UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

MANIPULACIÓN MÓVIL MEDIANTE APRENDIZAJE REFORZADO PROFUNDO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

NICOLÁS GUILLERMO MARTICORENA VIDAL

PROFESOR GUÍA:
JAVIER RUÍZ DEL SOLAR SAN MARTÍN

PROFESOR CO-GUÍA:
FRANCISCO LEIVA CASTRO

MIEMBROS DE LA COMISIÓN:
SEBASTIÁN ISAO PARRA TSUNEKAWA
FRANCISCO RIVERA SERRANO

Este trabajo ha sido parcialmente financiado por FONDECYT N°1201170.

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: NICOLÁS GUILLERMO MARTICORENA VIDAL
FECHA: 2021
PROF. GUÍA: JAVIER RUÍZ DEL SOLAR SAN MARTÍN

MANIPULACIÓN MÓVIL MEDIANTE APRENDIZAJE REFORZADO PROFUNDO

En los últimos años se ha observado un aumento en el interés de parte de la industria en el desarrollo de plataformas móviles de manipulación, con aplicaciones que van desde el manejo de inventarios de bodegas, robótica de servicio, robots rescatistas, entre otras. Estos sistemas generan un gran interés debido a la gran cantidad de tareas que son capaces de realizar debido a la versatilidad entregada por la movilidad de una base móvil y la destreza de un brazo robótico. Sin embargo varias soluciones actuales desacoplan el sistema, no aprovechando completamente las sinergias entre ambas componentes. Junto también al creciente avance presente en la área del aprendizaje reforzado, nace el interés de abordar el problema de manipulación móvil como un problema de aprendizaje reforzado.

En el presente trabajo se estudia la resolución del problema de manipulación móvil, haciendo uso de las herramientas del aprendizaje reforzado, debido a la capacidad que poseen estas herramientas de aprendizaje a la hora de resolver tareas complejas caracterizadas por alta dimensionalidad de estados.

En una primera etapa, el problema es separado en sus dos principales tareas, navegación y manipulación. Luego es propuesta y utilizada una arquitectura de un administrador (*manager*), encargado de la coordinación en la ejecución de las tareas. Se busca que tanto las tareas, como también la coordinación de estás sean aprendidas mediante aprendizaje reforzado. Resultando en tres tareas a resolver mediante aprendizaje reforzado, las cuales son navegación, manipulación y un manager de ambas.

La metodología utilizada a lo largo de cada tarea, se basa en primera instancia en la formulación de la tarea como un proceso de decisión de Markov, detallando sus distintas componentes. Luego es desarrollada cada tarea considerando sus condiciones episódicas, la parametrización de políticas y detalles de entrenamiento. Finalmente cada política es entrenada y posteriormente validada tanto en simulación como en el mundo real.

Los resultados obtenidos posicionan a la arquitectura propuesta, basada en el aprendizaje de una tarea compleja mediante la combinación de, un administrador (*manager*) con subpolíticas de bajo nivel obtenidas al resolver las tareas de navegación y manipulación de forma independiente, como una alternativa viable a la hora de resolver la problemática de la manipulación móvil. Donde incluso su potencial aún no se encuentra totalmente estudiado contando con varias componentes que pueden ser mejoradas.

A mis padres, Andrea y Fernando por darme las herramientas para ser la persona que yo quiera ser.

Agradecimientos

Quisiera comenzar agradeciendo a mi familia, especialmente a mis padres por no solamente entregarme cariño, si no también de todas las herramientas necesarias para desarrollarme como persona. A mi hermano por las discusiones nocturnas sobre mi área de estudio. Lucca y Cuca por todos los domingos familiares que me alegraron en este difícil año.

Agradecer a todos los miembros del laboratorio del DIE, por todos los buenos momentos y aprendizajes realizados. En especial a los miembros que ya egresaron Luz, Rodrigo M., Gonzalo, Gustavo, Matias P. y Matias M. Por todas las enseñanzas y sabiduría entregadas. Como a los miembros actuales Ulises, Giovanni, Rodrigo S., Javier, Lukas, José, Cristopher, Leo, Camilo, Rafa, Edu, Pablo, Nacha, JP y Hans por todo el apañe durante todos estos años. Gracias Ulises por el apañe sacando resultados antes que cerrará la universidad.

Gracias a la comisión y sus miembros por todos los comentarios realizados, gracias Isao, por todas las lecciones, consejos y ayudas a lo largo de mi carrera. Siempre estabas dispuesto a responder alguna inquietud mía. También gracias por siempre estar preocupado de que contara con alguna ventana de tiempo para hacer uso de los equipos de la universidad. Muchas gracias Francisco por toda la ayuda y comentarios constructivos a la hora del desarrollo de este trabajo, las reuniones a altas horas de la noche, la disponibilidad fuera de horarios laborales y las discusiones sobre nuevas ideas que surgían.

Agradecer al profesor Javier Ruiz del Solar por abrirme las puertas tanto en el laboratorio, como en el mundo de la academia, por permitirme ser parte de cuerpos docentes y siempre estar muy dispuesto a lo largo de este trabajo.

Agradecer a mis amigos, a los que conocí en la universidad y los que me acompañaron durante todo este proceso. Claudia, Pablo, Pep, Tomás y Claudito gracias por todas las conversaciones, buenos momentos y amistad. Pablo, Michelo, Oliden, Pipe, Joseto, Pato y Nico gracias por la amistad, tilteos y las magic. Gracias Camila, por acompañarme todos estos años por ser una buena amiga, compañera y ayudarme siempre que estaba en problemas.

Agradecer también al proyecto FONDECYT N°1201170 por haber financiado parcialmente el desarrollo de este proyecto.

Tabla de Contenido

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

1.1. Antecedentes generales

En la presente sección, se exponen antecedentes generales del problema de estudio, donde se busca entregar al lector tanto las características como desafíos de los manipuladores móviles, luego se mencionan los diversos avances de las técnicas del área del aprendizaje reforzado profundo, abarcando aplicaciones en robótica haciendo hincapié las ventajas que presentan frente a técnicas clásicas.

1.1.1. Manipuladores móviles autónomos

Los manipuladores móviles autónomos conocidos como AMM por sus siglas en inglés *Autonomous mobile manipulator*, son sistemas complejos de robótica que consisten generalmente de brazos robóticos montados sobre bases móviles. Estos sistemas cuentan con una gran movilidad, la cual es entregada por la base y una gran destreza a la hora de interactuar con el entorno otorgada por el manipulador [3]. El estudio de estas plataformas robóticas es conocido como manipulación móvil.

Para definir de forma correcta los objetivos y problemáticas que forman parte de los desafíos de manipulación móvil, resultan útiles las definiciones establecidas por el Comité Técnico de manipulación móvil de IEEE-RAS[1] [5], donde en primer lugar se define el objetivo de estos sistemas como: "El objetivo de un AMM es la ejecución de tareas complejas de manipulación en ambientes no estructurados y dinámicos". Dentro de la información disponible en este comité se encuentran los diversos requisitos y características del problema a resolver, entre los que se destacan:

- **Requisitos:**

  Generalización: Los sistemas deben realizar diversas tareas, adquirir nuevas habilidades y aplicarlas en situaciones desconocidas, por lo que se vuelve necesario que sean capaces de adaptarse e incluso mejorar su desempeño.

[1] El IEEE Robotics and Automation Society es una sociedad profesional del IEEE que vela por el desarrollo de la teoría y la practica de la robótica y la automatización [4]

Incerteza: Deben ser capaces de resolver problemas derivados de la incerteza tanto en el actuado como el sensado de forma explícita.

- **Características:**

  Alta dimensionalidad en estados: Sistemas versátiles que poseen variados actuadores y sensores lo cual llevan al sistema a poseer un espacio de estados de alta dimensionalidad, entendiendo como espacio de estados a las n variables que pueden entregar el estado de un sistema en un momento dado.

  Sistemas complejos: Integración de diversos componentes donde se incluyen sensado, manipulación y locomoción, por lo que es requerida una gran sinergia entre las capacidades de percepción, manipulación, aprendizaje, control y planificación.

1.1.2. Aprendizaje Reforzado en robótica

En los recientes años se han generado grandes avances en el área del aprendizaje reforzado profundo conocido en inglés como *Deep Reinforcement Learning*, desde ahora referido como DRL, entre los cuales se pueden destacar los trabajos de AlphaGO [6], el primer algoritmo capaz de ganarle al campeón mundial del deporte GO, o algoritmos capaces de alcanzar desempeños similar a un ser humano en vídeo juegos [7]. Estos y otros trabajos posicionan a los algoritmos de aprendizaje reforzado como soluciones ideales en casos donde se busque aprender comportamientos mediante interacciones.

Dentro de los grandes avances en el área del aprendizaje reforzado se pueden destacar diversos trabajos aplicados en robótica. Entre estos destacan aplicaciones en navegación [8], manipulación [9] e incluso tareas de control complejas como caminatas ágiles de robots cuadrúpedos [10], alcanzando incluso desempeños capaces de competir con alternativas clásicas.

1.2. Motivación

Dentro de las principales motivaciones de este trabajo se encuentra el creciente interés por parte de la industria en plataformas móviles de manipulación. Esto se ve reflejado en el aumento de plataformas comerciales masivas donde se pueden mencionar algunas como *Fetch* de la compañía *Fetch Robotics* [11], robot utilizado en aplicaciones de almacenamiento, HSR [12] la plataforma de robótica de servicio de los laboratorios de Toyota e incluso plataformas que buscan abaratar costos, tales como *Strech* de la compañía *hello-robot* [13], este aumento facilita y masifica el desarrollo de algoritmos para esta problemática. Dentro de las aplicaciones donde estas plataformas pueden ser aplicadas se encuentran logística de almacenamiento [14, 15, 16], rehabilitación asistida [17] y robótica de servicio [18].

Otras de las motivaciones que surgen consiste en el no uso de sinergias intrínsecas del problema, esto se ve evidenciado en que la gran mayoría de soluciones propuestas para este problema desacoplan el sistema, resolviendo entonces dos problemas separados, la navegación a un punto viable y, por otro lado, el de manipulación considerando que el brazo se encuentra fijo a un punto. Si bien estas formulaciones son capaces de manipular objetos, presentan una gran desventaja por el hecho de no aprovechar las sinergias existentes entre estos dos sistemas, entendiéndose estas sinergias como la capacidad de realizar ambas acciones al mismo tiempo, logrando, por ejemplo, tiempos de ejecución más cortos debido a que el robot es capaz de

posicionar el brazo mientras la base móvil aún se encuentra en movimiento.

Por otro lado, cabe destacar que si bien existen formulaciones acopladas conocidas como control de cuerpo completo o *Whole Body Control* (WBC)[19, 20, 21], estas requieren de modelos precisos de toda la cinemática del robot, por lo que alternativas como el aprendizaje reforzado profundo emergen como una solución interesante a esta problemática, sin la necesidad de un extenso estudio específico a cada robot.

1.2.1. Definición del problema

La problemática a resolver consiste en el aprendizaje de políticas que controlen un manipulador móvil a la hora de acercar su efector final a una posición 3D, lo cual es uno de los sub-problemas estudiados en la manipulación móvil. El algoritmo a utilizar para la obtención de la política será basado en aprendizaje reforzado profundo.

El objetivo general del robot es llevar el manipulador situado en el efector final a una posición 3D dada con respecto al origen del robot, manteniendo un sistema de percepción que permita al robot trasladarse durante su trayectoria. Esta posición es la ubicación del objeto a manipular, el entregar directamente esta ubicación remueve la componente de percepción dentro del proceso de aprendizaje.

Entre los requerimientos de la solución se encuentran:

- La política de control debe ser de cuerpo completo, lo cual consiste en que el robot haga uso tanto de la base móvil como del brazo robótico de forma conjunta.
- La política de control debe ser capaz de evadir obstáculos fijos y móviles, todo esto con el fin de que el robot pueda ser utilizado en un ambiente no estructurado y dinámico.
- La política obtenida luego del entrenamiento en simulación debe ser validada en un sistema real, donde se espera que los comportamientos del robot sean similares a los obtenidos en simulación.

1.2.2. Oportunidad

Existe un vasto desarrollo respecto al aprendizaje reforzado profundo en diversas áreas de estudio incluyendo la robótica, en las cuales se han presentado diversos trabajos que resuelven alguna de las dos principales tareas incluidas en los manipulador móviles [9, 8, 22, 23]. Sin embargo, los desarrollos actuales que hacen uso de aprendizaje reforzado para resolver el problema de manipulación móvil, carecen de las siguiente características: Buenos resultados al ser validados en un sistema físico o bien carencia de una validación física [16, 24, 25], no poseer un sistema de evasión de obstáculos dinámicos [24] o bien buscan resolver una generalización del problema mediante la restricción de movimientos de parte del brazo (por ende simplificando bastante el espacio de estados y por ende la dificultad) [26].

Agregando a esto, el gran interés que los manipuladores móviles autónomos generan en la actualidad y los crecientes avances en trabajos relacionados a la navegación local basada en DRL [1, 8, 27, 28, 29, 30], como también en la tarea de *Reaching* [31, 32, 33] parte del problema de manipulación, es que surge la oportunidad de extender estos sistemas, buscando aplicarlos a este problema en particular. Se espera que ideas utilizadas en las distintas formulaciones

basadas en DRL, logren ser aplicadas en este trabajo, ya que distintos requerimientos de estos sistemas tales como, la evasión de obstáculos fijos o dinámicos son igualmente requeridos.

1.3. Objetivos

En esta sección son presentados los objetivos a realizar a lo largo de este trabajo de Título, considerando en una primera instancia el objetivo general y luego se exponen los objetivos específicos.

1.3.1. Objetivo general

El objetivo general es la obtención de una aplicación funcional basada en aprendizaje reforzado profundo que sea capaz de resolver un sub-problema de la manipulación móvil, más específicamente el de llevar al efector a una posición 3D deseada en el mapa. El algoritmo debe ser capaz de evadir obstáculos fijos y móviles. Todo esto validado en un ambiente real con un robot *Clearpath Husky A200* [34] con un brazo *Universal Robots UR5* [35] montado sobre el, como también dos sensores laser Hokuyo UTM-30LX-EW montados lateralmente. Esta plataforma robótica denominada AMTC Husky es presentada en la Figura 1.1, nombre el cual es dado debido a que este desarrollo propio del AMTC[2] presenta diferencias con el paquete de manipulación móvil oficial desarrollado por la empresa Clearpath [36].

Figura 1.1: Render de plataforma de manipulación móvil AMTC Husky.

1.3.2. Objetivos específicos

A continuación se presentan los objetivos específicos de este trabajo:

- Creación de simulación de la plataforma de estudio, donde se debe considerar los diversos actuadores y sensores a utilizar.

[2]Advanced Mining Techonology Center (AMTC por su sigla en inglés) es el centro de investigación de tecnología aplicada en la minería de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile.

- Generación de ambientes de entrenamiento y validación en simulación, donde se consideran entornos simples, y otros más complejos y desafiantes.
- Implementación del algoritmo de aprendizaje, considerando el diseño de la parametrización de políticas de control.
- Validación de políticas en el mundo real, considerando las políticas obtenidas para las tareas de navegación local, *Reaching* de posición y manipulación móvil.

1.4. Estructura de la Memoria

La presente Memoria posee la siguiente estructura, en primer lugar son formalizados todas las problemáticas a resolver por algoritmos de aprendizaje reforzado profundo. Luego son entregados los detalles de la implementación realizada para resolver cada una de estas tareas, junto también a las curvas de aprendizaje obtenidas.

El Capítulo 2 presenta el marco teórico y el actual estado del arte con respecto tanto al aprendizaje reforzado como al de la problemática de la manipulación móvil. El capítulo comienza con una breve explicación del aprendizaje reforzado , continuando con las formalización de los procesos de aprendizaje. Luego son explicados los principales algoritmos de aprendizaje que serán utilizados durante el desarrollo de esta tesis. Posteriormente se presenta el estado del arte en el problema de manipulación móvil, abarcando desde los últimos trabajos clásicos e incluyendo las propuestas basadas en aprendizaje reforzado profundo. Haciendo énfasis en cuales son las problemáticas que no son resueltas en los enfoques clásicos y cuales son las deficiencias que poseen enfoques basados en aprendizaje reforzado que se buscan resolver en este trabajo.

El Capítulo 3 exhibe de manera formal la metodología utilizada. Se presentan los enfoques propuestos para la resolución del problema general y de cada una de las sub-tareas que requieren ser aprendidas. Se formalizan los procesos de aprendizaje de cada una de las subtareas donde se incluyen la definición de los problemas a resolver y se entregan los detalles de las soluciones propuestas para cada tarea. Finalmente, se presentan los detalles de la formulación de la tarea principal, incluyendo la motivación de la metodología utilizada, como también de las características que posee el acercamiento propuesto.

El Capítulo 4 ofrece y justifica los diversos detalles utilizados en el entrenamiento .Luego son presentadas las curvas de aprendizaje obtenidas en cada una de las tareas, haciendo uso exclusivo de las herramientas de simulación. Finalmente, son presentados los resultados de las políticas entrenadas en simulación en la plataforma real tomando en consideración solamente los mejores resultados obtenidos en la simulación.

El Capítulo 5 expone la validación de las políticas obtenidas a lo largo del desarrollo, donde se detallan los resultados obtenidos en simulación, entregando métricas de desempeño, para luego presentar la validación en el mundo real, donde se comentan los experimentos realizados, junto a visualizaciones de los comportamientos del agente a lo largo de los episodios.

Finalmente el Capítulo 6 entrega las conclusiones del trabajo realizado, haciendo énfasis en los resultados obtenidos. También aporta algunos lineamientos para trabajos futuros.

Capítulo 2

Marco teórico y estado del arte

2.1. Aprendizaje reforzado

El aprendizaje reforzado es un proceso de aprendizaje, donde se busca que un agente sea capaz de realizar acciones según la situación en la que se encuentra con tal de maximizar una señal numérica de recompensa [2].

Las principales componentes del aprendizaje reforzado radican en, al tratarse de un agente aprendiendo a interactuar con un ambiente con tal de cumplir una meta, es que estos agentes deben ser capaces de sensar su estado en el ambiente, como también ser capaces de realizar acciones que modifiquen su estado y por último contar con un objetivo definido a realizar. Dentro de los desafíos que se presentan durante el proceso de aprendizaje reforzado es el diseño de los estados y recompensas, ya que estos serán vitales a la hora de que el agente aprenda principios de causa y efecto, donde incluso se deben aprender efectos a largo plazo.

Para poder definir formalmente los procesos de interacción entre el agente y su ambiente se presentan los procesos de decisión de Markov [37], utilizados de forma estándar en tareas de aprendizaje reforzado. La definición entregada a continuación se encuentra basada en [2].

2.1.1. Proceso de decisión de Markov

Los Procesos de decisión de Markov o MDPs de las siglas en inglés *Markov Decision Process*, son una formalización clásica de modelar procesos secuenciales de toma de decisiones, en los cuales las acciones tomadas no sólo generar recompensas inmediatas si no también a los estados posteriores o situaciones, y por ende, a las recompensas futuras.

Los MDPs son una forma matemática idealizada del problema del aprendizaje reforzado, donde dentro de las definiciones se encuentran los *Agentes* como los entes encargados del aprendizaje y la toma de decisiones, además de definir los *Ambientes* como el espacio donde el agente interactúa considerando todos los componentes externos al agente. El principio de interacción entre un agente y su ambiente se ve representado en la Figura 2.1, esta interacción se basa en primer lugar por la realización de una acción de parte del agente, a lo cual, el ambiente responde con una nueva situación junto a una señal de recompensa relacionada al

objetivo.

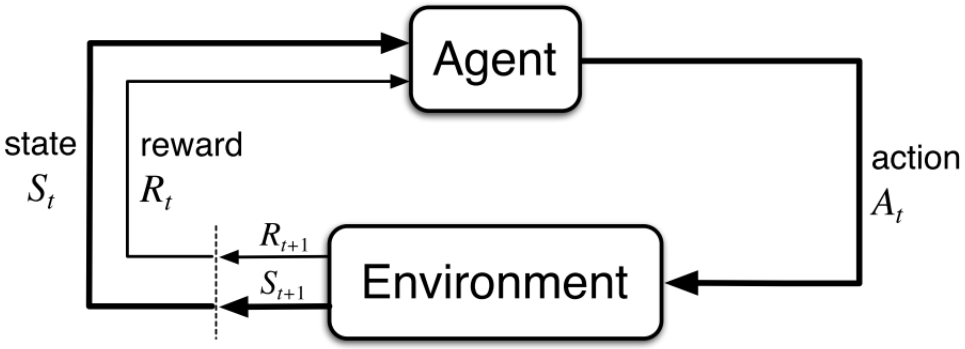


Figura 2.1: Ambiente de interacción de agente-ambiente en un MDP
Fuente: [2].

En un proceso de decisión secuencial, el agente y el ambiente interactúan en una secuencia de pasos de tiempo discretos $t = 0, 1, 2, ...$ Para cada uno de los pasos de tiempo el agente obtiene alguna representación del estado del agente S_t, y realiza una acción A_t, para luego, como consecuencia de la acción tomada el agente llegue a un nuevo estado S_{t+1} y obtenga una recompensa R_t.

El objetivo del aprendizaje reforzado es que el agente interactué con el ambiente mediante acciones, con la finalidad de maximizar la señal de recompensa, donde no solamente se maximice la recompensa instantánea, si no, más bien la recompensa acumulada al largo plazo. A la hora de definir el objetivo formalmente se define el concepto de una función de retorno, definido por una suma acumulada de las recompensas futuras, con la adición de un término de descuento γ, término que toma valores en el rango $[0, 1]$ y posee como finalidad la de evitar que la función de retorno adquiera valores infinitos en casos donde se tengan interacciones sin un instante final de tiempo.

La definición de la función de retorno se encuentra en la ecuación 2.1, donde R_t son aquellas recompensas asignadas a cada instante de tiempo o *pasos* de la interacción.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \tag{2.1}$$

Junto también a la capacidad que poseen los agentes de realizar acciones que afecten su estado actual, estos deben ser capaces de la toma de decisiones a la hora de seleccionar la acción a realizar, este mapeo entre estados y acciones a realizar se define formalmente mediante una política. Una política define el comportamiento a realizar por un agente, si un agente está siguiendo una política π en un instante de tiempo t, entonces existe una probabilidad $\pi(a \mid s)$ de que la acción $A_t = a$ dado que $S_t = s$.

El objetivo principal del aprendizaje reforzado es el de obtener políticas óptimas, donde una política óptima es aquella que maximice la función de valor $v_\pi(s)$ y la función de acción valor $q_\pi(s, a)$, las cuales están definidas por:

- La función de valor esperado de retorno, dado que el agente se encuentra en un estado **s** y sigue una política π, esta se encuentra presentada en la ecuación 2.2

$$v_\pi(s) \doteq E_\pi[G_t \mid S_t = s] \tag{2.2}$$

- La función de valor del par estado-acción es el valor esperado de retorno, dado que el agente se encuentra en un estado y realiza una acción a y posteriormente seguir una política π, esta se encuentra presentada en la ecuación 2.3

$$q_\pi(s, a) \doteq E_\pi[G_t \mid S_t = s, A_t = a] \tag{2.3}$$

2.1.1.1. Procesos de decisión de Markov parcialmente observables

Los procesos de decisión de Markov parcialmente observables o POMDPs por sus siglas en inglés (Partially observable Markov decision process), son una generalización de los MDPs, donde el agente no pude acceder de forma directa a los estados, si no a observaciones las cuales entregan información sobre el estado actual.[38]

La interacción de agente-ambiente formulada como una POMDPs, consiste en que en cada instante de tiempo t, el agente realiza una observación del ambiente o_t y ejecuta una acción a_t basada en una política $\pi(a_t \mid o_t)$, recibiendo una recompensa r_t y transicionando a un nuevo estado s_{t+1} y por ende el agente obtiene una nueva observación o_{t+1}.

El objetivo del aprendizaje reforzado aplicado en esta formulación es el mismo, obtener políticas óptimas, las cuales maximicen la función de valor $v_\pi(s)$, donde la diferencia radica que al agente no tener conocimiento absoluto del estado actual, este realiza acciones basadas en la esperanza de estar en un estado basado en las observaciones realizadas.

2.1.2. Q-Learning

Este algoritmo introducido por Watkins en 1989 [39] es definido bajo la siguiente regla de actualización.

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a q(s_{t+1}, a) - q(s_t, a_t)] \tag{2.4}$$

El principio es que mediante la interacción con el ambiente el agente sea capaz de aprender su función de valor del par estado-acción. Luego cuando termina el entrenamiento uno debe evaluar esta tabla de Q(s,a) en cada estado para obtener la mejor acción posible, la cual es obtiene por evaluar la siguiente ecuación:

$$a_{best} = max_a q(s, a) \tag{2.5}$$

Donde durante el entrenamiento es utilizada una estrategia ε greedy para la exploración, esta estrategia consiste en que con una probabilidad de 1-ε se seleccion la acción determinada por la Ecuación 2.5, mientras que con una probabilidad ε es seleccionada una acción aleatoria dentro del espacio de acciones discretas posibles.

2.2. Aprendizaje reforzado profundo

En la presente sección son exhibidos diversos avances en el área del aprendizaje reforzado profundo o DRL por sus siglas en inglés (Deep Reinforcement Learning), la cual se caracteriza con respecto al aprendizaje reforzado tradicional por el uso de redes neuronales profundas.

2.2.1. Deep Q-Networks (DQN)

El algoritmo DQN fue introducido por Mnih en los trabajos *Playing Atari with Deep Reinforcement Learning* [40] y *Human-level control through deep reinforcement learning* [7]. Fue el primer gran avance en lo que respecta a DRL, ya que permite aprender la función de Q al igual que en *Q-Learning*, pero mediante una red neuronal profunda, siendo la principal diferencia que DQN aproxima la función Q mediante una red neuronal.

Surge debido a que al aumentar la cantidad de estados posibles la tabla a la cual se llega en *Q-Learning* se vuelve insostenible, por lo que surge esta nueva idea de que la función de estado acción sea modelada y aproximada mediante el uso de redes neuronales.

Este trabajo también popularizó el mecanismo de *Experience replay*[41], el cual consiste en contar con buffer de experiencias pasadas, las cuales son muestreadas durante el entrenamiento. La ventaja que se obtiene por el uso de este mecanismo es que al realizar el descenso por gradiente, el tener muestras uniformes a lo largo de las experiencias vividas por el agente, suaviza el entrenamiento.

2.2.2. Deep Deterministic Policy Gradient (DDPG)

DDPG[42] surge como una solución a problemas con acciones continuas, esto debido a que Deep Q-Networks solamente funciona con acciones discretas, ya que por cada posible acción que pueda realizar el agente se debe poseer una salida de la red, lo cual se vuelve insostenible al tener acciones continuas si uno no desea perder mucha información discretizando.

Este algoritmo hace uso de dos redes, una conocida como *Actor* y la otra como *Critic*, con las funciones A(s) y C(s,a):

- *Actor* A(s), tiene como finalidad aprender cual es la acción óptima dado que el agente se encuentra en un estado s.
- *Critic* C(s,a), tiene la función de estimar la función de valor, la cual es entregar el valor esperado de retorno al realizar cierta acción a en un estado s.

2.3. Estado del arte en manipulación móvil

2.3.1. Planificadores basados en muestreo

Las soluciones basadas en planificadores de muestreo [43, 44] consisten en la construcción de árboles jerárquicos donde sus nodos y hojas son compuestos por diversas configuraciones, estas configuraciones se componen por los ángulos de las diferentes articulaciones del brazo robótico junto también a la pose 2D de la base en el mapa. Estas configuraciones deben

cumplir el requisito de ser libre de colisiones para ser consideradas viables. Los algoritmos de exploración para la construcción de los árboles varían según la implementación, pero mantienen el factor común de que los nodos del árbol están compuestos por las configuraciones de la plataforma.

Una vez obtenido el árbol, este es muestreado para obtener las diversas configuraciones que determinan la trayectoria a seguir por el manipulador móvil. Dichas configuraciones son interpoladas con el fin de obtener una trayectoria continua a seguir.

Las principales problemáticas de estas soluciones consisten principalmente en el requerimiento de un conocimiento completo del ambiente para obtener una solución viable para el sistema, por lo que al operar en un ambiente desconocido y dinámico agrega restricciones la utilización de un planificador *off-line*.

2.3.2. Control Predictivo

Las soluciones basadas en control predictivo [21, 19, 20] buscan solucionar el problema de planificación obteniendo una secuencia de acciones de control dada una función del objetivo. Estos métodos se basan en la teoría de control de sistemas, donde la componente predictiva es que estos controladores se basan en modelos, siendo éstos la piedra angular de esta metodología los cuales deben ser bien estudiados y formulados.

La principal ventaja que presentan estos controladores con respecto a controladores clásicos, es que estos sistemas son capaces de predecir acontecimientos futuros y poder tomar decisiones basadas en estos. Principalmente estos controladores son utilizados en sistemas dinámicos complejos.

Las problemáticas encontradas en estas soluciones consisten principalmente en el costo que implica el estudio del modelo cinemático del sistema, por la importancia que poseen a la hora del desempeño de éstos, junto también al ser estos sistemas limitados a la hora de evasión de obstáculos ya que estos deben ser modelados en la función objetivo, como también considerados en el modelo del sistema.

2.3.3. Propuestas basadas en DRL

A la hora de evaluar trabajos que busquen resolver el problema de manipulación móvil mediante aprendizaje reforzado, se debe definir primero que se considera como tal. En esta sección son considerados aquellos trabajos que buscan resolver la manipulación móvil como un problema completo, considerando tanto el brazo robótico como la base móvil, o bien trabajos que buscan obtener controladores de cuerpo completo para una plataforma AMM.

En el trabajo presentado por Wang et al. [24] se propone un sistema de manipulación basado en DRL de control de cuerpo completo aplicado a un manipulador móvil. Esta solución utiliza un detector de pose de objetos externo, con el fin de sobrellevar las diferencias entre la simulación y el mundo físico. En el proceso de aprendizaje es utilizado el algoritmo *Proximal Policy Optimization* (PPO) [45] pero también incluyen comparaciones con diferentes algoritmos de aprendizaje reforzado. Si bien el detector de objetos ayuda a sobrellevar las diferencias de la simulación no se logran resultados satisfactorios, estos principalmente

explicados debido a las oclusiones que el brazo genera en la cámara encargada de la detección del objeto

En el estudio realizado por Iriondo et al. [16] se presenta una solución que busca resolver la tarea de *Pick and Place*[1] con un robot KUKA IIWA, mediante aprendizaje reforzado profundo. Esta consiste en obtener un controlador para la base robótica el cual se comunique con una solución clásica de manipulación independiente que se aplica para el brazo robótico. Para llevar a cabo el entrenamiento se comparan los algoritmos de aprendizaje DDPG y PPO adamas de utilizar dos ambientes de entrenamiento donde se busca ir aumentando la dificultad al agregar obstáculos que el robot debe ser capaz de sortear. Las principales fallas de esta propuesta consisten en que el sistema no es validado en una plataforma física, al igual de no considerar obstáculos dinámicos y a no modelar el problema como un control de cuerpo completo ya que sólo se realizan acciones en la base móvil.

En la propuesta de Kindle et al. [26] se presenta un controlador de cuerpo completo basado en DRL como una alternativa que logre sobrellevar los problemas presentes en planificadores de trayectoria y controladores predictivos basados en modelos. La problemática que se busca resolver es el de control completo de un manipulador móvil restringido a un plano mientras el robot es capaz de esquivar obstáculos fijos, en la plataforma RoyalPanda una plataforma desarrollada en la universidad ETH Zurich constituida por un brazo robótico Franka Emika Panda montado sobre una base móvil clearpath Ridgeback. La solución propuesta se basa en el algoritmo de aprendizaje PPO [45]. La política es entrenada en simulación y posteriormente es validada en una plataforma física. Se considera que el presente trabajo no resuelve la problemática a estudiar debido a que al restringir el movimiento del brazo a un solo plano no se considera el problema con todos sus grados de libertad, simplificando el problema.

2.4. Navegación basada en DRL

Otras de las áreas ampliamente estudiadas sobre DRL aplicada en robótica es la resolución del problema de navegación [1, 8, 27, 28, 29, 30], este problema es de gran interés a la hora de analizar posibles alternativas de parametrizaciones de políticas debido al ser este uno de los sub-problemas relacionados con la manipulación móvil.

En el estudio realizado por Leiva et al. [1] se realiza un profundo estudio y comparación de distintas formulaciones para la interpretación de la información entregada por los sensores del robot, donde se entrega una aproximación robusta para el diseño de planificadores locales basados en aprendizaje reforzado, como también un analisis del impacto que diferentes diseños de observaciones y parametrizaciones cuando se hace uso directo de mediciones de rango y por último de las ventajas del desacople de las características de los sensores y como se puede utilizar esto para mejorar los desempeños en la validación a realizar en el mundo real.

Al ser la evasión de obstáculos un requisito en esta aplicación junto también a la necesidad de un manipulador móvil de contar con una navegación, es que dentro de las decisiones de diseño será utilizado como sistema base el framework de trabajo presentado por Leiva [1],

[1]La tarea de *Pick and Place*, consiste en un robot sea capaz de levantar un objeto (Pick) y desplazarlo a otro lugar (Place), normalmente hacer referencia a tareas de ensamblaje, empaquetado, recolección de colectores o inspección de líneas de producción.

buscando extenderlo a esta aplicación ya que incluyen las ideas introducidas en el trabajo anteriormente mencionado. A lo largo de este trabajo serán utilizados tanto una de las parametrizaciones propuestas, los ambientes de entrenamiento, los hiperparámetros de entrenamiento y la función de recompensa expuesta en el trabajo *Robust RL-Based Map-less Local Planning: Using 2D Point Clouds as Observations* [1], considerando pequeñas modificaciones debido a la diferencias de la plataforma de estudio.

2.5. *Reaching* de posición basado en DRL

Las habilidades de poder manipular objetos, es un área ampliamente estudiada en sistemas robóticos, debido a la gran cantidad de aplicaciones donde pueden ser utilizados, ya sea desde robótica de servicio, aplicaciones de ensamblaje, soldado en industria manufacturera e incluso en medicina [46]. Los manipuladores robóticos son de interés en el desarrollo de este trabajo, principalmente por ser uno de los componentes presentes en los manipuladores móviles y por ende en la tarea de manipulación móvil.

El problema de *Reaching* es uno de los sub-problemas de la manipulación robótica, ya que es la componente a cargo de guiar al efector final del brazo robótico a un objeto a manipular, donde al igual que en el caso de la navegación ha sido ampliamente estudiada mediante aprendizaje reforzado en variados trabajos [31, 32, 33, 47, 48, 49], donde principalmente en este trabajo de estudio serán utilizadas las componentes en común presentes en el diseño de las funciones de recompensa.

Cabe mencionar que el trabajo presentado por Cannon Lewis el 2019 titulado *How Much Do Unstated Problem Constraints Limit Deep Robotic Reinforcement Learning?* [31], sirvió como una guía en el diseño de esta tarea, ya que en este trabajo se hace uso del mismo brazo robótico (UR5) además de presentar como diversas restricciones geométricas afectan en el aprendizaje, siendo esto último de vital importancia en la solución propuesta. Los demás trabajos revisados sobre este problema en particular sirvieron a la hora del diseño de la función de recompensa utilizada, donde también cabe mencionar que el trabajo *Setting up reinforcement learning task with a real-world robot* provee de resultados experimentales del porque en caso particular del brazo robótico UR5, es preferible el uso de acciones de velocidad en las articulaciones.

Capítulo 3

Formulación del problema

Tal y como es mencionado en la introducción, la problemática a resolver consiste en la obtención de políticas de control para un AMM (ver Sección 1.1.1), los cuales al consistir de un brazo robótico montado sobre una base móvil, estos dos sistemas robóticos poseen un amplio estudio en torno a ellos. Cada uno de estos componentes son utilizados ampliamente en aplicaciones de robótica, existiendo una amplio campo de estudio en torno a cada una de las capacidades que entregan de forma separada. Dentro de estos campos se pueden mencionar navegación aplicada a robótica móvil y manipulación robótica.

La metodología propuesta se basa en parte en el trabajo introducido por Duan et al. [50]. En dicho trabajo se utiliza una arquitectura de aprendizaje reforzado jerárquico, donde se aprenden sub-políticas que realizan maniobras específicas de forma independiente para luego aprender una política maestra que toma la decisión de cual de las sub-tareas se encuentran activas en cada momento. Esta metodología es adaptada para la problemática de este estudio, definiendo para este caso dos sub-políticas a aprender: Navegación Local y *Reaching* de posiciones 3D; en conjunto también a un administrador o *Manager* encargado de seleccionar cada una de estas dos políticas o bien debido a las componentes que utiliza cada política, utilizar ambas simultáneamente. A lo largo de este capítulo se presentarán formalmente cada una de las sub-tareas a resolver junto también la tarea a resolver por el *Manager*.

La Sección 3.1 presenta la formulación de la primera sub-tarea a resolver, el problema de Navegación Local. En primer lugar, se explica a grandes rasgos en que consiste la navegación local abordando las desventajas que presentan enfoques clásicos. A continuación, se comentan diversos trabajos que utilizan herramientas de aprendizaje reforzado. Posteriormente se modela la tarea de Navegación Local haciendo uso de las herramientas introducidas en el marco teórico, formulando el problema como un *Proceso de decisión de Markov*.

Posteriormente, en la Sección 3.2 es presentado el segundo sub-problema a resolver denominado *Reaching de posición*. En primer lugar, se explica la problemática a resolver, luego se comentan trabajos relacionados a esta área que hacen uso de aprendizaje reforzado. Posteriormente se presentan diversas restricciones y simplificaciones que serán utilizadas en la resolución de este problema. Finalmente, se vuelve a hacer uso del las herramientas de los procesos de decisión de Markov a la hora de definir el problema.

Finalmente, en la Sección 3.3 se presenta el *Manager* de alto nivel, que tiene como función la toma de decisiones de cuál de los dos sub-sistemas se encuentra activo en cada instante. En primer lugar, se define cuáles son los alcances del sistema propuesto para luego definirlo formalmente como un problema de aprendizaje reforzado.

3.1. Navegación local

La navegación autónoma es una habilidad esencial en la robótica móvil y por lo tanto, ampliamente estudiada. Dentro de las formulaciones actuales se pueden separar en formulaciones clásicas y formulaciones basadas en aprendizaje de máquinas. Siendo estas últimas de gran interés debido a no requerir del extenso proceso de calibración de diversos parámetros.

Esta tarea es estudiada como una variante de la formulación desarrollada por Leiva et al. [1]; en dicha formulación son presentadas propuestas de diseño de observaciones y parametrizaciones de políticas basadas en aprendizaje reforzado, con las características de ser extensibles y robustas, siendo la presencia de una nube de puntos 2D de tamaño variable dentro de la observaciones, la característica que permite obtener empíricamente mejores resultados en contraste a otras propuestas, como también dota de la habilidad de poseer una alta robustez a perturbaciones en las observaciones.

El agente tiene como objetivo alcanzar una posición 2D objetivo (x,y), esto logrado mediante acciones de velocidad efectuadas en la base móvil. A la hora de definir si el robot se encuentra o no en la posición objetivo, es utilizado el eje principal de la base desde ahora referido como el eje principal del robot, el cual se encuentra definido en el centro de rotación del robot.

A lo largo de esta sección se presentan el modelamiento del problema donde se definen tanto las observaciones, acciones, las parametrizaciones de las políticas y la creación de objetivos durante el entrenamiento. Luego se introduce la función de recompensa comentando cada una de sus componentes.

3.1.1. Espacio de acciones

Debido a que la plataforma de estudio utilizada para el movimiento del robot es del tipo skid-steer, agregando que al igual que en el trabajo presentado por Leiva et al. [1] se consideran acciones continuas, ya que esto permite un control más fino de la base, es que se decide el uso de acciones a utilizar por el sistema de navegación se compongan por las siguientes:

- v_x: Velocidad lineal del robot [m/s], de la cual solamente será considerada la componente positiva, resultando en un espacio continuo en el rango $[0, v_x^{max}]$.
- v_θ: Velocidad rotacional del robot [rad/s], donde el rango de operación se compone por $[v_\theta^{min}, v_\theta^{max}]$.

3.1.2. Espacio de observaciones

Observaciones del agente se encuentran compuesta por tres componentes:

- O_{pcl}**:** Nube de puntos 2D, de tamaño variable, conformada por la información entregada por los láser montados sobre la plataforma. $(x_1, y_1, x_2, y_2, ..., x_n, y_n)$.
- O_{odom}**:** Estimación de las componentes de velocidad lineal y angular del agente $(\hat{v}_x, \hat{v}_\theta)$.
- $O_{\mathbf{target}}$**:** Posición relativa del punto objetivo con respecto al eje principal del robot en coordenadas polares $(\rho_{\text{target}}, \theta_{\text{target}})$.

En la figura 3.1 se presenta en color verde el diagrama de las acciones del robot junto a las observaciones O_{odom}, en rojo las observaciones correspondientes a O_{target} y por último en azul las observaciones O_{pcl} presentadas como el rango de medición de los lasers.

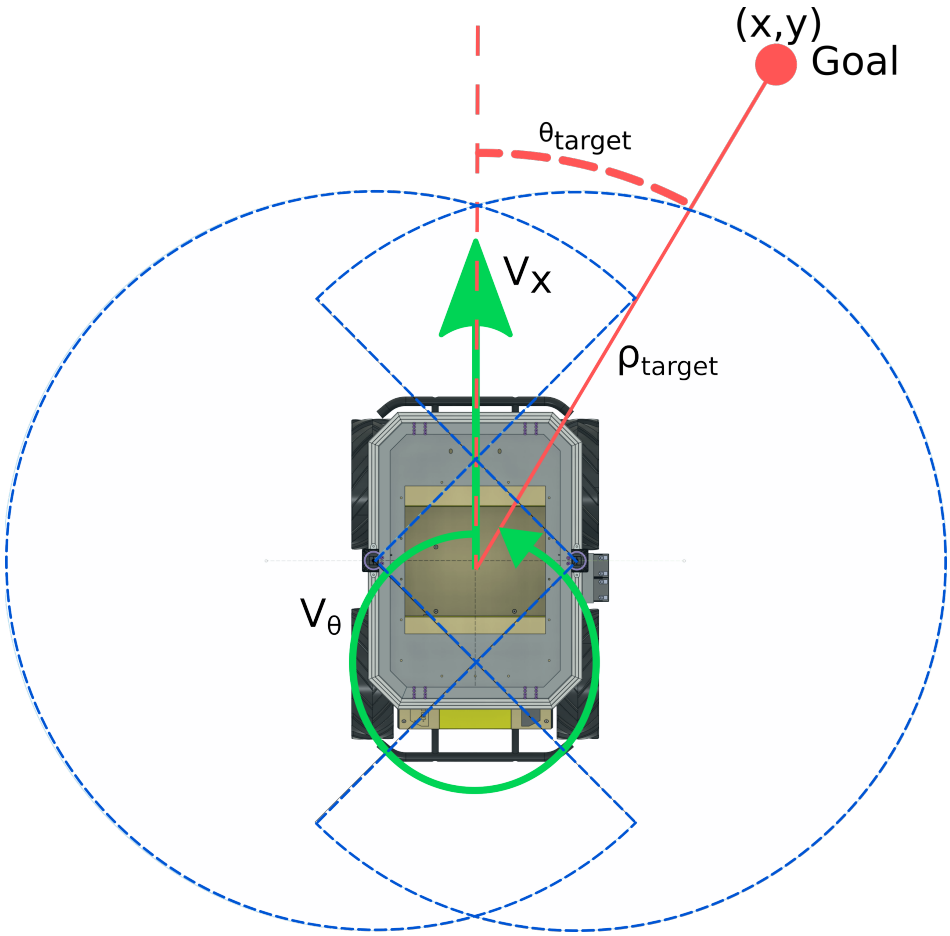
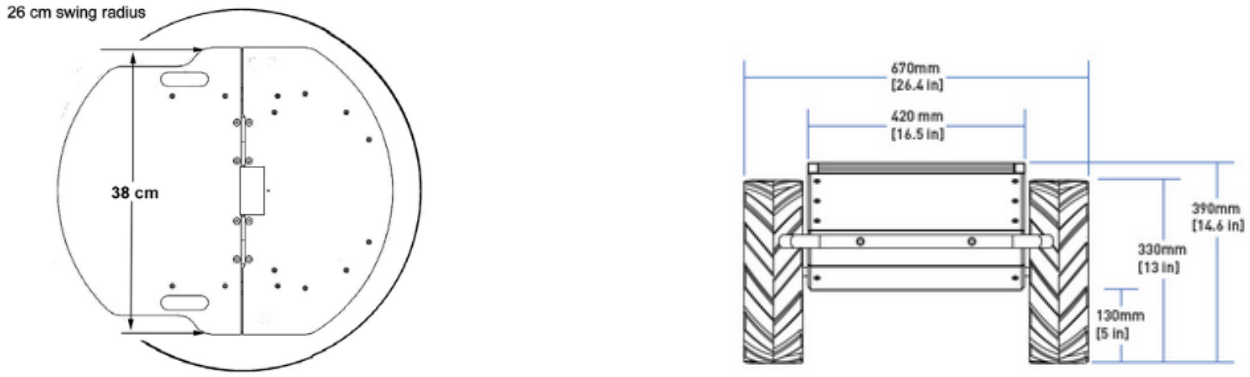


Figura 3.1: Diagrama de acciones de velocidad en la base móvil, observaciones tanto del objetivo como de los ángulos de visión que entregan los laser montados en la plataforma.

3.1.3. Ambiente de entrenamiento

El entrenamiento de las políticas es realizado en el simulador *Gazebo* [51], haciendo uso de los detalles de simulación expuestos en la Sección 4.1. El ambiente donde el robot es entrenado fue extraído del trabajo *Robust RL-Based Map-less Local Planning: Using 2D Point Clouds as Observations* [1] y re escalado, esto último debido a las diferencias de tamaño entre la plataforma Pionner 3-DX y Husky A200, diferencias que se pueden visualizar en la figura 3.2, por lo que un factor de re-escalado de 2X es aplicado. Una vez aplicado este factor

de re-escalado al ambiente de navegación complejo del trabajo mencionado [1] se obtiene el ambiente en Gazebo de dimensiones 16 x 16 m mostrado en la Figura 3.3, donde se incluye el robot en el centro del mapa a modo de entregar una referencia.



(a) Dimensiones Pionner 3-DX [52]

(b) Dimensiones husky [36]

Figura 3.2: Comparación dimensiones Adept Pionner 3-DX y Clearpath Husky A200.

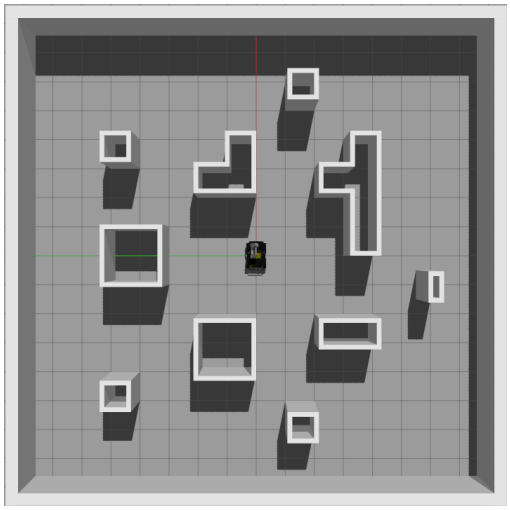


Figura 3.3: Ambiente de entrenamiento para navegación en Gazebo.

3.1.4. Función de recompensa

Tal y como es mencionado en el marco teórico 2.1, la función de recompensa es la componente encargada de guiar que comportamientos son deseados y cuales no. La función de recompensa a utilizar es una función diseñada para el problema de navegación y se basa principalmente en la presentada en el trabajo introducido por Leiva el año 2020 [1], con modificaciones en la componente de peligro, ya que en aquel trabajo se estudia la tarea de navegación en una plataforma con una geometría exterior similar a una circunferencia, a diferencia de la plataforma utilizada en este trabajo la cual posee una geometría exterior rectangular.

Esta señal de recompensa es presentada en la Ecuación (3.1), la cual se encuentra conformada por tres componentes. El primero $r^t_{\text{navigation}}$ consiste en la recompensa asignada a lo

largo de recorrido del robot, que se presenta en la Ecuación (3.2) y que tiene como objetivo el guiar al robot hacia el punto objetivo. El segundo componente $r^{\text{ success}}$ consiste en un valor positivo entregado cuando el robot logra llegar a la posición x,y objetivo, esto denotado bajo la condición que la distancia relativa del robot con la posición objetivo ($\rho_{\text{ target}}$) sea menor que un cierto umbral $\rho_{\text{ thresh}}$. Por último la señal $r^{\text{ collision}}$ consiste en un valor negativo que penaliza al agente en caso de que este colisione con alguno de los obstáculos que se encuentran en el ambiente.

$$r^t = \begin{cases} r^t_{\text{navigation}} & \text{si } \rho^t_{\text{target}} \geq \rho_{\text{thresh}} \\ r^t_{\text{success}} & \text{si } \rho^t_{\text{target}} < \rho_{\text{thresh}} \\ r^t_{\text{collision}} & \text{si el agente colisiona} \end{cases} \tag{3.1}$$

$$r^t_{\text{navigation}} = r^t_{\text{target}} + r^t_{\text{fov}} + r^t_{v\theta} + r^t_{\text{danger}} \tag{3.2}$$

Como fue mencionado, la componente $r_{\text{ navigation}}$ es la encargada de guiar al agente al objetivo, penalizando comportamientos no deseados. Esta se encuentre compuesta por los términos:

- r_{target} : Componente encargada de incentivar al agente a acercarse al objetivo. Incentiva el acercarse al objetivo a la mayor velocidad al mismo tiempo que penaliza al agente por alejarse del objetivo.

$$r^t_{\text{target}} = \frac{\hat{v}^t_x}{v^{\max}_x} \cos(\theta^t_{\text{target}}) + 5 \cdot \mathbb{1}_{\{\rho^t_{\text{target}}:\rho^t_{\text{target}}<\rho^{t-1}_{\text{target}}\}}(\rho^t_{\text{target}}) - 6 \tag{3.3}$$

- r_{fov} : Componente encargada de penalizar que el objetivo se encuentre fuera del campo de visión de 240° con respecto al eje principal del robot. Cabe destacar que al agregar esta componente se asume que los objetivos no requieren de planificación a largo plazo.

$$r^t_{\text{fov}} = (3\cos(\theta^t_{\text{target}}) - 5) \cdot \mathbb{1}_{\{\theta^t_{\text{target}}:\left|\theta^t_{\text{target}}\right|>120°\}}(\theta^t_{\text{target}}) \tag{3.4}$$

- r_{v_θ} **:** Término encargado de la penalización de altas velocidades y aceleraciones angulares. Se encuentra definido por 3.5 donde $K^t_{v_\theta}$ es definido en 3.6.

$$r^t_{v\theta} = -2K^t_{v_\theta} \cdot \mathbb{1}_{\{K^t_{v_\theta}:K^t_{v_\theta}>0{,}5\}}(K^t_{v_\theta}) \tag{3.5}$$

$$K^t_{v_\theta} = \frac{1}{2v^{\max}_\theta} \max\{2\left|\hat{v}^t_\theta\right|, \left|\hat{v}^t_\theta - \hat{v}^{t-1}_\theta\right|\} \tag{3.6}$$

- r_{danger} : Componente encargada de penalizar cuando el robot se encuentra en estados "peligrosos", considerando peligroso que el agente se encuentre muy cercano a un obstáculo. A la hora de evaluar esta función de recompensa, es considerada una distancia de umbral tanto en x como en y, siempre que alguna de las mediciones de O_{pcl} se encuentren dentro de este umbral se activa esta componente.

3.2. *Reaching* de posición

El segundo sub-problema a resolver consiste en la tarea de *Reaching* de Posición. Esta tarea busca acercar el efector final de un brazo robótico a una posición 3D (x, y, z) objetivo. La formulación del problema se basa principalmente en el ambiente *FetchReach-v1*, parte de las tareas robóticas incluidas en el entorno de desarrollo de aprendizaje reforzado *Gym* [53] los detalles de este ambiente se detalla en [33]. Junto también a las formulaciones de la tarea utilizadas en los trabajos [33], [31] y [47].

La principal motivación del estudio de esta tarea, es que consiste en una de las componentes a la hora de resolver el problema completo de manipulación de un objeto. En primera instancia, resolver este problema dota a un agente a ser capaz de acercar el efector final del manipulador al objeto a manipular deseado.

Cabe destacar que el efector final en un brazo robótico es considerado como aquella herramienta situada en el extremo del brazo robótico, encargada de interactuar con el objeto de interés, estos puedes ser *grippers* o bien otros tipos de herramientas como las de pulido [46]. Debido a que en este trabajo se busca resolver el problema de manipulación móvil, es considerado como efector final un gripper virtual, donde es considerado un sistema de referencia a una distancia de 0,12 m del último eslabón del brazo que no se encuentra fijo. Para decidir la distancia a la cual se consideraría el efector final, fue considerado un gripper robotiq 2F-85, donde fue extraída la distancia entre la base donde es montada en el brazo y el centro de las pinzas cuando se encuentran cerradas, donde en la Figura 3.4 es representado la pose del efector final mediante el eje de referencias dibujado.

3.2.1. Espacio de Acciones

Dado que la tarea a resolver consiste en acercar el brazo robótico a una posición objetivo, haciendo uso exclusivo del brazo, en que las acciones naturalmente deben ser efectuadas en cada una de las articulaciones. Al revisar tanto las interfaces de control que entrega el brazo UR5 como también el trabajo *Setting up a Reinforcement Learning Task with a Real-World Robot* [32] presentado el 2018, en el cual es diseñada una tarea de Reaching a ser entrenada en el mundo real haciendo uso de un brazo robótico UR5, dentro de los análisis de dicho trabajo es presentada una metodología para obtener trayectorias suaves haciendo uso de un controlador de posición, pero que al ser comparadas las respuestas del brazo real resultaba una mejor alternativa el uso de la interfaz de velocidad directamente, por lo que siguiendo estos lineamientos son utilizadas acciones de velocidad para cada una de las articulaciones.

Cabe destacar, que al igual que en la tarea de navegación, se decide utilizar acciones continuas, esto debido a que estas permiten ejercer control fino sobre cada uno de los grados de libertad del brazo, dando la posibilidad a obtener políticas más precisas. Al considerar todo lo mencionado da como resultado que las acciones a realizar por el sistema de *Reaching* se compone por:

- $V_{\phi_{1...6}}$: Velocidad angular [rad/s] de cada uno de los 6 grados de libertad del brazo UR5.

3.2.2. Espacio de observaciones

A la hora de definir las observaciones del agente para esta tarea fueron analizados diversos trabajos[31, 33, 47, 48], donde se pudieron observar dos grupos principalmente: Propiocepción del brazo e información del objetivo. Basándose en los trabajos mencionados se procede a utilizar las siguientes observaciones:

- **Propiocepción brazo**: Componentes que entreguen el estado actual del brazo robótico.
  - $\phi_{1,\dots,6}$: Ángulo actual de cada uno de las 6 articulaciones del brazo robótico UR5, de utilidad a la hora de detectar colisiones como también si el brazo se encuentra fuera de la zona de manipulación definida en la Sección 3.2.3.
  - $\dot{\phi}_{1,\dots,6}$: Velocidad actual de cada uno de las 6 articulaciones del brazo robótico UR5.
- **Objetivo**: Observaciones del estado actual del objetivo.
  - x_{arm}, y_{arm}, z_{arm}: Posición objetivo con respecto al efector final del brazo, define la distancia entre el efector y la posición objetivo.
  - x_{base}, y_{base}, z_{base}: Posición objetivo con respecto al eje principal del robot (definido en Sección 3.1).

Al concatenar todas las observaciones mencionadas se da lugar a un vector de dimensión 18.

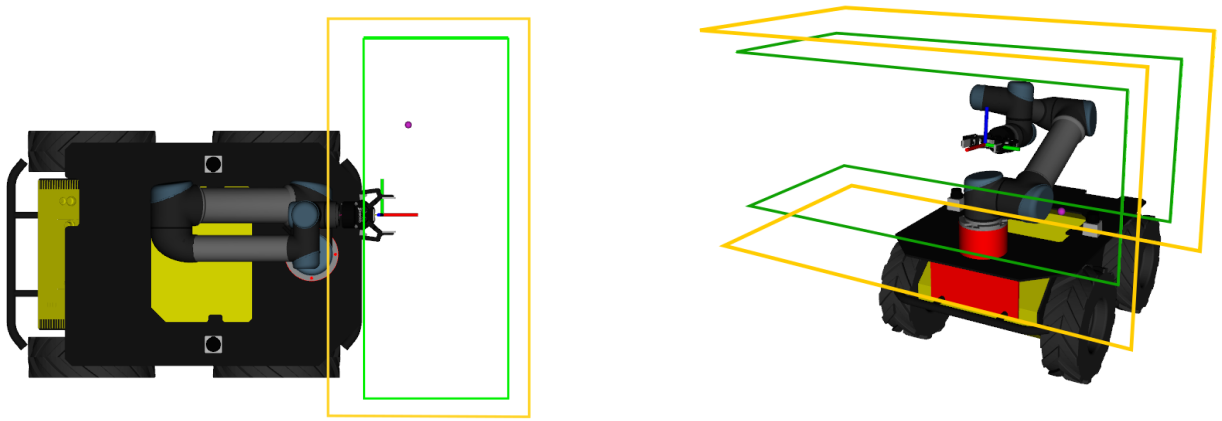
3.2.3. Ambiente de entrenamiento

El entrenamiento de esta tarea también es realizado en el simulador *Gazebo*, haciendo uso de las herramientas que entrega *ros_ control* [54], un conjunto de paquetes de software que permiten el control de diversos actuadores incluyendo las interfaces de control y interfaces de hardware. La ventaja que presenta utilizar esta aproximación son las similitudes existentes entre los comportamientos del brazo en simulación con la realidad, siendo esto último de vital importancia a la hora de transferir la política al mundo real.

Otro ámbito a considerar a la hora del aprendizaje de esta tarea, son diversas restricciones que se encuentran presentes en variadas formulaciones de la tarea de *Reaching*, cuando esta es formulada como un problema de aprendizaje reforzado. Este problema de restricciones no especificadas se encuentra detallado en el trabajo *How Much Do Unstated Problem Constraints Limit Deep Robotic Reinforcement Learning?* [31], donde se estudia en profundidad como restricciones espaciales afectan directamente al proceso de aprendizaje, en aquel trabajo se estudia directamente sobre las restricciones geométricas en las áreas efectivas del robot, utilizando un brazo robótico UR5 como plataforma de estudio. Motivados por los resultados expuestos en el trabajo introducido por Cannon Lewis el 2019 [31], es que se propone el uso de una zona de manipulación, junto también a la adición de una zona válida, como una restricción geométrica que ayude al proceso de aprendizaje.

La restricción geométrica propuesta para la tarea de *Reaching* de posición se justifica en el interés de que el robot sea capaz de manipular objetos frente a él, descartando el uso del brazo en la dirección trasera del robot. Está restricción geométrica es presentada en la

figura 3.4, donde en verde se ve la zona de manipulación, la cual restringe donde se pueden obtener posiciones objetivos. Por otro lado, en amarillo se presenta la zona válida, consistente en límites geométricos para el efector final, donde en caso de salir se da por finalizado el episodio. La zona válida permite ayudar al proceso de aprendizaje, ya que no se consideran estados que no sean de interés dado las restricciones propuestas. Otra de las ventajas que provee el definir una zona geométrica donde es válido el movimiento, es que al definir una altura mínima, esta permite que el brazo no colisione con el resto de la plataforma.



(a) Vista superior

(b) Vista isométrica

Figura 3.4: Vistas zonas de manipulación [verde] y zona válida [amarillo].

Para esta tarea también es definida una configuración del brazo considerada como reposo, esta configuración denominada *home* es presentada en la Figura 3.4. La configuración de reposo posee la característica de no extender el polígono exterior de la base, y también debido a que el brazo se encuentra recogido, se evita que todo el peso recaiga en alguna de las articulaciones. Los valores angulares para cada una de las articulaciones es presentado en la Ecuación 3.7, donde se especifican los ángulos en radianes.

$$\begin{aligned}&[\phi_{\text{elbow}}, \phi_{\text{shoulder\_lift}}, \phi_{\text{shoulder\_pan}}, \phi_{\text{wrist\_1}}, \phi_{\text{wrist\_2}}, \phi_{\text{wrist\_3}}] \\ &= [2,4201; -2,7052; -0,78539; 0,2071; 1,5707; 0,0]\end{aligned} \tag{3.7}$$

3.2.4. Función de recompensa

A la hora de definir la función de recompensa a utilizar durante esta tarea, se debe destacar que al tratarse de una tarea de precisión es requerida una componente que busque constantemente acercar el brazo al objetivo. Por otro lado, deben ser considerados los comportamientos no deseados en esta tarea, entre los cuales, se destacan: auto-colisiones, salida de zonas de manipulación, salida de zonas válidas, y por último comportamiento erráticos. Tomando estas consideraciones y basados igualmente en las funciones de recompensa utilizadas en los trabajos [31] y [48] , se define:

$$r^t = \begin{cases} r^t_{\text{reaching}} & \text{si } \rho^t_{\text{target}} \geq \rho_{\text{thresh}} \\ r^t_{\text{success}} & \text{si } \rho^t_{\text{target}} < \rho_{\text{thresh}} \\ r^t_{\text{collision}} & \text{si el agente colisiona} \\ r^t_{\text{outside\_valid}} & \text{si } P^t_{\text{eef\_base}} \text{ fuera de zona válida} \end{cases} \tag{3.8}$$

Donde $P^t_{\text{eef\_base}}$ es la posición en 3D (x,y,z) del efector final con respecto al eje principal del robot.

La función de recompensa propuesta mantiene la misma estructura general utilizada en la tarea de navegación local, donde dependiendo del estado actual se poseen diferentes funciones. Entre estas componentes se encuentran r^t_{reaching}, componente encargada de guiar al brazo a acercarse al objetivo. La segunda componente conocida como r^t_{success}, tiene como función entregar una recompensa positiva cuando se cumple la condición de éxito de un episodio, esto es cuando el efector final del brazo se acerca a la pose objetivo con una distancia igual o menor a ρ_{thresh}. Al igual que para el caso de navegación se le asigna una componente negativa conocida como $r^t_{\text{collision}}$, en caso de realizarse una colisión durante el episodio. Por último la componente $r^t_{\text{outside\_valid}}$ es la encargada de penalizar en caso de que el efector salga fuera de la zona válida definida en el ambiente de entrenamiento.

La componente r^t_{reaching} se define como:

$$r^t_{\text{reaching}} = r^t_{\text{distance}} + r^t_{\text{actions}} + r^t_{\text{outside}} \tag{3.9}$$

- r_{distance} : La componente de distancia es la encargada de penalizar al agente por estar lejano al objetivo, junto también a favorecer a medida que el agente se acerca. Posee una constante de offset $K_{\rho_{\text{arm}}}$, utilizada para mantener esta componente siempre negativa con valor máximo 0:

  $$r^t_{\text{distance}} = -\rho_{\text{arm\_t}} - \ln(\rho_{\text{arm\_t}}) - K_{\rho_{\text{arm}}} \tag{3.10}$$

  Donde es utilizada la función de ln, para obtener una componente de precisión donde sea premiada de mayor medida las distancias cercanas.

  $$K_{\rho_{\text{arm}}} = -\rho_{\text{thresh}} - \ln(\rho_{\text{thresh}}) \tag{3.11}$$

- r_{actions} : La segunda componente utilizada a lo largo de la trayectoria, consiste en una penalización a la magnitud de las acciones efectuadas por el agente, el objetivo es penalizar acciones de control de grandes magnitudes. Esta se calcula como la norma de las acciones al cuadrado, donde cada a_{i} representa la acción efectuada en cada una de las articulaciones:

  $$r^t_{\text{actions}} = -1 \cdot \|a_{\text{i}}\|^2 \tag{3.12}$$

- r_{outside} : Última componente utilizada durante la trayectoria, consiste en una recompensa negativa binaria, la cual penaliza que el agente se encuentre aun dentro de la zona válida pero fuera de la zona de manipulación:

  $$r^t_{\text{outside}} = -1 \cdot \mathbb{1}_{\{P^t_{\text{eef\_base}}: P^t_{\text{eef\_base}} \text{ fuera de la zona de manipulación}\}}(P^t_{\text{eef\_base}}) \tag{3.13}$$

Donde $P^{t}_{\text{eef\_base}}$ es la posición en 3D (x,y,z) del efector final con respecto al eje principal del robot.

3.3. Manipulación móvil

El problema de manipulación móvil tal y como es explicado en la sección 1.1.1, consiste en el control simultáneo de un brazo robótico con una base móvil, con la finalidad de manipular algún objeto deseado. En este caso de estudio se considera una simplificación del problema, removiendo la componente de manipular un objeto. Por lo que la tarea a resolver consiste en llevar al efector final del brazo o gripper a una posición deseada en un mapa.

Al mismo tiempo y motivados por el trabajo presentado por Duan et al. [50], donde se presenta una arquitectura de aprendizaje reforzado jerárquico, en la cual se aprenden sub-políticas que realizan maniobras espécificas de forma independiente para luego aprender una política maestra que toma la decisión de cual de las sub-tareas se encuentran activas en cada momento. Es que la tarea de manipulación móvil es modelada como el aprendizaje de un planificador de alto nivel, el cual haga uso de las políticas aprendidas en las dos sub-tareas anteriores.

3.3.1. Espacio de acciones

Al ser este problema formulado como el aprendizaje de un *manager* de alto nivel, es que las acciones disponibles por este agente consisten en cual sub-sistema ocupar. Estas acciones son modeladas como acciones discretas, donde se considera el uso de cada sub-sistema como bien el uso de ambos de forma simultánea, lo cual da como resultado las siguientes acciones posibles a realizar por el *manager*.

- **Navegación**: Acción que indica que el siguiente [paso] se realizará la acción de navegación entregada por el sub-sistema relacionado. Cabe mencionar que tanto por eficiencia energética como también seguridad, es que al agente al realizar esta acción, es enviada una señal de control al brazo con el fin de que vuelva a su pose inicial de reposo. Esta pose inicial de reposo se encuentra definida en la sección 3.2.3, donde por un lado no existe una gran carga en alguna articulación en particular, además de encontrarse el brazo en su totalidad dentro de los límites geométricos de la base, lo cual permite que al realizarse una acción de navegación, el sistema no colisiones debido a que el brazo no fue considerado a la hora del entrenamiento de las políticas de navegación.
- **Ambas**: Acción que indica que se utilizarán ambos sistemas de forma simultánea. Cabe destacar que al agregar esta acción se debe considerar que las tasas de control de las sub-tareas deben poseer multiplicidad entre ellas.
- **Reaching**: La última acción a considerar es la de hacer uso exclusivo del brazo.

3.3.2. Espacio de observaciones

A la hora del diseño de las observaciones para este caso de estudio, es que al tratarse de un manager, se buscan observaciones que entreguen un estado de alto nivel general del agente, considerando que las sub-tareas funcionan de forma correcta. Esto resulta en un espacio

dimensional más reducido con respecto a las observaciones utilizadas en las sub-tareas, lo cual simplifica el proceso de aprendizaje.

El espacio de observaciones propuesto es:

- ρ_{arm}: Distancia euclidiana del objetivo medida desde el efector final.
- $x_{\text{base}}, y_{\text{base}}$: Coordenadas 2D de la posición objetivo con respecto al eje de referencia de la base robótica. Esta observación permite al agente verificar si la base se encuentra alineada con el objetivo o no. Definiendo el estar alineado como que la proyección 2D del objetivo se encuentre dentro de los límites de la zona de manipulación definida en 3.2.3, ya que como es mencionado la sub-tarea de *Reaching* solo funciona dentro de ese rango.
- Diff: Distancia existente entre el objetivo a manipular y la observación limitada que posee la sub-política de reaching. Esta distancia es cero cuando la base se encuentra alineada y mayor a cero en cualquier otro instante dependiendo de la distancia hacia el objetivo medido desde la base.
- Energy: Debido a que las observaciones mencionadas no entregan al agente una noción de cual es el estado de movimiento actual, por lo que si alguna de las sub-políticas se encuentra oscilando en algún punto no se tiene una forma de detectar este comportamiento. Es por esto que se considera una observación de energía denominada *energy*, la cual consiste en la distancia absoluta recorrida en los últimos cinco steps de control, presentada en la Ecuación 3.14, donde se considera como valor máximo de energía que el efector se acerque a una velocidad de 0,5 m/s

$$\text{energy}^t = \max\{\left\|P^t_{\text{eef}} - P^{t-5}_{\text{eef}}\right\|^2, 1\} \tag{3.14}$$

  Donde P^t_{eef} consiste en la posición 3D (x,y,z) del efector final con respecto al mapa en el paso t.

3.3.3. Ambiente de entrenamiento

El ambiente donde es desarrollado el entrenamiento de esta tarea es utilizado el mismo ambiente físico que el utilizado detallado en la Sección 3.1.3, a diferencia del mencionado en la sub-tarea en este caso es considerado el robot completo en la simulación, simulando en conjunto el brazo con la base móvil. Cabe destacar que las zonas definidas en 3.2.3 también son utilizadas en esta tarea de aprendizaje, siendo esta zona referenciada al eje principal de la base móvil, presentado en la Sección 3.1.

3.3.4. Función de recompensa

La función de recompensa propuesta mantiene la misma estructura general utilizada en ambas sub-tareas, donde dependiendo del estado actual se poseen diferentes funciones. Entre estas competentes se encuentran r^t_{hl}, componente encargada de guiar al agente a acercarse al objetivo. La segunda componente conocida como r^t_{success}, tiene como función entregar una recompensa positiva cuando se cumple la condición de éxito de un episodio, esto es cuando el efector final del brazo se acerca a la posición objetivo con una distancia igual o menor a ρ_{thresh}, con el requerimiento de que la posición se encuentre completamente dentro del la zona

de manipulación. Al igual que para el caso de ambas sub-tareas se le asigna una componente negativa conocida como $r^t_{\text{collision}}$, en caso de realizarse una colisión durante el episodio. Por último, la componente $r^t_{\text{over\_object}}$ es la encargada de penalizar en caso de que la base se posicione sobre la posición objetivo, esto debido a que en la formulación actual el agente entra a un estado sin retorno, ya que la posición se encuentra fuera de la zona válida y la política de navegación ya no puede acercar más al agente.

$$r^t = \begin{cases} r^t_{\text{hl}} & \text{si } \rho^t_{\text{arm}} \geq \rho_{\text{thresh}} \\ r^t_{\text{success}} & \text{si } \rho^t_{\text{arm}} < \rho_{\text{thresh}} \\ r^t_{\text{collision}} & \text{si el agente colisiona} \\ r^t_{\text{over\_object}} & \text{si geometría de la base se encuentra sobre la posición objetivo} \end{cases} \tag{3.15}$$

Donde

$$r^t_{\text{hl}} = r^t_{\text{arm}} + r^t_{\text{reaching\_far}} + r^t_{\text{outside}} + r^t_{\text{energy}} \tag{3.16}$$

- r^t_{arm}: Recompensa por acercarse al la posición objetivo, se ocupa la misma función que r^t_{distance}:

$$r^t_{\text{arm}} = -\rho_{\text{arm\_t}} - \ln(\rho_{\text{arm\_t}}) - K_{\rho_{\text{arm\_t}}} \tag{3.17}$$

- $r^t_{\text{reaching\_far}}$: La segunda componente, tiene como función el penalizar cuando el agente decide utilizar la sub-política de *Reaching* cuando se encuentra lejano a la posición objetivo. Si bien se busca que el agente sea capaz de aprender este comportamiento, esta componente se adiciona buscando por un lado ayudar al aprendizaje y, por otro lado, el evitar que el agente aprenda a mover el brazo en estados donde la política de navegación se encuentre realizando un movimiento preciso a baja velocidad:

$$r^t_{\text{reaching\_far}} = -10 \cdot \mathbb{1}_{\{\rho^t_{\text{arm}}:\rho^t_{\text{arm}}>3{,}0\}}(\rho_{\text{arm}}) \cdot \mathbb{1}_{\{a^t:a^t!=\text{Navegación}\}}(a^t) \tag{3.18}$$

- r^t_{outside}: Componente que penaliza de forma constante el que el robot no se encuentre alineado con la posición objetivo.

$$r^t_{\text{outside}} = -1 \cdot \mathbb{1}_{\{x_{\text{base}},y_{\text{base}}:x_{\text{base}},y_{\text{base}}\text{fuera de la zona de manipulación}\}}(x_{\text{base}}, y_{\text{base}}) \tag{3.19}$$

- r^t_{energy}: La última componente de la recompensa hace uso de la observación de la energía (energy) presente en la Ecuación 3.14, donde se penaliza tener energías bajas ya que esto significa que alguno de los dos sub-políticas se encuentra oscilando en alguna configuración.

$$r^t_{energy} = -1 + \text{energy}^t \tag{3.20}$$

Capítulo 4

Entrenamiento y evaluación del sistema

4.1. Simulación plataforma robótica

Tal y como es mencionado en la Sección 2.1, el aprendizaje reforzado requiere de interacciones entre un agente y un ambiente durante el proceso de aprendizaje. Este proceso de aprendizaje requiere de una alta cantidad de interacciones distintas, por lo cual es un proceso inviable a realizar en una plataforma real, donde también la simulación permite poseer un ambiente controlado.

Uno de los principales problemas a la hora del entrenamiento en simulación, corresponde al *reality gap* existente, por lo que se busca que la simulación utilizada sea la más fidedigna posible. Siguiendo esta línea, en este trabajo son utilizadas las herramientas de simulación en *Gazebo* proveídas por los fabricantes tanto de la plataforma móvil Husky [55], como del brazo robótico UR5 [56].

Dentro de las otras herramientas utilizadas en simulación se destacan:

- **Simulación de sensores lasers**: Para la simulación de los sensores laser Hokuyo UTM-30LX-EW montados en la plaforma, son utilizados los plugins de sensores nativos de *Gazebo*, utilizando el plugin *GpuRaySensor*[1], al cual se le deben especificar los parámetros: cantidad de puntos, resolución angular, ángulo mínimo y máximo de visión, rango máximo, rango mínimo, resolución de distancia y por último el modelo de ruido utilizado en las mediciones de rango[2].
- **Detección de colisiones**: Otro de los tipos de sensores que se encuentran disponibles en gazebo es un sensor de contacto, estos se encuentran implementados en el plugin ContactSensor [3]. Para facilitar la detección de colisiones de todo el modelo, se agrega una plataforma solida en la parte superior del robot presentada en la Figura 4.1 donde en naranjo se observa la geometría de colisión, como también se indican las dimensiones de esta. Esta geometría de colisión es considerada como el polígono exterior del robot, y

[1]Referencia clase GpuRaySensor API Gazebo 9.

[2]Todos estos parámetros se encuentran en los datasheets del fabricante `https://www.hokuyo-aut.jp/search/single.php?serial=170`

[3]Referencia clase ContactSensor API Gazebo 9.

se diseño de tal forma que sea el primer objeto en colisionar, esto permite solamente se requiera un sensor de contacto para considerar las colisiones de la base móvil. Por otro lado, para la detección de colisiones del brazo robótico fueron utilizadas las geometrías proveídas de la simulación de este mismo.

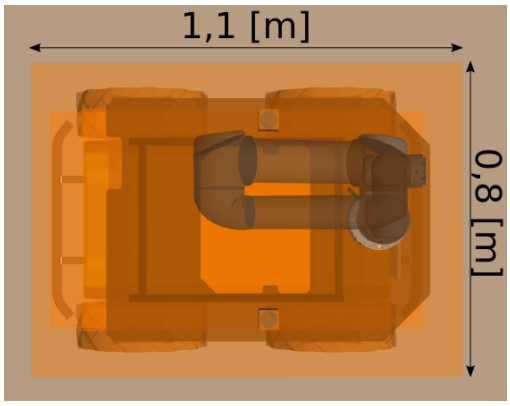


Figura 4.1: Sensor de colisiones utilizado en simulación.

- **Movimiento base ideal**: Una de las grandes dificultades a la hora de similar el movimiento de una plataforma del tipo skid-steer, es que sus rotaciones se basan en deslizamientos en sus ruedas, el problema es que estas rotaciones no consistentes a lo largo de la simulación, por lo que se decidió a utilizar un plugin de movimiento ideal el cual permite tener rotaciones consistentes y similares al robot real.
- **Localización ideal**: Debido a que los objetivos de manipulación móvil y los de navegación son muestreados en un mapa, es requerida una localización del robot en cada instante para poder obtener la posición del objetivo localmente, es por esto que fue utilizado el servicio de obtención del estado actual de un modelo simulado, (GetModelState)[4] ya que esto permite no considerar un sistema de localización que además de requerir recursos computacionales, puede inducir errores si su funcionamiento no es correcto durante el entrenamiento.

El entrenamiento de todas las tareas es realizado en el simulador *Gazebo* 9, haciendo uso del framework *ROS melodic* y de la librería de Deep learning *pytorch* 1.4.0. Todo corriendo de forma nativa en *Ubuntu* 18.04.

4.2. Navegación local

4.2.1. Condiciones episódicas de entrenamiento

Durante el entrenamiento de políticas es de suma importancia abarcar la mayor cantidad de situaciones a las cuales el robot se pueda enfrentar, para lograr esto es necesario contar con una gran variedad de objetivos de navegación. Con la finalidad de lograr una variedad en los objetivos de navegación es se utiliza el siguiente procedimiento:

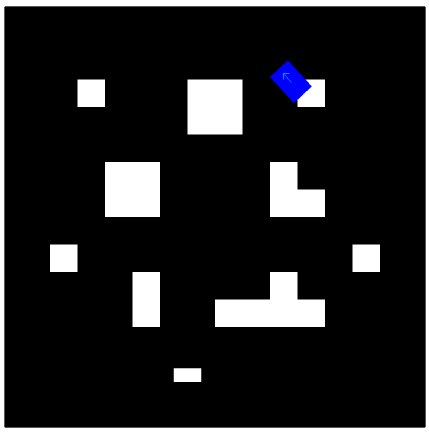
En primer lugar, se obtiene el polígono exterior de la plataforma de estudio, en este caso

[4]Referencia gazebo_msgs/GetModelState Service

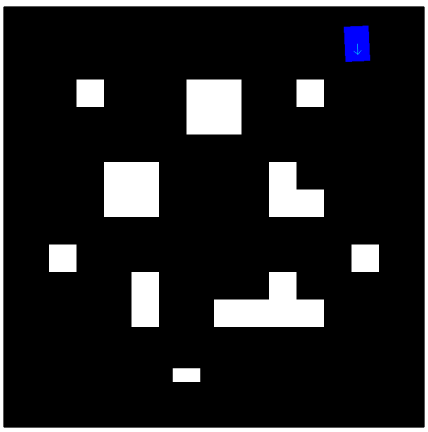
el polígono exterior considera el sensor de colisión utilizado en simulación (ver Sección 4.1), un rectángulo de las dimensiones 1,1 x 0,8 [m], donde se considera un margen extra de un 20 %. Junto también se debe contar con la grilla de ocupancia del ambiente en el cual se está desempeñando el robot.

Una vez obtenidos estos dos requisitos son muestreadas tuplas (x, y, θ), de posibles objetivos del robot dentro del ambiente. Cada una de estas propuestas debe pasar por dos filtros, los cuales buscan que los objetivos sean complejos y al mismo tiempo factibles.

- Distancia mínima: Con el objetivo de no generar objetivos muy simples en donde no sea necesario evadir obstáculos es que se requiere que la pose objetivo muestreada este al menos a 5 metros del origen, ya que esta distancia asegura que al menos deberá evadir alguno de los obstáculos del ambiente.
- Objetivo no colisionado: Otro de los requisitos que deben tener la selección aleatoria es que sean objetivos realistas, por lo que uno de los requisitos mínimos son que los objetivos sean libres de colisiones. Este requisito es verificado utilizando la grilla de ocupancia del ambiente de entrenamiento y las dimensiones exteriores del robot. Una visualización de ejemplos rechazados por colisiones y un objetivo sin colisión son presentados en la figura 4.2.



(a) Ejemplo objetivo rechazado por colisión.



(b) Ejemplo objetivo exitoso.
.

Figura 4.2: Selección aleatoria de objetivos de navegación.

Otro ámbito a considerar dentro de las condiciones episódicas son las condiciones de término de estos, estas condiciones denotan si un episodio fue exitoso o no, junto también a denotar en que momento se empieza un nuevo episodio. Las condiciones de término utilizadas en entrenamiento son las siguientes:

- Fuera de tiempo: Con el fin de evitar que un episodio se extienda por un período de tiempo muy elevado, ocasionando que los estados observados por el agente sean sesgados, es que se utiliza un valor máximo de pasos que el agente puede realizar, en caso que el agente se extienda la cantidad de pasos límites el episodio termina.
- Colisión: En caso de realizarse una colisión por parte del agente el episodio se da por terminado, junto también que el agente recibe una recompensa negativa $r^t_{\text{collision}}$ definida

en la Sección 3.1.4.

- Éxito: En caso de cumplirse el objetivo el episodio termina, en este caso se considera como exitoso un episodio cuando la distancia al objetivo (ρ^t_{target}) es menor que el umbral ρ_{thresh}. En caso de tenerse un episodio exitoso el agente recibe una recompensa positiva r^t_{success} definida en la Sección 3.1.4.

A la hora de iniciar un nuevo episodio, independiente de la razón por la cual el anterior episodio terminó, el robot es trasladado al centro del ambiente de entrenamiento siempre con la misma orientación, posteriormente es obtenido un objetivo tal y como es explicado en la presente Sección. Una vez realizado esto se da comienzo al nuevo episodio de entrenamiento.

4.2.2. Parametrización de políticas

Para el entrenamiento es utilizado el algoritmo DDPG [42], el cual utiliza dos redes neuronales independientes, las cuales son la parametrización de la política (*Actor*) y la función de acción-valor (*Critic*). En el caso de esta tarea es utilizado la arquitectura multi-modal PCL presentada en el trabajo [1], esta arquitectura se encuentra representada en la Figura 4.3, donde la red hace uso de las flechas azules y por el otro lado las flechas rojas indican que son componentes exclusivo de la red *Critic*. La ventaja que presenta el uso de una arquitectura multi-modal es el poseer distintos extractores de características para cada una de las fuentes de información.

Esta arquitectura multi-modal separa las tres observaciones del agente, por un lado para el caso de O_{pcl} es utilizado el extractor de características de la arquitectura *PointNet* [57] ya que esta arquitectura permite que la cantidad de puntos que alimentan la rama no sea fijo. Para las otras dos componentes de las observaciones, las cuales al ser siempre de tamaño fijo, se utiliza una arquitectura fully connected manteniendo sus características en ambos casos. Una vez obtenidas las representaciones intermedias estas son concatenadas y posteriormente alimentadas a las capas fully connected finales, para luego finalmente separar dependiendo de sí, se trata de la red *Actor* alimentar las dos capas independientes fully connected finales separando los comandos de velocidad debido a la diferencia en sus funciones de activación, por otro lado al tratarse de la red *Critic*, esa salida es alimentada a una capa fully connected y es utilizada una función de activación linear para obtener la estimación de la función de valor.

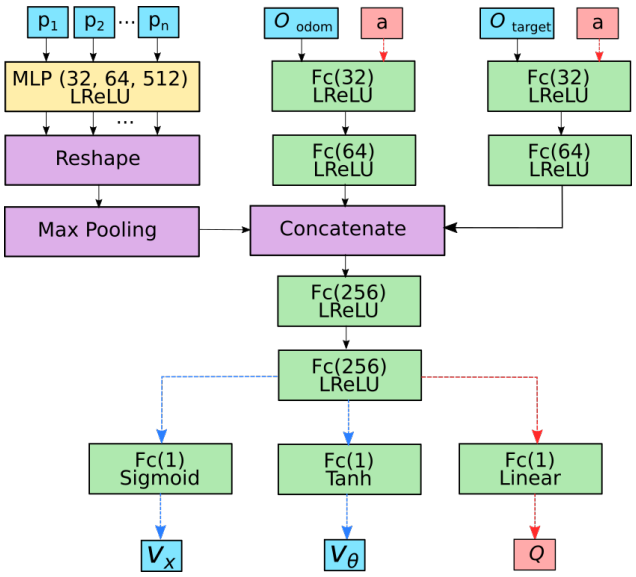


Figura 4.3: Parametrización de política, Navegación Local.
Fuente: Adaptada de [1].

Por último los hiperparámetros utilizados en el aprendizaje de esta tarea son presentados en la tabla 4.1.

	Parámetro	Valor
DDPG	Tasa de aprendizaje *Actor*, *Critic*	0,0001; 0,001
	red *Critic*	L2 10^{-2}
	Factor de descuento γ	0,99
	Tamaño de Batch	256
	Tamaño Replay Buffer	200000
	Factor de suavizado	0,001
Control	Frecuencia de control (hz)	5
	$v_x^{max}(m/s)$	0,5
	$v_\theta^{max}(rad/s)$	1,0
Precisión	$\rho_{\text{thresh}}(m)$	0,15

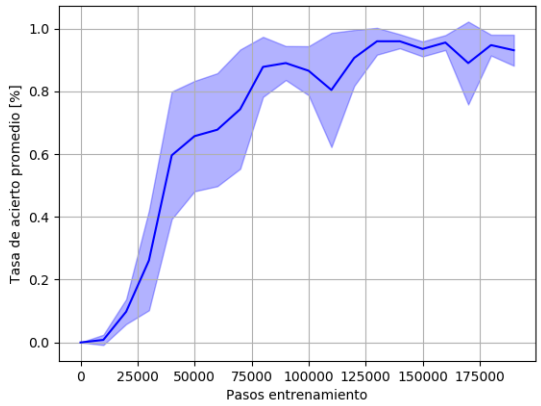
Tabla 4.1: Hiperparámetros navegación local.
Adaptados de [1].

4.2.3. Entrenamiento

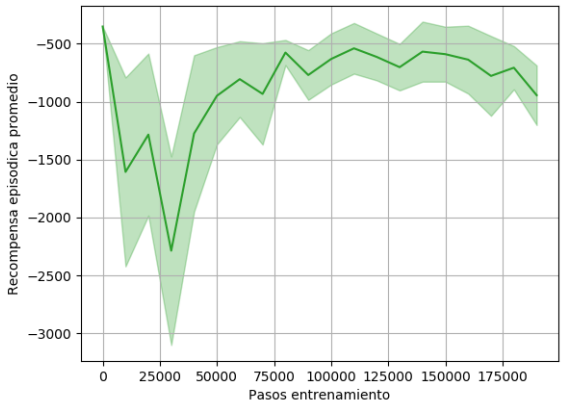
El entrenamiento de esta tarea es llevado a cabo en el simulador *Gazebo*, en el ambiente descrito en la Sección 3.1.3, para lograr una mayor eficiencia en simulación en el entrenamiento sólo es utilizada la base debido a los recursos que consume simular el brazo. El entrenamiento de cada modelo considera 200.000 [pasos], donde cada 10.000 [pasos] son guardados los pesos con el fin de ir evaluando el proceso de aprendizaje.

Con el objetivo de evaluar la robustez del proceso de aprendizaje es que son entrenadas cinco políticas independientes, las cuales una vez ya entrenadas son testeados 50 episodios

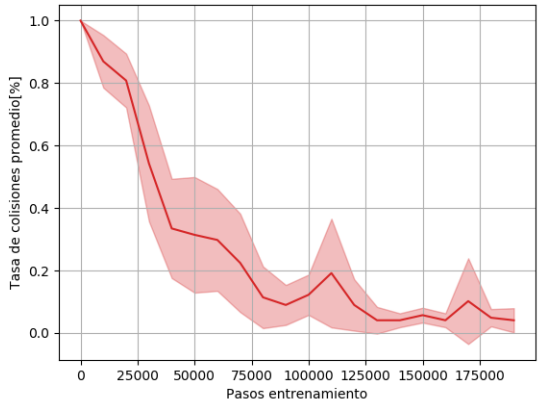
con los mismos objetivos a lo largo de las cinco políticas. Esta validación es realizada cada 10.000 pasos, y se extraen las tasas de acierto, tasas de colisiones, recompensas promedio y pasos promedios en terminar un episodio, todos estos valores son promediados a lo largo de las cinco políticas y presentados en la figura 4.4.



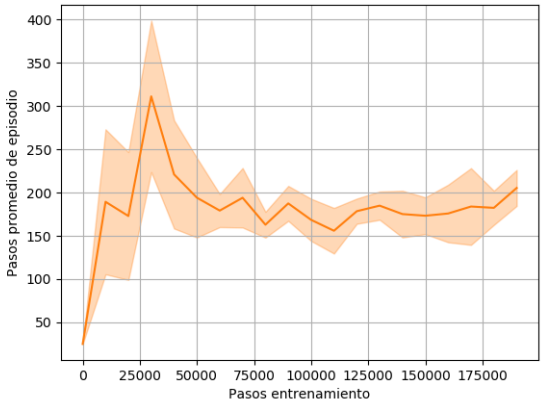
(a) Tasa de acierto.



(b) Recompensa episódica promedio.



(c) Tasa de colisiones.



(d) Pasos promedio.

Figura 4.4: Curvas aprendizaje tarea: navegación local.

Al analizar las curvas de aprendizaje se puede extraer lo siguiente: en primer lugar se obtuvo una convergencia estable a lo largo de las cinco instancias, esto representado por la tasa de acierto promedio cercana al 90 % observada en la Figura 4.4a, como también en las varianzas expuestas en las cuatro curvas. Por otro lado, al analizar la información entregada por la Figura 4.4d, se puede evidenciar que en la primeras etapas de entrenamiento el agente aprende a realizar evasión de obstáculos debido a que los episodios en promedio tienen una duración mayor, esto último es respaldado en la Figura 4.4c donde se ve un rápido decaimiento de los episodios terminados en colisiones.

Al comparar los resultados obtenidos y los presentados en el trabajo *Robust RL-Based Map-less Local Planning: Using 2D Point Clouds as Observations* [1], específicamente comparando la parametrización PCL entrenada en el ambiente SW2 debido a ser los utilizados en

este caso de estudio, se puede evidenciar que no se logran las mismas métricas presentadas. Principalmente al observar las tasas de acierto, se puede ver que no fue lograda la tendencia de aproximadamente 100 % lograda en el trabajo mencionado.

Si bien los resultados obtenidos en entrenamiento evidencian una correcta obtención de políticas de navegación local a lo largo de las cinco instancias independientes, estos resultados aún requieren de diversos ajustes de magnitudes en las componentes de recompensa o bien en los hiperparámetros utilizados.

4.3. *Reaching* de posición

4.3.1. Condiciones episódicas

Tal y como es mencionado en la Sección 4.2.1 es de suma importancia el abarcar la mayor cantidad de situaciones posibles, esto aplicado a la tarea de *Reaching* crea diferentes condiciones, entre éstas se encuentran que es deseable que la política aprenda a llegar a la posición objetivo desde múltiples direcciones, para lograr esto, al inicializar un episodio se verifican dos condiciones del episodio anterior: termino en colisión y termino fuera de zona válida. En caso de que se cumpla alguna de estas condiciones el agente comienza su siguiente episodio en una configuración estable conocida, denominada *home*. En caso de que no se encuentre en una colisión o fuera de la zona válida, el agente inicia su siguiente episodio en la última configuración alcanzada por el episodio anterior.

Por otro lado, para asegurar que los resultados obtenidos sean representativos a lo largo de toda la zona de manipulación, es que la metodología a la hora de muestrear objetivos es utilizar tres distribuciones uniformes a lo largo de los ejes x,y y z considerando los límites de la zona de manipulación (ver Sección 3.2.3).

Cabe mencionar que esta tarea a ser *Reaching* de posición, es una simplificación de la tarea de *Reaching*, en la tarea completa se busca que un efector final se alineé con una pose objetivo. Alinearse con una pose objetivo agrega la dificultad de que el efector debe llegar a la posición objetivo con una cierta configuración, esta restricción no es considerada en el desarrollo de este trabajo por lo que el efector puede llegar en cualquier orientación.

Al igual que en la tarea de navegación local deben ser consideradas las condiciones de término de los episodios. Las condiciones de término utilizadas en el entrenamiento de la tarea de *Reaching* de posición son las siguientes:

- Fuera de tiempo: Al igual que en la tarea de navegación para evitar que un episodio se extienda por un período de tiempo muy elevado, ocasionando que los estados observados por el agente sean sesgados, es que se utiliza un valor máximo de pasos que el agente puede realizar, en caso que el agente se extienda la cantidad de pasos límites, el episodio termina.
- Colisión: En caso de realizarse una auto-colisión de parte del agente, el episodio se da por finalizado, resultando en una recompensa negativa $r^t_{\text{collision}}$ definida en la Sección 3.2.4.
- Outside: Si el efector final del manipulador sale de la zona válida de manipulación

definida en la Sección 3.2.3, el episodio se da por finalizado y el agente recibe una recompensa negativa $r^t_{\text{outside\_valid}}$ definida en la Sección 3.2.4.

- Éxito: En caso de cumplirse el objetivo el episodio termina, en este caso se considera como exitoso un episodio cuando la distancia al objetivo (ρ^t_{target}) es menor que el umbral ρ_{thresh}. En caso de tenerse un episodio exitoso el agente recibe una recompensa positiva r^t_{success} definida en la Sección 3.1.4.

A diferencia de los detalles de entrenamiento de la tarea de navegación local presentados en la Sección 4.2.1, es que en esta tarea dependiendo de como termina un episodio se decide la posición inicial del siguiente episodio. En primer lugar, si un episodio termina en una colisión o fuera de la zona válida, el siguiente episodio comenzará en la posición de reposo del brazo definida en la Sección 3.2.3. Por otro lado, si un episodio termina debido a quedarse sin tiempo o bien resulta ser un episodio exitoso, el siguiente episodio comenzará en la última configuración del brazo, la motivación detrás de esto consiste en que la política sea capaz de alcanzar una posición objetivo desde todas las direcciones posibles, y no solamente desde la posición de reposo.

4.3.2. Parametrización de políticas

Igualmente al tratarse de una tarea con acciones continuas se propone utilizar DDPG como algoritmo de aprendizaje, por lo que son necesarias dos redes independientes denominadas *Actor* y *Critic*. La arquitectura propuesta para este caso de estudio se basa en la arquitectura de baja dimensionalidad presentada en el trabajo *Continous control with deep reinforcement learning* [42], donde solamente fue modificado el tamaño de las capas como también la función de activación.

La arquitectura utilizada es presentada en la figura 4.5, donde al igual que en el caso de navegación local (ver Sección 4.2.2) se utilizan flechas rojas para denotar las componentes exclusivas de la red *Critic* y en azul las componentes exclusivas de la red *Actor*. Posteriormente en la tabla 4.2 se presentan los hiperparámetros utilizados.

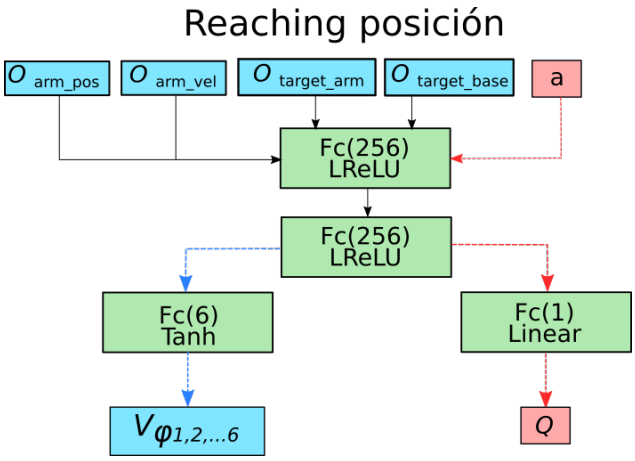


Figura 4.5: Parametrización de política, tarea: *Reaching* de posición

	Parámetro	Valor
DDPG	Tasa de aprendizaje *Actor*, *Critic*	0,0001; 0,001
	Regularizador red *Critic*	L2 10^{-2}
	Factor de descuento γ	0,99
	Tamaño de Batch	256
	Tamaño Replay Buffer	1.000.000
	Factor de suavizado	0,001
Control	Frecuencia de control (hz)	10
	$\mathrm{v}^{max}_{\phi_{1,\ldots,6}}(rad/s)$	0,5
Precisión	$\rho_{\text{thresh}}(m)$	0,02

Tabla 4.2: Hiperparámetros *Reaching* de posición.

4.3.3. Entrenamiento

Al igual que en el caso de navegación el entrenamiento es llevado a cabo en el simulador *Gazebo*, en este caso tomando el ambiente descrito en 3.2.3, haciendo uso exclusivo del brazo con el fin de reducir los costos computacionales de estar simulando la base. Cada iteración de entrenamiento considera 1.000.000 pasos, donde cada 20.000 pasos son guardados los pesos, para luego ser utilizados a la hora de obtener las curvas de entrenamiento.

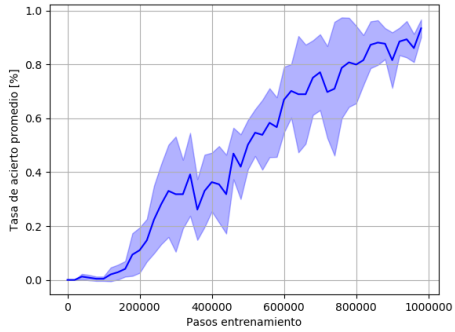
Manteniendo el mismo procedimiento utilizado en la tarea de navegación es que son entrenadas cinco políticas independientes, las cuales son validadas con 50 episodios en cada uno de los puntos guardados durante el entrenamiento. Este proceso de validación del proceso de aprendizaje entrega las curvas de aprendizaje presentadas en la figura 4.6.

Al analizar las curvas de aprendizaje se puede observar que el proceso de aprendizaje en esta tarea es realizado de forma exitosa, esto al observar las tasa de acierto alcanzando valores cercanos al 90 %. También al observar la curvas de recompensa promedio presentada en la Figura 4.6b es que a medida que se maximiza la recompensa el agente alcanza un mejor desempeño, lo cual indica un correcto diseño en este término.

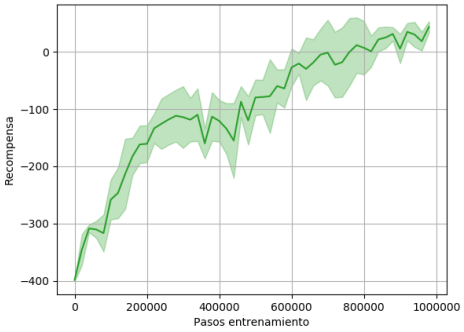
Al observar los comportamientos de las colisiones a lo largo del entrenamiento, presentados en la Figura 4.6c, se puede evidenciar que el comportamiento de evasión de auto colisiones es logrado en las primeras etapas de entrenamiento, considerando que desde el primer quinto del entrenamiento (paso 200.000), se observan menos de un 5 % de auto colisiones.

En los resultados expuestos en la Figura 4.6d, se puede evidenciar que al finalizar el entrenamiento se logra evitar que el efector final salga de la zona valida de manipulación. Otro comportamiento observado durante el entrenamiento, resulta al analizar la gran baja de pasos promedios en las primeras etapas de entrenamiento, presentado en la Figura 4.6e, se deben a que el agente en una primera etapa explota el comportamiento de salir rápidamente de la zona válida (subida en la Figura 4.6d), comportamiento rápidamente corregido según lo observado en ambas curvas.

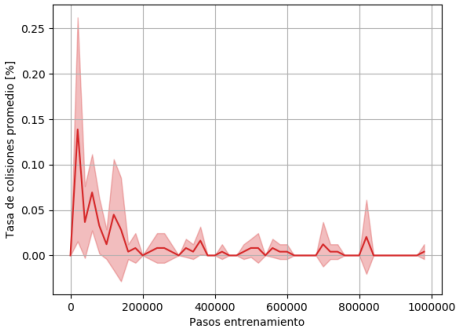
Finalmente al analizar los resultados del entrenamiento con respecto a los pasos promedios y la tasa de acierto, se observa que es logrado el comportamiento final deseado, siendo este comportamiento el de que el agente sea capaz de llevar al efector final a una posición objetivo,



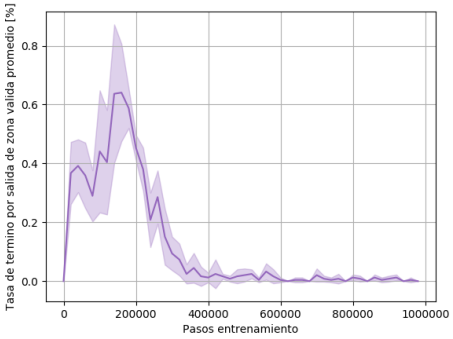
(a) Tasa de acierto.



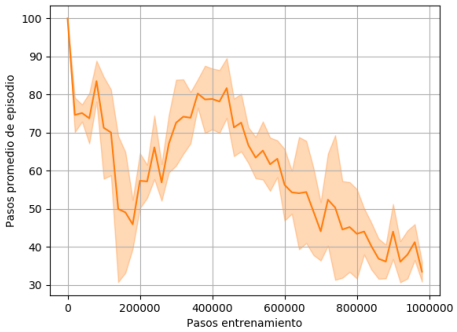
(b) Recompensa promedio.



(c) Tasa de Colisiones.



(d) Tasa de límite espacial.



(e) Pasos promedio.

Figura 4.6: Curvas de aprendizaje tarea: *Reaching* de posición.

como también en una cantidad de tiempo baja.

4.4. Manipulación móvil

4.4.1. Condiciones episódicas

En primer lugar el procedimiento de obtención de objetivos se basa en el utilizado en la metodología explicada en 4.2.1, con la diferencia que en este caso es considerada la zona de manipulación a la hora de tomar la geometría del robot. La metodología, en primer lugar, obtiene una tupla $(x_{\text{base}}, y_{\text{base}}, \theta_{\text{base}})$ de alguna posible pose del robot en el mapa, posteriormente es muestreada un objetivo de *Reaching* $(x_{\text{goal}}, y_{\text{goal}}, z_{\text{goal}})$ haciendo uso de la metodología explicada en 4.3.1, este objetivo se encuentra referenciado localmente con el robot. Finalmente, esta posición obtenida se traslada a la pose (x, y, θ)[5] dando como resultado una posición 3D objetivo en el mapa.

Al igual que en las tareas de navegación local y *Reaching* de posición deben ser consideradas las condiciones de término de los episodios. Las condiciones de término utilizadas en el entrenamiento de la tarea de manipulación móvil son las siguientes:

- Fuera de tiempo: Al igual que en la tareas de navegación y *Reaching* de posición para evitar que un episodio se extienda por un período de tiempo muy elevado, ocasionando que los estados observados por el agente sean sesgados, es que se utiliza un valor máximo de pasos que el agente puede realizar, en caso que el agente se extienda la cantidad de pasos límites el episodio termina.
- Colisión: En caso de realizarse una colisión de parte del agente, el episodio se da por finalizado, resultando en una recompensa negativa $r^t_{\text{collision}}$ definida en la Sección 3.3.4.
- Base sobre objetivo: Debido a la restricción que posee el sistema de alto nivel, es que existe un estado sin retorno. Este estado es alcanzado cuando la base se posiciona sobre el objetivo a manipular, en esta configuración por un lado la sub-política de *Reaching* no es capaz de alcanzar el objetivo ya que se encuentra fuera de su zona útil, mientras la sub-política de navegación busca seguir acercando la base. $r^t_{\text{over\_object}}$ definida en la Sección 3.2.4.
- Éxito: En caso de cumplirse el objetivo el episodio termina, en este caso se considera como exitoso un episodio cuando la distancia al objetivo (ρ^t_{target}) es menor que el umbral ρ_{thresh}. En caso de tenerse un episodio exitoso el agente recibe una recompensa positiva r^t_{success} definida en la Sección 3.1.4.

4.4.2. Parametrización de políticas

A diferencia de las dos tareas desarrolladas anteriormente, la formulación utilizada no hace uso de acciones continuas, si no más bien de acciones discretas, por lo que el algoritmo de aprendizaje utilizado es Deep-Q-Network [7]. La red neuronal a utilizar es la presentada en la Figura 4.7, está arquitectura normaliza y concatena todas las observaciones, para posteriormente alimentar dos capas fully connected, la representación obtenida alimenta la capa final de salida, está última capa posee como función de activación una de tipo linear y tres salidas, las cuales son las estimaciones de la función de valor del par estado-acción separadas según la acción, representadas en la Figura mediante $Q_{a\text{i}}$.

[5]El procedimiento consiste en trasladar la posición $(x_{\text{goal}}, y_{\text{goal}}, z_{\text{goal}})$ objetivo como si el robot se encontrara localizado en la pose $(x_{\text{base}}, y_{\text{base}}, \theta_{\text{base}})$.

La representación de la red presentada en la Figura 4.7, muestra en azul las diferentes observaciones, en verde presenta las capas de la red, en rojo se entregan las tres estimaciones de la función de valor para cada acción discreta, y por último mediante la representación de un switch se busca mostrar la idea que esta red escoge cual de las políticas será utilizada (Navegación local o *Reaching* de posición).

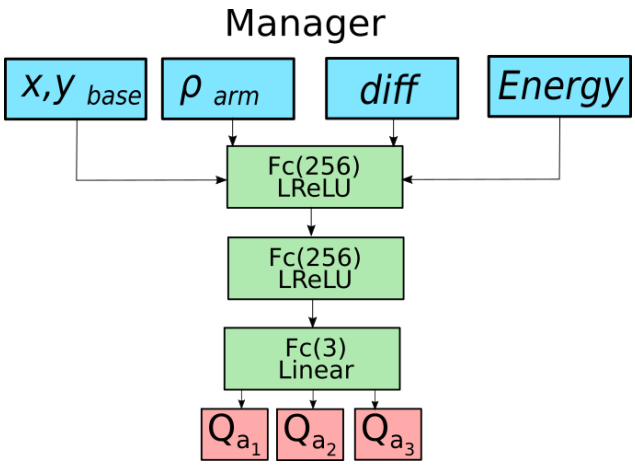


Figura 4.7: Parametrización de política manipulación móvil.

Los distintos hiperparámetros utilizados se presentan en la tabla 4.3, donde es importante mencionar que la frecuencia de control debe ser igual a la frecuencia de control menor entre las sub-políticas. Debido también a las diferencias en las frecuencias de control es que al realizarse una acción de *Reaching* son efectuados dos pasos, esto para mantener tanto la frecuencia de control de *Reaching* como del manager intactas. Esta restricción impuesta se debe a que se busca que ambas políticas funcionen de forma síncrona, además de permitir al manager poder cambiar entre ellas en el menor tiempo posible.

	Parámetro	Valor
DQN	Tasa de aprendizaje	0,001
	Factor de descuento gamma	0,99
	Tamaño de Batch	256
	Tamaño Replay Buffer	200.000
Control	Frecuencia de control (hz)	5
Precisión	$\rho_{\text{thresh}}(m)$	0,02

Tabla 4.3: Hiperparametros entrenamiento manipulación móvil.

4.4.3. Entrenamiento

Manteniendo la línea de trabajo en el aprendizaje de los dos sub-tareas, es que el entrenamiento del manager también es realizado en *Gazebo*, la diferencia es que al necesitar de todo el robot no se pueden remover componentes para acelerar la simulación. El entrenamiento es realizado en un total de 200.000 pasos, donde cada 10.000 pasos son guardados los pesos para su posterior validación.

Con el fin de estudiar la robustez en el proceso de aprendizaje de esta tarea es que cinco iteraciones independientes son entrenadas, cabe mencionar que en estas cinco iteraciones son utilizados los mismos pesos para las sub-políticas. Esto se realiza para solamente considerar al agente administrador como causante de un mejor o peor desempeño.

Una vez realizado el entrenamiento de las cinco iteraciones independientes, son validados 50 episodios a lo largo del entrenamiento, validando cada 10.000 pasos. Los resultados al promediar los cinco entrenamientos se presentan en la Figura 4.8.

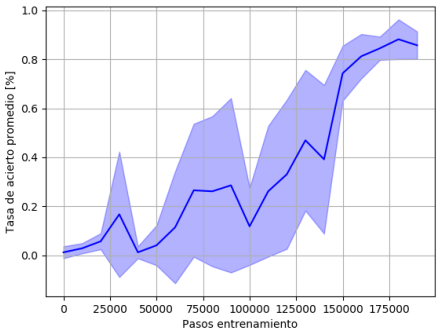
Si bien los gráficos no presentan tasas de éxito tan altas comparadas con las dos tareas estudiadas anteriormente, siguen manteniendo un valor alto lo que da a entender que el aprendizaje de esta tarea es realizado de forma correcta, pero bien aún hay espacio para realizar mejoras, ya sea, en la función de recompensa como en parámetros del entrenamiento.

Al analizar los resultados obtenidos, en primer lugar se puede observar un correcto diseño con respecto a la función de recompensa, ya que al observar las Figuras 4.8a y 4.8b se evidencia que la explotación de la función de recompensa (Valores más altos) van de la mano con una subida en la tasa de acierto del agente. Cabe destacar que esto no significa que la función de recompensa funcione a la perfección, pero si entrega una idea de que esta función es capaz de guiar al agente a un comportamiento similar al deseado.

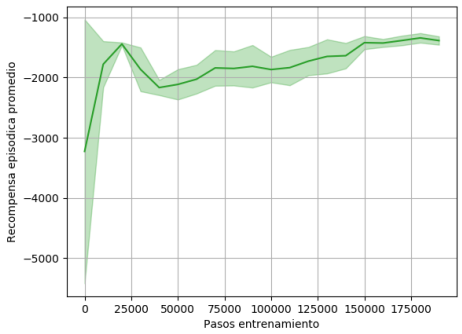
Otro ámbito interesante a analizar son los resultados expuestos en la Figura 4.8e, donde se ve que durante el entrenamiento no ocurrieron colisiones, esto es explicado principalmente por dos razones: la primera consiste en que al utilizarse la política ya entrenada de *Reaching* de posición, tal y como se observó en la Sección 4.3, al final del entrenamiento no existen auto colisiones de parte de los movimientos del brazo. La segunda razón consiste en al ser utilizadas las políticas entrenadas de navegación, junto al uso del mismo ambiente donde fueron entrenadas se suma que al considerar que el movimiento del brazo al interior de la zona de manipulación no agranda el polígono exterior en gran medida, por lo que si las políticas obtenidas de navegación considera un espacio suficiente no deberían ocurrir colisiones.

Por último, al observar los comportamientos expuestos en las Figuras y , se puede evidenciar que el agente aprende rápidamente a no "sobre navegar"[6], pero al observar los pasos promedios se extrae que la baja en la tasa de acierto se debe a que el comportamiento de alinearse aún no es aprendido.

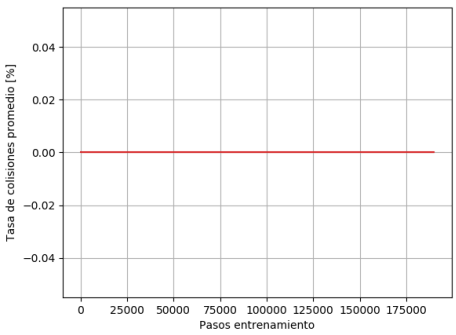
[6]Debido al estado de no retorno, si el agente se acerca demasiado al objetivo el episodio se da por finalizado.



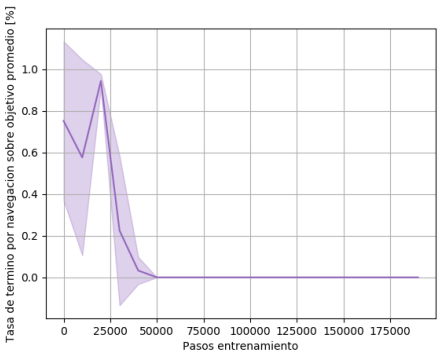
(a) Tasa de acierto.



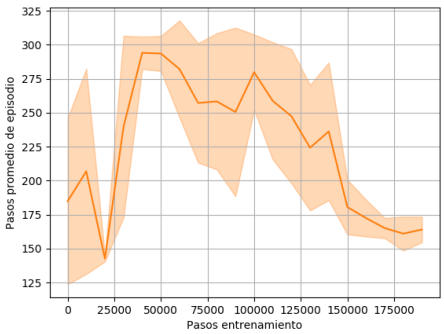
(b) Recompensa episódica promedio.



(c) Tasa de colisiones.



(d) Tasa de termino por base sobre objetivo.



(e) Pasos promedios.

Figura 4.8: Curvas aprendizaje tarea: Manipulación móvil.

Capítulo 5

Validación de políticas

Una de las tareas más importantes a la hora del desarrollo de políticas mediante aprendizaje reforzado, consiste en la validación de estas mismas. Esta validación permite obtener métricas de desempeño del comportamiento esperado, por un lado buenas métricas obtenidas en simulación permiten el continuar con la validación en el mundo real. Desde otra perspectiva las posteriores validaciones en el mundo real entregan una referencia de si fue posible una buena transferencia de la política. El presente capítulo se encuentra dividido en dos secciones, separando de esta forma las validaciones en simulación con las efectuadas en el mundo real.

La Sección 5.1 presenta las métricas de desempeño obtenidas, al evaluar políticas entrenadas en simulación. Estas métricas son separadas según la tarea desarrollada, junto también a ser calculadas promediando las tres instancias independientes obtenidas en cada una de ellas.

Posteriormente la Sección 5.2 son detallados los resultados de la validación en el mundo real, donde en primer lugar son presentadas las características del ambiente de validación utilizado. Luego son presentados los detalles de la validación física de navegación junto a sus resultados. A continuación se presentan los resultados y detalles de la validación de las políticas de *Reaching* de posición. Luego se entregan las trayectorias obtenidas a la hora de validar las políticas del administrador (manager). Por último, son presentados algunos análisis obtenidos al simular un ambiente que hace uso de dimensiones similares al ambiente utilizado en el mundo real.

5.1. Validación de políticas en simulación

Con el fin de poseer una métrica más representativa de los comportamientos aprendidos, se procede a evaluar 500 episodios para cada uno de los modelos entrenados según cada tarea. Es importante destacar que los mismos 500 objetivos son utilizados para cada uno de los modelos, esto para descartar que alguno de los modelos sea evaluado con ejemplos más "simples".

Los ambientes utilizados para la validación de las tareas de navegación local y manipula-

ción móvil se componen por el ambiente donde fue realizado el entrenamiento (ver Sección 3.1.3), junto a un nuevo ambiente de validación presentado en la Figura 5.1. El ambiente de validación utilizado mantiene las mismas dimensiones de 16 x 16 m, pero cuenta con distintos obstáculos fijos.

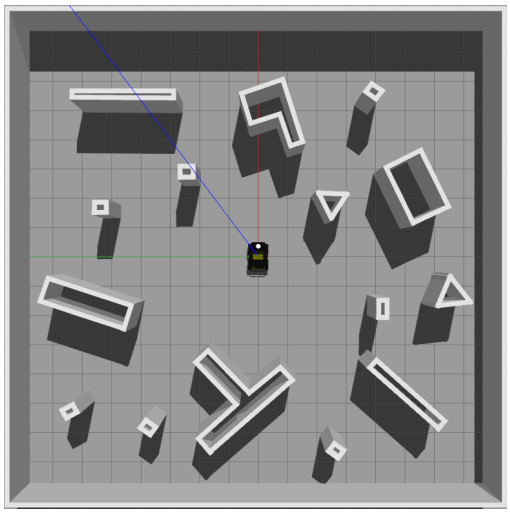


Figura 5.1: Ambiente validación en simulación.

Una vez evaluados todos las políticas obtenidas, se obtienen las tasas de acierto, pasos promedios y tasas promedio de término de episodios dependiendo de la tarea para cada una de las tareas estudiadas. Estos promedios son presentados en las Tablas 5.1, 5.3 y 5.4.

Concepto	Ambiente entrenamiento	Ambiente validación
Tasa de acierto	0,984 $\pm 0,011$	0,806 $\pm 0,089$
Tasa de *Timeout*	0,002 $\pm 0,003$	0,041 $\pm 0,027$
Tasa de colisiones	0,012 $\pm 0,007$	0,152 $\pm 0,078$
Pasos promedios	183,459 $\pm 29,260$	211,964 $\pm 35,572$

Tabla 5.1: Evaluación políticas en simulación tarea navegación local.

En primer lugar, al analizar los resultados de la validación de la tarea de navegación local presentados en la Tabla 5.1, se puede evidenciar que al validar en el ambiente donde fue realizado el entrenamiento se obtiene un buen desempeño, el cual se ve notoriamente reducido al modificar el ambiente. La baja en las métricas de desempeño pueden ser explicadas por las diferencias presentes entre los ambientes, donde se destacan las formas de los obstáculos como también las distancias entre ellos, donde principalmente se produjeron una mayor cantidad de colisiones.

Posteriormente, una vez obtenidas las métricas de las 5 políticas, es escogido el mejor resultado en el ambiente de entrenamiento para ser el utilizado tanto en la validación en el mundo real, como también el utilizado durante el entrenamiento de la tarea de manipulación móvil. Las métricas de la mejor política en el ambiente de entrenamiento es presentado en la Tabla 5.2, donde también es incluido las métricas de aquella política en particular en el ambiente de validación.

Concepto	Ambiente entrenamiento	Ambiente validación
Tasa de acierto	0,996	0,964
Tasa de *Timeout*	0,0	0,024
Tasa de colisiones	0,004	0,015
Pasos promedios	147,5	166,894

Tabla 5.2: Evaluación mejor política en simulación navegación local.

Concepto	Promedio	Mejor
Tasa de acierto	0,898 $\pm 0,093$	0,974
Tasa de *Timeout*	0,0973 $\pm 0,094$	0,02
Tasa de colisiones	0,0 $\pm 0,0$	0,0
Tasa de límite espacial	0,004 $\pm 0,006$	0,006
Pasos promedios	37,092 $\pm$10,688	28,254

Tabla 5.3: Evaluación políticas en simulación tarea *Reaching* de posición.

Los resultados obtenidos al validar las políticas de *Reaching* de posición presentados en la Tabla 5.3, muestran que a lo largo de las cinco políticas obtenidas, la presencia de colisiones y salida del efector de la zona válida son prácticamente inexistentes. En los resultados también son expuestos los resultados de la mejor política obtenida, ya que es esta la que será validada en el mundo real.

Concepto	Ambiente entrenamiento	Ambiente validación
Tasa de acierto	0,908 $\pm 0,050$	0,836 $\pm 0,056$
Tasa de *Timeout*	0,092 $\pm 0,050$	0,158 $\pm 0,056$
Tasa de colisiones	0,000 $\pm 0,001$	0,007 $\pm 0,002$
Tasa de navegación sobre posición objetivo	0,00 $\pm 0,00$	0,00 $\pm 0,00$
Pasos promedios	155,887 $\pm 7,873$	162,437$\pm 10,482$

Tabla 5.4: Evaluación políticas en simulación tarea manipulación móvil.

Por último, los resultados obtenidos al validar las políticas de manipulación móvil presentados en la Tabla 5.4, evidencian un comportamiento estable a lo largo de las cinco políticas en el ambiente de entrenamiento, a la hora de ser evaluados en el ambiente de validación se puede evidenciar una baja similar a la reportada en la tarea de navegación local.

5.2. Validación de políticas en el mundo real

Con la finalidad tanto de obtener una estimación del comportamiento de las políticas en un ambiente real, como también el validar si las políticas entrenadas son transferibles en el mundo real, se realizan validaciones para cada una de las políticas. Solamente para el caso de *Reaching* se entregan resultados cuantitativos, esto debido a lo controlado que es el ambiente experimental en esta tarea, para las demás tareas son presentadas curvas de comportamiento las cuales serán analizadas cualitativamente.

5.2.1. Características del ambiente de validación

Para el caso de las tareas de navegación local y manipulación móvil la validación es realizada en un ambiente controlado con presencia de obstáculos fijos. El ambiente posee las dimensiones de 8x5[m]. Cabe destacar que a la hora de evaluar las políticas en el ambiente real, no se cuenta con un sistema de localización externo ideal como el utilizado en simulación. Es por lo anterior que la posición del robot con respecto al ambiente es obtenida mediante un sistema de localización. Siendo más especificos es utilizada la implementación de AMCL [58] disponible en las herramientas de navegación proveídas por ROS [59]. Esta herramienta hace uso tanto de la odometría del robot, como de la información entregada por los lasers y las diferencias que estás tienen con un mapa previamente realizado. El mapa utilizado se encuentra en la Figura 5.2, donde se pueden observar los obstáculos fijos y las dimensiones del ambiente real. El mapa utilizado fue generado haciendo uso de la herramienta de *gmaping* [60] también provista en las herramientas de ROS.

Por otro lado, a la hora de evaluar la tarea de *Reaching* de posición, los requisitos de un ambiente varían, ya que, al estar esta tarea definida sin obstáculos que el agente debe esquivar. Es por esta razón que el único requisito que posee el ambiente donde se validará las políticas de *Reaching* es poseer suficiente espacio libre, para lograr que el brazo pueda moverse libremente.

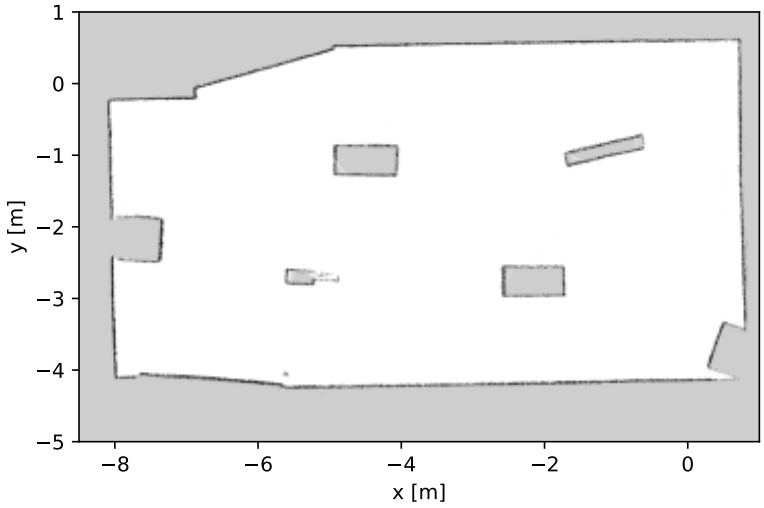


Figura 5.2: Mapa obtenido del ambiente de validación tareas navegación local y manipulación móvil.

5.2.2. Navegación

Para la validación de las políticas de navegación en el mundo real, es utilizado el ambiente mencionado en 5.2.1, en el cual el agente es testeado por 10 episodios. En cada uno de estos episodios son guardadas las posiciones de cada paso realizada por la política, con la finalidad de obtener las trayectorias realizadas.

Las trayectorias obtenidas son presentadas en la figura 5.3, donde cada color indica un

episodio distinto y el numero indica el extremo donde se encontraba el objetivo. Cabe mencionar que de estas 10 trayectorias la trayectoria 7 y 8 fallaron. Por un lado la trayectoria 8 termino bajo la condición de *timeout*, esto en parte debido a que el robot se deslocalizo cercano al objetivo, resultando en los comportamientos erráticos que se pueden evidenciar en la trayectoria. Por otro lado la trayectoria 7, en el último segmento el agente colisionó con una de los obstáculos montados en el ambiente.

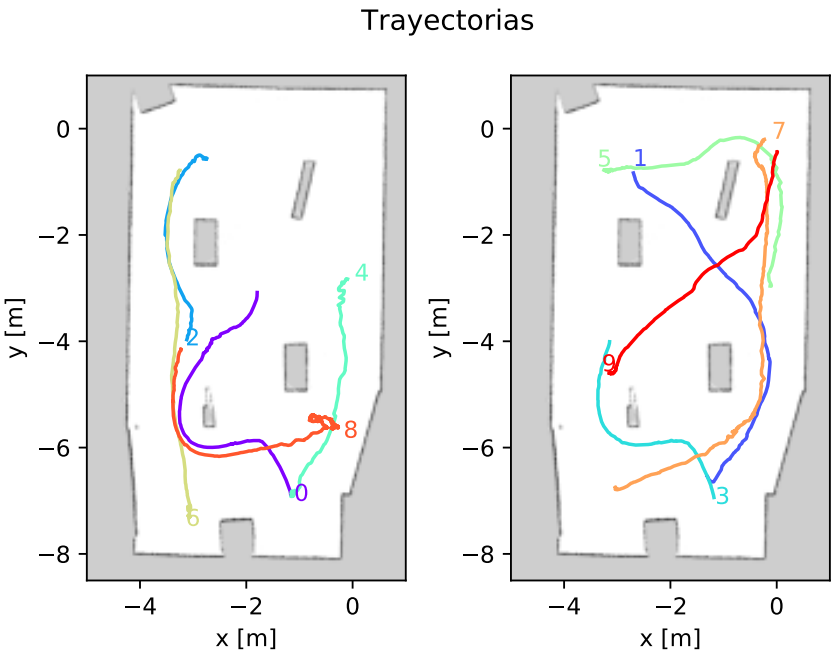


Figura 5.3: Trayectorias mundo real tarea navegación local.

Si bien los resultados obtenidos no permiten realizar un análisis cuantitativo, es posible evidenciar una correcta transferencia de la política al mundo real, esto al visualizar las trayectorias efectuadas por el agente, siendo este capaz de navegar dentro del ambiente.

5.2.3. *Reaching* de posición

Debido a como esta formulada esta tarea, y al también asumir que las medidas del estado actual del brazo entregada por el sistema de control del brazo robótico UR-5 son lo suficientemente precisas, a la hora de evaluar estas políticas se puede utilizar la misma metodología utilizada en simulación.

Cabe destacar que para evitar auto-colisiones causadas por el movimiento del brazo, debido a lo riesgoso que es que el brazo colisione consigo mismo. Es que fue implementada una capa de seguridad adicional, la cual se basa en utilizar los ángulos actuales del brazo y predecir su siguiente configuración. Esto mediante una extrapolación lineal utilizando el comando de velocidad actual, se consideran 10 pasos de control (1 [s]) a la hora de generar la predicción. En caso de que la predicción obtenida resulte ser una auto-colisión el comando de velocidad no es ejecutado y se contabiliza como una colisión.

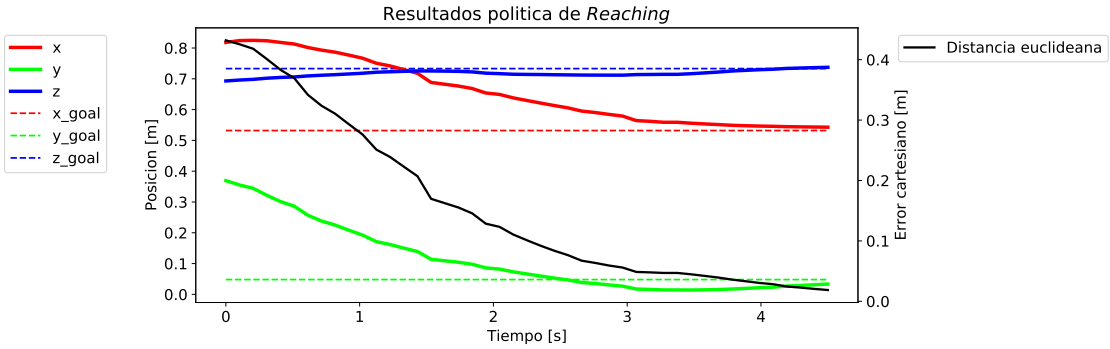
Para la evaluación es utilizada la política entrenada que obtuvo las mejores métricas al ser evaluado en simulación. Esta política es evaluada un total de 100 episodios, donde las métricas

promediadas se encuentran presentadas en la tabla 5.5. Estos resultados al ser comparada con los resultados presentados en 5.3, se pueden observar resultados muy similares, a excepción del mejor desempeño observado en la tasa de acierto, lo cual es explicado debido a que se evaluó la mejor de las cinco políticas obtenidas.

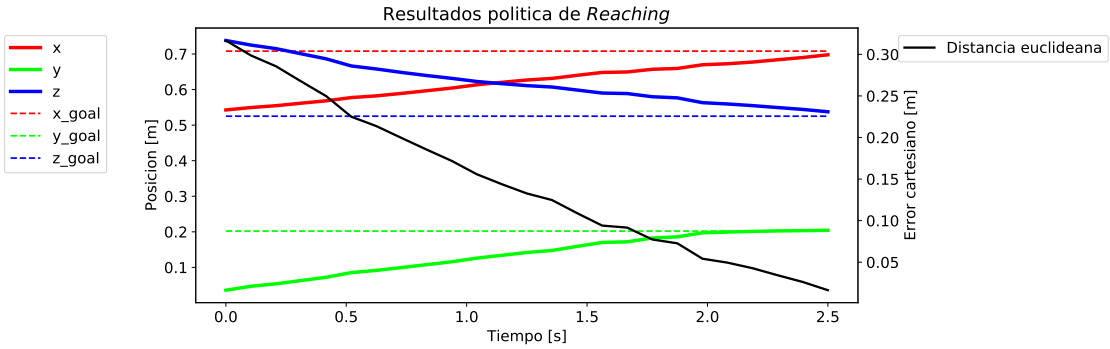
Concepto	Valor
Tasa de acierto	0,970
Tasa de *Timeout*	0,03
Tasa de colisiones	0,0
Tasa de limite espacial	0,00
Steps promedios	41,805

Tabla 5.5: Evaluación políticas en plataforma real tarea *Reaching* de posición.

Junto también a las métricas obtenidas se obtuvieron también los comportamientos del brazo a lo largo de cada episodio, estos son presentados en las figuras 5.4 y 5.5. En estas figuras se presenta el comportamiento de las 3 componentes de la posición del efector final del brazo, medida con respecto al eje principal del robot a lo largo de cada episodio. Se presentan con líneas punteadas los objetivos de aquellos episodios y en negro una curva que presenta el error a lo largo de la trayectoria.



(a) Ejemplo trayectoria 1.



(b) Ejemplo trayectoria 2.

Figura 5.4: Trayectorias de efector final tarea *Reaching* de posición.

Es importante destacar que para el caso presentado en la figura 5.5 , se presenta un ejemplo de un episodio fallido por que el agente se quedo sin tiempo. En este ejemplo se puede ver

que si bien el agente no fue capaz de llegar con la precisión necesaria a la posición objetivo logra mantener una posición relativamente cercana.

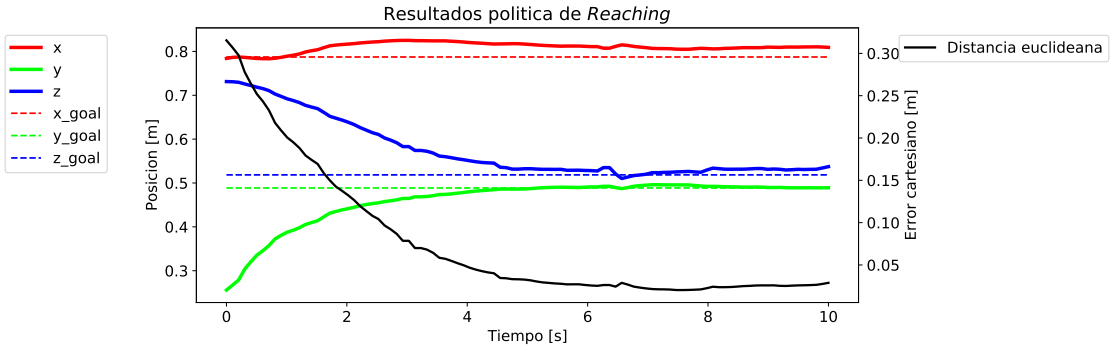


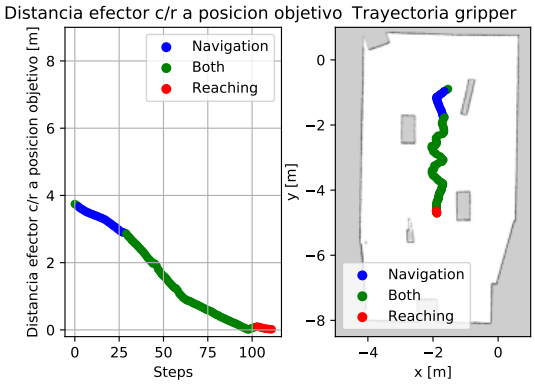
Figura 5.5: Ejemplo *timeout*, tarea: *Reaching* de posición.

Por último, al comparar los resultados obtenidos en simulación con los obtenidos en el mundo real, se verifica que el uso de las herramientas de simulación de *Gazebo* en conjunto a las herramientas oficiales de simulación del brazo, no difieren en gran medida con la realidad, logrando que la transferencia de la política sea prácticamente directa.

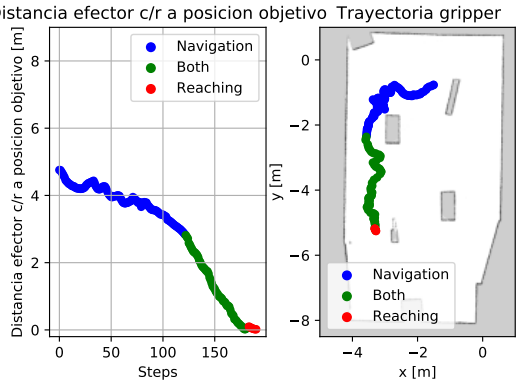
5.2.4. Manipulación Móvil

Por último son evaluadas las políticas de manipulación móvil en el mundo real, esto se lleva a cabo en el ambiente descrito en 5.2.1. A la hora de obtener objetivos son utilizadas dos procedimientos distintos, el primero consiste en simplemente entregarle un posición 2D (x,y) en el mapa y el agente obtiene alguna altura aleatoria dentro de los rangos de la zona de manipulación, esto resulta en posición 3D (x,y,z) objetivo en el mapa. El segundo procedimiento consiste en posicionar un poste en el ambiente, se obtiene la posición de este en el mapa (x,y) y es utilizada una altura constante a lo largo de los intentos resultando en una posición (x,y,z) fija, dicha posición es utilizada como objetivo. Este último permite probar varios intentos con la misma posición objetivo, como los presentados en las figuras 5.6c y 5.6d

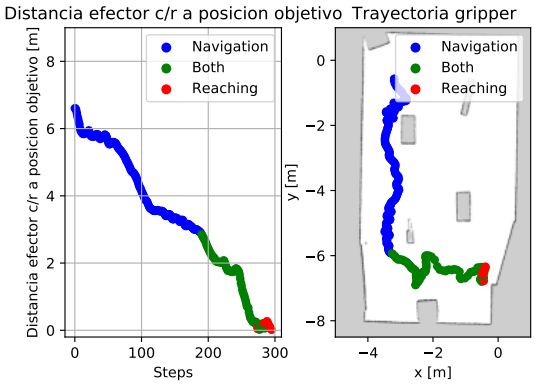
Las trayectorias obtenidas son separadas por episodio antes de ser presentadas en la Figura 5.6. A la izquierda se muestran la evolución de la distancia del efector final con la posición objetivo, como también de las acciones escogidas. Junto a esta visualización se encuentran también las trayectorias recorridas por el efector final a lo largo de los episodios.



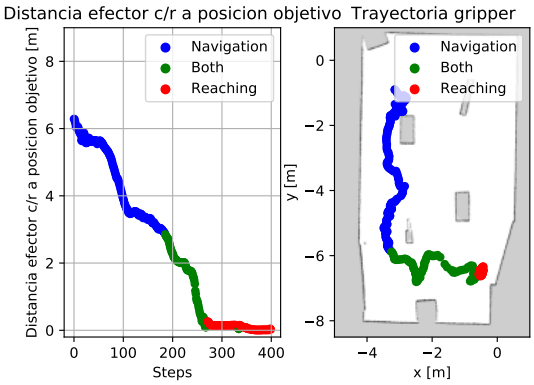
(a) Caso A.



(b) Caso B.



(c) Caso C.



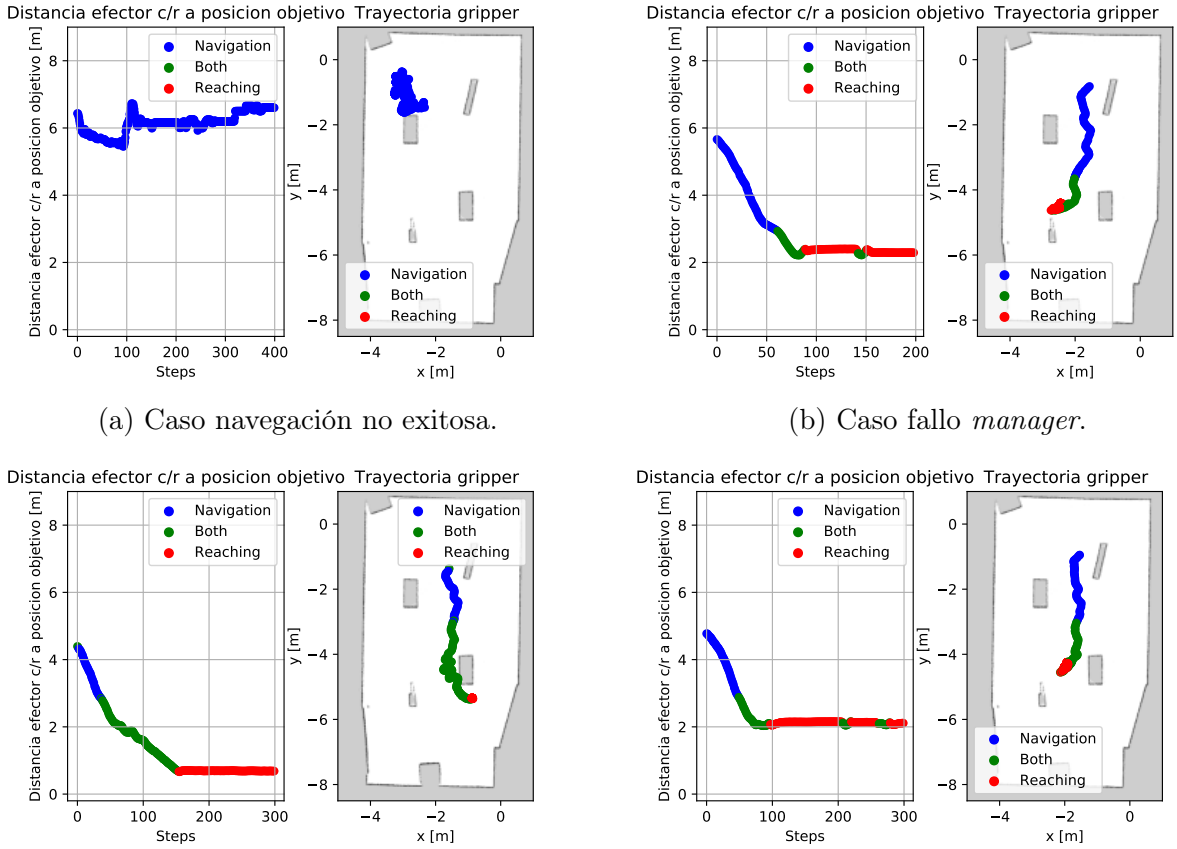
(d) Caso D.

Figura 5.6: Trayectorias de efector final tarea manipulación móvil, casos exitosos.

Al analizar los resultados obtenidos, se puede observar que, el *manager* es capaz de guiar al efector final a una posición objetivo. Pero cabe destacar que el principal problema que presentan las políticas obtenidas es en la transición de una acción de navegación a una acción de ambos sistemas ya que esta transición se realiza siempre cercano a los 3 mt. Esto podría ser explicado debido a que la componente $r_{reaching\_far}$ presente en la recompensa descrita en 3.3.4, es la única que entrega una desventaja al utilizar el brazo a largas distancias. Si bien esta componente logra en parte el comportamiento deseado, posee el problema de no adaptarse y simplemente aprender una condición con una distancia fija.

Por otro lado también se presentan distintos ejemplos de trayectorias obtenida donde el sistema falla. Estos resultados son presentados en la figura 5.7. Dentro de las fallas observadas se lograron evidenciar dos comportamientos. Por un lado se tiene el caso donde la navegación falla, presentado en la Figura 5.7a, donde se puede ver que si bien el agente *manager* decide utilizar la sub-política de navegación este se encuentra atascado en el estado actual, no siendo capaz de sortear el primer obstáculo. Por el otro lado se tienen representadas en las demás trayectorias problemas con el *manager*, donde en la figura 5.7b se puede observar que el agente se encuentra haciendo uso exclusivo de las sub-política de *Reaching*. Este problema se cree es debido a un problema de exploración y muestreo de objetivos durante el aprendizaje de esta tarea, ya que este comportamiento no es observado a la hora de validar en simulación

debido a que es utilizada la misma metodología a la hora de obtener objetivos.



(a) Caso navegación no exitosa.

(b) Caso fallo *manager*.

(c) Caso fallo *manager*.

(d) Caso fallo *manager*.

Figura 5.7: Trayectorias y distancia del efector final tarea manipulación móvil, casos fallidos.

Una posible solución para el problema de exploración encontrado, consiste en remover una de las condiciones utilizadas en la metodología de obtención de objetivos de navegación (ver Sección 3.1.3), consistente en la distancia mínima de 5 metros con respecto al centro del ambiente, al remover esta condición se asegura que el agente si observe casos donde los objetivos se encuentren a sus espaldas (considerando el eje x del robot) y a distancias cercanas. Es importante destacar que debido a que el *manager* inicia con una política ya funcional de navegación local, siempre que un objetivo se encuentre a su espalda el agente primero se orientara a el [1] y luego se acercará, esto ultimo genera que nunca durante el entrenamiento el agente observe objetivos cercanos a su espalda.

Al igual que en las tareas de navegación local y *Reaching* de posición, es posible evidenciar una correcta transferencia de las políticas entrenadas en simulación al mundo real. Pero cabe destacar que para un correcto desempeño de esta política es necesario añadir un sistema de detección de objetos, con la finalidad de poder verificar el correcto desempeño a la hora de acercarse a un objeto físico.

[1]Este comportamiento se debe a la componente r_{fov} de la función de recompensa utilizada en la tarea de manipulación móvil (Ver Sección 3.1.4).

5.2.5. Simulación ambiente real

Debido a las diferencias entre las dimensiones de los ambientes utilizados en simulación (16x16 [m]), con respecto a las dimensiones del ambiente real (8x5 [m]), es que se procedió a utilizar el mapa del ambiente real presentado en la Figura 5.2 para crear un nuevo ambiente en simulación que fuera lo más similar al real. El ambiente simulado se presenta en la Figura 5.8, donde también es incluido el robot a modo de referencia de las dimensiones.

En este nuevo ambiente de simulación se vuelve a realizar la validación de 500 episodios para cada una de las políticas obtenidas tanto para la tarea de navegación local, como la tarea de manipulación móvil. Esta validación se realiza utilizando un sistema de localización ideal, haciendo uso de las herramientas de simulación, esto es realizado con el fin de solamente observar el comportamiento de las políticas en un espacio reducido. Una vez realizada esta validación las métricas son promediadas y presentadas en las Tablas 5.6 y 5.7.

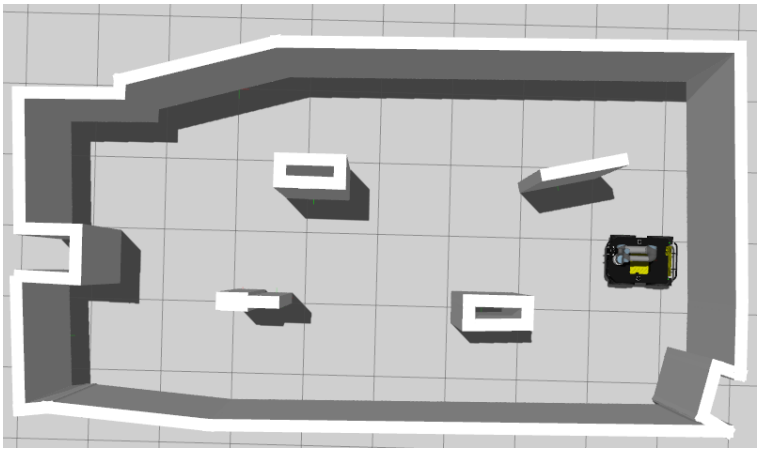


Figura 5.8: Simulación de ambiente real.

Concepto	Ambiente real simulado
Tasa de acierto	0.722±0,051
Tasa de *Timeout*	0.059±0,078
Tasa de colisiones	0.218±0,091
Pasos promedios	107.663±79,046

Tabla 5.6: Evaluación políticas en ambiente real simulado tarea navegación local.

En primer lugar al revisar los resultados obtenidos en la tarea de navegación local, se puede observar un decaimiento aún mayor en la tasa de acierto, este comportamiento puede ser explicado al revisar la tasa de colisiones obtenidas, la cual es la más alta a lo largo de todas las validaciones. La gran tasa de colisiones presente en esta validación, dan un indicio de que las diferencias existentes en los espacios libres donde el agente debía navegar entre los ambientes, demuestran que el agente fue entrenado en un ambiente más simple que el ambiente donde fue validado en el mundo real. Estos resultados demuestran la importancia de considerar dentro del diseño del ambiente de entrenamiento, las dimensiones donde es posible que el agente pueda navegar.

Concepto	Ambiente real simulado
Tasa de acierto	$0{,}554 \pm 0{,}092$
Tasa de *Timeout*	$0{,}272 \pm 0{,}092$
Tasa de colisiones	$0{,}107 \pm 0{,}006$
Tasa de navegación sobre posición objetivo	$0{,}067 \pm 0{,}014$
Pasos promedios	$133{,}608 \pm 17{,}321$

Tabla 5.7: Evaluación políticas en ambiente real simulado tarea manipulación móvil.

Finalmente los resultados expuestos en la Tabla 5.7, denotan una gran baja en las métricas obtenidas en comparación a los ambientes de validación, si bien el análisis efectuado para el caso de navegación es extensible a esta tarea, no es la única razón del mal comportamiento reportado. Debe ser considerada no solamente la subida de la tasa de colisiones, si no también la subida en la tasa de la condición de *Timeout*, siendo esto ultimo un punto importante a considerar, ya que también entrega indicios de un problema de exploración. A diferencia de los problemas de exploración presentados en las trayectorias del mundo real, este caso puede deberse también a lo simple que es el ambiente donde fue entrenada la política en comparación al utilizado en el mundo real.

Ambos resultados expuestos presentan una gran diferencia con las validaciones en los ambientes utilizados a lo largo de la simulación, esto sumado a los resultados cualitativos en el mundo real, indican que si bien se obtienen buenos resultados al transferir las políticas, estos resultados podrían verse perjudicados por las grandes diferencias en el diseño de los ambientes. Esto último agrega una componente importante a considerar a considerar en trabajos futuros, la cual consiste en mejorar el diseño de los espacios libres en los ambientes donde el agente es entrenado, con la finalidad que este observe casos más complejos y donde se requiera una mayor precisión en los comandos de velocidad efectuados.

Capítulo 6

Conclusiones

En el presente trabajo se realizó un estudio del estado del arte relacionado tanto al aprendizaje reforzado profundo, como de la tarea de manipulación móvil. Por un lado en el ámbito del aprendizaje reforzado se presentaron tanto la formalización de los procesos de aprendizaje como también diferentes algoritmos de aprendizaje, parte del actual estado del arte. Por otro lado fue presentado el estado del arte en la resolución de la tarea de manipulación móvil, explicando en que principios se basan los diferentes acercamientos clásicos, introducir diferentes acercamientos basados en DRL, comentando principalmente la idea general y las actuales deficiencias.

Posteriormente la problemática a resolver es modelada como un administrador o manager, separando las principales componentes que participan a la hora de realizar manipulación móvil, siendo estas las tareas de navegación local y *Reaching* de posición. Ambas tareas consideradas parte del problema de manipulación móvil son modeladas como un proceso de aprendizaje reforzado, formalmente definidas y explicadas las diferentes decisiones de diseño, a la vez la tarea de manipulación móvil es formalmente modelada e introducida como un planificador que hace uso de las políticas de navegación local y *Reaching* de posición.

Debido a la necesidad de contar con una simulación del robot para poder realizar un correcto entrenamiento mediante aprendizaje reforzado, es que fueron utilizadas de forma exitosa las diversas herramientas de simulación provistas por *Gazebo* como también repositorios oficiales de simulación tanto de la base móvil Husky como del brazo robótico UR5, sumado a la simulación de los sensores lasers Hokuyo UTM-30LX-EW montados en el robot, resultando en una correcta implementación de la simulación de la plataforma de estudio, donde pueden ser entrenados diversos algoritmos de aprendizaje reforzado. Debido a que esta simulación considera los diversos sensores y actuadores a utilizar se cumple de forma exitosa el primer objetivo planteado.

Luego son desarrollados e implementados los algoritmos de aprendizaje para cada una de las tareas propuestas, siguiendo en todos los casos una misma metodología de entregar en primer lugar las condiciones episódicas, para luego presentar la parametrización de las políticas utilizadas junto a sus hiperparámetros de entrenamiento. Por último, se detallan consideraciones en el entrenamiento de cada una de las tareas en conjunto a las curvas de

aprendizaje obtenidas.

Los resultados obtenidos en las diferentes curvas de entrenamiento, validan las implementaciones de los algoritmos de aprendizaje Deep Q-Net [7] y DDPG [42], como también de las diversas parametrizaciones utilizadas en cada política. Los resultados obtenidos durante entrenamiento muestran una convergencia estable a lo largo de las instancias independientes, como también tasas de acierto superiores al 80 % para todas la tareas, a esto agregado que las diferentes curvas expuestas demuestran que los agentes aprenden no solamente a no realizar diversos comportamientos no deseados, si no más bien a converger en políticas capaces de realizar las tareas planteadas. Todos estos resultados validan el objetivo especifico planteado de implementación tanto de algoritmos de aprendizaje, como del diseño de parametrizaciones y modelado de las diversas tareas.

Posteriormente son validadas las políticas en simulación, tanto en el ambiente donde fue entrenado como en un nuevo ambiente nunca antes visto por los agentes, entregando las diferentes métricas de desempeño obtenidas. Las métricas obtenidas en esta etapa validan el correcto aprendizaje de las tareas modeladas, donde dentro de los resultados obtenidos destacan la estabilidad en el desempeño a lo largo de las diferentes políticas entrenadas, donde incluso al enfrentarse a un ambiente desconocido se logran mantener tasas de acierto altas. Estos resultados logran cumplir con uno de los objetivos planteados correspondientes a la generación de ambientes de entrenamiento y validación que resulten ser desafiantes para el agente.

Luego se da paso a las validaciones físicas de todas las políticas entrenadas. Si bien los resultados presentados en la plataforma física para las tareas de navegación local y manipulación móvil son cualitativos, logran validar que el uso de herramientas de simulación física orientada a robótica, resultan útiles a la hora del desarrollo de algoritmos de aprendizaje reforzado. Esto sumado a los buenos resultados obtenidos en la validación física de las políticas de *Reaching* de posición, donde fue superado el *reality gap* debido a una correcta simulación de los diversos componentes del robot. Estas validaciones cumplen con otro de los objetivos propuestos en este trabajo, consistente en la validación de políticas entrenadas en simulación en el mundo real.

Los resultados obtenidos en la validación física de la tarea de manipulación móvil, logran perfilar la metodología aplicada como una alternativa interesante a la hora de resolver la tarea de manipulación móvil, donde incluso cabe mencionar que esta metodología no requiere que las sub-tareas sean aprendidas mediante RL, dando espacio a utilizar enfoques clásicos a la hora de resolver alguna o todas las sub-tareas intrínsecas en la tarea global a resolver.

A grandes rasgos los resultados obtenidos en el mundo real evidencian una correcta transferencia de las políticas obtenidas, pero indican un problema de exploración durante el entrenamiento, este problema de exploración puede ser explicado por los amplios espacios que se tienen en el ambiente donde fue entrenado, como también debido a condiciones impuestas a la hora del muestreo de objetivo. Para los problemas de exploración son propuestas diferentes posibles soluciones, las cuales son: remover la condición de distancia impuesta en el muestreo de objetivos de manipulación móvil, y el rediseño del ambiente donde es entrenado el agente, considerando en este rediseño la aplicación final donde se busca que el robot se desempeñe, ya que esto determina ciertas características tales como los espacios libres disponibles.

Finalmente es posible concluir que si bien los resultados obtenidos están lejos de ser perfectos, se cumplió el objetivo general propuesto, consistente en la obtención de una aplicación funcional basada completamente en aprendizaje reforzado que sea capaz de llevar a un efector a una posición 3D deseada en el mapa, aplicación la cual fue entrenada en simulación y posteriormente probada en el mundo real.

6.1. Trabajo Futuro

Al analizar los resultados obtenidos en la validación de las políticas de manipulación móvil, el primer paso a realizar consiste en la re-iteración del entrenamiento pero considerando distintas condiciones episódicas con el objetivo de evitar las fallas encontradas.

Debido a los resultados prometedores obtenidos mediante la metodología propuesta, se abre una nuevo foco de estudio a la hora de resolver la problemática de manipulación móvil, donde aún existe un amplio desarrollo a realizar, tanto en la funciones de recompensa como en las observaciones del agente administrador, como también en la prueba de esta metodología en otros sistemas robóticos con diferentes características a las poseídas por el conjunto AMTC_husky.

Al considerar los resultados obtenidos al simular el ambiente real utilizado, demuestran que se debe considerar un ambiente diferente para realizar el entrenamiento. Los lineamientos propuestos para realizar esta tarea de re-diseño del ambiente consisten, en primer lugar en definir el tipo de aplicación donde se busca que el robot se desenvuelva, ya que esta característica entrega las dimensiones donde es esperado que el robot navegue. Una vez obtenidas estas dimensiones se propone a diseñar un ambiente donde existan pasillos o cruces que cumplan estas restricciones de espacio.

Una componente importante a considerar a la hora de manipular objetos, es que estos poseen ciertas orientaciones donde un gripper es capaz de sostenerlo. Esta restricción se traduce en una modificación en la tarea de *Reaching* de posición a una tarea de *Reaching* de pose, lo cual agrega la complejidad del alineamiento del efector con respecto a la orientación de la pose. Es por esto que un paso natural a la hora manipular objetos consiste en la implementación y entrenamiento de la tarea mencionada, considerando también su inclusión en la arquitectura de alto nivel desarrollada.

Otra de las deficiencias encontradas en la formulación consiste en la no capacidad de la política de *Reaching* de evadir colisiones externas. Es por esto que otro desarrollo futuro a realizar consiste en el entrenamiento de esta tarea considerando que el brazo no solamente puede colisionar consigo mismo, si no también con diversos obstáculos en el ambiente.

Por último también nace la necesidad de contar con un sistema dedicado al reconocimiento de objetos, como de puntos de *grasping* para la obtención de los objetivos de manipulación móvil. Todo esto para no depender de la precisión en la localización del robot en un mapa, si no de una referencia local del objeto con respecto al robot, tal y como son formuladas las diversas observaciones.

Bibliografía

[1] F. Leiva and J. Ruiz-del Solar, "Robust rl-based map-less local planning: Using 2d point clouds as observations," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5787–5794, 2020.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[3] H. Zhang, Q. Sheng, Y. Sun, X. Sheng, Z. Xiong, and X. Zhu, "A novel coordinated motion planner based on capability map for autonomous mobile manipulator," *Robotics and Autonomous Systems*, vol. 129, p. 103554, 2020. [Online]. Available: https://doi.org/10.1016/j.robot.2020.103554

[4] IEEE-RAS, "About ras," 2021. [Online]. Available: https://www.ieee-ras.org/about-ras

[5] *IEEE-RAS Technical-committee Mobile Manipulation [Online]*, available: https://www.ieee-ras.org/mobile-manipulation.

[6] 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. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1527–1533, 2017.

[9] I. Popov, N. M. O. Heess, T. P. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerík, T. Lampe, Y. Tassa, T. Erez, and M. A. Riedmiller, "Data-efficient deep reinforcement

learning for dexterous manipulation," *ArXiv*, vol. abs/1704.03073, 2017.

[10] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, pp. 1–14, 2019.

[11] Melonee Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch & Freight: Standard Platforms for Service Robot Applications," *Workshop on Autonomous Mobile Service Robots, held at the 2016 International Joint Conference on Artificial Intelligence*, pp. 2–7, 2016. [Online]. Available: http://fetchrobotics.com/research/

[12] *IEEE Robots Human Support Robot(HSR) [Online]*, available: https://robots.ieee.org/robots/hsr/.

[13] *IEEE Spectrum Ex-Googler's Startup Comes Out of Stealth With Beautifully Simple, Clever Robot Design. [Online]*, available: https://spectrum.ieee.org/automaton/robotics/home-robots/hello-robots-stretch-mobile-manipulator.

[14] R. E. Andersen, E. B. Hansen, D. Cerny, S. Madsen, B. Pulendralingam, S. Bøgh, and D. Chrysostomou, "Integration of a Skill-based Collaborative Mobile Robot in a Smart Cyber-physical Environment," *Procedia Manufacturing*, vol. 11, no. June, pp. 114–123, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.promfg.2017.07.209

[15] S. Thakar, L. Fang, B. Shah, and S. Gupta, "Towards Time-Optimal Trajectory Planning for Pick-and-Transport Operation with a Mobile Manipulator," *IEEE International Conference on Automation Science and Engineering*, vol. 2018-Augus, pp. 981–987, 2018.

[16] A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, A. Fernandez, and J. Molina, "Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning," *Applied Sciences (Switzerland)*, vol. 9, no. 2, 2019.

[17] A. Jain and C. C. Kemp, "EL-E: An assistive mobile manipulator that autonomously fetches objects from flat surfaces," *Autonomous Robots*, vol. 28, no. 1, pp. 45–64, 2010.

[18] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow, "Mobile manipulation through an assistive home robot," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5313–5320, 2012.

[19] G. B. Avanzini, A. M. Zanchettin, and P. Rocco, "Constraint-based Model Predictive Control for holonomic mobile manipulators," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-December, no. i, pp. 1473–1479, 2015.

[20] R. Ancona, "Redundancy modelling and resolution for robotic mobile manipulators: a general approach," *Advanced Robotics*, vol. 31, no. 13, pp. 706–715, 2017. [Online]. Available: https://doi.org/10.1080/01691864.2017.1326842

[21] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-Body MPC for a Dynamically Stable Mobile Manipulator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3687–3694, 2019.

[22] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, 2019.

[23] I. Sarantopoulos, M. Kiatos, Z. Doulgeri, and S. Malassiotis, "Split Deep Q-Learning for Robust Object Singulation," 2019. [Online]. Available: http://arxiv.org/abs/1909.08105

[24] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. Petillot, and S. Wang, "Learning mobile manipulation through deep reinforcement learning," *Sensors (Switzerland)*, vol. 20, no. 3, pp. 1–18, 2020.

[25] C. Li, F. Xia, R. Martin-Martin, and S. Savarese, "HRL4IN: Hierarchical Reinforcement Learning for Interactive Navigation with Mobile Manipulators," no. CoRL, 2019. [Online]. Available: http://arxiv.org/abs/1910.11432

[26] J. Kindle, F. Furrer, T. Novkovic, J. J. Chung, R. Siegwart, and J. Nieto, "Whole-Body Control of a Mobile Manipulator using End-to-End Reinforcement Learning," no. June, 2020. [Online]. Available: http://arxiv.org/abs/2003.02637

[27] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[28] K. Lobos-Tsunekawa, F. Leiva, and J. Ruiz-Del-Solar, "Visual navigation for biped humanoid robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3247–3254, 2018.

[29] K. Yokoyama and K. Morioka, "Autonomous Mobile Robot with Simple Navigation System Based on Deep Reinforcement Learning and a Monocular Camera," *Proceedings of the 2020 IEEE/SICE International Symposium on System Integration, SII 2020*, pp. 525–530, 2020.

[30] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[31] W. Cannon Lewis, M. Moll, and L. E. Kavraki, "How Much Do Unstated Problem Constraints Limit Deep Robotic Reinforcement Learning?" *arXiv*, 2019.

[32] A. Rupam Mahmood, D. Korenkevych, B. J. Komer, and J. Bergstra, "Setting up a reinforcement learning task with a real-world robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4635–4640.

[33] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv*, pp. 1–16, 2018.

[34] C. Robotics, "Husky ugv - outdoor field research robot by clearpath," 2021. [Online]. Available: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

[35] U. Robots, "Ur5 collaborative robot arm," 2021. [Online]. Available: https://www.universal-robots.com/products/ur5-robot/

[36] *Husky unmanned ground vehicle robot, Manipulator Package [Online]*, available: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot.

[37] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.

[38] T. Jaakkola, S. Singh, and M. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," *Advances in Neural Information Processing Systems*, vol. 7, 11 1999.

[39] C. J. C. H. Watkins, "Learning from delayed rewards. PhD thesis," 1989.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," pp. 1–9, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[41] L.-J. Lin, "Reinforcement learning for robots using neural networks," Ph.D. dissertation, USA, 1992, uMI Order No. GAX93-22750.

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[43] F. Burget, "BI 2 RRT *: An Efficient Sampling-Based Path Planning Framework for Task-Constrained Mobile Manipulation."

[44] G. Oriolo and C. Mongillo, "Motion planning for mobile manipulators along given end-effector paths," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, no. April, pp. 2154–2160, 2005.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[46] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, 1st ed. Springer Publishing Company, Incorporated, 2013.

[47] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5049–5059, 2017.

[48] S. Luo, H. Kasaei, and L. Schomaker, "Accelerating Reinforcement Learning for Reaching Using Continuous Curriculum Learning," *Proceedings of the International Joint Conference on Neural Networks*, no. February, 2020.

[49] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking

Reinforcement Learning Algorithms on Real-World Robots," no. CoRL, pp. 1–31, 2018. [Online]. Available: http://arxiv.org/abs/1809.07731

[50] J. Duan, S. E. Li, Y. Guan, Q. Sun, and B. Cheng, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data," *IET Intelligent Transport Systems*, vol. 14, no. 5, pp. 297–305, 2020.

[51] O. S. R. Foundation, "Gazebo," 2021. [Online]. Available: http://gazebosim.org/

[52] "Pioneer 3 operations manual," 2006. [Online]. Available: https://www.inf.ufrgs.br/~prestes/Courses/Robotics/manual_pioneer.pdf

[53] O. AI, "Open ai a toolkit for developing and comparing reinforcement learning algorithms." 2020. [Online]. Available: https://gym.openai.com/

[54] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017. [Online]. Available: http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf

[55] "husky_gazebo," 2020. [Online]. Available: http://wiki.ros.org/husky_gazebo

[56] "ur_gazebo," 2020. [Online]. Available: http://wiki.ros.org/ur_gazebo

[57] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 77–85, 2017.

[58] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: efficient position estimation for mobile robots," *Proceedings of the National Conference on Artificial Intelligence*, no. Handschin 1970, pp. 343–349, 1999.

[59] B. P. Gerkey, "Amcl," 2020. [Online]. Available: http://wiki.ros.org/amcl

[60] ——, "gmapping," 2019. [Online]. Available: http://wiki.ros.org/gmapping