APRENDIZAJE DE MOVIMIENTOS EN ROBOT HUMANOIDE A PARTIR DE INFERENCIA DE OBJETIVOS

YEISON ESTIVEN SUAREZ HUERTAS

UNIVERSIDAD SANTO TOMÁS
INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C
2020

APRENDIZAJE DE MOVIMIENTOS EN ROBOT HUMANOIDE A PARTIR DE INFERENCIA DE OBJETIVOS

YEISON ESTIVEN SUAREZ HUERTAS

Proyecto de grado presentado como requisito para optar al título de
INGENIERO ELECTRÓNICO

UNIVERSIDAD SANTO TOMÁS
INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C
2020

UNIVERSIDAD SANTO TOMÁS

Ingeniería Electrónica

Este proyecto de grado fue aceptado para la carrera de Ingeniero(a) Electrónico(a) en Junio 2019

Director: Edgar Camilo Camacho Poveda
Magíster en Ingeniería Electrónica
Universidad Nacional
Bogotá, Colombia

Co-directora: Carolina Higuera Arias
Magíster en Ingeniería Electrónica y de Computadores
Universidad de los Andes
Bogotá, Colombia

Jurados: Juan Manuel Calderón Chavez
Doctorado en Ingeniería Eléctrica
University of South Florida
Florida, Estados Unidos

Carlos Andrés Torres Pinzón
Doctorado en Ingeniería Electrónica, Automática y Comunicaciones
Universidad Rovira i Virgili
Tarragona, España

Sustentación proyecto de grado: ____ de Junio de 2020, BOGOTÁ D.C

Yeison Estiven Suarez Huertas _____

NOTA DE ACEPTACIÓN

El trabajo de grado titulado ***APRENDIZAJE DE MOVIMIENTOS EN ROBOT HUMANOIDE A PARTIR DE INFERENCIA DE OBJETIVOS****, realizado por el estudiante* ***YEISON ESTIVEN SUAREZ HUERTAS****, cumple con los requisitos exigidos por la* ***UNIVERSIDAD SANTO TOMÁS*** *para optar al título de* ***INGENIERO ELECTRÓNICO.***

Juan Manuel Calderón Chavez _____

Edgar Camilo Camacho _____

DEDICATORIA

A mi madre, que con su apoyo incondicional logró que me convirtiera en un hombre de bien para la sociedad; a mis docentes que con sus conocimientos me brindaron la posibilidad de expandir mis horizontes y a mis amigos mas cercanos que me brindaron el apoyo emocional para seguir adelante en la vida.

AGRADECIMIENTOS

Agradezco a mi madre, Luz Nélida, por ofrecerme la oportunidad de estudiar con todas las comodidades que con gran esfuerzo pudo ofrecerme, a todos mis docentes de universidad, principalmente a los profesores Camilo y Carolina quienes vieron en mí las capacidades necesarias para lograr este trabajo de grado. También, agradezco a todos mis amigos dentro de la universidad como José, Andrea, Alison y Juan Sebastián; y a mis amigos cercanos, Laura, Lina, Yohans y Jhon, todos ellos son prueba para mí de que el ser humano nunca va a estar completo en su vida si no hay personas valiosas con quienes vivirla.

Índice general

Índice de cuadros

Índice de figuras

Resumen

Este documento presenta la aplicación del aprendizaje por refuerzo inverso (*IRL* por sus siglas en inglés) en un robot humanoide, con el fin de realizar movimientos de las extremidades superiores.

Para verificar el funcionamiento del algoritmo, se seleccionó el robot *Poppy Torso*, el cual es parte de un proyecto más grande y de código abierto conocido como *The Poppy Proyect*. Dicho prototipo fue construido en el Grupo de Estudio y Desarrollo en Robótica (GED), y se realizó la programación para crear una interfaz con un computador. Además, se creó el modelo de simulación basado en la cinemática obtenida de los modelos contenidos en los repositorios del proyecto.

El *IRL* se basa en el aprendizaje a partir de las demostraciones (trayectorias) de un experto. Con el fin de obtener una utilidad final lo más cercana a la utilidad obtenida por el experto en su recorrido, previamente se implementa un aprendizaje por refuerzo (*RL* por sus siglas en inglés) con una recompensa plenamente establecida dentro del entorno diseñado, el cual logró cumplir el objetivo que corresponde a generar movimientos desde un punto aleatorio hasta un punto establecido.

A continuación, se tomaron dichas trayectorias, y se aplicaron al *IRL*, el cual desconoce la función de recompensa original. A partir del entrenamiento, el sistema logra, además de ejecutar la tarea, inferir la recompensa adecuada para lograr el objetivo implícito en las trayectorias del experto.

Finalmente, se obtienen como resultados 2 modelos y un arreglo de pesos, el primer modelo obtenido a través de *RL* utilizado para generar

trayectorias y el segundo modelo obtenido de los entrenamientos dentro del entorno conociendo únicamente las trayectorias del experto. La función del entorno no es la misma que se usa para aprendizaje por refuerzo inverso, esto se explica con más detalle dentro del marco teórico y en el capítulo acerca del *IRL*.

El robot en simulación logra en la mayoría de los casos (con un porcentaje del 97.5 % realizado sobre 1000 pruebas) llegar a su objetivo, tanto por aprendizaje por refuerzo como por refuerzo inverso. El refuerzo inverso demuestra una utilidad final menor que la del refuerzo directo, y esto tiene una base matemática, ya que la diferencia de la utilidad del aprendiz con respecto al experto en el mejor de los casos es cero, por ende, el aprendiz no puede hacerlo mejor de lo que el experto puede.

Introducción

En la sociedad actual, estamos acostumbrados a trabajar con herramientas tecnológicas que nos faciliten las tareas y sean también las que realicen, en lo posible, todo el trabajo pesado. La robótica también es una de estas herramientas de suma importancia para mejorar la productividad y desarrollo humano, aspectos como la producción en cadena, recolección de cosechas, e incluso, avances médicos se dan gracias a máquinas que operan principalmente de dos maneras: operadas o programadas por los seres humanos. En este trabajo se busca incursionar en una tercera forma de lograr esto, la cual corresponde al aprendizaje de la máquina a partir de las demostraciones presentadas por un experto.

El objetivo principal del trabajo es poder diseñar un algoritmo de aprendizaje por refuerzo inverso (*IRL* por sus siglas en inglés) basado principalmente en los trabajos de Pieter Abbeel, Andrew Y. Ng y Sutton en [1, 5]. El IRL, a diferencia del aprendizaje por refuerzo, intenta observar otro agente conocido como experto y que tiene un objetivo por meta, es así, que el problema se traslada y no necesitamos conocer la recompensa que pueda entregar el entorno, sino uno o varios objetivos que se quieran cumplir. Pero, para realizar esta tarea se necesita de la trayectoria del agente experto, esto resulta muy conveniente cuando no se tiene clara la recompensa del entorno o cuando el entorno es tan grande y variado en cuestión, que es muy difícil lograr extraer una recompensa que sea efectiva para realizar una tarea que en principio no es tan compleja para un ser humano, por ende, es muy importante que las trayectorias que se han tomado del agente experto contengan un objetivo implícito. La

forma en la que se aborda la creación de este agente experto, es utilizar aprendizaje por refuerzo dentro de un entorno, diseñado matemáticamente a partir de la cinemática descrita en los archivos del robot *Poppy Torso*, y así, crear una función recompensa para el entorno, luego se pone a prueba el modelo diseñado y se almacenan los estados que realizó este agente experto entrenado, con el fin de ser la base sobre la cual va a funcionar el aprendizaje por refuerzo inverso.

Se explicarán inicialmente algunas de las temáticas correspondientes al aprendizaje por refuerzo, este tipo de aprendizaje está inspirado en la psicología conductista y al igual que en esta, se busca que un agente intente maximizar la recompensa (o el premio) de un entorno. Esta forma de entrenamiento se diferencia de algunos modelos actuales en los que se intenta imitar el comportamiento de una persona, animal u objeto, ya que implícitamente es el agente el que debe explorar su entorno y poder decidir, por su cuenta y basado en lo que va aprendiendo, qué acción tomar.

A continuación se presenta la construcción del robot humanoide *Poppy Torso*, este robot hace parte de un proyecto de código y equipo abierto, con lo cual es permitido el uso de sus modelos de simulación en 3D y de su programación para hacer pruebas en el laboratorio sobre un equipo físico. De ahí que, para la elección de esta plataforma, un aspecto clave fue la robustez del proyecto *The Poppy Project* [3]. Gracias a su documentación de la plataforma robótica y el uso del marco de trabajo *ROS*, se logra realizar una comunicación entre los equipos disponibles en el laboratorio y la placa de desarrollo que se instaló sobre el robot, cabe aclarar que esto abstrae la capa física para el algoritmo, ya que se pueden realizar los entrenamientos en simulación y lograr un resultado muy cercano en el robot físico. Aparte de la comunicación, se desarrollan otras interfaces que permiten una depuración del robot, además, se adaptó un robot futbolista del laboratorio que pertenece al grupo GED para dotar al robot humanoide con la capacidad de moverse por un entorno, la interfaz para la depuración del robot futbolista también fue incluida como parte del desarrollo del

trabajo y se explica más adelante.

Después, se muestran la metodología usada para la construcción del experto basado en Aprendizaje por Refuerzo (*RL* por sus siglas en inglés). Este modelo está basado en una red neuronal, a la que se le realizó un barrido de parámetros para determinar cuales entregaban los mejores resultados. Estos resultados son obtenidos al evaluar la red dentro del entorno, cuya tarea principal consiste en alcanzar un punto objetivo dentro del espacio de estados que el brazo del robot puede alcanzar. Luego de obtener el modelo entrenado, es posible extraer las trayectorias que toma y visualizar su comportamiento, también se visualiza el progreso del entrenamiento finalizado.

Partiendo del modelo entrenado por RL (Experto), se tomaron aquellas trayectorias que cumplieran con el objetivo implícito del entorno. Estas trayectorias fueron la base para implementar el algoritmo de IRL, cuyo modelo contaba con los mismos hiperparámetros del experto. La principal diferencia con éste es que no se obtiene la recompensa del entorno, ya que se asume que ésta no se conoce, en su lugar se establece un conjunto de características que se estima, hacen parte de la recompensa implícita, para que les sean atribuidos pesos ponderados, con el fin de que las utilidades obtenidas se acerquen a la del experto. Finalmente se obtiene el modelo entrenado por IRL, su progreso en cada actualización de pesos, gráficas para evaluar su recorrido hasta el punto objetivo y el arreglo de pesos para las funciones que componen la función recompensa.

La mayor parte del desarrollo del algoritmo y del documento en general, tuvo lugar en el laboratorio de robótica de la Universidad Santo Tomás, principalmente por los equipos con alta capacidad de cómputo disponibles y la ayuda brindada por los docentes calificados en las áreas de control, modelado, aprendizaje de máquinas e inteligencia artificial, todo ello, con el fin de lograr un avance significativo en esta área del conocimiento y lograr dejar material documentado y probado que pueda ser de ayuda para la facultad de ingeniería electrónica, así como, para el Grupo de Estudio y Desarrollo en Robótica (GED) perteneciente de la universidad.

Capítulo 1

Planteamiento del problema

Un robot puede realizar acciones que resulta muy difíciles o incluso imposibles para un ser humano, también son capaces de reaccionar a estímulos o situaciones con una rapidez superior, logrando alertar de situaciones potencialmente peligrosas antes de que las personas se percaten. Sin embargo, para que el robot pueda realizar determinadas acciones que cumplan con el objetivo deseado, se necesita de una o varias personas capacitadas en conocimientos de electrónica, programación, mecánica y cinemática con el fin de lograr la programación, esto genera que la forma en la que se le puedan cambiar acciones o movimientos puede estar fuertemente restringida y requiere que el usuario tenga también algunos conocimientos en estas áreas, debido a esto se puede generar una barrera entre la persona que necesita la ayuda del robot y el mismo robot.

Para cerrar esta barrera, el robot podría ser capaz de aprender a realizar determinadas acciones que se necesiten, de una forma más fácil, sin que se recurra a tener que programarlo para cada movimiento, y en lo posible, que cualquier persona sea capaz de hacerlo. Con esto, se puede ser más general en la elección del robot cuando no se necesiten tantos grados de libertad para él y el desarrollo del algoritmo también puede ser más general y transversal entre los robots usados.

Hoy en día existen algoritmos de aprendizaje de máquina, como por

ejemplo, aprendizaje por refuerzo o aprendizaje supervisado, sin embargo, estos algoritmos necesitan que el objetivo de la tarea a realizar sea planteado explícitamente de forma matemática. Existen tareas que resultan muy complejas por completar en un entorno y estos objetivos implícitos en las tareas resultan muy difíciles de plantear matemáticamente, aunque estos mismos objetivos pueden contenerse dentro de las acciones que realiza un experto. Por esta razón, resulta pertinente buscar un método que permita a un algoritmo de aprendizaje de máquina, inferir los objetivos que están implícitos dentro de un conjunto de demostraciones realizadas por el agente experto.

Con base al problema anterior, surge la pregunta de investigación: *¿Cómo inferir la función de recompensa y la política a partir de aprendizaje por refuerzo y aprendizaje por refuerzo inverso para llevar a cabo movimientos en las extremidades superiores de un robot humanoide?*

Capítulo 2

Justificación

Retomando la pregunta de investigación propuesta en el planteamiento del problema ¿porque se debería inferir una función de recompensa?, ya que, el objetivo principal del documento es que un algoritmo para un robot humanoide aprenda a realizar acciones y que, además, estos movimientos no sean programados con anterioridad. Se plantean problemas relacionados con aprendizaje de máquina para los cuales se requiere de un entorno en el cual sea la máquina quien aprende a moverse con determinadas acciones que tiene disponible, pero la forma en la que sabe que acción puede ser mejor que otra y que, finalmente cumpla con el objetivo o la tarea asignada, está directamente relacionada con la función de recompensa implícita dentro del entorno.

Está claro que, sin la recompensa del entorno no es posible realizar un entrenamiento que cuente con un desempeño destacable, entonces ¿qué pasa cuando no es posible extraer de forma implícita este valor numérico, que le permite al algoritmo conocer que tan bien realiza una tarea o más bien, que tarea debería hacer? Es en esta situación donde se aplican los conocimientos de un experto, pero no un experto en programación o robótica, sino un experto en realizar la tarea que se le quiere enseñar al robot humanoide, ya sea de un ser humano o de algo que conozca cual es el objetivo o la tarea a realizar. Porque es ahora que el aprendizaje se va a tomar desde los movimientos que el experto realice en el entorno, ya que implícito en lo

que el experto hace se encuentra la tarea que se quiere realizar. Este tipo de aprendizaje es conocido como aprendizaje por refuerzo inverso, sobre el cual se plantea la matemática necesaria para que un agente aprendiz logre realizar las tareas que realiza un agente experto, basándose en la trayectoria que toma dentro del entorno.

Gracias a que ahora el algoritmo puede aprender de las trayectorias de un experto, se requiere del entorno hablado anteriormente. El uso de un robot humanoide resulta especialmente útil a la hora de implementar el algoritmo de aprendizaje por refuerzo inverso, debido a que el robot puede observar a una persona y extraer la trayectoria que la misma tomó para completar una tarea, luego en esta trayectoria puede generar un espacio de acciones que sean acordes a su cinemática para finalmente, aprender a moverse de manera similar a la persona de quien aprendió, incluso, no es completamente necesaria una persona, puede utilizarse el algoritmo para aprender de otro robot o de algún modelo simulado que ya ha sido entrenado.

Dicho esto, para escoger el robot humanoide sobre el cual trabajar, se disponía de varias alternativas, pero finalmente se optó por usar los modelos del robot *Poppy Torso*, ya que pertenecen a un proyecto libre que cuenta con facilidades de uso en lenguajes de programación como *python* y ofrece también la opción de ser simulado en *Gazebo* y en *V Rep*. Aparte de las facilidades ofrecidas por la librería, se deben tener en cuenta los tiempos de entrenamiento, para lo cual se emplea la cinemática ofrecida en los repositorios del robot y de esa manera se logra construir un entorno más liviano sobre el cual corren los algoritmos de aprendizaje.

Capítulo 3

Objetivos

3.1. Objetivo General

Inferir la función de recompensa y la política a partir de aprendizaje por refuerzo y aprendizaje por refuerzo inverso, para llevar a cabo movimientos en las extremidades superiores de un robot humanoide.

3.2. Objetivos Específicos

- Participar en la construcción conjunta del robot humanoide Poppy-Torso, en la parte de la programación de motores y entorno, en el semillero SERO del grupo de investigación GED.
- Definir los estados, acciones, política de exploración y funciones características u objetivos que conforman la recompensa.
- Implementar un algoritmo para aprender la función de recompensa mediante la inferencia de los pesos de los objetivos que la conforman.
- Evaluar el desempeño de la política aprendida, comparando las acciones ejecutadas por el robot humanoide simulado con las del experto.

Capítulo 4

Marco Teórico

4.1. Proceso de decisión de Markov

Un proceso de decisión de Markov es un proceso de control estocástico de tiempo discreto en donde a través de un marco matemático se logra modelar la toma de decisiones en situaciones en las que existe una parte aleatoria y otra controlada.

Se busca que los MDP sean una forma sencilla de resolver el proceso de aprender por iteraciones para lograr un objetivo, con lo que se relaciona o interactúa se llama entorno, el agente y el entorno interactúan constantemente, el agente realiza acciones y el entorno responde a esas acciones generando nuevas situaciones para el agente, el entorno es también quien genera las recompensas, que son valores numéricos que el agente busca maximizar a través del tiempo por medio de sus acciones.

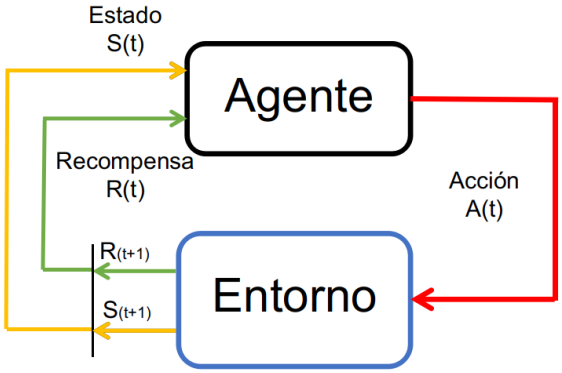


Figura 4.1. La interacción entre el agente y el entorno en un MDP [1]

Acción (A): La capacidad de poder realizar un cambio en el entorno donde se encuentra el aprendiz.

Estado (S): Es el entorno donde se encuentra el agente.

Recompensa (R): Valor numérico que indica cual es la tarea a aprender y el beneficio obtenido de una acción al llegar a un estado.

Para lograr definir un proceso de decisión de Markov es necesario:

- Un conjunto de estados $s \in S$
- Un conjunto de acciones $a \in A$
- Una función de transición de los términos de la probabilidad que de s conduzca a s'
- Una función de recompensa $R(s) \in \mathbb{R}$
- Un estado de inicio
- Un estado terminal (no necesariamente)

Un proceso es Markoviano si, para conocer la probabilidad de alcanzar el siguiente estado s', es suficiente con el estado actual y no es necesariamente la historia de estados anteriores.

4.2. Aprendizaje por refuerzo

El problema del aprendizaje por refuerzo puede ser descrito como un Proceso de Decisión de Markov, donde por cada instante de tiempo t, el agente recibe el estado de su entorno s_t y toma una acción en orden de aprender como mejorar la tarea. Como retroalimentación, el agente recibe una recompensa numérica r_{t+1} y se mueve al nuevo estado s_{t+1} [1].

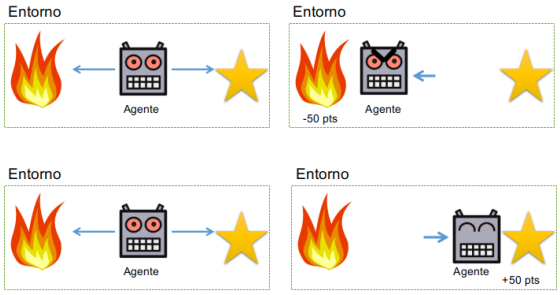


Figura 4.2. El agente realiza acciones en el entorno y el entorno responde a estas acciones

El agente deberá aprender la mejor manera de realizar la tarea, que se refleja en las recompensas que obtiene cada vez que realiza una acción. Por esta razón, el agente tiene que aprender a maximizar las recompensas a largo plazo. Para hacer esto, el agente asigna, con la función $Q(s_t, a_t)$, la utilidad con descuento esperada de tomar medidas a_t en el estado s_t si se comporta de manera óptima a partir de entonces. A partir del valor Q, el agente puede generar una política $\pi_t(s, a)$ que indica la probabilidad de aplicar la acción a_t cuando está en el estado s_t que asegura la recompensa máxima a largo plazo.

El algoritmo de Q-learning es ampliamente utilizado en problemas relacionados con aprendizaje por refuerzo. Actualiza los valores Q de acuerdo con sus mejores acciones, sin tener en cuenta la política actual, que puede incluir acciones exploratorias. La regla de actualización para cada valor Q es:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha[r + \gamma \max_{a \in A} Q(s_{t+1}, a)]$$

Donde:

- r: recompensa en el tiempo t ($r \in \mathbb{R}$)
- α: taza de aprendizaje en el tiempo t ($0 \leq \alpha \leq 1$)
- γ: factor de descuento ($0 \leq \gamma < 1$)

Política (π): Es la estrategia que debe emplear el agente para determinar la siguiente acción según el estado actual.

Valor Q: Es el rendimiento esperado a largo plazo con descuento según la política y la acción actual.

Tasa de aprendizaje (α): Un valor numérico que determina que tanto se va a actualizar el valor Q a partir de la recompensa recibida, si es muy grande, el resultado del aprendizaje puede no llegar a converger, si es muy pequeño, el aprendizaje puede llevar demasiado tiempo y haber convergido mucho antes.

Factor de descuento (γ): Un valor numérico que determina la importancia de los nuevos valores de aprendizaje a comparación de los anteriores, cuando es 1, se tienen en cuenta todos los valores pasados y el presente por igual, cuando es 0, solo se tiene en cuenta el valor actual por lo que el aprendizaje nunca va a converger.

Tasa de exploración (ϵ): Un valor numérico entre 0.0 y 1.0, que indica la probabilidad de tomar una acción aleatoria en lugar de la establecida por la política. Esto permite que el agente pueda explorar (como el nombre del parámetro lo indica) acciones nuevas, ya sea para descubrir que tienen buen desempeño, o para reafirmar que no son la acción correcta.

4.3. Aprendizaje por refuerzo inverso

De acuerdo con Abbeel y Ng en [5], dado que el campo del aprendizaje por refuerzo se basa en el hecho de que la función de

recompensa es la definición más clara, sólida y transferible de la tarea, tiene sentido considerar un enfoque del aprendizaje por refuerzo en el que la función de recompensa es lo que hay que aprender. El problema de derivar la función de recompensa de las observaciones se conoce como Aprendizaje por refuerzo inverso (IRL).

El aprendizaje por refuerzo inverso asume que un experto ya optimizó una función de recompensa $R(t)$ que es desconocida para el agente de aprendizaje, que puede expresarse como una combinación lineal de características conocidas ϕ_i que describen posibles subtareas o habilidades para aprender:

$$r(s) = w_1\phi_1(s) + w_2\phi_2(s) + \cdots + w_n\phi_n(s) = \boldsymbol{w^T\phi(s)}$$

Porque el experto conoce la verdadera función de recompensa $\boldsymbol{w^{*T}\phi(s)}$, con $\|\boldsymbol{w^*}\|_1 \leq 1$ para las recompensas limitadas por 1, la expectativa de utilidad de la política de expertos es:

$$\begin{aligned} U^{\pi_E}(s) &= \mathbf{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s)\middle|\pi_E\right] = \mathbf{E}\left[\sum_{t=0}^{\infty}\gamma^t \boldsymbol{w^{*T}\phi(s)}\middle|\pi_E\right] \\ &= \boldsymbol{w^{*T}}\mathbf{E}\left[\sum_{t=0}^{\infty}\gamma^t \boldsymbol{\phi(s)}\middle|\pi_E\right] = \boldsymbol{w^{*T}}\mu(\pi_E) \end{aligned}$$

Donde, $\mu(\pi_E)$ es conocido como *Características de la función* dada la política π_E. Para el caso del agente de aprendizaje, su utilidad se puede expresar como:

$$U^{\tilde{\pi}}(s) = \boldsymbol{w^T}\mu(\tilde{\pi})$$

El objetivo del aprendizaje por refuerzo inverso es encontrar valores para los pesos $w_1, w_2, \ldots, w_n$ tal que la utilidad de la política aprendida por el agente $\tilde{\pi}$ esté cerca de la utilidad de la política de expertos π_E:

$$\|\mu(\tilde{\pi}) - \mu(\pi_E)\|_2 \leq \epsilon$$

Esto significa que el algoritmo está buscando una función de recompensa $R = \boldsymbol{w^T\phi(s)}$, para el cual la política de expertos funciona mejor que el

alumno por un margen: $U^{\pi_E}(s) \geq U^{\tilde{\pi}}(s) + \Delta$. En términos de expectativas de características:

$$|\boldsymbol{w}^{\boldsymbol{T}}\mu(\tilde{\pi}) - \boldsymbol{w}^{*\boldsymbol{T}}\mu(\pi_E)| \leq \Delta$$

Y, al aproximar $\mathbf{w}$ a $\mathbf{w}^*$:

$$\mathbf{w}^{\mathbf{T}}|\mu(\tilde{\pi}) - \mu(\pi_E)| \leq \Delta$$

Debido a que se busca un margen Δ, el problema del aprendizaje por refuerzo inverso es similar al resuelto en máquinas de vectores de soporte (SVM), para encontrar el hiperplano de margen máximo que separa dos conjuntos de puntos [5]. Esta similitud se puede ver tomando las expectativas de la característica experta como clase +1 y las otras como clase -1, como se muestra en la Figura (Ver 4.3).

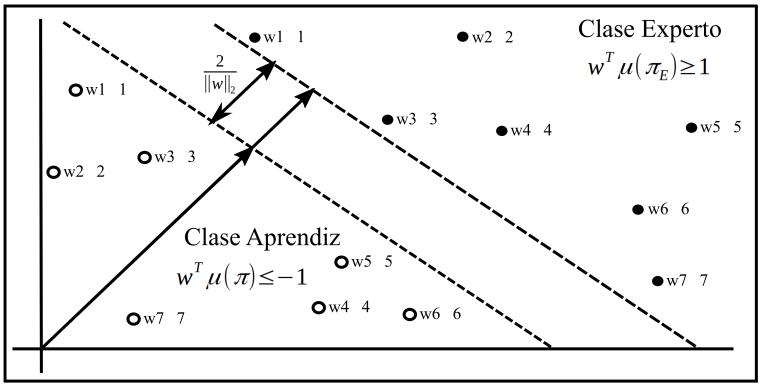


Figura 4.3. Aprendizaje por refuerzo inverso como un problema de clasificación SVM

La distancia del vector desde el origen al hiperplano clase +1 es $1/\|\mathbf{w}\|_2$ y la distancia del vector desde el origen al hiperplano clase -1 es $-1/\|\mathbf{w}\|_2$. Por lo tanto, el tamaño del margen es:

$$\frac{1}{\|\mathbf{w}\|_2} - \frac{-1}{\|\mathbf{w}\|_2} = \frac{2}{\|\mathbf{w}\|_2}$$

Para maximizar el margen, se tienen que minimizar los pesos, resultando el siguiente problema de programación cuadrática:

$$\text{mín}\, \frac{1}{2}\|\mathbf{w}\|_2^2$$

En cuanto a las restricciones, están relacionadas con la ecuación:

$$\mathbf{w^T}|\mu(\tilde{\pi}) - \mu(\pi_E)| \leq \Delta$$

Debido a la política de expertos tiene que hacerlo mejor que el alumno por un margen:

$$\mathbf{w^T}|\mu(\tilde{\pi}) - \mu(\pi_E)| \leq \frac{2}{\|\mathbf{w}\|_2}$$

Y, dado que las recompensas limitadas en 1, se tiene $\|\mathbf{w}\|_2 \leq \|\mathbf{w}\|_1 \leq 1$.

El problema completo de programación cuadrática se formula como:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 \\ \text{sujeto a} \quad & \mathbf{w^T}\left[\mu(\pi_E) - \mu(\tilde{\pi})\right] \geq -2 \end{aligned}$$

En el siguiente algoritmo se resume el proceso para aproximar $\mathbf{w}$ a $\mathbf{w}^*$ a través de la programación cuadrática y las expectativas de las características tanto del alumno como del experto.

Algorithm 1: Actualización de pesos

Result: $\mathbf{w}$ y estado del optimizador:= {viable, inviable}

1 **Requirements:** *Aprendiz* $\mu(\tilde{\pi})$ y *Experto* $\mu(\pi_E)$ Características esperadas, Solucionador de optimización convexa;

2 Función Objetivo = mín $\frac{1}{2}\|\mathbf{w}\|_2^2$;

3 Restricciones = $\mathbf{w^T}\left[\mu(\pi_E) - \mu(\tilde{\pi})\right] \geq -2$;

4 $\mathbf{w}, estado$ =Optimizador.resolver(Función Objetivo, Restricciones);

Con base en la matemática aplicada anteriormente, se desarrolla el algoritmo (Ver 2), previamente se requiere la trayectoria del experto y así poder calcular la utilidad del mismo, además de definir las Características

$\boldsymbol{\phi}$ necesarias para definir una función recompensa del entorno.

Algorithm 2: Aprendizaje por refuerzo inverso

Requirements: Trayectorias del experto, Características $\boldsymbol{\phi}$, Factor de descuento γ, Tasa de aprendizaje α, Tasa de exploración ϵ, Episodios;
Inicialización de los pesos $\mathbf{w}$;
Calcular características esperadas $\mu(\pi_E)$ de las trayectorias del experto;
for *episodio en episodios* **do**
    Inicializar el entorno;
    Obtener el estado actual s_t;
    while *No este completo* **do**
        Selecciona la acción a desde la política $\tilde{\pi}$ tomado desde $Q(s,)$ con exploración ϵ-*greedy*;
        Tomar acción a y observar s_{t+1} ;
        Calcular recompensa $r = \mathbf{w^T}\boldsymbol{\phi}(\boldsymbol{s_{t+1}})$;
        Actualizar política con QLearning: $Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha[r + \gamma \max_{a' \in A} Q(s_{t+1}, a')]$;
        Actualizar $s_t \leftarrow s_{t+1}$;
        if *Se tienen que actualizar los pesos* **then**
            Guardar y adjuntar trayectoria;
            Calcular expectativas de características $\mu(\tilde{\pi})$ desde las trayectorias del aprendiz;
            Estado=*inviable* $\mathbf{w}, Estado =$ Actualizar Pesos($\mu(\tilde{\pi})$, $\mu(\pi_E)$);
            if *Estado=inviable* **then**
                Eliminar la ultima característica del aprendiz;
        Verificar si se completo el entorno;
    end
end

4.4. Aprendizaje por refuerzo con red neuronal

Tal como se describe en [1], el objetivo del aprendizaje por refuerzo es aprender buenas políticas para problemas de decisión secuencial optimizando una señal acumulativa de la recompensa futura [6]. Una DQN (*Deep Q Network*) es una red neuronal multicapa en la que la entrada son los estados s y su salida es un vector de los valores de la acción $Q(s, *, \theta)$ donde θ son los parámetros de la red. la ecuación objetivo utilizada en las redes DQN es la siguiente:

$$\mathbf{Y}_t^{DQN} \equiv R_{t+1} + \gamma \max_{\mathbf{a}} Q(S_{t+1}, a, \theta_t)$$

En el algoritmo (Ver 3) se muestra como se realiza el entrenamiento en la red, además de incluir la memoria, adaptado de [7]:

Algorithm 3: DQN con repetición de experiencia

Requirements: Memoria inicializada D hasta su capacidad N;
Inicializar $Q(s, a, \theta)$ con pesos aleatorios;
for *episodio en episodios* **do**
    Inicializar el entorno;
    Obtener el estado actual s_t;
    while *No este completo* **do**
        Seleccionar una acción aleatoria a si el valor aleatorio es menor al ϵ, si no, seleccionar la acción desde $a_t = \max_a Q(s_t, a, \theta)$;
        Tomar acción a, observar recompensa $\mathrm{R}_t y \mathrm{s}_{t+1}$;
        Almacenar (s_t,a,r,s_{t+1}) en memoria D;
        Actualizar $s_t \leftarrow s_{t+1}$;Tomar una muestra $M <= N$ de la memoria D ;
        Actualizar $\mathrm{Y}_t^{DQN} =$
        $$\begin{cases} R_t & \text{si } \mathrm{s}_{t+1}\text{Estado Final} \\ R_t + \gamma \max \mathrm{a} Q(S_{t+1}, a, \theta_t) & \text{si no } \mathrm{s}_{t+1}\text{Estado Final} \end{cases}$$
    end
end

Capítulo 5

Antecedentes

5.1. Aprendizaje por demostraciones

Durante mucho tiempo, los estudios realizados sobre el aprendizaje por refuerzo han aumentado para diversas aplicaciones, ya que frecuentemente es muy difícil encontrar una política adecuada que permita armar una función de mapeo para las acciones que se van a realizar en el entorno. El aprendizaje a partir de demostraciones *(Learning from Demonstration - LfD)* ha resultado ser también una forma efectiva de obtener una buena política, obteniéndose de un experto que realiza las acciones óptimas [8–10].

Un ejemplo sobre LfD se encuentra en [11], donde se presenta un estudio general y algunas técnicas usadas en la academia, como dividir el problema en dos fases fundamentales, la primera, y en la que se quiere hacer énfasis, esta conformada por el conjunto de pares estado-acción grabados en un conjunto de datos durante la ejecución de las tareas realizadas por un experto. Además, los autores afirman que la similitud entre los espacios de estado-acción del experto con respecto al aprendiz determina el tipo de algoritmo requerido para procesar y transferir la información.

5.2. Adquisición de datos

En [12] se puede observar cómo se obtiene el conjunto de datos por medio de demostraciones kinestésicas y en [13, 14] por teleoperación. Específicamente en [14], se propone un sistema de aprendizaje profundo por imitación en un entorno de realidad basado en una red neuronal profunda. Aún así, ambos métodos requieren un operador experto, de lo contrario, el conjunto de datos contendrá errores que el algoritmo aprendería. Debido a esto, algunos trabajos actuales han creado demostraciones de conjunto de datos con personas [15, 16], con esto se logra que las demostraciones contengan movimientos óptimos, trasladando el problema en el dominio del espacio del individuo al del robot [17].

5.3. Inferencia de objetivos

Para lograr inferir la función de recompensa, en [5] los autores, Pieter Abbeel y Andrew Y. Ng, usan las demostraciones del experto, ya que contiene la información de la tarea de forma específica. Los autores parten considerando que el experto se comporta de forma óptima, maximizando su recompensa y expresándose como una combinación lineal de funciones características conocidas. Por medio de aprendizaje por refuerzo inverso, el aprendiz debe resolver el problema de optimización y así logra asignarle a la función de recompensa los pesos de cada característica tal que el valor de la política aprendida no difiera demasiado de la política del experto.

Aunque existen dos líneas definidas para la solución de los problemas de aprendizaje por demostración: aprendizaje supervisado y aprendizaje por refuerzo, existen investigaciones que buscan unir los dos conceptos para lograr aprovechar los beneficios de cada uno. En [18] se propone el uso del algoritmo de cascada supervisada en aprendizaje por refuerzo inverso, el cual aproxima valores de Q a través de algoritmos basados en un puntaje *Multi-Class Classification (MCC)* a partir del conjunto de demostraciones. Luego, se construye el conjunto de datos aumentado con aproximaciones del valor de la recompensa para cada par estado-acción

con base en los valores Q. Finalmente se predice la función de recompensa con algoritmos de regresión. También se propone el uso del algoritmo de clasificación estructurada para aprendizaje por refuerzo inverso, en el que se desea modelar la recompensa como una combinación lineal de características. En este último método, el valor esperado de las demostraciones del experto es estimado a través del algoritmo*Least-Squares Temporal Difference (LSTD)*. Luego se buscan funciones Q con algoritmos basados en puntaje MCC dentro del conjunto de posibles combinaciones lineales de las políticas en función del par estado-acción.

5.4. Uso de un experto

También se puede utilizar las demostraciones del experto para tener una aproximación a una política correcta, de modo que se puede generalizar a través del aprendizaje por refuerzo. Esta solución es usada en [19] donde en una red neuronal convolucional de 13 capas con aprendizaje supervisado se aproxima la política de movimientos de una persona en el juego de Go, luego, la red con los pesos se entrena con aprendizaje con refuerzo para lograr obtener la política a través de las exploraciones y así maximizar la cantidad de juegos ganados. Ahora, en [20] se busca comenzar el aprendizaje con una buena política ya que ahora se busca que el entorno no sea virtual para el aprendizaje. Los autores presentan un algoritmo de *Deep Q-learning from demonstrations* para volver a entrenar la red con los datos de demostración, combinando las pérdidas por diferencias temporales y supervisadas. Con la pérdida supervisada se puede lograr que el algoritmo copie el comportamiento del experto, mientras que con las de diferencias se busca satisfacer una función de valor que pueda cumplir con la ecuación de Bellman para poder continuar con el aprendizaje por refuerzo.

Con los trabajos mencionados anteriormente se asume que el experto se comporta de manera óptima y que el experto y el aprendiz toman decisiones sobre el mismo proceso de decisión de Markov. Por otro lado, si

no se cumplen estas condiciones, las demostraciones del experto no serían consistentes y no sería factible implementar su política en el aprendiz. En [21] se estudia este problema, donde se busca que un agente aprenda una política y también la representación de la función de recompensa a partir de un conjunto de demostraciones inconsistentes. Se tiene que realizar un conjunto de datos de entrenamiento con trayectorias consistentes, obteniéndose de las exploraciones del agente, aún así, no hay garantía de que las trayectorias posean la información relevante para el aprendizaje de la función de recompensa ya que provienen de un MDP distinto. Para solucionarlo, se evalúan la similitud de las trayectorias con demostraciones del experto. Mientras que para evaluarlo, los autores utilizan el algoritmo de *Affine Invariant Feature*. Ya extraído el conjunto de datos, el algoritmo de aprendizaje por refuerzo inverso puede ser aplicado y así obtener la función de recompensa.

Capítulo 6

Trabajo Previo

El desarrollo de este trabajo partió de los resultados obtenidos en el artículo *Inverse Reinforcement Learning Application for Discrete and Continuous Environments* [2], para el cual se realizó una contextualización del algoritmo de aprendizaje por refuerzo inverso, y se evaluó su funcionamiento en un ambiente continuo y uno discreto, ofrecidos en la suite *Open-AI gym*.

El primer entorno de trabajo fue *Taxi-v2* (Ver 6.1), es un entorno discreto y el objetivo consiste en llevar a un pasajero desde su ubicación a un destino, tanto el pasajero como el destino pueden aparecer en 4 puntos (R,G,Y,B) de forma aletoria con la condición de no ocupar el mismo lugar, el taxi (Agente) aparece de forma aleatoria dentro de los estados posibles permitidos y debe llegar hasta el pasajero, recogerlo, llevarlo hasta el destino y dejarlo.

El taxi tiene 6 acciones, norte, sur, este, oeste, dejar pasajero y recoger pasajero, su entorno consiste en una rejilla de 5x5 con paredes y los 4 posibles puntos (R,G,Y,B), cada paso que da disminuye su recompensa en -1, si deja o intenta recoger al pasajero donde no debe, disminuye -10, y si completa el entorno con éxito, se le da una recompensa de 20.

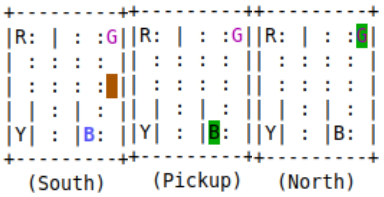


Figura 6.1. Entorno Taxi-v2. [2]

Al aplicar el algoritmo de aprendizaje por refuerzo inverso (Ver 6.2), se tuvieron en cuenta 2 funciones recompensa, ambas consistían en la distancia entre el taxi y su objetivo actual (pasajero o destino), se diferenciaban en que una tomaba la distancia euclidiana (linea recta) y la otra utilizando manhattan (cuadros de distancia). Ambas formas logran converger en una solución, pero la que mejor rendimiento tuvo fue la función recompensa con distancia euclidiana.

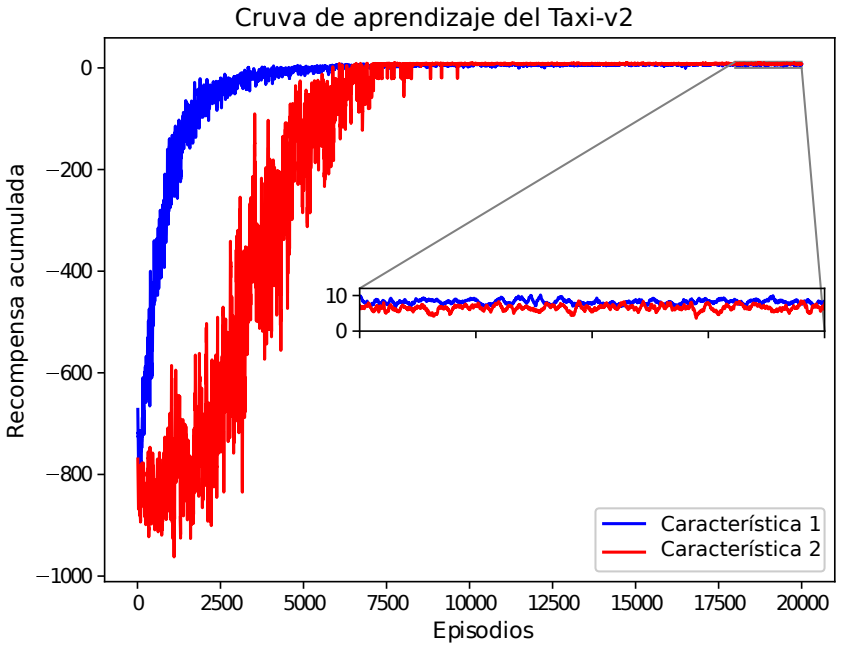


Figura 6.2. Curva de aprendizaje de dos funciones recompensa distintas (Distancia euclidiana y manhattan). [2]

El segundo entorno de trabajo era el *Cartpole-v1* (Ver 6.3), un entorno continuo que consiste en mantener en equilibrio el mayor tiempo posible un péndulo invertido moviendo su base únicamente hacia la izquierda o derecha. Cada paso aumenta su recompensa total en 1, entre mayor tiempo, mayor sera la recompensa total (Ver 6.4). Sus estados consistían en el ángulo y la velocidad angular actual del péndulo, así como la posición y velocidad de la base.

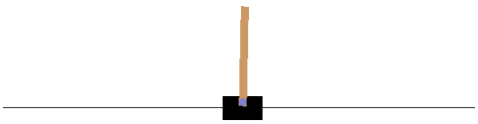


Figura 6.3. Entorno Carpole-v1. [2]

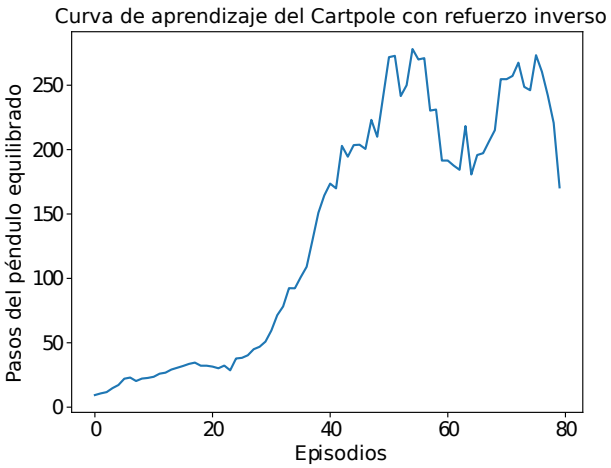


Figura 6.4. Curva de aprendizaje del Cartpole-v1. [2]

Aunque en ambos entornos se aplicó aprendizaje por refuerzo inverso, en el entorno discreto (Taxi-v2) se utilizó tabla Q, en cambio para el entorno continuo (Carpole-v1) se utilizo una red neuronal implementada en Keras.

Capítulo 7

Desarrollo Metodológico

Para la primera etapa del desarrollo, se recopila información acerca de los diferentes trabajos realizados con aprendizaje por refuerzo y aprendizaje por refuerzo inverso aplicados en robótica. Se utilizaron diferentes bases de datos como IEEExplore, Scopus y ScienceDirect, además, se tuvieron en cuenta diferentes tipos de conferencias relacionadas con el tema con el fin de ampliar el marco de información relevante.

En la segunda etapa de desarrollo, se establece un marco de trabajo sobre el cual se desarrolla un entorno, este consiste de un modelo del robot con su cinemática, la opción de visualización del movimiento y también con la posibilidad de mostrar los movimientos en un simulador, la definición de los estados que va a devolver el entorno y una función recompensa del entorno basada en la distancia entre el punto objetivo que se desea alcanzar y la posición de la mano del robot humanoide. Dentro de esta etapa esta la programación del robot humanoide físico, esta cuenta con las muestras del programa usado en la comunicación de los motores y también la comunicación con el programa de simulación con el fin de abstraer esa capa al momento de ser usada por el entorno.

Para la tercera fase, se creó un modelo de red neuronal con el fin de entrenarla como agente experto, este agente experto aprende a moverse en el entorno utilizando aprendizaje por refuerzo. Al momento de realizar un barrido de parámetros para mejorar sus oportunidades de aprendizaje se

tomaron en cuenta los tiempos que se demora aprendiendo, además de los hiperparámetros que mejor rendimiento tuvieron. Luego, cuando el agente experto consiguió una política que le permita resolver el entorno en un porcentaje mayor al 95 % de los casos, se realizaron varias pruebas más y finalmente se almacenaron las trayectorias que lo llevan a completar el entorno.

En la cuarta fase, las trayectorias del experto son utilizadas por el aprendiz, el aprendiz es otro modelo de red neuronal que conserva los hiper parámetros del experto para su entrenamiento, pero que no conoce la recompensa del entorno y por medio de aprendizaje por refuerzo inverso intenta extraer las utilidades del experto para el usarlas como guía al momento de completar el entorno.

Para la quinta fase, se realizan pruebas comparando las trayectorias del experto con las del aprendiz, finalizando con las pruebas del modelo del aprendiz en un entorno de simulación. Dependiendo de los resultados obtenidos en las pruebas, se realiza una retroalimentación de algunos de los parámetros en la implementación del aprendizaje.

En la figura (Ver 7.1), se muestran estas cinco fases del desarrollo metodológico.

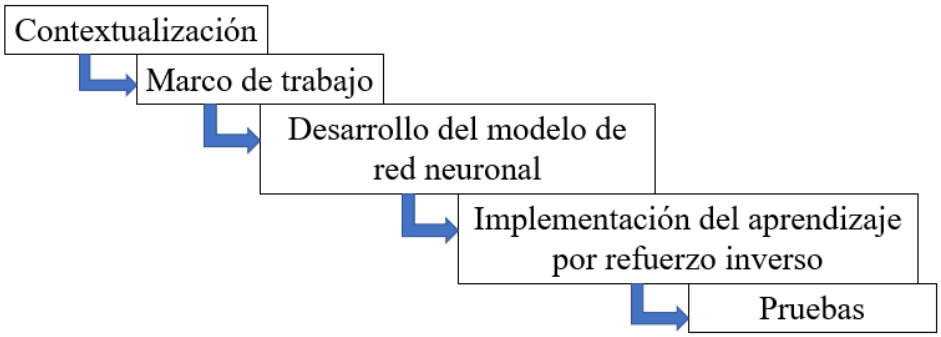


Figura 7.1. Diagrama del desarrollo metodológico

Capítulo 8

Marco de trabajo

8.1. Construcción y programación de la plataforma robótica Poppy Torso

Dentro del laboratorio de robótica de la Universidad Santo Tomás, se planteó la construcción de una plataforma robótica humanoide, que permitiera representar los movimientos de las extremidades superiores de un ser humano y que además fuera un proyecto libre y de código abierto. A partir de estas especificaciones, se escogió *The Poppy Project* [3], y más específicamente, su versión denominada *Poppy Torso* (Ver 8.1) ya que dentro de sus repositorios se encontraba un modelo 3D listo para simulación, también contaba con los modelos cinemáticos y una librería de comunicación entre el equipo y los servomotores.

Figura 8.1. Robot humanoide Poppy Torso [3]

La programación del robot se realizó en *python* estableciendo comunicación con los motores mediante una placa controladora de uso libre y abierto conocida como OpenCM (Ver 8.2):

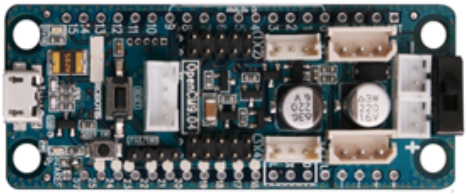


Figura 8.2. OpenCM 9.04 B

La comunicación entre el robot físico y la simulación de V Rep es implementada a través de la librería de control incluida en [3], esto permite una fácil y rápida transición entre los dos modelos, especialmente útil cuando se requiere realizar movimientos de ambas formas.

Los motores usados fueron de proyectos anteriores, Dinamixel MX-28 y AX-12 (Ver 8.3).

Figura 8.3. Motores Dinamixel

Las impresiones 3D fueron realizadas dentro de la universidad, gracias a la impresora perteneciente al grupo de Modelado, Electrónica y Monitoreo (MEM) de la universidad.

Durante la impresión 3D pueden aparecer algunas imperfecciones que ocasionan irregularidades en la pieza y dificultan el ensamblado con los motores, además de que algunas de estas piezas fueron impresas nuevamente debido a que presentaban grietas o se rompían directamente. (Ver 8.4).

Figura 8.4. Brazos con motores

El ensamble final (Ver 8.5) cuenta con un espacio en la cabeza que puede ser usado para llevar una cámara y dotar al robot de visión.

Figura 8.5. Poppy Torso

Además, para que pueda moverse, se adaptó un robot futbolista existente (Ver 8.6) en el laboratorio, pero antes se programo una interfaz gráfica basada en una matriz de comunicación (Ver 8.1) de los robots que permitía el movimiento independiente de las funciones del robot futbolista.

Trama Robot X								
	7	6	5	4	3	2	1	0
Byte 0	ID							
Byte 1	Velocidad Motor 1 - Bits 7:0							
Byte 2	Velocidad Motor 2 - Bits 7:0							
Byte 3	Velocidad Motor 3 - Bits 7:0							
Byte 4	Velocidad Motor 4 - Bits 7:0							
Byte 5	M1 - Dir	M1 - Bit9	M2 - Dir	M2 - Bit9	M3 - Dir	M3 - Bit9	M4 - Dir	M4 - Bit9
Byte 6	Tipo Pateo	Potencia de Pateo - Bits 6:0						
Byte 7	?	?	Velocidad Drebler					
Byte 8	Checksum							

Cuadro 8.1. Matriz de comunicación para los robots futbolistas

Figura 8.6. Robot futbolista adaptado

8.2. Simulación del robot humanoide Poppy Torso

La naturaleza del aprendizaje por refuerzo, en especial en etapas tempranas del entrenamiento, implica la exploración de acciones que no necesariamente presentan buena utilidad. El tomar dichas acciones puede llevar a movimientos que comprometan la integridad física del robot. Además, la cantidad de episodios que implica un entrenamiento llevaría al desgaste de las partes mecánicas del dispositivo, teniendo en cuenta que dichos entrenamientos pueden durar horas, o incluso días. Por estas

razones, y a pesar de tener la plataforma robótica totalmente funcional, se decide no hacer uso de esta para los entrenamientos.

A partir de lo anteriormente expuesto, se procede a la creación de un modelo de simulación que permita tener en cuenta las características físicas del robot ya construido, basado principalmente en su cinemática (Ver 8.7). Esta cinemática es la que utilizan los simuladores modernos (Ver 8.8) para modelar el comportamiento y la física del robot y se encuentra dentro del repositorio oficial en [3].

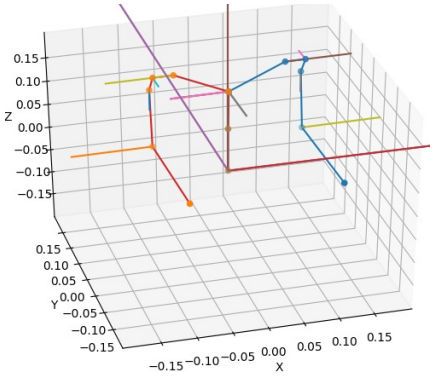


Figura 8.7. Imagen de la cinemática inversa usando la librería IKPY [4]

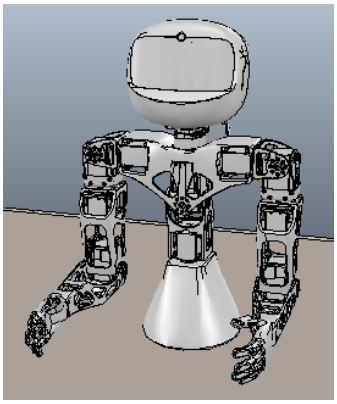


Figura 8.8. Simulación de Poppy Torso en V Rep

URDF (Siglas en ingles de ***Unified Robot Description Format***) es un formato de archivos ampliamente usado para describir los modelos de

los robots, dentro de los repositorios del proyecto Poppy [3] se encuentran 3 archivos cada uno correspondiente a ***Poppy Humanoid***, ***Poppy Torso*** y ***Ergo Jr***. Para el caso del entorno actual, se usa la cinemática del brazo izquierdo descrito en ***Poppy Torso***, dentro del modelo que se puede describir usando la librería IKPY se deben tener en cuenta ciertas diferencias con el estándar URDF.

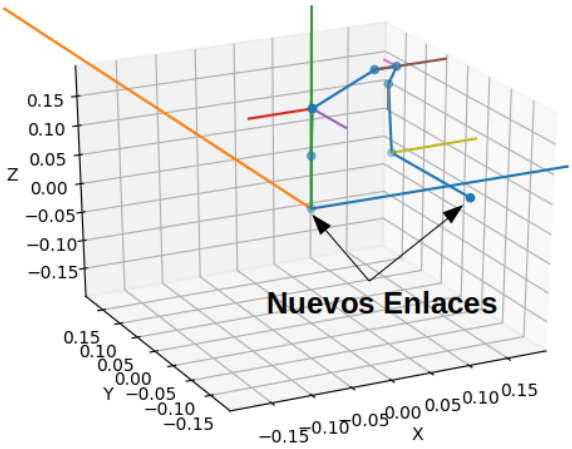


Figura 8.9. Enlaces que conforman el modelo del brazo izquierdo, 9 en total

Como se observa en la figura (Ver 8.9), se crean 2 nuevos enlaces dentro de la descripción de la cinemática del brazo izquierdo, y esto ocurre porque para la libraría IKPY los enlaces son las articulaciones en UDRF, y para conocer el punto final, este debe ser un enlace, para UDRF un enlace sería una conexión entre 2 articulaciones.

Con la cinemática completa, se definen los demás aspectos y funciones del entorno, el cual debe poder inicializarse, permitir ingresar una acción y devolver un estado junto con la recompensa a dicha acción y por último indicar el estado siguiente o si se completó el entorno.

Para el caso específico del brazo, se toman las siguientes definiciones:

$$A_{Oi} = \text{Ángulo objetivo del motor } i$$
$$A_{Ai} = \text{Ángulo actual del motor } i$$

$$\text{Acción} = rad([-1, 1]) \tag{8.1}$$

$$\begin{aligned}\text{Estado} = [&A_{O0} - A_{A0}, A_{A0}, A_{O1} - A_{A1}, A_{A1},\\ &A_{O2} - A_{A2}, A_{A2}, A_{O3} - A_{A3}, A_{A3}]\end{aligned} \tag{8.2}$$

$$\text{Recompensa} = -\sqrt{\sum ([\mathbf{A_O}] - [\mathbf{A_A}])^2} \tag{8.3}$$

La recompensa es la norma dos de la diferencia de los ángulos, y se propone negativa para asegurar que el algoritmo entrenará hasta minimizarla (maximizar la recompensa).

En el brazo se toman en cuenta solo 4 enlaces de los 9 mencionados anteriormente, ***l_shoulder_y***,***l_shoulder_x***, ***l_arm_z***, ***l_elbow_y*** , cada enlace tiene como propiedad su ángulo actual en radianes, y en la descripción del enlace se incluyó su limite inferior y superior, por ende cuando nos referimos al vector de ángulos actuales, su longitud es de 4 y como la diferencia de los vectores de los ángulos finales y los actuales también es de 4, el estado resulta ser un vector de 8 posiciones en total, en una forma más comprensible, se tiene la posición actual del brazo y lo que le hace falta para llegar al objetivo.

La recompensa tiene un modificador, dentro del entorno existe un contador que almacena las veces que el estado cumple con la condición de completar el entorno, cuando ese número de veces es mayor a 1, la recompensa se multiplica por el 90 %, ya que es negativa, esto indicaría que la perdida es menor, cuando el contador es mayor a 10 el entorno devuelve la señal de completado y la recompensa se multiplica ahora por el 80 %.

Capítulo 9

Implementación del aprendizaje por refuerzo

Previo a definir los hiperparámetros de la red del experto, se realizó un barrido de parámetros para mejorar el rendimiento del entrenamiento, una muestra de un barrido que se hizo, fue tomar el tamaño de la memoria, el tiempo de entrenamiento y el valor final de exploración (ϵ), aunque se tomaron más consideraciones como la tasa de aprendizaje (α) y el factor de descuento (γ).

Al final, se dejaron los mismos hiperparámetros de la red neuronal para el experto y el aprendiz. Luego de realizar un barrido de parámetros mas extenso, el tamaño del lote a entrenar se limitó a 10000 datos, la tasa de aprendizaje (α) es de 0.001, el valor de gamma (γ) es de 0.9 y comienza con una tasa de exploración de 0.8 finalizando con 0.2 de forma lineal, todo esto dentro de 100.000 iteraciones para el experto y 10 iteraciones cada una de 10.000 iteraciones en refuerzo para el aprendiz.

Para la implementación del aprendizaje por refuerzo, se crea un modelo MPL (*Multi-Layer-Perceptron*) de 2 capas con 64 neuronas en *Tensorflow*, este modelo se entrena usando un algoritmo de *Double Q-Learning* [6, 7], la memoria implementada en la red neuronal utiliza un algoritmo de *Prioritized Experience Replay* [22], así se garantiza no sobre-entrenar la red ni llenar la memoria con valores muy similares que

Tamaño de la memoria	Exploración final	Tiempo de entrenamiento (seg)	% Éxito (100 pruebas)
100	0.6	431.19	5 %
10000	0.6	411.28	45 %
50000	0.6	398.99	21 %
100	0.2	383.47	8 %
10000	0.2	399.49	17 %
50000	0.2	431.23	36 %
100	0.01	437.11	3 %
10000	0.01	401.89	28 %
50000	0.01	429.52	27 %

Cuadro 9.1. Tabla de pruebas de 9 modelos entrenados con refuerzo cambiando sus hiper parámetros

aporten poco al entrenamiento.

El resultado del entrenamiento se puede ver reflejado en las siguientes gráficas (Ver 9.4), esta es una secuencia normal generada a partir de dos puntos XYZ, uno inicial y otro final:

Los valores para los hiperparámetros del entorno son un punto clave en la obtención de un buen modelo, la siguiente gráfica (Ver 9.2) muestra un modelo con hiperparámetros sin perfeccionar.

Realizando un barrido de parámetros correcto, la trayectoria resultante es tal como se visualiza en (Ver 9.4), se almacenaron algunos parámetros del entrenamiento para visualizar su comportamiento (Ver 9.3).

En este entrenamiento (Ver 9.3) se logra ver claramente que la recompensa converge en un valor, la recompensa de entrada es una normalización interna realizada dentro del modelo, la pérdida del modelo esta convergiendo cerca del 0.

Con este entrenamiento se obtiene el agente experto, de las pruebas realizadas la tasa de éxito para completar el entorno esta en un 97.5 % de 1000 pruebas, con el agente experto se realizan 100 pruebas y se almacenan

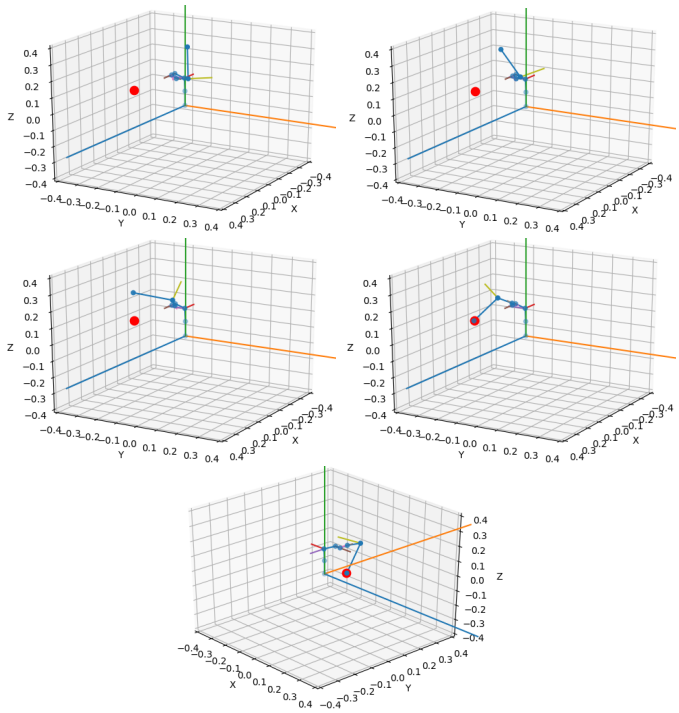


Figura 9.1. Secuencia del brazo alcanzando el objetivo, la última inferior es desde otra perspectiva

las trayectorias que ha tomado el agente.

En la figura (Ver 9.4), se muestra la trayectoria del experto partiendo desde un punto inicial aleatorio hasta un punto final aleatorio, se nota claramente como intenta llegar en el menor número de pasos que puede.

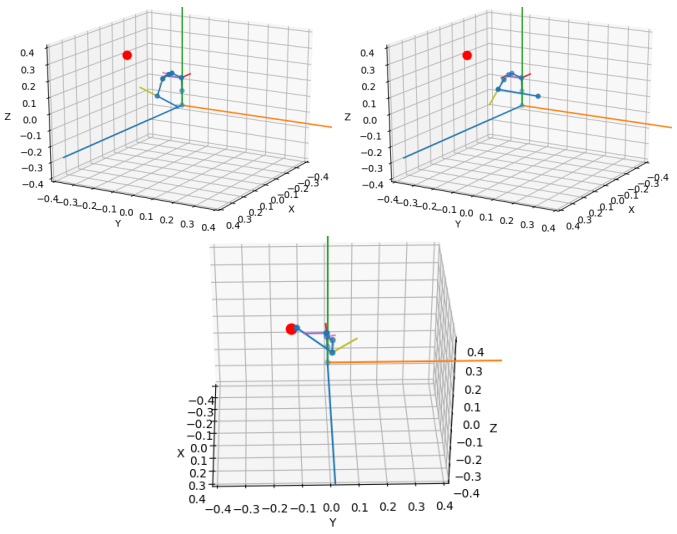


Figura 9.2. Secuencia del brazo alcanzando el objetivo, toma mas pasos que el modelo final y aunque alcanza el objetivo, queda solo en el borde del mismo

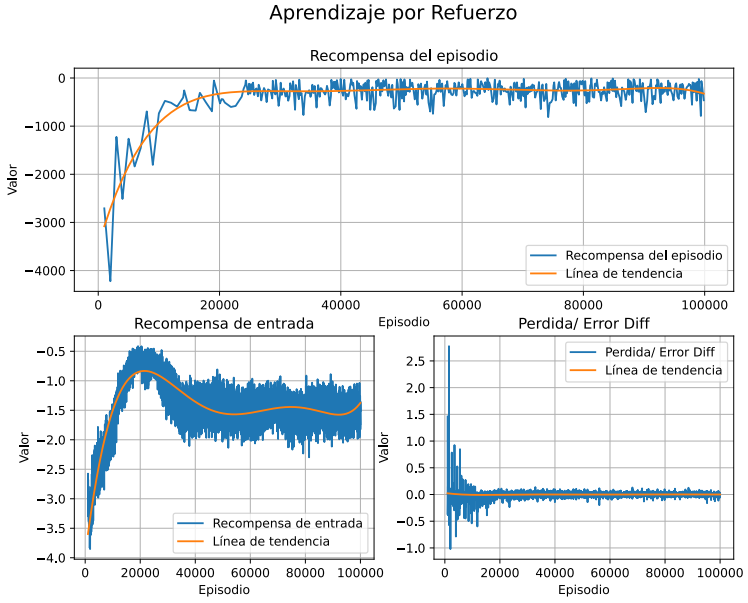


Figura 9.3. Entrenamiento del experto, el agente logra converger a un valor alrededor de los 40.000 episodios

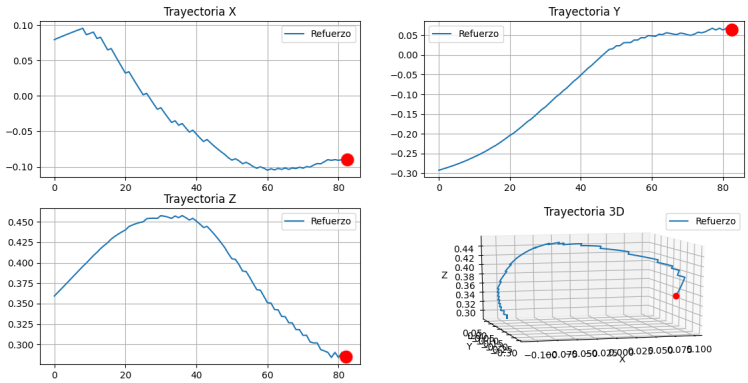


Figura 9.4. Trayectoria desde un punto inicial a un punto final, llegando a su objetivo

Capítulo 10

Implementación del aprendizaje por refuerzo inverso

Por medio de la trayectoria generada por el agente experto creado utilizando únicamente aprendizaje por refuerzo, se aplica el algoritmo de aprendizaje por refuerzo inverso [5], añadiendo dos hiper parámetros más, el primero es el factor de descuento para el inverso (γ) diferente al de la red neuronal, este se utilizará cuando se van a calcular las utilidades tanto del agente experto como del aprendiz, el segundo hiper parámetro es la cantidad de veces que se realiza el calculo de los pesos.

Las características del entorno no cambian, excepto que no se toma su recompensa para el entrenamiento, en cambio, se formula otra función recompensa la cual el entrenamiento ajustará los pesos entre los componentes de la misma para intentar recrear la misma función que tenia el entorno originalmente.

$$A_{Oi} = \text{Ángulo objetivo del motor } i$$
$$A_{Ai} = \text{Ángulo actual del motor } i$$

$$\text{Acción} = rad([-1, 1]) \tag{10.1}$$

$$\begin{aligned}\text{Estado} = [&A_{O0} - A_{A0}, A_{A0}, A_{O1} - A_{A1}, A_{A1},\\ &A_{O2} - A_{A2}, A_{A2}, A_{O3} - A_{A3}, A_{A3}]\end{aligned} \tag{10.2}$$

$$\text{R. Inverso} = [-W_1 * \sqrt{\sum ([\mathbf{A_O}] - [\mathbf{A_A}])^2}, W_2 * \frac{1}{N.Pasos}] \tag{10.3}$$

Ahora visualizando los entrenamientos del inverso (Ver 10.1)(Ver 10.2), se evidencia el impacto que tienen los pesos en las recompensas, ya que esta recompensa es una utilidad que se ha definido previamente y no es la que devuelve el entorno, de ahí que se parta del principio de necesitar un agente experto.

En la figura (Ver 10.3), se muestra una trayectoria aleatoria generada y cómo el agente es capaz de llegar hasta su destino de una forma rápida cumpliendo con el movimiento permitido por la cinemática del modelo del robot.

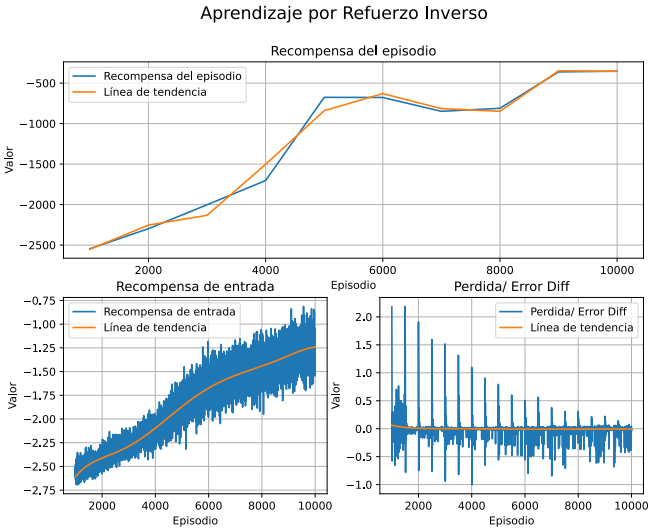


Figura 10.1. Primeras 10000 iteraciones, el entrenamiento aún no encuentra un valor para los pesos que logre converger

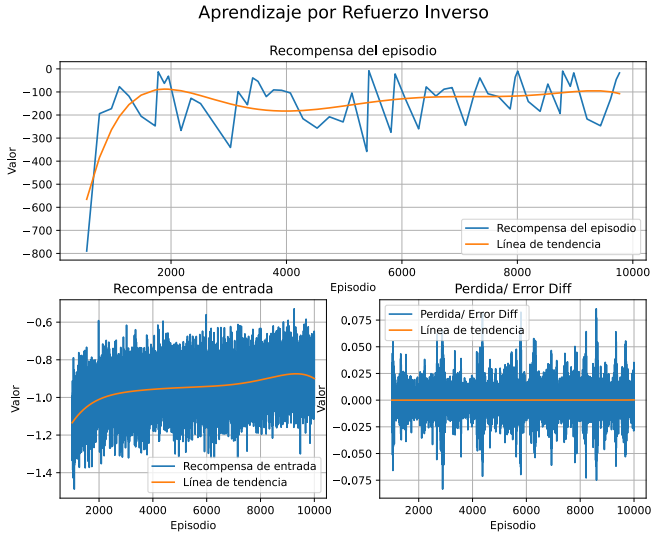


Figura 10.2. Últimos 10.000 de 100.000 iteraciones, el entorno converge y ya se pueden obtener los pesos de la función objetivo desconociendo completamente las recompensas del entorno

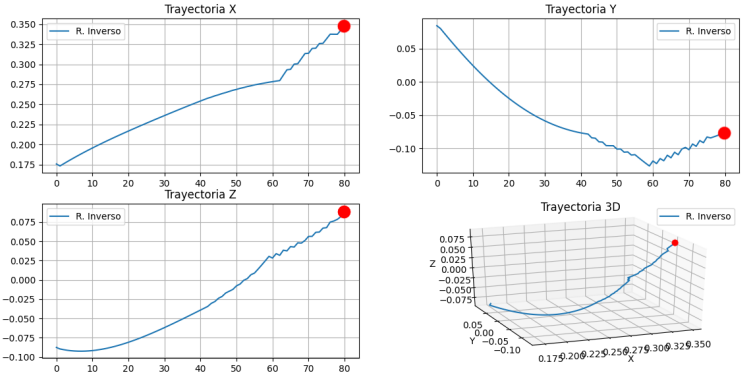


Figura 10.3. Trayectoria desde un punto inicial a un punto final, llegando a su objetivo

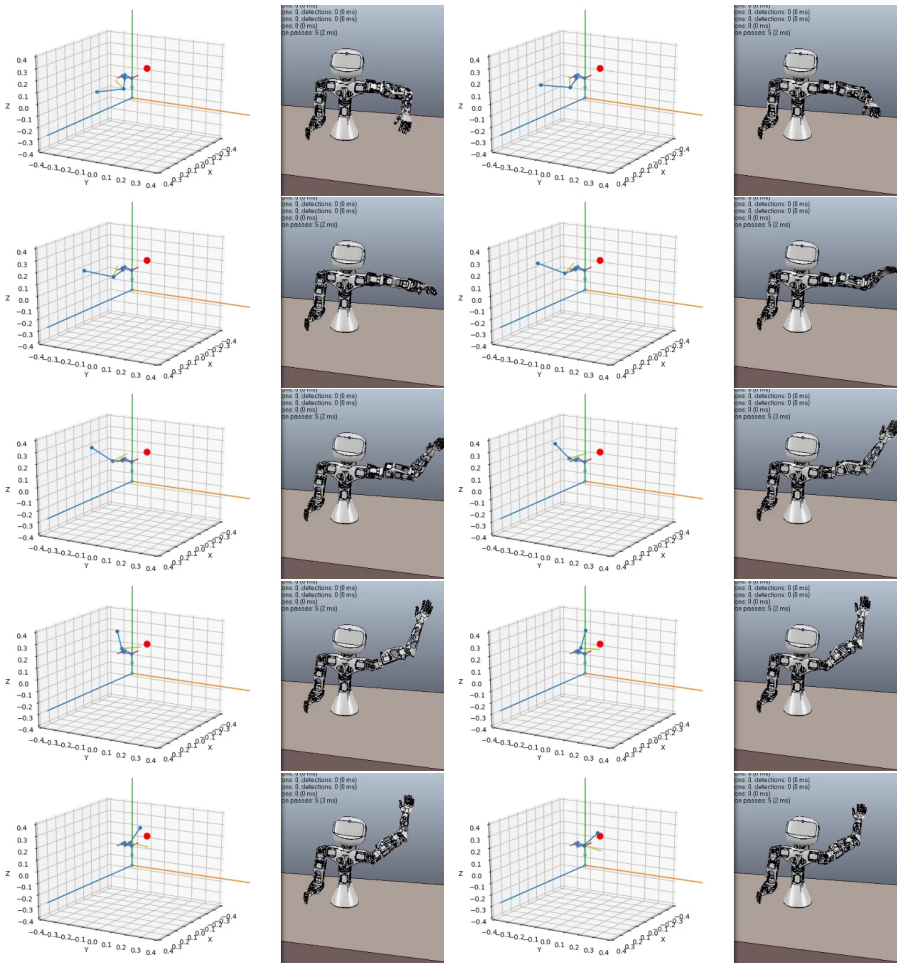


Figura 10.4. Secuencia del brazo alcanzando el objetivo, de izquierda a derecha, de arriba a abajo

Capítulo 11

Análisis de resultados

Luego de los entrenamientos, la función de recompensa con los pesos extraídos del aprendizaje queda definida de la siguiente forma:

$$\text{R. Inverso} = [-0{,}5575 * \sqrt{\sum ([\mathbf{A_O}] - [\mathbf{A_A}])^2}, 0{,}0237 * \frac{1}{N.Pasos}] \quad (11.1)$$

Donde se muestra claramente que se le da una mayor relevancia al valor que le indica que tan lejos esta del punto, por el contrario, la parte de la función que le indica cuantos pasos ha recorrido no ha sido muy relevante y esto sucede porque no es fijo el número de pasos que puede llegar a necesitar para ir desde un punto inicial al punto final.

Como se muestra en la figura (Ver 11.1) partiendo de los mismos puntos y el mismo destino, las trayectorias que toman el aprendiz y el experto no son necesariamente las mismas, además de que el entrenamiento realizado mediante el aprendizaje por refuerzo inverso, no puede tener una utilidad mayor que la del agente experto.

La utilidad en este caso fue de -38.086, un valor basándose en la sumatoria de la recompensa final de las trayectorias y tomadas de la función recompensa del entorno, mas no de la función de recompensa planteada para el aprendizaje por refuerzo inverso. Llegar hasta el objetivo les tomo 119 pasos a ambos modelos, cuando la distancia entre los puntos es mayor o cuando el brazo necesita hacer un giro sobre alguno de sus motores, los modelos pueden tomar una mayor cantidad de pasos

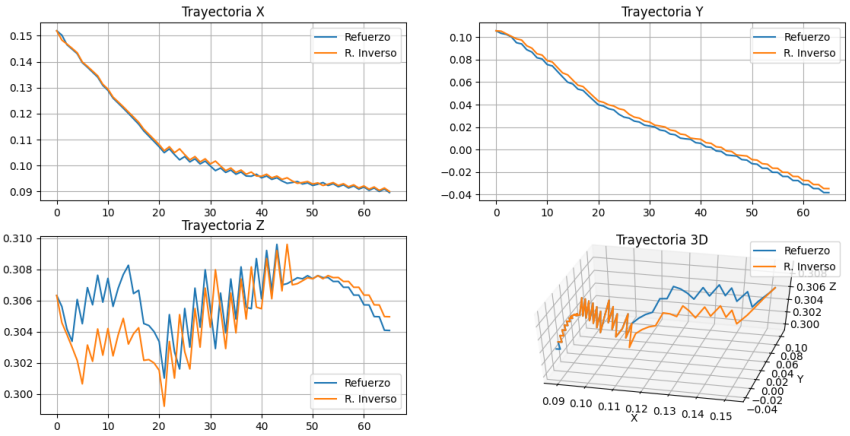


Figura 11.1. Trayectorias en XYZ del brazo con ambos modelos de entrenamiento

para completar la tarea y en muchos casos la diferencia en las trayectorias no resulta muy evidente (Ver 11.2).

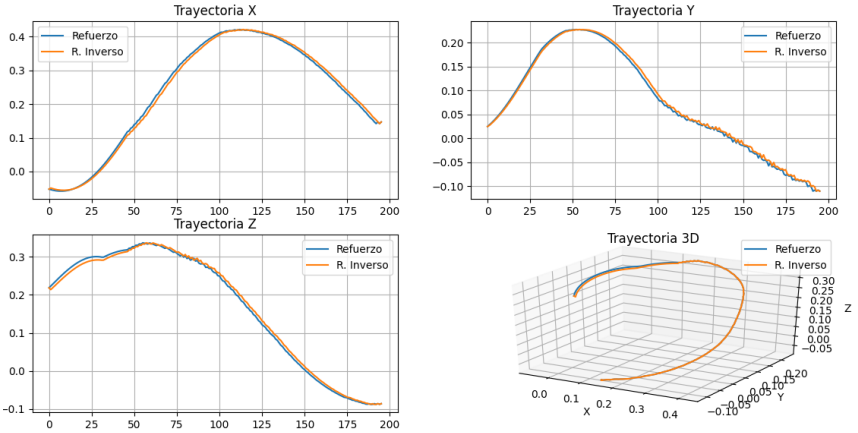


Figura 11.2. Nueva trayectoria, ambos modelos comienzan y terminan en el mismo punto

Finalmente, para realizar las pruebas pertinentes del robot humanoide sobre un entorno simulado (Ver 11.3), se hace uso del simulador V Rep EDU (hoy en día, el nombre ha cambiado a CoppeliaSim Edu conservando sus características), las siguientes figuras muestran un

ejemplo del comportamiento del modelo con aprendizaje por refuerzo inverso

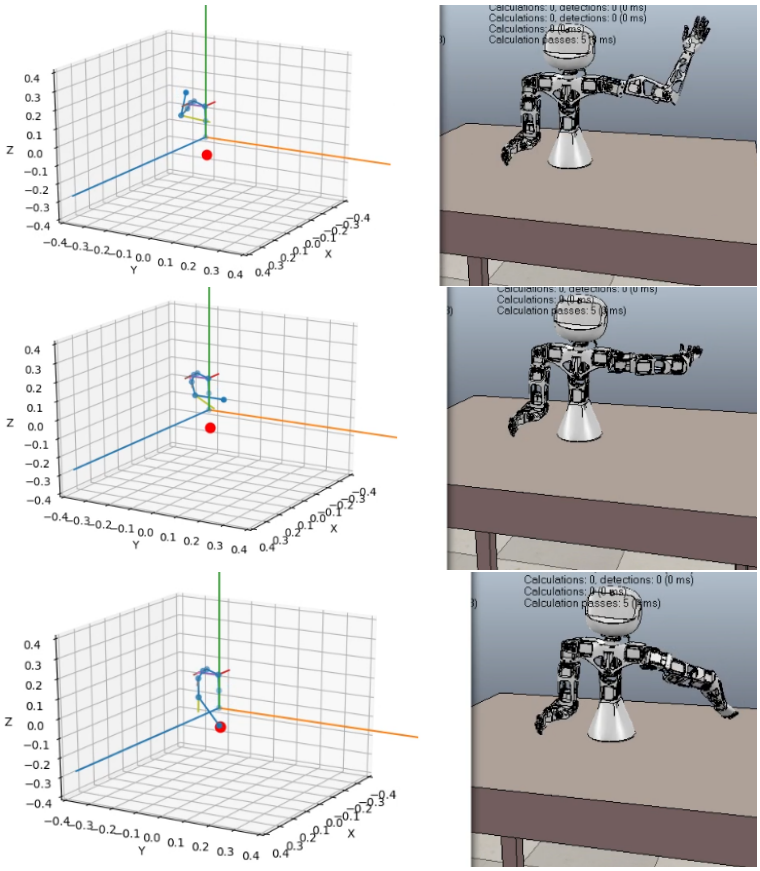


Figura 11.3. Simulación en V Rep de Poppy Torso siguiendo una trayectoria a un punto usando el modelo aprendido por aprendizaje por refuerzo inverso

Conclusiones

Como se presentó en los resultados, las trayectorias generadas por el experto determinan la utilidad final que puede llegar a alcanzar el aprendiz, es claro que el aprendiz en este caso no puede realizar mejor la tarea que el agente experto. Además, la función de recompensa del entorno era desconocida para el aprendiz, por lo que al plantearse una función aparte, se debían tomar las utilidades que el experto tuviera en esta nueva función y por eso las trayectorias almacenadas corresponden únicamente a estados y no pares estado-acción. Conocer que tan relevante puede ser cada una de las funciones que componen esta nueva función de recompensa es de suma importancia al extraer el comportamiento del entorno para futuros entrenamientos, recordar que la sumatoria aunque sea lineal, no obliga a que todos sus elementos (objetivos) tengan la misma relevancia al completar la tarea implícita en el entorno. Para el caso especifico del aprendiz en el documento, el segundo objetivo era tener el menor número de pasos, si esta tarea hubiera tenido mayor relevancia que acercarse a su objetivo, realmente nunca hubiera completado el entorno, ya que buscaría la manera de terminar el episodio rápidamente al golpearse a si mismo o alguna otra forma en la que termine el entorno sin lograr el objetivo principal de alcanzar el punto.

Además, implementar y desarrollar la cinemática del robot basándose únicamente en el modelo del mismo resulta muy conveniente para realizar estos entrenamientos al reducir drásticamente el cálculo matemático extra que realizan los simuladores robóticos modernos como Gazebo o V Rep, a cambio, se pierden posibles colisiones o comportamientos inesperados y la

física del robot, pero de todas formas no logra tener una gran incidencia sobre el resultado final del modelo.

Otro punto a tener en cuenta a la hora de realizar cualquier tipo de entrenamiento es el barrido de hiperparámetros, esta parte puede resultar muy obvia al comienzo pero realizando pequeñas pruebas sobre un entorno bien estructurado y funcional, puede ser un factor clave al lograr reducir los tiempos de entrenamiento y hacerlos posibles sobre computadores más tradicionales con pocos recursos en los que un parámetro como la memoria del agente puede llegar a ocupar toda la RAM del equipo, incluso se puede lograr un entrenamiento en un equipo de muy limitados reduciendo el elevado consumo de un equipo con mejores características.

Finalmente, con el desarrollo de estos algoritmos en el documento junto con la construcción del robot humanoide Poppy Torso y el código fuente usado, se espera abrir un abanico de nuevos proyectos en el laboratorio de robótica sobre aprendizaje de máquina, ya sea utilizando estos recursos o basándose en ellos.

Capítulo 12

Trabajo Futuro

Futuros experimentos pueden realizarse con el trabajo desarrollado en este documento, el robot físico derivado del trabajo realizado se puede usar en demostraciones de visión artificial si se llega a dotar al mismo de una cámara, como por ejemplo enseñarle a saludar cuando una persona salude con la mano dentro del campo de visión del robot, así como puede ser implementado para otros algoritmos de aprendizaje no mencionados en este documento.

El entorno simulado basado en el robot real que se creo para desarrollar los algoritmos no cuenta actualmente con detección de colisión ni con el tamaño de las piezas, esto puede generar que a la hora de implementar el modelo sobre el robot real, este pueda quedarse atascado consigo mismo y provocar algún daño no contemplado.

De igual forma, esto abre la posibilidad de mejorar el código del entorno y diseñar mejores formas de tener en cuenta aspectos técnicos, o de finalmente utilizar equipos mas potentes y mayores tiempos de entrenamiento al utilizar simuladores profesionales como el usado para evaluar estos modelos desarrollados.

Otra posibilidad que puede ser implementada a futuro es tomar las muestras del experto directamente de demostraciones en videos por parte de los seres humanos, claro, para movimientos sencillos teniendo en cuenta los grados de libertad que posee el robot y los ángulos de captura de los

videos.

Usando aprendizaje por refuerzo se pueden crear expertos como se explicó en capítulos anteriores, con esto es posible generar modelos expertos con la cinemática del robot *Poppy Torso* para otros robots que compartan características similares. Estos nuevos modelos basados en personas pueden ser utilizados en otros robots humanoides y finalmente ser comparados con los modelos derivados del trabajo en este documento.

Capítulo 13

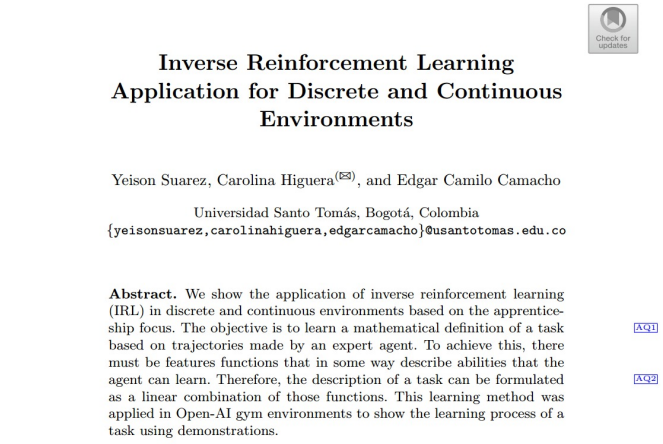
Productos y Publicaciones

El primer producto de la investigación de los algoritmos de aprendizaje utilizados en [5] se encuentra en *Inverse Reinforcement Learning Application for Discrete and Continuous Environments* (Ver 13.1) [2] dentro de la publicación del *AETA 2019 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application* (ISBN 978-3-030-53020-4) de la revista Springer Nature y que ha sido clasificado como Q3 por *SCIMAGO*, en ese documento ya se han realizado pruebas del algoritmo de aprendizaje por refuerzo inverso en dos entornos ofrecidos dentro del *Open-AI gym suite*. Este artículo fue presentado en ingles, dentro de la ponencia del AETA 2019 en Bogotá.

Además, se realizó el registro del software desarrollado para obtener dichos resultados ante la Dirección Nacional de Derechos de Autor, bajo el registro *13-78-368*. Dicho software es abierto y se encuentra en el siguiente repositorio: `https://github.com/Yeisonint/IRL-Application-for-Discrete-and-Continuous-Environments`

Se diseñó además un programa gráfico (Ver 13.2) para la prueba del robot futbolista y sus partes, este programa permite ayudar a verificar el correcto funcionamiento de cada uno de los motores, velocidad del rodillo y potencia del pateo de la pelota, la matriz de comunicación utilizada le permitía enviar información codificada a los 12 robots al mismo tiempo con un canal de comunicación común. El registro de este software ante la

DNDA es *13-74-371*.



Inverse Reinforcement Learning Application for Discrete and Continuous Environments

Yeison Suarez, Carolina Higuera(✉), and Edgar Camilo Camacho

Universidad Santo Tomás, Bogotá, Colombia
{yeisonsuarez,carolinahiguera,edgarcamacho}@usantotomas.edu.co

Abstract. We show the application of inverse reinforcement learning (IRL) in discrete and continuous environments based on the apprenticeship focus. The objective is to learn a mathematical definition of a task based on trajectories made by an expert agent. To achieve this, there must be features functions that in some way describe abilities that the agent can learn. Therefore, the description of a task can be formulated as a linear combination of those functions. This learning method was applied in Open-AI gym environments to show the learning process of a task using demonstrations.

AQ1

AQ2

Figura 13.1. Portada del artículo

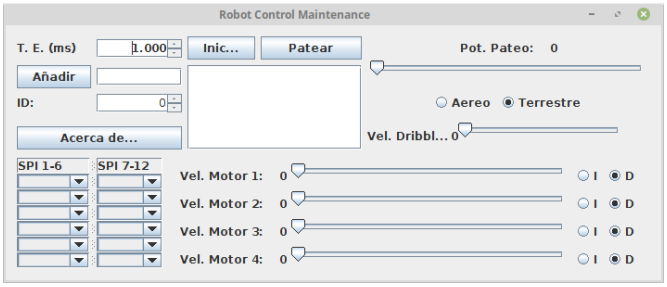


Figura 13.2. Programa de control utilizando OpenJDK 8

Bibliografía

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 1998.

[2] C. H. Yeison Suarez and E. C. Camacho, "Inverse reinforcement learning application for discrete and continuous environments," in *AETA 2019 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application*, Springer International Publishing, 2020.

[3] J. G. Matthieu Lapeyre, Pierre Rouanet, "The poppy project." `https://www.poppy-project.org/`, 2012.

[4] P. Manceron, "Ikpy library." `https://github.com/Phylliade/ikpy`, 2018.

[5] P. Abbeel and A. Y. Ng, ""apprenticeship learning via inverse reinforcement learning, " in Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, (New York, NY, USA)," 2004.

[6] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." `https://arxiv.org/pdf/1509.06461.pdf`, 2015.

[7] D. S. A. G. I. A. Volodymyr Mnih, Koray Kavukcuoglu, "Playing atari with deep reinforcement learning." `https://arxiv.org/pdf/1312.5602.pdf`, 2013.

[8] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa, *"Imitation learning for locomotion and manipulation, "in 2007 7th IEEE-RAS International Conference on Humanoid Robots.* Nov. 2007.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal, ""movement imitation with nonlinear dynamical systems in humanoid robots, " in Proceedings 2002 IEEE International Conference on Robotics and Automation," vol. 2, p. 1398–1403, May 2002.

[10] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, p. 263–279, 2013.

[11] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, ""a survey of robot learning from demonstration, " Robotics and Autonomous Systems," vol. 57, no. 5, p. 469, 2009.

[12] S. Calinon, F. Guenter, and A. Billard, ""on learning, representing, and generalizing a task in a humanoid robot, " IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)," vol. 37, p. 286–298, Apr. 2007.

[13] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, *"Learning and generalization of motor skills by learning from demonstration, " in 2009 IEEE International Conference on Robotics and Automation.* May 2009.

[14] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel, ""deep imitation learning for complex manipulation tasks from virtual reality teleoperation, " CoRR," vol. abs/1710.04615, 2017.

[15] B. C. Stadie, P. Abbeel, and I. Sutskever, *"Third-Person Imitation Learning, " ArXiv e-prints.* Mar. 2017.

[16] P. Sermanet, K. Xu, and S. Levine, ""unsupervised perceptual rewards for imitation learning, " CoRR," vol. abs/1612.06699, 2017.

[17] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine, *"Time-contrastivenetworks: Self-supervised learning from video.* 2018.

[18] B. Piot, M. Geist, and O. Pietquin, ""bridging the gap between imitation learning and inverse reinforcement learning," vol. 28, p. 1814–1826, Aug. 2017.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484 EP –, Jan 2016. Article.

[20] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys, *Deep q-learning from demonstrations.* feb 2018.

[21] G. Masuyama and K. Umeda, *"Apprenticeship learning based on inconsistent demonstrations, " in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2015.

[22] I. A. D. S. Tom Schaul, John Quan, "Prioritized experience replay." `https://arxiv.org/pdf/1511.05952.pdf`, 2016.