencia tomada, se agregan 100 neuronas en cada capa tanto para el actor como para el crítico. Después de entrenar el agente, se tiene un buffer de experiencia de 4968 transiciones.

Como resultado de este proceso de entrenamiento, el robot busca avanzar hacia adelante, realizando una apretura de piernas con un movimiento rápido, activando una condición de paro correspondiente a la pérdida de equilibrio. Dicho movimiento asemeja a lo que el ser humano realiza como "desplante". En este caso se mide la distancia comprendida de punta a punta entre los pies del robot, la cual es de casi 28 cm, maximizando así la distancia de avance. La Figura 5.4 ilustra el comportamiento descrito anteriormente.

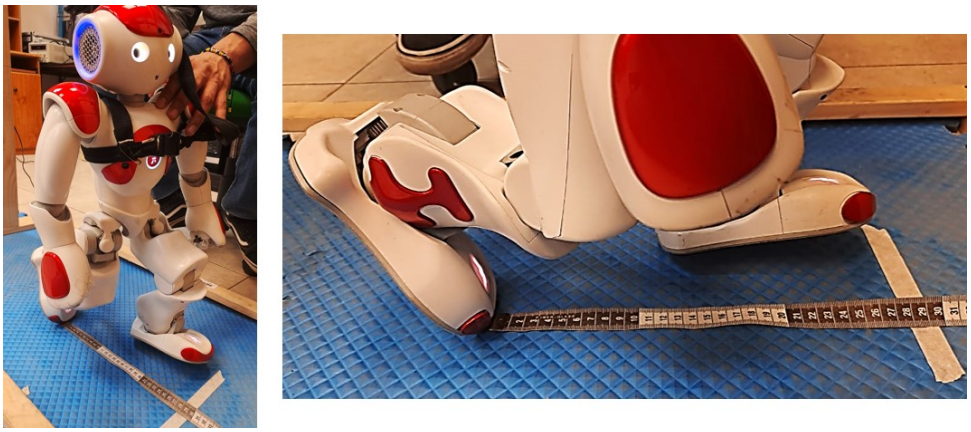


Figura 5.4: Desplante

Como parte de la validación de un ambiente personalizado es posible realizar la prueba de un agente no definido y no entrenado, que prueba acciones al azar. Dentro de esta validación las acciones observadas mostraban, en general, una tendencia a avanzar hacia adelante.

5.2. Implementación de RL para el ajuste de la rigidez basado en la observación de tiempo y distancia recorrida en un ciclo de marcha

Para el entrenamiento de este enfoque se utilliza un solo agente TD3, con una política "MlpPolicy", una arquitectura para las redes del actor y el crítico personalizada que consta de dos capas, con 400 y 300 neuronas, respectivamente y una razón de aprendizaje de 1×10^{-3}. El resto de los parámetros del agente mantienen sus valores predeterminados, y se pueden consultar en la documentación del algoritmo TD3 [33].

Debido a la forma en que se plantea la función de recompensa es necesario agregar un par de condiciones para evitar que la función se indetermine al presentarse una división por cero.

El vector propuesto como acción por el agente una vez entrenado es:

$$V_{low}^{Act} = \begin{bmatrix} 0.818 & 0.688 & 0.221 \end{bmatrix} \tag{5.1}$$

El video se muestra en el siguiente enlace: Enfoque de rigidez. Tras la ejecucción, se obtienen los siguientes resultados:

Se recorrió una distancia de **0.358 m.** en un tiempo de **2.5032 s.**. Con esto se obtiene una velocidad de avance de **0.1430** m/s. Y la recompensa obtenida fue de -100097.8521.

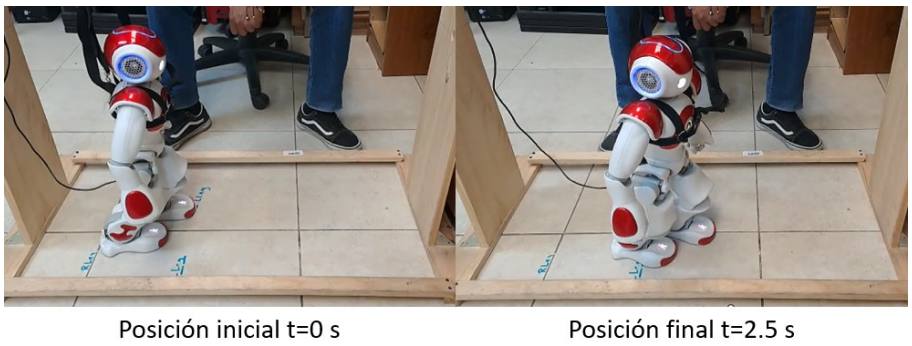


Figura 5.5: Implementación del enfoque de rigidez

En la Figura 5.5 se muestran la posición inicial y final del robot, se observa una ligera desviación lateral respecto a la posición inicial, al final del caminado, la cual se puede asociar con el funcionamiento inadecuado de la articulación de la rodilla derecha.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Se optó por el uso de técnicas de Aprendizaje Reforzado por las ventajas que representan los métodos libres de modelo, considerando que el modelo dinámico asociado al robot trabaja con 25 GDL. Se desarrolló una plataforma para el proceso de comunicación entre un programa de RL y el robot NAO, obteniendo tiempos de muestreo de alrededor de 100 ms. Se abordaron dos enfoques distintos para la optimización del caminado de un robot humanoide. En el primero de ellos se tiene una retroalimentación constante, se controlan las posiciones angulares de las articulaciones que se seleccionan y se toma como refererencia el caminado por el comando preprogramado del robot, para lo que se obtuvo la caracterización de dicho caminado; en el segundo enfoque, se considera una retroalimentación al final de la ejecución del programa, se controla el valor de rigidez, la velocidad máxima para el movimiento de las articulaciones y el tiempo de espera entre envío de comandos y se definen posturas para generar un ciclo de marcha. Se crearon ambientes personalizados dentro de un entorno virtual para la implementación de las técnicas de RL en el robot, para cada uno de los enfoques abordados. Se seleccionaron y probaron agentes para cada enfoque. Se probaron funciones de recompensa, ajustando los pesos y los objetivos buscados en cada una hasta llegar a las propuestas en este documento, correspondientes a aquellas mediante las que se obtuvieron los mejores resultados. Se construyó una estructura de riel con arnés para evitar daños por caídas durante los entrenamientos. Estos se realizaron en primer instancia con el robot virtual, sin embargo, al implementar en el robot físico los resultados de los entrenamientos realizados con el robot virtual, el comportamiento obsevado difería entre uno y otro. Esto se debe a que el modelo virtual no considera los efectos del valor de rigidez, de la gravedad y de la fricción de la superficie en la que se trabaje. Se realizaron entrenamientos con el robot real, encontrando limitaciones físicas por parte del robot, tales como la duración de la batería y el calentamiento en los motores de algunas de sus articulaciones, en especial en la rodilla derecha.

La comunicación asíncrona provoca interrupciones durante el entrenamiento, dificultando entrenamientos largos.

El caminado del robot NAO se puede optimizar maximizando la distancia recorrida por paso, tal como se logra en el enfoque 1 con el control de ambas piernas; o maximizando la velocidad de avance, lo cual se logra en el enfoque 2.

Para el enfoque 1 se considera que el desplante obtenido maximiza la distancia de avance por paso, sin embargo, al lograr solo 1 paso y llegar a una condición de paro la ejecución termina. Aunado a esto, durante la validación del ambiente personalizado se observa al robot realizar movimientos erraticos orientados al avance en el eje x, con lo cual se puede decir que la función de recompensa plantea de forma adecuada el objetivo de avace.

Para el enfoque 2, los resultados muestran un caminado con una velocidad de **0.143 m/s**, la cual es mayor que la que alcanza el caminado con la configuración predeterminada para una distancia de 1 metro, la cual es de 0.1 m/s. Con esto en consideración, el objetivo de optimizar la marcha se logra al tener una velocidad mayor, teniendo la velocidad como criterio de optimización.

Para realizar una comparación entre estos dos enfoques es necesario tener en cuenta que no es posible compararlos cuantitativamente, puesto que se tienen funciones de recompensa diferentes en cada enfoque. A pesar de esto sí es posible compararlos cualitativamente, considerando el logro del caminado como criterio de comparación. Bajo esta premisa, el segundo enfoque resulta tener un mejor desempeño que el primero, al sí lograr el caminado del robot.

6.2. Trabajo futuro

De forma general, se recomienda la búsqueda de una solución para la inestabilidad asociada a la comunicación asíncrona o buscar una plataforma en la cual no se presenten tiempos de muestreo tan elevados.

Una alternativa a los tiempos de entrenamiento requeridos y a las limitaciones físicas del robot es la implementación de modelos subrogados que permiten entrenamientos virtuales y validaciones físicas, o bien, el uso de un gemelo virtual o robot virtual en el que se consideren los efectos de la rigidez, gravedad y, de ser posible, fricción, permitiendo entrenamientos combinados, que a la hora de implementar en el robot físico o virtual no presenten diferencias considerables.

Para el primer enfoque se recomienda continuar con el entrenamiento para obtener mejores resultados. Además de esto, se recomienda probar con multiagentes que controlen cada una de las extremidades o conjuntos de ellas de forma independiente. Se hace hincapié en que dado el número de articulaciones que se están controlando en este enfoque, especialmente si se desea agregar el movimiento de los brazos en una continuación de este trabajo, el número de estados posibles y el número de acciones posibles incrementa considerablemente respecto a las 5 articulaciones con las que se realiza el seguimiento de trayectoria, por ende, el tiempo de entrenamiento requerido ha de resultar mayor.

Para el segundo enfoque se pueden optimizar las poses que conforman la marcha generada, ya sea para obtener una marcha más natural o más estable, que permita recorrer una mayor distancia. También, existe la posibilidad de definir valores para el parámetro de rigidez por cada extremidad o, con un mayor número de acciones de control, para cada articulación, con lo que se podría optimizar el consumo de energía.

Apéndice A

Apéndice A.Articulaciones, rangos de movimiento.

La Figura A.1 muestra las articulaciones de la cabeza del robot y su rango de movimiento.

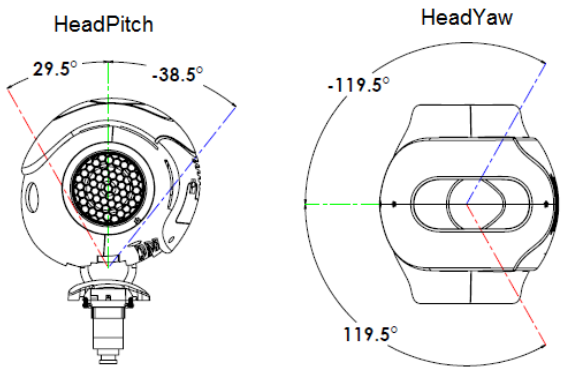


Figura A.1: Rango de movimiento de las articulaciones HeadPitch y HeadYaw

La Figura A.2 muestra las articulaciones de la pelvis del robot y su rango de movimiento.

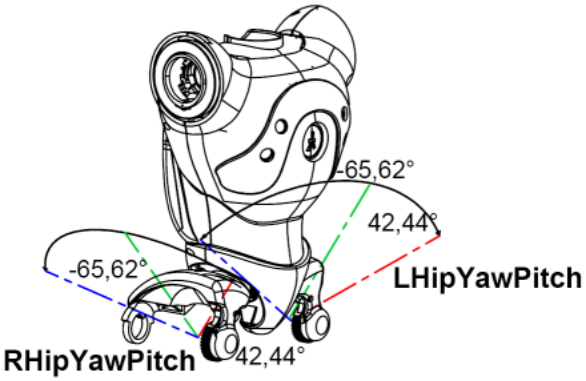


Figura A.2: Rango de movimiento de las articulaciones RHipYawPitch y LHipYawPitch

La Figura A.3 muestra las articulaciones del brazo izquierdo del robot y su rango de movimiento.

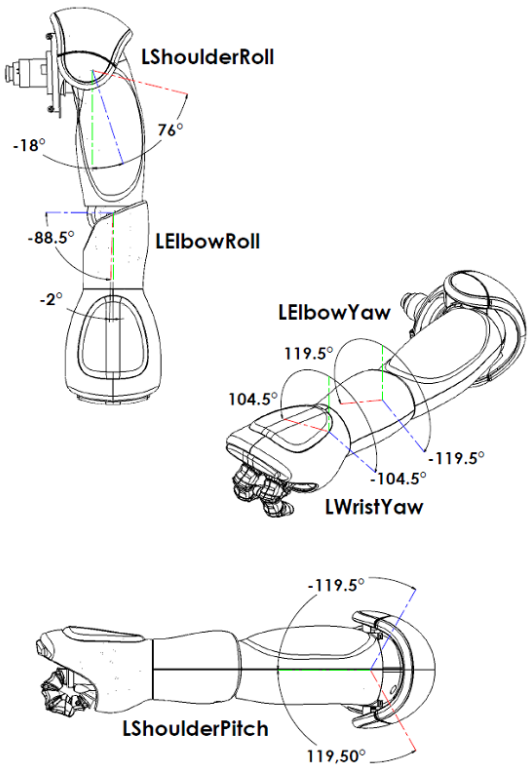


Figura A.3: Rango de movimiento de las articulaciones LShoulderRoll, LElbowRoll, LElbowYaw, LWristYaw y LShoulderPitch

La Figura A.4 muestra las articulaciones del brazo derecho del robot y su rango de movimiento.

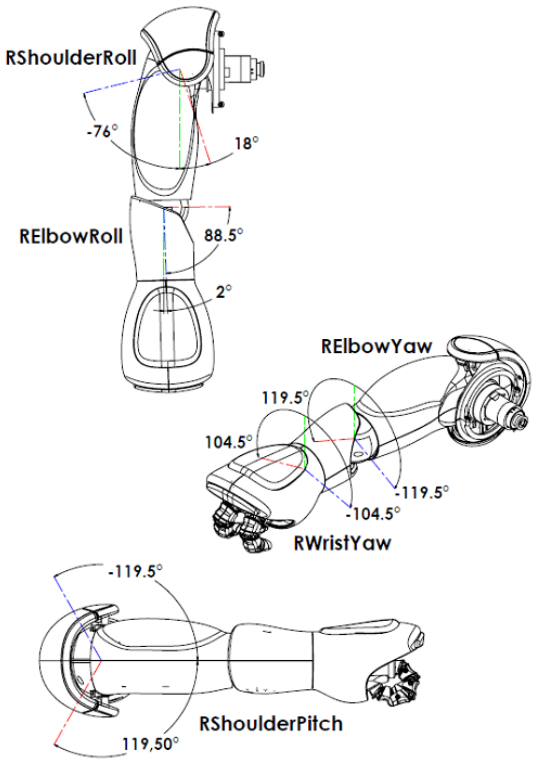


Figura A.4: Rango de movimiento de las articulaciones RShoulderRoll, RElbowRoll, RElbowYaw, RWristYaw y RShoulderPitch

La Figura A.5 muestra las articulaciones de la pierna izquierda del robot y su rango de movimiento.

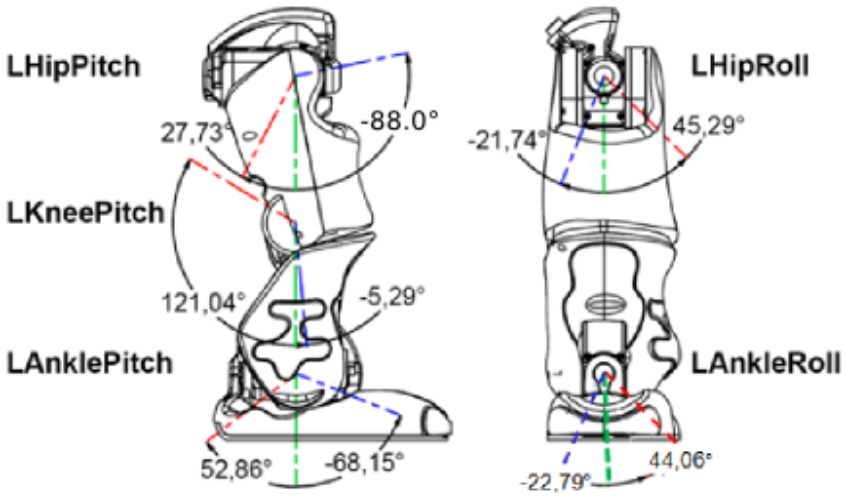


Figura A.5: Rango de movimiento de las articulaciones LhiPitch, LKneePitch, LAnkle-Pitch, LHipRoll y LAnkleRoll

La Figura A.6 muestra las articulaciones de la pierna derecha del robot y su rango de movimiento.

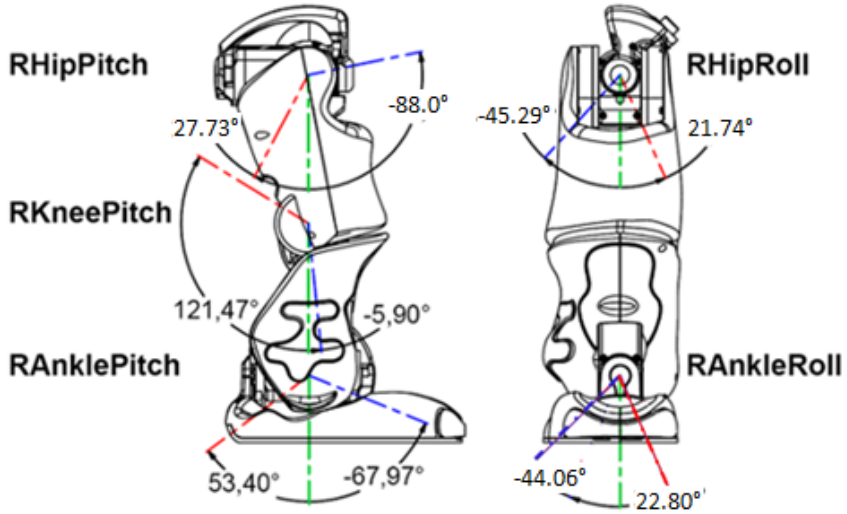


Figura A.6: Rango de movimiento de las articulaciones RhiPitch, RKneePitch, RAnkle-Pitch, RHipRoll y RAnkleRoll

Bibliografía

[1] A. . U. R. Group, "Pepper y nao, robots para la enseñanza," https://www.aldebaran.com/es/pepper-and-nao-robots-education, 2022.

[2] I. Kato, S. Ohteru, H. Kobayashi, K. Shirai, and A. Uchiyama, *Information-Power Machine with Senses and Limbs.* Vienna: Springer Vienna, 1974, pp. 11–24. [Online]. Available: https://doi.org/10.1007/978-3-7091-2993-7_2

[3] M. Vukobratovic and B. Borovac, "Dynamic balance concept and the maintenance of the dynamic balance in humanoid robotics," in *2008 6th International Symposium on Intelligent Systems and Informatics*, 2008, pp. 1–11.

[4] T. K. Tsutomu Mita, Toru Yamaguchi and T. Kawase, "Realization of a high speed biped using modern control theory," *International Journal of Control*, vol. 40, no. 1, pp. 107–119, 1984.

[5] E. J. Ortiz Medrano, *Sistema interactivo de robots humanoides.* Tesis de maestría, CINVESTAV Unidad Zacatenco, 2021.

[6] "Honda," https://www.honda.mx/asimo, 2023.

[7] B. Dynamics, "Atlas," https://www.bostondynamics.com/atlas, 2023.

[8] B. B. Alexander Zai, *Deep Reinforcement Learning in Action.* Manning Publications, 2020.

[9] M. Abreu, N. Lau, A. Sousa, and L. P. Reis, "Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2019, pp. 1–8.

[10] T. Hester, M. Quinlan, and P. Stone, "Generalized model learning for reinforcement learning on a humanoid robot," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2369–2374.

[11] X. Wu, S. Liu, T. Zhang, L. Yang, Y. Li, and T. Wang, "Motion control for biped robot via ddpg-based deep reinforcement learning," in *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, 2018, pp. 40–45.

[12] C. Kouppas, M. Saada, Q. Meng, M. King, and D. Majoe, "Hybrid autonomous controller for bipedal robot balance with deep reinforcement learning and pattern generators," *Robotics and Autonomous Systems*, vol. 146, p. 103891, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889021001767

[13] O. S. K. L. Hyobin Jeong, Inho Lee and J.-H. Oh, "A robust walking controller optimizing step position and step time that exploit advantages of footed robot," *Robotics and Autonomous Systems*, vol. 113, pp. 10–22, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018305372

[14] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.

[15] F. Garcia, J. y Fernandez, "Safe exploration of state and action spaces in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 45, p. 515–564, 2012.

[16] J. García and D. Shafie, "Teaching a humanoid robot to walk faster through safe reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 88, p. 103360, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197619302921

[17] A. G. Sutton, R. S. y Barto, *Reinforcement Learning: An Introduction.* London, England: A Bradford Book, 2018.

[18] S. Ravichandiran, *Hands-On Reinforcement Learning with Python.* Livery Place 35 Livery Street Birmingham B3 2PB, UK.: Packt Publishing Ltd, 2018.

[19] L. Graesser and W. Keng, *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*, ser. Addison-Wesley data and analytics series. Addison-Wesley, 2020.

[20] OpenAI, "Part 2: Kinds of rl algorithms," https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html, 2018.

[21] "Rl algorithms," https://stable-baselines.readthedocs.io/en/master/guide/algos.html, 2021.

[22] S. Robotics, "Nao - motors," http://doc.aldebaran.com/1-14/family/robots/motors_robot.html, 2006.

[23] ——, "Glossary," http://doc.aldebaran.com/1-14/glossary.html, 2006.

[24] ——, "Robot simulations¶," http://doc.aldebaran.com/1-14/dev/tools/robot-simulation.html#choregraphe-reference-simulated-robot, 2006.

[25] ——, "Naoqi on the robot," http://doc.aldebaran.com/1-14/dev/tools/naoqi.html#what-is-naoqi, 2006.

[26] P. S. Foundation, "Idle," https://docs.python.org/es/3/library/idle.html, 2023.

[27] I. GitHub, "Gym," https://github.com/openai/gym, 2023.

[28] P. S. Foundation, "Glosario," https://docs.python.org/es/3/glossary.html#term-virtual-environment, 2023.

[29] "Gym documentation," https://www.gymlibrary.dev/content/basic_usage/, 2022.

[30] T. M. Inc., "Train ddpg agent to swing up and balance cart-pole system," https://la.mathworks.com/help/reinforcement-learning/ug/train-ddpg-agent-to-swing-up-and-balance-cart-pole-system, 2022.

[31] "Train humanoid walker," https://la.mathworks.com/help/sm/ug/humanoid_walker, 2022.

[32] "Ddpg," https://stable-baselines3.readthedocs.io/en/master/modules/ddpg.html, 2022.

[33] "Ddpg," https://stable-baselines3.readthedocs.io/en/master/modules/td3.html, 2022.