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

NAVEGACIÓN AUTÓNOMA EN TÚNELES USANDO APRENDIZAJE REFORZADO OFFLINE

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

RUDY ANDRÉS GARCÍA ALVARADO

PROFESOR GUÍA:
JAVIER RUÍZ DEL SOLAR SAN MARTÍN
PROFESOR CO-GUÍA:
FRANCISCO LEIVA CASTRO

MIEMBROS DE LA COMISIÓN:
MARCOS ORCHARD CONCHA
EDUARDO MORALES MANZANARES

Este proyecto ha sido parcialmente financiado por:
FONDECYT 1201170 y ANID BECAS/MAGÍSTER NACIONAL 22210730

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
Y AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: RUDY ANDRÉS GARCÍA ALVARADO
FECHA: 2023
PROF. GUÍA: JAVIER RUÍZ DEL SOLAR SAN MARTÍN

NAVEGACIÓN AUTÓNOMA EN TÚNELES USANDO APRENDIZAJE REFORZADO OFFLINE

Una de las grandes limitantes de ocupar aprendizaje reforzado en aplicaciones reales es que usualmente este tipo de métodos requiere interactuar con el ambiente por largos períodos, lo que puede resultar costoso e inclusive peligroso dependiendo de la tarea. Por otro lado, el aprendizaje reforzado offline permite entrenar agentes accediendo a datos previamente recolectados, prescindiendo de la interacción con el ambiente y por tanto reduciendo riesgos y costos. Es por esta razón que resulta prometedor aplicar este tipo de métodos para optimizar procesos en industrias que poseen muchas restricciones. En este trabajo se propone un método que utiliza aprendizaje reforzado offline para el entrenamiento de agentes capaces de controlar de forma autónoma vehículos mineros de gran envergadura al interior de túneles, mejorando la eficiencia y seguridad en la operación de este tipo de máquinas al ofrecer una alternativa a la operación realizada por humanos, la cual posee ineficiencias y peligros inherentes a la minería. Adicionalmente, se realizan diversos experimentos en un entorno simulado los cuales validan el método propuesto y explican los principales requerimientos para poder aplicar el método en un entorno real.

A mi familia.

Agradecimientos

Quiero agradecer a mi familia, que me ha apoyado incondicionalmente a lo largo de mi carrera, estando conmigo en mis peores momentos. A mi papá, Rodrigo García, que siempre ofreció lo mejor de sí para sacarme adelante. A mi mamá, Sandra Alvarado, que con su cálido afecto materno me permitió mantenerme firme. Y a mis hermanos, Ronny y Rodrigo García, que con risas me sacaron del estrés. A Cristian Rodríguez, mi tío, pues su apoyo incondicional y fraterno me hizo sentir acompañado inclusive estando tan lejos de mi hogar.

A mis amigos, que siempre creyeron en mí y me motivaron a perseguir esta carrera desde el inicio. Al Jano, Alejandro Saldivia, fiel compañero que la última década me acompaño en los torcidos caminos de la vida y me motivó a distraerme cuando el panorama se veía difícil. A, María José, o la Jo, que se mantuvo estoica junto a mí en momentos donde más lo necesité. A todos ellos quienes me acompañaron, escucharon y animaron los últimos años.

A mi profesor, Javier, principal promotor de esta tesis. Quién con su profesionalismo y expertiz académica guió esta tesis por el camino correcto. Gracias por creer en mí. A Francisco, mi profesor co-guía, pues gracias a su experiencia y sabiduría me hizo ver simple lo que era complejo. Al Advanced Mining Technology Center (AMTC), que me prestaron equipos importantes para poder realizar incontables experimentos.

A la Agencia Nacional de Investigación y Desarrollo (ANID) que gracias a su beca Magíster Nacional 2021-22210730 financió gran parte de este magíster y me dio soporte económico.

Muchísimas gracias a todos.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Lista de acrónimos

AWAC Advantage Weighted Actor-Critic.

AWR Advantage Weighted Regression.

CQL Conservative Q-Learning

CRR Critic Regularized Regression.

CWP Critic-Weighted Policy.

DDPG Deep Deterministic Policy Gradient.

DQN Deep Q-Networks

DL Deep Learning. Aprendizaje profundo.

IRL Inverse Reinforcement Learning. Aprendizaje reforzado inverso.

IQL Implicit Q-Learning.

LHD Load-Haul-Dump. Cargar-Transportar-Descargar.

LiDAR Light Detection and Ranging. Sistema de medición y detección de objetos mediante láser.

MAV Micro Aerial Vehicle. Micro vehículo aéreo.

MDP Markov Decision Process. Proceso de decisión de Markov.

ML Machine Learning. Aprendizaje de máquinas.

MPC Model Predictive Control. Control predictivo por modelo.

OOD Out of distribution. Fuera de la distribución.

ORL Offline Reinforcement Learning. Aprendizaje reforzado offline.

PCA Principal Component Analysis.

PID Proportional-integral-derivative. Proporcional-integral-derivativo.

PO-MDP Partiable Observable Markove Decision Process. Proceso de decisión parcialmente observable.

RL Reinforcement Learning. Aprendizaje reforzado.

ROS Robotic Operating System. Sistema operativo robótico.

TD3 Twin Delayed Deep DDPG.

Capítulo 1

Introducción

1.1. Motivación

En las últimas décadas diversos métodos basados en aprendizaje tales como aprendizaje supervisado, aprendizaje no-supervisado o aprendizaje reforzado, han florecido y presentado alternativas para resolver problemas que nunca antes pudieron haber sido abarcados de manera programática, revolucionando así la tecnología y brindando soluciones de todo tipo en diversos ámbitos como por ejemplo en seguridad, sistemas de recomendación, análisis de datos, visión computacional, conducción autónoma o robótica [20] [31].

En particular, el área del aprendizaje reforzado o *reinforcement learning* (RL) es una de las más prometedoras por su capacidad para resolver una diversa gama de problemas desafiantes, principalmente asociados a la toma de decisiones. Inspirado en la psicología conductista, su objetivo es encontrar (o entrenar) agentes que sean capaces de tomar acciones en un entorno con el fin de maximizar una recompensa. Este objetivo se puede abordar de diversas maneras, cada una de ellas con una fuerte base matemática. Por ejemplo, en [42] se presentan algoritmos de aprendizaje reforzado como Q-learning o SARSA. Uno de los denominadores comunes de todos estos algoritmos es que de una forma u otra requieren poder utilizar un *ambiente*, simulado o real, de tal forma que el agente a entrenar pueda interactuar con él y recibir retroalimentación sobre el resultado de sus acciones. En el caso del uso de simuladores, si se busca un ambiente que capture fielmente todos los detalles es necesario tener conocimiento experto en el uso de éste. Además, a medida que escala la complejidad, disminuye la eficiencia en la adquisición de datos, acabando en simulaciones más lentas que desaceleran el aprendizaje que un agente pueda tener dentro de él. A contra parte, en el caso de utilizar un ambiente real se captura toda la complejidad real del problema pero se vuelve sumamente costoso realizar experimentos. Si bien han surgido exitosas aplicaciones usando este tipo paradigma, estas siguen siendo limitadas por las dificultades mencionadas.

Una alternativa a estas limitaciones es el *offline reinforcement learning* u ORL, el cual cambia la forma en la cual se concibe la relación entre el agente a entrenar y el ambiente (ver Figura 1.1). Inspirado por el éxito del aprendizaje supervisado, en este paradigma se busca entrenar un agente usando un conjunto de datos recolectado previamente mediante la interacción de otro agente (experto u otro) con el ambiente, eliminando la necesidad de tener que interactuar de manera onli-

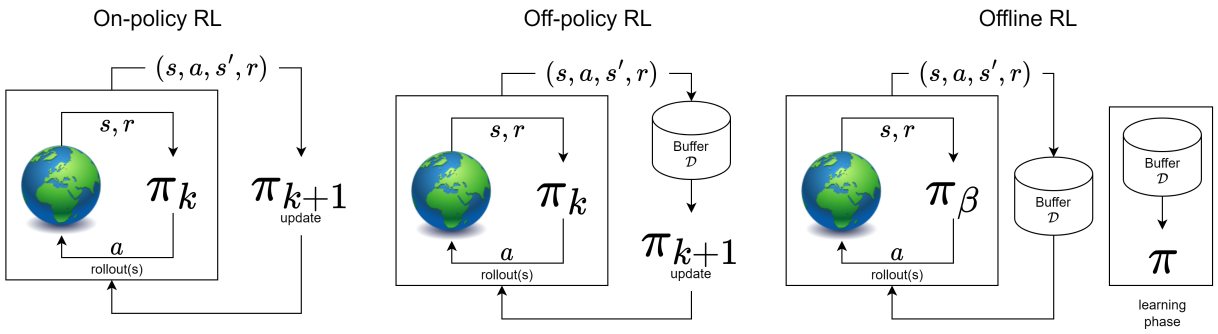

Figura 1.1: Tres paradigmas de aprendizaje reforzado. (a) En on-policy RL la política es actualizada a partir de la experiencia recolectada por sí misma. (b) En off-policy RL la política es actualizada a partir de un *buffer*, el cual contiene un conjunto de experiencias previas recolectadas por políticas anteriores durante el entrenamiento y recibe experiencias nuevas adquiridas por la política en entrenamiento. (c) En offline RL la política es actualizada a partir de un conjunto fijo de experiencias previas que pudo haberse construido a partir de la experiencia de una o más políticas desconocidas. La Figura fue inspirada de [26].

ne con el ambiente durante el aprendizaje y permitiendo la aplicación de este tipo de métodos en problemas donde de otra forma hubiese sido infactible. Cabe mencionar que este enfoque permite acercar gran parte de los avances recientes de áreas de *Machine Learning* (ML) o *Deep Learning* (DL) al aprendizaje reforzado. Por ende, para la comunidad científica resulta atractivo que el mismo éxito del reconocimiento de patrones podría ser llevado a sistemas de toma de decisiones en diferentes escenarios como en medicina, chatbots o robótica.

En este trabajo se busca aprovechar las ventajas que ofrece el aprendizaje reforzado para resolver un problema específico de conducción autónoma en la industria minera. Las máquinas LHD (*Load, Haul, Dump*; Carga, Acarreo y Desgarga) son vehículos cruciales en la proceso productivo de las mineras. Cumplen el rol de cargar, acarrear y descargar material minado desde un punto hacia otro. Es de tal relevancia que más del 75 % de las mineras alrededor del mundo utilizan este tipo de maquinaria [45]. Dependiendo de la empresa fabricante de la máquina y del modelo, estos vehículos pueden tomar diversas dimensiones, pero en promedio rondan en los 7 metros de largo, 3 metros de alto y 3 metros de ancho, alcanzando capacidades de carga entre 3 y 21 toneladas, es decir, son máquinas de gran envergadura (ver Figura 1.2). Es por esta razón que su operación al interior de túneles en minas puede ser una tarea bastante compleja. Inclusive, en ciertos escenarios el espacio disponible para navegar o realizar giros puede ser muy limitado, por lo tanto la realización de estas maniobras puede dificultar a los operadores más experimentados. En general, estas máquinas se operan de tres formas: (i) *presencial*, con un operador al interior de la máquina, (ii) *remota*, con un operador en una sala de operación conectada remotamente a la máquina o (iii) *autónoma*, a través de algún software especializado. Por ende, este trabajo se enmarca en el contexto de una solución autónoma.

A nivel metodológico, en esta tesis se busca en primer lugar validar un modelo que permita entrenar agentes de navegación autónoma. Para validar este modelo, se entrena un agente usando aprendizaje reforzado, el cual mediante acceso al ambiente podrá probar que el diseño propuesto es válido. Adicionalmente, gracias a este proceso de validación se extraen datos de operación de este agente autónomo, lo que permitirá construir un conjunto de datos vasto, diverso y apto para la realización de diversos experimentos. También, se extraen datos de teleoperación humana, los cuales constituirán otro tipo de conjunto de datos, el cual si bien es más pequeño, es mucho más cercano a

Figura 1.2: Fotografía de LHD modelo LF-11H producido por la empresa alemana GHH [3].

la realidad y a la forma de operación que se podría encontrar al interior de una mina. Luego, usando estos datos se entrenan agentes usando aprendizaje reforzado pero esta vez de manera offline. Estos agentes son extensivamente evaluados con la intención de determinar qué método de aprendizaje reforzado es el mejor para esta aplicación particular y también determinar las características de los datos necesarios para poder alcanzar cierto nivel de rendimiento.

1.2. Hipótesis

Las hipótesis de este trabajo es la siguiente:

Es posible utilizar aprendizaje reforzado offline para entrenar agentes autónomos capaces de navegar en simulaciones de minas subterráneas, obteniendo un rendimiento similar a los agentes obtenidos utilizando aprendizaje reforzado online.

1.3. Objetivos

Dicho lo anterior, el objetivo general de este trabajo es resolver el problema de navegación autónoma de un LHD al interior de una mina usando aprendizaje reforzado offline en un entorno simulado y determinar las condiciones más relevantes para la aplicación de este método en un entorno real. En consecuencia, este trabajo tiene el potencial de revolucionar la industria minera al mejorar la eficiencia, seguridad y rentabilidad de las operaciones de las máquinas LHD. Además, también puede tener implicaciones para otras industrias que utilizan conducción autónoma, incluyendo la logística, la construcción y la agricultura. Para cumplir con este objetivo, se establecen los siguientes objetivos específicos:

- Efectuar una revisión exhaustiva de la literatura sobre aprendizaje reforzado offline y navegación autónoma.
- Formalizar el problema de la navegación autónoma en minas como un problema de aprendi-

zaje reforzado.

- Implementar un enfoque basado en aprendizaje reforzado offline para la navegación autónoma de vehículos LHD en minas.
- Entrenar y evaluar agentes de navegación usando aprendizaje reforzado profundo online.
- Validar experimentalmente la eficacia del método propuesto en un entorno simulado.
- Recopilar datos de operación durante la navegación de LHD por parte de agentes autónomos y humanos.
- Evaluar críticamente el rendimiento de los algoritmos de aprendizaje reforzado offline y su eficiencia en términos de la cantidad de datos requeridos.
- Identificar las fortalezas, debilidades y oportunidades futuras que surjan a partir de la implementación del método propuesto.

1.4. Estructura de la tesis

La estructura de esta tesis es la siguiente. En el primer capítulo se presenta la introducción con una contextualización y presentación del problema a resolver, seguida de la revisión bibliográfica relevante en el capítulo dos. El tercer capítulo se enfoca en el marco teórico y se describe formalmente el aprendizaje reforzado, así como diversos algoritmos relacionados. En el cuarto capítulo se describe la metodología utilizada, incluyendo aspectos de simulación, modelamiento, planning, evaluación, entre otros. En el quinto capítulo se presentan los resultados obtenidos, incluyendo los resultados de aprendizaje reforzado online, offline y la generación de datasets. Finalmente, en el sexto capítulo se discuten los resultados y en el último capítulo se presenta la conclusión.

Capítulo 2

Revisión de la literatura

Uno de los pilares de este trabajo es el realizado por Mascaró et al. [30]. En dicha investigación se identifican diversas dificultades que deben ser sorteadas para resolver exitosamente el problema de navegación de LHDs, tales como la capacidad de poder localizar la máquina dentro de una topología de túneles y navegar de forma segura a través de ellos. Luego, se propone un sistema íntegro que resuelve todos los componentes necesarios para lograr la navegación autónoma: localización usando percepeción basada en sensores LiDAR, planning usando la topología de los túneles y control usando el método MPC (*Model Predictive Control*). Este sistema propuesto por Mascaró et al. posee un alto de grado de ingeniería en cada uno de sus componentes y por ende requiere del ajuste de muchos hiper-parámetros, los cuales se encuentran presentes desde el modelamiento de la máquina, pasando por su cinemática, hasta el funcional de costo usado en el módulo de control. A diferencia del trabajo mencionado, en este se busca prescindir del alto grado de customización usando un sistema de navegación basado en aprendizaje profundo, el cual a su vez unifica todos los componentes en un único modelo que se encarga de la percepción y control.

Por otro lado, en el trabajo propuesto por Mascaró et al. se plantea una representación topológica de la mina que denominan *Topological Map*. En dicha representación se representa jerárquicamente tanto los túneles, las intersecciones y sus conexiones, permitiendo así inferir rutas globales óptimas entre un punto y otro mediante la solución a un problema de optimización que resuelven usando el algoritmo de Dijkstra [7]. Si bien la presente tesis aborda el problema de planning, lo hace de una forma local y no global, ya que el enfoque está orientado hacia el aprendizaje de la navegación misma. Cabe mencionar que es perfectamente posible integrar el planning propuesto en [30] para obtener misiones locales que luego sean ejecutadas por el sistema propuesto. Otro aspecto a destacar es que en [30] se realiza una validación del sistema propuesto usando una máquina real en un ambiente de prueba pero también al interior de una mina real, logrando resultados aceptables si se compara al sistema con un operador humano.

El problema de la navegación autónoma de vehículos mineros terrestres en el interior de las minas ha sido poco explorado por la comunidad científica, es tal el caso que al momento de la escritura de esta tesis y según la investigación realizada, [30] es uno de los pocos que resuelve este problema de forma completa. Sin embargo, existe una serie de trabajos enfocados en este problema pero utilizando vehículos aéreos ([33], [28], [8]). Por ejemplo, en [28] se propone un sistema capaz de navegar de forma autónoma un vehículo aéreo pequeño o MAV (*Micro Aerial Vehicle*) al interior

de túneles. Este sistema utiliza sensores como cámaras y LiDARs para inspeccionar el ambiente y modelarlo a medida que se realiza la navegación. Si bien estos trabajos abordan un problema de alta complejidad, no tienen tantas restricciones espaciales en la navegación debido a que usualmente los vehículos aéreos son muchos más pequeños que los terrestres. No obstante, existe una serie de dificultades que ambos problemas comparten, tales como la percepción y el control.

En [25] se presenta un trabajo enfocado en el desarrollo y evaluación de algoritmos para permitir la operación autónoma y semi-autónoma de LHDs en entornos de minería subterránea. El primer modo de operación se basa en la navegación completamente autónoma en entornos estáticos, utilizando un marco existente de navegación reactiva basado en lógica difusa, algoritmos de detección de características para seguir túneles y localización topológica basada en datos de escáner de rango láser 2D. Este modo ha sido evaluado en pruebas cuantitativas y cualitativas en entornos interiores utilizando robots de investigación tradicionales. El segundo modo de operación explorado es la operación semiautónoma, donde la funcionalidad de autonomía local a bordo del vehículo ayuda a un operador remoto en la conducción del vehículo a lo largo de una ruta libre de colisiones. Se realizó un estudio de usabilidad en una mina real que muestra que la autonomía local tiene el potencial de mejorar significativamente la productividad de un LHD operado remótamente. Basado en estos resultados, se extendió dicho trabajo a un sistema comercial de tele-operación para minas subterráneas. Este sistema se ha verificado en experimentos realizados en una mina de prueba con un LHD de 38 toneladas y en simulaciones para demostrar que funciona en entornos de minas subterráneas arbitrarias. El trabajo presentado en en [25] parece ser sólido y bien desarrollado, ya que se ha investigado y evaluado el uso de sus métodos en entornos reales. Sin embargo, una posible limitación es que el sistema presentado en el trabajo es comercial, lo que limita el acceso a detalles específicos del sistema y a la evaluación independiente de su rendimiento y eficacia. En general, aunque el trabajo presenta resultados interesantes y prometedores, es importante considerar la falta de información detallada y la falta de antecedentes en la literatura sobre el sistema comercial utilizado.

En paralelo al desarrollo de esta tesis, en [19] se realiza un trabajo que tiene una intersección importante: el modelamiento e implementación de un ambiente de simulación para la navegación de LHDs en minas subterráneas. El objetivo principal de tal trabajo es modelar un LHD y lograr un ambiente de simulación robusto y lo más fiel posible a lo que sería una mina, todo esto basado en ROS/Gazebo [29]. Dado este objetivo, se propone un ambiente bastante avanzado que incluye factores como la modelación cinemática del LHD, sistemas de control para diversas partes de la máquina, la adquisición de un mapa basado en uno real y también un exhaustivo análisis del rendimiento del ambiente de simulación. En el presente trabajo también se desarrolla e implementa un ambiente de simulación para el LHD en el mismo tipo de software (Gazebo/ROS), sin embargo dado que el objetivo es distinto, se alcanza un grado de robustez menor. No obstante, la perspectiva respecto a algunos aspectos como el control y el modelamiento de la máquina es bastante similar.

En [41] se aborda el problema de navegación autónoma usando aprendizaje reforzado offline. Una de las motivaciones principales de dicho trabajo es la observación de que usando aprendizaje reforzado offline es posible extraer políticas sin interacción directa con el ambiente, reduciendo riesgos y potenciales costos. Sin embargo, dicho trabajo va un paso más allá en la dirección asociada al riesgo. Basándose en [14], se proponen mejoras con un fuerte sustento teórico que buscan mejorar la seguridad y estabilidad de las políticas extraídas, lo cual resulta ser de especial interés en aplicaciones de navegación autónoma. Para validar los algoritmos propuestos, se realizan diversos

experimentos en un ambiente de conducción autónoma simple que simula la navegación de autos en autopistas y también maniobras de estacionamiento. Los resultados obtenidos son satisfactorios, mejorando sobre otros métodos del estado del arte en términos de seguridad y eficiencia. Si bien el sistema propuesto es bastante interesante desde un punto de vista teórico, no es claro su impacto en una aplicación más cercana a la realidad.

En [43] se propone un método basado en aprendizaje reforzado en el que se entrena un agente capaz de controlar continuamente un robot móvil sin la necesidad de un mapa. La propuesta incluye utilizar una variante de aprendizaje reforzado denominado *aprendizaje reforzado orientado al objetivo*, en el que establecen como observaciones una composición entre la codificación del objetivo, el estado del robot y las lecturas obtenidas por un sensor LiDAR montado en el robot. Utilizando este modelo, se aplicó el método *Asynchronous Deep Deterministic Policy Gradient*, una versión ampliada del método DDPG [27], para el entrenamiento en un entorno simulado, lo que condujo a la obtención de buenos resultados. Luego, transfieren la política aprendida a un ambiente real y se observa que el agente es capaz de cumplir con éxito los objetivos solicitados. Con lo anterior, se evidencia que este tipo de modelamiento, simple pero efectivo, es suficiente para lograr aprender una política capaz de navegar en ambientes restringidos. La presente tesis comparte algunas similitudes con este trabajo, como por ejemplo el enfoque en el modelamiento y el tipo de navegación. Sin embargo, existen otros aspectos que se busca extender tales como la utilización de modelos más complejos y aún más relevante: prescindir de la necesidad de interactuar con el ambiente. Una posible dirección de trabajo que describe [43], pero que no se explora en el presente trabajo, es la transferencia de la política a un ambiente real. Si bien es interesante realizar este tipo de experimentos, es de suma importancia acortar la brecha entre el simulador y la realidad para lograr navegación exitosa y libre de riesgos. Este problema puede ser abordado aumentando la fidelidad del simulador o también usando métodos de transferencia de simulación a realidad [48]. Sin embargo, el objetivo de esta tesis no es tal.

Otra línea de investigación interesante tiene que ver sobre el entendimiento de los datos que requiere un método de aprendizaje reforzado offline para resolver el problema en cuestión. En [40] se realiza un exhaustivo estudio sobre esta materia. El objetivo de este trabajo es determinar cómo un conjunto de datos afecta el rendimiento de una política obtenida usando offline RL. Teniendo esto en consideración, se proponen dos formas de describir un conjunto de datos, las cuales permiten caracterizarlos en cuanto a la cobertura de los datos que contienen y también su calidad. Para validar la propuesta, realizan diversos experimentos con conjuntos de datos obtenidos de ambientes muy simples (usualmente utilizados por la comunidad para investigación), los cuales luego son usados para entrenar diversos tipos de algoritmos de aprendizaje reforzado. Tras lo anterior, obtienen resultados interpretables sobre qué algoritmos se desempeñan mejor dependiendo de qué datos se utilizan para entrenar sus políticas. Sin embargo, la mayoría de los experimentos realizados utilizan ambientes cuyos espacios de observación y acción son discretos, y aquellos que tienen espacios continuos deben ser discretizados. Esto limita la aplicación de las caracterizaciones propuestas a problemas con espacios de estado/acción continuos. De igual forma y en vista de resolver el caso específico planteado en esta tesis, en el presente se realizan experimentos asociados a poder caracterizar los datos necesarios en términos de cantidad y calidad, de tal forma de proveer una intuición sobre cuantos y cómo obtener los datos en una potencial aplicación real.

Los trabajos mencionados anteriormente abarcan una amplia gama de soluciones y enfoques: desde modelar el ambiente en el que se moverá la máquina, modelar la dinámica de la misma

máquina, hasta mejorar la seguridad en la navegación a través de algoritmos avanzados, y desarrollar simuladores precisos para capturar los detalles en entornos complejos como lo son las minas subterráneas. Además, también se han explorado técnicas de planificación basadas en la topología del ambiente. Este trabajo se enfoca en resolver el problema de navegación autónoma de manera práctica, abordando un contexto específico. Al mismo tiempo, incorpora un marco teórico reciente y prometedor que ofrece ventajas atractivas para el desarrollo de este tipo de tecnologías.

Capítulo 3

Marco teórico

En las siguientes secciones se introduce formalmente el problema de aprendizaje reforzado o *reinforcement learning* (RL) y el de *offline reinforcement learning* (ORL). Además, se describe teóricamente los métodos que serán estudiados en este trabajo.

3.1. Aprendizaje reforzado

El aprendizaje reforzado o *reinforcement learning* consiste en un problema de control de un sistema dinámico definido por una proceso de decisión de Markov (MDP, de sus siglas en inglés), el cual a su vez puede ser un proceso observable o no. En el contexto de esta formalización teórica se utilizará la definición de MDP completamente observable. Más detalles sobre procesos parcialmente observables se pueden encontrar en [26].

El proceso de decisión de Markov o MDP se define por la tupla $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, d_0, r, \gamma)$. Donde $\mathcal{S}$ corresponde al conjunto de estados, $\mathcal{A}$ al conjunto de acciones, T a la distribución condicional de transiciones de la forma $T(s_{t+1}|s_t, a_t)$ que define la dinámica del sistema, d_0 a la distribución de probabilidad del estado inicial s_0, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a la función de recompensas y finalmente $\gamma \in (0, 1]$ define el factor de descuento.

Dada esta formulación, el objetivo final del aprendizaje reforzado es aprender una política que defina una distribución sobre acciones condicionadas a estados $\pi(a_t|s_t)$. Por otro lado se define como "trayectoria" τ a la secuencia de tuplas de estado-acción visitados por la política. En concreto, se define como $\tau = ((s_0, a_0), ..., (s_H, a_H))$ una trayectoria de largo H, la cual puede ser finita o infinita. Dada una MDP $\mathcal{M}$ y una política π, se define también la distribución de la trayectoria $p_\pi(\tau)$.

$$p_\pi(\tau) = d_0(s_0) \prod_{t=1}^{H} \pi(a_t|s_t) T(s_{t+1}|s_t, a_t) \tag{3.1}$$

Dadas estas definiciones se tiene el objetivo del aprendizaje reforzado $J(\pi)$, el cual puede se define como la esperanza de los retornos descontados sobre la distribución de las trayectorias de una política π. A grandes rasgos el problema de aprendizaje reforzado consiste en encontrar políticas

que maximicen el retorno promedio descontado definido en $J(\pi)$.

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right] \tag{3.2}$$

Dos cantidades de suma importancia para RL son la función de valor del estado $V_\pi(s)$ y la función de valor del par estado-acción $Q_\pi(s, a)$. $V_\pi(s_t)$, definida en (3.3), determina el retorno descontado esperado en un estado para una política arbitraria π en el contexto de el MDP, es decir, qué tan bueno es el estado s_t para la política π. En segundo lugar, $Q_\pi(s_t, a_t)$, definida en (3.4), determina el retorno esperado si se toma la acción a_t en el estado s_t dado una política π en el contexto de un MDP.

$$V_\pi(s_t) = \mathbb{E}_{\tau \sim p(\tau|s_t)} \left[\sum_{t'=t}^{H} \gamma^{t'-t} r(s_t, a_t) \right] \tag{3.3}$$

$$Q_\pi(s_t, a_t) = \mathbb{E}_{\tau \sim p(\tau|s_t, a_t)} \left[\sum_{t'=t}^{H} \gamma^{t'-t} r(s_t, a_t) \right] \tag{3.4}$$

Adicionalmente, se notan dos distribuciones marginales de $p_\pi(\tau)$ que resultarán útiles para definiciones y discusiones posteriores. En primer lugar, $d^\pi(s)$, que refiere a la frecuencia en que la política π visita el estado s promediada sobre todos los instantes de tiempos. Dicho en otras palabras, la frecuencia en que una política visita un estado, sin importar cuando. En segundo lugar, se define $d_t^\pi(s_t)$ como la frecuencia en que una política visita un cierto estado al instante t.

Intuitivamente, el objetivo del aprendizaje reforzado detallado en (3.2) representa el promedio ponderado de las recompensas que obtiene un agente dada las decisiones que tomó y los estados que visitó. A grandes rasgos, los diversos métodos de aprendizaje reforzado buscarán maximizar este objetivo.

Una de las técnicas más populares para resolver el objetivo enunciado anteriormente es Q-learning. Esta técnica busca encontrar iterativamente la función de valor Q de la Ecuación (3.4) usando la Ecuación de optimalidad de Bellman. Por esta razón se le denomina un método del tipo value-based, en contraste a métodos tipo policy-based que buscan directamente una política parametrizada. La Ecuación de optimalidad de Bellman establece que la función de valor óptima Q^* satisface la siguiente restricción.

$$Q^*(s, a) = \mathbb{E}[r(s, a) + \gamma \max_a Q^*(s_{t+1}, a_{t+1}) | s_t = s, a_t = a)] \tag{3.5}$$

Gracias a las propiedades de esta ecuación, es posible definir actualizaciones iterativas las cuales convergen a la función Q óptima. A esta técnica se le denomina Temporal Difference learning o TD learning. Al aplicar esta técnica en el contexto de Q-learning es posible establecer la regla de actualización iterativa definida en la Ecuación (3.6).

$$Q^{k+1}(s, a) \leftarrow Q^k(s, a) + \alpha[r + \gamma \max_{a'} Q^k(s', a') - Q^k(s, a)] \tag{3.6}$$

Esta ecuación indica que la función de valor del instante $k+1$ ocupa la función de valor del instante previo y se corrige dependiendo del error que tenga respecto a la optimalidad de Bellman. Una forma simple de ver esto es que se busca minimizar el error entre el lado izquierdo y derecho de la Ecuación 3.5. En la Ecuación (3.6), α corresponde a la tasa de aprendizaje y pondera a lo que se denomina el TD-error. A una variante de este error, que tiene mejores propiedades de convergencia y estabilidad [42], se le denomina error cuadrático de Bellman y corresponde al cuadrado del error descrito anteriormente. Esta variante se utiliza usualmente en la optimización de aproximadores universales, tal como se verá más adelante.

Finalmente, una vez se encuentra la función Q^* óptima, es posible extraer una política óptima observando que la política óptima es aquella que entrega el mejor valor dada la función aprendida. Es decir,

$$\pi^*(a|s) = \begin{cases} 1 & \text{si } a = \arg\max_{a \in \mathcal{A}} Q^*(s,a) \\ 0 & \text{sino} \end{cases} \tag{3.7}$$

Este método simple pero básico puede ser aplicado directamente para espacios discretos, en cuyo caso se le denomina tabular Q-learning. Sin embargo, presenta dificultades para resolver problemas en espacios de estado continuos. Si bien un espacio continuo se puede discretizar, en algunos escenarios esto no es factible debido a que la cardinalidad del espacio que resultaría de la discretización sería muy grande. Es por esta razón que surge la necesidad de utilizar aproximadores para la función Q. Aunque los aproximadores lineales pueden ser utilizados, otra opción atractiva son los aproximadores no lineales basados en redes neuronales. Gracias a su capacidad como aproximadores universales, las redes neuronales proporcionan la expresividad requerida para abordar una gran variedad de problemas.

En el trabajo pionero de Mnih et al. en [31], se utiliza esta técnica en conjunto con otras mejoras como por ejemplo la introducción de un *replay memory* para de-correlacionar las transiciones utilizadas en la actualización de la Ecuación (3.6). Estas innovaciones permitieron mejorar la estabilidad del entrenamiento y alzaron las redes neuronales como una alternativa viable para la representación de la función Q en problemas más complejos con espacios de estado continuos. En concreto, dada una función Q parametrizada a través de una red neuronal de parámetros θ, se puede definir el funcional de costo de la siguiente manera,

$$L(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}[y_i - Q_{\theta_i}(s,a))^2] \tag{3.8}$$

donde $y_i = r + \gamma \max_{a'} Q_{\theta_{i-1}}(s', a')$ es el objetivo o TD-target para la iteración i-ésima y $\mathcal{D}$ corresponde al replay buffer mencionado anteriormente. Nótese que para el cálculo del target se utilizan los parámetros de una iteración previa para estabilizar el aprendizaje. Dicho de otra forma, se congelan los parámetros por cierta cantidad de pasos y se usan para entrenar la función Q de la próxima iteración de tal forma de que el target no cambie mucho durante un período breve de entrenamiento. En este punto cabe aclarar que las iteraciones refieren a las actualizaciones y no al entrenamiento. Luego, una vez la nueva función Q es mejorada, se vuelven a congelar los parámetros y se repite el proceso en una nueva iteración. Esto implica la necesidad de tener dos versiones de la función, una que está en otra entrenamiento y otra que se utiliza para el cómputo de y_i, usualmente a esta última función se le denomina target. Finalmente, gracias a que las parametrizaciones

son diferenciables es posible ocupar el método de descenso de gradiente para optimizar el funcional de la Ecuación (3.8). A la aplicación de todas las técnicas anteriores en el contexto de Q-learning se le denomina Deep Q-Networks (DQN).

Una limitación importante de DQN es que solo es aplicable a espacios de acciones discretos. La razón detrás de esta limitación es que con el método planteado no es posible calcular el máximo de la función Q para el cálculo del target pues dado que las acciones son continuas se requeriría una evaluación extensiva solo para una actualización, lo que es impráctico. Esta es la motivación principal detrás del método *Deep Deterministic Policy Gradient* o DDPG [27], el cual introduce una forma alternativa de calcular el máximo de la función Q.

Al igual que en DQN, en DDPG se utiliza una red neuronal para parametrizar la función Q y con eso es posible lograr que esta función sea diferenciable respecto a las acciones. Por tanto, DDPG plantea aproximar el máximo usando una función μ de parámetros ϕ, es decir, $\max_a Q(s, a) \approx Q(s, \mu_\phi(s))$. Así, el funcional de costo de la Ecuación (3.8) sigue siendo válido y se reemplaza la búsqueda del máximo con el cálculo de $\mu_\phi(s)$.

Dicho lo anterior, el problema reside ahora en definir la función μ_ϕ. Este método plantea una salida muy práctica: dado que el objetivo es el mismo que debe cumplir la política, es decir, maximizar la función de valor Q, se establece μ como la política. Por tanto, a diferencia de DQN, DDPG plantea una política explícita y parametrizada mientras que DQN posee una política implícita que se deriva utilizando la función de valor Q. Es por esta razón que a DDPG se le denomina un método del tipo actor-crítico o actor-critic, donde el crítico corresponde a la función de valor Q y el actor a la política. Al igual que con el caso de la función Q, en DDPG se utilizan dos versiones de la política, una en entrenamiento y otra que se denomina target. Esto con la misma motivación de estabilizar el target y con eso el entrenamiento.

Si bien DDPG es un método precursor muy importante dentro de *Reinforcement Learning*, existen muchos otros, con diversas variantes en la forma de realizar el entrenamiento, definir la política, entre otros. Recientemente se ha acuñado el término *Deep Reinforcement Learning* (DRL) para denotar a todos estos métodos similares a DQN o DDPG que se basan fuertemente en el uso de redes neuronales profundas para parametrizar sus componentes. El uso de DRL está en alza debido a su capacidad para abordar problemas complejos como la robótica, la conducción autónoma, los juegos complejos y la optimización de procesos empresariales.

En la próxima sección se describe el método *Twin Delayed DDPG* o TD3 [13]. Este método es utilizado en profundidad en experimentos posteriores realizados en este trabajo por ende se realiza una descripción más profunda de su funcionamiento.

3.1.1. Algoritmo TD3

Twin Delayed DDPG o TD3 [13] es un algoritmo de *Deep Reinforcement Learning* actor-crítico off-policy que opera en espacios de acción continuos. En concreto, TD3 es una evolución del método *Deep Deterministic Policy Gradient* [27] y emplea diversas técnicas para mejorar falencias de este tipo de métodos. Por ejemplo, uno de los principales modos de fallo de DDPG es su sobre-estimación de valores. Debido a que la política se extrae maximizando la función de valores Q, cualquier error de sobre-estimación que tenga dentro del soporte del espacio de acciones derivará en políticas que escogen acciones que supuestamente entregan gran valor pero no es así. A continuación se describen diversas técnicas que usa TD3 para aliviar este problema, entre las que destacan el uso de más de una función de valor Q y el suavizado de las acciones durante el entrenamiento.

En primer lugar y tal como se mencionó anteriormente, este método aprende dos funciones Q, a diferencia de DDPG [27] que solo aprende una. Estas dos funciones, Q_{ϕ_1} y Q_{ϕ_2}, de parámetros ϕ_1 y ϕ_2, son aprendidas mediante minimización del error cuadrático medio de Bellman [42]. Concretamente, dado un buffer $\mathcal{D}$, las ecuaciones (3.9) y (3.10) muestran el funcional a minimizar.

$$\mathcal{L}(\phi_1, \mathcal{D}) = \mathbb{E}_{(s,a,s',r)\sim\mathcal{D}}\left[(Q_{\phi_1}(s,a) - y(r,s',d))^2\right] \tag{3.9}$$

$$\mathcal{L}(\phi_2, \mathcal{D}) = \mathbb{E}_{(s,a,s',r)\sim\mathcal{D}}\left[(Q_{\phi_2}(s,a) - y(r,s',d))^2\right] \tag{3.10}$$

donde (s, a, s', r) es una transición arbitraria muestrada del buffer $\mathcal{D}$ e y corresponde al objetivo o *target*, el cual representa el valor esperado futuro y se describe en (3.11).

$$y(r,s',d) = r + \gamma \min_{i=1,2} Q_{\phi_i,target}(s', a'(s')) \tag{3.11}$$

Note que para calcular el valor esperado futuro se está utilizando el mínimo del valor predicho por los mismos funcionales Q pero de una versión previa ($Q_{\phi_i,target}$). En este caso se utiliza el mínimo para ayudar a evitar problemas de estimación que podrían contener los funcionales. Adicionalmente, se utiliza una versión de una iteración previa pues al mantener la función Q fija por cierta cantidad de pasos se ayuda a estabilizar el target y por ende la estabilidad del entrenamiento. Otro detalle importante a notar es el cálculo de la acción en el target, el cual se describe en (3.12).

$$a'(s') = \text{clip}(\pi_{\theta,target}(s') + \text{clip}(\epsilon, -c, c), a_{\text{mín}}, a_{\text{máx}}) \tag{3.12}$$

$$\epsilon \sim \mathcal{N}(0, \sigma) \tag{3.13}$$

donde clip corresponde a la función que satura su entrada dado un mínimo o máximo, $a_{\text{mín}}$ y $a_{\text{máx}}$ corresponden a los valores máximos y mínimos de la dimensión de la acción correspondiente y $\pi_{\theta,target}(s')$ corresponde a la acción que predice la política objetivo o target, la cual proviene de la misma política de entrenamiento pero obtenida de una iteración anterior. A esta técnica se le denomina *target policy smoothing* y tiene como objetivo perturbar las acciones que predice la política de tal forma que el crítico no desarrolle máximos locales en acciones que puedan parecer óptimas durante el entrenamiento.

Finalmente, la política o actor π de parámetros θ es encontrada mediante maximización del valor esperado predicho por alguno de los críticos.

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{s\sim\mathcal{D}}[Q_{\phi_1}(s, \pi_\theta(s))] \tag{3.14}$$

Es importante precisar que, al igual que la mayoría de métodos de DRL, tanto la política como el crítico se parametrizan usando redes neuronales profundas debido a su buen rendimiento como funciones aproximadores universales. También cabe notar que para lograr la convergencia en este tipo de métodos se realizan actualizaciones secuenciales del crítico y luego de la política usando descenso de gradiente sobre los funcionales enunciados en (3.9), (3.10) y (3.14). Adicionalmente los autores de TD3 sugieren actualizar la política una vez por cada dos actualizaciones de los críticos para estabilizar el cálculo del target (3.11) y reducir posibles inestabilidades en el entrenamiento.

3.2. Aprendizaje reforzado offline

El problema que busca resolver offline RL pretende ser el mismo que el de RL pero reemplazando la interacción con el ambiente por un conjunto de transiciones previamente muestreado que se denota como $\mathcal{D} = \{(s_t^i, a_t^i, s_{t+1}^i, r_t^i)\}$. Teóricamente, se busca optimizar el mismo objetivo mostrado en la Ecuación (3.2). Cabe mencionar que las transiciones pueden ser generadas por una política arbitraria π_β, de tal forma que las tuplas $(s, a) \in \mathcal{D}$ son muestreadas según $s \sim d^{\pi_\beta}(s)$, $a \sim \pi_\beta(\cdot|s)$. Usualmente, este tipo de formulación se conoce como off-policy RL, sin embargo la diferencia radica en que en tal caso aún se requiere interacción adicional con el ambiente para agregar o modificar el conjunto de datos $\mathcal{D}$.

Una de las dificultades de este problema es que el agente no tiene acceso al ambiente, y por ende, no puede explorar. Esto implica que el aprendizaje debe darse solo a través de las transiciones recolectadas $\mathcal{D}$. Debido a esto, es crítico para que el aprendizaje sea factible, que $\mathcal{D}$ sea lo suficientemente grande y lo suficientemente variado, de tal forma de suplir la incapacidad de explorar del agente. Cabe notar que caracterizar las condiciones mínimas que debe tener este dataset para que un algoritmo de offline RL logre recuperar una política óptima es todavía un problema abierto [40].

Otra de las dificultades, y una de las más relevantes, es el *distributional shift*. Dado que el agente es entrenado bajo cierta distribución en sus datos y evaluado en una distribución distinta, al momento de tomar acciones en situaciones que el agente no conoce es posible que cometa errores. A este fenómeno se le conoce como distributional shift. En [39] se provee un análisis teórico sobre este problema. En resumen, muestran que cuando se entrena una política de manera completamente offline (en ese caso con aprendizaje supervisado), la cantidad de errores que comete la política en una trayectoria de largo H está acotada y la cota crece cuadráticamente con H. Así mismo, cuando se entrena la política de manera online, permitiéndole acceder a las acciones óptimas (DAgger [39]), esta cota crece linealmente con H. Esto da cuenta de la severidad de este problema, inclusive en el caso online, en el cual se cuenta con las correcciones necesarias.

Intuitivamente, considere que se evalúa una política ya entrenada mediante alguno de estos métodos y que posee una error de generalización igual a ϵ en cada decisión que toma la política. Al momento de la evaluación, es probable que la política entre en estados que no haya visto en entrenamiento. Una vez que eso pasa, esta cota de error ϵ ya no existe pues solo es válida bajo la distribución en que la política fue entrenada, en tal momento se dice que el agente toma una acción OOD (out-of-distribution). Dicho lo anterior, sin la cota ϵ es probable que la política se equivoque, tras lo cual, llegará a un estado probablemente distinto, y por ende seguirá equivocándose. Este mismo tipo de razonamiento se extiende a offline RL, con la diferencia de que la severidad del problema es mayor ya que en el caso online, dependiendo del algoritmo, es posible corregir el error cometido a lo largo de una trayectoria gracias a que es posible interactuar y recibir feedback sobre el resultado de sus acciones OOD. En la Figura 3.1 se retrata este problema e inclusive permite realizar una observación muy importante acerca de sus efectos: dado que se espera que la política tome acciones avaras (que entreguen el máximo retorno), al momento de evaluar la función Q, se encontrarán valores demasiado optimistas fuera del soporte de la distribución de los datos. Esta simple observación ha sido clave para determinar cómo los algoritmos de ORL deben abordar el problema de RL y qué dificultades deben superar.

Diversos tipos de algoritmos de offline RL que resuelven estas dificultades han sido propuestos en la literatura. En primer lugar, una corriente de algoritmos buscan estimar el retorno de la política

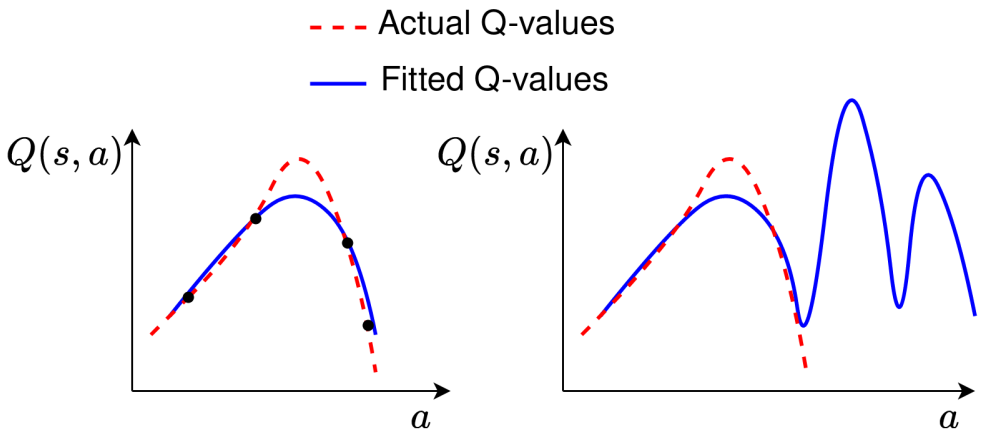

Figura 3.1: Izquierda: En rojo Q-value para un *estado fijo* en función de la acción inducida por la política que recolecta el dataset. En azul el Q-value que aprende la política dadas las muestras puntuales (puntos negros) de transiciones que tiene de la política que recolecta el dataset. **Derecha:** Mismo caso que en la figura izquierda pero utilizando la función $Q(s, a)$ en acciones lejanas a la distribución de los datos.

y a partir de esa evaluación seleccionar la política que mejor desempeño tiene. Por ejemplo, mediante *Importance Sampling* (IS) se puede estimar el retorno (3.2) y a partir del retorno deducir un gradiente [37] en la dirección que permita mejorar el retorno [46]. Alternativamente, es posible deducir directamente el gradiente usando también IS [6]. Desafortunadamente, el denominador común de estos métodos es que sus estimadores suelen sufrir de muy alta varianza, resultando en políticas de bajo rendimiento.

Una segunda corriente de algoritmos basados en programación dinámica busca aprender la función Q y derivar una política óptima a partir de este funcional [24], no obstante, suelen presentar notables problemas de distributional shift. Para mitigar esto, surgen métodos que tratan de mantener la política a aprender lo más cercana a la política que recolectó los datos. Esta noción de cercanía es implementada mediante medidas probabilísticas, como por ejemplo, la divergencia KL [14], [32]. Por otro lado, también es posible utilizar la incertidumbre epistémica del Q-value y observando que ésta incerteza será grande para acciones OOD, se puede optimizar buscando valores más conservadores en tal escenario [34]. Una alternativa a estos métodos que buscan restringir las políticas es *Conservative Q-Learning* [23] o CQL. En este método en particular se modifica el objetivo agregando un término de penalización de tal forma de que la función Q aprendida sea siempre menor que el verdadero valor Q y por ende se eviten las sobre-estimaciones que genera el distributional shift. Este tipo de modificación resulta en políticas que tienden a tomar acciones conservadoras, lo que explica el nombre del algoritmo.

Por otro lado, en online RL existe una clasificación natural de sus métodos dependiendo de si aprenden un modelo de transición del ambiente o no. Este mismo tipo de clasificación puede extenderse a ORL. En consecuencia, surgen algoritmos o métodos *model-based* y *model-free*, según si aprenden o no un modelo de transición, respectivamente. Por ejemplo, CQL es un método *model-free* pues no usa un modelo explícito del ambiente. El reciente trabajo de Figueiredo et al. [38] ejemplifica de excelente manera la taxonomía de los métodos offline RL y los diversos tipos existen. En la siguientes secciones se describen cuatro algoritmos *model-free*.

3.2.1. Algoritmo CQL

Tal como fue mencionado en la sección anterior, existe una variedad de métodos enfocados en resolver el problema de offline RL usando el mismo objetivo que RL pero proponiendo restricciones a las políticas para atenuar el *distributional shift*. Kumar et al., en [23], proponen un enfoque alternativo. En vez de restringir la política, se modifica el objetivo de la función Q a aprender y desde la cual se deriva la política. Este método tiene una fuerte inspiración en los métodos basados en incerteza pero no requiere estimar la incerteza explícitamente. A su vez, dado que solo afecta a la función Q, es aplicable a métodos basados en el aprendizaje de esta función y también en métodos tipo actor-crítico[1].

El objetivo que se utiliza para aprender la función Q_ϕ (parametrizada por ϕ) en métodos de online RL es el error esperado respecto a la optimalidad de Bellman dado un conjunto de transiciones $\mathcal{D}$ (caso off-policy), tal como se describe en la Ecuación (3.15) [2]. Cabe mencionar que ϕ' corresponde a los parámetros de la función Q objetivo congelada o target.

$$\mathcal{E}(\phi, \mathcal{D}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[(r(s,a) + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s')} Q_{\phi'}(s', a') - Q_\phi(s,a))^2\right] \tag{3.15}$$

Dado lo anterior, CQL propone agregar un término de penalización, generando un objetivo alternativo $\mathcal{E}'$.

$$\mathcal{E}'(\phi, \mathcal{D}) = \alpha \mathcal{C}(\phi, \mathcal{D}) + \mathcal{E}(\phi, \mathcal{D}) \tag{3.16}$$

El propósito de este término es reducir valores excesivamente grandes de Q_ϕ. Cabe notar que aún con este término de penalización, el término original $\mathcal{E}$ seguirá forzando la optimalidad de Bellman. Originalmente, en [23] se demuestra que escogiendo

$$\mathcal{C}(\phi, \mathcal{D}) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\mu(a|s)}[Q_\phi(s,a)] \tag{3.17}$$

$$\mu(s,a) = d^{\pi_\beta}(s)\mu(a|s) \tag{3.18}$$

con $\mu(a|s)$ cualquier distribución tal que $\mu \subset$ supp $\hat{\pi}_\beta$, donde supp $\hat{\pi}_\beta$ es el soporte de la distribución empírica que recogió los datos (es decir, la distribución condicional de las acciones en el dataset), se logran dos propiedades muy importantes [3]:

- La función $Q_\phi(s,a)$ aprendida al optimizar el objetivo en (3.16) es una cota del valor real de Q_π. Es decir, $Q_\phi(s,a) \leq Q_\pi(s,a)$. Siempre y cuando α sea escogido adecuadamente.
- A mayor cantidad de datos disponibles $|\mathcal{D}(s,a)|$, el valor teórico de α para garantizar la cota del punto anterior es menor.

Estos resultados permiten resolver directamente la sobre-estimación que provoca el *distributional shift*. El problema con la penalización detallada en las ecuaciones (3.17) y (3.18) es que la cota inducida está muy relajada, dicho de otra manera $|Q_\phi(s,a) - Q_\pi(s,a)|$ es muy grande. Por ende,

[1] Para más información sobre este tipo de métodos, refiérase a [42].

[2] Para más información sobre Optimalidad de Bellman y topología (e.g. métodos off-policy) de algoritmos RL refiérase a [42].

[3] Teorema 3.1 en [23]

dado que $Q_\phi(s,a)$ es una cota inferior, los valores que entrega la función Q_ϕ son muy pesimistas o conservadores. Tras la observación anterior, los autores proponen una cota más justa. Para tales efectos, demuestran que al escoger

$$\mathcal{C}(\phi,\mathcal{D}) = \mathbb{E}_{s\sim\mathcal{D},a\sim\mu(a|s)}[Q_\phi(s,a)] - \mathbb{E}_{(s,a)\sim\mathcal{D}}[Q_\phi(s,a)] \tag{3.19}$$

$$\mu(s,a) = d^{\pi_\beta}(s)\pi(a|s) \tag{3.20}$$

se logra una cota más justa que en el caso anterior. Intuitivamente, dadas las ecuaciones (3.19) y (3.20), lo que se busca es minimizar los valores de Q sobre acciones que la política está escogiendo (incentivar el pesimismo) mientras que maximiza los valores de Q dadas las acciones presentes en el dataset (incentivar las acciones en el dataset). Esto permite al método ser pesimista al tomar decisiones en situaciones que no ha visto en el conjunto de entrenamiento.

Cabe mencionar que las dos formas mostradas anteriormente son demostradas inicialmente de manera iterativa utilizando una forma fija de los funcionales estimados $\hat{V}(s,a)$ y $\hat{Q}(s,a)$. No obstante, en el apéndice D de [23] también se incluyen demostraciones cuando estas estimaciones son realizadas mediante funciones parametrizadas lineales y no-lineales.

Dados estos resultados teóricos, los autores proponen un esquema general al cual denominan como CQL. En este se combinan los hallazgos encontrados para proponer una penalización de la siguiente forma

$$\mathcal{C}(\phi,\mathcal{D}) = \mathbb{E}_{s\sim\mathcal{D},a\sim\mu(a|s)}[Q_\phi(s,a)] - \mathbb{E}_{(s,a)\sim\mathcal{D}}[Q_\phi(s,a)] \tag{3.21}$$

$$\mu(s,a) = d^{\pi_\beta}(s)\mu(a|s) \tag{3.22}$$

$$\mu(a|s) = \arg\max_{\hat{\mu}} \mathbb{E}_{s\sim\mathcal{D}}[\mathbb{E}_{a\sim\hat{\mu}(a|s)}[Q_\phi(s,a)] + \mathcal{R}(\hat{\mu})] \tag{3.23}$$

donde en (3.23) el término $\mathcal{R}(\hat{\mu})$ corresponde a un regularizador, cuya elección determinará una familia de variantes de CQL. Cabe notar que $\mu(a|s)$ es escogido de esta manera para que en promedio, a lo largo del dataset, en (3.21) se escojan acciones que maximicen el valor de Q y por ende se pueda resolver el problema de sobre-optimismo de las acciones OOD discutido anteriormente.

Una decisión simple y práctica para $\mathcal{R}(\hat{\mu})$ es $\mathcal{R}(\hat{\mu}) = \mathcal{H}(\hat{\mu})$. Es decir, se fija el regularizador como la entropía. Esto permite que la masa de la distribución μ no se puntualice en las acciones de valores erróneamente altos. Teóricamente, aún se tiene una cota bajo el valor real de la función Q_π con este regularizador. Al resolver el problema de optimización presentado en (3.23) con esta elección se encuentra que $\mu(a|s) \propto \exp(Q(s,a))$. Es decir, las acciones con valores más altos van a ser más penalizados y viceversa. Con lo anterior se logra una forma cerrada para la penalización $\mathcal{C}(\mathcal{D},\phi)$.

$$\mathcal{C}(\phi,\mathcal{D}) = \mathbb{E}_{s\sim\mathcal{D}}\left[\log\sum_a \exp(Q_\phi(s,a))\right] \tag{3.24}$$

Así mismo, distintos regularizadores $\mathcal{R}(\hat{\mu})$ pueden ser utilizados. Por ejemplo, se puede definir una distribución a priori y determinar la regularización como la divergencia KL respecto a esa distribución. Nótese que escoger el regularizador de entropía es equivalente a escoger la divergencia KL con una distribución a priori uniforme.

Una vez definido el esquema de aprendizaje para la función Q regularizada es posible obtener la política mediante *policy optimization* siguiendo la lógica general de algoritmos de policy iteration. En (3.25) se recupera la política dada la función Q.

$$\pi = \arg\max_{\hat{\pi}} \mathbb{E}_{s\sim D, a\sim\hat{\pi}(a|s)}[Q_\phi(s,a)] \tag{3.25}$$

3.2.2. Algoritmo IQL

Los principales enfoques para resolver el problema de *distributional shift* en la literatura se basan en o restringir la política para que no se desvíe demasiado de la política que recolectó los datos [12] o en su lugar regularizar la función de valor para asignar bajos valores a acciones que se encuentren fuera de la distribución de los datos (como por ejemplo CQL). *Implicit Q-Learning* (IQL) busca un enfoque distinto: evitar por completo el *distributional shift*, evitando por completo evaluar funciones de valor en acciones fuera de la distribución.

La idea principal de este trabajo se basa en reemplazar la función de valor por una estimación del expectil superior de una *distribución* sobre los valores respecto a las acciones del conjunto de datos para cada estado. Para conseguir esta estimación, se utiliza regresión de expectiles. De esta forma, durante entrenamiento, en vez de hacer una actualización de Bellman que requiere encontrar el máximo valor usando la última política (que podría dictar acciones fuera de la distribución de los datos), se utiliza un estadístico de una distribución que está condicionada a las acciones presentes en los datos, sin la necesidad de restricciones adicionales. Una vez que converge este modelo de la función de valor, se utiliza *Advantage-weighted Regression* [36] para extraer la política, la cual si requiere restricciones pero no influye en el entrenamiento de la función de valor.

La función base a optimizar por la mayoría de los métodos de offline RL se muestra en la Ecuación (3.26) y busca satisfacer la optimalidad de Bellman.

$$\mathcal{L}(\theta, \mathcal{D}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma \max_{a'} Q_{\hat{\theta}}(s',a') - Q_\theta(s,a)\right)^2\right] \tag{3.26}$$

Cabe notar que para estimar el máximo de la función de valor Q se suele utilizar la última política en entrenamiento, la cual se entrena para justamente maximizar la función Q. IQL en cambio propone el funcional de la Ecuación (3.27).

$$\mathcal{L}(\theta, \mathcal{D}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma \max_{\substack{a'\in\mathcal{A} \\ \text{s.t } \pi_\beta(a'|s')>0}} Q_{\hat{\theta}}(s',a') - Q_\theta(s,a)\right)^2\right] \tag{3.27}$$

Donde esta vez el máximo de la función de valor Q se obtiene solo en el soporte de las acciones en la distribución de los datos. En la práctica, para obtener este máximo se explota la estocacicidad del TD target respecto a las acciones a', es decir, en el término: $r + \gamma Q_{\hat{\theta}}(s', a')$. De esta manera, es posible introducir la regresión de expectiles como método alternativo para el entrenamiento de la función de valor.

La regresión de expectiles plantea que dada una variable aleatoria X, su expectil $\tau \in (0, 1)$ se puede encontrar como la solución al siguiente problema de optimización.

$$\arg\min_{m_\tau} \mathbb{E}_{x\sim X}[L_2^\tau(x - m_\tau)] \tag{3.28}$$

donde $L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$. Luego, por extensión, el expectil de una distribución condicional se puede encontrar como,

$$\arg\min_{m_\tau(x)} \mathbb{E}_{(x,y)\sim\mathcal{D}}[L_2^\tau(y - m_\tau(x)] \tag{3.29}$$

Al usar la formulación dada en (3.29) para reemplazar la evaluación de la política en la Ecuación (3.27), se obtiene la siguiente función objetivo:

$$\mathcal{L}(\theta, \mathcal{D}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[L_2^\tau(r(s,a) + \gamma Q_{\hat{\theta}}(s', a') - Q_\theta(s,a))\right] \tag{3.30}$$

Así, minimizando este objetivo perfectamente diferenciable se encuentra la función Q_θ, tal que el objetivo en la actualización de Bellman de la Ecuación (3.27) corresponde al expectil superior de la distribución inducida. El último detalle que toma en cuenta IQL es que el término asociado al TD-*target* en la Ecuación (3.30) contiene también la estocacicidad de la dinámica del ambiente representada por $s' \sim p(\cdot|s,a)$. Por ende, para eliminar este problema, se introduce una segunda función de valor V_ψ de parámetros ψ que aproxima el expectil buscado pero solo respecto a la distribución de las acciones.

$$L(\psi, \mathcal{D}) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(Q_{\hat{\theta}}(s,a) - V_\psi(s))] \tag{3.31}$$

Dado lo anterior, se usa esta función auxiliar para encontrar la función Q usando la actualización estándar de Bellman. Cabe notar que así toda la estocacicidad asociada a la transición de estados queda promediada en el cálculo de la esperanza.

$$\mathcal{L}(\theta, \mathcal{D}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2\right] \tag{3.32}$$

Hasta este punto IQL plantea solo un mecanismo para aprender una función de valor Q. Para extraer una política, se usa el método *advantage-weighted regression* (AWR) [36]. Este método busca extraer la política π de parámetros ϕ mediante la solución a un problema de máxima verosimilitud sobre las tuplas estado-acción recolectadas, pero incentivando exponencialmente aquellas acciones que brindan mayor valor. Justamente para determinar las acciones que entregan más valor se usan las funciones de valor aprendidas en los pasos anteriores. El objetivo para la extracción de la política usando AWR se describe en la Ecuación (3.33).

$$L_\pi(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\exp(\beta(Q_{\hat{\theta}}(s,a) - V_\psi(s)))\log\pi_\phi(a|s)]] \tag{3.33}$$

Finalmente, el algoritmo de IQL consta de dos etapas: Q-learning y *Policy Extraction*. En la etapa de Q-learning se minimiza mediante descenso de gradiente las ecuaciones (3.30) y (3.32) y en la etapa de policy extraction se extrae la política minimizando (3.33).

3.2.3. Algoritmo AWAC

La principal motivación detrás de los enfoques de offline RL es la posibilidad de aprovechar datos estáticos para obtener una política óptima o, en su defecto, proporcionar un punto de partida favorable para que los métodos de RL online puedan resolver el problema de manera más eficiente y/o rápida. El método *Advantage Weighted Actor-Critic* (AWAC) [32] se propone como una solución a esta problemática, ya que es capaz de explotar datos estáticos, aliviar el cambio en la distribución mediante restricciones implícitas y mejorar significativamente a través de entrenamiento online.

A grandes rasgos este método se basa en métodos de tipo actor-crítico (e.g., SAC [18]). Por tanto, iterativamente se entrena una función de valor Q usando TD-learning [42] y luego la política se aprende maximizando el valor de la función Q en entrenamiento. Dado un dataset $\mathcal{D}$, y una red neuronal aproximadora de parámetros θ, la función Q_θ se obtiene mediante minimización del funcional de costo mostrado en la Ecuación (3.34).

$$\mathcal{L}(\theta, \mathcal{D}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma \max_{a'} Q_{\hat{\theta}}(s', a') - Q_\theta(s,a)\right)^2\right] \tag{3.34}$$

Luego, dada una función Q objetivo, AWAC plantea encontrar la política π mediante optimización de la función de ventaja $A(s,a) = Q(s,a) - V(s)$. Nótese que optimizar la función de ventaja $A(s,a)$ es equivalente a optimizar la función $Q(s,a)$ pues tal optimización se realiza sobre el dominio de las acciones. La razón de utilizar la ventaja en lugar de únicamente la función de valor es que se reduce la varianza, lo que deriva en mayor estabilidad en el entrenamiento. Adicionalmente, se impone la restricción de la Ecuación (3.36) para evitar problemas de distributional shift, forzando implícitamente que la política aprendida no se aleje mucho de la distribución de los datos.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{a\sim\pi(\cdot|s)}[A(s,a)] \tag{3.35}$$

$$\text{s.t. } D_{\text{KL}}(\pi|\pi_\beta) \leq \epsilon \tag{3.36}$$

Donde π_β denota la política inducida por la distribución de los datos. Aplicando las condiciones de Karush-Kuhn-Tucker (KKT) para encontrar una solución analítica al problema en (3.35) se puede encontrar una solución no-paramétrica. Esta solución no-paramétrica puede ser proyectada usando las muestras de la política π_β (es decir los datos) a un espacio paramétrico minimizando la divergencia KL entre la distribución paramétrica (la política deseada) y la solución no-paramétrica. Tras lo anterior, se logra definir el siguiente problema de optimización para recuperar la política π_ϕ de parámetros ϕ:

$$\pi_\phi = \arg\max_{\phi} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\log\left(\pi_\phi(a|s)\right)\exp\left(\frac{1}{\beta}A(s,a)\right)\right] \tag{3.37}$$

De la Ecuación (3.37) se desprenden dos observaciones relevantes. En primer lugar, se observa que el problema a resolver se asemeja a uno de máxima verosimilitud de los datos, es decir, se busca maximizar la probabilidad de los datos observados. Sin embargo, en lugar de maximizar la probabilidad sin más, se utiliza un peso asignado a cada dato, el cual se refleja en el término denominado *advantage-weighted* en el nombre del algoritmo. Este peso es una medida de la ventaja de tomar una acción en un estado dado, y su objetivo es guiar la política a elegir acciones que proporcionen una mayor ventaja. Este término permite que la política se adapte de manera dinámica a los cambios en el entorno, ya que las acciones con mayor ventaja tendrán una mayor probabilidad de ser elegidas, logrando así aumentar la eficiencia del algoritmo, ya que se busca maximizar la probabilidad de los datos observados considerando la ventaja de las acciones.

Además, al no utilizar una restricción explícita sobre un modelo específico de π_β, se otorga mayor libertad a π_θ, lo cual resulta especialmente útil en el proceso de fine-tuning con RL online. Esto permite adaptar la política a los cambios en el entorno de manera más flexible y eficiente.

3.2.4. Algoritmo CRR

Desarrollado por Google, *Critic-regularized Regression* (CRR) [47] es un método de la familia de aprendizaje reforzado offline de tipo actor-crítico. Al igual que CQL, IQL y AWAC, este método

nace con el objetivo de resolver el problema de *distributional shift*. En este caso, la propuesta es bastante similar a los métodos anteriores: evitar el cálculo de la función de valor Q para tuplas estado-acción que no estén en el dataset de entrenamiento. Para lograrlo, se plantea una regularización en la optimización de la política, que busca incentivar acciones que estén en el dataset, pero filtrando por aquellas que tienen más valor según la función de valor entrenada.

Partiendo por el aprendizaje de la política, en CRR se propone resolver el siguiente problema de optimización.

$$\arg\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}}[f(Q_\theta, \pi, s, a)\log(\pi(a|s))] \tag{3.38}$$

donde $\mathcal{D}$ corresponde al dataset de entrenamiento, Q_θ a la función de valor de parámetros θ y π a la política buscada. Como es de notar, el objetivo es bastante simple y solo se restringe a que la función f sea no-negativa, escalar y monótonamente creciente respecto a Q_θ. Luego, las características de la política aprendida dependen de la elección de f. Por ejemplo, para el caso en que $f := 1$, la Ecuación (3.38) correspondería al mismo objetivo que Behavioral Cloning. Sin embargo, es necesario una mejor elección pues no necesariamente todas las transiciones contenidas en $\mathcal{D}$ son buenas. Para tal propósito es de interés ocupar la función de valor Q_θ, la cual en este método, a diferencia de otros, es estocástica y similar a lo propuesto en [4]. En [47] se proponen dos opciones para la implementación de este filtro:

$$f := \mathbb{1}[\hat{A}_\theta(s,a) > 0] \tag{3.39}$$

$$f := \exp(\hat{A}_\theta(s,a)/\beta) \tag{3.40}$$

donde $\hat{A}_\theta(s,a)$ corresponde a una estimación de la función de ventaja, similar a lo que propone AWAC en la Ecuación (3.35). En efecto, al utilizar la Ecuación (3.40) como filtro en la Ecuación (3.38), se recupera un problema similar al detallado en la Ecuación (3.37), y por tanto, se puede observar que esta implementación deriva de un problema de máxima verosimilitud restringido a la distribución de los datos usando la divergencia KL. Sin embargo, CRR propone diversas alternativas para la estimación de la función de ventaja. En las ecuaciones (3.41), (3.42) se describen las dos principales estimaciones de la función de ventaja. Otras estimaciones son estudiadas, similares a las de AWAC, pero se observa que tienen desventajas críticas que las hacen no funcionar correctamente en la práctica.

$$\hat{A}_{\text{mean}}(s_t,a_t) = Q_\theta(s_t,a_t) - \frac{1}{m}\sum_{j=1}^{m} Q_\theta(s_t,a^j),\ \text{donde } a^j \sim \pi(\cdot|s_t) \tag{3.41}$$

$$\hat{A}_{\text{max}}(s_t,a_t) = Q_\theta(s_t,a_t) - \max_{j=1..m} Q_\theta(s_t,a^j),\ \text{donde } a^j \sim \pi(\cdot|s_t) \tag{3.42}$$

Algunas observaciones relevantes sobre estas estimaciones es que para el caso en que m es pequeño, $\hat{A}_{\text{mean}}$ puede tomar valores demasiado optimistas debido a la naturaleza estocástica de la función Q_θ. Por otro lado, al usar el valor máximo de las acciones muestreadas en (3.42), $\hat{A}_{\text{max}}$ se vuelve una opción más conservadora.

Para el aprendizaje del crítico, no se proponen innovaciones importantes. El esquema de aprendizaje es similar al de los métodos estudiados anteriormente. Se minimiza el función de la Ecuación (3.43) de tal forma de satisfacer la optimalidad de Bellman.

$$\mathcal{L}(\theta,\mathcal{D}) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma \max_{a'} Q_{\hat{\theta}}(s',a') - Q_\theta(s,a)\right)^2\right] \tag{3.43}$$

Otro aspecto importante que plantea este método refiere al despliegue de la política. En concreto, se propone aprovechar la función Q durante la evaluación para mejorar la selección de acciones. Para lograr tal objetivo, en (3.44) se plantea un problema de optimización pero esta vez usando la política ya aprendida.

$$L(q, \pi) = \mathbb{E}_{s\sim\mathcal{D}}[\mathbb{E}_q[Q_\theta(s, a)] + \beta \mathrm{KL}[q(\cdot|s), \pi(\cdot|s)]] \tag{3.44}$$

donde KL denota la divergencia de Kullback-Leibler. La solución a este problema arroja que la política alternativa q toma una forma como la descrita en (3.45). En [47] se le denota a esta política como *Critic-Weighted Policy* (CWP).

$$q(a|s) \propto \exp\left(\frac{1}{\beta} Q_\theta(s, a)\right) \pi(a|s) \tag{3.45}$$

Luego, para muestrear una acción de q se utiliza *importance-sampling*. Es decir, se muestrean n acciones de $\pi(\cdot|s)$ y se le asignan pesos según $\exp(\frac{1}{\beta} Q_\theta(s, a))$. Finalmente, la acción seleccionada es la muestreada de la distribución discreta definida en (3.46).

$$P(a_i) = \frac{\exp(\frac{1}{\beta} Q_\theta(s, a_i))}{\sum_{j=1}^{n} \exp(\frac{1}{\beta} Q_\theta(s, a_j))} \tag{3.46}$$

En [47] se evalúa este método variando las elecciones de filtro y estimación de ventaja en ambientes de diversa complejidad y naturaleza. Tras lo anterior se encuentran varias observaciones, de las cuales destaca que utilizar el filtro de la Ecuación (3.40) suele dar mejores resultados en problemas complejos, mientras que el filtro de le Ecuación (3.39) destaca en problemas más simples. Por otro lado, utilizar la política CWP siempre da mejores resultados comparado a no utilizarla. En general, comparativamente a otros métodos, CRR se desempeña muy bien en ambientes cuyos espacios de observaciones tienen alta dimensionalidad. Por ende, resulta ser un método bastante competitivo dentro de offline RL.

A modo de resumen, a continuación se presenta una breve descripción y discusión de los cuatro métodos de offline RL presentados anteriormente: CQL, IQL, AWAC y CRR.

CQL (*Conservative Q-Learning*) es un algoritmo que aborda el problema de *distributional shift* al restringir la política aprendida para que se mantenga cerca de la política que extrajo los datos mientras aprende una función Q más conservadora. A pesar de su buen rendimiento en diversas tareas, presenta desafíos en el ajuste de hiperparámetros, los cuales si no son cuidadosamente ajustados, pueden determinar la falla completa del método.

Por otro lado, IQL (*Implicit Q-Learning*) es un novedoso método de offline RL que aprovecha la capacidad de generalización de las redes neuronales para estimar el valor de la mejor acción disponible en un estado dado, sin consultar directamente una función Q con acciones no vistas. Esto se logra aproximando un expectil superior de la distribución de valores con respecto a la distribución de acciones del conjunto de datos para cada estado. Las ventajas incluyen facilidad de implementación y rendimiento en comparativa al estado del arte. Sin embargo, se encuentra menos probado y validado debido a su novedad.

En contraste, AWAC (*Advantage Weighted Actor Critic*) es un algoritmo de RL tipo actor-crítico que combina eficientemente los beneficios del aprendizaje *off-policy* con la estabilidad de los

métodos de actor-crítico. Es versátil y aplicable tanto a entornos offline como online, demostrando un buen rendimiento en tareas de control continuo. Sin embargo, requiere entrenar tanto un actor como un crítico, lo que aumenta la complejidad del algoritmo y lo hace más sensible al ajuste de hiperparámetros.

Por último, CRR (*Critic Regularized Regression*) es un algoritmo tipo actor-crítico que mejora la estabilidad del aprendizaje al regularizar las actualizaciones de la política utilizando la función Q aprendida. Al manejar el problema del *distributional shift* mediante la regularización de los valores que predice, demuestra un buen rendimiento en una variedad de tareas de control continuo. Sin embargo, el rendimiento del método puede ser sensible a la elección de hiperparámetros, como el coeficiente de regularización, y requiere un entrenamiento separado del crítico, lo que aumenta la complejidad y el costo computacional en general.

Capítulo 4

Metodología

Este trabajo busca resolver el problema de conducción autónoma al interior de túneles usando offline RL, explotando datos de operación recolectados previamente, prescindiendo de los costos y riesgos inherentes de la conducción. Por otro lado, la utilización de offline RL se basa en que existe un dataset de transiciones previamente generado y que fue extraído de un proceso de decisión de Markov (MDP) dado. En esta sección se define en primer lugar un proceso de decisión de Markov en el contexto de aprendizaje reforzado. Este proceso de decisión es implementado en un ambiente de simulación que luego será utilizado para entrenar un agente de aprendizaje reforzado usando TD3. Mediante el entrenamiento y evaluación de este agente se cumplen tres propósitos: (i) validar que el diseño del proceso de decisión es correcto, (ii) extraer un conjunto de datos grande y diverso de la operación de este agente autónomo, (iii) establecer un antecedente o baseline del rendimiento que obtendría un agente autónomo entrenado con DRL bajo el proceso de decisión diseñado. Adicionalmente, una vez se tiene lo anterior, se extraen datos operacionales adicionales pero usando teleoperación humana, lo que permitirá establecer un conjunto de datos más pequeño pero cuyas trayectorias son más representativas de lo que se obtendría si es que se capturaran datos de teleoperación de un operador real de un LHD. Ya con estos conjuntos de datos preparados y caracterizados, se utilizan para el entrenamiento de agentes de aprendizaje reforzado offline, lo que finalmente permitirá dilucidar la factibilidad de aplicar este tipo de métodos en estas aplicaciones, cómo se compara el rendimiento con agentes de DRL que sí tienen acceso al ambiente, qué características son necesarias en los datos para lograr una navegación exitosa usando ORL, entre otros.

Dicho lo anterior, en esta sección se define el ambiente de simulación, el modelo de RL y la especificación de su proceso de decisión de Markov, la metodología de validación/evaluación, el proceso de entrenamiento y las estrategias utilizadas para la de extracción de datos.

4.1. Simulación

El ambiente minero al interior de un túnel es sumamente complejo. En él co-existen personas, maquinaria y equipos, muchos de ellos en movimiento y en constante cambio. Por esta razón, es deseable que el simulador a desarrollar capte la mayor cantidad de detalles posibles, de tal manera de reducir la brecha de éste con la realidad. El problema con esto es que a mayor fidelidad en la

simulación, mayor es el requerimiento en la capacidad de cómputo (lo que en casos puede llegar a ser restrictivo), por lo que se requerirá alcanzar un punto medio entre alta fidelidad y eficiencia.

A raíz de lo anterior, se escoge utilizar el motor de simulación Gazebo. Gazebo es un simulador de código abierto especializado en robótica y desarrollado por OpenRobotics. Años de desarrollo y madurez han catapultado a Gazebo como uno de los simuladores más ampliamente utilizado tanto en el ámbito académico como industrial. Cuenta con la capacidad de configurar casi todos los aspectos de la simulación, desde la física hasta los sensores. También, posee una variedad de *plugins* que permiten aumentar la complejidad de los ambientes según requiera el desarrollador. Otra característica muy importante es que cuenta con integración con ROS (*Robotic Operating System*) [29]. De esta manera, el código utilizado para el control de robots (usando ROS) en el simulador de Gazebo es directo de ocupar en un robot real.

La simulación del ambiente cuenta con tres componentes: la especificación del robot (LHD), la especificación del mundo en donde se desplegará el robot y la especificación del controlador del robot. Cada uno de estas componentes se detalla en las siguientes secciones.

4.1.1. Especificación del LHD

Gazebo utiliza el formato de especificación SDFormat[1] o SDF para describir casi todos los aspectos de la simulación. En concreto, el modelo del LHD estará dado en este formato. Para facilitar el desarrollo de esta especificación se utiliza el lenguaje de macros *Xacro*[2]. Este lenguaje permite crear la especificación final del LHD de manera dinámica especificando parámetros, encapsulando componentes, entre otros.

A grandes rasgos, el LHD se basa en cinco componentes: (i) la base, (ii) el frontis, (iii) el brazo, (iv) la pala, y (v) las ruedas (ver Figura 4.1). Estos componentes poseen masa, inercia, geometría y mallas de colisión bien definidas en su especificación. Todos estos componentes interactúan entre ellos mediante articulaciones o joints. Estas articulaciones pueden ser fijas o móviles. Un caso particular de este modelo es que dado el problema planteado no interesa el movimiento de la pala, por ende tanto el brazo como la pala están fijos, es decir, sus articulaciones no se pueden mover. No es este el caso para la articulación de las ruedas y la unión base-frontis. En cuanto a sensores, el LHD cuenta con un único sensor tipo LiDAR que realiza escaners de su ambiente a una frecuencia de 40 Hz, muestreando un total de 48 puntos equidistantes en el plano circular horizontal que captan. Este plano circular tiene una resolución radial de 1 cm y un rango de entre 0,1 a 30 metros.

Adicionalmente a la estructura de los componentes y su relación entre ellos, también se definen geometrías de colisión. Estas geometrías de colisión corresponden a mallas 3D que en contacto con otro elemento de la simulación arrojan una alerta. Estas alertas resultarán de particular utilidad para poder determinar el fin de un episodio producto de una colisión.

Por otro lado, se define un modelo de colisión de la máquina basado en su geometría. El objetivo de este modelo es lograr una abstracción lo más simple posible, pero que permita conocer los puntos más vulnerables de la máquina solo conociendo la ubicación del punto central o pivote de la máquina (ver Figura 4.2). El supuesto principal es que la máquina se puede modelar como dos

[1]Especificación de SDFormat: `http://sdformat.org/spec`.
[2]Especificación y descripción del lenguaje Xacro: `https://github.com/ros/xacro/wiki`.

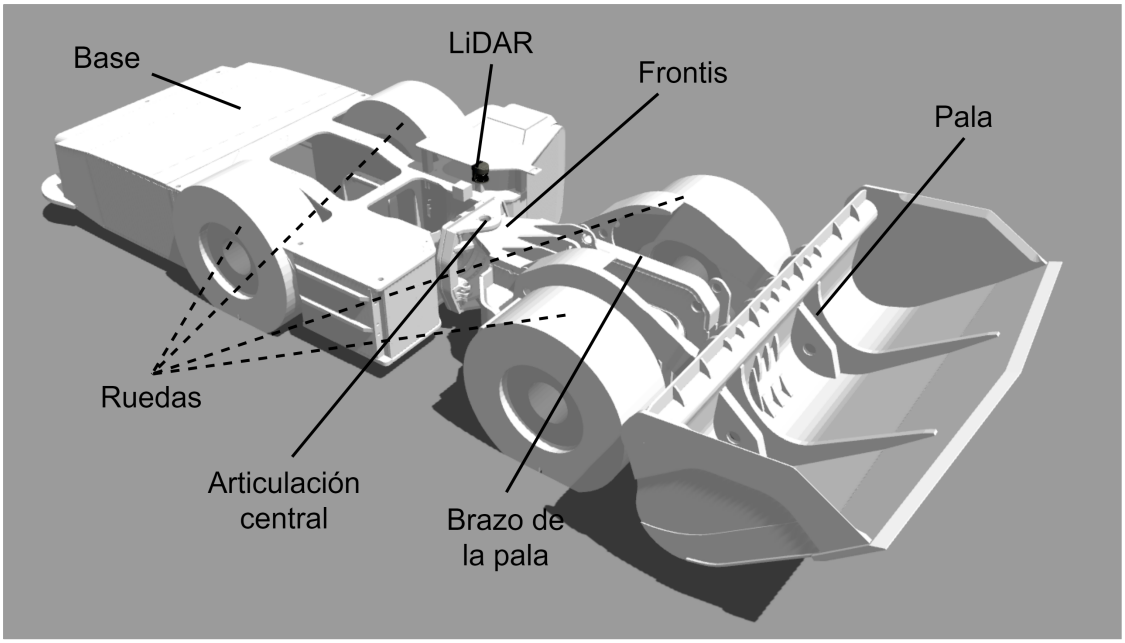

Figura 4.1: Descripción de la especificación del LHD y sus partes.

rectángulos conectados mediante un punto de pivote que representa la articulación central. Dicho lo anterior, dado un ángulo arbitrario de la articulación y la geometría de la máquina se pueden derivar 8 puntos que corresponden a las esquinas de estos rectángulos. Se utilizan estos puntos y no otros, porque corresponden a los puntos en los que es más probable que la máquina colisione. En las secciones posteriores se utilizará este modelo para poder ayudar al cálculo de la función de recompensa.

Por otro lado cabe notar que existen parámetros específicos de cada componente asociados a su física y su interacción con otros elementos de la simulación. Por ejemplo, estos parámetros controlan qué sucede cuando un componente del LHD choca con otro elemento de la simulación, cómo se da la fricción de las ruedas con el piso, cuales son las velocidades máximas de las articulaciones y cómo reaccionan ante estímulos externos, entre otros. El ajuste de estos parámetros se realizó mediante ensayo y error hasta alcanzar el comportamiento deseado.

4.1.2. Especificación del mundo

Al igual que la especificación del LHD, el mundo donde ocurrirá la simulación debe ser especificado (también en formato SDF). Para esto, se debe describir todos los modelos que estarán dentro (estáticos o dinámicos), las propiedades físicas del mundo, iluminación y visualización. En concreto, para esta simulación se busca representar el interior de túneles. Si bien es posible detallar modelos complejos de túneles que cuenten con texturas detalladas y mallas de colisión complejas, se buscó llegar a una representación mínima que capte lo más importante: la topología al interior de las minas.

Para esto, se diseño un mapa, etiquetado como *key*, que permite al robot enfrentarse a muros estrechos, virajes cerrados, virajes amplios y paredes irregulares. Este diseño se inspira fuertemente en la topología de una mina real pero dado que no es posible simular una mina subterránea entera,

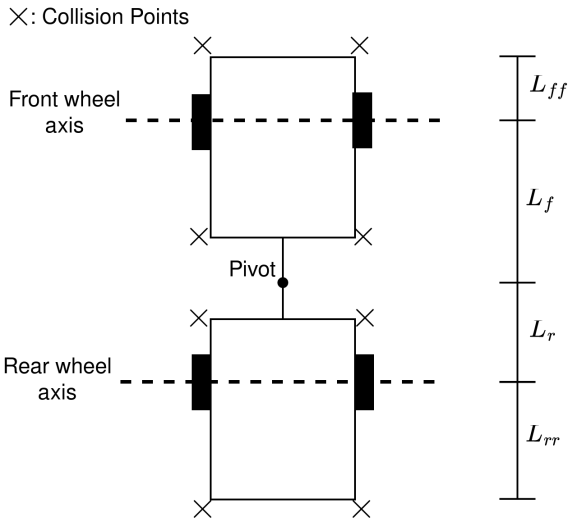

Figura 4.2: Modelo de colisión de la máquina. L_f corresponde a la distancia del pivote al eje del tren de ruedas delantero, L_{ff} a la distancia del eje del tren de ruedas delantero a la parte delantera de la máquina, L_r a la distancia del pivote al eje del tren de ruedas trasero y L_{rr} a la distancia del eje del tren de ruedas trasero a la parte trasera de la máquina. Se señala con 'X' los puntos de colisión del modelo.

se buscó preservar las características asociadas a los túneles y sus intersecciones. En la Figura 4.3 se expone una visualización de toda la topología simulada. La mayoría los experimentos se realizarán utilizando este mundo.

Por otro lado se implementó un segundo mapa, etiquetado como *ks* (ver Figura 4.4), y que es de carácter auxiliar pues servirá únicamente para tareas de evaluación. En la Figura 4.5 se tiene una vista isométrica de este mapa. Como se puede observar es mucho más grande que el mapa *key* y tiene más irregularidades en sus túneles. Dado que es más complejo, no se utiliza en entrenamiento por restricciones de tiempo de cómputo.

4.1.3. Especificación del controlador

Una vez se tiene la simulación del modelo del LHD en el mundo deseado, hace falta diseñar actuadores para el movimiento de la máquina. En este caso, existen dos componentes principales que requieren de control: las ruedas y la articulación base-frontis.

Para controlar el movimiento de las ruedas se utiliza un modelo cinemático que representa la dinámica de la máquina en la vida real y toma en cuenta factores como las revoluciones por minuto de las ruedas, su geometría y la velocidad objetivo. Este modelo también considera que las ruedas de la izquierda de la máquina giran a la misma velocidad angular, mismo caso para las ruedas de la derecha. En resumen, el funcionamiento de este controlador se basa en comparar la velocidad objetivo deseada con la velocidad actual de las ruedas, para determinar la velocidad angular deseada de las ruedas. Lo anterior dará origen a una señal de control que será ponderada dependiendo del ángulo y velocidad de la articulación central de la máquina. Finalmente, las velocidades angulares a aplicar a cada rueda son entregadas a los controladores PID de bajo nivel implementados en ROS para cada eje de las ruedas. Los controladores PID están implementados usando interfaces definidas en ROS y para efectos prácticos se utilizó un controlador ideal. Considérese como ideal

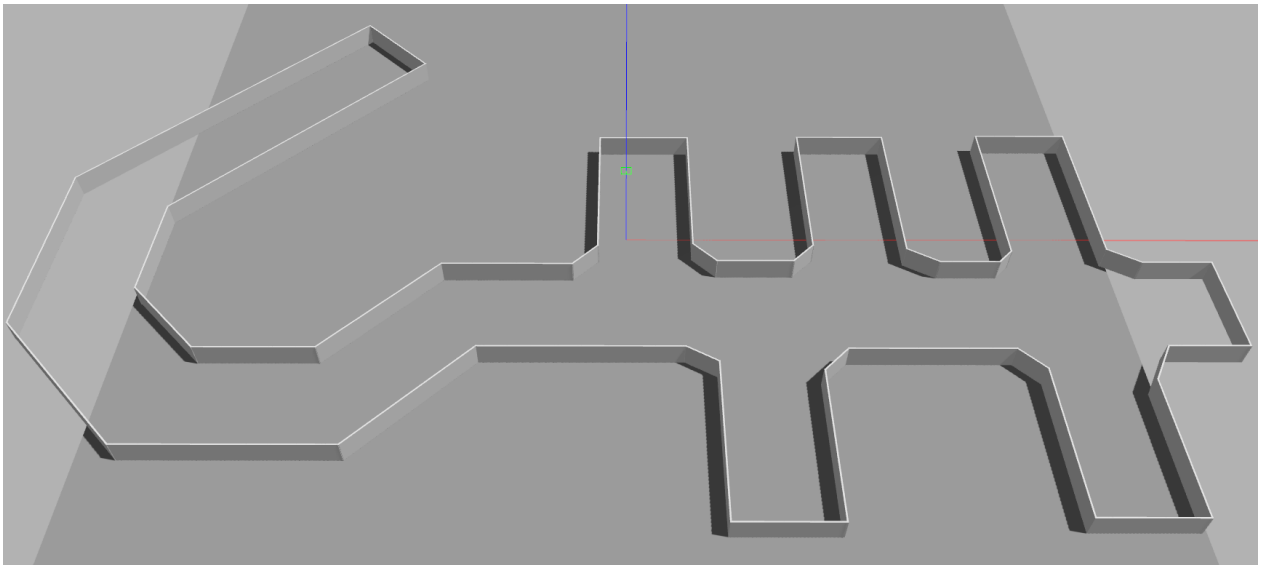

Figura 4.3: Visualización del mapa *Key*.

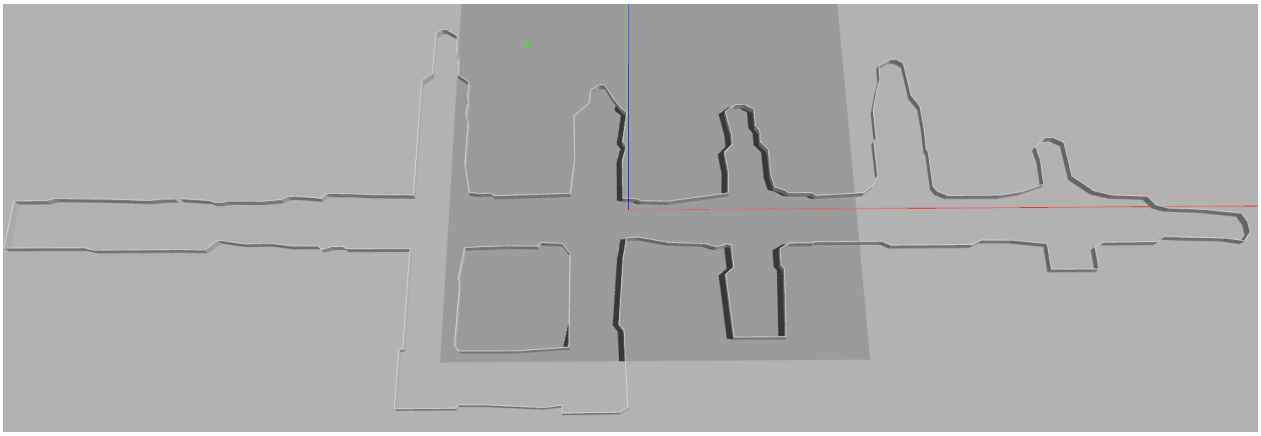

Figura 4.4: Visualización del mapa *KS*, vista completa.

aquel controlador que alcanza la señal objetivo en un tiempo muy pequeño. Cabe notar que si bien el controlador es capaz de alcanzar aceleraciones angulares muy grandes, esto no ocurre en la simulación pues el aumento o decremento de la velocidad angular se da en valores limitados y pequeños.

Contrario al caso de las ruedas, el control de la articulación base-frontis es mucho más sencillo. Ya que se trata de un movimiento que no debe ser coordinado, la velocidad angular se controla de manera independiente usando un controlador de velocidad ideal que provee ROS. Cabe notar que el ángulo de la articulación está limitado a un rango de -44 a 44 grados respecto al eje transversal de la máquina.

4.2. Modelo RL

En esta sección se describen el modelamiento así como diversos aspectos asociados al modelamiento del problema como uno de aprendizaje reforzado. Cabe recordar que este modelamiento

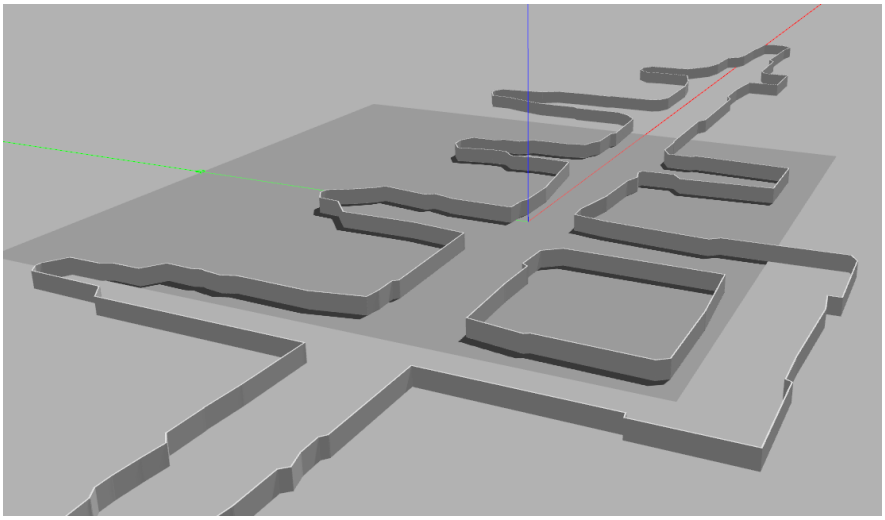

Figura 4.5: Visualización del mapa *KS*, vista isométrica.

será luego utilizado por un método de online RL (TD3) por diversos motivos. En primer lugar, mediante el uso de un método de online RL se permitirá confirmar la validez del modelo propuesto. En segundo lugar, se establecerá un baseline o punto comparativo que luego será de utilidad para el estudio de métodos de tipo offline RL. Finalmente, gracias al mismo entrenamiento será posible extraer conjuntos de datos que luego servirán para el entrenamiento de agentes usando offline RL.

4.2.1. PO-MDP

Se formula el problema como un proceso de decisión de Markov parcialmente observable (PO-MDP) episódico. Se define el PO-MDP como la tupla $(\Omega, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, donde Ω corresponde al espacio de observaciones, $\mathcal{A}$ corresponde al espacio de acciones, $\mathcal{P}$ corresponde a la matriz de transición de estados intrínseca del ambiente o dinámica del ambiente, $\mathcal{R}$ refiere a la función de recompensa que se describe más adelante en la sección 4.2.4 y γ a la tasa de descuento.

4.2.2. Observaciones

Las observaciones $o_t \in \Omega$ en el instante t para el episodio k tienen 58 dimensiones y se componen de:

- Distancia relativa al objetivo w_k en el eje x.
- Distancia relativa al objetivo w_k en el eje y.
- Magnitud de la velocidad en el instante t, o v_t.
- Magnitud de la velocidad en el instante $t-1$, o v_{t-1}.
- Coseno del ángulo hacia el objetivo, o $\cos \alpha_t$.
- Seno del ángulo hacia el objetivo, o $\sin \alpha_t$.
- Ángulo de la articulación de la máquina en el instante t, o θ_t.
- Ángulo de la articulación de la máquina en el instante $t-1$, o θ_{t-1}.
- Magnitud de la velocidad objetivo en el instante $t-1$, o $\hat{v}_t$.
- Velocidad angular objetivo de la articulación en el instante $t-1$, o $\hat{v}_{\theta_{t-1}}$.

- Rangos de lectura del LiDAR (48 componentes).

En lo que refiere al último componente, los rangos de lectura del LiDAR, es importante precisar que se obtienen a partir de mediciones equi-espaciadas angularmente en un plano circular paralelo al piso. Es decir, cada medición consecutiva de rango (distancia al punto más cercano en el ambiente) está separada por 7,5 grados.

Cada componente de la observación está normalizada al rango [-1, 1] utilizando una transformación lineal basada en sus valores máximos y mínimos exactos o estimados. Esta normalización es de suma importancia para la correcta convergencia de los métodos a estudiar más adelante.

Es importante mencionar que la elección de estas observaciones se realiza considerando dos aspectos. Primero, se busca que esta definición contenga la información necesaria para poder tomar buenas decisiones considerando la dinámica del ambiente. Por ende, dado que la máquina cambia su estado en el tiempo, se agregan componentes desfasadas temporalmente, con la intención de proveer información sobre el pasado y en consecuencia de su dinámica. En segundo lugar, se considera la pregunta, ¿qué información mínima necesito para llegar al destino sin colisionar?. La respuesta a esta pregunta es subjetiva y dependerá de todo el sistema empleado, pero a grandes rasgos, la definición dada permite conocer tanto el estado de la máquina, el estado del ambiente y su objetivo, lo que se podría considerar como la información mínima necesaria.

4.2.3. Acciones

Las acciones son bi-dimensionales y para el instante t se define la acción como la tupla $(\hat{v}_t, \hat{v_{\theta t}}) \in \mathcal{A}$ donde:

- $\hat{v}_t$ corresponde a la *magnitud de la velocidad objetivo* y se sitúa en el rango [-1, 1]. En la implementación esta velocidad es transformada linealmente al rango [0, $\hat{v}_{\text{max}}$] y entregada a un controlador de alto nivel que ejecuta el comando en bajo nivel. Por simplicidad, el espacio de acción está diseñado para permitir que la máquina vaya solo hacia adelante.
- $\hat{v_{\theta_t}}$ corresponde a la *velocidad angular objetivo* de la articulación de la máquina y se sitúa en el rango [-1, 1]. Al igual que la velocidad objetivo, esta cantidad es transformada linealmente al rango [-$\hat{v}_{\theta_{\text{max}}}$, $\hat{v}_{\theta_{\text{max}}}$] y entregada a un controlador de bajo nivel.

4.2.4. Función de recompensa

La función de recompensa diseñada en muestra en la Ecuación (4.1) y se compone de tres términos aditivos densos que buscan promover el avance hacia los objetivos locales (r_o), navegación alejada de los muros (r_w) y conducción suave (r_d). Adicionalmente, existen dos componentes de la recompensa, más dispersos (o *sparse*), que recompensan en caso de éxito (r_s) y penalizan en caso de colisión (r_c). Así, las ecuaciones (4.2), (4.4), (4.7), (4.8) y (4.9) contienen las expresiones de esta función de recompensa para cada observación o_t y acción a_t tomada en el instante t. Es importante mencionar que todos los hiper-parámetros utilizados en las diversas componentes fueron encontrados mediante prueba y error en experimentos preliminares.

Para cada instante t se define la recompensa $r(o_t, a_t)$ como una ponderación lineal de diversos componentes.

$$r(o_t, a_t) = w_d * (r_o(o_t, a_t) + r_w(o_t, a_t) + r_d(o_t, a_t)) + r_s(o_t, a_t) + r_c(o_t, a_t) \tag{4.1}$$

Donde w_d es cumple el rol de un ponderador para controlar el peso de las recompensas densas respecto a las recompensas dispersas (*sparse*). En concreto, $w_d = 1$.

A continuación se describe cada componente de la Ecuación 4.1.

Avance hacia el objetivo

$$r_o(o_t, a_t) = |v_t| \cos \alpha_t \tag{4.2}$$

Donde $|v_t|$ corresponde a la rapidez normalizada de la máquina (como referencia este valor se encuentra aproximadamente entre 0 y 1), α al ángulo entre la orientación de la máquina y el vector hacia el objetivo (ver Figura 4.6).

Navegación alejada de los muros

$$d_w = \min_{i \in \mathcal{C}, j \in \mathcal{Z}} ||p_{c_i} - p_{l_j}||_2 \tag{4.3}$$

$$r_w = -\mathbb{1}_{(d_w \leq \hat{d_w})} \cdot d_w \tag{4.4}$$

Donde $p_{c_i} \in \mathcal{C}$ corresponde al punto i-ésimo de los puntos de colisión $\mathcal{C}$ definidos según el modelo de colisión (ver Figura 4.2), $p_{l_j} \in \mathcal{Z}$ al punto j-ésimo de la nube de puntos $\mathcal{Z}$ obtenida por el LiDAR de la máquina. Intuitivamente, este término busca penalizar inversamente proporcional a la mínima de las distancias de los puntos más críticos de la máquina (sus esquinas) a un muro. La indicatriz se utiliza para penalizar solo cuando la distancia mínima es menor que un límite máximo previamente definido ($\hat{d_w}$). Para este trabajo, $\hat{d_w} = 1.2$ metros.

Conducción suave

$$K_t^1 = |\hat{v}_t - \hat{v}_{t-1}| \tag{4.5}$$

$$K_t^2 = |\hat{v_{\theta_t}}| \tag{4.6}$$

$$r_d(o_t, a_t) = -\mathbb{1}_{K_t^1 > \hat{K_t^1}} \cdot K_t^1 - \mathbb{1}_{K_t^2 > \hat{K_t^2}} \cdot K_t^2 \tag{4.7}$$

Donde los términos buscan K_t^1 y K_t^2 representan el aumento en rapidez de la máquina y la magnitud de la velocidad angular objetivo, respectivamente. Así, cuando estos términos son muy grandes, dependiendo de los límites establecidos por $\hat{K_t^1}$ y $\hat{K_t^2}$, se penaliza. Esto promueve que no hayan cambios bruscos de velocidad y que preferentemente no hayan velocidades angulares grandes en la articulación central, mejorando la suavidad en la conducción de la máquina. Los límites utilizados son $\hat{K_t^1} = \hat{K_t^2} = 0.3$.

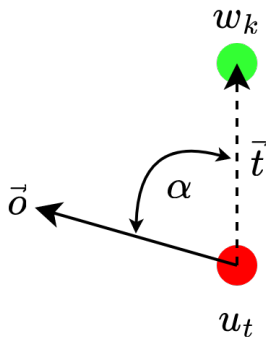

Figura 4.6: Cálculo del ángulo orientación hacia el objetivo (α). u_t corresponde a la ubicación de la máquina en el instante t y w_k a la ubicación del objetivo local del episodio k. En la práctica, este valor se calcula usando el producto punto entre la orientación de la máquina (o) y el vector que dirige la máquina hacia el objetivo local (t). Es decir, $\cos\alpha = \frac{\vec{o}\cdot\vec{t}}{|\vec{o}||\vec{t}|}$.

Alcance del objetivo

$$r_s(o_t, a_t) = \begin{cases} 100 & ||u_t - w_k||_2 < d_{\text{min}} \\ 0 & ||u_t - w_k||_2 \geq d_{\text{min}} \end{cases} \tag{4.8}$$

Donde u_t corresponde a la ubicación espacial de la máquina en el instante t y w_k al objetivo local del respectivo episodio k. Por otro lado, d_{min} refiere a la distancia euclidiana mínima para considerar un objetivo como alcanzado (tolerancia). Para todos los experimentos se utilizó $d_{\text{min}} = 4$ metros. Si bien esta tolerancia puede parecer grande, cabe recordar que la ubicación espacial u_t de la máquina es un punto ubicado en el centro de ésta.

Evasión de colisiones

Este componente busca evitar, mediante una penalización, que el agente colisione con su entorno. En la Ecuación 4.9 se describe este componente de manera condicional a la colisión. Cabe recalcar que la detección de las colisiones se realiza (en simulación) mediante sensores ubicados en toda la geometría del LHD.

$$r_c(o_t, a_t) = \begin{cases} -500 & \text{al colisionar} \\ 0 & \text{en otro caso} \end{cases} \tag{4.9}$$

4.2.5. Estados terminales

Las condiciones para terminar un episodio son cualquiera de las siguientes:

- **Objetivo local alcanzado**: Se considera un objetivo como alcanzado cuando la distancia del LHD al objetivo es menor que la tolerancia d_{min}.
- **Máxima cantidad de steps alcanzado (timeout)**: Se considera que se alcanza la máxima cantidad de steps cuando la longitud de una trayectoria supera un límite pre-establecido.

- **Colisión**: Se considera como colisión cuando se activa alguno de los sensores de colisión especificados en la máquina.

4.2.6. Exploración y explotación

Para controlar la exploración y la explotación a lo largo del entrenamiento se utiliza ruido tipo Ornstein-Uhlenbeck, o también llamado por sus siglas: "ou". Este tipo de ruido es especialmente útil para esta aplicación, pues al estar auto-correlacionado temporalmente posee propiedades interesantes como por ejemplo el *mean-reverting*. Esta propiedad permite que el proceso retorne a su media de forma natural. Por ejemplo, para el caso de navegación es importante que el ángulo de la articulación varíe para poder explorar el ambiente, pero también que no se aleje mucho de su media (ángulo cero) pues sino existiría una predominancia de episodios negativos. En la Ecuación (4.11) se define formalmente este proceso en su versión discreta.

$$x_0 = 0 \tag{4.10}$$

$$x_{t+1} = x_t + \theta(\mu - x_t)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1) \tag{4.11}$$

Donde $\theta = 1$, $\sigma > 0$, $\mu = 0$ y $\Delta t = 0.15$. Este proceso o ruido es aplicado de manera independiente a las dos componentes de la acción en entrenamiento. Luego, para controlar el balance entre exploración y explotación se decrece linealmente la escala σ hasta llegar a 0, el valor inicial, final y el momento en el que σ decrece hasta 0 está especificado en los hiperparámetros de la Tabla 5.2.

Otro tipo de ruido usualmente utilizado es el de tipo Gaussiano. Este tipo de ruido no es apropiado para este tipo de problema pues por su naturaleza es posible que ocurran cambios muy grandes en la velocidad angular objetivo de la articulación. Lo que se traduce en un comportamiento tipo zig-zag que luego es aprendido por la política.

4.3. Planificación y selección de objetivos

Uno de los supuestos de este trabajo es que en su operación, al interior de los túneles de una mina, se conoce la ubicación de la máquina por algún algoritmo de localización y también un objetivo global que debe alcanzar la máquina [3]. Este objetivo global puede estar muy lejos de la máquina, y a muchos túneles e intersecciones de distancia, por ende, se utilizan puntos intermedios que debe alcanzar la máquina.

Entonces, *el problema de navegación a un punto arbitrario al interior de una mina se puede descomponer como uno de navegación de varias misiones cortas de forma secuencial*. Cada una de estas misiones se define como un episodio en el contexto del PO-MDP definido.

Para poder simular los objetivos de cada misión se utiliza una representación gráfica de los túneles y su topología. Esta representación gráfica codifica semánticamente las posibles ubicaciones de un objetivo (ver Figura 4.8). A partir de esta representación se extraen todos los posibles puntos objetivo y adicionalmente los puntos posibles en los que puede aparecer la máquina. Cabe notar que para definir estos puntos se realiza una transformación del espacio coordenado de las imágenes al espacio coordenado de la simulación. En el caso del mapa *key* mostrado en la Figura 4.8, se dispone

[3] En [30] se propone un método robusto y eficiente para resolver el problema de localización así como el de planning a escala global, es decir, transversal a todo el interior de la mina.

de un total de 620 puntos de aparición predefinidos distintos, mientras que para los objetivos se cuenta con 5097 puntos diferentes. Es importante destacar que la gran cantidad de puntos conlleva ventajas y desventajas en cuanto a la combinación de puntos de inicio y final.

Por un lado, al haber muchas combinaciones, se incrementa significativamente la variedad de misiones a las que el agente puede enfrentarse, lo que contribuye a una mejor generalización del mismo. Por otro lado, debido a la gran cantidad de puntos, resulta difícil preestablecer todas las combinaciones válidas, lo que justifica el proceso de evaluación y selección de objetivos que se describe a continuación.

Dado que se tiene un conjunto de puntos objetivo y la máquina ya se encuentra en un cierto punto del mapa, para definir su misión o episodio se escoge un objetivo candidato de forma aleatoria y luego se construye una ruta óptima Γ hacia este objetivo. Esta ruta óptima corresponde a la ruta más corta y se encuentra usando el algoritmo Dijkstra [7]. La implementación de este algoritmo viene dada por ROS y solo se debió proveer una representación gráfica del mapa (muy similar al de la Figura 4.8). Finalmente, para ver si el objetivo candidato es seleccionable, se deben cumplir dos criterios:

- **Distancia**: El largo de la ruta Γ debe encontrarse en un rango pre-definido $[d^{\Gamma}_{\text{min}}, d^{\Gamma}_{\text{max}}]$. Para este largo se utiliza el hecho de que Γ se compone de una colección de puntos. Entonces, si la ruta Γ posee L puntos, denotados como Γ_i con $i \in \{1, ..., L\}$, el largo de la ruta, $|\Gamma|$, se calcula como,

$$|\Gamma| = \sum_{i=1}^{L-1} ||\Gamma_{i+1} - \Gamma_i||_2 \tag{4.12}$$

  Por tanto, el criterio de distancia establece que $d^{\Gamma}_{\text{min}} \leq |\Gamma| \leq d^{\Gamma}_{\text{max}}$.
- **Ángulo**: Este criterio busca que el inicio de la ruta sea factible. Para esto se calculan todos los ángulos definidos entre la orientación de la máquina y los vectores dirección hacia todos los puntos del primer quinto de la trayectoria. El cálculo se hace de la misma manera que se describe en la Figura 4.6, pero tomando como objetivo los primeros puntos de la trayectoria. Luego, si no existen diferencias mayores a 20 grados en estos ángulos (osea no se requieren giros bruscos) y el promedio es menor a 30 grados (el giro es suave), se dice que el objetivo es elegible.

Para ilustrar de mejor manera el proceso de selección de objetivos, en la Figura 4.7 se muestra una visualización para el caso de un objetivo elegible. Esta visualización permite apreciar la escala espacial en la selección de objetivos.

En resumen, para cada misión se selecciona un objetivo candidato y se calcula su ruta óptima. Luego, si tanto la ruta como el objetivo cumplen con los criterios de selección, se fija el candidato como objetivo y se da inicio a la misión o episodio. En caso contrario, se selecciona un nuevo candidato. Si no se encontrase ningún objetivo válido en un máximo de intentos, se re-ubica la máquina en otro punto aleatorio del mapa. Es importante mencionar que este procedimiento de planificación de objetivos se utiliza únicamente para entrenamiento y para las rutinas de evaluación. En un potencial despliegue de esta aplicación los objetivos deben ser planeados con una visión global, de tal forma de que el cumplimiento de cada misión de forma secuencial sirva para alcanzar un objetivo global.

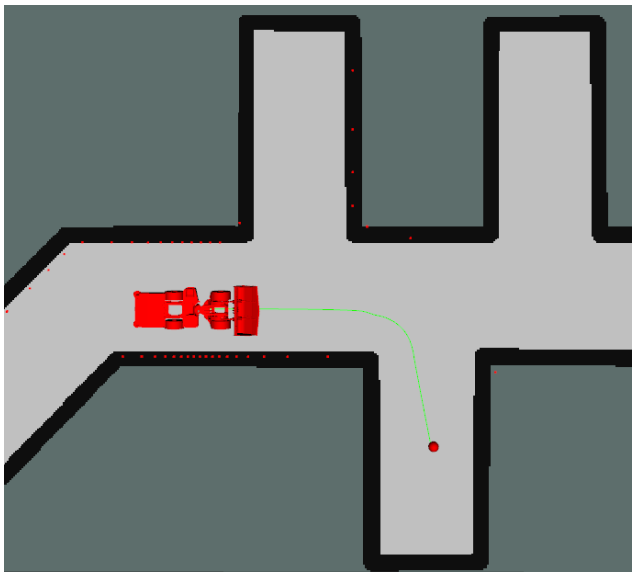

Figura 4.7: Visualización de un objetivo elegible (punto rojo) y la ruta utilizada para la evaluación de este objetivo (trayectoria verde).

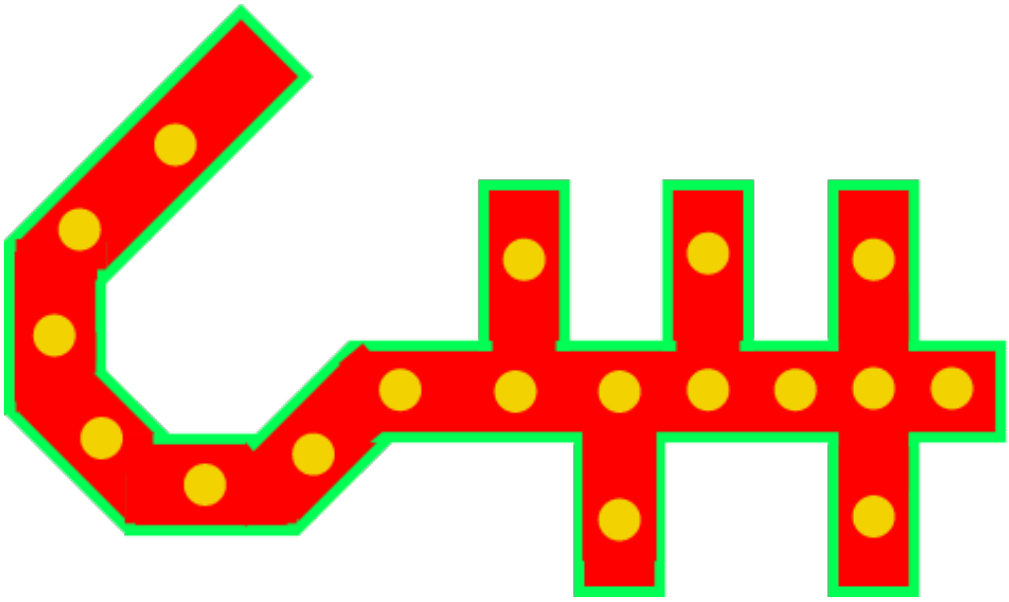

Figura 4.8: Representación auxiliar semántica del mapa. En rojo puntos posibles de aparición de la máquina. Los puntos posibles de aparición son previamente computados considerando las características geométricas del LHD y de los túneles, de tal forma de evitar colisiones iniciales con el ambiente. En verde los muros. En amarillo los posibles objetivos locales.

4.4. Evaluación

Una vez se tiene un agente o política entrenada usando RL, es de suma importancia poder evaluarlo de tal forma de poder validar que el modelo planteado esté correcto y la toma de decisiones del agente no presente complicaciones o comportamientos indeseados.

Para eso, la metodología de evaluación hace uso del mismo ambiente de entrenamiento pero esta vez usando las decisiones finales de la política. En total, se evalúa por un total de 100 episodios, desde los cuales se extraen las siguientes métricas.

- **Tasa de éxito**: Razón de episodios exitosos sobre episodios totales.
- **Cantidad de colisiones**: Cantidad de episodios en la cual ocurren colisiones.
- **Cantidad de fuera de tiempo o timeout**: Cantidad episodios en que ocurrió un timeout.
- **Rapidez promedio o average speed**: Promedio y desviación estándar de la rapidez promedio de los episodios. Calculada como el promedio de la magnitud euclidiana de la velocidad en todos los instantes de los episodios.
- **Steps**: Promedio y desviación estándar del largo de los episodios.

Estas cinco métricas son suficientes para poder caracterizar objetivamente el performance de una política en el cumplimiento de sus misiones. Sin embargo se incluyen adicionalmente capturas de videos durante la evaluación, las que permiten comparar y observar subjetivamente el rendimiento de las políticas.

A diferencia del caso de RL, en los planteamientos que involucran offline RL en una aplicación real no se cuenta con un ambiente en el cual evaluar. La razón de esto es que en estos casos la motivación de usar offline RL es precisamente lo prohibitivo de contar con un ambiente adecuado. Por tanto, la evaluación se debe realizar sin acceso al ambiente. Este es un problema abierto y actualmente existen diversos métodos que permiten realizar esto [35]. Sin embargo, dado que esto se escapa del alcance de este trabajo, se decidió utilizar las políticas entrenadas usando offline RL y evaluarlas en un ambiente de manera online. Así, es posible enfocarse en la caracterización de la solución al problema de navegación más que al problema de evaluación de políticas de manera offline.

No obstante lo anterior, existen algunas métricas que permiten dilucidar a grandes rasgos el performance de los métodos de offline RL y extraer conclusiones a partir de ellas. Cabe notar que estas métricas son evaluadas *durante* el entrenamiento en un conjunto de **validación** que contiene transiciones que son utilizadas solo para tal propósito.

- **Value scale estimation**

  Esta métrica refleja el valor promedio que entrega la función Q en el conjunto donde se evalúa $\mathcal{D}$. Cabe notar que para encontrar el máximo dado un estado s_t contenido en el dataset se ocupa la misma política π en entrenamiento. Por ende,

$$\mathbb{E}_{s_t \sim \mathcal{D}}[\max_a Q_\theta(s_t, a)] \tag{4.13}$$

- **Initial value**

  Esta métrica busca estimar el retorno promedio que obtendría una política π si, dado un

estado inicial s_0, se siguen las decisiones de la política hasta el estado terminal. Para esto simplemente se evalúa la función de valor Q en el estado inicial, dada la acción que tomaría la política en ese estado. Mientras más alto este valor se esperaría que la política entregue retornos mayores y por ende tenga un mejor comportamiento.

$$\mathbb{E}_{s_0 \sim \mathcal{D}}[Q(s_0, \pi(s_0))] \tag{4.14}$$

4.5. Extracción y estudio de datasets

Para la extracción de datos se utilizarán dos tipos de estrategias, las que a su vez definirán dos *tipos* de datasets. Cada dataset, acorde a lo descrito en el marco teórico, corresponde a un conjunto de transiciones $\mathcal{D} = \{(s_t, r_t, a_t, s_{t+1})\}_{t=1}^{n}$, donde n corresponde a la cantidad de transiciones. Cada transición representa en sí la decisión tomada por alguna política, lo que diferencia a las estrategias indicadas más adelante. Finalmente cabe recordar que para el caso particular de esta tesis se modela el proceso de decisión de Markov subyacente como uno parcialmente observable, por ende los estados s_t contenidos en los datasets corresponden realmente a las observaciones o_t descritas en la sección 4.2.

- **Replay**: Los métodos de tipo off-policy RL requieren un buffer para el almacenamiento de las transiciones encontradas durante el entrenamiento. Al final del entrenamiento es usual que este buffer se encuentre lleno y contenga las transiciones asociadas a las acciones tomadas por el agente en la última parte del entrenamiento. Este buffer es totalmente apropiado para crear un dataset y de hecho, es una forma de generarlos que se ha utilizado bastante en ésta área de investigación ([40], [15], [11], [17]). Cabe notar que dependiendo de cuán grande sea el buffer, la longitud total del entrenamiento, y la configuración del ruido, este buffer contendrá episodios con transiciones más o menos ruidosas y más o menos exitosas. En concreto, para la extracción de este tipo de datos se hará uso del método TD3.
- **Teleoperación**: Otra fuente de datos interesante a utilizar es el de teleoperación de la máquina. Para esto se utilizará el mismo ambiente de simulación ocupado con RL pero esta vez las acciones serán dadas por un humano. El propósito de este dataset es poder emular lo que sería el proceso de recolección de datos de la operación de una máquina real. Para extraer este tipo de datos es fundamental que el humano utilice la misma interfaz que usan los agentes autónomos, de tal forma de poder realizar una comparación justa. Dado que la dimensión de control de la articulación del LHD en la interfaz de acción descrita en la sección 4.2.3 puede ser poco natural para un humano, se utiliza un controlador PID que alivia este problema. En el anexo B se describe en detalle la implementación de este controlador.

Para poder caracterizar los datasets generados y poder analizar el impacto en el performance de los agentes entrenados se utilizarán principalmente dos métricas.

- **Tamaño**: Cantidad de episodios almacenados en el dataset.
- **Calidad**: Retorno promedio de los episodios almacenados en el dataset.

Otras métricas adicionales como la desviación estándar, mínimo y máximo del retorno en el dataset también son incluidas con el objetivo de evidenciar la distribución del retorno en cada conjunto de datos.

Capítulo 5

Resultados

En las siguientes secciones se presentan los resultados de entrenamiento de diversos agentes en diferentes modalidades. Primero, en la sección 5.1 se muestran los resultados del entrenamiento de un agente usando aprendizaje reforzado profundo, concretamente el método TD3. Luego, en la sección 5.2 se presentan y describen diversos conjuntos de datos generados a partir del entrenamiento del agente de la sección previa y también de otras metodologías. Finalmente, en la sección 5.3 se muestran los resultados del entrenamiento de agentes mediante aprendizaje reforzado offline usando las diversas fuentes de datos descritas en la sección 5.2.

5.1. Deep Reinforcement Learning

Una de las cosas más importantes para poder entrenar agentes usando aprendizaje reforzado es contar con una interfaz adecuada. Por ende, para el entrenamiento del agente TD3 se implementó una interfaz tipo Gym, la cual posee los métodos principales utilizados en este estándar [5]: *step*, *reset* y *close*. Llámese a una instancia de esta interfaz conectada al simulador como *ambiente*. Para el caso particular de este trabajo, los métodos más importantes de esta interfaz realizan las siguientes acciones:

- **reset**: Detiene la máquina, ubica la pala a una posición neutral (centrada en ángulo 0), la mueve a un punto inicial aleatorio seleccionado desde el conjunto de puntos posibles y finalmente selecciona un objetivo siguiendo la metodología explicada en la sección 4.3.
- **step**: Envía la señal de acción definida según el modelo (ver sección 4.2.3) a los controladores (ver sección 4.1.3). En este método se realizan algunas transformaciones a las acciones del modelo para que se ajusten al rango aceptado por los controladores. Una vez enviada la acción, se obtiene la observación del estado del simulador, se evalúa la función de recompensa y se verifica si el ambiente se encuentra en un estado terminal o no. Esto se realiza conforme a lo establecido en el modelo PO-MDP definido en la sección 4.2.
- **close**: Limpia cualquier variable local requerida por el ambiente y ejecuta cualquier proceso necesario para finalizar correctamente la instancia del ambiente.

Como se mención en la sección 5.1, el *ambiente* está conectado al *simulador*. En concreto, el ambiente es una representación programada en Python mientras que el simulador corresponde

Tabla 5.1: Hiper-parámetros usados para la configuración del ambiente.

Hiper-parámetro	Valor
Distancia mínima de objetivo local	10 metros
Distancia máxima de objetivo local	40 metros
Tolerancia de alcance al objetivo	4 metros
Frecuencia de actualización de estados	20 Hz
Largo LHD (para modelos geométricos)	12.33 metros
Ancho LHD (para modelos geométricos)	4.63 metros
Cantidad de mediciones LiDAR	48
Rango mínimo LiDAR	10 cm
Rango máximo LiDAR	30 metros
Resolución LiDAR	1 cm
Tasa de actualización LiDAR	40 Hz

al motor de Gazebo, junto con componentes adicionales de ROS. La interfaz obtiene el estado del simulador y ejecuta acciones sobre éste de manera programática, de tal forma de simplificar el proceso de recreación de un episodio, tal como se describió anteriormente. Dependiendo del hardware que ejecute este software podrían alcanzarse diversas velocidades en la simulación. Todos los experimentos fueron ejecutados en un computador personal con una tarjeta de video GTX 1660 Ti y un procesador Intel(R) Core(TM) i5-9400F de 2.90GHz.

La configuración del ambiente incluye la definición de muchos parámetros, los que van desde la física del simulador en Gazebo hasta otros de más alto nivel como por ejemplo la resolución del planner de alto nivel descrito en la sección 4.3. Debido a lo extensivo de esta lista, en la Tabla 5.1 se compilaron los parámetros más importantes y relevantes para efectos del presente estudio.

En cuanto al entrenamiento del agente, se utilizó el ambiente descrito anteriormente y una implementación propia del algoritmo de entrenamiento TD3. Para parametrizar tanto el actor como los críticos se utilizaron redes neuronales simples (*Multi-Layer Perceptron*) de 3 capas con 758, 512 y 512 neuronas, cada una de ellas activadas por una función no-lineal tipo ReLU. Para el caso particular del actor se utiliza una función de salida tipo tangente hiperbólica pues el recorrido de esta función permite limitar el rango de las acciones al rango aceptado por el ambiente. El entrenamiento se realiza por un total de 1 millón de steps, utilizando un *buffer* con capacidad para 400 mil steps. Más detalles sobre los hiperparámetros utilizados para el entrenamiento de este agente se describen en la Tabla 5.2. Los resultados de este entrenamiento se muestran en la Figura 5.1, en los cuales se observa la evolución de la tasa de éxito y retorno calculado como el promedio móvil de los últimos 100 episodios durante el entrenamiento.

En primer lugar, gracias a la Figura 5.1 se observa que la convergencia es bastante rápida, al cabo de 400 mil pasos de entrenamiento la política ya alcanza una tasa de éxito cercana al 100 %. Si bien se podría haber extraído una política aceptable en ese punto, el entrenamiento se prolonga por 600 mil pasos adicionales. El propósito de esto es dar más tiempo a la política a poder maximizar el retorno asociado a recompensas densas. Son específicamente estas recompensas las que incentivan directamente un comportamiento más deseable de la máquina en términos de suavidad, seguridad, entre otros. Al final del entrenamiento, el retorno promedio se estabiliza lo que implica que no está ocurriendo un aprendizaje adicional y por otro lado la tasa de éxito alcanza valores cercanos al

Tabla 5.2: Hiper-parámetros usados para entrenamiento de agentes TD3. †: ver Sección 4.2.6. ‡: ver Ecuación (3.12). $\dot{+}$: ver Ecuación (3.13).

Hiper-parámetro	**Valor**
Pasos de entrenamiento	1,000,000
Tamaño de buffer	400,000
Neuronas por capa	768-512-512
Función de activación	ReLU
Tasa de aprendizaje actor	0.001
Tasa de aprendizaje crítico	0.003
Tamaño de batch	256
Número de críticos	2
Tasa de descuento	0.99
Máxima norma de gradiente	0.5
Polyak average	0.005
Tipo de ruido de exploración	Ornstein–Uhlenbeck (OU)
OU promedio (μ) †	0
OU inercia (θ) †	1
OU step (Δt) †	0.15
OU escala inicial (σ) †	0.4
OU escala final (σ) †	0
OU step escala final (% pasos totales) †	0.6
Policy noise (σ) ‡	0.2
Policy noise clip ($a_{\text{min}} = a_{\text{max}}$) $\dot{+}$	0.5

100 %.

Posterior al entrenamiento se evalúa la política entrenada usando la metodología de evaluación descrita en la sección 4.4. Los resultados se encuentran en la Tabla 5.6. En ellos se observa algo muy similar a lo que se tiene en entrenamiento. La tasa de éxito es muy buena (99.2 %) al evaluar la política entrenada usando TD3 en el mapa *key*, lo que implica que la política completó la mayoría de sus misiones con éxito, sin colisionar ni quedarse quieta. Por otro lado, la velocidad promedio de la máquina es aceptable. Si bien el promedio esconde los máximos y mínimos, en el despliegue se observa que la máquina no alcanza velocidades muy grandes que puedan llegar a ser riesgosas. Otro aspecto que revela los resultados de la Tabla 5.6 (segunda fila, TD3*) es que la capacidad de la política para generalizar es muy buena, pues inclusive al desplegarla en el mapa *ks*, que es totalmente distinto al mapa en la que fue entrenado, es capaz de navegar sin mayores inconvenientes.

5.2. Generación de datasets

Para extraer los datasets necesarios para offline RL, se utilizaron las dos estrategias descritas en la sección 4.5. En primer lugar, se guarda una serie de *buffers* provenientes del entrenamiento de agentes usando TD3. Para los distintos experimentos, los hiperparámetros de entrenamiento se mantuvieron constantes y solo se varía el largo del entrenamiento o el tamaño del buffer, de tal manera de obtener buffers representativos de distintos balances de exploración y explotación. Por

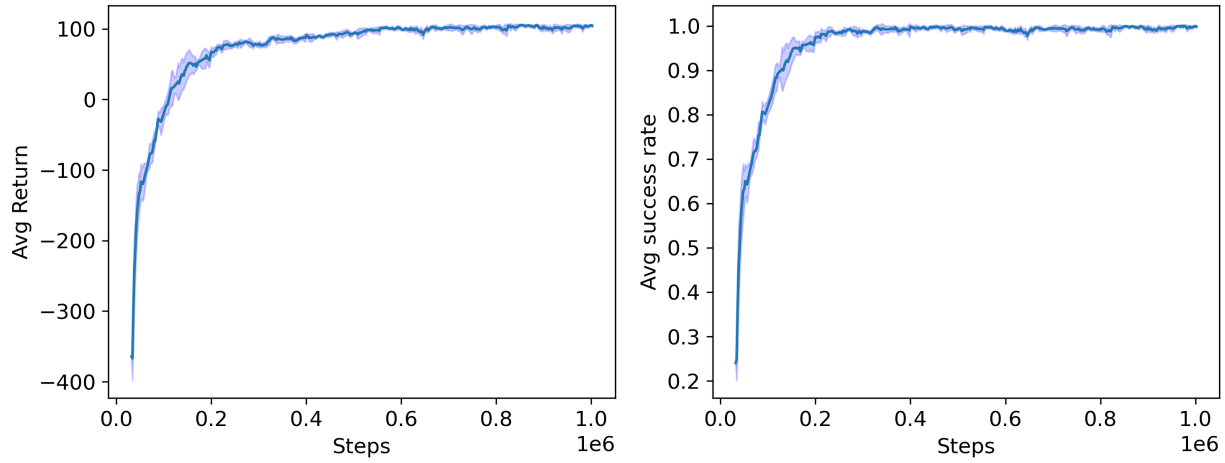

Figura 5.1: Promedio móvil (últimos 100 episodios) del retorno y tasa de éxito durante el entrenamiento utilizando TD3. Resultados promediados sobre 5 ejecuciones con semillas aleatorias.

ejemplo, dado un buffer de tamaño fijo, es esperable que la calidad de las trayectorias almacenadas sean mejores si se llena este buffer en un entrenamiento largo versus un entrenamiento corto, pues en el primero la política que se está entrenando tiene mucho más tiempo para interactuar con el ambiente y por ende sus últimas interacciones serán mejores. Dicho lo anterior, el primer replay buffer que se obtiene, es el etiquetado como *replay #1*, el cual corresponde a los últimos 400 mil pasos de entrenamiento mostrado en la Figura 5.1. Los replay buffers etiquetados como *replay #2*, *replay #3* y *replay #4* se extraen de experimentos de entrenamiento distintos, los cuales varían en longitud y tamaño del buffer. En la Figura 5.2 se muestra la tasa de éxito y el retorno promedio móvil a lo largo de estos entrenamientos. Cabe recordar que se utilizó TD3 con los mismos hiper-parámetros indicados en la Tabla 5.2, excepto por la cantidad de pasos de entrenamiento y tamaño del replay buffer. Como se puede apreciar, al extraer los datasets en momentos diferentes, las políticas que lo generaron exhiben retornos promedio diferentes, por ende, las trayectorias asociadas tienen diferentes retornos. Algunas métricas interesantes como la distribución del retorno de los episodios y la cantidad de estos se encuentran en la Tabla 5.3.

La segunda estrategia a utilizar, acorde a lo detallado en 4.5, corresponde a la teleoperación. En este caso, el autor del presente trabajo teleoperó la máquina usando un control externo. Más detalles sobre el proceso de teleoperación se encuentran en el anexo B. En general, el proceso de extracción es lento, para lograr un total de 1000 episodios, se requirieron aproximadamente 3 horas y 20 minutos continuos de operación. Los cuales fueron divididos en sesiones de 200 episodios, de tal forma de evitar imprecisiones del operador. Si bien es de esperar que la calidad de la operación sea al menos igual de buena que en el caso de un agente entrenado usando aprendizaje reforzado, no es natural controlar la máquina usando los controladores implementados, lo que redundó en degradación de algunas trayectorias.

Adicionalmente a los datasets descritos en la Tabla 5.3. Se derivaron datasets adicionales a partir de los anteriores de tal forma de obtener mayor diversidad en cuanto a cantidad y calidad. En primer lugar, todos los datasets descritos en la Tabla 5.4 corresponden al muestreo aleatorio de episodios del dataset denominado *replay #1*. Cada nuevo dataset se extrae de tal forma de corresponder en tamaño a una fracción del dataset original. Gracias al muestreo aleatorio se mantiene, en promedio,

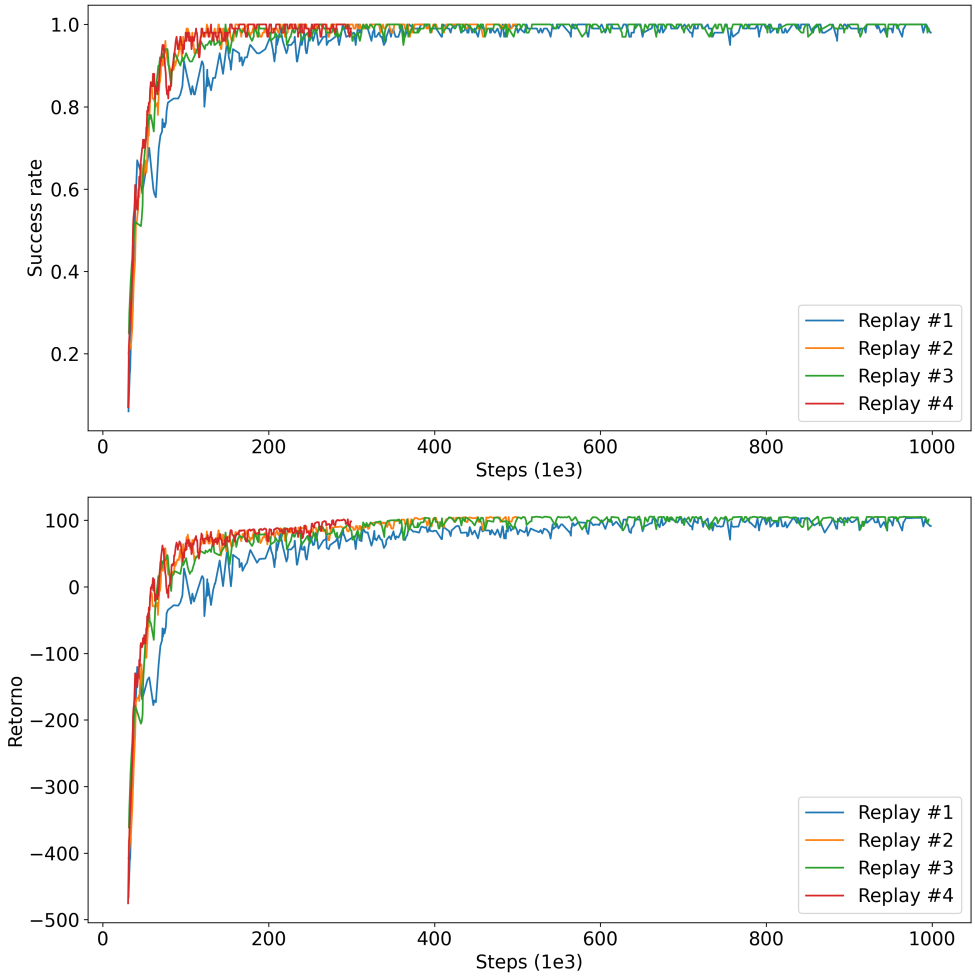

Figura 5.2: Promedio móvil (últimos 100 episodios) del retorno y tasa de éxito durante el entrenamiento utilizando TD3 para la extracción de diferentes datasets.

la calidad de las trayectorias que contiene cada dataset, de esta forma se logra aislar una única variable de interés: la cantidad de datos. Esto posibilita realizar diversos experimentos que buscan entender mejor el impacto de esta variable. Dicho lo anterior, en la Tabla 5.4 se evidencia que el retorno promedio de cada dataset es aproximadamente el mismo que el del dataset original, con desviaciones muy pequeñas.

El mismo procedimiento que fue ejecutado para derivar los datasets de la Tabla 5.4 se hizo para generar los datasets de la Tabla 5.5. La diferencia en este caso es que el dataset fuente corresponde al de teleoperación, etiquetado como *teleop* en la Tabla 5.3. Sin embargo, debido a que la cantidad de episodios es mucho menor, existen algunas variaciones importantes en la calidad de las trayectorias que contienen, lo que se ve reflejado en los retornos promedios.

5.3. Offline Reinforcement Learning

En primer lugar, con el propósito de buscar el mejor método de offline RL para el problema planteado se realizaron múltiples experimentos utilizando varios métodos: CQL [23], IQL [21],

Tabla 5.3: Descripción de datasets extraídos. Todos los buffers de tipo replay fueron extraídos del entrenamiento de agente usando TD3 (5.1). **Replay #1**: Últimos 400 mil pasos de un entrenamiento de 1 millón de steps. **Replay #2**: Últimos 400 mil pasos de un entrenamiento de 500 mil steps. **Replay #3**: Últimos 200 mil pasos de un entrenamiento de 1 millón de steps. **Replay #4**: Últimos 200 mil pasos de un entrenamiento de 300 mil steps. **Teleop**: 1000 episodios recolectados mediante teleoperación realizada por un humano.

Dataset	**# Episodios**	**Retorno**			
		avg	std	max	min
Replay #1	16776	96.61	59.85	116.06	-802.02
Replay #2	14922	88.65	58.12	117.43	-654.41
Replay #3	8496	101.02	45.68	118.01	-531.81
Replay #4	7189	83.07	61.35	115.91	-690.04
Teleop	1000	86.07	102.86	120.15	-534.77

Tabla 5.4: Descripción de datasets muestreados aleatoriamente del dataset Replay #1. **Replay #1 - X %** denota que el dataset posee una cantidad de X % de episodios respecto al dataset base.

Dataset	**# Episodios**	**Retorno**			
		avg	std	max	min
Replay #1 - 5 %	838	98.88	49.78	115.24	-774.12
Replay #1 - 10 %	1677	99.07	45.17	115.30	-560.33
Replay #1 - 15 %	2516	96.77	58.29	116.07	-520.07
Replay #1 - 20 %	3355	95.84	63.36	114.89	-674.35
Replay #1 - 25 %	4191	96.57	59.14	115.73	-565.01
Replay #1 - 50 %	8388	95.94	62.76	115.77	-774.12
Replay #1 - 75 %	12582	97.21	56.94	116.06	-774.12

AWAC [32] y CRR [47]. Para cada uno de ellos se hizo un ajuste de sus hiper-parámetros preliminar tomando como criterio la métrica de validación: initial value. En [35] se realiza un exhaustivo análisis de cómo esta estrategia de selección de hiper-parámetros es adecuada para métodos de offline RL. El resultado final de este proceso corresponde al conjunto de hiper-parámetros que se encuentra detallado en las tablas 5.7, 5.8, 5.9 y 5.10. Cabe destacar que todas las parametrizaciones, tanto de actores como críticos, son idénticas a las utilizadas para el agente TD3, es decir, son redes neuronales multicapa con la misma cantidad de neuronas y funciones de activación. Además, tanto para estos y todos los experimentos siguientes que implican el uso de algoritmos de offline RL se utilizó la excelente implementación que ofrece la librería **d3rlpy** [44], la cual contiene todos los algoritmos a estudiar.

En la Figura 5.3 se observa la evolución de algunas métricas de validación de los modelos de offline RL durante su entrenamiento. En concreto, se observa que en validación el promedio del valor inicial o initial value es pesimista o conservador para todos los métodos excepto IQL. Esto quiere decir que en promedio estos métodos tendrán bajos retornos en las trayectorias del conjunto evaluado. De la misma forma, la estimación del valor promedio complementa la observación anterior. Sin embargo, dado que esta métrica es un promedio del valor sobre todos los estados en el conjunto de validación, se observa un comportamiento mucho más suave durante el entrenamiento.

En la Tabla 5.6 se exhiben los resultados de la evaluación de las políticas ya entrenadas de la

Tabla 5.5: Descripción de datasets muestreados aleatoriamente del dataset Teleop. **Teleop - X %** denota que el dataset posee una cantidad de X % de episodios respecto al dataset base.

Dataset	# Episodios	Retorno			
		avg	std	max	min
Teleop - 5 %	50	85.46	119.27	120.15	-504.67
Teleop - 10 %	100	96.93	61.27	116.98	-498.97
Teleop - 15 %	150	88.10	99.32	120.15	-534.77
Teleop - 20 %	200	95.27	74.36	118.14	-506.35
Teleop - 25 %	250	98.82	55.31	119.78	-507.30
Teleop - 50 %	500	82.99	111.00	120.15	-510.08
Teleop - 75 %	750	89.59	93.81	120.09	-534.77

Tabla 5.6: Resultados de evaluación por 500 episodios en mapa *key* usando la última política entrenada mediante diferentes métodos de online y offline RL. Para el caso de los métodos de offline RL se utilizó el dataset *replay #1*, descrito en la Tabla 5.3. *: A diferencia de los otros resultados, la evaluación se realizó en el mapa *ks*.

		Success Rate	# Collision	# Timeouts	Average Return	Average speed	Steps
Online RL	**TD3**	0.992	4	0	100.80	3.13 ± 0.72	24.57 ± 5.95
	TD3*	0.996	2	0	104.92	3.28 ± 0.68	25.80 ± 5.68
Offline RL	**CQL**	0.878	61	0	31.20	2.96 ± 0.71	24.29 ± 6.06
	IQL	0.998	1	0	104.12	3.16 ± 0.66	26.31 ± 6.20
	AWAC	0.998	1	0	104.23	3.13 ± 0.64	27.58 ± 6.44
	CRR	0.996	2	0	99.85	3.17 ± 0.67	27.41 ± 6.60

Figura 5.3. La mayoría de los métodos alcanzan una tasa de éxito aceptable y superior al 99.6 % mientras que CQL falla en la tarea logrando solo un 87.8 % de éxito.

En la Tabla 5.11 se muestran los resultados de experimentos realizados con los métodos de offline RL propuestos pero esta vez siendo entrenados usando únicamente datos de teleoperación, es decir, el dataset *Teleop* de la Tabla 5.3. En este escenario, se observan diferencias significativas. En primer lugar, CQL alcanza un pésimo rendimiento, colisionando en promedio 211 veces cada 500 intentos, marcando una tasa de éxito muy baja igual al 57.7 %. En segundo lugar, CRR e IQL degradan su rendimiento bastante comparados a cuando se utiliza el dataset *replay #1*. Es decir, estos métodos no son muy robustos respecto a la cantidad de datos de entrada. Finalmente, AWAC logra los mejores resultados, alcanzando un 95.2 % de éxito. Adicionalmente cabe notar que de las veces que falla, aproximadamente un tercio es debido a timeouts, es decir, evita colisionar, lo que demuestra un mejor grado de seguridad respecto a los otros métodos. En el vídeo ubicado en `https://youtu.be/7AR9PUetEyg` se puede observar el comportamiento de AWAC y adicionalmente del agente TD3 descrito en la sección anterior.

El propósito de los análisis comparativos previos es comprobar empíricamente qué método entrega el mejor resultado dado el problema planteado y los datos recolectados. A vista de los resultados obtenidos, se escogen los métodos IQL, AWAC y CRR para la realización de los experimentos posteriores pues presentan los mejores rendimientos.

Tabla 5.7: Hiper-parámetros usados para entrenamiento de agentes IQL.

Hiper-parámetro	**Valor**
Número de épocas	50
Tamaño de batch	256
Neuronas por capa	768-512-512
Función de activación	ReLU
Tasa de aprendizaje actor	0.0003
Tasa de aprendizaje crítico	0.0003
Tasa de descuento	0.99
Número de críticos	2
Polyak average	0.005
Máximo valor de ventaja (3.33)	50
Temperatura inversa (β) (3.33)	10
Expectil (τ) (3.31)	0.9

Tabla 5.8: Hiper-parámetros usados para entrenamiento de agentes AWAC.

Hiper-parámetro	**Valor**
Número de épocas	50
Tamaño de batch	256
Neuronas por capa	768-512-512
Función de activación	ReLU
Tasa de aprendizaje actor	0.0003
Tasa de aprendizaje crítico	0.0003
Tasa de descuento	0.99
Número de críticos	2
Polyak average	0.005
Número de muestras acciones (3.37)	1
Temperatura (β) (3.37)	1

Después de la elección anterior, se llevaron a cabo más estudios utilizando los algoritmos IQL, AWAC y CRR, variando la cantidad y calidad de los datos disponibles. Para ello, se realizaron experimentos con los datasets de tipo replay descritos en la Tabla 5.3. La Figura 5.4 muestra la tasa de éxito de estos algoritmos. Se observa que todos los métodos obtuvieron buenos resultados, independientemente del conjunto de datos utilizado. En el único caso en que se observó una pequeña disminución en la tasa de éxito fue con el dataset *replay #4*, que se debe a que es más pequeño y tiene peores trayectorias. En general, los resultados indican que el rendimiento no se degrada significativamente al utilizar los datasets de tipo replay de la Tabla 5.3. Es posible que la cantidad de datos utilizados sea mucho mayor que el mínimo necesario para comenzar a observar una degradación significativa en el rendimiento, y que la calidad o diversidad de las trayectorias sea suficiente para aproximar una política óptima.

Adicionalmente, se realizaron más experimentos para estudiar el impacto de la cantidad de datos o episodios disponibles en el rendimiento de las políticas obtenidas usando IQL, CRR y AWAC. Primero, en la Tabla 5.12 se muestran los resultados de evaluación de la políticas entrenadas usando estos métodos y todos los datasets de la Tabla 5.4, es decir, datasets de diversos tamaños. En primer

Tabla 5.9: Hiper-parámetros usados para entrenamiento de agentes CQL.

Hiper-parámetro	Valor
Número de épocas	50
Tamaño de batch	256
Neuronas por capa	768-512-512
Función de activación	ReLU
Tasa de aprendizaje actor	0.0003
Tasa de aprendizaje crítico	0.0003
Tasa de aprendizaje α (3.16)	0.0001
Tasa de aprendizaje α (3.16)	0.0001
Tasa de descuento	0.99
Número de críticos	2
Polyak average	0.005
Número de muestras acciones	1
Valor inicial de α (CQL) (3.16)	1
Threshold de α (CQL)(3.16)	10
Valor inicial de α (SAC)	1

lugar, se observa que no existen degradaciones importantes en la tasa de éxito a lo largo de los diferentes datasets, en general la mayoría de los resultados tienen tasas de éxito mayores al 98 %. La diferencia más substancial que se observa en estos resultados es al usar el dataset *replay #1 - 5 %* o *replay #1 - 10 %* con los métodos IQL y CRR, donde se observa una degradación en la tasa de éxito de un 2 o 3 % respecto a los datasets de mayor tamaño. Por otro lado, dentro de los métodos evaluados destaca AWAC, el cual mantiene de forma consistente una tasa de éxito mayor al 98.4 % a lo largo de todos los datasets, lo que sugiere cierto robustez al conjunto de datos de entrada. Cabe mencionar que en la mayoría de los conjuntos todos los métodos alcanzan un retorno promedio mayor al presente en los datos, es decir, mejoran por sobre la política que extrajo los datos. Esta es una característica importante de los métodos de offline RL pues gracias a su capacidad crítica son capaces de generalizar de mejor manera.

En la Tabla 5.13 se presentan los resultados de experimentos similares a los de la Tabla 5.12, pero utilizando conjuntos de datos pequeños extraídos mediante teleoperación, como se muestra en la Tabla 5.5. En este caso, la escala en la cantidad de datos es mucho menor, variando entre 50 y 1000 episodios, en contraste con los experimentos de la Tabla 5.12, que variaban entre 838 y 12582 episodios. La observación inicial es que la mayoría de los métodos no logran superar el 90 % en la tasa de éxito. El peor método, CRR, alcanza solo el 63.4 % de éxito, siendo entrenado con 50 episodios, mientras que el mejor método, AWAC, logra el 91.8 % de éxito siendo entrenado solo con 150 episodios. En general, AWAC presenta los mejores resultados, lo cual es consistente con el análisis de la Tabla 5.12. Sin embargo, no se observa una tendencia clara en el rendimiento de la política en relación con la cantidad de datos utilizados.

En resumen, los resultados obtenidos al utilizar conjuntos de datos pequeños extraídos mediante teleoperación son inferiores a los obtenidos con los conjuntos de datos de tipo replay, lo que se debe a la menor cantidad y calidad de los datos disponibles en el conjunto *Teleop*. Además, los resultados sugieren que, al utilizar conjuntos de datos pequeños, el método de aprendizaje influye en el rendimiento de la política. En este caso, AWAC parece ser el método más adecuado para el

Tabla 5.10: Hiper-parámetros usados para entrenamiento de agentes CRR.

Hiper-parámetro	Valor
Número de épocas	50
Tamaño de batch	256
Neuronas por capa	768-512-512
Función de activación	ReLU
Tasa de aprendizaje actor	0.00003
Tasa de aprendizaje crítico	0.0003
Tasa de descuento	0.99
Número de críticos	1
Frecuencia de actualización de política	100
Número de muestras acciones (3.46)	4
Función de ventaja ((3.41), (3.42))	mean
Tipo de filtro ((3.38), (3.39))	exp

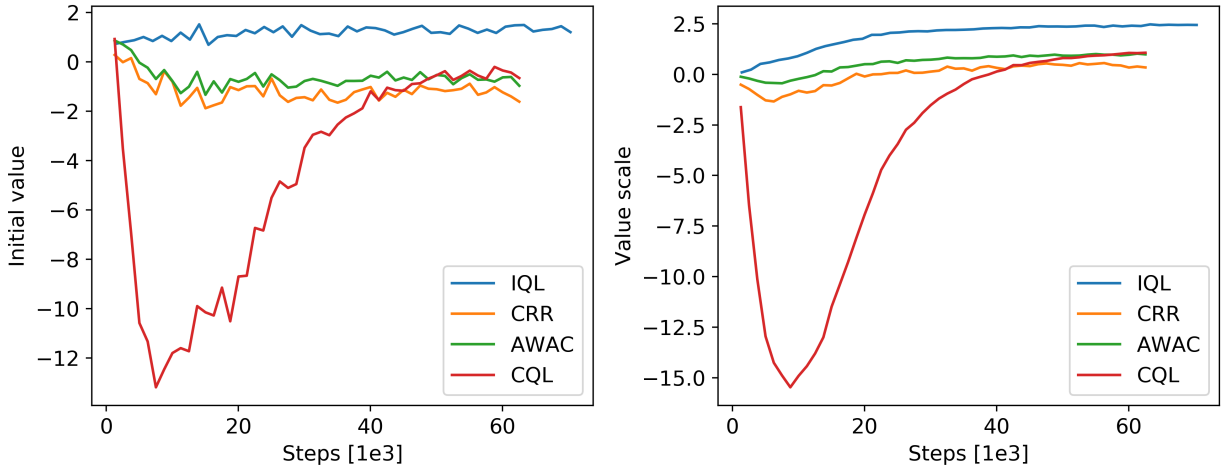

Figura 5.3: Initial value y value scale estimation en validación durante entrenamiento para varios métodos de offline RL usando dataset *replay #1*.

aprendizaje con conjuntos de datos pequeños extraídos mediante teleoperación.

Tabla 5.11: Resultados de evaluación por 500 episodios de último checkpoint entrenado usando diferentes algoritmos de ORL y dataset de teleoperación (teleop). Estos resultados están agregados sobre 5 semillas diferentes y aleatorias.

	Success Rate	# Collision	# Timeouts	Average Return	Average Speed	Average Steps
IQL	0.902 ± 0.043	45.4 ± 20.8	3.6 ± 1.9	46.87	3.08 ± 0.74	27.04 ± 15.89
CQL	0.577 ± 0.074	211.6 ± 36.8	0.0 ± 0.0	-153.45	2.13 ± 0.57	35.94 ± 16.23
AWAC	0.952 ± 0.008	16.6 ± 3.7	7.6 ± 1.9	82.34	3.00 ± 0.76	29.80 ± 22.57
CRR	0.909 ± 0.017	44.4 ± 9.2	1.2 ± 1.0	48.73	3.10 ± 0.73	26.66 ± 10.78

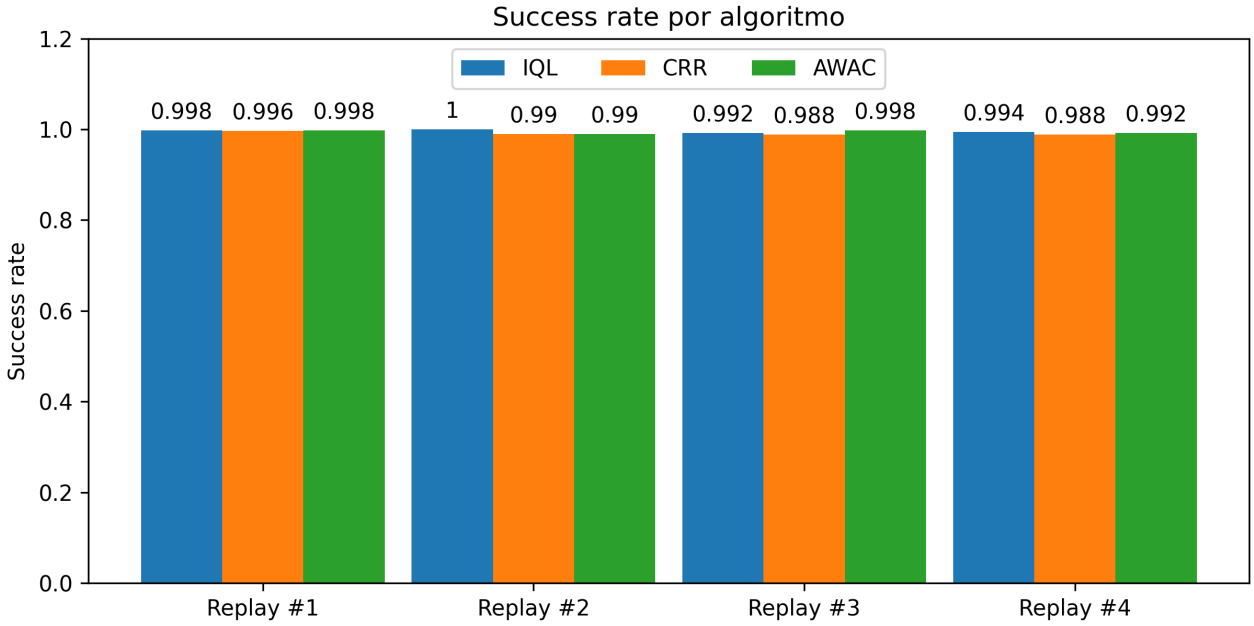

Figura 5.4: Success rate calculado sobre 500 episodios por algoritmo de ORL entrenado en varios tipos de datasets.

Tabla 5.12: Resultados de evaluación por 500 episodios de último checkpoint entrenado usando IQL, CRR o AWAC y datasets de diversos tamaños generados a partir del dataset Replay #1.

	Data	**IQL**		**CRR**		**AWAC**	
	Average Return	Succes Rate	Average Return	Succes Rate	Average Return	Succes Rate	Average Return
Replay #1 - 5 %	98.88	0.982	94.35	0.958	77.67	**0.994**	101.73
Replay #1 - 10 %	99.07	0.970	87.28	0.962	80.38	**0.984**	94.52
Replay #1 - 15 %	96.77	0.992	99.23	0.988	93.79	**0.998**	104.12
Replay #1 - 20 %	95.84	**0.998**	104.17	0.992	99.37	0.996	103.30
Replay #1 - 25 %	96.57	**0.996**	101.67	0.992	99.78	0.990	99.03
Replay #1 - 50 %	95.94	**0.998**	104.30	0.994	100.83	0.998	103.17
Replay #1 - 75 %	97.21	0.994	102.00	**0.996**	102.97	0.994	102.16

Tabla 5.13: Resultados de evaluación por 500 episodios de último checkpoint entrenado usando IQL, CRR o AWAC y datasets de diversos tamaños generados a partir del dataset *Teleop*.

	Data	**IQL**		**CRR**		**AWAC**	
	Average Return	Succes Rate	Average Return	Succes Rate	Average Return	Succes Rate	Average Return
Teleop - 5 %	85.46	0.828	-2.87	0.634	-121.10	**0.854**	17.71
Teleop - 10 %	96.93	**0.908**	47.80	0.658	-114.68	0.878	31.36
Teleop - 15 %	88.10	0.794	-26.10	0.774	-38.88	**0.918**	59.39
Teleop - 20 %	95.27	0.852	11.11	0.716	-75.71	**0.882**	33.96
Teleop - 25 %	98.82	0.868	23.54	0.780	-34.52	**0.888**	39.77
Teleop - 50 %	82.99	**0.914**	55.82	0.848	12.01	0.900	47.87
Teleop - 75 %	89.59	0.878	32.64	0.882	32.67	**0.910**	53.30

Capítulo 6

Discusión

Simulación

Dada las especificaciones del mundo, controladores y de la máquina misma se observa que el rendimiento de la simulación es bastante aceptable, logrando una aceleración promedio de 4.8 veces el tiempo de simulación normal. También, cabe destacar dos observaciones importantes sobre el rendimiento de la simulación. En primer lugar, dependiendo de la complejidad del mapa y de sus actores, la velocidad de la simulación puede degradarse seriamente. Como referencia, al utilizar el mapa KS en lugar del mapa Key, la aceleración de la simulación decae a 2.6 veces el tiempo de simulación normal. Esto puede resultar prohibitivo en caso de querer ejecutar entrenamientos largos. En segundo lugar, sorprendentemente el recurso más determinante en la velocidad de la simulación es el procesador del computador. Si bien muchas partes de la simulación se aceleran usando la tarjeta gráfica, para este caso, todo indica que las componentes más demandantes se calculan usado el procesador. Algunas alternativas viables para mejorar el rendimiento de la simulación son simplificar la malla de colisiones usadas en las diversas partes del modelo del LHD y/o utilizar un simulador diferente.

Modelamiento

Uno de los objetivos de los experimentos realizados con Online RL es validar el modelamiento del PO-MDP definido en la sección 4.2. Es posible afirmar que si es posible resolver el problema planteado con al menos un método de online RL, entonces el modelamiento es válido. La definición exacta de *resolver el problema* es subjetiva y puede proyectarse a algún criterio objetivo dependiendo del diseñador. En este caso, dado que se busca navegar hacia un objetivo sin colisionar, la tasa de éxito es un buen indicador para definir este criterio. Por consiguiente, analizando los resultados exhibidos en la Tabla 5.6, se puede afirmar que el modelamiento es válido pues el método TD3 es capaz de alcanzar un 100 % de éxito en evaluación. No obstante lo anterior, si bien esto da luces de que el modelo es válido, no necesariamente es óptimo. Su optimalidad vendrá dada únicamente por el comportamiento objetivo del diseñador.

Una dificultad que nace en el proceso de modelamiento es poder definir un criterio exacto para indicar cuándo el problema está resuelto en función de una métrica, dicho de otra forma: establecer una metodología de evaluación de agentes de navegación. Este sub-problema es un tópico abierto

en la comunidad pues tal como se mencionó anteriormente este tipo de definiciones es subjetiva. Si bien se han hecho esfuerzos para resolver este problema de forma transversal [9], para el caso de aplicación de este trabajo resulta importante avanzar en un marco de trabajo que se ajuste a los objetivos específicos de la industria. Las métricas de evaluación utilizadas en este trabajo son un primer acercamiento y sin lugar a duda sirven su propósito.

En lo que respecta al proceso de definición del modelamiento, diseñar la función de recompensa resultó ser la etapa más difícil, pues no existe una única función válida y siempre dependerá del contexto del problema. La función de recompensa descrita en (4.1) se logró tras muchos experimentos, los cuales a través de largos y costosos entrenamientos permitieron determinar qué formulación o hiper-parámetros ofrecía el mejor comportamiento. Este proceso de diseño podría acelerarse y/o mejorarse a través de métodos como *Inverse Reinforcement Learning* [2].

En cuanto a los otros componentes del modelo, se observó que ocurren degradaciones importantes en la estabilidad y velocidad de convergencia si las observaciones no están correctamente normalizadas. Por otro lado, en el espacio de acciones se evaluó controlar directamente el ángulo de la articulación de la máquina en lugar de controlar la velocidad de ésta. Diversos experimentos mostraron que controlar la articulación directamente deriva en comportamientos de tipo zig-zag pues es más fácil transferir el ruido inherente de los agentes en la señal de control de la máquina.

Otro punto interesante sobre el modelamiento es la percepción. Según lo establecido en 4.2, los agentes cuentan, a grandes rasgos, con dos tipos de información: del estado de la máquina y su objetivo, y las mediciones de un sensor LiDAR. Para disponer la información del LiDAR se utilizó una estrategia simple: usar las distancias medidas entre el sensor y el objetivo más cercano para distintos ángulos equi-espaciados en un plano circular paralelo al piso. Luego, en caso de que el objetivo más cercano esté muy lejos se satura cada una de estas distancia a un máximo de 50 metros. Esta formulación, similar a lo expuesto en [43], es suficiente para dar noción al agente del ambiente que lo rodea. Sin embargo, no se descarta la posibilidad de tratar esta información de una forma distinta, como por ejemplo una nube de puntos. Esta nube de puntos podría ser utilizada por un modelo más especializado [16], de tal forma de extraer mejores características y proveer de mejores bondades al sistema de percepción. No obstante lo anterior, el modelo establecido en este trabajo resulta rápido y efectivo.

Uno de los supuestos más fuertes en el modelamiento es la dirección de movimiento. Por simplicidad se restringe a que el movimiento sea siempre hacia adelante. Sin embargo, el método propuesto sigue siendo válido para el caso en que el movimiento sea solo hacia atrás. Algunos experimentos preliminares mostraron que el rendimiento se mantiene. En consecuencia, es totalmente factible realizar maniobras complejas coordinadas por un planificador de más alto nivel que escoja qué controlador ocupar (hacia adelante o atrás) dependiendo de la misión en curso.

Online y Offline Reinforcement Learning

En primer lugar, al analizar los resultados de la Tabla 5.6 se observa un hallazgo importante: dado un conjunto de datos grande y variado, el mejor método de offline RL obtiene resultados tan buenos como un método de online RL. Inclusive, las diferencias comparativas en evaluación son mínimas y por ende el rendimiento es bastante similar. Adicionalmente, acorde a los resultados de la Tabla 5.11, los agentes de offline RL son capaces de generalizar muy bien a datos de teleoperación que contienen menos cantidad de trayectorias y de peor calidad. Esto aborda directamente uno de

los objetivos principales de este trabajo: prescindir de la interacción con el ambiente para entrenar agentes de navegación inteligentes. Luego, la dificultad viene en obtener un conjunto de datos adecuados. Si es posible lograr lo anterior, entonces es altamente probable que el método planteado en este trabajo sea aplicable a un entorno real.

Por otro lado, los resultados obtenidos al evaluar el agente entrenado usando online RL en un mapa distinto (mapa *ks*) al de entrenamiento son muy buenos. Aún cuando el ambiente de evaluación es más complejo que el de entrenamiento, el agente es capaz de realizar las misiones encomendadas sin mayores problemas. Este comportamiento viene dado por la simplicidad del modelamiento de las observaciones, en particular lo que refiere a la percepción. Por ende, el agente es capaz de entender la topología de los túneles circundantes sin problemas inclusive sin haberlos visitado antes. Otro aspecto que explica estos resultados es que si bien el nuevo ambiente es más complejo, comparativamente la estructura de los túneles no varía mucho. Esta misma observación se podría hacer sobre entornos mineros reales, lo que realza la potencial capacidad de generalizar de este método para este tipo de aplicaciones.

En cuanto a la generación de datos, se logró satisfactoriamente el objetivo propuesto. Se crearon conjuntos de datos de diferentes tamaños y calidades, que permitieron llevar a cabo los experimentos descritos en la sección 5. Estos conjuntos de datos pueden ser utilizados en el futuro para entrenar nuevos agentes o para realizar otros estudios. Es importante señalar que en los experimentos que utilizan conjuntos de datos procedentes de un agente de DRL (TD3), existe cierto sesgo, ya que estos datos están cerca del óptimo. Sin embargo, la generación de diversos conjuntos de datos en diferentes etapas del entrenamiento (como se describe en la Tabla 5.4) ayuda a despejar este sesgo en los experimentos subsiguientes.

Posteriormente, en los resultados de la Tabla 5.6 se evalúa la capacidad que tienen diversos métodos de ORL para resolver el problema planteado. En concreto, se observa que si se despeja la variable asociada a datos, mediante la utilización de una cantidad y calidad suficiente, no todos los métodos son apropiados y se descarta CQL como un método candidato. Por otro lado, al usar datos de teleoperación, que son los más cercanos a una aplicación real, se observa que AWAC presenta los mejores resultados pues es mucho más robusto que sus competidores respecto a los datos de entrenamiento. Un modo de falla común que suele afectar a CQL es su sensibilidad a la selección de hiperparámetros. Debido a que el ajuste de hiperparámetros realizado no fue excesivamente minucioso, es posible que este factor explique la diferencia en rendimiento con el resto de métodos. Varios autores han observado lo mismo, por ejemplo, en [17] se observa que este método falla por completo en algunos escenarios debido a esto y en [22] se estudia con detención este problema. Por otro lado, el método AWAC, al ser más simple en el tipo de regularización aplicada, es mucho menos sensible a hiper-parámetros y a la distribución de los datos, por tanto gracias a esta versatilidad suele presentar mejores resultados inclusive en conjuntos de datos más pequeños o de calidad variable.

Los resultados alcanzados por los agentes entrenados usando offline RL son muy coherentes con otros alcanzados recientemente en la literatura. En particular, en [17] se evalúa extensivamente el mismo conjunto de métodos en una tarea de manipulación robótica simulada y real. En dicho trabajo se extraen observaciones similares: CQL desempeña mal mientras que CRR, IQL y AWAC entregan mejores resultados. Además, en [17] se muestra que para la tarea en cuestión CRR y AWAC suelen presentar resultados robustos cuando decae la calidad del conjunto de entrenamiento. Esto

resulta consistente con los resultados de la Tabla 5.5, donde el mismo par de métodos destaca en la tarea de navegación cuando se utilizan datos de teleoperación con menor cantidad de trayectorias y peor retorno promedio. Adicionalmente, en [49] se evalúan los métodos IQL y AWAC para una tarea de manipulación robótica pero con fuerte enfoque práctico. En dicho trabajo llegan a conclusiones similares, los agentes ORL poseen mejor rendimiento que las políticas que extraen los datos y tanto IQL como AWAC obtienen resultados competitivos. Por otro lado, también observan que en cierto momento, utilizar conjuntos de datos más grandes no mejora sustancialmente el rendimiento de los agentes.

Por otro lado, al analizar los resultados de la Figura 5.4 y de la Tabla 5.12 es posible determinar un momento aproximado en que empieza a decaer el rendimiento de los agentes. Este momento indica la cantidad mínima necesaria para alcanzar al menos un 99 % en la tasa de éxito y se ubica en aproximadamente 1500 episodios, los cuales pueden o no ser perfectos ya que mientras exista una diversidad suficiente el agente entrenado podrá tomar decisiones adecuadas. Esta aseveración está respaldada por los resultados de la Tabla 5.13, la que indica que no necesariamente una mayor calidad de trayectorias en los datasets se traduce en mejores resultados. Esto tiene sentido pues tanto AWAC como los otros métodos de ORL proponen formas de discriminar críticamente qué tan buena es una acción en cierto estado, por ende información sobre malas decisiones puede ser tan útil como la de buenas decisiones. Esta razón fundamental es la que alza a los métodos de offline RL por sobre métodos más tradicionales de aprendizaje supervisado, como por ejemplo *Behavioral Cloning*.

Dicho lo anterior, es de suma relevancia poder analizar la diversidad de los datos. Ya que monitorear esta variable es difícil en espacios de observación y acción continuos, se construyó una visualización auxiliar que puede resultar útil para el proceso de extracción de datos. La idea fundamental es poder visualizar qué tanto se ha recorrido el espacio de tuplas observación-acción en relación a un conjunto de datos dado.

El primer paso para construir esta visualización es utilizar la técnica PCA (*Principal Component Analysis*) [10] para establecer un espacio vectorial bi-dimensional apropiado. Este espacio se encuentra al ajustar PCA sobre un conjunto de datos de referencia. En este caso, dado que no solo se quiere visualizar la diversidad de las observaciones, sino también de las acciones, se concatenan las observaciones y acciones de un dataset para crear el conjunto de datos que utilizará PCA. Luego, gracias a las características de PCA, el espacio generado será tal que captura la mayor cantidad de varianza en los datos y por ende será propicio para visualizar la diversidad de estos. Finalmente, si se toma cualquier conjunto de tuplas observación-acción y se proyecta en este espacio bi-dimensional, es posible generar la visualización de la Figura 6.1.

En primer lugar, en la Figura 6.1 se observa que al proyectar el dataset de referencia, se observa exactamente lo esperado: el espacio está lleno (h). Luego, al usar datasets más pequeños (a-g) correspondientes a teleoperación se observa cómo el espacio se va recorriendo cada vez más cuando se incrementa la cantidad de episodios recolectados. Esta completitud del espacio recorrido representa de forma indirecta la diversidad de los datos recolectados. Al cabo de 750 episodios, la diversidad es bastante aceptable, concordando con los resultados de la Tabla 5.13. Es de esperar que al seguir recolectando datos después de los 750 episodios seguirá aumentando la cantidad-diversidad de los datos y con ello el rendimiento de los agentes entrenados.

Una desventaja de esta visualización es que se debe definir y contar un dataset de referencia. Una

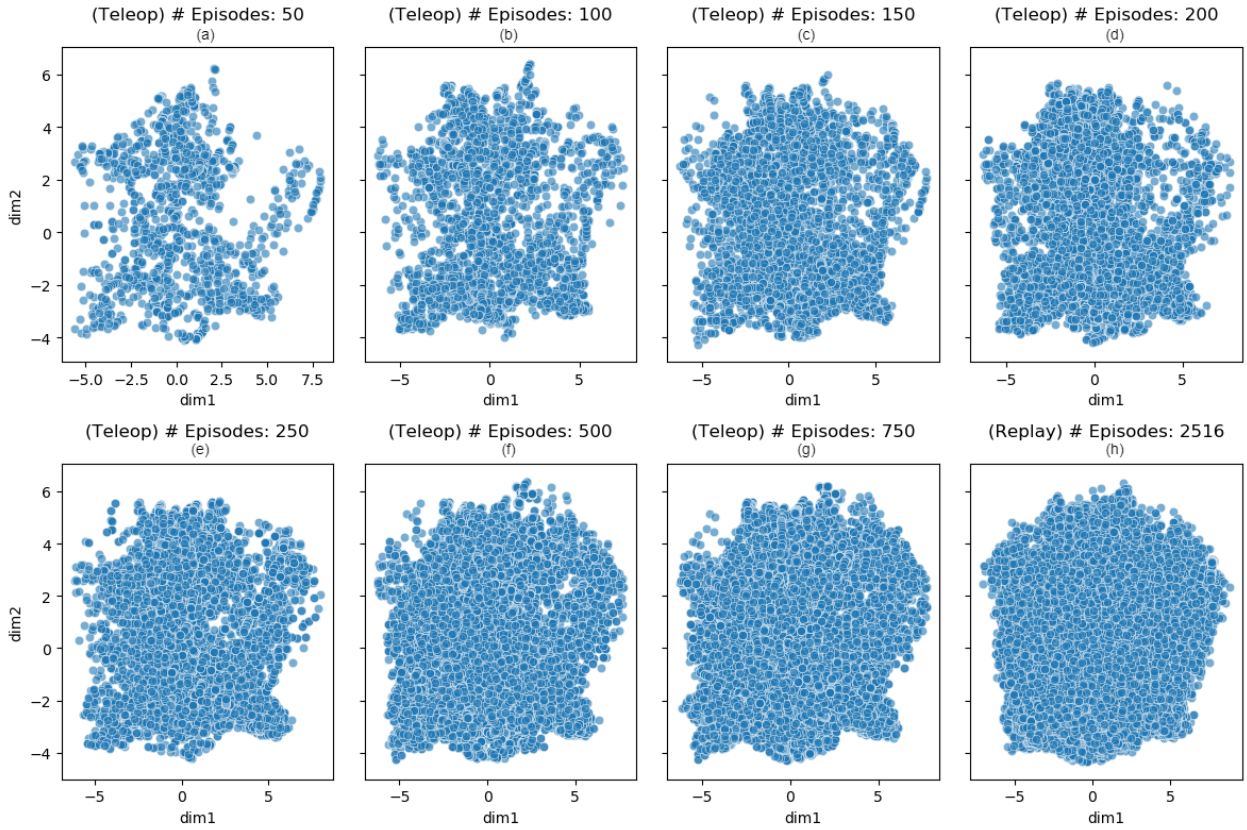

Figura 6.1: Visualización de la evolución en la diversidad de los datos a medida que se realiza el proceso de recolección de datos. Para la referencia se utilizó el dataset *replay #1 - 15 %* (h).

heurística para escoger este dataset es que se sepa a priori que los resultados obtenidos usándolo sean buenos. Luego, por ejemplo, en una aplicación real, es posible utilizar un dataset que obtiene buenos resultados en simulación como referencia para auxiliar el proceso de extracción de datos en la realidad. Esto siempre y cuando la brecha entre la simulación y la realidad no sea muy grande pues sino la distribución de los datos en simulación no serán representativa de la realidad y por ende la referencia no será válida.

Por último, gracias a los resultados de la sección 5 y la presente discusión, es posible afirmar que es factible ocupar offline RL para resolver el problema enunciado. Sin lugar a dudas el principal reto está en la generación de un dataset que posea suficiente cantidad y diversidad. Si mediante teleoperación de máquinas reales es factible extraer la cantidad necesaria de datos, entonces es muy probable que la navegación de un agente entrenado mediante offline RL sea exitosa. Si bien los métodos estudiados presentaron una buena capacidad de generalización, aún existe la posibilidad de que ocurran imprevistos en la operación producto de situaciones borde no capturadas en los datos y en las cuales el agente no pueda generalizar con suficiente confidencia. En consecuencia, para reducir la probabilidad de ocurrencia de estas situaciones es importante contar con un dataset de experiencias diversas. No obstante lo anterior, dada la naturaleza de la aplicación, sigue siendo muy importante la implementación de medidas de seguridad que prevengan cualquier accidente o daño ante alguna reacción imprevista del agente.

Si bien todo lo expuesto plantea la factibilidad de aplicar este método al escenario real de ope-

ración de un LHD en una mina, recolectar sobre 1000 episodios sigue siendo algo difícil de lograr por problemas de logística y costos. Con el objetivo de sortear esta dificultad, en el anexo A se propone un método de recolección de datos alternativo. Este método acelera varias veces la cantidad de episodios recolectados en comparación a un enfoque más ingenuo en el que un teleoperador extrae un episodio a la vez. Por consecuencia y gracias a esta metodología se mejora drásticamente la factibilidad de ocupar offline RL en un entorno real.

Capítulo 7

Conclusión

El objetivo de este trabajo es resolver el problema de conducción autónoma de LHDs al interior de minas utilizando aprendizaje reforzado offline. Para lograrlo, se ha planteado una metodología que aborda todos los componentes necesarios para su solución en un ambiente simulado tomando como antecedente lo observado en una exhaustiva revisión bibliográfica. Algunos de los componentes implementados más relevantes para el desarrollo del presente son el simulador, el modelamiento del problema como uno de aprendizaje reforzado, los algoritmos requeridos y los procedimientos de evaluación.

Con lo anterior, se realizaron diversos experimentos que buscan validar la eficacia del método propuesto así como estudiar algunas dimensiones como por ejemplo la cantidad de datos. Estos experimentos mostraron que el modelamiento realizado es efectivo y que dadas ciertas condiciones tanto métodos de aprendizaje reforzado online como offline pueden resolverlo. Por otro lado, se demostró empíricamente que dentro de los métodos de aprendizaje reforzado offline estudiados el que presenta mejores resultados es *Advantage Weighted Actor-Critic* (AWAC), el cual inclusive en escenarios donde existe una poca cantidad de datos logra alcanzar tasas de éxito mayores al 99 %. Además, estos experimentos permitieron derivar una noción de la cantidad mínima de datos requerida para resolver el problema con una buena tasa de éxito, la cual se establece alrededor de los 1500 episodios. Adicionalmente, se provee una método para acelerar la adquisición de datos en un entorno real, de tal forma de posibilitar la aplicación del presente trabajo en un entorno minero dadas sus restricciones de tiempo y costos.

Finalmente, el desarrollo de este trabajo permitió divisar ciertas direcciones de trabajo futuro, tales como optimizaciones en el simulador que permitan mejorar la experimentación, mejorías en el diseño de la función de recompensa, nuevas rutinas de evaluación, posibles mejoras en el sistema de percepción, entre otros.

Se espera que esta tesis sea de utilidad para avanzar en la automatización de la navegación de vehículos mineros y así poder disminuir los costos de operación, mejorar la seguridad y optimizar la eficiencia de la operación en una industria tan importante para Chile y el mundo.

Bibliografía

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[2] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *CoRR*, abs/1806.06877, 2018.

[3] GHH Solid as a Rock. Lf-11h, Jul 2017.

[4] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. Distributed distributional deterministic policy gradients. *CoRR*, abs/1804.08617, 2018.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[6] Ching-An Cheng, Xinyan Yan, and Byron Boots. Trajectory-wise control variates for variance reduction in policy gradient methods. *CoRR*, abs/1908.03263, 2019.

[7] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[8] Taha Elmokadem and Andrey V Savkin. A method for autonomous collision-free navigation of a quadrotor uav in unknown tunnel-like environments. *Robotica*, 40(4):835–861, 2022.

[9] Gabriele Ermacora, Daniele Sartori, Manfredi Rovasenda, Ling Pei, and Wenxian Yu. An evaluation framework to assess autonomous navigation linked to environment complexity. In *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1803–1810, 2020.

[10] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[11] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

[12] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement lear-

ning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[13] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[14] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[15] Çaglar Gülçehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel J. Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando de Freitas. RL unplugged: Benchmarks for offline reinforcement learning. *CoRR*, abs/2006.13888, 2020.

[16] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020.

[17] Nico Gürtler, Sebastian Blaes, Pavel Kolev, Felix Widmaier, Manuel Wuthrich, Stefan Bauer, Bernhard Schölkopf, and Georg Martius. Benchmarking offline reinforcement learning on real-robot hardware. In *The Eleventh International Conference on Learning Representations*, 2023.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[19] Yuanjian Jiang, Pingan Peng, Liguan Wang, Jiaheng Wang, Yongchun Liu, and Jiaxi Wu. Modeling and simulation of unmanned driving system for load haul dump vehicles in underground mines. *Sustainability*, 14(22):15186, 2022.

[20] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, abs/1806.10293, 2018.

[21] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *CoRR*, abs/2110.06169, 2021.

[22] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

[23] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan,

and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020.

[24] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *J. Mach. Learn. Res.*, 4(null):1107–1149, December 2003.

[25] Johan Larsson. *Unmanned operation of load-haul-dump vehicles in mining environments*. PhD thesis, Örebro university, 2011.

[26] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

[27] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[28] Sina Sharif Mansouri, Christoforos Kanellakis, Dariusz Kominiak, and George Nikolakopoulos. Deploying mavs for autonomous navigation in dark underground mine environments. *Robotics and Autonomous Systems*, 126:103472, 2020.

[29] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In *Proceedings of the 13th ACM SIGBED International Conference on Embedded Software (EMSOFT)*, pages 1–10, 2016.

[30] Mauricio Mascaró, Isao Parra-Tsunekawa, Carlos Tampier, and Javier Ruiz-del Solar. Topological navigation and localization in tunnels—application to autonomous load-haul-dump vehicles operating in underground mines. *Applied Sciences*, 11(14):6547, 2021.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *Deep Learning Workshop, Neural Information Processing Systems (NIPS)*, 2013.

[32] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *CoRR*, abs/2006.09359, 2020.

[33] Tolga Özaslan, Giuseppe Loianno, James Keller, Camillo J Taylor, Vijay Kumar, Jennifer M Wozencraft, and Thomas Hood. Autonomous navigation and mapping for inspection of penstocks and tunnels with mavs. *IEEE Robotics and Automation Letters*, 2(3):1740–1747, 2017.

[34] Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The uncertainty bellman equation and exploration. In *International Conference on Machine Learning*, pages 3836–3845, 2018.

[35] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Çaglar Gülçehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *CoRR*, abs/2007.09055, 2020.

[36] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019.

[37] Doina Precup, Richard Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, 06 2000.

[38] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *arXiv preprint arXiv:2203.01387*, 2022.

[39] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010.

[40] Kajetan Schweighofer, Markus Hofmarcher, Marius-Constantin Dinu, Philipp Renz, Angela Bitto-Nemling, Vihang P. Patil, and Sepp Hochreiter. Understanding the effects of dataset characteristics on offline reinforcement learning. *CoRR*, abs/2111.04714, 2021.

[41] Tianyu Shi, Dong Chen, Kaian Chen, and Zhaojian Li. Offline reinforcement learning for autonomous driving with safety and exploration enhancement. In *Machine Learning for Autonomous Driving Workshop on Neural Information Processing Systems (NIPS)*, 2021.

[42] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[43] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 31–36. IEEE, 2017.

[44] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021.

[45] Ratan Raj Tatiya. *Surface and underground excavations, 2nd edition: Methods, techniques and equipment*. CRC Press, 2018.

[46] Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148. PMLR, 2016.

[47] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7768–7778. Curran Associates, Inc., 2020.

[48] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[49] Gaoyue Zhou, Liyiming Ke, Siddhartha Srinivasa, Abhinav Gupta, Aravind Rajeswaran, and Vikash Kumar. Real world offline reinforcement learning with realistic data source. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022.

Anexos

Anexo A

Metodología de aceleración en la extracción de datos mediante teleoperación

Una de las ventajas del modelo planteado en la sección 4.2 es que la definición inicial de un episodio se da por dos componentes: (i) el estado inicial de la máquina y (ii) el objetivo de navegación, el que debe estar en una vecindad local de la máquina. Por ende, dado un objetivo de navegación único, es posible crear episodios virtuales alternativos *imaginando* otros objetivos que no necesariamente sean a los que se esté dirigiendo el teleoperador, similar a lo propuesto por Andrychowicz et al. en [1]. De esta forma, se obtienen observaciones y recompensas alternativas, las cuales en conjunto con las acciones tomadas más el cálculo del estado terminal permitirán obtener transiciones adicionales y por ende acelerar la extracción de datos.

En concreto, dado un objetivo principal w_k y un conjunto de $\{w_k^1, w_k^2, ..., w_k^n\}$ objetivos alternativos, es posible crear un total de $n+1$ episodios, donde en cada episodio alternativo e_k^j, al instante t, se pueden generar las observaciones y recompensas alternativas según las ecuaciones (A.1) y (A.2).

$$\hat{o}_t = f(o_t, w_k^j) \tag{A.1}$$

$$\hat{r}_t = r(\hat{o}_t, a_t) \tag{A.2}$$

donde f es la transformación que computa las nuevas cantidades relevantes en la observación. En este caso, deberá modificar únicamente la distancia relativa y ángulo hacia el objetivo alternativo w_k^j. $\hat{r}_t$ queda totalmente definido por la nueva observación $\hat{o}_t$ y la acción tomada en el instante t, reutilizando la formulación dada en la sección 4.2. A modo de resumen, para cada instante t de un episodio alternativo es posible generar la tupla $(\hat{o}_t, \hat{r}_t, a_t)$.

Algunas consideraciones importantes sobre este método es que para que cada episodio alternativo tenga sentido los objetivos asignados deben estar poco correlacionados entre ellos. Si no fuera así, los episodios alternativos serían similares entre ellos y proveerán poca diversidad al dataset construido. Una heurística sencilla para forzar esta de-correlación es seleccionar objetivos tales que todos estén a una distancia mínima entre ellos.

En la Figura A.1 se ilustra el proceso de extracción de datos con generación de episodios alternativos. En ésta, se crean 5 objetivos alternativos virtuales mientras el operador se dirige hacia el

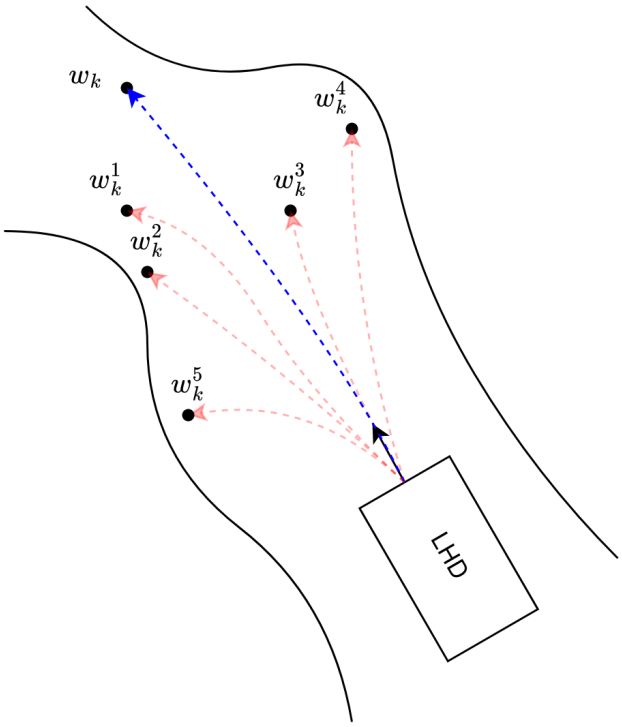

Figura A.1: Esquema representativo del proceso de extracción de datos imaginando episodios virtuales. En azul la ruta real de la máquina. Cada punto w_k^j define un objetivo local virtual.

objetivo w_k (trayectoria azul). En cada instante, se pueden generar transiciones imaginando que el objetivo fuera cada uno de los 5 alternativos. Usualmente las transiciones de episodios alternativos tendrán asignadas bajas recompensas pues el LHD no se está dirigiendo hacia sus objetivos virtuales. Sin embargo, este tipo de trayectorias de baja recompensa son igualmente útiles pues se está dando una demostración de qué no hacer.

Tal como se representa en la Figura A.1, es importante que los objetivos estén separados para proveer información importante. Esto ilustra cómo por cada trayectoria principal se pueden generar al menos 5 episodios nuevos, acelerando enormemente el proceso de recolección de datos.

Se estima que al teleoperar en el mapa *KS* 4.4 usando esta metodología se podrían generar aproximadamente 840 episodios por un recorrido completo de la máquina en el mapa, asumiendo un promedio de 5 episodios alternativos por cada episodio real. Por ende, 2 o 3 recorridos serían suficientes para obtener un dataset suficiente que permita entrenar un agente de navegación inteligente. La cantidad de recorridos puede ser disminuida si se generan más objetivos alternativos. Mientras no haya mucha correlación entre ellos las trayectorias generadas serán de utilidad. Es importante señalar que la metodología implementada ha sido validada de manera parcial. Aunque se ha comprobado que los objetivos alternativos se seleccionan adecuadamente, la validación no se ha llevado a cabo en su totalidad para la extracción final de datos. Esto implica que aún queda trabajo por realizar para asegurar la efectividad completa del enfoque propuesto.

Gracias a lo anterior, esta metodología se propone como una alternativa que posibilita drásticamente la aplicación del método planteado en esta tesis en un entorno de la vida real.

Anexo B

Extracción de dataset de teleoperación en simulación

Se debieron tomar algunas consideraciones adicionales para efectuar el proceso de teleoperación, la primera ocurre al momento de ejecutar acciones en el ambiente mediante teleoperación. Dado que se desea consistencia en los datasets generados, no se puede modificar en ningún caso el ambiente, de tal forma de que la interfaz sea constante a lo largo de los diversos experimentos. En particular, el espacio de acciones detallado en la sección 4.2.3 define a la **velocidad angular** como el controlador de la articulación del LHD. Sin embargo, controlar esta variable manualmente es sumamente innatural y difícil para un humano. Para facilitar este proceso se diseñó un controlador PID que cumple el rol de intermediario entre el ambiente y el operador. La Figura B.1 describe la relación entre estas entidades. El objetivo es que el operador controle una variable mucho más natural: el ángulo de la articulación. Así, gracias al controlador PID es posible utilizar la misma interfaz y los datos o transiciones son consistentes.

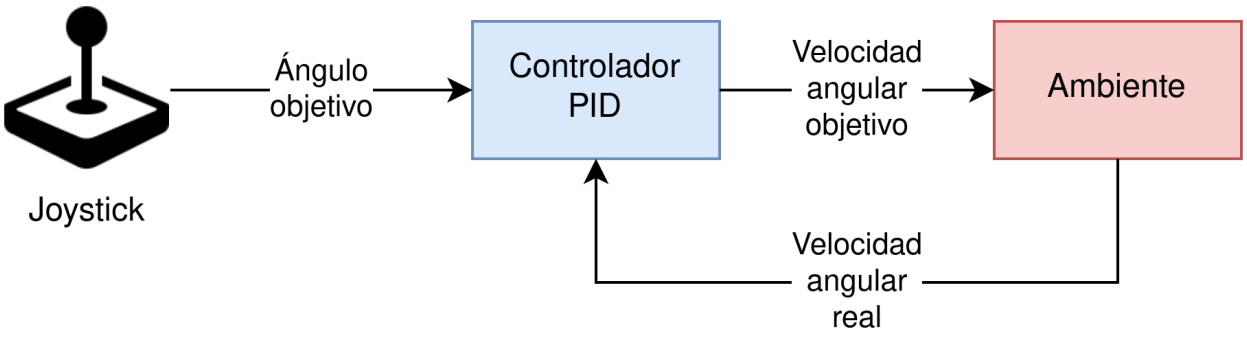

Figura B.1: Esquema de la relación entre el joystick, el controlador PID y el ambiente.

Para ajustar el controlador PID se hicieron algunas pruebas empíricas que permitieron determinar los parámetros más apropiados que entregaran estabilidad en la respuesta del controlador y también velocidad. Los parámetros utilizados son $K_p = 2$ para el control proporcional, $K_i = 0.1$ para el control integral y $K_d = 0.05$ para el control derivativo. Adicionalmente, para mejorar la suavidad de la teleoperación, se implementó un suavizado simple, que consta en promediar la velocidad angular actual con las últimas tres, de tal forma de minimizar giros bruscos.

Lo anterior define un método para controlar solo la velocidad angular. Sin embargo, las acciones también contienen la velocidad objetivo de la máquina. Para este caso, no se utiliza un controlador adicional pues la interfaz si es natural de controlar por un humano. Únicamente se aplica la misma estrategia de suavizado, promediando las últimas tres velocidades objetivos.

En cuanto al joystick de la Figura B.1, se utilizó un Joystick original de la consola de videojuegos Xbox 360. Esta elección se basa en que este recurso era de fácil acceso y simple de integrar. Sin embargo, es perfectamente posible utilizar otro tipo de hardware.