UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIE E INGENIERÍA DE LA COMPUTACIÓN
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS
INTELIGENCIA ARTIFICIAL

BÚSQUEDA Y RESCATE CON UAV'S UTILIZANDO APRENDIZAJE POR REFUERZO DE UNA CARGA ÚTIL EN CAÍDA

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
DANIEL ALEJANDRO ALONSO BASTOS

TUTOR
DR. PAUL ERICK MÉNDEZ MONROY
IIMAS-Mérida

CIUDAD UNIVERISTARIA, ENERO 2024

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y

HONESTIDAD ACADÉMICA Y PROFESIONAL

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a ala institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado "Búsqueda y rescate con UAV's utilizando aprendizaje por refuerzo de una carga útil en caída" con que presenté para obtener el grado de Maestro en Ciencia e Ingeniería de la Computación, es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Programa de Posgrado, citando las fuentes, ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad e los actos de carácter académico administrativo del proceso de titulación/graduación

Atentamente

Daniel Alejandro Alonso Bastos, 310043786

«It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers.»

Patrick Rothfuss

Agradecimientos

Una de las secciones más importantes del presente trabajo es, sin duda, esta sección, ya que es gracias a todas las personas y experiencias que me han guiado hasta este punto, en el cual me siento profundamente agradecido y contento.

Agradezco a la UNAM por brindarme el espacio y los recursos para mi formación, desde la licenciatura hasta la maestría. A lo largo de mi trayectoria académica y profesional, he tenido el privilegio de conocer a personas excepcionales de quienes estoy profundamente agradecido por los valiosos conocimientos y lecciones que me han brindado. Adicionalmente, he tenido la fortuna de colaborar con colegas, compañeros y amigos que han enriquecido mi experiencia día a día, con un agradecimiento especial a Gibran Rage, Juan Pablo Blasco, Mariano Benlliure, Diego García, Alejandro Maza, Javier Santibañez, Xavier Leroux, Guillermo Fernández del Busto. Mi especial reconocimiento y agradecimiento al Dr. Paul Méndez por todas las pláticas, asesorías y consejos que me brindó en todo este camino, preocupándose por mí en el ámbito escolar, profesional y emocional, buscando siempre lo mejor.

Sin duda alguna, algo que potencializó mi persona y mi carrera fue pertenecer a la escudería UNAM Motorsports, que me brindó la oportunidad de trabajar hombro a hombro con un equipo de personas extraordinarias que también me brindaron la oportunidad de aprender de cada una de ellas. Un proyecto donde los momentos más difíciles, desafiantes, pero también satisfactorios en mi vida. Agradezco a Luis Rivero por permitirme conocer el proyecto, a Rodrigo Méndez que fue una piedra angular en mi joven carrera académica y profesional, quien me enseñó a ir más allá de mis límites. Agradezco a todas las personas que confiaron en mi como yo confié en ellas, fue muy grato crecer en paralelo con todas ellas.

Todo esto no podría ser posible si no fuera por el apoyo de la familia, las palabras se quedan pequeñas y necesitaría una sección completa para expresar el profundo agradecimiento que siento. A mi mamá y a mi hermana, simplemente son y serán mi modelo a seguir, las personas que siempre me han guiado, aconsejado, regañado para convertirme en la persona que soy hoy, las personas que aun cuando las situaciones son complicadas me enseñaron a salir adelante, las amo.

A mis hermanos Jesús y Oscar, por el apoyo incondicional en cada momento, por el orgullo que siento al ver las personas que son hoy y, sobre todo, saber que tuve y

tengo el privilegio de crecer a su lado. A su madre Gabriela y a su padre Lorenzo†, quienes me brindaron todo su apoyo y se preocupaban por mí, quienes me abrieron las puertas de su casa y familia, son y serán como mis segundos padres.

A mi extraordinaria pareja y mejor amiga, Yoali Arenas, quiero expresar mi más sincero agradecimiento. Valoro y aprecio todo su apoyo, paciencia y ayuda constante que han sido cruciales no solo para la culminación de este trabajo, sino que para cada uno de mis proyectos. Gracias por ser mi fuente diaria de motivación, por acompañarme en mi crecimiento y aconsejarme para ser una mejor persona. Su presencia ha sido esencial en mi viaje, convirtiéndola en mi confidente y aliada más valiosa. Estoy emocionado por continuar compartiendo éxitos y desafíos a su lado, construyendo un futuro de logros compartidos y crecimiento mutuo. Además, quiero extender mi agradecimiento a toda su familia, su apoyo constante, cálida hospitalidad y ejemplo inspirador de sus logros han sido fundamentales. Agradezco su generosidad y amor, elementos que han enriquecido mi vida de manera única.

A mis abuelos Bebey† y Marino†, a mis tíos Isabel, Wanda, Carlos, George, Fernando que hicieron lo imposible para ayudarme y apoyarme en todas las situaciones que más lo necesitaba, agradezco porque nunca me faltó un techo, un plato de comida y un abrazo por parte de ellos.

A mis grandes amigos que también los considero como mis hermanos. Kazuyuki, una gran persona que he tenido la oportunidad de conocer desde el primer día de preparatoria hasta ahora, que me ha aconsejado y ayudado a exigirme más cada día. Raúl Mendoza, una gran persona que me ha enseñado muchas cosas. Carlos Luna, una persona extraordinaria que me ha enseñado el significado de perseverancia, de quien estoy profundamente orgulloso de ver todos los logros que ha conseguido, siendo una motivación personal. Y, con una gran distinción a Alejandro Trejo†, que hoy te dedico este trabajo, que simplemente te adelantaste y que me deja con un gran reto y motivación, vivir mi vida por dos, espero volvernos a encontrar y poder contarte todo lo que sucedió en tu ausencia y que tú me cuentes en la mía, te quiero hermano. Aprendí demasiado de ti, me acompañaste en momentos muy difíciles y tuvimos grandes experiencias, extraño las profundas charlas y consejos.

Agradezco por todas las enseñanzas y experiencias que me han brindado todas las personas con las que he tenido oportunidad de laborar y compartir mi vida, ha sido un gran camino con grandes aprendizajes.

Agradezco a mi papá Romeo, su pareja Sandy, y a mis hermanas Alejandra y Camila por siempre preocuparse por mí y brindarme un amor sincero y caluroso. Aunque no siempre estemos juntos, son personas importantes en mi vida, los quiero.

Agradezco al programa UNAM-PAPIIT IN105623 por el soporte parcial a esta investigación. Por último, agradezco a los sinodales del trabajo por brindarme comentarios y espacios de discusión para generar un mejor trabajo.

Resumen

Los sistemas multi-robot consisten en un conjunto de robots que pueden cooperar y comunicarse entre sí para realizar determinadas tareas, siendo un campo emergente que ha recibido mucha atención por la ventaja de emplear esfuerzos coordinados para lograr tareas y aplicaciones complejas, abriendo el abanico de aplicaciones.

Los globos de gran altitud o *high-altitude balloons* (HAB's) son globos no tripulados que se llenan con helio o hidrógeno, alcanzando generalmente ≈ 35 km sobre el nivel del mar. Los globos meteorológicos son el tipo más común de HAB's, los cuales están equipados con sensores para obtener mediciones de las condiciones meteorológicas como la temperatura, presión, humedad, velocidad del viento, entre otras. Los HAB's presentan varias ventajas sobre el uso de satélites bajo la misma aplicación, principalmente tienen una mejor relación costo-beneficio, carga recuperable y el diseño de toda la misión es más sencilla. Dentro del funcionamiento de un HAB, llega un punto en el que estalla el globo, tal que la carga útil cae de manera ralentizada con ayuda de un paracaídas, pero la trayectoria del descenso tiene una alta incertidumbre, generando una posibilidad no recuperar la carga útil.

El objetivo del presente trabajo es modelar un Proceso de Decisión de Markov Parcialmente Observable (POMDP) para la coordinación de un sistema multi-robots UAV's con el objetivo de búsqueda y rescate de la carga útil de un globo de gran altitud (HAB), resolviéndolo mediante algoritmos de aprendizaje por refuerzo.

La contribución del trabajo es un marco de trabajo que emplea tecnologías existentes para soluciones problemas que impliquen que un sistema multi-robot de UAV's solucione una tarea en común, al menos computacionalmente, con la capacidad de modificar la dinámica y control de un UAV, generar un ambiente de simulación que capture la dinámica del entorno e implementar algoritmos para realizar la tarea en común, en este caso particular, se implementaron algoritmos de aprendizaje por refuerzo.

La simulación de las trayectorias que emulan el descenso de una carga útil fueron realizadas mediante un *script* de Python que simula múltiples trayectorias en función del lugar, día y hora de lanzamiento del HAB utilizando el software *Cambridge University Spaceflight Prediction*, guardando su posición (x, y, z) para cada paso de tiempo en archivos CSV, siendo estos los que se utilizan para el entrenamiento.

El entorno de simulación fue desarrollado con *Unreal*, teniendo la capacidad de simular la física del entorno (e.g., gravedad, viento), asó como modelar las interacciones entre los múltiples agentes en el entorno (UAV's, carga útil, red, suelo). Después se utilizó *AirSim* que implementa la dinámica y control de un dron, tal que se pueden generar un conjunto de acciones ejecutables en el entorno de simulación mediante APIs.

Una vez teniendo el entorno de simulación, la flexibilidad de controlar a los UAV's dentro del entorno y obtener un espacio de observaciones mediante APIs, en teoría, ya se cuentan con los elementos necesarios para entrenar un modelo de aprendizaje por refuerzo, por lo que el entorno de simulación se envolvió en un entorno de entrenamiento de *OpenAI Gym* para tener la flexibilidad de entrenar algoritmos de aprendizaje por refuerzo, pero el objetivo no es desarrollar algoritmos sino que probar algoritmos existentes, por lo que se utilizó la librería *Stable Baseline3* que contiene una serie de algoritmos de aprendizaje por refuerzo implementados, pero necesitan un entorno de entrenamiento de *OpenAI Gym*, con estos pasos, se desarrolló el marco de trabajo que puede servir para entrenar UAV's para solucionar distintos problemas.

Adicionalmente, para utilizar aprendizaje por refuerzo es necesario definir un espacio de acciones y una señal de recompensa. En el caso del espacio de acciones, se generaron 2 posibles espacios, uno discreto y uno continuo. Mientras que, para la señal de recompensa está compuesta por varios componentes que incentivan distintos objetivos y restricciones como aproximarse lo más rápido posible a la carga útil, estar dentro de una zona de captura, estar dentro de la región horizontal y vertical definida por las leyes mexicanas, realizar la captura de manera exitosa lo más centrado a la red, y lo más suave posible.

Los algoritmos probados fueron PPO con un espacio discreto y continuo de acciones y SAC con un espacio continuo de acciones, dichos algoritmos fueron seleccionados en función de la literatura encontrada [1, 2, 3, 4, 5]. Para cada uno de los escenarios se ejecutaron 5 experimentos para comparar su desempeño, así como validar que los distintos grupos fueran significativamente diferentes entre sí utilizando la prueba Wilcoxon. Los resultados mostraron que utilizar el algoritmo SAC daba mejores resultados que PPO para resolver este problema en particular, debido a su capacidad de manejar espacios de acciones continuos y su estrategia de exploración eficiente mediante la maximización de entropía, y además uno de los puntos importantes que se observaron a lo largo de los resultados es que es posible que aumentando el número de pasos de entrenamiento se pudiera lograr una convergencia de los resultados e inclusive comenzar a realizar las capturas.

El objetivo princiapl de la tesis se cumplió de manera parcial, ya que se consiguió desarrollar el POMDP y un marco de trabajo permitiendo el entrenamiento de un par de algoritmos de aprendizaje por refuerzo (PPO y SAC) con distintos espacios de acciones (discreto y continuo), pero con la capacidad y flexibilidad de utilizar más

algoritmos de aprendizaje por refuerzo.

Asimismo, el objetivo de realizar la búsqueda se cumple, porque los resultados muestran que los UAV's conforme se entrenaron, fueron aprendiendo estrategias para buscar cada vez con mayor agilidad la carga útil. El objetivo de realizar el rescate no se cumplió, es decir, los UAV's no consiguieron rescatar la carga útil en ningún momento con el mejor modelo entrenado (SAC con un espacio de acciones continuas).

A pesar de no haber logrado la captura exitosa, los resultados sugieren que el modelo de aprendizaje por refuerzo implementado ha permitido que los UAV's aprendan estrategias de búsqueda al acercarse cada vez más a la carga útil. La diversidad en las trayectorias y las oportunidades de mejora identificadas proporcionan una base sólida para experimentos futuros. Con ajustes cuidadosos en los hiperparámetros y restricciones, así como la consideración de mejoras en la lógica de la captura, se espera que el modelo alcance un rendimiento más consistente y logre la captura de la carga útil. Este trabajo sienta las bases para futuras investigaciones en el campo del control distribuido de UAV's para tareas de captura de objetos móviles.

Abstract

Multi-robot systems consist of a set of robots that can cooperate and communicate with each other to perform specific tasks. This emerging field has garnered considerable attention due to the advantage of employing coordinated efforts to achieve complex tasks and applications, expanding the range of potential applications.

High-altitude balloons (HABs) are unmanned balloons filled with helium or hydrogen, typically reaching approximately 35 km above sea level. Weather balloons are the most common type of HABs, equipped with sensors to obtain measurements of meteorological conditions such as temperature, pressure, humidity, wind speed, among others. HABs offer several advantages over satellites for similar applications, primarily in terms of cost-effectiveness, recoverable payload, and mission simplicity. During the operation of a HAB, there comes a point where the balloon bursts, causing the payload to descend slowly with the help of a parachute. However, the descent trajectory has high uncertainty, posing a risk of payload unrecoverability.

The objective of this work is to model a Partially Observable Markov Decision Process (POMDP) for the coordination of a multi-robot UAV system with the goal of searching and rescuing the payload of a high-altitude balloon (HAB), solving it through reinforcement learning algorithms.

The contribution of this work is a framework that utilizes existing technologies to address problems involving a multi-robot UAV system solving a common task, at least computationally. This framework has the capability to modify the dynamics and control of a UAV, generate a simulation environment capturing the dynamics of the surroundings, and implement algorithms to perform the common task. In this specific case, reinforcement learning algorithms were implemented.

The simulation of trajectories emulating the descent of a payload was carried out using a Python script that simulates multiple trajectories based on the location, day, and time of HAB launch, using the Cambridge University Spaceflight Prediction software. The position (x, y, z) for each time step was saved in CSV files, which were used for training.

The simulation environment was developed with Unreal, capable of simulating environmental physics (e.g., gravity, wind) and modeling interactions between multiple

agents in the environment (UAVs, payload, net, ground). Subsequently, AirSim was used, implementing the dynamics and control of a drone, enabling the generation of a set of executable actions in the simulation environment through APIs.

Once the simulation environment was established, with the flexibility to control UAVs within the environment and obtain an observation space through APIs, the necessary elements for training a reinforcement learning model theoretically existed. The simulation environment was wrapped in an OpenAI Gym training environment to have the flexibility to train reinforcement learning algorithms. However, the goal is not to develop algorithms but to test existing ones. Therefore, the Stable Baseline3 library, which contains a series of implemented reinforcement learning algorithms, was used. With these steps, the framework was developed, which can be used to train UAVs to solve various problems.

Additionally, to use reinforcement learning, it is necessary to define an action space and a reward signal. In the case of the action space, two possible spaces, discrete and continuous, were generated. Meanwhile, the reward signal is composed of several components incentivizing different objectives and constraints, such as approaching the payload as quickly as possible, staying within a capture zone, remaining within the horizontal and vertical region defined by Mexican laws, performing a successful capture as centered on the net as possible, and as smoothly as possible.

The tested algorithms were PPO with discrete and continuous action spaces and SAC with a continuous action space. These algorithms were selected based on the literature found [1, 2, 3, 4, 5]. For each scenario, 5 experiments were conducted to compare their performance and validate that the different groups were significantly different using the Wilcoxon test. The results showed that using the SAC algorithm yielded better results than PPO for solving this particular problem, due to its ability to handle continuous action spaces and its exploration strategy through entropy maximization. Additionally, one important observation throughout the results is that increasing the number of training steps might lead to result convergence and even initiate successful captures.

The main objective of the thesis was partially achieved, as the POMDP and a framework were developed, allowing the training of a couple of reinforcement learning algorithms (PPO and SAC) with different action spaces (discrete and continuous). However, the capacity and flexibility to use more reinforcement learning algorithms were also included.

Likewise, the objective of conducting the search was fulfilled, as the results showed that the UAVs, as they were trained, learned strategies to search for the payload more efficiently. The objective of performing the rescue was not achieved; in other words, the UAVs did not manage to rescue the payload at any point with the best-trained model (SAC with continuous action spaces).

Despite not achieving successful capture, the results suggest that the implemented reinforcement learning model has enabled UAVs to learn search strategies by approaching the payload more closely over time. The diversity in trajectories and identified improvement opportunities provide a solid foundation for future experiments. With careful adjustments to hyperparameters and constraints, as well as consideration of improvements in capture logic, it is expected that the model will achieve more consistent performance and successfully capture the payload. This work lays the groundwork for future research in the field of distributed control of UAVs for mobile object capture tasks.

Índice general

1 Introducción

Los sistemas multi-robot o *multi-robot systems* (MRS) es un campo dentro del área de investigación de robots móviles (Fig. 1.1). Los sistemas multi-robot consisten en un conjunto de robots que pueden cooperar y comunicarse entre sí para realizar determinadas tareas [6]. Es un campo emergente que ha recibido mucha atención por la ventaja de emplear esfuerzos coordinados para lograr tareas y aplicaciones complejas [7], abriendo el abanico de aplicaciones. En la Fig. 1.2 se muestra una tabla comparativa entre los diversos enfoques para tratar a los MRS.

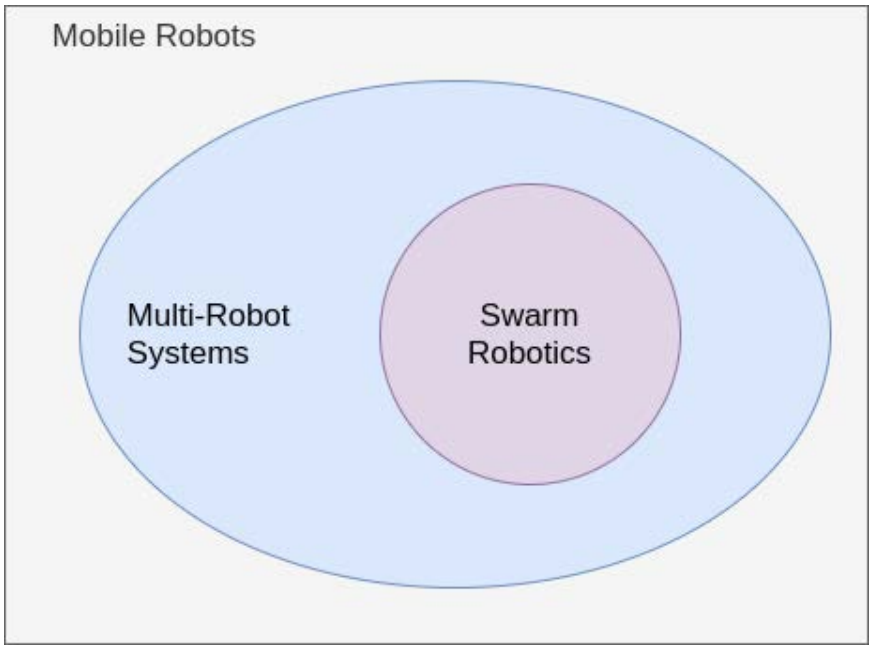


Figura 1.1: Campos de estudio. Tomada de [8].

Comparison of swarm robotics and other systems				
	Swarm Robotics	**System of multiple robots**	**Sensor network**	**Multi-agent system**
Population Size	Variation in great range	Small	Fixed	In a small range
Control	Decentralized and autonomous	Centralized or remote	Centralized or remote	Centralized or hierarchical or network
Homogeneity	Homogenous	Usually heterogeneous	Homogenous	Homogenous or heterogeneous
Flexibility	High	Low	Low	Medium
Scalability	High	Low	Medium	Medium
Environment	Unknown	Known or unknown	Known	Known
Motion	Yes	Yes	No	Rare
Typical applications	Post-disaster relief Military application Dangerous application	Transportation Sensing Robot football	Surveillance Medical care Environmental protection	Net resources management Distributed control

Figura 1.2: Comparación de robótica de enjambres y otros sistemas. Tomada de [9].

La robótica de enjambre o *swarm robotics* (SR) es una subárea dentro de los MRS, es un campo emergente bioinspirado que adapta el fenómeno de los enjambres naturales a la robótica para formar una colección de robots razonablemente sencillos, de manera distribuida y descentralizada. Las principales características en estos sistemas es que son escalables, flexibles y robustos. Además, muestran auto-organización, autonomía, cooperación y coordinación entre ellos, al no tener alguna entidad central que los controle [8, 10].

Los MRS coordinan el comportamiento de los agentes tal que producen un comportamiento complejo que permite llevar a cabo diversas tareas que son imposibles de realizar por un solo individuo [10]. Existen varias definiciones de enjambre presentes en la literatura, pero una definición simple es que es un gran grupo de individuos que interactúan localmente con objetivos comunes [11]. Llevándolo a la idea de *swarm robotics*, se busca construir sistemas de enjambres de robots relativamente sencillos que interactúen entre ellos realizar el objetivo común.

Las tareas y aplicaciones de los MRS son muy amplias, y pueden utilizarse en una gran variedad de tareas donde el humano se encuentra limitado para alcanzar los objetivos. Las principales actividades que pueden realizarse son [10, 12]: actividades peligrosas, actividades que aumenten o disminuyan en el tiempo, actividades que requieren redundancia, actividades en regiones específicas.

Dada la versatilidad del uso de los MRS se cuenta con distintas aplicaciones en diversos campos como la agricultura, medicina, astronomía, energía, entre otras más. En la Fig. 1.3 se muestran las principales tareas identificadas en la literatura como e.g., la formación de figuras y patrones, transporte de objetos, búsqueda de recursos, cobertura de una zona. Adicionalmente, en la Tabla 1.1 se muestran algunas de las aplicaciones encontradas en la literatura, pero existen muchos más proyectos y aplicaciones en el campo de MRS [13].

Figura 1.3: Tareas principales en los sitemas multi-robot [10, 14].

Tabla 1.1: Ejemplo de aplicaciones de MRS por área.

Área	Aplicaciones
Agropecuario	SAGA es una plataforma experimental que buscan demostrar la importancia del uso de SR para trabajos de monitoreo, mapeo e inspección de malezas [15, 16, 17].
	Uso de UAV[1] para la inspección de cultivos para controlar la salud de los mismos, al tomar fotografías con mayor resolución que las imágenes satelitales [9].

[1]Un dron es un vehículo aéreo no tripulado o *Unmanned Aerial Vehicle* (UAV).

	El proyecto MARS despliega a varios robots que son orquestados por una unidad centralizada que se encuentra en la nube, con el objetivo de aumentar el rendimiento agrícola por hectárea, reducir insumos y el impacto ambiental [18].
	Recolección autónoma de cultivos de cereales al emplear versiones autónomas de los vehículos agrícolas tradicionales [19].
	Robots agrícolas para el cuidado de las plantas [20].
	Convertir tierras de cultivo distantes y/o inaccesibles que permanecen ociosas en campos de cultivos inteligentes, mediante el uso de UAV's al facilitar las tareas y operaciones agrícolas a distancia [21].
Aéreo	Monitoreo de tráfico de una ciudad virtual *SwarmCity* [22].
	Coordinación de UAV's en la búsqueda de objetivos [23].
Ambiental	Red de sensores acústicos submarinos formada por un enjambre de robots heterogéneo que se utiliza para la monitorización a largo plazo de entornos submarinos al recolectar datos de distintos sensores como temperatura, presión, etc [24].
	Exploración de entornos submarinos mediante la recopilación de datos sobre las propiedades del agua y la existencia de obstáculos [25].
	Lucha contra la propagación de incendios forestales [26].
Industria	Enjambre de robots para tareas en almacén [27, 28]
	Cooperación entre robots para transportar un objeto más grande a un destino [29].
	Pipebots, micro robots de servicio que buscan emplear para la gestión e inspección de las infraestructuras subterráneas, lugares peligroso y de difícil acceso [30].
Construcción	Enjambre de robots para construir de manera colaborativa e *in situ* formas tubulares independientes de materiales compuestos [31].
	Simulación de robots escaladores descentralizados capaces de atravesar y extender una estructura de marco bidimensional, además de anticipar y prevenir fallos estructurales [32]
	Sistema de construcción multi-agente inspirado en las termitas, donde un usuario especifica una estructura deseada, después el sistema genera las reglas para que los robots trepadores autónomos aseguren la construcción de la estructura [33].

1.1. Caso de estudio

Los globos de gran altitud o *high-altitude balloons* (HAB's) son globos no tripulados que se llenan con helio o hidrógeno, alcanzando generalmente ≈ 35 km sobre el nivel del mar. Los globos meteorológicos son el tipo más común de HAB's, los cuales están equipados con sensores para obtener mediciones de las condiciones meteorológicas como la temperatura, presión, humedad, velocidad del viento, entre otras [34, 35].

Los HAB's presentan varias ventajas sobre el uso de satélites bajo la misma aplicación, tienen una mejor relación costo-beneficio, alternativa ecológica al no emplear cohetes convencionales, disminución de la basura espacial, carga recuperable y el diseño de toda la misión es más sencilla. Además de los globos meteorológicos, hay más aplicaciones como multi-plataformas para realizar pruebas de vuelo de satélites compactos [36] y exploración e investigación estratosférica [37], el proyecto *Loon* de Google X busca proporcionar acceso a internet en zonas remotas [38], turismo espacial [39], mercadotecnia, e incluso plataformas de despegue para cohetes [40].

El funcionamiento típico de un HAB consiste en llenar un globo con helio o hidrógeno y conectarle una carga útil que contendrá los componentes necesarios para la misión (e.g., sensores, cámaras) y un paracaídas. Una vez que se tiene todo, se libera el globo tal que asciende a la estratosfera, durante el ascenso el globo comenzará a expandirse debido a la diferencia de presiones, y llegará un punto en el que el globo no soporte la presión interna, y por lo tanto, estallará, en ese momento, se alcanzó la altura máxima de la misión. Una vez que estalla el globo, el paracaídas se desplegará de manera progresiva por la densidad del aire[2], ralentizando la caída hasta aterrizar, como se observa en la Fig. 1.4. Finalmente, un equipo de búsqueda y rescate se encarga de recuperar la carga útil, junto con los datos de la misión para ser analizados posteriormente.

[2] A grandes alturas, la densidad del aire es baja, a medida que el globo descienda, la densidad del aire irá aumentando.

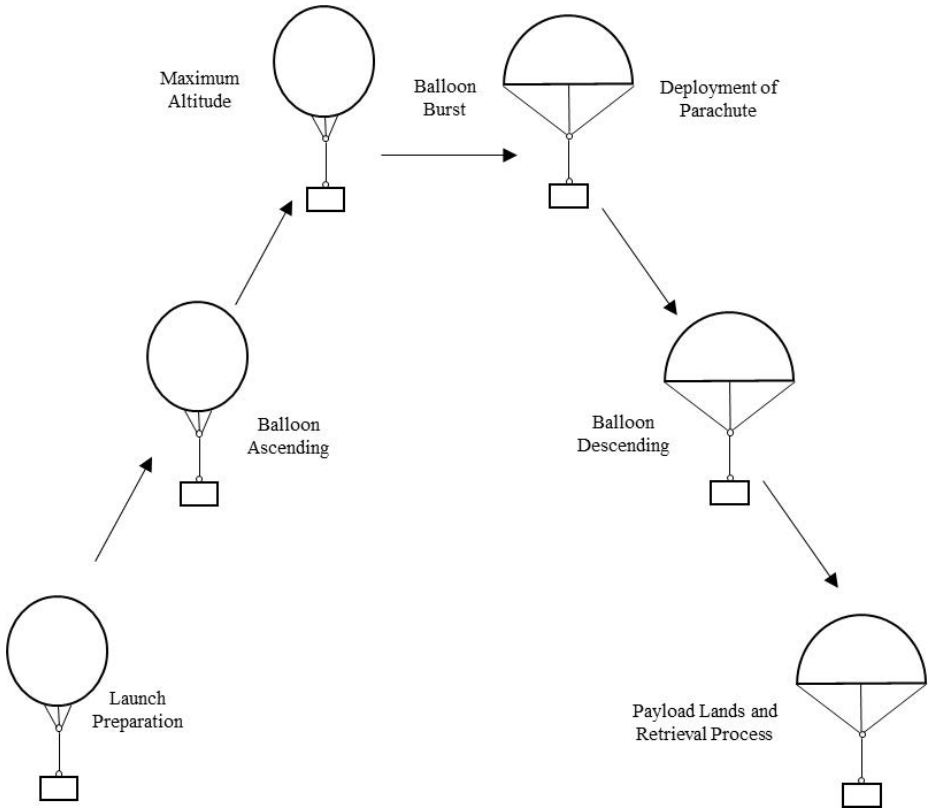


Figura 1.4: Operación de un HAB. Tomada de [34].

La trayectoria de vuelo en el descenso es caótica dado que está en función de las condiciones ambientales, lo que genera incertidumbre en el punto de caída de la carga útil, lo cual tiene varios inconvenientes como la caída en zonas de poco acceso para la recuperación, caída en zonas urbanas que puede generar accidentes, y sobre todo el riesgo de no recuperar la carga útil, que aunque es un sistema relativamente de bajo costo, la carga útil puede contener componentes muy caros [41]. Por los puntos mencionados se vuelve relevante considerar a la recuperación como una operación crítica dentro de la misión. Existen trabajos como la dirección autónoma del paracaídas empleando servomotores [35, 41], una técnica de doble globo, donde uno tiene una válvula automática para liberar el helio y controlar el ascenso, mientras que el otro globo funciona como un paracaídas [42].

En el presente trabajo se busca implementar un MRS de UAV's para recuperar la carga útil y transportarla hacia un cierto objetivo, y según nuestro conocimiento, no se ha realizado dicha aplicación para el rescate de la carga útil. La elección de algoritmos de aprendizaje por refuerzo para abordar el problema se basa en la capacidad para aprender políticas de toma de decisiones autónomas a partir de la interacción con entornos complejos y dinámicos. Los algoritmos de aprendizaje por refuerzo ofrecen una solución prometedora al permitir que el UAV líder aprenda, de manera adaptativa, las acciones óptimas en función de las observaciones recibidas, optimizando así su desempeño con múltiples objetivos y restricciones a lo largo del tiempo.

1.2. Objetivos

A continuación, se describirán el objetivo principal del presente trabajo, así como los objetivos particulares.

1.2.1. Objetivo de la tesis

Modelar un Proceso de Decisión de Markov Parcialmente Observable (POMDP) para la coordinación de un sistema multi-robots UAV's con el objetivo de búsqueda y rescate de la carga útil de un globo de gran altitud (HAB), resolviéndolo mediante algoritmos de aprendizaje por refuerzo.

1.2.2. Objetivos particulares

- Desarrollar un entorno de simulación que permita el entrenamiento de los algoritmos de aprendizaje por refuerzo al emular el comportamiento e interacciones de los multi-agentes considerando la física del entorno, tal que se obtenga información del entorno y se realice una acción del o los agentes en cada paso de tiempo.
- Probar distintos algoritmos de aprendizaje por refuerzo para resolver el POMDP que tiene como objetivo buscar y rescatar la carga útil de un HAB mediante un sistema multi-robot UAV's.

1.2.3. Contribución y relevancia

El campo de aplicación de los sistemas multi-robot es muy amplio, es el presente de algunas áreas y el futuro de muchas más. La aplicación se seleccionó porque rescatar la carga útil es una parte crítica del éxito de una misión HAB, adicionalmente la solución puede ser aplicable en el mediano plazo con proyectos dentro de la UNAM y personales. Además, es un problema que envuelve múltiples tareas como la búsqueda, rescate y traslado de un objeto.

El objetivo es realizar un primer paso para solucionar y optimizar más problemas en el sector aeroespacial y en otros sectores, dado que con el trabajo se persigue contar con una metodología, conocimiento e implementación del comportamiento de un sistema multi-robot que tienen un objetivo común, al menos en la parte computacional, siendo una base sólida para escalar y trasladar la solución a diferentes problemáticas y campos de aplicación.

La contribución es un marco de trabajo o *framework* (Fig. 3.1) que emplea tecnologías existentes para solucionar problemas que impliquen que un sistema multi-robot de UAV's solucione una tarea en común, al menos computacionalmente, con la capacidad de modificar la dinámica y control de un UAV, generar un ambiente de simulación que capture la dinámica del entorno e implementar algoritmos para realizar la tarea

común, que en este caso particular, se implementarán algoritmos de aprendizaje por refuerzo.

La solución propuesta si bien es específica, se pueden ir añadiendo trabajos posteriores como por ejemplo, control inteligente de la caída por medio de servomotores para disminuir el riesgo de perder la carga útil [35, 41], coordinación entre varios equipos de UAV's para recuperar la carga útil, implementar algoritmos de aprendizaje por refuerzo en cada UAV, tal que emerja la cooperación entre los UAV's siendo un sistema descentralizado, abriendo camino al campo de robótica de enjambres o *swarm robotics*.

Además, la solución es trasladable a diferentes problemas en distintos campos como, por ejemplo, búsqueda y rescate de personas, coordinación para el manejo de incendios forestales, asistencia en persecuciones, mantenimiento de granjas (animales, eólicas, solares), transporte, logística, vigilancia y seguridad, inspección en zonas de difícil acceso o en zonas peligrosas, medicina, entre muchas otras más.

Una de las ventajas de los sistemas multi-robot es la escalabilidad, flexibilidad y robustez que presentan estos sistemas, además de que se pueden desarrollar robots relativamente baratos al no estar equipados con mucha tecnología pero que con la coordinación y cooperación puedan realizar tareas más complejas, es por eso, que es un campo emergente con un abanico de aplicaciones muy amplio.

1.2.4. Resumen del contenido de la tesis

La tesis está dividida en seis capítulos diferentes:

- **Capítulo 1.** Conocer los objetivos de la presente tesis, así como las justificaciones y motivaciones.
- **Capítulo 2.** Explorar las principales definiciones en aprendizaje por refuerzo, generando las bases para describir a alto nivel los distintos algoritmos, así como su clasificación.
- **Capítulo 3.** Describir el marco de trabajo utilizado, describiendo las trayectorias simuladas de la carga útil en función de múltiples lugares de lanzamiento, así como la descripción del desarrollo del entorno de simulación donde se realizará el entrenamiento, simulando las distintas interacciones entre los agentes (UAV's, carga útil, ambiente).
- **Capítulo 4.** Describir los distintos componentes de los algoritmos de aprendizaje por refuerzo utilizados en la presente tesis: el espacio de acciones, el espacio de observaciones y las distintas señales de recompensa que se obtienen en cada paso de tiempo, así como las recompensas asociadas a los puntos o estados terminales.
- **Capítulo 5.** Analizar los resultados obtenidos del entrenamiento.

- **Capítulo 6.** Conocer las conclusiones y trabajo a futuro propuesto.

2 Aprendizaje por refuerzo

El aprendizaje por refuerzo (RL, por sus siglas en inglés de *Reinforcement Learning*) es un campo de la inteligencia artificial que ofrece una alternativa al aprendizaje supervisado, en donde no se utilizan datos etiquetados, sino que una función de recompensa que indica que tan bueno o malo es el comportamiento de un agente (tomador de decisiones) que está interactuando con un entorno y que busca lograr un objetivo.

En un escenario común de aprendizaje por refuerzo, el tomador de decisiones llamado agente interactúa con su entorno[1] en pasos de tiempo discretos t. En cada paso de tiempo, el ambiente produce un estado s_t del espacio de estados $\mathcal{S}$, al recibir el agente dicha observación o estado del ambiente, el agente realiza una acción según una política $\pi(a_t|s_t)$ que representa la probabilidad de tomar una acción a_t del conjunto de acciones posibles $\mathcal{A}$ (o en un ambiente determinista $\pi(s_t) = a_t$) estando en el estado s_t. La ejecución de esta acción influye sobre el entorno, cambiando su estado a uno nuevo. Un paso de tiempo después $t + 1$, el entorno le comunica al agente el valor de la acción que acaba de realizar mediante una recompensa $r_{t+1} \in \mathbb{R}$, además el agente recibe la observación correspondiente al nuevo estado del entorno s_{t+1}, tal como se muestra en la Fig. 2.1.

Obteniendo así una secuencia de estados, acciones y recompensas:

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, ... \tag{2.1}$$

Las tareas episódicas son aquellas que terminan las interacciones del agente y el entorno en un cierto tiempo T (e.g., cuando se gana o pierde un juego, cuando el robot colisiona), a dicha secuencia se le conoce como episodio; mientras que las tareas continuas realizan interacciones sin un límite, tal que:

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, ..., r_T, s_T. \tag{2.2}$$

Además del agente y del entorno, un sistema de aprendizaje por refuerzo contiene cuatro elementos principales [43]:

- **Política** (π). Es un mapeo del conjunto de estados del entorno $\mathcal{S}$ al conjunto de las posibles acciones $\mathcal{A}$, i.e., la política π toma un estado s como entrada y regresa una acción a como salida. Una política determinista es un mapeo

[1] En el presente trabajo los términos de entorno y ambiente se utilizan indistintamente.

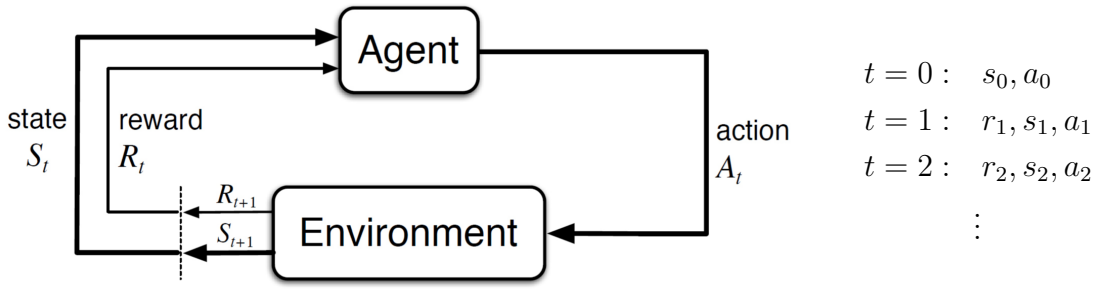


Figura 2.1: Diagrama del flujo de aprendizaje por refuerzo. Tomada de [43].

$\pi : \mathcal{S} \to \mathcal{A}$ (e.g., si la batería de un robot es baja entonces la acción del robot es ir a recargar la batería, tal que $\pi(baja) = recargar$). Por otro lado, una política estocástica es una mapeo $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, i.e., la política genera como salida la probabilidad de que el agente tome la acción a dado que se encuentra en el estado s, tal que:

$$\pi(a|s) = P(A_t = a|S_t = s). \tag{2.3}$$

La política entonces determina cómo un agente selecciona una acción según el estado del entorno que se le presenta.

- **Señal de recompensa**. La señal de recompensa es un valor escalar que define el objetivo del problema de aprendizaje por refuerzo. En términos generales guía al agente en su aprendizaje indicándole qué tan buenas o malas son las acciones que toma. También se podría pensar que es la forma de comunicar al agente qué es lo que queremos que se logre, no el cómo queremos que el agente logre la recompensa.

El objetivo del agente es maximizar la recompensa acumulada, a esta medida se le conoce como retorno esperado G_t. Si el agente se encuentra en el tiempo t, el retorno esperado a partir de dicho momento se define como la suma de recompensas para cada acción futura:

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T. \tag{2.4}$$

donde T es el estado terminal en una tarea episódica. Sin embargo, las tareas continuas no tienen un límite T lo que ocasiona que la Ec. (2.4) se vuelva una suma infinita. Para asegurar convergencia a un valor finito, se introduce un factor de descuento $\gamma \in [0, 1]$, resultando en la siguiente ecuación:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{2.5}$$

El factor de descuento γ determina la importancia de las recompensas futuras. Un factor de $\gamma = 0$ hará que el agente sea miope y únicamente tenga en cuenta

las recompensas actuales ($G_t = r_{t+1}$), mientras que un factor $\gamma = 1$ hará que el agente se esfuerce por obtener una recompensa alta a largo plazo, dicho de otra forma, a mientras mayor sea el valor de γ el agente se preocupa más por el futuro lejano, mientras menor sea el valor de γ el agente se preocupa más por las recompensas inmediatas.

- **Función de valor** ($\nu(s)$) Para que la política π del agente decida qué acción tomar en el tiempo t, el agente necesita estimar que tan bueno es estar en un cierto estado, i.e., la cantidad total de recompensa que un agente puede esperar acumular en el futuro partiendo de ese estado, la función de valor cuantifica entonces el retorno esperado para el estado s siguiendo la política π. De manera intuitiva, si un estado es muy cercano al estado objetivo, por lo general el valor será alto, mientras que, si cualquier acción en un determinado estado aleja al agente del objetivo, el valor será bajo.

  Dado que las recompensas que el agente puede esperar recibir en el futuro dependen de la acción que tome, la función valor se define con respecto a la política π. El valor de un estado s siguiendo la política π, denotado como $\nu_\pi(s)$, es el retorno esperado si el agente empieza en el estado s y sigue la política π para seleccionar todas las acciones para cada paso de tiempo:

$$\nu_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|S_t = s\right], \tag{2.6}$$

  donde $\mathbb{E}_\pi[\cdot]$ es el valor esperado de una variable aleatoria. De manera análoga existe una función de valor para acciones llamada función acción-valor $q_\pi(s, a)$ la cual cuantifica el retorno esperado si el agente se encuentra en el estado s y realiza una acción a siguiendo la política π para seleccionar todas las acciones para cada paso de tiempo:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|S_t = s, A_t = a\right]. \tag{2.7}$$

  Una propiedad fundamental de las funciones de valor es que satisfacen una relación recursiva entre el valor de un estado y el valor de sus estados sucesores inducida por la propiedad de Markov. La ecuación de Bellman para ν_π expresa esta relación:

$$\nu_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma\nu_\pi(s')], \ \forall s \in \mathcal{S}. \tag{2.8}$$

  La Ec. (2.8) calcula la media de todas las posibilidades y las pondera por su probabilidad de ocurrencia. En resumen, el valor del estado inicial debe ser igual

al valor descontado de su próximo estado esperado más la recompensa. En la Fig. 2.2 se ilustra lo descrito utilizando diagramas *backup* para ν_π y q_π, donde cada círculo blanco representa un estado, cada círculo negro representa un par estado-acción, y las flechas representan las recompensas [44].

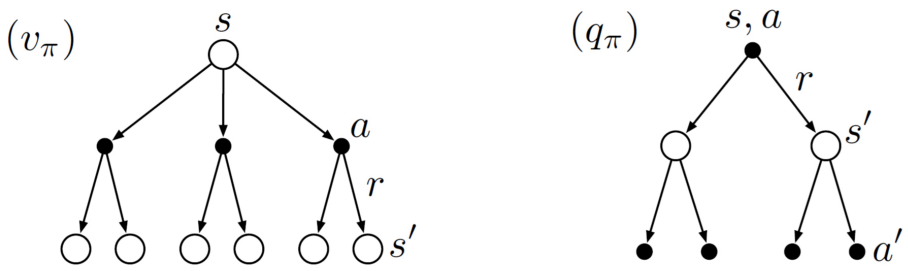


Figura 2.2: Diagrama *backup* para ν_π y q_π. Tomado de [44].

Las funciones valor $\nu_\pi(s)$ y acción-valor $q_\pi(s, a)$ evalúan respectivamente qué tan bueno es un estado y una pareja estado-acción cuando siguen una política π. Además, se puede definir una función de ventaja (*advantage function*) como:

$$A_\pi(s, a) = q_\pi(s, a) - \nu_\pi(s). \tag{2.9}$$

La función ventaja evalúa que tanto mejora seleccionar la acción a en lugar de utilizar la acción elegida por la política π, dicho de otro modo, es la recompensa extra que se obtiene al realizar la acción a. Si $A_\pi(s, a) > 0$ significa que tomar la acción a es mejor que seguir la política π, en caso de que $A_\pi(s, a) < 0$ significa que tomar la acción a es peor que seguir la política π.

- **Modelo del entorno**. Este elemento puede o no estar presente, y es cualquier entidad que permita al algoritmo de aprendizaje por refuerzo obtener información acerca del comportamiento del entorno. La presencia o ausencia de este elemento permite realizar una clasificación de los algoritmos de aprendizaje por refuerzo en algoritmos libres de modelo o basados en modelo.

  - En los algoritmos basados en modelo, se requiere de un modelo del entorno o bien, el agente intenta modelar el problema, i.e., aprender el Proceso de Decisión de Markov, una vez que se conocen las funciones de recompensa y transición, el problema puede resolverse mediante optimización.
  - En los algoritmos libres de modelo, no hay un conocimiento del entorno y por tanto, el agente tiene que reunir experiencia en forma de muestras, aprendiendo directamente cómo optimizar la recompensa, e.g., en la conducción autónoma no se conoce el modelo del entorno y además es difícil aprender un modelo del entorno [44, 45].

2.1. Procesos de Decisión de Markov (MDP)

En aprendizaje por refuerzo el agente toma una acción según el estado que recibe, para un paso de tiempo t, la interacción del agente con el entorno genera una secuencia de estados, acciones y recompensas $s_0, a_0, r_1, s_1, a_1, ..., r_{t-1}, s_{t-1}, a_{t-1}, r_t, s_t, a_t$. Cuando el entorno responde a la acción en el tiempo $t+1$ considera únicamente el estado y acción en el paso de tiempo anterior (s_t, a_t), i.e., el entorno no considera las acciones y estados anteriores al tiempo t, por lo tanto el estado s_{t+1} es únicamente dependiente del estado anterior s_t, entonces se asume que los estados tienen la propiedad de Markov, i.e., carecen de memoria. Entonces, el qué tan bien o cuánta recompensa está recolectando el agente no tiene ningún efecto sobre el entorno.

Por lo tanto, se puede definir cómo el entorno genera el estado y recompensa del siguiente paso de tiempo en función del estado y acción en el paso anterior:

$$p(s', r|s, a) = P(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a). \tag{2.10}$$

El problema de aprendizaje por refuerzo es un Proceso de Decisión de Markov (MDP por sus siglas en inglés *Markov Decision Process*) definido por una tupla de 5 elementos $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:

- $\mathcal{S}$: conjunto de estados.
- $\mathcal{A}$: conjunto de acciones.
- $\mathcal{P}$: función de transición que representa la dinámica del entorno en el siguiente paso $t+1$, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$.

  La función $\mathcal{P}$ define entonces la dinámica del MDP, p especifica la distribución de probabilidad para cada caso de s y a:

  $$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \ \forall \ s \in \mathcal{S}, a \in \mathcal{A}(s). \tag{2.11}$$

- $\mathcal{R}$: función de recompensas $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R} \to [0, 1]$, o en el caso determinista $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$.
- $\gamma \in [0, 1]$ es el factor de descuento.

Sin embargo, no siempre es fácil obtener el estado con toda la información del entorno en el tiempo t y, en algunos entornos solamente se puede obtener información parcial del mismo. La observación se define como información incompleta del estado. En este caso, el problema de aprendizaje por refuerzo se define como un Proceso de Decisión de Markov Parcialmente Observable (POMDP por sus siglas en inglés de *Partially Observable Markov Decison Process*) que está compuesto por una tupla de 7 elementos $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \Omega, \mathcal{O})$, los primeros 5 elementos son los mismos que en MDP, mientras que:

- Ω: es el conjunto de observaciones.
- $\mathcal{O}$: es la función de probabilidad de observación condicional, $\mathcal{O} : \mathcal{S} \times \Omega \rightarrow [0, 1]$. Define la probabilidad de observar o cuando se está en el estado s.

Para tratar el POMDP existen varios enfoques como inferir el estado a partir de la observación utilizando una distribución de probabilidad denominada estado de creencia, o bien el estado oculto puede simplemente ignorarse lo cual, puede generar ciertos problemas ya que observaciones similares pueden derivarse de estados diferentes [45].

2.2. Exploración y explotación

Uno de los principales desafíos en el aprendizaje por refuerzo radica en el equilibrio entre la exploración y la explotación, conocido como el dilema de exploración *vs* explotación. El agente debe explotar lo que ya sabe para obtener recompensas, pero también debe explorar para obtener información adicional.

Para abordar dicho dilema, existen varias estrategias, dos de las más conocidas son el enfoque ϵ-*greedy* y el enfoque *softmax*. En ϵ-*greedy* el agente elige la mejor acción con una probabilidad de $1 - \epsilon$ y una acción aleatoria con probabilidad ϵ. Si bien la explotación es esencial para maximizar las recompensas, la exploración continua es necesaria para aprender sobre el entorno y refinar las estimaciones de retorno par todas las combinaciones estado-acción, para esto se ha propuesto reducir el valor de ϵ a medida que el agente adquiere más conocimiento sobre el entorno, permitiéndole moverse hacia una explotación más intensiva a medida que avanza en su aprendizaje.

Por otro lado, el enfoque *softmax* permite diferenciar las acciones según su calidad, evaluada por los valores Q. La próxima acción es seleccionada en función de la distribución de Boltzmann, controlada por un parámetro de temperatura T, permitiendo una exploración más suave que ϵ-*greedy* [45].

Tanto ϵ-*greedy* como *softmax* funcionan en el caso de acciones discretas, para acciones continuas la exploración se suele hacer muestreando acciones aleatorias en función de la salida del modelo. Usualmente se utiliza la probabilidad gaussiana $\pi(a, s) \sim N(f(s), \sigma)$ [45].

2.3. *On-policy vs Off-policy*

En el aprendizaje por refuerzo, la elección entre métodos *on-policy* y *off-policy* implica un compromiso entre la estabilidad teórica y la eficiencia en la recopilación y el uso de datos. Los métodos *on-policy* requieren que el agente siempre explore dentro del contexto de su política actual, lo que garantiza cierta estabilidad teórica, pero puede limitar la convergencia hacia la política óptima. Por otro lado, los métodos *off-policy* superan esta limitación al utilizar dos políticas separadas: una para la recopilación

de datos y otra para la toma de decisiones, lo que los hace más eficientes en el uso de datos y acelera el proceso de entrenamiento. Sin embargo, esta flexibilidad puede conllevar a una menor estabilidad teórica en comparación con los métodos *on-policy*.

2.4. Clasificación de los principales algoritmos de RL

Los algoritmos de aprendizaje por refuerzo pueden clasificarse de diversas maneras según diferentes criterios, e.g., algoritmos basados en modelo y libres de modelo, basados en políticas y basados en valores, *on-policy* y *off-policy*. En la Fig. 2.3 se muestra la clasificación entre métodos tabulares y funciones de aproximación utilizadas en el presente trabajo [45]:

- **Métodos tabulares**. Estos métodos asumen un MDP finito, i.e., que se tiene un número finito de estados y acciones. Los algoritmos tabulares son adecuados para problemas con un número relativamente pequeño de estados y acciones.
- **Funciones de aproximación**. Estos métodos un MDP puede ser no finito, y utilizan una función paramétrica como una red neuronal para aproximar los valores de estado ν o estado-acción q.

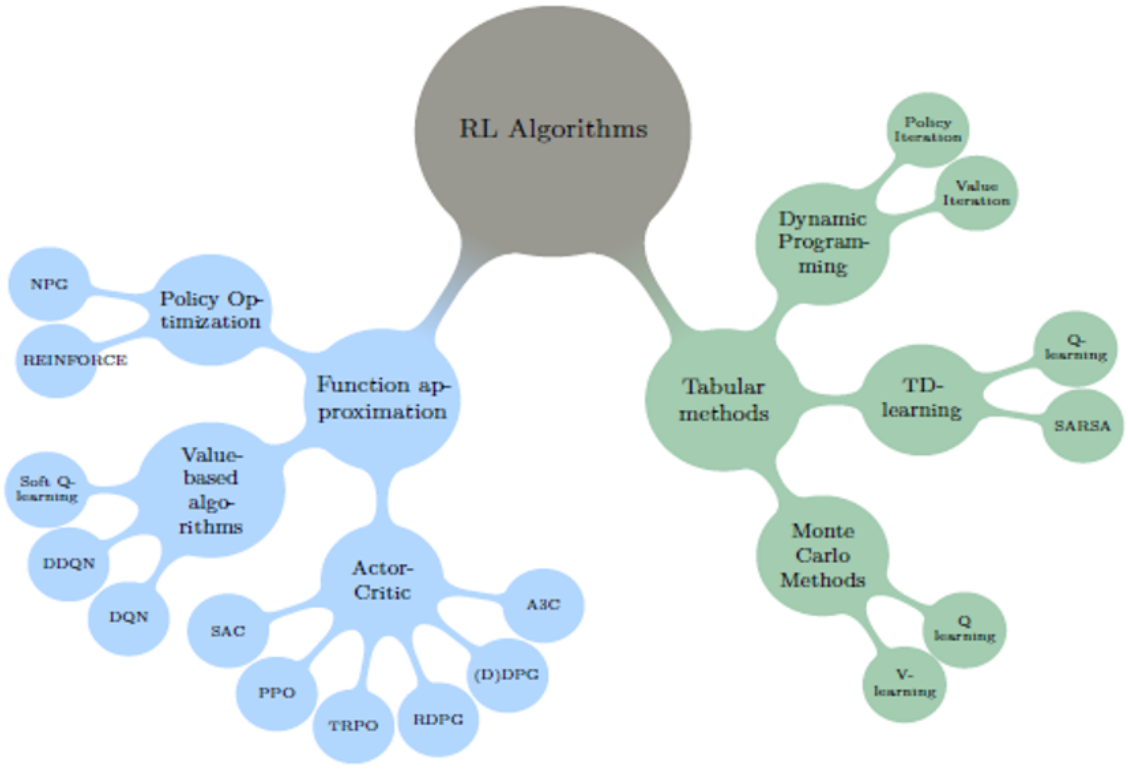


Figura 2.3: Clasificación de los principales algoritmos de RL. Tomado de [45].

2.4.1. Métodos tabulares

A continuación, se hará una breve introducción a los métodos tabulares, los cuales asumen que el MDP es finito, i.e., los conjuntos $\mathcal{S}, \mathcal{A}, \mathcal{R}$ son finitos. El objetivo de estos métodos es llenar una tabla o matriz para almacenar los valores de estado o estado-acción (Tabla 2.1). Estos enfoques se ajustan especialmente a situaciones en las cuales el número de estados y acciones es limitado, permitiendo abordar problemas con dimensiones pequeñas en los espacios de estados y acciones.

Tabla 2.1: Ejemplo de las tablas de valores.

Estado	Valor
s_0	10
s_1	5
$\vdots$	$\vdots$
s_n	15

(a) Tabla ν-valor.

Estado \ Acciones	$\uparrow$	$\downarrow$	$\leftarrow$	$\rightarrow$
s_0	7	6	5	6
s_1	8	7	8	9
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_n	10	8	9	9

(b) Tabla q-valor.

La idea entonces es ejecutar una gran cantidad de episodios y promediar los retornos obtenidos para cada estado o estado-acción, tal que la tabla ν-valor contiene el retorno esperado por estar en un estado s, mientras que la tabla q-valor contiene el retorno esperado si el agente está en un estado s y realiza una acción a, e.g., de la Tabla 2.1, si el agente sigue una política π, se encuentra en el estado s_1 y realiza la acción de ir hacia abajo $\downarrow$, entonces $q_\pi(s_1, \downarrow) = 7$.

El objetivo en una tarea de aprendizaje por refuerzo es encontrar una política que genere la mayor recompensa a largo plazo. Por lo tanto, se considera la política π' mejor que π si el retorno esperado es mayor o igual que el de la política π para todos los estados:

$$\nu_{\pi'}(s) \geq \nu_\pi(s) \rightarrow \pi' \geq \pi, \ \forall s \in \mathcal{S}. \tag{2.12}$$

Existe al menos una política óptima garantizada pero que puede ser no única π_* que satisface $\pi_* \geq \pi, \forall \pi$. La función de estado-valor óptima se denota como ν_* y puede definirse como [44]:

$$\nu_*(s) = \arg\max_\pi \nu_\pi(s), \forall s \in \mathcal{S} \tag{2.13}$$

$$= \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma \nu_*(s')]. \tag{2.14}$$

La Ec. (2.14) es una forma de la ecuación de optimalidad de Bellman donde la política se sustituye por un término de maximización. De manera similar se puede ob-

tener la función óptima de acción-valor q_*, definiendo así una forma de las ecuaciones de optimalidad de Bellman para q_*:

$$q_*(s,a) = \arg\max_{\pi} q_\pi(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2.15}$$

$$= \sum_{s',r} p(s',r|s,a)[r + \gamma \arg\max_{a'} q_*(s',a')]. \tag{2.16}$$

En la Fig. 2.4 se observan las ecuaciones de optimalidad de Bellman para ν_* y q_*, donde se muestra que ahora en los puntos de decisión el agente escoge la acción que maximiza la retorno esperado.

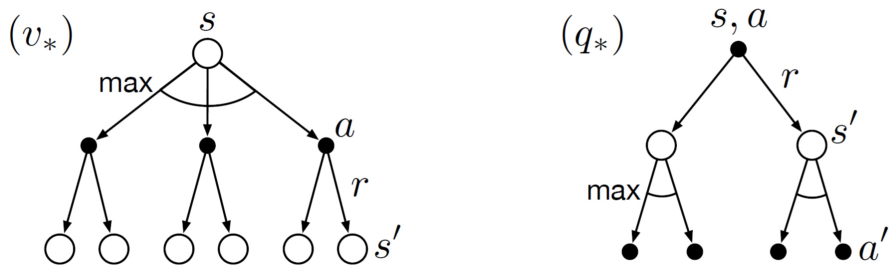


Figura 2.4: Diagrama *backup* para ν_* y q_*. Tomado de [44].

Programación dinámica

La programación dinámica es utilizada para calcular una política óptima π_* de un MDP finito con un conocimiento completo del entorno, i.e., se conocen los estados, las acciones, la función de recompensa y la función de transición. Al tratarse de un MDP finito se tiene un número finito de políticas, y se puede encontrar la política óptima π_* de forma iterativa sin la interacción real del agente con el entorno. Los algoritmos con este enfoque se clasifican con el término de programación dinámica, siendo iteración de política el método más popular el cual de manera iterativa mejora la política π hasta encontrar la política óptima π_*, y consta de dos fases: evaluación de la política y mejora de la política.

El primer paso es evaluar la política actual π al calcular su función valor $\nu(s_t)$. La siguiente fase que es la de mejora de la política, que estima el valor de la nueva política π' seleccionando la acción que parece mejor según $q_\pi(s,a)$. Ambas fases trabajan en conjunto hasta que se encuentran una política π y una función valor ν que no cambien en ninguna de las fases, i.e., que ya no pueden mejorarse, esto significa que se ha alcanzado un punto fijo, siendo éste la política óptima π_*, de lo contrario, la sucesión de evaluación y mejora de la política se realiza de nuevo, la Fig. 2.5 muestra dicha interacción de las fases hasta converger a π_* y ν_*.

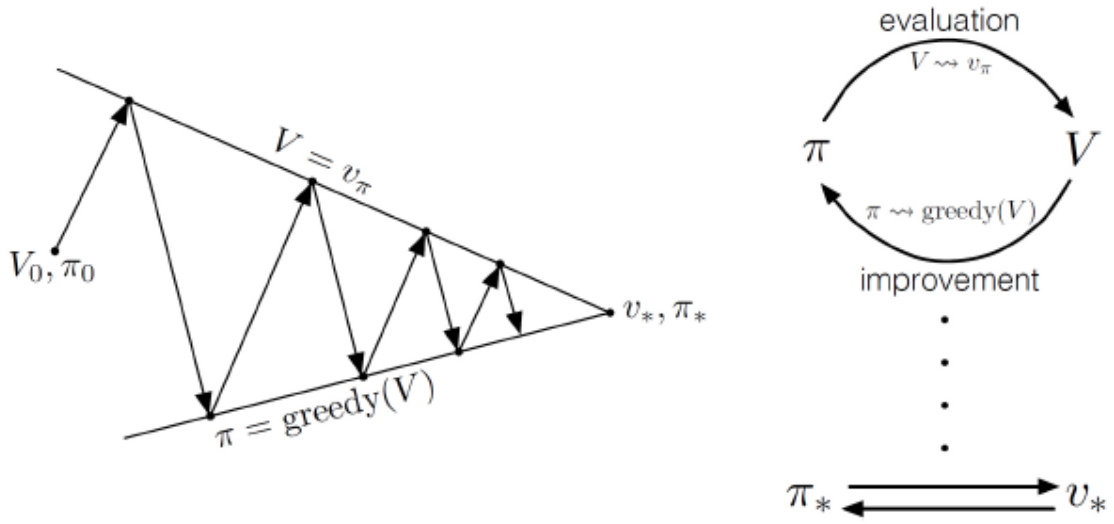


Figura 2.5: Iteración de política. Tomado de [44].

Métodos Monte-Carlo

El método Monte-Carlo (MC) es libre de modelo (i.e., no existe un conocimiento previo del entorno, así como de la función de transición), además es un método que sirve para aproximar q_π y ν_π al aprender de episodios completos (i.e., todos los episodios deben terminar) de experiencia.

El objetivo entonces, es aprender q_π o ν_π de episodios completos de experiencia siguiendo la política π, al calcular los promedios de retornos observados durante la interacción con el entorno. Para un estado específico s, la función acción-valor (de igual manera se realiza para la función valor $V(S_t)$, se actualiza con la siguiente ecuación:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \underbrace{[G_t - Q(S_t, A_t)]}_{\delta_t}, \tag{2.17}$$

donde δ_t es el término de error y $\alpha \in [0, 1]$ es un parámetro que controla la tasa de aprendizaje, buscando que al realizar un número de experimentos suficientemente grandes se podrá estimar el valor de q_π. Valores muy bajos de α incentiva al agente a considerar una historia más larga de retornos al estimar la función acción-valor, mientras que incrementar α incentiva al agente a centrarse más en los retornos muestreados más recientes. Dicho de otra forma, valores más altos de α resulta en un aprendizaje más rápido, pero valores demasiado altos de α pueden impedir que MC converja a π_*.

Una característica adicional de MC es que no utiliza el proceso de *bootstrapping*, i.e., no actualiza sus estimaciones basándose en las estimaciones de estados sucesores, en lugar de eso, confía en la experiencia real muestreada durante la interacción con el entorno.

Diferencias Temporales

El método de Diferencias Temporales (TD) es la base de muchos algoritmos avanzados, combinando elementos de los métodos MC y programación dinámica. A diferencia de los métodos MC, los algoritmos TD no esperan hasta el final de un episodio para estimar los valores q_π, ν_π, lo que les otorga la ventaja de ser más rápidos en la actualización de valores durante la interacción con el entorno. La actualización de la función acción-valor se realiza con la siguiente ecuación (SARSA):

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \underbrace{[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]}_{\delta_t}, \tag{2.18}$$

donde α es la tasa de aprendizaje, δ_t es el error TD, $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$ es una estimación alterna y $Q(S_t, A_t)$ es un el estimado actual. Según como se calcule la estimación alterna se obtienen tres algoritmos ampliamente utilizados: SARSA que es un enfoque de política (*on-policy*), *Q-learning* que es un enfoque fuera de política (*off-policy*) y *Expected* SARSA que es *on-policy*.

El nombre de SARSA viene de la tupla $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$. En cada paso de tiempo, el agente recibe una recompensa y transita a un nuevo estado. Posteriormente, el agente selecciona otra acción de acuerdo con su política y actualiza la tabla Q con la recompensa recibida y el valor Q del nuevo estado y la nueva acción.

En contraste, mientras que SARSA selecciona la acción a realizar mediante ϵ-*greedy*, en *Q-learning* se utiliza una política *greedy*, i.e., la actualización se realiza con la acción que maximiza Q para el nuevo estado:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right]. \tag{2.19}$$

El método *Expected* SARSA usa el valor esperado del siguiente par estado-acción, considerando la probabilidad de que el agente selecciona cada posible acción posible. Generalmente consigue mejores resultados que SARSA, y su ecuación de actualización es la siguiente:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_{a \in A} \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \right]. \tag{2.20}$$

De manera general, el agente que aprende con métodos TD comienza a tomar mejores decisiones desde el primer momento. Una de las ventajas de *off-policy* es la separación del aprendizaje de la exploración, permitiendo una exploración más intensa.

Las diferencias temporales en n pasos es una familia de métodos que aprenden a base de la experiencia, utilizando *bootstrapping* en n pasos, es decir, el agente

interactúa n pasos y obtiene n recompensas, obteniendo una estimación del retorno en n pasos:

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n}). \tag{2.21}$$

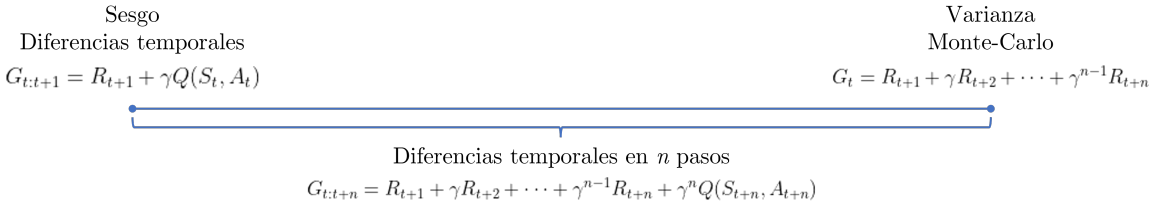
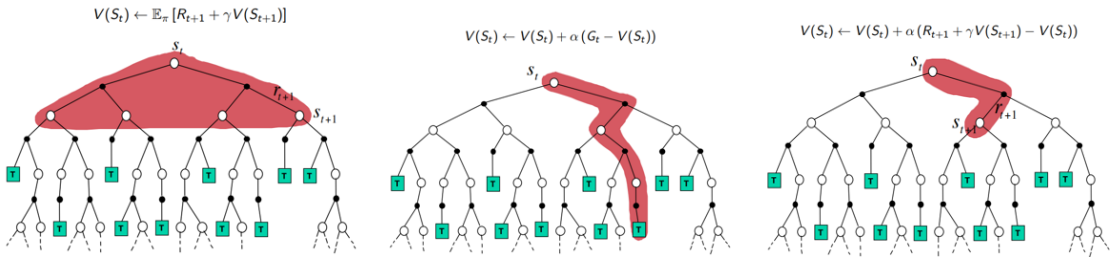


Figura 2.6: Diferencias temporales en n pasos

La elección de n determina la cercanía a cada uno de estos métodos, tal como se muestra en la Fig. 2.6. A mayor número de pasos n, menor es el sesgo, pero es mayor la varianza, y viceversa. Este método retiene la ventaja de los métodos TD de aprender de forma constante y uniforme, al mismo tiempo que nos da control sobre el sesgo y la varianza de las estimaciones. En la práctica, los valores intermedios de n suelen tener mejores resultados.

En la Fig. 2.7 se muestra un esquema de la manera en que los distintos métodos revisan para obtener la función valor máxima.



(a) Programación dinámica (b) Monte-Carlo (c) Diferencias Temporales

Figura 2.7: Comparativa entre los distintos métodos. En programación dinámica se revisan todos los hijos del árbol y se obtiene el máximo, en Monte-Carlo se revisa toda la rama de inicio a fin, y en Diferencias Temporales solamente da un vistazo a n pasos adelante. Tomada de [46].

2.4.2. Aproximación de funciones

Los métodos tabulares suponen un MDP finito, si el número de estados y acciones es limitado y pequeño es posible representar la función acción-valor en una tabla, diccionario u otra estructura finita. Sin embargo, en problemas con un espacio de estados y/o acciones muy grandes e inclusive infinitos, prácticamente se vuelve imposible representarlos en una estructura finita debido a la limitación de memoria y

a la dificultad de buscar valores en tiempo real. Para abordar este desafío en MDPs grandes o infinitos se recurre a la aproximación de funciones. Los algoritmos de esta clase no convergerán necesariamente a la solución óptima como ocurre en los métodos tabulares [44, 45].

La aproximación de funciones se basa en generalizar las experiencias previas con estados o acciones para estimar funciones de valor. En lugar de mantener un registro de cada valor individual se utilizan algoritmos que toman muestras de datos de la función deseada (e.g., $Q(s, a)$) y tratan de generalizar a partir de dichas muestras para obtener una aproximación de toda la función mediante una serie de parámetros entrenables θ [44]:

$$\widehat{\nu}(s;\theta) \approx \nu_\pi(s), \tag{2.22}$$

$$\widehat{q}(s,a;\theta) \approx q_\pi(s,a). \tag{2.23}$$

En la Fig. 2.8 se muestra que la aproximación de funciones toma como entrada el conjunto de estados y/o acciones para aproximar las funciones $\widehat{\nu}(s;\theta)$, $\widehat{q}(s,a;\theta)$ dados una serie de parámetros entrenables θ, que en la figura están declarados como **w**.

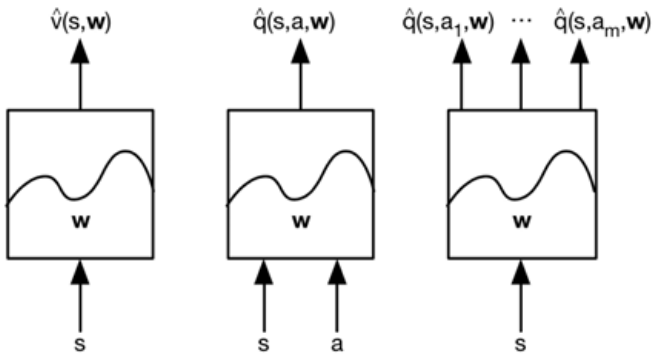


Figura 2.8: Variantes de aproximación de funciones. Tomada de [47].

En la práctica, las redes neuronales profundas (*Deep Neural Networks* o DNNs) han demostrado ser una herramienta eficaz para la aproximación de funciones de aprendizaje por refuerzo, esto es por su capacidad de aprender representaciones complejas de los datos. La combinación del aprendizaje por refuerzo con DNNs se le conoce como aprendizaje por refuerzo profundo o *Deep Reinforcement Learning* (DRL), permitiendo abordar problemas en los que el espacio de estadios es continuo o prácticamente infinito.

Los algoritmos de aproximación de funciones se pueden dividir en tres categorías [45]:

- **Algoritmos basados en valor (*value-based*)** se basan en inferir Q para cada par estado-acción para después inferir la política óptima a partir de ella. En la

aproximación de funciones, el espacio de estados puede ser muy grande, pero para utilizar los algoritmos basados en valores el espacio de acciones tiene que ser finito.

- **Algoritmos basados en la política (*policy-based*)** permiten tener un espacio de acciones grande o infinito calculando directamente la política.
- **Actor crítico (*actor-critic*)** se utilizan en lugar de los algoritmos basados en la política porque son más estables y calculan tanto la política como el valor del estado al mismo tiempo.

Algoritmos basados en valor (*value-based*)

Los algoritmos basados en valor se basan en el cálculo del valor Q para cada par estado-acción. Al igual que *Q-learning* y SARSA, se basan en el hecho de que conocer la función Q es suficiente para obtener la política óptima $\pi_*(a) = \arg\max_a Q(a, s)$. En esta sección se explorará el algoritmo *Deep Q-learning* que es el más famoso, así como sus variantes. El algoritmo calcula Q en el caso de un espacio de estados no finito [45].

Deep Q-learning

Deep Q-Learning fue introducido por Mhnih en 2013 con el artículo *Playing Atari with Deep Reinforcement Learning* [48], en el cual implementan *Q-learning* con imágenes de entrada. En estos algoritmos las estimaciones de los q-valores se producen a través de una red neuronal, tomando como entrada un estado (Fig. 2.9), además de que aprenden de la experiencia recolectada por el agente al interactuar con el entorno, como en los métodos SARSA y *Q-learning*.

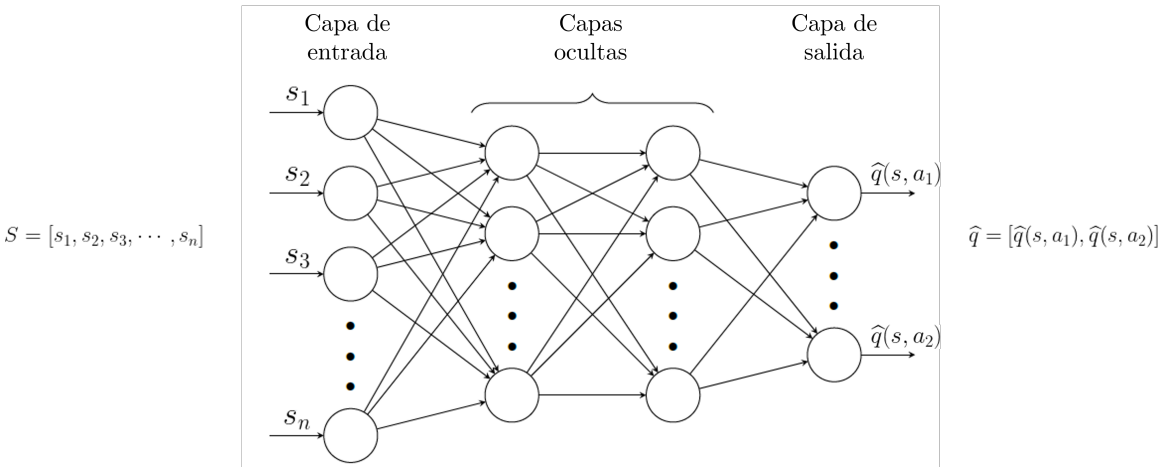


Figura 2.9: Esquema de la red neuronal en Deep Q-learning.

Muchos algoritmos de aprendizaje suponen que los datos son independientes e idénticamente distribuidos (iid), pero no es el caso en el aprendizaje por refuerzo,

además es inestable cuando se utilizan redes neuronales para representar el valor de las acciones [45]. El algoritmo *Deep Q-Learning* aborda estos retos utilizando dos características: la repetición de experiencias (*experience replay*) [48, 49] y fija los objetivos *Q-targets* (*fixed- Q-targets*) [50].

La repetición de experiencias se basa en la idea de que se puede aprender mejor si hacemos múltiples pasadas sobre la misma experiencia. Se basa en construir una base de datos (búfer o *buffer*) que contenga una colección de experiencias en forma de tuplas (S, A, R, S'). Las tuplas se van almacenando durante el entrenamiento cada vez que el agente interactúa con el entorno. El *buffer* tiene un tamaño limitado y al llenarse, se sustituyen las experiencias antiguas por las nuevas.

Al tratarse de estados secuenciales, los estados sucesivos están fuertemente correlacionados. Para eliminar la correlación entre estados, durante el entrenamiento las actualizaciones del modelo (i.e., la red neuronal) se realizan con mini-lotes muestreados aleatoriamente del *buffer*.

Para que el proceso de aprendizaje sea estable, se realiza una copia de los parámetros θ de la red neuronal, generando θ_{target} que son los pesos de una red neuronal *target* separada que no cambia durante el paso de aprendizaje.

Para que el proceso de aprendizaje sea estable, se realiza una copia de la red neuronal para calcular los valores *target* ($\theta_{target} \leftarrow \theta$), esta red no cambia en el paso de aprendizaje.

El algoritmo entonces busca entrenar los parámetros de una red neuronal, en lugar de modificar los valores de una tabla. La función de pérdida que se busca minimizar es el error cuadrado medio:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left[y_i - \widehat{y}_i \right]^2 , \tag{2.24}$$

$$\mathcal{L}(\theta) = \frac{1}{|K|} \sum_{i=1}^{|K|} \left[\underbrace{r_i + \gamma \max_{a'} \widehat{q}(s_{i+1}, a'; \theta_{target})}_{target \rightarrow y_i} - \underbrace{\widehat{q}(s, a; \theta)}_{\widehat{y}_i} \right]^2 , \tag{2.25}$$

donde y_i es el *target*, $\widehat{y}_i$ es la estimación, $K = (S, A, R, S') \sim B$ que es una muestra aleatoria del *buffer*. Una vez evaluada la función de pérdida se actualizan los parámetros θ de la red.

En conclusión, el algoritmo *Deep Q-learning* usa dos redes neuronales separadas con la misma arquitectura. Los pesos de la red *target* (θ_{target}) se actualizan con menor frecuencia que los pesos de la red original (θ), y al fijar los objetivos *Q-targets* se obtiene un entrenamiento más estable.

Mejoras y variantes a Deep Q-learning

En esta sección se abordarán algunos trabajos posteriores que mejoran el algoritmo de *Deep Q-learning*. El problema del DQN clásico está en que y_i (Ec. 2.25) también es una función de Q, lo que puede provocar un comportamiento inestable y una sobreestimación de los valores [45], esto es porque es posible que la operación max se equivoque, especialmente en las etapas iniciales, dado que los q-valores siguen evolucionando y es posible que no se haya reunido suficiente información para determinar la mejor acción. La precisión de los q-valores tiene una alta dependencia de las acciones que se hayan probado, así como de los estados vecinos explorados.

El algoritmo *Double Q-Learning* o *Double DQN* (DDQN) [51, 52] realiza estimaciones más robustas al seleccionar la última acción usando un conjunto de parámetros θ (selecciona la mejor acción), pero evaluándolo con un conjunto distinto de parámetros θ' (evalúa la acción seleccionada). El valor de y_i para DDQN es:

$$y_i^{DDQN} = r_i + \gamma \widehat{q}\left(s', \max_{a'} \widehat{q}\left(s', a; \theta\right); \theta'\right). \tag{2.26}$$

DDQN es una versión más eficiente y estable de DQN [52]. Otra mejora está en la repetición de experiencias, dado que algunas experiencias son más relevantes que otras. En el método *Prioritized Experience Replay* [53] las transiciones tienen probabilidades de ser seleccionadas, buscando muestrear con mayor frecuencia las transiciones más importantes o prioritarias del *buffer*. La prioridad de una muestra está en función de la magnitud del error TD $\delta = r_{t+1} + \gamma \max_{a'} \widehat{q}\left(s_{t+1}, a'\right) - q\left(s_t, a_t\right)$, mejorando la estabilidad y eficiencia de la red Q [45].

El algoritmo *Hindsight Experience Replay* (HER) [54] busca abordar el tema de las recompensas esparsas (pocas y alejadas), en entornos en los que la exploración no siempre es suficiente para alcanzar el objetivo, y sin ajustar o afinar la función de recompensa. La idea principal de HER es reproducir episodios fallidos cambiando el objetivo inicial por el estado que el agente alcanzó realmente [45].

Además, hay trabajos que exploran nuevas arquitecturas como Dueling DQN [55], que es una arquitectura con dos secuencias, uno estima el valor del estado $V(s)$ y el otro estima la función de ventaja $A(s, a)$ (Ec. 2.9). Después, ambas secuencias se combinan para producir los valores Q-valores, tal que $Q(s, a) = V(s) + A(s, a)$. La idea clave de la arquitectura es que, en muchos entornos, las acciones sólo importan a veces, y al calcular explícitamente tanto el valor como la ventaja en lugar de calcular directamente Q, la red neuronal aprenderá en qué estado importa realmente la acción. En pocas palabras, la función de ventaja estima si una acción es mejor que la media y, por tanto, determina si la acción tiene poca importancia en el estado considerado [45].

En esta sección se abordaron algunas de las mejoras propuestas para *Deep Q-learning*, pero existen más trabajos (*Multi-step Bootstrap*, *Distributional DQN*, *Noisy DQN*). El algoritmo *Rainbow* [56] combina varias de las mejoras de DQN como:

DDQN, *Prioritized Experience Replay*, *Dueling DQN*, *Distributional DQN*, *Noisy DQN*.

Algoritmos basados en la política (*policy-based*)

Los algoritmos basados en valor primero estiman la calidad de un estado y después construyen una política que conduce a buenos estados, es decir, estos algoritmos no dicen que acción se debe tomar explícitamente, sino que indican cuánta recompensa recibirá el agente desde cada estado o estado-acción, por lo tanto, se selecciona la mejor opción según la calidad del estado o estado-acción utilizando estrategias como ϵ-*greedy*.

Los algoritmos basados en la política buscan optimizar directamente la política que decida qué acción tomar en cada momento de forma explícita, sin necesidad de aprender la función valor $V(s)$ o acción-valor $Q(s,a)$, definiendo una política que decide las probabilidades de tomar cada acción para cada estado $\pi(a|s)$. En la Fig. 2.10 se muestra el comparativo de los pasos en los algoritmos basados en valor y en política. Las ventajas de los algoritmos basados en la política son [45, 57, 58]:

- Capacidad de representar acciones continuas, siendo mejores para entornos estocásticos y entornos con acciones continuas o de muchas dimensiones [2].
- Optimizan directamente la función de política, sin intermediarios.
- Capacidad de tratar con políticas estocásticas.
- Convergencia más rápida.

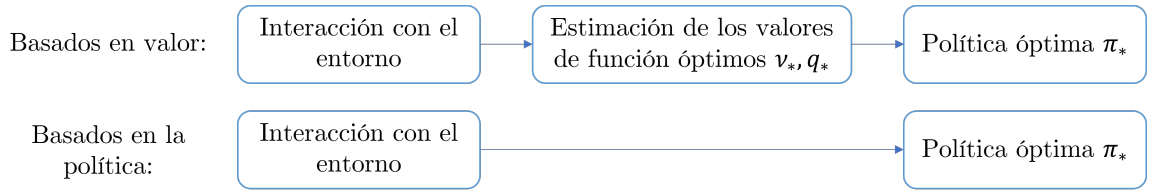


Figura 2.10: Comparativo de los pasos de los algoritmos basados en valor y basados en la política. Los algoritmos basados en la política optimizan directamente la política π.

Existen varios métodos de aproximación de políticas (e.g., algoritmos evolutivos, *hill climbing*, recocido simulado) pero los métodos de gradiente de política (*Policy Gradient Methos*) son los más utilizados debido a su eficacia [59], siendo un subconjunto de los algoritmos basados en la política. En el presente trabajo se enfocará en

[2]Deep Q-learning es un algoritmo eficaz pero únicamente puede aplicarse en espacios de acción discretos. Para el caso de acciones continuas es posible discretizarlas, pero se enfrenta a la maldición de la dimensionalidad: el número de acciones posibles crece exponencialmente con el número de dimensiones de la acción [45].

los métodos gradiente de política.

Los métodos de gradiente de política utilizan un aproximador de funciones parametrizado con un vector de parámetros θ (clásicamente para éstos métodos es una red neuronal con sus pesos y sesgo) para estimar las probabilidades de tomar cada acción dado un estado $\pi_\theta(a|s) = P(a|s;\theta)$. En pocas palabras, la red neuronal es la política. En el caso de un espacio discreto de acciones, la salida de la red neuronal será un vector de probabilidades sobre todas las acciones posibles, mientras que en el caso de un espacio continuo de acciones, la salida de la red neuronal es la media (y potencialmente la varianza) de una distribución gaussiana [45, 59].

Una trayectoria τ está compuesta por el proceso secuencial de toma de decisiones $\tau = (s_1, a_1, ..., s_T, a_T)$. Al muestrear las acciones, la política π influye en la probabilidad en la que se observa cada secuencia posible de estados y acciones a lo largo del horizonte temporal T. Cada trayectoria tiene una probabilidad de ocurrir $P(\tau)$ y una recompensa asociada $R(\tau)$.

El objetivo es encontrar una política óptima (*optimal policy*) que maximice el retorno esperado en una trayectoria τ. La función objetivo $J(\theta)$ se define como:

$$J(\theta) = \mathbb{E}_{\tau\sim\tau_\theta}\left[R(\tau)\right] = \int P(\tau;\theta)R(\tau)d\tau, \tag{2.27}$$

donde $\tau \sim \tau_\theta$ representa la distribución de las trayectorias bajo la política π_θ. En la Ec. 2.27 se están sumando[3] las probabilidades de cada trayectoria multiplicadas por su respectiva recompensa. El objetivo es encontrar los parámetros θ que maximicen la función objetivo:

$$\max_\theta J(\theta) = \max_\theta \int P(\tau;\theta)R(\tau)d\tau. \tag{2.28}$$

Al utilizar una política estocástica, se muestrean una variedad de acciones que producen diferentes trayectorias, permitiendo observar cuáles acciones producen las mejores recompensas. Dichas observaciones proporcionan una dirección de actualización para los parámetros θ de la política π_θ. La dirección se obtiene utilizando el gradiente de la función objetivo:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau\sim\tau_\theta}\left[R(\tau)\right] \tag{2.29}$$

$$= \int \nabla_\theta P(\tau;\theta)R(\tau)d\tau. \tag{2.30}$$

A continuación se aplican un par de artilugios matemáticos, el primer es multiplicar la Ec. 2.30 por $P(\tau;\theta)/P(\tau;\theta)$ y el segundo es el uso de una entidad logarítmica:

[3]En un conjunto finito de trayectorias se realiza la sumatoria, mientras que en un conjunto infinito se utiliza la integral.

$$\nabla_\theta J(\theta) = \int P(\tau;\theta) \underbrace{\frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)}}_{\nabla_\theta \log P(\tau;\theta)} R(\tau) d\tau \tag{2.31}$$

$$= \int P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) R(\tau) d\tau \tag{2.32}$$

$$= \mathbb{E}\left[\nabla_\theta \log P(\tau;\theta) R(\tau)\right]. \tag{2.33}$$

El término que hace falta tratar es $P(\tau;\theta)$, en un entorno totalmente determinista, se podría calcular la trayectoria que produce cada política π_θ y encontrar la política que produce la mayor recompensa acumulada. Sin embargo, la mayoría de los problemas de aprendizaje por refuerzo tienen un componente estocástico, por lo que se calcula la probabilidad de que se produzca una determinada trayectoria de recompensa dada la política. En resumen, se tienen un par de distribuciones de probabilidad: la política $\pi_\theta = P(a|s;\theta)$ y la transición de estados $P(s_t+1|s_t,a_t)$. En cada paso de tiempo se multiplica la probabilidad de muestreo de la acción $\pi_\theta(a_t|s_t)$ con la probabilidad de transición $P(s_t+1|s_t,a_t)$, y posteriormente se multiplican dichas probabilidades por cada paso temporal para hallar la probabilidad de que ocurra la trayectoria [59]:

$$P(\tau;\theta) = \left[\prod_{t=0}^{T} P(s_{t+1}|s_t,a_t) \cdot \pi_\theta(a_t|s_t)\right]. \tag{2.34}$$

Existen un par de problemas para calcular $P(s_{t+1}|s_t,a_t)$ porque requiere de un modelo explícito del entorno (en muchas ocasiones no se cuenta con el entorno), y el segundo es que al ser un producto de probabilidades, para horizontes temporales largos y probabilidades pequeñas, las probabilidades de trayectoria se vuelven demasiado pequeñas, provocando inestabilidad numérica. Sustituyendo la expresión de $P(\tau;\theta)$ en $\nabla_\theta \log P(\tau;\theta)$ y utilizando propiedades de los logaritmos:

$$\nabla_\theta \log P(\tau;\theta) = \nabla_\theta \log \left[\prod_{t=0}^{T} P(s_{t+1}|s_t,a_t) \cdot \pi_\theta(a_t|s_t)\right] \tag{2.35}$$

$$= \nabla_\theta \left[0 \sum_{t=0}^{T} \log P(s_{t+1}|s_t,a_t) + \sum_{t=0}^{T} \log \pi_\theta(a_t|s_t)\right] \tag{2.36}$$

$$= \nabla_\theta \sum_{t=0}^{T} \log \pi_\theta(a_t|s_t). \tag{2.37}$$

La Ec. 2.37 resuelve el tema de la estabilidad numérica, evitando la explosión y/o desvanecimiento de gradientes, esto se debe a que por la propiedad de los logaritmos las probabilidades son aditivas y ya no multiplicativas. Además, de que se elimina el término $P(s_t+1|s_t,a_t)$ porque el gradiente es respecto a θ, y ese término no depende

de θ, es decir, ya no es necesario tener el modelo explícito del entorno. Sustituyendo la Ec. 2.37 en la Ec. 2.33:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log P(\tau;\theta) R(\tau)\right] \tag{2.38}$$

$$= \mathbb{E}\left[\nabla_\theta \sum_{t=0}^{T} \log \pi_\theta(a_t|s_t) R(\tau)\right] \tag{2.39}$$

$$= \mathbb{E}\left[\left(\nabla_\theta \sum_{t=0}^{T} \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=0}^{T} r(s_t, a_t)\right)\right]. \tag{2.40}$$

Las ecuaciones hasta el momento tratan sobre la expectativa real, es decir, incluye todas las trayectorias posibles. Los problemas típicos de aprendizaje por refuerzo son intratables desde el punto de vista computacional (de lo contrario, se podría aplicar simplemente programación dinámica), por lo que es necesario trabajar con una muestra de trayectorias [59]. La estimación de un gradiente aproximado a partir de un número finito de muestras N se observa en la siguiente ecuación:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i}^{N} \left[\left(\nabla_\theta \sum_{t=0}^{T} \log \pi_\theta(a_t^i|s_t^i)\right)\left(\sum_{t=0}^{T} r(s_t^i, a_t^i)\right)\right]. \tag{2.41}$$

La Ec. 2.41 es un expresión manejable, en donde se muestrean las trayectorias y sus respectivas recompensas, aunque no se tenga conocimiento del modelo del entorno. Lo único que se necesita es una política definida explícitamente y diferenciable respecto a θ.

Para actualizar los parámetros θ de la política se utiliza el ascenso de gradiente para mover los parámetros en dirección tal que aumente la esperanza del retorno. La regla de actualización es:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta). \tag{2.42}$$

El algoritmo REINFORCE [60] (gradiente de política + Monte-Carlo) se puede utilizar para resolver entornos con espacios continuos de acciones, pero se realizan modificaciones para mejorar su desempeño. Además, REINFORCE incrementa la probabilidad de las buenas acciones y disminuye la probabilidad de las malas acciones.

El principio clave de la política de gradiente es asignar una alta probabilidad a las acciones cuando las recompensas son altas y viceversa. La Ec. 2.40 tiene el problema de que todas las acciones de una trayectoria son tratadas igual. Además, el gradiente de la política tiene una varianza alta y una convergencia lenta. La primera modificación para mejorar el método, es agregar causalidad, lo que significa que la política en el momento t sólo puede afectar a las recompensas en el tiempo t' si $t' > t$ [45], partiendo de la Ec. 2.41:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \right) \left[\sum_{t'=t}^T r(s_{t'}^i, a_{t'}^i) \right]. \tag{2.43}$$

Con esta nueva ecuación, las acciones mejoran proporcionalmente a las recompensas futuras [45]. La segunda modificación es para reducir la varianza al añadir una línea base (*baseline*), de forma que sólo se refuercen las acciones que sean mejores que la media, una línea base muy común es la suma esperada de recompensas [45]:

$$b = \frac{1}{N} \sum_{i=0}^N r(\tau^i). \tag{2.44}$$

Es posible añadir la línea base dado que [45]:

$$\begin{aligned}
\mathbb{E}_\pi \left[\nabla_\theta \log \pi_\theta(\tau) b \right] &= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) b d\tau && (2.45)\\
&= \int \nabla_\theta \pi_\theta(\tau) b d\tau && (2.46)\\
&= b \nabla_\theta \int \pi_\theta(\tau) d\tau && (2.47)\\
&= b \nabla_\theta 1 && (2.48)\\
&= 0. && (2.49)
\end{aligned}$$

El gradiente entonces se calcula como:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \right) \left[\sum_{t'=t}^T r(s_{t'}^i, a_{t'}^i - b) \right]. \tag{2.50}$$

Actor-Crítico (*Actor-Critic*)

Los algoritmos de Actor-Crítico buscan combinar lo mejor de los algoritmos basados en valor y los basados en política. Como se mencionó con anterioridad existe un compromiso entre el sesgo y la varianza, donde se busca: evitar una alta varianza y alto sesgo, y procurar una baja varianza y un bajo sesgo. Los algoritmos de Actor-Crítico utilizan las técnicas de los algoritmos basados en valor para reducir la varianza asociada a los métodos basados en política, siendo una intersección entre ambos tal como se muestra en la Fig. 2.11.

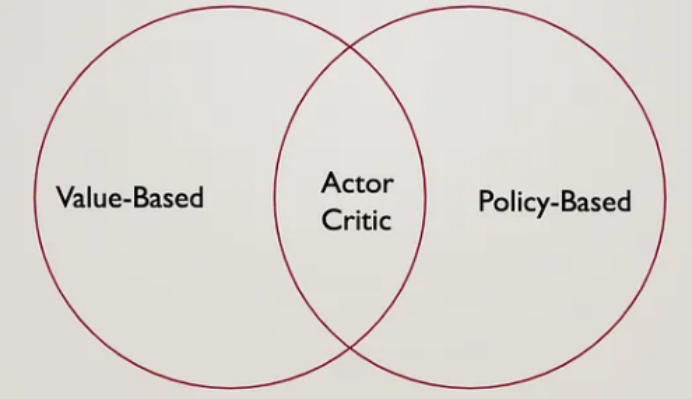


Figura 2.11: Iteración de política. Tomado de [57].

En los métodos tabulares se exploraron un par de formas para estimar los retornos esperados (véase Fig. 2.6):

- Monte-Carlo (REINFORCE), donde se suman las recompensas para un episodio completo, y después se promedian los estimados para la colección de episodios recolectados por el agente, implicando una mayor varianza, pero sin sesgo.
- Diferencias Temporales (Actor-Crítico) que tiene baja varianza y bajo sesgo, permitiendo al agente aprender más rápido, pero con más problemas para converger.

Los algoritmos Actor-Crítico aprenden al ajustar la probabilidad de realizar buenas o malas acciones. El Actor (algoritmo basado en política) realiza acciones bajo la política $\pi(a|s;\theta)$ en el entorno, y el Crítico (algoritmo basado en valor) estimará que tan buenas o malas fueron las acciones del actor al estimar el valor de los estados ($\widehat{\nu}(s;\phi)$, $\widehat{q}(s,a;\phi)$) y refinando la estimación de la ventaja, permitiendo eficientar el aprendizaje del agente.

En la Ec. 2.43 se muestra la aproximación del gradiente con un número finito de muestras. La idea de dicha ecuación es obtener la dirección donde se encuentran buenas recompensas. Sin embargo, el término $\sum_{t'=t}^{T} r(s^i_{t'}, a^i_{t'})$ es una estimación de una muestra única de las recompensas futuras partiendo del estado s_t y realizando la acción a_t, generando una alta varianza en el gradiente de la política. El objetivo es reducir la varianza, por lo que sería más eficiente estimar mejor las recompensas futuras partiendo del estado s_t y realizando a_t, y como se discutió anteriormente, un buen estimador es el Q-valor $Q_\pi(s,a) = \mathbb{E}[R_t|S_t = s, A_t = a]$ [45, 61]:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i}^{N}\sum_{t=0}^{T}\left(\nabla_\theta \log \pi_\theta(a^i_t|s^i_t)\right) Q(s^i_t, a^i_t). \tag{2.51}$$

Como se mencionó, agregar una línea base (e.g., la suma de las recompensas) permite reducir la varianza. En este caso, se puede calcular la una media más precisa utilizando los Q-valores:

$$b = \frac{1}{N}\sum_{i=0}^{N} Q(s_t^i, a_t^i) \approx \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}\left[Q(s_t, a_t)\right] = V(s_t). \tag{2.52}$$

El cálculo del gradiente agregando la línea base se convierte en:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i}^{N}\sum_{t=0}^{T}\left(\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\right)\left[Q(s_t^i, a_t^i) - V(s_t^i)\right]. \tag{2.53}$$

La función de ventaja $A(s,a) = Q(s,a) - V(s)$ indica que tanto la acción a es mejor que la acción promedio, por lo tanto, sustituyendo la función de ventaja en el gradiente:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i}^{N}\sum_{t=0}^{T}\left(\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\right)\left[A(s_t^i, a_t^i)\right]. \tag{2.54}$$

Para calcular la función de ventaja, se puede utilizar la aproximación $Q(s_t, a_t) \approx r(s_t, a_t) + \gamma V(s_{t+1})$, por lo tanto [45]:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) = r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t). \tag{2.55}$$

En resumen, los algoritmos Actor-Crítico tienen dos componentes con distintas funciones de activación, es decir, se tienen dos redes neuronales distintas. El primer componente es el actor que es la política π parametrizada por θ, y el segundo que es el crítico (Q,V o A dependiendo del algoritmo) parametrizado por ϕ que estima la calidad de la salida o *output* del actor [45]. A continuación, se abordarán algunos algoritmos de Actor-Crítico, de manera no exhaustiva.

Gradiente de política determinista (*Deterministic Policy Gradient*): DDPG, RDPG, D4PG

El algoritmo *Deep Deterministic Policy Gradient* (DDPG) [62, 63] es una combinación de los algoritmos Actor-Crítico con DQN, permitiéndole trabajar con entornos donde las políticas son deterministas, utilizando una política estocástica para explorar. Para mejorar la eficiencia en el cálculo del gradiente, DDPG deriva la función Q solamente respecto a las acciones. Además, utiliza técnicas como la repetición de experiencias (*experience replay*) y una red objetivo (*target*). La red objetivo tiene una lenta actualización $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ con $\tau \acute{}1$ [45], contribuyendo a la estabilidad del algoritmo. DDPG ha demostrado un rendimiento significativamente superior en comparación con DQN en juegos de Atari [64, 65]. Sin embargo, DDPG es altamente sensible a la selección de hiperparámetros y menos estable en comparación con algoritmos que actualizan la política en lotes [45].

El algoritmo *Recurrent Deterministic Policy Gradient* (RDPG) [66] es una extensión de DDPG que utiliza redes neuronales recurrentes (RNN). Esta variante resulta particularmente eficiente en entornos parcialmente observables, donde el agente tiene

que recordar eventos pasados para tomar decisiones informadas. Al igual que DDPG, RDPG utiliza la repetición de experiencias junto la red objetivo, e incorpora la capacidad de mantener la memoria a largo plazo por medio de las RNNs.

El algoritmo *Distributed Distributional Deterministic Policy Gradients* (D4PG) [67] es otra extensión que aborda la distribución de los retornos en lugar del retorno esperado. D4PG utiliza múltiples actores para muestrear trayectorias y realiza actualizaciones a la distribución del crítico [45].

Aprendizaje por refuerzo con múltiples agentes trabajadores: A2C, A3C, ACER

Al igual que los algoritmos Actor-Crítico, el algoritmo *Advantage Actor-Critic* (A2C) utiliza dos redes neuronales, una para estimar la política (Actor) $\pi(a|s;\theta)$ y la otra para evaluar la calidad de la salida del actor (Crítico). En este caso se emplean Diferencias Temporales para realizar la evaluación de la función valor $\widehat{\nu}(s;\phi)$, por lo tanto utiliza *bootstrapping* que introduce sesgo pero disminuye la varianza. Adicionalmente utiliza la función ventaja A para guiar la optimización de la política (Ec. 2.55), reforzando las acciones que obtienen mejores resultados y desincentivando las que obtienen peores, con la idea de que si $A > 0$ indica que tomar la acción a es mejor que seguir la política, mientras que si $A < 0$ tomar la acción a es peor que seguir la política.

El algoritmo D4PG propone a múltiples actores para recolectar datos, el algoritmo *Asynchronous Advantage Actor Critic* (A3C) [68] paraleliza el cálculo del gradiente para evitar el uso del búfer de repetición. En lugar de realizar la propagación con lote de episodios, se hacen propagaciones hacia atrás más pequeñas en paralelo con lotes más pequeños [45]. A3C está compuesto por un agente global y por varios trabajadores que contienes redes locales, es decir, sus propios parámetros.

El agente global es responsable de actualizar y sincronizar los pesos de la red neuronal compartida (entre los actores y críticos). Los trabajadores son responsables de interactuar con su propia instancia del entorno, realizando una copia local de la red global, la ejecución de episodios y el cálculo de sus respectivos gradientes. Posteriormente, la red global se actualiza con los gradientes locales del trabajador y después se vuelven a realizar los pasos de copiar la red global, ejecutar los episodios y calcular los gradientes. La actualización de la red es asíncrona porque cada trabajador lo hace en paralelo. El agente global y los trabajadores tienen cada uno dos grafos (o un grafo compartido, dependiendo de la arquitectura), con los parámetros correspondientes para estimar la política y la función valor [45], tal como se muestra en la Fig. 2.12.

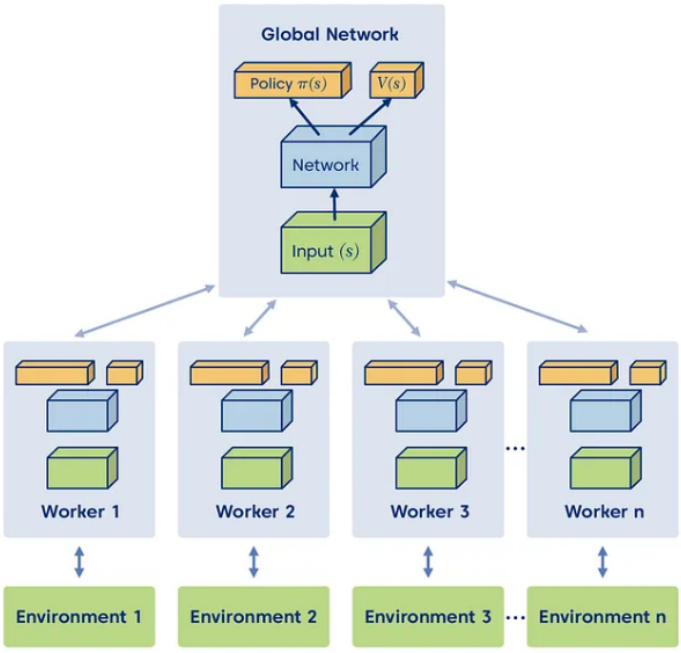


Figura 2.12: Arquitectura de alto nivel del algoritmo A3C. Tomado de [69].

El problema del asincronismo de A3C es que los trabajadores están potencialmente trabajando con versiones antiguas de las políticas. A2C es la versión síncrona de A3C, el cual espera hasta que todos los trabajadores hayan terminado y después actualiza el agente global y copia los pesos actualizados a los trabajadores. A2C funciona equivalente o incluso mejor que A3C [45].

Existen otros algoritmos como Actor-Critic with Experience Replay (ACER) [70] que es una extensión de A3C, con un búfer de repetición para la mejora de la eficiencia de la muestra. En el algoritmo Impala [71] los actores no realizan el cálculo del gradiente, sino que envían la transición completa al agente global para realizar la actualización.

Algoritmos de regiones de confianza (*Trust Region Algorithms*): TRPO, ACKTR, PPO

Los métodos de gradiente de política tienen algunos problemas que abordar. Aunque se tenga una señal de recompensa y una dirección de la actualización de la política por medio del gradiente, escoger la tasa de aprendizaje α es todo un reto, si es muy pequeña el aprendizaje será muy lento, pero si es muy grande será superado por el ruido, ya que el siguiente lote se recogerá utilizando una mala política [45].

Debido al riesgo de que se produzcan grandes cambios en la política, las muestras solamente se usan una vez. Posteriormente se utiliza la política actualizada para volver a muestrear, tal que las muestras antiguas dejan de ser representativas, lo que resulta ineficiente. Adicionalmente, otro problema es que las trayectorias de recom-

pensa pueden variar mucho, provocando oscilaciones en las actualizaciones [72].

El enfoque para resolver estos problemas es asegurarse de que al actualizar la política nunca se aleje demasiado de la política anterior. Esta idea se introdujo en el algoritmo *Trust Region Policy Optimization* (TRPO) [73], el cual añade la restricción Kullback-Leibler (KL) para asegurar que la política no se aleje demasiado de la "región de confianza"[74].

El algoritmo TRPO propone mejorar iterativamente la política. En cada actualización, la política se modifica para mejorar el rendimiento mientras se acerca a la política anterior. La divergencia de KL mide qué tanto una política π_θ cambia dada una actualización. El objetivo real a maximizar es $\eta(\theta) = \mathbb{E}\left[\gamma^t r(s_t)\right]$ pero es muy difícil de calcular, por lo que los autores introducen una función objetivo simplificada (surrogada) [45, 72]:

$$L(\theta) = J(\pi_{\theta+\Delta\theta}) - J(\pi_\theta) \approx \mathbb{E}\left[\frac{\pi_{\theta+\Delta\theta}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s,a)\right], \tag{2.56}$$

donde $\pi_{\theta+\Delta\theta}$ representa la nueva política y π_θ representa la política anterior. Esta función surrogada es precisa cuando $\pi_{\theta+\Delta\theta}$ y π_θ están próximos, por lo que ambas distribuciones se restringen con la divergencia KL. Por lo tanto, el objetivo del TRPO es [45]:

$$\max_\theta \sum_n^N \frac{\pi_{\theta+\Delta\theta}(a_n|s_n)}{\pi_\theta(a_n|s_n)} A_n, \tag{2.57}$$

$$\text{sujeto a: } \overline{KL}\left[\pi_\theta, \pi_{\theta+\Delta\theta}\right]. \tag{2.58}$$

El algoritmo ACKTR [75] es una variación de TRPO que utiliza la curvatura aproximada con factor de Kronecker en lugar de la divergencia de KL para la optimización de la región de confianza.

La restricción KL agrega una complejidad adicional al proceso de optimización y en ocasiones pueden conducir a un comportamiento indeseado en el entrenamiento. El algoritmo *Proximal Policy Optimization* (PPO) [76] es una versión simplificada de TRPO, y encuentra la forma de mantener las nuevas políticas cercanas a las antiguas con una implementación más sencilla y que empíricamente funcionan al menos igual que bien que TRPO. Existen dos variantes de PPO, PPO-Penalty y PPO-Clip, siendo esta última la que se usa principalmente por su rendimiento [45, 74]. La primera se define a partir de la Ec. 2.58, pero en lugar de tener la restricción separada, la penalización por la divergencia KL se añade directamente a la función surrogada, tal que:

$$L^{KLPEN}(\theta) = \mathbb{E}\left[\frac{\pi_{\theta+\Delta\theta}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s,a) - \beta KL\left[\pi_{\theta+\Delta\theta}, \pi_\theta\right]\right]. \tag{2.59}$$

Después de cada actualización, PPO comprueba el tamaño de la actualización. Si la divergencia $d = \mathbb{E}[KL\left[\pi_{\theta+\Delta\theta}, \pi_\theta\right]]$ supera una divergencia objetivo d_{targ} en más de 1.5, en la siguiente iteración se penaliza más la divergencia duplicando el valor de β. En caso de que las actualizaciones sean demasiado pequeñas, se reduce β a la mitad, ampliando la región de confianza.

$$\begin{aligned} &\text{si} \quad d < d_{targ}/1.5 && \beta \leftarrow \beta/2, \\ &\text{si} \quad d > d_{targ} \times 1.5 && \beta \leftarrow \beta \times 2. \end{aligned} \tag{2.60}$$

Sin embargo, esta función surrogada puede sustituirse por una más estable. PPO-Clip no tiene una restricción KL en el objetivo, sino que se basa en un recorte o *clipping* especializado en la función objetivo para eliminar los incentivos que hacen que la nueva política se aleje de la antigua, tal que:

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min(r_t(\theta)A_t, \text{clip}(s_t(\theta), 1-\epsilon, 1+\epsilon)A_t)\right], \tag{2.61}$$

donde r_t es el cociente de probabilidad bajo las políticas nueva y antigua $r_t(\theta) = \pi_{\theta+\Delta\theta}(a_t|s_t)/\pi_\theta(a_t|s_t)$, A_t es la función de ventaja en el tiempo t, y la función clip fuerza a $r_t(\theta) \in [1-\epsilon, 1+\epsilon]$.

El algoritmo pertenece a la familia de Actor-Crítico, por lo tanto, la función valor V también es calculada. En el caso donde tanto la política y la función valor compartan los mismos parámetros de la una red neuronal, la función objetivo es [45]:

$$L(\theta) = \mathbb{E}\left[L_t^{PG}(\theta) + c_1 L_t^{VF}(\theta) - c_2 S[\pi_\theta(s_t)]\right], \tag{2.62}$$

donde L^{PG} puede ser L^{KLPEN} o L^{CLIP}, L^{VF} es el error cuadrático del valor, S es una entropía para fomentar la exploración. Los algoritmos son comparados en:

- El *benchmark* de juegos de Atari, donde el rendimiento de PPO compite con el de ACER.
- Tareas de robótica utilizando el simulador Mujoco, donde el rendimiento del algoritmo PPO supera de manera importante a A2C y TRPO.

Adicionalmente, PPO presenta una ventaja de ser más sencillo que TRPO y ACER, con un rendimiento, al menos igual. Además, permite que la red de políticas y valores compartan parámetros [45].

Entropía con aprendizaje por refuerzo: SQL, SAC

Un problema en el aprendizaje por refuerzo es que los agentes se vuelvan altamente especialistas, dado que fueron entrenados para resolver una tarea muy específica de la mejor manera posible, en pocas palabras, los agentes aprenden sólo una forma de resolver la tarea. Una forma de resolverlo es añadiendo entropía, y maximizar tanto la suma de recompensas como la entropía, obligando así al agente a explorar y aprender

todas las formas posibles de resolver la tarea, tal como se presenta en *Reinforcement Learning with Deep Energy-Based Policies* [77]. La política óptima del algoritmo *Soft Q-Learning* (SQL) es [45]:

$$\pi_* = \arg\max_{\pi} \mathbb{E}\left[\sum_t r_t + \mathcal{H}\left(\pi(\cdot, s_t)\right)\right]. \tag{2.63}$$

SQL es el predecesor de *Soft Actor-Critic* (SAC) [78], siendo la versión de actor-crítico de SQL, combinando las actualizaciones fuera de política y máxima entropía. Un trabajo posterior [78] amplía SAC para el ajuste automático de hiperparámetros, conduciendo a un entrenamiento más estable. Ahora, muchos algoritmos (e.g., A3C, PPO) incluyen la entropía en su función surrogada para mejorar la exploración y robustez [45, 79].

Aprendizaje por refuerzo multiagente (*Multi Agent RL*)

El aprendizaje por refuerzo multiagente (MARL) son sistemas donde múltiples agentes utilizan técnicas de aprendizaje por refuerzo en un entorno, buscando comprender y optimizar cómo los agentes individuales pueden aprender y coordinar sus acciones para alcanzar objetivos comunes o competitivos. Es un campo importante porque en el mundo real se interactúa con múltiples agentes. Los juegos de Markov están definidos por la tupla [80]:

$$\langle n, S, A_1, ..., A_n, O_1, ..., O_n, R_1, ..., R_n, \pi_1, ..., \pi_n, T\rangle, \tag{2.64}$$

donde:

- n: es el número de agentes.
- $\mathcal{S}$: conjunto de los estados del entorno.
- $\mathcal{A} : A_1 \times A_2 \cdots \times A_n$ es el conjunto de acciones del i-ésimo agente.
- $\mathcal{O}$ es el conjunto de observaciones del i-ésimo agente.
- $\mathcal{R}$ es la función de recompensa del i-ésimo agente, $\mathcal{R}_i : \mathcal{S} \times \mathcal{A}_i \to \mathcal{R}$
- π_i es la política del i-ésimo agente, $\pi_i : \mathcal{O}_i \to \mathcal{A}_i$
- T es la función de transición de estados, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

Para abordar los problemas con multiagentes se puede pensar en dos enfoques extremos:

- El enfoque más sencillo sería entrenar todos los agentes de manera independiente sin considerar la existencia de los otros agentes, es decir, cada agente aprende su propia política considerando a los demás como parte del entorno. Con este enfoque surge un problema, porque los agentes al estar aprendiendo simultáneamente el entorno, desde la perspectiva del agente cambia de manera

dinámica, a esto se le conoce como entornos no estacionarios. En la mayoría de los algoritmos simples de RL, se asumen que el entorno es estacionario, lo que conduce a ciertas garantías de convergencia, pero en entornos no estacionarios estas garantías ya no se mantienen.

- El segundo enfoque es considerar la existencia de los múltiples agentes, tal que una sola política sea aprendida por todos los agentes tal que regrese un vector con las acciones de cada agente $\pi : \mathcal{S} \rightarrow \mathcal{A}_1 \times \mathcal{A}_2 \cdots \times \mathcal{A}_n$, generando una recompensa global para todos los agentes $\mathcal{R} : \mathcal{S} \times \mathcal{A}$. El problema con este enfoque es que el tamaño del espacio de acciones aumentará exponencialmente con el número de agentes. Este enfoque sólo funciona bien cuando cada agente conoce todo sobre el entorno, porque hay entornos que son parcialmente observables o que los agentes tienen una visión local, entonces cada agente tendrá una observación diferente del estado del entorno.

Existen diferentes tipos de entornos que existen en los sistemas multiagentes como cooperación, competencia y entornos mezclados.

2.5. Conclusiones

En este capítulo se abordaron algunos de los algoritmos de aprendizaje por refuerzo de manera no exhaustiva, centrándose en los algoritmos libres de modelo. Pero existen más algoritmos [81] como DAgger (*Dataset Aggregation*) [61] que utiliza etiquetas humanas para mejorar el aprendizaje por imitación, Dyna-Q [43] que es un bucle de constante aprendizaje de la función de valor o la política a partir de la experiencia real y simulada, utilizando la experiencia real para construir y perfeccionar el modelo dinámico, y después se utiliza ese modelo para producir experiencia simulada que complemente el entrenamiento de la función de valor y/o política.

Existen dos tareas fundamentales en el aprendizaje por refuerzo, la predicción y el control. En tareas de predicción, el objetivo es evaluar una política dada estimando el valor V o Q de realizar acciones siguiendo dicha política. En las tareas de control, la política es desconocida, por lo tanto, el objetivo es encontrar la política óptima que permita obtener la mayor cantidad de recompensas.

Los algoritmos de aprendizaje por refuerzo pueden clasificarse de diversas maneras según diferentes criterios (e.g., algoritmos basados en modelo y libres de modelo, *on-policy* y *off-policy*). Por lo tanto, es necesario conocer los requerimientos del problema para resolver para seleccionar o priorizar los algoritmos a probar. En la Fig. 2.13 se muestra un diagrama para seleccionar los algoritmos de aprendizaje por refuerzo según sus características.

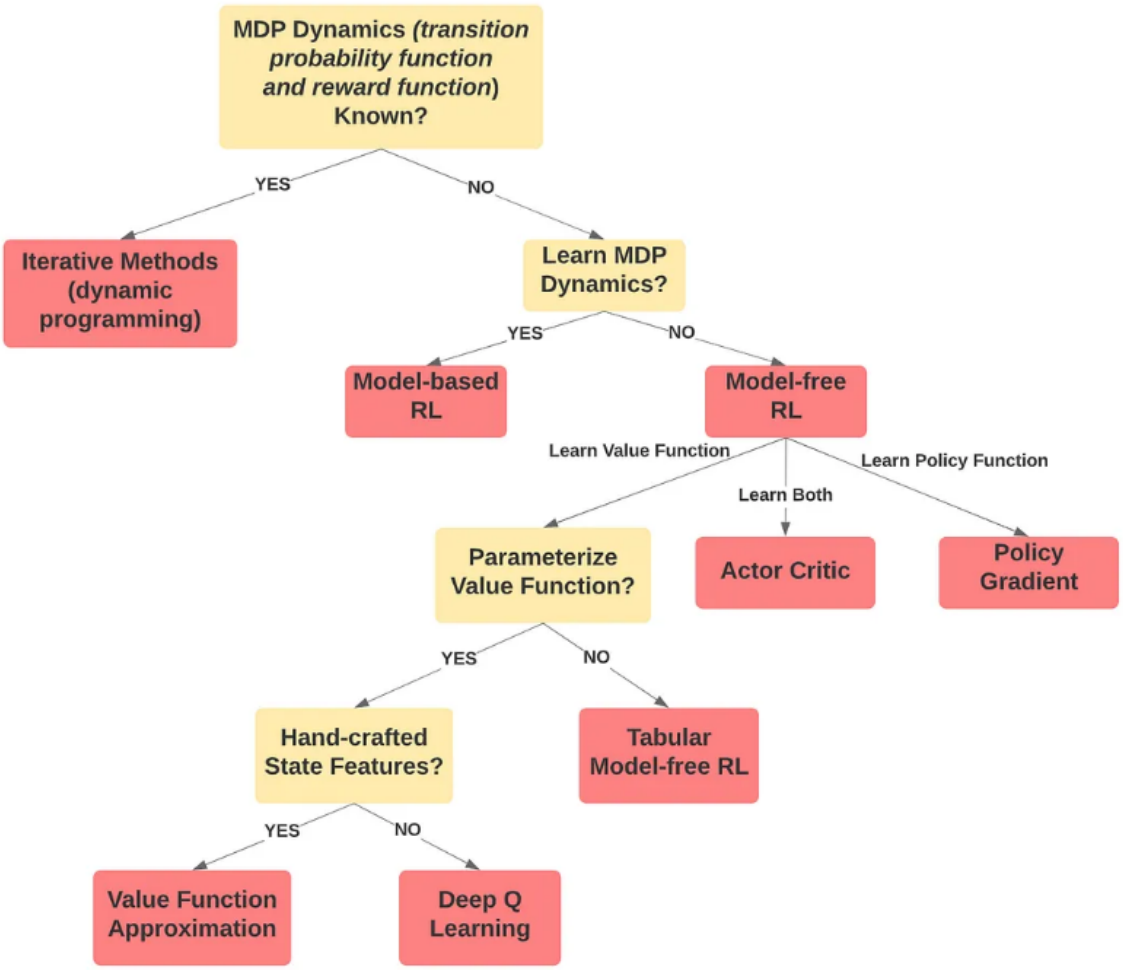


Figura 2.13: Diagrama alto nivel para la selección de algoritmos. Tomado de [82].

El problema que se busca solucionar en el presente trabajo es un algoritmo libre de modelo porque no se cuenta con un modelo explícito del entorno, por lo que los UAV's tendrán que aprender directamente de la experiencia. En trabajos similares donde ocupan un UAV en entornos simulados y libres de modelo, utilizan una diversidad de algoritmos para resolver las tareas, principalmente DQN, PPO, SAC, A2C, A3C, y adicionalmente utilizan un espacio de acciones discreto [1, 2, 3, 4, 5].

3 Desarrollo del marco de trabajo

Los simuladores recrean digitalmente el comportamiento y las características de vuelo de los UAV's en un entorno virtual, lo que permite realizar pruebas, experimentos y entrenamientos en entornos seguros y controlados, además de que permiten generar escenarios ficticios a través de datos sintéticos. Al combinar los algoritmos de aprendizaje por refuerzo con los simuladores, permite a los UAV's aprender y adaptarse a un entorno de manera autónoma. Actualmente, existen varios simuladores como: *Hector* ROS [83], *RotorS* [84], *PyBullet* [85], *AirSim* [86], *Flightmare* [87], entre otros.

En el presente trabajo se realizó la revisión de literatura y ejecución de pruebas de los simuladores *Hector* ROS, *Flightmare*, *PyBullet* y *AirSim*. Al final, se optó por utilizar *AirSim* por ser un simulador de código abierto, multiplataforma (UAV's y carros) y cuenta con documentación, comunidad y mejora continua. *AirSim* está construido sobre *Unreal Engine*, en donde ya tiene implementado la dinámica y control de un dron. El objetivo de *AirSim* es desarrollar una plataforma de investigación de inteligencia artificial para experimentar con algoritmos de visión computacional y aprendizaje por refuerzo [88], para ello *AirSim* tiene APIs para incorporar y manipular vehículos de manera independiente.

El *framework* propuesto en la Fig. 3.1 se basa en la arquitectura de *AirSim* [89] extendiéndola a un problema de aprendizaje por refuerzo, para ello el simulador está envuelto en un entorno de entrenamiento de *OpenAI Gym*[1], donde el agente que estará aprendiendo interactúa con el ambiente, que en este caso es el simulador. Un agente realiza una acción A_t y se manda al entorno de simulación, el cual generará un estado deseado de la formación de los UAV's en el siguiente paso de tiempo, para lograr dicho estado se emplea *AirSim* mediante su API.

[1] *OpenAI Gym* es un estándar para los entornos de entrenamiento en problemas de aprendizaje por refuerzo.

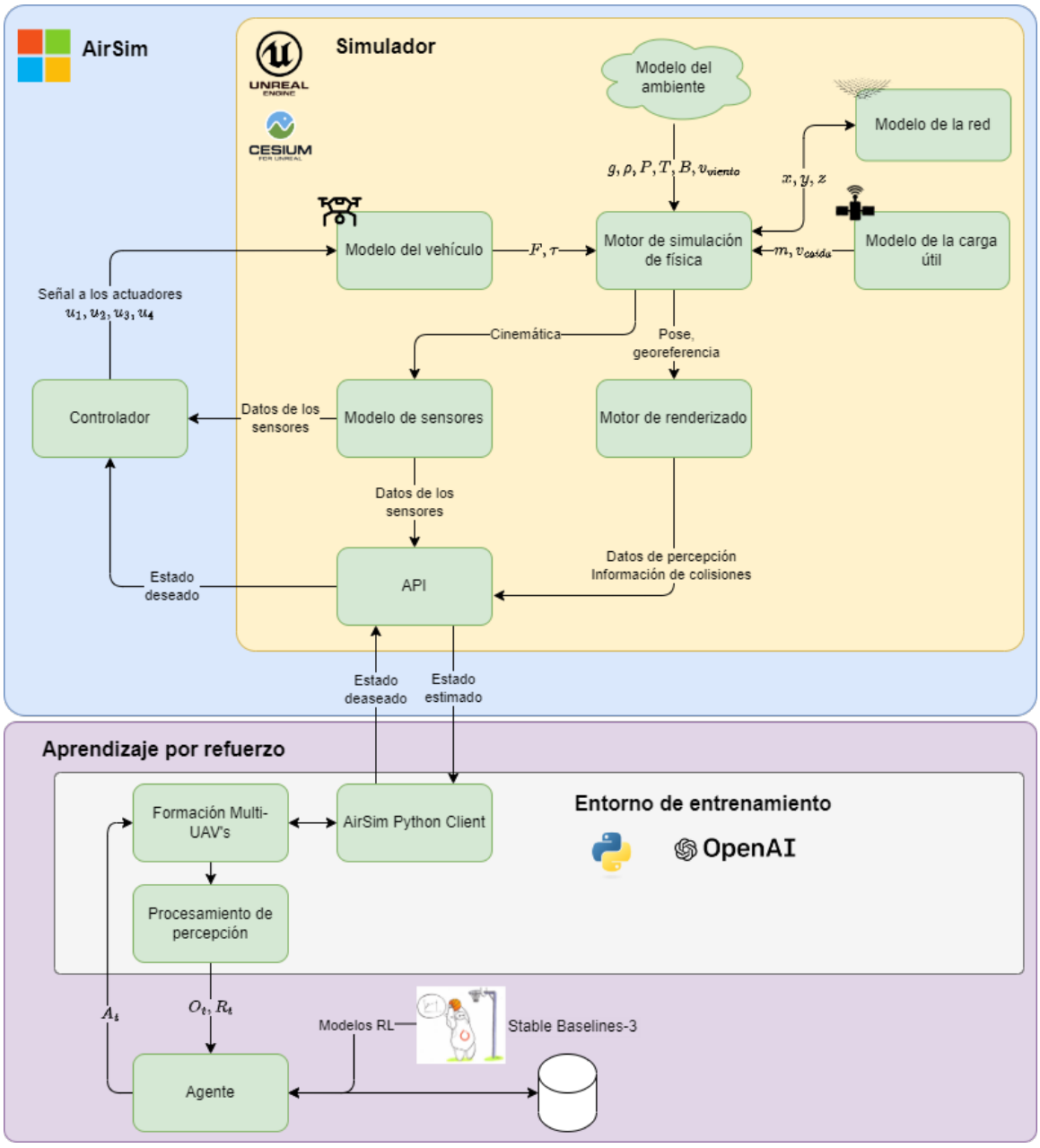


Figura 3.1: *Framework* propuesto [89, 90].

Durante la simulación, el simulador genera los datos de los sensores (barómetro, giroscopio, acelerómetro, magnetómetro, GPS) en función del mundo simulado. Un controlador PID en cascada[2] toma como entrada la información de los sensores junto con el estado deseado obtenido de la capa de aprendizaje por refuerzo mediante la API para calcular el momento de cada rotor u_1, u_2, u_3, u_4 del dron. El modelo del vehículo toma como entrada el momento de cada rotor para calcular el empuje F y torque τ generado por cada rotor según la dinámica del vehículo.

[2]El controlador PID en cascada es el utilizado por *AirSim* por defecto, pero se pueden emplear distintos controladores como PX4 [91], ROSFlight [92]

De igual manera existe un modelo dinámico para la red que calcula su posición en cada paso de tiempo al tomar el promedio de la posición de los UAV's. Además, se tiene un modelo dinámico de la carga útil que describe su trayectoria durante el descenso mientras emite información de su geolocalización. La información de los modelos dinámicos y la información del ambiente[3] como la gravedad, densidad del aire, presión atmosférica, campo magnético, GPS, velocidad del viento se introducen al motor de simulación física para calcular el siguiente estado cinemático de los multiagentes en el mundo simulado.

Después de calcular las posiciones de todos los agentes se renderiza el siguiente estado, obteniendo las colisiones de cada objeto mediante *Unreal Engine*, posteriormente se devuelve a la API la información del estado estimado, los datos de los sensores y colisiones. Dicha información se procesa en el entorno de entrenamiento que le envía al agente una observación O_t del ambiente y una recompensa R_t.

Para poner a prueba el *framework* de punta a punta, se realizó un tutorial en el que se entrena un UAV con el algoritmo DQN para seguir líneas de alta tensión [93]. El entrenamiento se ejecutó con éxito, con un tiempo de entrenamiento de 3.2 días para 10,000 épocas. El tiempo de entrenamiento se disminuyó al realizar un par de modificaciones: la primera fue desactivar el entorno gráfico y la segunda fue aumentar el *clock speed* que es un parámetro que acelera el tiempo del ambiente en relación con el reloj del mundo real, con ambas modificaciones se logró disminuir el tiempo de entrenamiento a 6.6 horas por cada 10,000 épocas.

En las siguientes secciones se describirá con mayor profundidad la dinámica del descenso de la carga útil (Modelo de la carga útil) y la formación de los múltiples UAV's (Formación Multi-UAV's).

3.1. Modelo de la carga útil

En la Fig. 3.2 se muestra la metodología general empleada para incorporar la dinámica del descenso de la carga útil de una manera más realista en el entorno de simulación. El primer paso es seleccionar distintas zonas de lanzamiento candidatas al explorar en *meteoblue* [94] zonas con bajas velocidades del viento, una vez definidas se analizó la información histórica meteorológica (dirección y velocidad del viento, temperatura, humedad, precipitación, presión y radiación solar) de los últimos 90 días de las Estaciones Meteorológicas Automáticas (EMAS) [95] más cercanas de las zonas candidatas, seleccionando así las EMAS candidatas.

Después de seleccionar las EMAS candidatas se realizó un *script* de Python que se conecta al software *Cambridge University Spaceflight* (CUSF) *Landing Prediction*

[3]Proporcionan la verdad para los modelos de sensores simulados.

[96] para generar predicciones de las trayectorias de la carga útil en una ventana de 10 días para cada hora y para cada EMA, obteniendo un total de 1,260 archivos CSV, ejecutando un proceso de extracción, transformación y carga (ETL) de los datos para generar métricas de interés como la altitud relativa, consolidando todos los archivos en un solo CSV.

Con el archivo CSV consolidado se prosigue a calcular métricas, como el tiempo de llegada de los UAV's, altura crítica en la cual se deben de lanzar los UAV's, entre otras. La altura crítica es importante porque ayuda a seleccionar un conjunto de datos en el tiempo de interés, de esta forma no es necesario simular toda la caída de la carga útil, solo un intervalo de tiempo, disminuyendo el tiempo de entrenamiento. Una vez identificada la altura crítica se prosigue a utilizar un método de interpolación de puntos con polinomios cúbicos naturales o *spline natural*, para convertir la escala temporal de minutos a cuadros por segundo o *frames* por segundo (FPS), dado que CUSF genera la información de la trayectoria en una escala de minutos.

El siguiente paso es filtrar la información de cada trayectoria para únicamente tener los datos del descenso dentro de la altura crítica definida, después se ejecutan los *splines naturales* para interpolar datos en las variables de interés que son la latitud, longitud y altitud. Después se calculan los puntos de lanzamiento de los UAV's en la simulación, para finalmente guardar los archivos CSV con la trayectoria de la carga útil (tiempo, latitud, longitud y altitud) por separado en un banco de trayectorias.

El entorno de simulación seleccionará de manera aleatoria algún archivo CSV dentro del banco de trayectorias, inicializando así la ubicación de lanzamiento de los UAV's, el punto inicial del descenso y la posición en cada paso de tiempo (en cada *frame*) de la carga útil en el entorno de simulación.

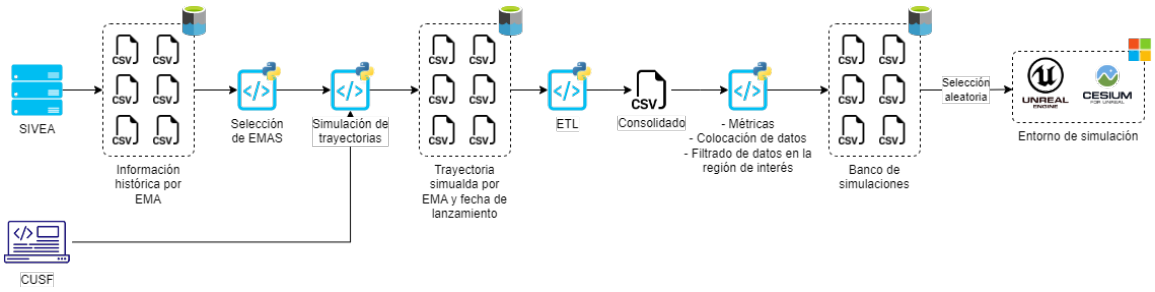


Figura 3.2: Metodología para la simular el descenso de la carga útil en el entorno de simulación.

El objetivo final es generar un banco de trayectorias que contenga distintos archivos CSV con la posición de la carga útil *frame* por *frame* durante el descenso en el rango definido por la altura crítica. El flujo consiste en que durante el entrenamiento, el entorno de simulación lea aleatoriamente un archivo CSV, procesándolo en *Unreal* para: inicializar el entorno de simulación en una ubicación específica que está definida

por la latitud, longitud y altitud para cada EMA, y para cada paso del tiempo en el entorno virtual (i.e., para cada *frame*) actualizar la posición de la carga útil en la ubicación especificada (latitud, longitud y altitud) simulando así la dinámica de la carga útil durante el descenso dentro de un rango definido, que en este caso es la altura crítica. Tal como se muestra en al Fig. 3.2.

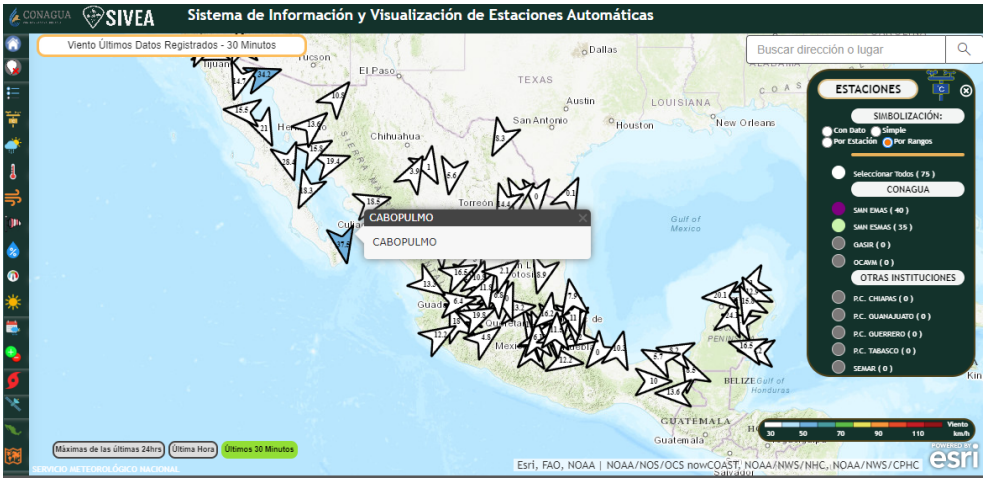
A continuación, se describirán con mayor detalle cada uno de los pasos mencionados.

3.1.1. Selección de puntos de lanzamiento

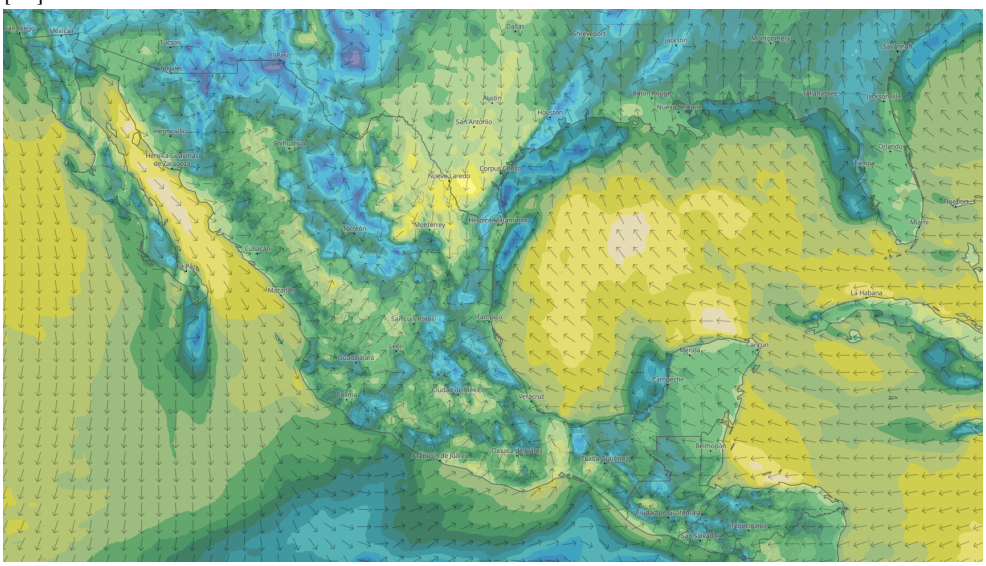
El primer paso es seleccionar algunos puntos candidatos de lanzamiento, identificando lugares con una baja velocidad del viento para que la carga útil no se aleje demasiado del punto de lanzamiento, permitiendo el traslado del equipo de rescate al punto aproximado de aterrizaje, aumentando así las probabilidades de capturar la carga útil. Para seleccionar los puntos candidatos se emplearon un par de fuentes de información:

- El Sistema de Información y Visualización de Estaciones Automáticas (SIVEA) que contiene información histórica y actual meteorológica como precipitación, temperatura, velocidad y dirección del viento, entre otras para las distintas Estaciones Meteorológicas Automáticas (EMAS) [95].
- *meteoblue* es un servicio[4] que produce datos meteorológicos (temperatura, dirección y velocidad del viento, precipitación, entre otras) de alta calidad en cualquier punto terrestre o marítimo del mundo, además de tener la información meteorológica histórica, actual y proyectada [94].

[4]Un servicio que tiene datos abiertos pero cuenta con una suscripción para acceder a más funcionalidades y datos.



(a) Información de la dirección y velocidad del viento por EMA. Tomado del SIVEA [95].



(b) Información de la dirección y velocidad del viento en el territorio mexicano. Tomado de *meteoblue* [94]

Figura 3.3: Información de la dirección y velocidad del viento en distinta granularidad por fuente de información.

La selección de los puntos candidatos consistió en un análisis visual sencillo, en el cual se aprovechó la granularidad de la información del servicio *meteoblue* para identificar un conjunto de zonas cercanas a la Ciudad de México con menor velocidad del viento (zonas en color azul en la Fig. 3.3b) y que fueron constantes en distintas horas y días (1 semana después del día actual). Una vez seleccionados el conjunto de zonas, se identificaron las EMAS correspondientes a las zonas, seleccionando así a las estaciones: ATLACOMULCO, CORDOBA, IGUALA, IMTA, LAGUNASDEZEMPOALA, PEROTESMN y VALLEDEBRAVO.

Una vez seleccionados los puntos candidatos, el siguiente paso es realizar un análi-

sis con el objetivo de entender los datos históricos del viento para cada estación, y de esta forma priorizar las zonas que son candidatas reales como un punto de lanzamiento. Para realizar el análisis se utilizó la información del SIVEA, que tiene de manera abierta e inmediata la información meteorológica de los últimos 90 días para cada estación, aunque se puede obtener mayor información histórica al realizar la petición vía correo electrónico a la Comisión Nacional del Agua (CONAGUA), mientras que en *meteoblue* el acceso a la información histórica es por medio de una suscripción de paga.

La información meteorológica que contiene el SIVEA para cada estación es: fecha con frecuencia de 10 minutos, dirección del viento y de ráfaga, velocidad del viento y de ráfaga, temperatura del aire, humedad relativa, presión atmosférica, precipitación y radiación solar. Para el análisis basta con la información de la velocidad del viento. En la Fig. 3.4 se observa que la EMA TOLUCA tienen 3 puntos atípicos, por lo que se decide eliminar dichos puntos, además el EMA NEVADODETOLUCA tiene un valor promedio y varianza mayor que las demás EMAS, por lo que se decide descartar, quedando la Fig. 3.5 de la cual se concluye:

- Las EMAS IGUALA e IMTA no tienen observaciones, por lo tanto, son descartadas como candidatas a puntos de lanzamiento.
- La EMA PEROTESMN contiene muy pocas observaciones, por lo tanto, es descartada como candidata a punto de lanzamiento.
- La EMA NEVADODETOLUCA tiene un valor promedio y varianza mayor respecto a las demás EMAS, por lo tanto, es descartada como candidata a punto de lanzamiento.
- Las EMAS restantes se mantienen como candidatas dado que se tienen todas las observaciones y además el valor promedio y varianza se encuentran dentro de un rango aceptable.

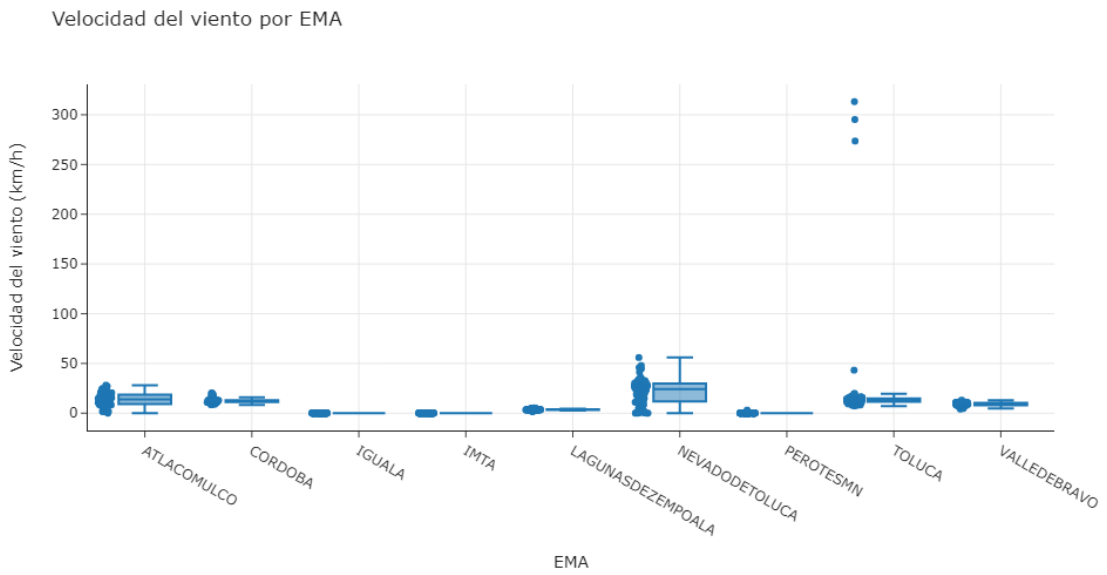


Figura 3.4: Diagrama de caja o *box plot* de la velocidad del viento en los últimos 90 días por EMA.

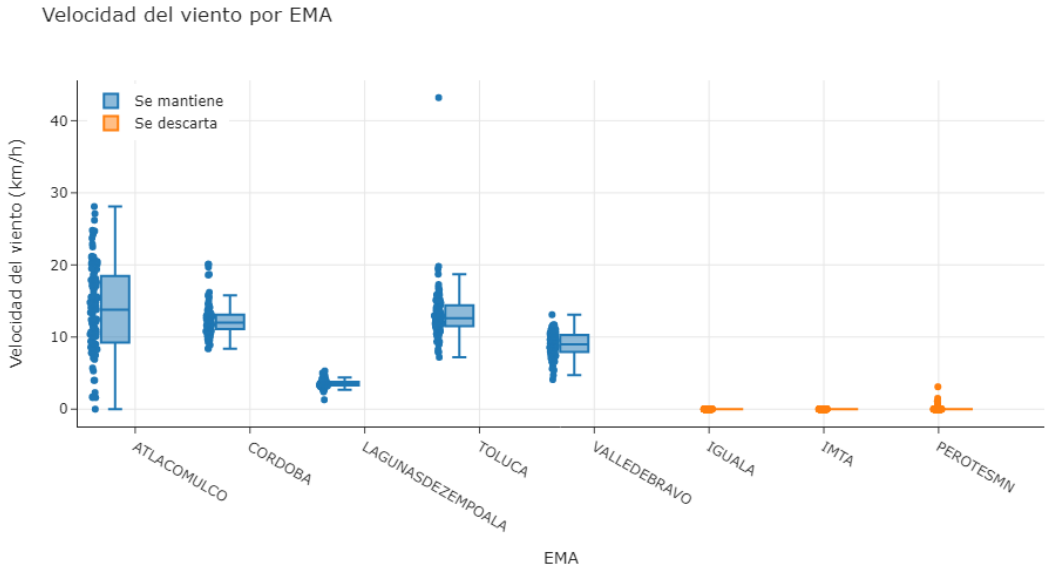


Figura 3.5: Velocidad del viento en los últimos 90 días por EMA, quitando los puntos atípicos de TOLUCA y descartando a NEVADODETOLUCA.

En la Fig. 3.6 se muestra la velocidad promedio del viento por hora para cada EMA, se observa que existe una variación de la velocidad del viento en cada hora.

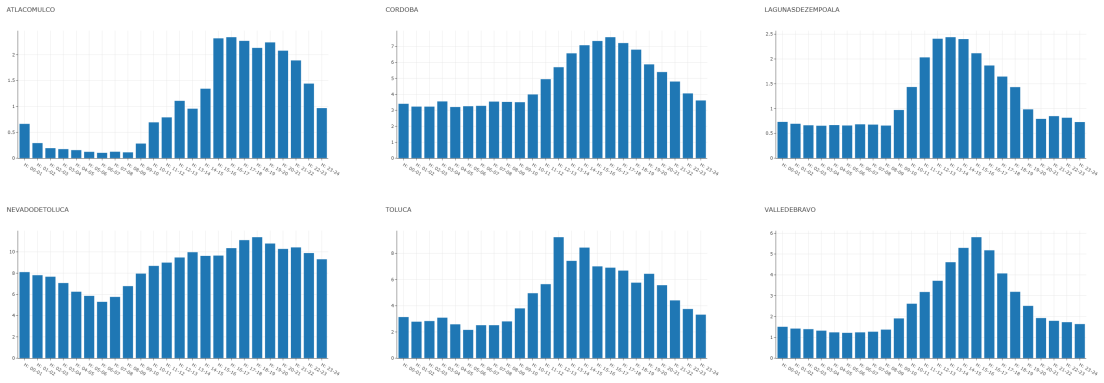


Figura 3.6: Velocidad promedio horaria del viento por EMA.

Nótese que el objetivo de esta sección es obtener un conjunto de puntos de lanzamiento factibles para generar distintas trayectorias del descenso de la carga útil, con la finalidad de tener mayor diversidad en los datos para el entrenamiento de los UAV's con los algoritmos de aprendizaje por refuerzo. El análisis para determinar los puntos candidatos puede ser más exhaustivo al solicitar la información histórica con una ventana de tiempo mayor (e.g., 10 años) de cada una de las estaciones, permitiendo seleccionar de mejor manera los puntos de lanzamiento, así como las mejores fechas y horarios.

3.1.2. Generación de trayectorias de la misión

La dinámica en el descenso de la carga útil es importante para tener una diversidad en el entrenamiento de los algoritmos de aprendizaje por refuerzo, es por lo que se evaluaron tres alternativas:

- Incorporar en el ambiente de simulación las condiciones ambientales como la velocidad del viento para que con el motor de simulación de física se perturbara la carga útil y así generar las posiciones de la carga útil durante el descenso, pero implica conocer un modelo que genere un perfil de velocidad del viento en función de la altitud, o las distribuciones de probabilidad en función de datos históricos para ingresarlo al modelo del ambiente, esa información histórica se puede obtener en *meteoblue* con una suscripción de paga, ya que el SIVEA no cuenta con la información de la velocidad del viento en función de la altitud.

- Generar una nueva posición en cada paso de tiempo de la simulación para la carga útil, lo que implica realizar un modelo que genere esas nuevas posiciones, el modelo puede ser un modelo estocástico que tome como referencia algunas trayectorias encontradas en la literatura.

- Utilizar un software que pronostique la trayectoria completa de la misión, es decir, desde el punto de lanzamiento hasta el punto de aterrizaje, tal que incorpore la ubicación, día y hora del lanzamiento y considere el clima, así como

los parámetros de la misión como la altura promedio en que explota el globo, velocidad de ascenso y descenso.

De las tres alternativas se decidió por la tercera ya que se pueden generar trayectorias del descenso más realistas incorporando información de la misión y del clima. El software *Cambridge University Spaceflight* (CUSF) *Landing Prediction* por la Universidad de Cambridge [96] está basado en el modelo *Global Forecast System* de la *National Oceanic and Atmospheric Administration* (NOAA). El software permite pronosticar la trayectoria completa de la misión, desde el despegue hasta el aterrizaje.

El software CUSF toma como insumo información acerca del lanzamiento: nombre o latitud y longitud de la ubicación, altitud, día y hora. Adicionalmente se le ingresa información acerca del HAB: velocidad de ascenso, altitud a la cual explota el globo (*burst*) y la velocidad de descenso; estos parámetros se obtienen directamente del proveedor Strato Flights, que para el presente trabajo se utiliza el kit de *Weather Balloon 800* ya que es un kit relativamente barato y para personas que apenas están iniciando [5].

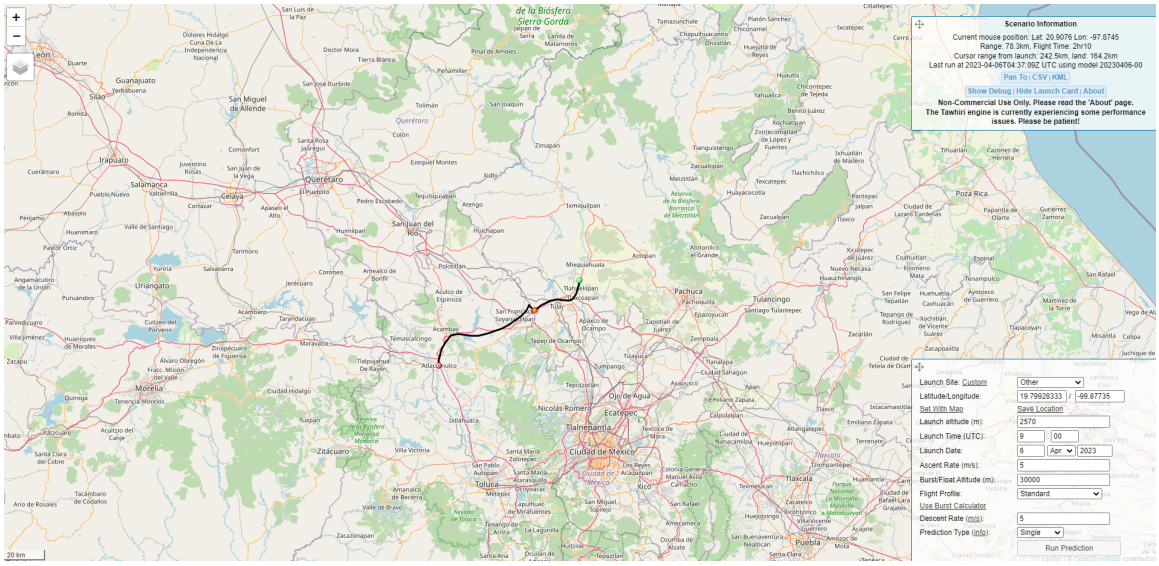


Figura 3.7: Ilustración de la generación de una trayectoria utilizando CUSF ingresando información del lanzamiento y del HAB [96].

CUSF genera un archivo CSV con la información minuto a minuto de la latitud, longitud y altitud de la carga útil durante todo el tiempo de la misión, como se observa en la Fig. 3.7. Una limitante de este software es que el clima no se puede pronosticar a largo plazo lo que limita simular trayectorias muy alejadas del día en el que se realizan las simulaciones, en el caso de CUSF únicamente se puede simular las trayectorias dentro de una ventana de 10 días.

[5] https://www.stratoflights.com/en/shop/weather-balloon-800/

Una vez identificado el software, se realizó un *script* de Python para simular múltiples trayectorias, guardando los archivos CSV de manera automática al considerar a cada EMA como un punto de lanzamiento y usando los parámetros de la Tabla 3.1, simulando así la trayectoria para cada hora durante una ventana de 10 días (1,200 trayectorias para cada punto de lanzamiento), es decir, se generaron un total de $n_{EMA} \times 24 \times 10 = 1,200$ archivos CSV que contienen la información de múltiples lanzamientos.

Tabla 3.1: Latitud, longitud y altitud de cada EMA

EMA	Estado	Latitud	Longitud	Altitud
ATLACOMULCO	México	19.80	-99.88	2,570
CORDOBA	Veracruz	18.89	-96.92	852
LAGUNASDEZEMPOALA	Morelos	19.05	-99.31	2,820
TOLUCA	México	19.29	-99.71	2,730
VALLEDEBRAVO	México	19.38	-100.08	2,477

El siguiente paso es extraer, limpiar, transformar los datos en cada uno de los archivos CSV generados calculando la altura relativa en cada punto tomando como referencia la altitud inicial, la distancia de Haversine[6] relativa al punto inicial de lanzamiento, y el estatus del vuelo: ascenso, explosión (*burst*) o descenso. Una vez procesados los datos se guardan en una base de datos que consolida toda la información generada en el paso anterior, dicha base de datos se analizará posteriormente para generar métricas con la finalidad de filtrar los puntos que se encuentren dentro del rango de interés para realizar el rescate de la carga útil, y así disminuir el número de pasos en cada simulación.

3.1.3. Generación de la trayectoria de descenso de la carga útil

En esta sección se desarrollan tres actividades: identificar métricas que permitan generar un entendimiento de la ventana de tiempo y distancia necesaria para realizar el rescate de la carga útil, obtener el número de observaciones necesarias en función de las métricas para entrenar los algoritmos, y por último generar un archivo CSV para cada una de las trayectorias simuladas que contenga los datos de latitud, longitud y altitud en cada paso de tiempo, con el objetivo de utilizar esos datos en el entorno de simulación para colocar en cada *frame* a la carga útil, simulando así el descenso.

El primer paso consiste en analizar las distintas trayectorias simuladas por EMA, observando una diversidad en las trayectorias tal como se muestra en la Fig. 3.8,

[6]La distancia de Haversine permite calcular la distancia entre dos puntos geográficos a partir de sus respectivas coordenadas (latitud y longitud).

lo que significa que el software efectivamente genera trayectorias distintas en función de la fecha y hora del lanzamiento, incluyendo el uso de datos meteorológicos, teniendo una diversidad en el conjunto de datos para el entrenamiento de los modelos.

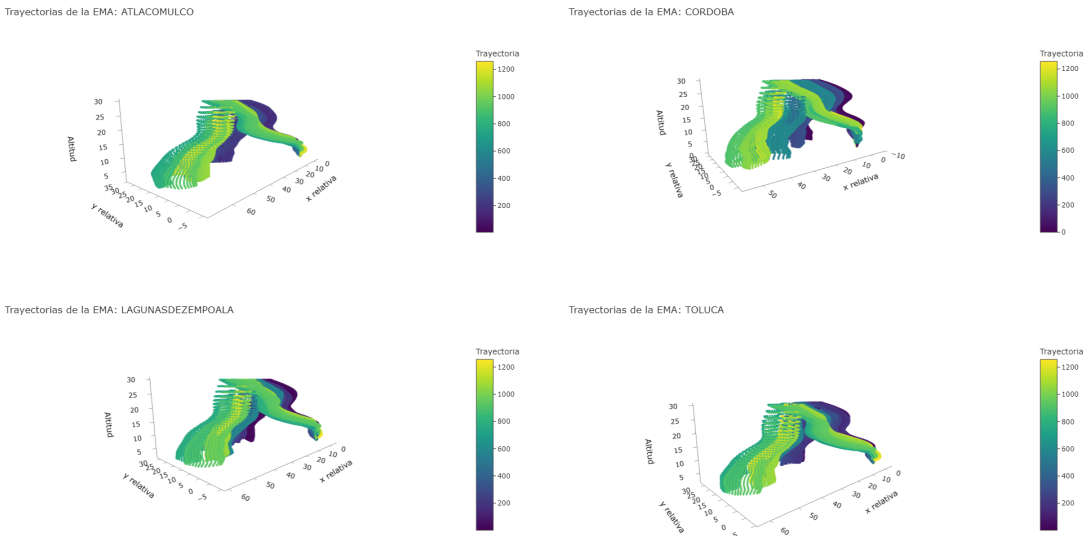


Figura 3.8: Trayectorias simuladas para cada EMA con distinto horarios. El eje x y y representan la distancia relativa de la carga útil respecto al punto de lanzamiento en los respectivos ejes, mientras que el color es para diferenciar las distintas trayectorias generadas.

En la Fig. 3.9 se muestra un *box plot* con la duración de la misión, es decir, desde el lanzamiento del HAB hasta el aterrizaje de la carga útil en cada una de las EMAS, con un promedio de 2.2 horas, siendo CORDOBA el de mayor promedio con 2.37 horas además de tener poca varianza, por lo tanto, es una buena opción para realizar el lanzamiento dado que se busca un mayor tiempo de vuelo para que el equipo de rescate pueda acercarse lo más posible, y así aumentar las probabilidades de rescate.

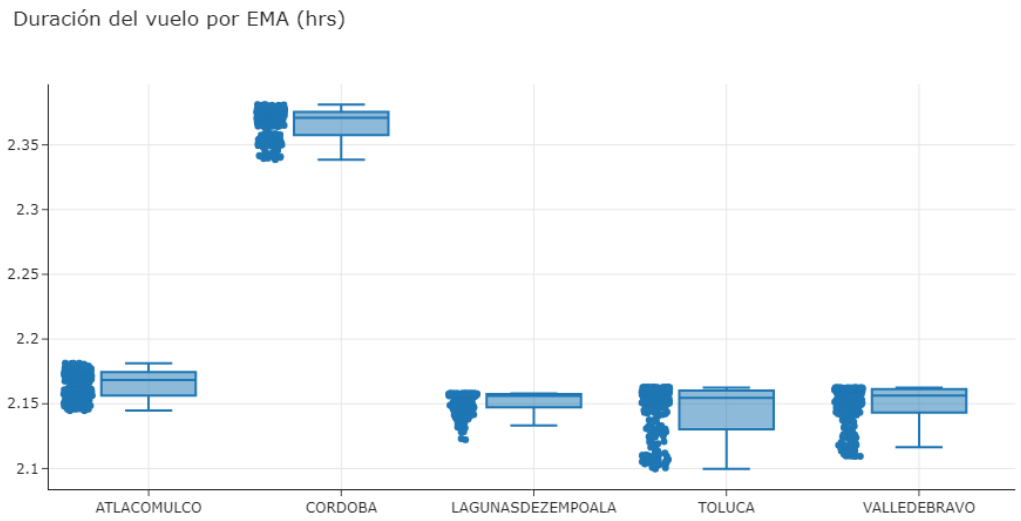


Figura 3.9: Duración en horas de toda la misión para cada EMA.

En la Fig. 3.10 se muestra un *box plot* con la distancia relativa[7] del punto de aterrizaje respecto al punto de lanzamiento. La distancia relativa promedio es de aproximadamente 48.5 km, siendo CORDOBA la EMA con el menor promedio (42 km), por lo tanto es una buena opción para realizar el lanzamiento dado que se busca que la carga útil aterrice lo más cercano posible del punto de lanzamiento.

[7]La distancia empleada a largo de la sección es la distancia de Haversine, salvo que se especifique lo contrario.

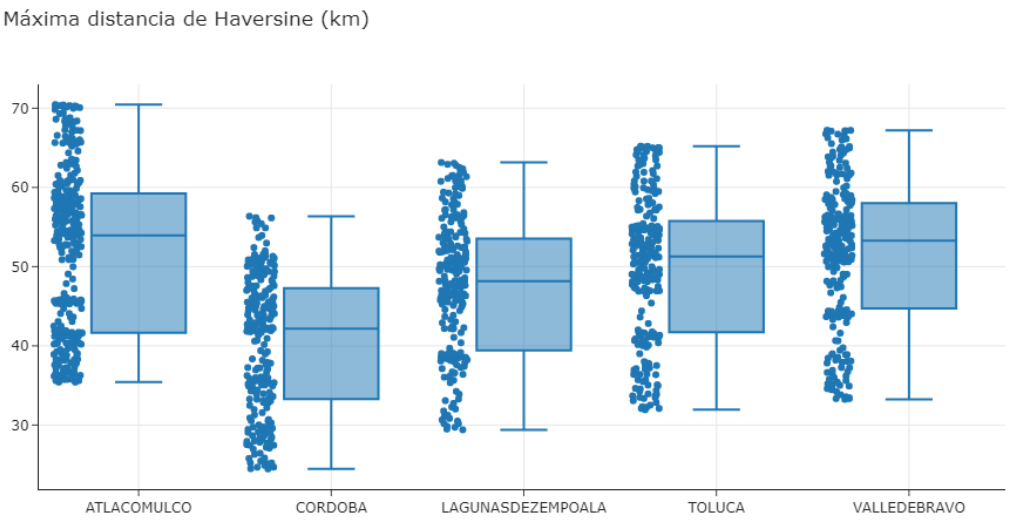


Figura 3.10: Máxima distancia relativa de Haversine en kilómetros desde el punto de lanzamiento hasta el punto de aterrizaje para cada EMA.

Posteriormente, se realizó el mismo análisis, pero únicamente con la información del descenso. En la Fig. 3.11 se muestra un *box plot* de la duración en minutos durante el descenso con un promedio de 39.3 minutos, siendo CORDOBA el de mayor promedio (45 minutos). En la Fig. 3.12 se muestra un *box plot* de la distancia relativa del punto de aterrizaje respecto al punto de máxima altitud (*burst*) con un promedio de 13.2 km, siendo CORDOBA el de menor promedio con 10.6 km, es decir, a partir de que el globo explota, la carga útil se aleja en promedio 10.6 km.

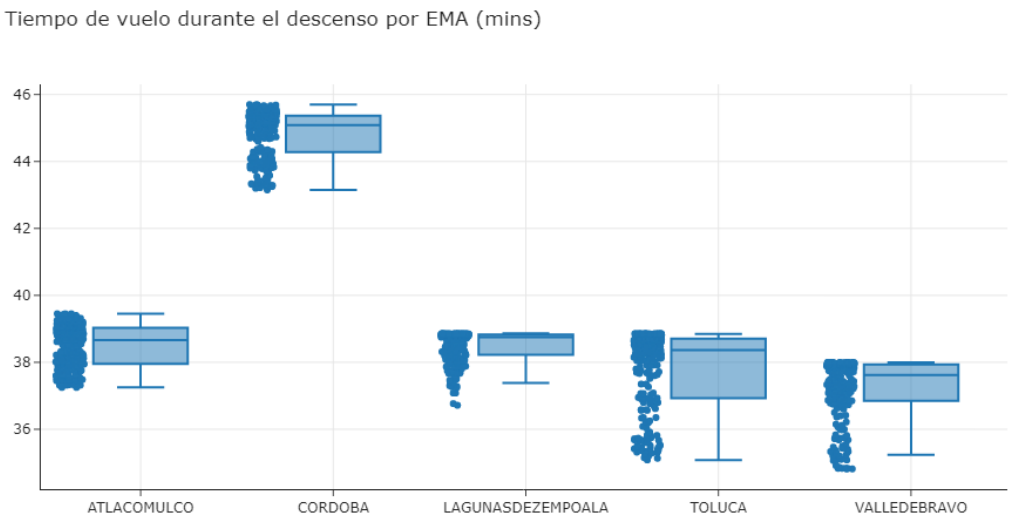


Figura 3.11: Duración en minutos únicamente del descenso para cada EMA.

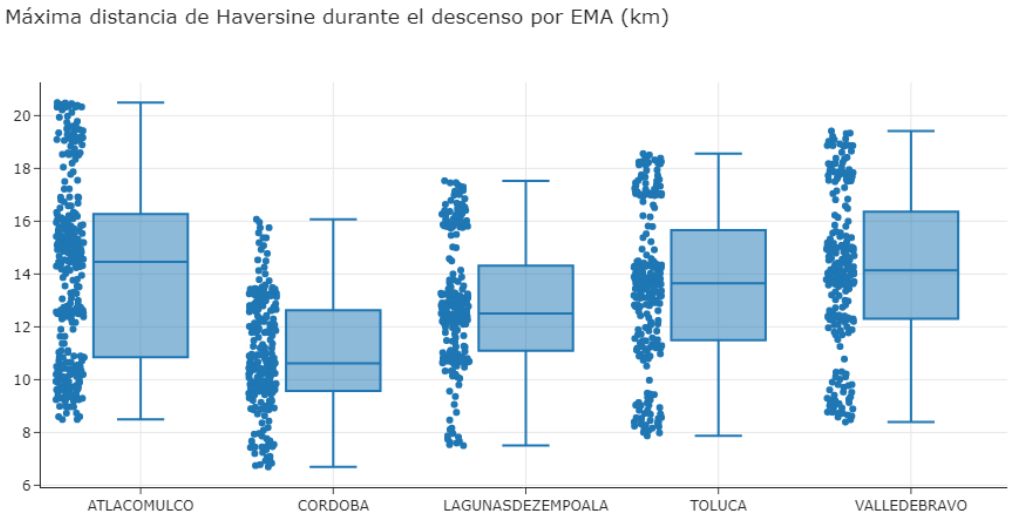


Figura 3.12: Máxima distancia relativa de Haversine en kilómetros del punto máxima altitud al punto de aterrizaje para cada EMA.

A continuación, se calculan una serie de métricas para acotar el número de pasos en cada simulación. Como primer paso es necesario conocer las regulaciones para los UAV's en México, la Secretaría de Relaciones Exteriores menciona que los UAV's no deben volar a más de 122 m de altura (h_{max}), ni más allá de 457 m de distancia horizontal del piloto (r_{max}). Siempre se debe mantener una distancia mayor de 9.2 km de cualquier aeropuerto y no pueden usarse a menos de 46 m de distancia de

las personas [97]. En la Fig. 3.13 se muestra un diagrama ilustrando cada una de las métricas utilizadas.

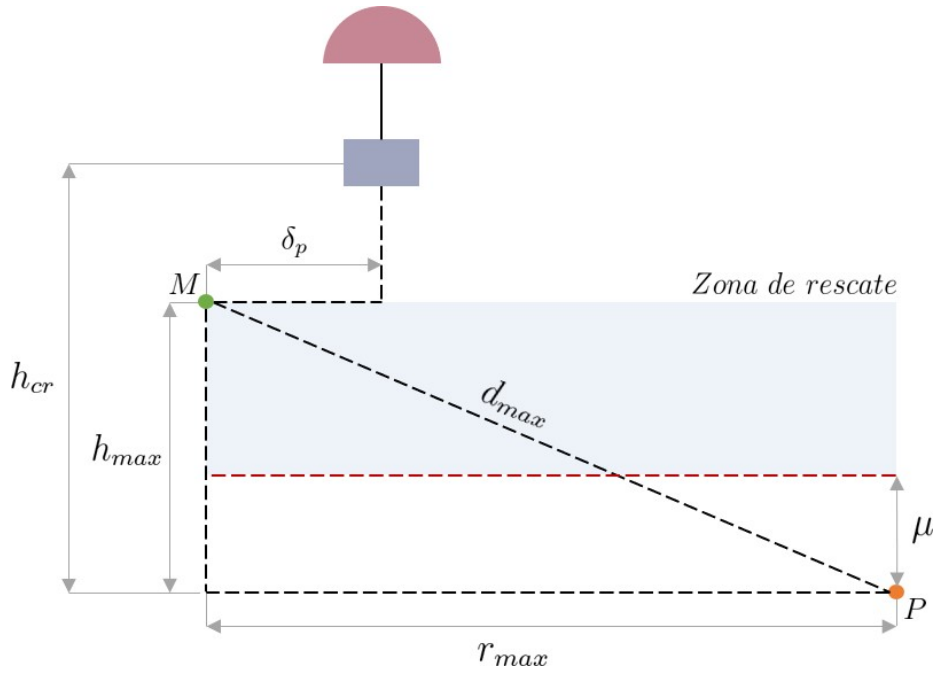


Figura 3.13: Diagrama para el cálculo de las métricas.

La primera métrica es el tiempo de llegada de un UAV t_l en línea recta del punto de lanzamiento de los UAV's P al punto máximo permitido M, tal que el tiempo de llegada se calcula como:

$$t_l = \frac{d_{max}}{\langle v_{dron} \rangle} = \frac{\sqrt{h_{max}^2 + r_{max}^2}}{\langle v_{dron} \rangle}, \tag{3.1}$$

donde $\langle v_{dron} \rangle$ es la velocidad promedio de un UAV, para el presente trabajo se utilizó un promedio de UAV's recreativos que es de 5 m/s ($\approx$ 20 km/h), sustituyendo los valores en la Ec. (3.1) se obtiene que el tiempo de llegada es de aproximadamente 1.58 minutos.

La segunda métrica es la altura crítica h_{cr} que indica a qué altura de la carga útil se debe lanzar a los UAV's para rescatar la carga útil, es decir, si los UAV's son lanzados cuando la altura de la carga útil $h \geq h_{cr}$ entonces se cuentan con las condiciones para atrapar la carga útil, pero si $h < h_{cr}$ entonces las condiciones para atrapar la carga útil se vuelven adversas. Para calcular h_{cr} se considera el peor escenario donde la carga útil llega al punto máximo permitido M, obteniendo la siguiente ecuación:

$$h_{cr} = h_{max} + \alpha \times t_l \times v_c, \tag{3.2}$$

donde el primer componente es la altura máxima permitida, y el segundo componente es la distancia que recorre la carga útil para llegar al punto M en el tiempo de

llegada de los UAV's (t_l) en función de la velocidad de caída de la carga útil v_c y un parámetro α que emula un factor de seguridad, tratando de aumentar a h_{cr} por no considerar diversos factores como que los UAV's no llegan en línea recta del punto P al punto M o que la carga útil no cae verticalmente, sino que tiene una pendiente de caída, y otros factores que no se estén considerando.

El cálculo de la velocidad de caída de la carga útil v_c se aproximó con la información de los puntos en cada minuto de las trayectorias generadas en esta subsección, obteniendo las velocidades promedio en función de la altitud según la EMA, tal como se muestra en la Fig. 3.14.

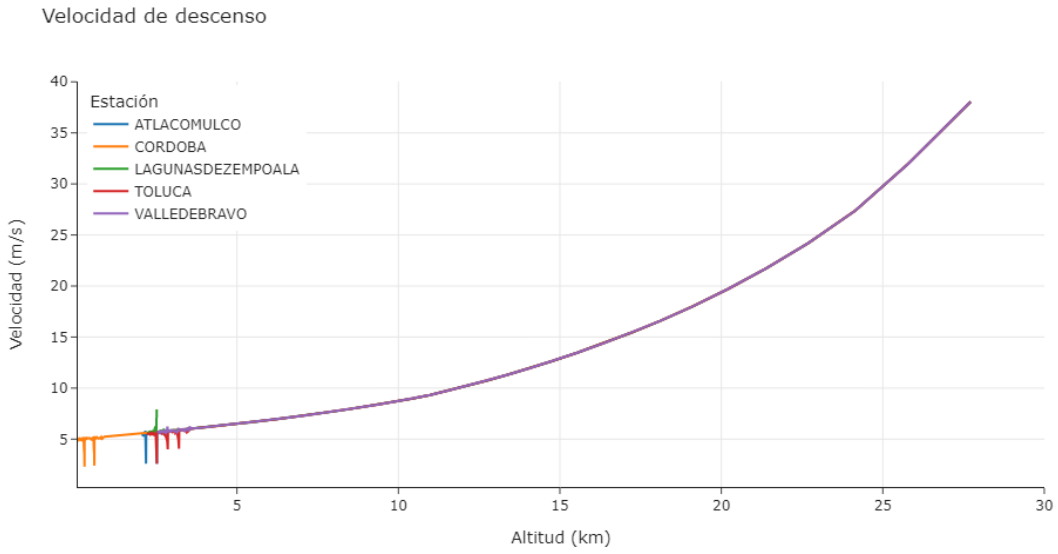


Figura 3.14: Perfil de velocidades durante el descenso en función de la altitud.

En la Fig. 3.14 se observa que la velocidad de la caída no es lineal respecto a la altitud, y esto se debe al efecto del paracaídas, además se observa velocidades similares para cada EMA y esto es porque se tiene el mismo producto (carga útil y paracaídas).

El rescate se efectúa en una cierta región de la altura por las limitaciones reglamentarias, por lo tanto, se decide utilizar los últimos 5 km de la misión para obtener la velocidad de caída v_c. En la Tabla 3.2 se muestran las velocidades máximas, promedio y mínimas (m/s) para cada EMA, se decidió utilizar la velocidad máxima como la velocidad de caída de la carga útil para tener una mayor altura crítica obtenido para el presente trabajo $v_c = 6.38$ m/s.

Tabla 3.2: Velocidades máximas, promedio y mínimas de descenso por EMA (m/s)

EMA	Máximo	Promedio	Mínimo
ATLACOMULCO	6.38	5.48	2.60
CORDOBA	6.38	5.01	2.30
TOLUCA	6.38	5.57	2.60
VALLEDEBRAVO	6.38	5.82	5.54

Por lo tanto, la altura crítica en la Ec. (3.2) considerando a $\alpha = 1.5$ y $v_c = 6.38$ es $h_{cr} = 1027.33$ m, por lo que los UAV's deben ser lanzados cuando la altura de la carga útil $h \geq 1027.33$ m. Además de que la métrica genera información relevante para rescatar la carga útil en una situación real, sirve para acotar el tiempo de entrenamiento, ya no es necesario simular toda la dinámica de la carga útil en el descenso, solamente es necesario simular el comportamiento dentro de la altura crítica considerando el peor escenario para capturar la carga útil cumpliendo con las restricciones de vuelo por ley.

La tercera métrica es el tiempo de rescate crítico t_r que se tiene para ejecutar con éxito la misión, es decir, la ventana de tiempo que se tiene para capturar la carga útil. El UAV puede ser lanzado con mayor antelación sin que rebase la altura máxima permitida. El tiempo de rescate crítico está dado por:

$$t_r = \frac{h_{max} - \mu}{v_c}, \tag{3.3}$$

donde μ representa un umbral para realizar la captura, es decir, si $\mu = 0$ implica que se tiene toda la altura impuesta por ley (h_{max}) para interceptar a la carga útil, que en este escenario se tiene un tiempo de rescate $t_r = 19.12$ s. Pero esto no es real, ya que existen objetos con altura como árboles que pueden interferir en la misión, por ejemplo, si $\mu = 60$ entonces se tienen 62 m para realizar la captura, siendo así un tiempo de rescate $t_r = 9.72$ s. El tiempo de autonomía promedio de los UAV's es de 20 a 30 minutos, dado que la misión (lanzamiento de los UAV's del punto P, captura de la carga útil, y regreso al punto P) ocurre en una ventana de tiempo mucho menor, se descarta esta variable para el presente trabajo.

Por último, se calcula la métrica δ_p que mide la desviación en términos de la latitud y longitud de la carga útil partiendo de la altura crítica (h_{cr}) respecto a la altura máxima permitida (h_{max}). Para calcularla se utilizó la información de las trayectorias simuladas, en la Fig. 3.15 se observa un *box plot* con la métrica δ_p por EMA, con un promedio de 300 m, que es menor que el radio horizontal permitido (r_{max}).

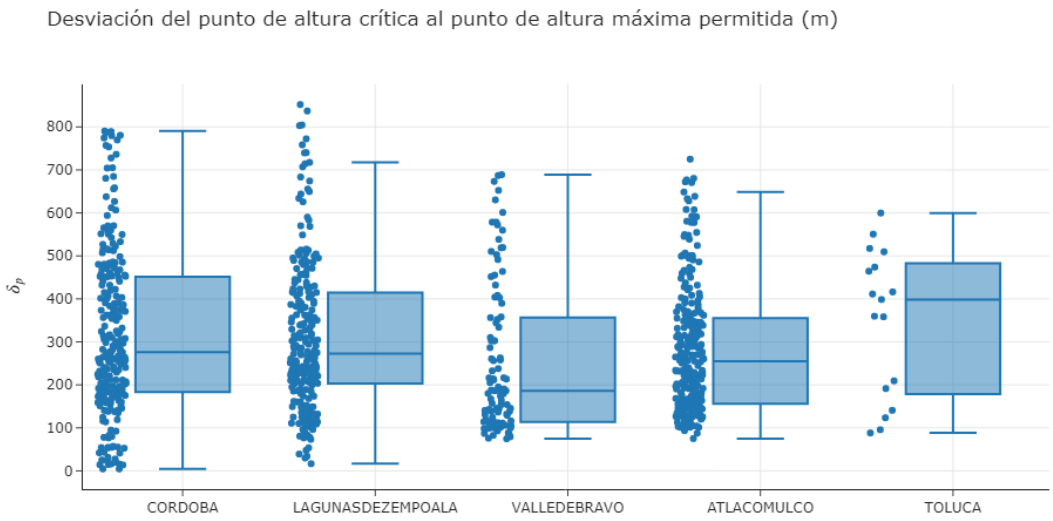


Figura 3.15: Desviación en términos de la latitud y longutd δ_p de la carga útil partiendo de la altura crítica rescto a la altura máxima permitida, por EMA.

Una interpretación más intuitiva es que una vez alcanzada la altura crítica, en promedio se desvía 300 m al punto donde se puede alcanzar según las normas federales, es decir, el punto de despegue de los UAV's puede estar a un radio de 300 m para que tengan buenas posibilidades de capturar la carga útil. Nótese que también existe una alta variación en esa métrica, llegando a casos donde $\delta_p > r_{max}$, que en esos casos se estarían violando las restricciones de vuelo por ley, por lo que, lanzar en TOLUCA no es recomendable porque el δ_p promedio es de 400 m, valores muy cercanos a lo permitido.

Una vez determinadas todas las métricas, el objetivo del siguiente paso es generar un archivo CSV por cada trayectoria simulada tal que contenga la información de la latitud, longitud y altitud de la carga útil en cada *frame* (dado que la escala del tiempo en el entorno de simulación está en FPS) dentro de la altura crítica durante el descenso, reduciendo así el tiempo de entrenamiento al acotar el tiempo de interés.

Las trayectorias simuladas generadas con CUSF contienen la información de latitud, longitud y altitud minuto a minuto como se observa en la Fig. 3.16. Para tener el detalle *frame* a *frame* (e.g., q segundo entre *frames*) se utilizaron *splines* naturales que son un método de interpolación de datos, es decir, coloca una cierta cantidad x de puntos intermedios entre 2 datos conocidos.

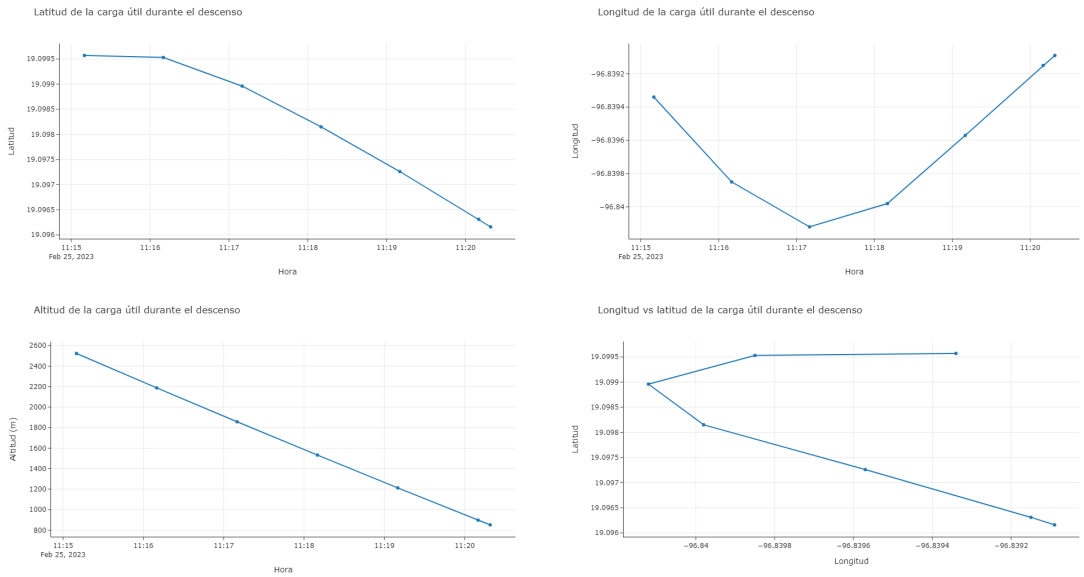


Figura 3.16: Gráficas que muestran la dinámica de la latitud, longitud y altitud de la carga útil en función del tiempo durante el descenso, para una trayectoria en específico. La información de las trayectorias generados en CUSF está en una escala de minutos.

Para el cálculo del número de puntos a colocar n_{spline} en el *spline natural* se divide la curva en $n-1$ segmentos, donde n es el número de puntos reales en el conjunto de datos. Cada segmento está definido por un nodo inicial y final, que corresponde a una ventana de tiempo de 1 minuto, por lo que, se deberán colocar un total de 60 puntos para tener la escala del tiempo en segundos, salvo el último segmento que tiene una duración variable, este último segmento tendrán $m+1$ puntos colocados donde m es la duración en segundos del último segmento, por lo tanto la ecuación que calcula el número de puntos del *spline natural* se define como:

$$n_{spline} = (n-2) \times 60 + (m+1). \tag{3.4}$$

La simulación se ejecuta en *frames* por segundo (FPS), por lo que la ecuación anterior se debe ajustar para calcular el número de *frames* en la simulación, de tal manera que se obtenga la información del descenso en cada *frame*, es decir, el tiempo se debe multiplicar por los FPS, obteniendo la siguiente ecuación:

$$n_{spline} = FPS \times [(n-2) \times 60 + m] + 1. \tag{3.5}$$

Por ejemplo, si una trayectoria simulada tiene un total de 7 puntos que representan la latitud, longitud y altitud de la carga útil en cada minuto, se tiene un total de 6 segmentos $(n-1)$, donde 5 de ellos $(n-2)$ tienen una duración de 60 segundos y el último tiene una duración de 9 segundos, y considerando que la simulación va a 30 FPS[8], entonces se deben colocar 9,271 puntos para tener toda la trayectoria en una escala de *frames*:

[8]Incrementar el número de FPS incrementa el tamaño del archivo CSV, así como el tiempo de entrenamiento.

$$n_{spline} = FPS \times [(n-2) \times 60 + m] + 1 = 30 \times [(7-2) \times 60 + 9] + 1 = 9,271. \quad (3.6)$$

En la Figs. 3.17, 3.18, 3.19 se muestran los datos resultantes para una trayectoria específica que están dentro del rango de la altura crítica durante el descenso. Los puntos son los datos conocidos en cada minuto, la línea naranja representa el caso en que se haga una interpolación lineal y en verde están los $n_{spline} = 9,271$ datos colocados utilizando *spline naturales*[9], esto se ejecuta para cada variable de interés: latitud, longitud y altitud, por lo tanto, al final se obtiene la información de cada variable en una escala en segundos, con la finalidad de utilizar dicha información en el entorno de simulación.

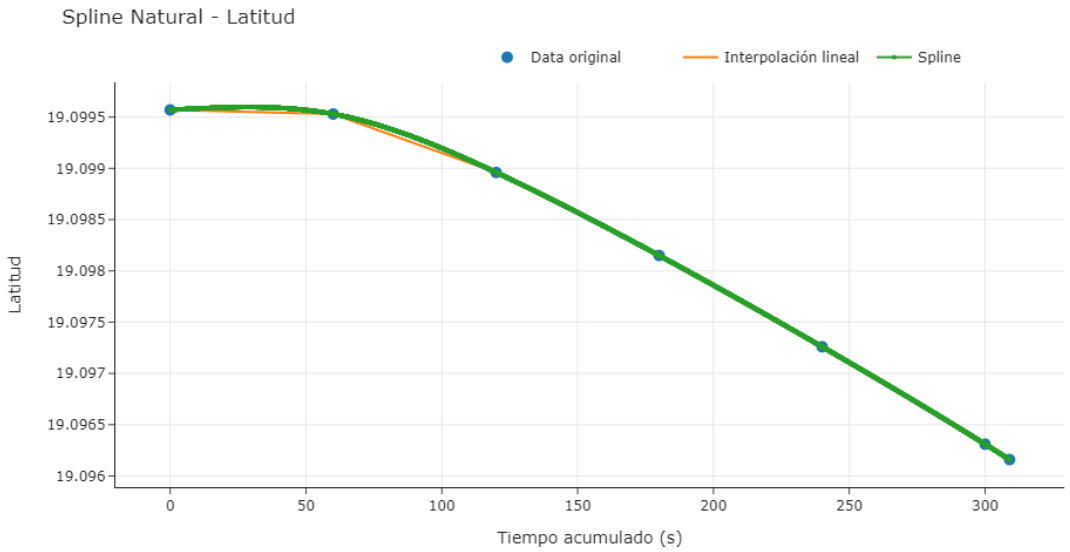


Figura 3.17: Interpolación de datos con *spline natural* para la latitud.

[9]Nótese que el número de datos colocados corresponden a la trayectoria analizada en cuestión, cada trayectoria tendrá su propio número de datos colocados calculados con la Ec. (3.5) en función de la duración en segundos del último segmento.

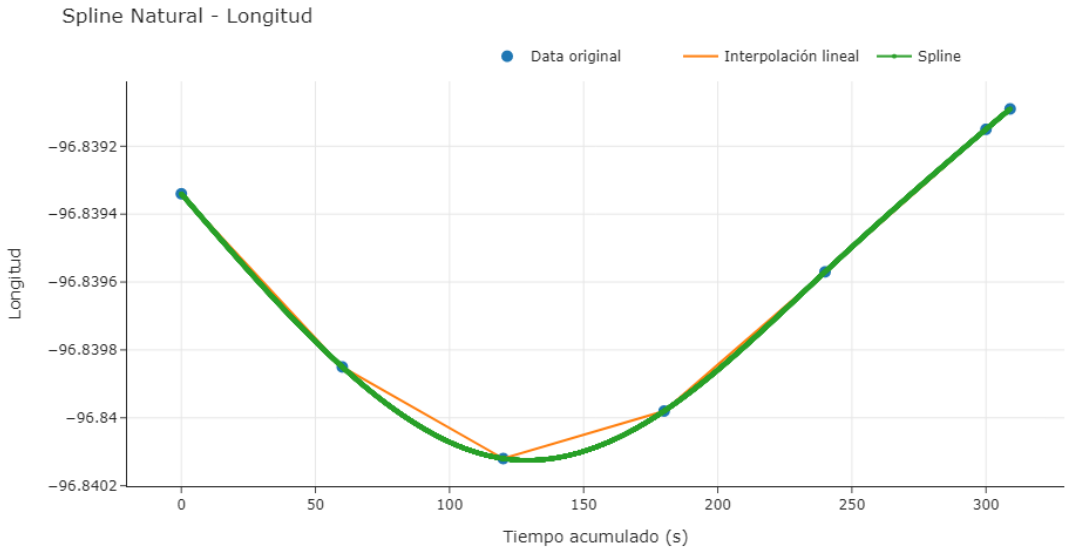


Figura 3.18: Interpolación de datos con *spline natural* para la longitud.

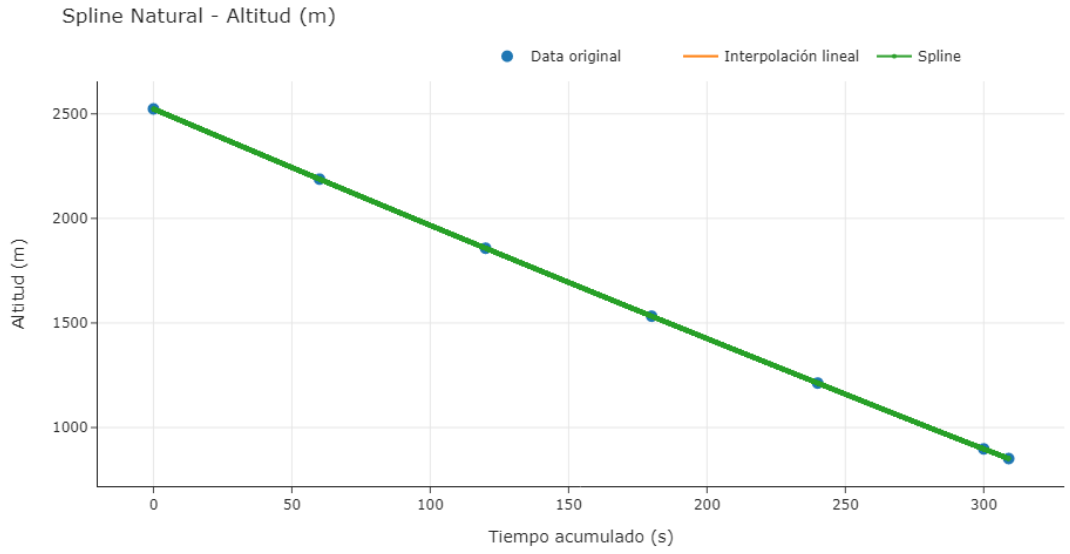


Figura 3.19: Interpolación de datos con *spline natural* para la altitud.

En la Fig. 3.20 se muestra un *box plot* donde se observa el número de puntos colocados utilizando los *splines* dentro de la altura crítica para cada EMA, el valor promedio es de 216 puntos en cada archivo CSV, además se observa que existen trayectorias con un número pequeño de observaciones, como VALLEDEBRAVO Y TOLUCA, con eso en consideración se decidió generar los archivos CSV para aquellas trayectorias que tuvieran por lo menos 100 puntos, generando así al final de todos los pasos un total de 1,159 archivos CSV en el banco de trayectorias.

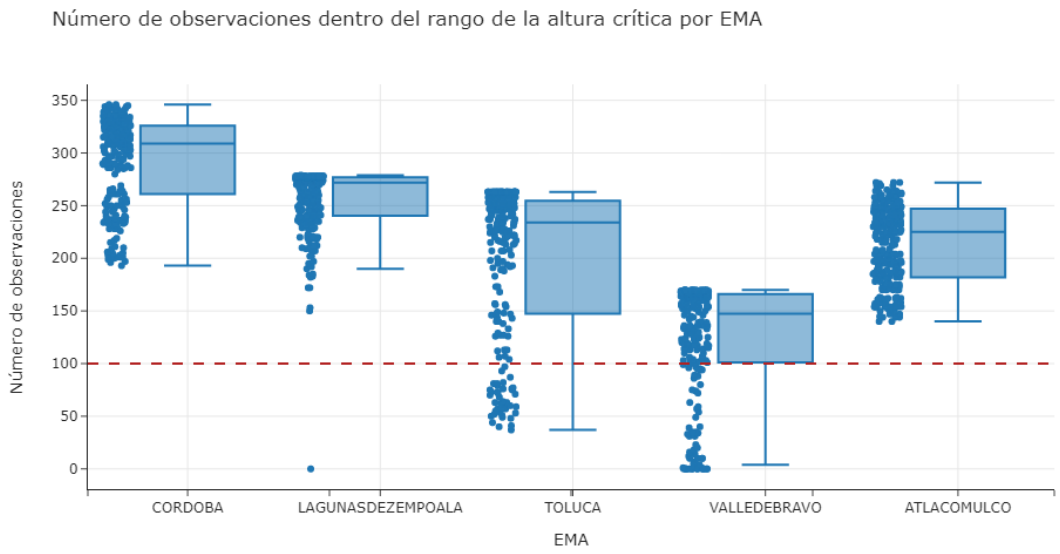


Figura 3.20: Número de observaciones dentro del rango de la altura crítica h_{cr} para cada EMA.

3.2. Modelo de la formación Multi-UAV's

Una vez ya obtenidos todas las métricas importantes en el sistema (calculadas en la sección 3.1) se prosigue a calcular un punto de despegue P (definido en la Fig. 3.13) de los UAV's tal que se tengan posibilidades de realizar el rescate de la carga útil. Para ello se realizó un algoritmo que para cada trayectoria simulada obtiene la posición más alejada (x_{max}, y_{max}) de un punto de referencia, que por simplicidad se decidió utilizar un sistema cartesiano cuyo punto de referencia es el origen $(0, 0, 0)$.

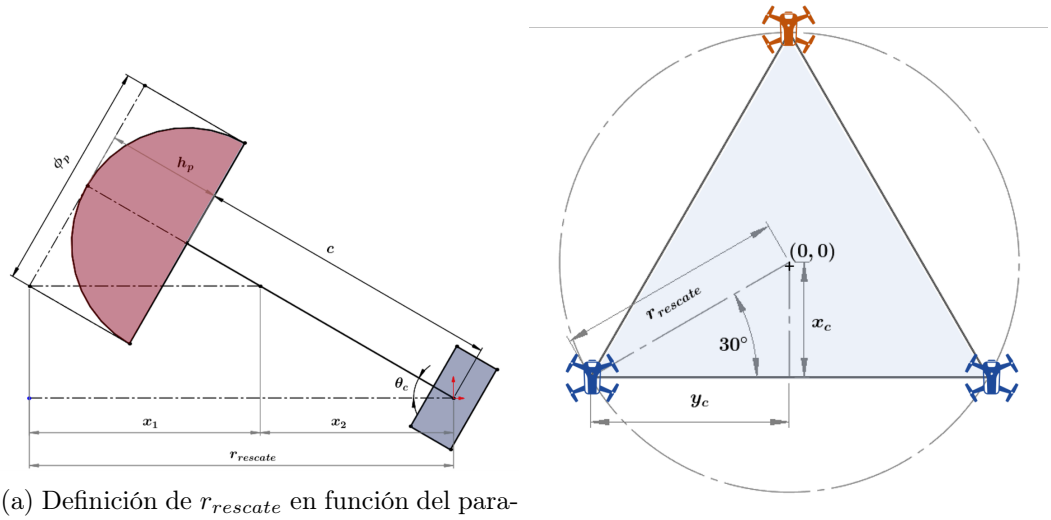
Una vez identificado el punto más alejado se calculó la posición a la que debe estar el punto P, que como se mencionó los UAV's no deben de volar más allá de 457 m de distancia horizontal (r_{max}), tal que $P = (x_{max} - r_{max}, y_{max} - r_{max})$. Con esa información se calculó la latitud y longitud del punto P. Adicionalmente se calcularon las posiciones x, y de la carga útil si el punto de despegue P se encontrara en el origen, es decir, si el sistema se traslada al punto $(0, 0, 0)$ con el objetivo de que la simulación siempre empiece en el mismo punto, aunque se tiene la libertad de seleccionar el origen que más se adecúe a las necesidades.

El siguiente paso es parametrizar la formación de los UAV's en función de un radio de rescate $r_{rescate}$, dicho radio es el área de recepción de la carga útil en una red, para calcular dicho radio se consideran las dimensiones de la carga útil[10] junto con las del paracaídas en una posición crítica $\theta_c = 20°$, además de añadir una tolerancia con los objetivos de evitar alguna colisión del paracaídas con los UAV's y de tener

[10]Dimensiones de la carga útil de un kit que vende Strato Flights (Fig. 3.21).

una mayor área de recepción, como se muestra en la Fig. 3.22a. El valor de rescate definido para la presente tesis es de 4 m, una vez definido su valor, se colocan 3 UAV's de tal manera que formen un triángulo para extender la red y asegurar que se tenga dicho radio de rescate, tal como se muestra en la Fig. 3.22b.

Figura 3.21: Kit de Strato Flights. Tomado de [98].



(a) Definición de $r_{rescate}$ en función del paracaídas.

(b) Formación triangular de los UAV's.

Figura 3.22: El tamaño del triángulo de la formación está en función del radio de rescate. El esquema líder-seguidor toma a un UAV como líder (naranja), mientras que los UAV's restantes (azules) son los seguidores que mantendrán la formación triangular en cada paso de tiempo.

La posición en el plano XY del UAV líder está definida como $P_1 = (x_0 + r_{rescate}, 0)$, mientras que la posición del UAV que se encuentra en la esquina inferior izquierda está definida como $P_2 = (x_0 - x_c, y_0 - y_c)$ y la posición del UAV en la esquina inferior derecha está definida como $P_3 = (x_0 - x_c, y_0 + y_c)$, donde:

$$(x_0, y_0) = (0, 0), \tag{3.7}$$

$$x_c = \frac{r_{rescate}}{2}, \tag{3.8}$$

$$y_c = \frac{\sqrt{3}}{2} r_{rescate}. \tag{3.9}$$

El archivo CSV de cada trayectoria cuenta con la información de la latitud, longitud y altitud de la carga útil *frame* por *frame* resultantes de realizar el *spline* a las trayectorias obtenidas por el software CUSF, así como la transformación de dichas coordenadas a un sistema cartesiano, adicionalmente en esta sección se añadió el punto de despegue P de los UAV's.

3.3. Modelo de la red

El CAD de la red se realizó con el software Blender, la red es un triángulo parametrizado con el radio de rescate $r_{rescate}$ mencionado en la sección anterior, además se modeló como una tela empotrada a cada una de las aristas y que está sujeta a la gravedad, esto con el objetivo de emular la forma de la red dentro de la simulación, como se muestra en la Fig. 3.23.

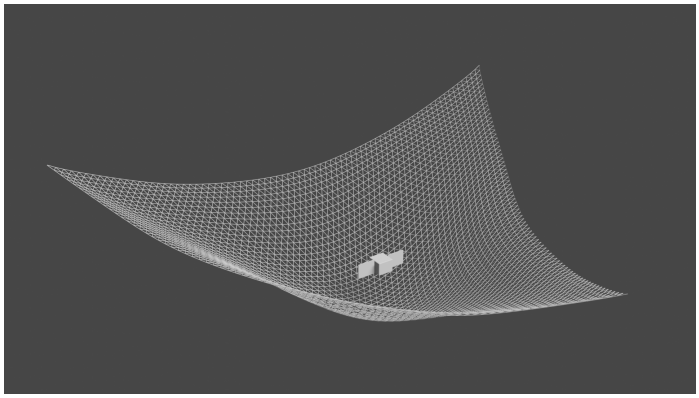


Figura 3.23: Red y carga útil.

El movimiento de la red dentro de la simulación está en función de la posición de los UAV's, tal que en cada paso de tiempo se obtiene la posición de cada uno de los UAV's, para que posteriormente se le asignen los valores promedio (x, y, z) al centro de la red, de esta manera se emula el movimiento de la red en conjunto de los UAV's. En la Fig. 3.24 se muestra el *Blueprint*[11] en donde se modela el movimiento de la red.

[11]Los *Blueprints* en *Unreal* son una representación gráfica que permite crear interacciones y la lógica del ambiente de simulación sin necesidad de escribir código, aunque también se pueden realizar funciones propias e incluirlas en los *Blueprints*

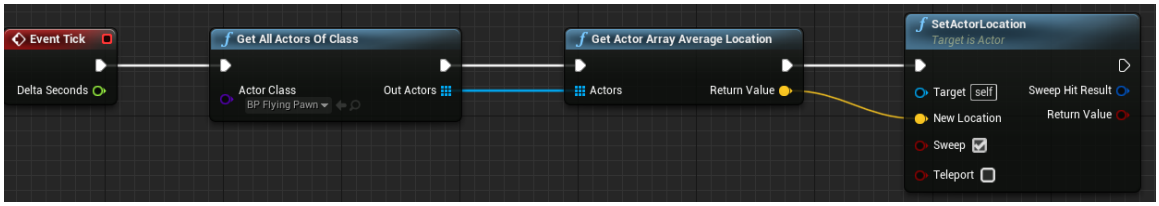


Figura 3.24: *Blueprint* de *Unreal* del comportamiento de la red en cada *tick*.

Si bien, se podrían incorporar algunas interacciones más complicadas de la red con los UAV's para disminuir el error en la posición de la red en cada paso de tiempo, o inclusive simular la dinámica de la red, se decidió mantener a la red como un cuerpo rígido y que su posición sea la posición promedio de la formación de los UAV's para agilizar el tiempo de entrenamiento sin comprometer los resultados.

3.4. Descripción del entorno de simulación

El entorno de simulación consta de tres agentes que están interactuando: la red, la carga útil y conjunto de UAV's. En las secciones anteriores se describió la dinámica de cada uno de los agentes dentro de la simulación, en este apartado se abordarán las interacciones y el flujo del entorno de simulación.

La definición de las interacciones entre los agentes es un factor importante para el correcto desempeño de la simulación, la información de las colisiones son recolectadas por medio de la API de *AirSim*, dicha información se toma como entrada en cada observación del algoritmo de aprendizaje por refuerzo. A continuación, se describen las interacciones de cada agente con el entorno:

- Los UAV's detectan colisiones con la carga útil y con cualquier objeto en el entorno virtual (e.g., el piso), ignorando las colisiones con la red.
- La red detecta colisiones con la carga útil, ignorando las colisiones con los UAV's y con cualquier objeto en el entorno virtual.
- La carga útil detecta colisiones con los UAV's, la red y cualquier objeto en el entorno de simulación.

Para modelar las colisiones es necesario definir las mallas de colisión, dichas mallas se superponen a los modelos 3D, lo que permiten definir con precisión las áreas de interacción de los objetos en el mundo virtual. Las mallas de colisión compleja permiten representar formas más detalladas y precisas, ofreciendo una mayor fidelidad en la detección de colisiones, lo que resulta en interacciones más realistas.

Para cada uno de los agentes es necesario definir la malla de colisión, en el caso de los UAV's la malla ya viene integrada en *AirSim*, para la red y la carga útil se realizaron mallas complejas de colisión, tal que capturaran la forma de cada uno (Fig.

3.25).

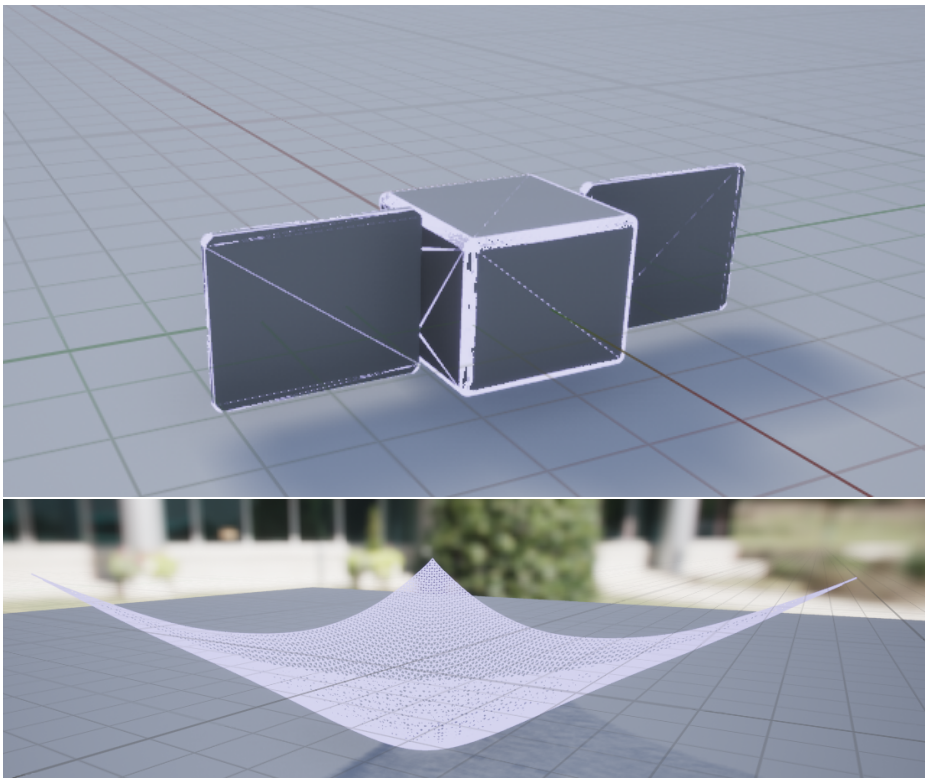


Figura 3.25: Malla de colisión compleja para la carga útil y la red. La malla está representada por el color lavanda.

Una vez definidas las interacciones y mallas de colisión de los agentes, se prosigue a desarrollar el flujo del que tendrá la carga útil en el entorno de simulación. En el *setup* de la simulación se colocan los UAV's en el origen $(0, 0, 0)$, el cual representa el punto P de despegue, es decir, el sistema se traslada a dicho origen tal como se discutió en la sección 3.2. En la Fig. 3.26 se muestra el proceso que se realiza al iniciar una simulación, que en pocas palabras toma un archivo de manera aleatoria del banco de trayectorias, y después escribe en un archivo CSV compartido la dirección del archivo que está leyendo, con el objetivo de que ese archivo sea el puente de comunicación entre *AirSim* y *Unreal*.

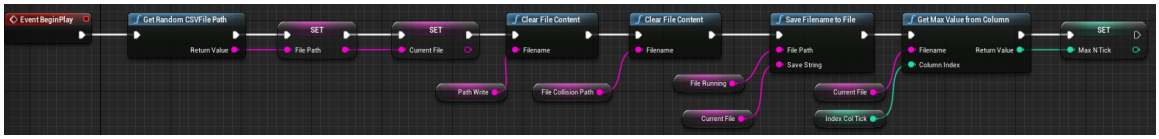


Figura 3.26: Flujo realizado en el paso inicial de la simulación.

En la Fig. 3.27 se muestra el flujo que sucede en cada paso de tiempo en la carga útil. Para cada paso de tiempo primero se determina si la trayectoria de la carga útil

se reinició, es decir, si la simulación se reinició, posteriormente se determina si la carga útil ya colisionó con la red. En caso no existir colisión con la red, se abre el archivo CSV seleccionado aleatoriamente en el paso inicial, y se lee la fila que corresponde al número de *tick* (que es el mecanismo central que permite que la simulación ejecute el código cuadro a cuadro) para obtener la información de la posición de la carga útil en *tick* particular. Adicionalmente se realizan verificaciones de que la altitud de la carga útil sea mayor que cero, esto para evitar algún problema numérico en la simulación. Posteriormente se le asigna la posición obtenida del archivo CSV a la carga útil en el entorno de simulación. Finalmente se incrementa un contador que registra el número de *ticks* realizados, y adicionalmente se guarda la información en un archivo CSV para tener registro en cada paso de tiempo la posición de la carga útil y así poder validar el correcto funcionamiento del entorno de simulación respecto a la dinámica de la carga útil.

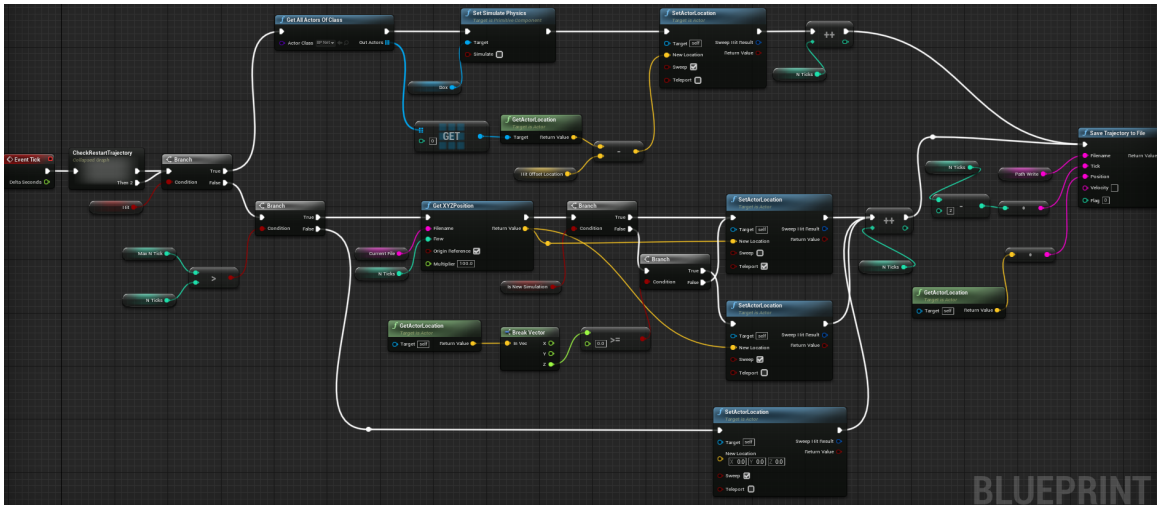


Figura 3.27: Flujo realizado en cada paso de tiempo.

En caso de que la carga útil ya haya colisionado con la red, en primera instancia se desactiva la física de la carga útil en el entorno de simulación, i.e., ya no estará sujeta a las leyes físicas en el entorno de simulación, esto es con la finalidad de mover a la carga útil en conjunto con la red. Para realizarlo, primero se obtiene la posición de la red, para después asignar la posición de la carga útil como la posición de la red menos la posición relativa de colisión, emulando así el transporte de la carga útil después de realizar la captura.

Ahora bien, falta definir cómo se identifica la colisión de la carga útil con la red. En la Fig. 3.28 se muestra el flujo que ocurre cuando se activa el evento de colisión, i.e., cuando la carga útil colisiona con algún objeto. En el momento en que se identifica una colisión, se escribe en un archivo CSV compartido entre *AirSim* y *Unreal* el objeto que colisionó y además con qué agente colisionó, permitiendo leer dicha información y saber si existieron colisiones durante el entrenamiento. Una vez escrita

dicha información en el archivo compartido se genera una bandera indicando que la carga útil colisionó, y además se guarda la posición relativa de la carga útil respecto a la red en el momento del impacto, dicha posición es el vector de posición relativa de colisión mencionado en el párrafo anterior.

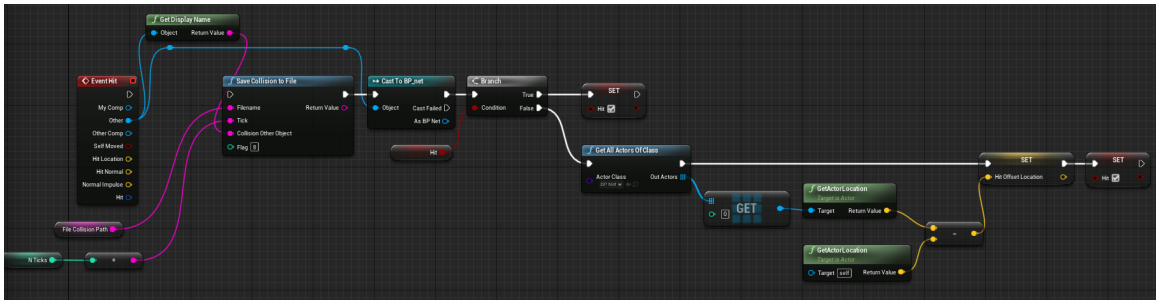


Figura 3.28: Flujo realizado en el momento que se origina una colisión entre la carga útil y la red.

Es importante mencionar que tal como está programado el flujo de la carga útil en cada paso de tiempo, es posible que la carga útil haga contacto con la parte exterior de la red considerándolo como una captura exitosa, lo cual no es verdad. Para atacar este problema se decidió incorporar en la recompensa del algoritmo de aprendizaje por refuerzo que la carga útil además de hacer contacto con la red tiene que ingresar por el área formada por el triángulo que conforma la red.

Al final de cada paso de tiempo se guarda en un archivo CSV con la información de la posición en cada *tick* de la carga útil. Como ejercicio se validó que efectivamente la carga útil esté realizando la trayectoria definida por el archivo seleccionado aleatoriamente del banco de trayectorias, la escritura y guardado del CSV cuando existe una colisión de la carga útil y de los UAV's.

4 Desarrollo del modelo de aprendizaje por refuerzo

El objetivo de este capítulo es describir los distintos componentes del problema modelado como aprendizaje por refuerzo, es decir, el espacio de acciones, el espacio de observaciones y la señal de recompensa descompuesta en múltiples componentes para que la formación de UAV's realice la captura de una carga útil de manera exitosa, en el menor tiempo posible y dentro del rango legalmente permitido por las leyes mexicanas.

En el *framework* propuesto en la sección 3 existe una capa donde se aplican algoritmos de aprendizaje por refuerzo, dichos algoritmos se aplican mediante la biblioteca *Stable Baselines3* [99][1] que tiene una amplia gama de algoritmos compatibles con espacios de acciones continuos y discretos, permitiendo a los usuarios centrarse en la formulación del problema y la experimentación con diversos algoritmos sin necesidad de realizar implementaciones desde cero.

En la Fig. 4.1 se muestran los distintos algoritmos implementados en la biblioteca *Stable Baselines3*, así como el soporte de un espacio de acciones discretas o continuas para cada algoritmo, siendo una referencia para seleccionar el o los algoritmos a iterar para resolver el problema. Las características `Box, Discrete, MultiDiscrete, MultiBinary` en el encabezado de la tabla muestran qué tipos de espacio de acciones utilizan los algoritmos, siendo [99]:

- `Box`: Una caja de N dimensiones que contine cada punto del espacio de acciones, es decir, un espacio de acciones continuo.
- `Discrete`: Una lista de acciones posibles, donde en cada paso de tiempos sólo se puede utilizar una de las acciones.
- `MultiDiscrete`: Una lista de acciones posibles, donde en cada paso de tiempo sólo se puede utilizar una acción de cada conjunto discreto.
- `MultiBinary`: Una lista de acciones posibles, donde en cada paso de tiempo se puede utilizar cualquiera de las acciones en cualquier combinación.

[1] En el presente trabajo se utilizó la versión 1.6.1 porque era la última versión disponible cuando se comenzó a realizar los distintos experimentos, actualmente existen versiones posteriores que incluyen más algoritmos.

Name	Box	Discrete	MultiDiscrete	MultiBinary	Multi Processing
ARS [1]	✓	✓	✗	✗	✓
A2C	✓	✓	✓	✓	✓
DDPG	✓	✗	✗	✗	✓
DQN	✗	✓	✗	✗	✓
HER	✓	✓	✗	✗	✗
PPO	✓	✓	✓	✓	✓
QR-DQN [1]	✗	✓	✗	✗	✓
SAC	✓	✗	✗	✗	✓
TD3	✓	✗	✗	✗	✓
TQC [1]	✓	✗	✗	✗	✓
TRPO [1]	✓	✓	✓	✓	✓
Maskable PPO [1]	✗	✓	✓	✓	✓

Figura 4.1: Algoritmos de aprendizaje por refuerzo implementados en *Stable Baselines3*, junto con algunas características: soporte para acciones discretas o continuas, multiprocesamiento. Tomada de [99].

Para entrenar los algoritmos de aprendizaje por refuerzo es necesario tener un entorno de simulación. Es posible realizar entornos personalizados y entrenar los algoritmos de la biblioteca *Stable Baselines3* extendiendo métodos *Gym* de *OpenAI*. El entorno necesita ser una clase heredada de `gym` de OpenAI con los siguientes métodos:

- `__init__`: Es el constructor de la clase del entorno de simulación, y aquí es necesario definir el estado de acciones `action_space` con las características mencionadas en la Fig. 4.1 y el estado de observaciones `observation_space`, siendo una clase especial de *Gym*.
- `reset`: Este método reinicia el entorno, regresando el valor del estado de observaciones, es decir, comienza una nueva simulación cuando termina la anterior (e.g., por una colisión de los UAV's).
- `step`: El método tiene un parámetro de entrada que es un valor de acción (`action`) que se encuentre dentro del estado de acciones `action_space`. El agente realiza la acción deseada y el entorno devuelve el estado al que le ha llevado realizar esa acción en una tupla de 4 elementos:
  - `state`: La información del estado al que se llega por realizar la acción `action`.
  - `reward`: La recompensa obtenida por realizar la acción.

- `done`: Un valor booleano, donde el valor positivo es cuando se alcanzó un punto final y tiene que ser reiniciado, y el negativo en caso contrario.
- `info`: Un diccionario para ir guardando información, es útil para la detección y corrección de errores.

Además, es posible utilizar *AirSim* como un entorno de entrenamiento o gimnasio mediante una envolvente de *Gym*, donde es necesario extender y reimplementar los siguientes métodos específicos: `step`, `_get_obs`, `_compute_reward`, `reset` [99], que se explicarán más adelante.

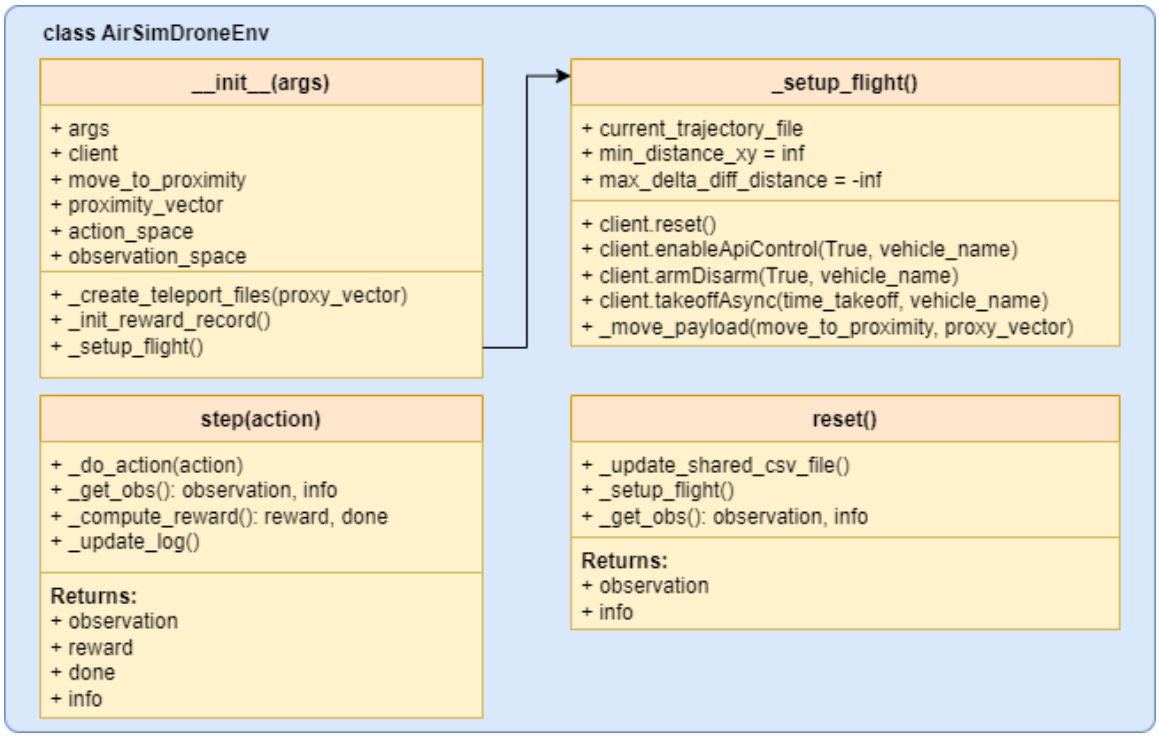


Figura 4.2: Esquema a alto nivel (no exhaustivo) de la clase `AirSimDroneEnv` que contiene los principales atributos y métodos empleados en el presente trabajo para realizar el entrenamiento en un entorno personalizado.

En la Fig. 4.2 se muestra un esquema a alto nivel de los principales métodos utilizados para poder realizar el entrenamiento en un entorno personalizado de *AirSim*, sobrescribiendo los métodos de *Gym*. A continuación, se describirán con más detalle con el objetivo de brindar un mejor entendimiento del algoritmo implementado, así como los principales atributos que pueden considerarse como hiperparámetros que se pueden modificar para generar distintos experimentos.

- El constructor `__init__` toma una serie de argumentos (`args`) que permiten inicializar el entorno de simulación (e.g., el valor de las recompensas asociados a cada componente, velocidad de los UAV's, si es discreto o continuo el espacio de acciones), así como:

- La conexión con *AirSim* mediante `client`.
- El argumento `move_to_proximity` es una opción para indicar si la formación de UAV's se moverá artificialmente a un punto cercano de la carga útil antes de comenzar a realizar el entrenamiento. Las opciones son `None` donde no se realiza ningún movimiento, `move` que indica que la formación de UAV's se mueve a una posición definida por el vector `proxy_vector` en línea recta y una vez alcanzada esa posición se comienza a realizar el entrenamiento, y por último la opción `teleport` que emula el comportamiento de la opción `move` pero en lugar de mover la formación de UAV's a la posición `proxy_vector` mejor inicializa la carga útil en una posición relativa al `proxy_vector` disminuyendo el tiempo de entrenamiento porque se evita simular el movimiento de los UAV's a `proxy_vector` en una línea recta, tal que:

$$(x, y)_{payload} = (x, y)_{payload,inicial} - (x, y)_{proxy\_vector}, \tag{4.1}$$

  donde $(x, y)_{payload,inicial}$ es la posición original de la carga útil en la simulación de su trayectoria. La altura se coloca directamente del componte de `proxy_vector`, tal que $z_{payload} = z_{proxy\_vector}$.
- La definición del estado de acciones `action_space` necesario para utilizar *Gym*.
- La definición del estado de observaciones `observation_space` necesario para utilizar *Gym*.

- Adicionalmente se tienen métodos que sirven para inicializar el entorno de simulación:
  - El método `_create_teleport_files(proxy_vector)` como entrada el argumento `proxy_vector` que es un vector definido en sus componentes (x, y, z) que determina en qué posición relativa se inicializará la carga útil para comenzar su descenso, modificando los archivos CSV que contienen la simulación de la trayectoria de la carga útil en los distintos escenarios, y que se leen desde *Unreal* para simular el descenso de la carga útil.
  - El método `_init_reward_record()` inicializa un diccionario donde se guarda información relevante para la detección y corrección de errores, manteniendo el registro del valor de los distintos componentes de la recompensa, así como sus principales parámetros.
  - El método `_setup_flight()` es utilizado para inicializar los controles de los UAV's mediante la API de *AirSim* ejecutando una serie de métodos: `enableApiControl`, `armDisarm`, `takeoffAsync`, `join` por medio de la conexión `client`, permitiendo así poder mandar comandos de movimiento a la formación de UAVS's (e.g., mover a un UAV con una cierta velocidad en cada uno de sus componentes). Adicionalmente tiene otros atributos y métodos importantes:

  - El atributo `current_trajectory_file` contiene el nombre del archivo CSV que se está utilizando para simular la trayectoria.
  - El atributo `min_distance_xy` es la distancia euclidiana[2] mínima global, que busca incentivar a la formación a acercarse lo más rápido posible a la carga útil (véase la subsección 4.3.1), inicializándose con un número muy grande.
  - El atributo `max_delta_diff_distance` es la máxima diferencia de la distancia mínima global respecto a la distancia actual, que busca incentivar a la formación a acercarse lo más rápido posible a la carga útil (véase la subsección 4.3.1), inicializándose con un número muy pequeño.
  - El método `_move_payload(move_to_proximity, proxy_vector)` realiza el movimiento de la formación de UAV's al punto definido por `proxy_vector` en caso de que `move_to_proximity = move`.

- El método `step` es obligatorio para generar el entorno personalizado, que toma como entrada una acción `action`, ejecuta la acción en el entorno con el método `_do_action(action)`, después se obtiene la observación (`observation`) que genera el entorno así como un diccionario con información relevante (`info`) con el método `_get_obs()` y posteriormente se calcula la recompensa (`reward`) y si se alcanzó un punto final (`done`) utilizando el método `_compute_reward()` [3]. Adicionalmente se generó el método `_update_log()` que actualiza información relevante del entrenamiento con el objetivo de validar el correcto funcionamiento del código (e.g., la acción realizada y su respectiva recompensa en el paso de tiempo). El método retorna una tupla de 4 elementos: `observation, reward, done, info`.

- El método `reset` reinicia el entorno de simulación cuando se llega a un punto final (e.g., captura exitosa de la carga útil, colisión de los UAV's con algún objeto, colisión de la carga útil con algún elemento que no sea la red, fin de la trayectoria simulada). Para resetear el entorno de simulación se selecciona un nuevo archivo CSV que contiene una nueva trayectoria y se actualiza en el archivo compartido que tiene *AirSim* con *Unreal*, definiendo un nuevo episodio por medio del método `_update_shared_csv_file()`. Posteriormente se inicializa la conexión de los UAV's y los parámetros iniciales con el método `_setup_flight()`, y por último se obtiene la observación inicial del entorno con el método `_get_obs()`, retornando la observación `observation` y el diccionario de información `info`.

[2]Es importante mencionar que en el presente trabajo se utilizó únicamente distancia euclidiana, por lo que, cuando en el texto se habla acerca de la distancia hace referencia a la distancia euclidiana.

[3]Los métodos `_get_obs()` y `_compute_reward()` son necesarios para el entrenamiento en el entorno utilizando *AirSim*.

4.1. Descripción del espacio de acciones

El espacio de acciones es el conjunto de las posibles acciones que la formación de UAV's pueden realizar, pudiendo ser discreto, continuo o híbrido. En el presente trabajo se desarrollaron distintas opciones con el fin de poder experimentar en trabajos futuros el impacto de seleccionar un conjunto de acciones respecto a otro, pero únicamente se utilizaron dos opciones que se describirán en la presente sección. En cada opción, se traducen las acciones seleccionadas del espacio de acciones a un vector de velocidad $\boldsymbol{v} \in [-v_{max}, v_{max}] \in \mathbb{R}^3$ con la velocidad para cada componente del UAV líder[4] permitiendo mover a la formación de UAV's, tal que:

$$\boldsymbol{v} = (v_x, v_y, v_z)\,. \tag{4.2}$$

1. El espacio de acciones de la primera opción está compuesto por un conjunto discreto de 4 elementos. Los primeros 3 elementos están dados por un conjunto de 3 posibles acciones que definen la dirección unitaria (atrás, mantenerse en la misma posición, adelante) $\{-1, 0, 1\}$ para cada uno de los componentes de la velocidad. Mientras que el último componente es para definir el valor de $\lambda_v \in [0, 1]$ que representa la proporción respecto a la velocidad máxima de los UAV's v_{max}. El hiperparámetro `discrete_vel` define el número de intervalos en el que se discretiza el espacio $\in [0, 1]$ (e.g., si `discrete_vel = 5` $\rightarrow (0, 25\,\%, 50\,\%, 75\,\%, 100\,\%)$). El espacio de acciones para esta opción está entonces, definido como:

$$\mathcal{A} = \{(v_x, v_y, v_z) \in \{-1, 0, 1\}\,, \lambda_v \in \{0, 0.25, 0.5, 0.75, 1\}\}\,. \tag{4.3}$$

   La forma de definir el espacio de acciones en *Gym* es por medio de las funciones descritas `Box, Discrete, MultiDiscrete, MultiBinary` en el inicio del capítulo. En el Código 4.1 se muestra la forma declarada del espacio de acciones en *Gym* para la primera opción, siendo valores discretos para los 4 elementos del espacio de acciones.

```
MultiDiscrete([3,3,3,discrete_vel])
```

Código 4.1: Opción discretizada de la dirección en cada componente y discretizada en la proporción de la velocidad λ_v.

   Ahora bien, el vector de velocidad $\boldsymbol{v}$ que mueve a la formación de UAV's está definido por la acción seleccionada para cada componente $(v_{i;\mathcal{A}})$ multiplicado por un valor de velocidad $v \in \mathbb{R}^+$, tal que:

$$\boldsymbol{v} = v\,(v_{x;\mathcal{A}}, v_{y;\mathcal{A}}, v_{z;\mathcal{A}})\,, \tag{4.4}$$

[4]En el presente trabajo se utilizó el esquema líder-seguidor, es decir, el líder es el que se entrena con los algoritmos de aprendizaje por refuerzo, mientras que los seguidores siguen al líder pero manteniendo la formación original.

donde v es igual a la proporción de la velocidad máxima seleccionada del espacio de acciones ($\lambda_{v;\mathcal{A}}$) multiplicada por la velocidad máxima de los UAV's, siendo $v = \lambda_{v;\mathcal{A}} \cdot v_{max}$.

Esta opción permite realizar movimientos en cada componente x, y, z en cada paso de tiempo, pero con una misma velocidad v con diferente dirección (-1,0,1).

2. El espacio de acciones de la segunda opción está compuesto por un conjunto continuo de 3 elementos, donde cada elemento define la dirección y proporción en un rango de [-1,1] para cada componente de la velocidad respectivamente:

$$\mathcal{A} = \{(v_x, v_y, v_z) \in [-1, 1]\}. \tag{4.5}$$

En el Código 4.2 se muestra la forma declarada del espacio de acciones en *Gym* para la segunda opción, siendo valores continuos en un rango de [-1,1] para los 3 elementos del espacio de acciones.

```
Box(-1,1,shape=(1,3))
```

Código 4.2: Opción continua en la dirección y proporción de la velocidad

El vector de velocidad $\boldsymbol{v}$ que mueve a la formación de UAV's está definido por la acción seleccionada para cada componente ($v_{i;\mathcal{A}}$) multiplicado por la velocidad máxima de los UAV's, tal que:

$$\boldsymbol{v} = v_{max} \left(v_{x;\mathcal{A}}, v_{y;\mathcal{A}}, v_{z;\mathcal{A}}\right). \tag{4.6}$$

Esta opción permite realizar movimientos en cada componente con una velocidad distinta en cada dirección en cada paso de tiempo, pero incrementa el espacio de búsqueda. Adicionalmente ya no es necesario utilizar λ_v dado que al ser valores continuos entre $[-1, 1]$ se está calculando la proporción de la velocidad máxima.

Es importante mencionar que es posible generar más conjuntos de acciones para experimentar, pero se tiene pensado explorar en trabajos posteriores. La selección de la opción a utilizar en el entrenamiento está dada por el valor definido en el hiperparámetro `action_space_option`, siendo la primera opción `discrete_all` y la segunda `vel_continuous`.

4.2. Descripción del espacio de observaciones

El espacio de observaciones desempeña un papel fundamental en el modelado del entorno ya que proporciona al agente la información necesaria para tomar decisiones. En el presente trabajo el espacio de observaciones fue diseñado para capturar distintos aspectos, incluyendo la posición $\boldsymbol{p} \in \mathbb{R}^3$, velocidad $\boldsymbol{v} \in \mathbb{R}^3$ y altitud $h \in \mathbb{R}^+$ tanto de la carga útil ($\boldsymbol{p}_{carga}, \boldsymbol{v}_{carga}, h_{carga}$) como de la formación de los UAV's [5]

[5]La posición de la formación de los UAV's está definida como el centro de la formación.

$(\boldsymbol{p}_{UAVs}, \boldsymbol{v}_{UAVs}, h_{UAVs})$. Así como la distancia euclidiana relativa entre ambos $d_{rel} \in \mathbb{R}$.

Adicionalmente se incluyen variables binarias para indicar si la formación se encuentra dentro del rango horizontal $\delta_{rango,horizontal} \in \{0,1\}$ y altitud $\delta_{rango,altitud} \in \{0,1\}$ permitidos por las leyes mexicanas. El espacio de observaciones está definido como:

$$\begin{aligned}\mathcal{O} = \{&\boldsymbol{p}_{carga} \in \mathbb{R}^3 \mid (p_{x,carga}, p_{y,carga}, p_{z,carga}) \in [-2r_{max}, 2r_{max}],\\ &\boldsymbol{p}_{UAVs} \in \mathbb{R}^3 \mid (p_{x,UAVs}, p_{y,UAVs}, p_{z,UAVs}) \in [-2r_{max}, 2r_{max}],\\ &\boldsymbol{v}_{carga} \in \mathbb{R}^3 \mid (v_{x,carga}, v_{y,carga}, v_{z,carga}) \in [0, v_c],\\ &\boldsymbol{v}_{UAVs} \in \mathbb{R}^3 \mid (v_{x,UAVs}, v_{y,UAVs}, v_{z,UAVs}) \in [0, v_{max}],\\ &h_{carga} \in \mathbb{R} \mid h_{carga} \in [0, 2h_{max}],\\ &h_{UAVs} \in \mathbb{R} \mid h_{UAVs} \in [0, 2h_{max}],\\ &d_{rel} \in \mathbb{R} \mid d_{rel} \in [0, \infty),\\ &\delta_{rango,horizontal} \in \{0,1\},\\ &\delta_{rango,altitud} \in \{0,1\}\},\end{aligned} \tag{4.7}$$

donde r_{max} u h_{max} son los rangos horizontales y verticales por las leyes mexicanas, v_c es la velocidad de descenso máxima de la carga útil (véase 3.1.3) y v_{max} es la velocidad máxima de los UAV's. Las observaciones de posición es respecto al origen de la simulación que es el punto $(0,0,0)$,i.e., es el punto de despegue en todas las simulaciones de los UAV's.

El espacio de observaciones se definen en el código mediante las funciones `Box, Discrete, MultiDiscrete, MultiBinary`, tal como se muestra en el Código 4.3.

```
observation_space = Dict({
    'payload_position': Box(-MAX_RADIUS_HORIZONTAL*2,
   MAX_RADIUS_HORIZONTAL*2, shape=(3,), dtype=np.float32),
    'formation_position': Box(-MAX_RADIUS_HORIZONTAL*2,
   MAX_RADIUS_HORIZONTAL*2, shape=(3,), dtype=np.float32),
    'payload_velocity': Box(0, MAX_PAYLOAD_VELOCITY, shape=(1,)
   , dtype=np.float32),
    'formation_velocity': Box(0, DRONE_MAX_VELOCITY, shape=(1,)
   , dtype=np.float32),
    'payload_height': Box(0, MAX_ALTITUDE*2, shape=(1,), dtype=
   np.float32),
    'formation_height': Box(0, MAX_ALTITUDE*2, shape=(1,),
   dtype=np.float32),
    'distance_to_payload': Box(0, np.inf, shape=(1,), dtype=np.
   float32),
    'inside_permitted_radius': Discrete(2),
    'height_zone': Discrete(3),
        })
```

Código 4.3: Definición del espacio de observaciones.

4.3. Función de recompensa

En cada paso de tiempo la formación de UAV's realiza una acción, y el entorno le regresa a la formación de UAV's un nuevo estado en forma de observación, así como una recompensa. La función de recompensa retorna un valor numérico que permite guiar a la formación de UAV's para realizar el objetivo, que en este caso es la captura de una carga útil con una serie de restricciones, e.g., realizar la captura en el rango permitido legalmente.

La función de recompensa $R \in \mathbb{R}$ está compuesta por varios componentes en donde se tienen distintos incentivos (recompensas positivas) y penalizaciones o castigos (recompensas negativas) que se describirán en las siguientes subsecciones con mayor detalle, pero en general la formación de UAV's obtiene recompensas en cada paso de tiempo por:

- Aproximarse lo más rápido posible a la carga útil en cada paso de tiempo, obteniendo recompensas intermedias según se vaya aproximando los UAV's a la carga útil ($R_{dist} \in [0, 1000] \subset \mathbb{R}^+$).
- Estar dentro de una zona de captura, definida por una cierta geometría alrededor de la carga útil (e.g., una esfera de 3 metros de radio, siendo la carga útil el centro de la esfera), obteniendo nulas recompensas en zonas muy alejadas a la carga útil y mayores recompensas en zonas cercanas a la carga útil ($R_{zona} \in [0, 1000] \subset \mathbb{R}^+$).
- Estar dentro de la región horizontal y vertical permitida por las leyes mexicanas ($R_{rango} \in [0, 1000] \subset \mathbb{R}^+$).
- Tener la carga útil dentro de un prisma donde la base es el triángulo formado por los UAV's, el plano central es la posición de la formación de UAV's y la altura está definida a partir de un desfase por arriba del plano central $h_{zona,inf}$ y un desfase por abajo del plano central $h_{zona,sup}$, dicho prisma es dinámico y se mueve con la formación de UAV's. Esta recompensa ($R_{captura} \in [0, 10000] \subset \mathbb{R}^+$) busca guiar a la formación de UAV's a capturar la carga útil cuando ésta se encuentra muy cerca de la formación de UAV's.

Además, la formación de UAV's obtiene recompensas negativas o castigos por:

- Cada paso de tiempo que esté en el entorno de simulación, incentivando que realice la captura lo más rápido posible ($R_{tiempo} \in [0, 100] \subset \mathbb{R}^+$).

Las recompensas descritas anteriormente se obtienen en cada paso de tiempo mientras la formación de UAV's busca capturar la carga útil. Adicionalmente, existen recompensas positivas y negativas asociadas cuando se tienen eventos terminales, es decir, aquellos eventos que causan la finalización de un episodio. La formación de UAV's obtienen:

- Una recompensa positiva por capturar la carga útil de manera exitosa ($R_{captura} \in [0, 10000] \subset \mathbb{R}^+$), es decir, cuando la carga útil colisiona con la red. Nótese que la señal $R_{captura}$ también está en las recompensas descritas anteriormente, esto es porque dicha señal tiene un componente para guiar a la formación de UAV's para realizar la captura y la otra componente es la señal de captura exitosa.
- Una recompensa negativa asociada a la colisión de algún UAV de la formación con cualquier otro agente (piso, carga útil, red) dentro del entorno de simulación ($R_{colision} \in [0, 1000] \subset \mathbb{R}^+$).
- Una recompensa negativa asociada a la colisión de la carga útil con cualquier otro agente que no sea la red (piso o algún UAV) dentro del entorno de simulación ($R_{colision} \in [0, 1000] \subset \mathbb{R}^+$).
- Una recompensa negativa asociada a la finalización de la trayectoria de descenso de la carga útil, es decir, el número de puntos de la trayectoria simulada de la carga útil se terminaron ($R_{fin} \in [0, 1000] \subset \mathbb{R}^+$).

Por lo tanto, la señal de recompensa general R está dada por:

$$R = \begin{cases} R_{captura} & \text{si captura exitosa,} \\ -R_{colision} & \text{si existe alguna colisión,} \\ -R_{fin} & \text{si termina el descenso de la carga,} \\ R_{dist} + R_{zona} + R_{rango} + R_{captura} - R_{tiempo} & \text{de lo contrario.} \end{cases} \tag{4.8}$$

El rango propuesto en las distintas señales de recompensa es una primera iteración, dando un mayor rango a la recompensa por captura porque es el objetivo que se busca conseguir. La experimentación y ajuste en el valor de las distintas recompensas es una oportunidad para explorar en trabajos posteriores. Las recompensas negativas tienen un rango positivo dentro de los reales porque en la Ec. (4.8) están restando en la señal de recompensa general.

4.3.1. Recompensas por distancia

En esta sección se aborda el componente de la recompensa por distancia R_{dist}, cuyo objetivo es guiar a la formación mediante recompensas intermedias para que se acerque lo más rápido posible a la carga útil, siendo un componente muy importante, ya que de otra manera las recompensas serían más esparsas, lo que repercute en la convergencia y/o tiempo de entrenamiento de los algoritmos.

La señal de recompensa asociada a la distancia se divide en dos partes: búsqueda y rescate. La búsqueda está asociada para que la formación se acerque lo más rápido posible a una cierta zona que rodea a la carga útil, una vez dentro de esa zona la señal de recompensa cambiaría porque ahora lo que se busca es que la formación realice el rescate de una manera más suave y controlada.

Los componentes de la recompensa asociada a la distancia son:

- Una recompensa que busca incentivar a la formación a aproximarse lo más rápido posible a la zona de rescate ($R_{dist,min} \in \mathbb{R}^+$).
- Una recompensa que busca incentivar que la formación se encuentre siempre por debajo de la carga útil y que disminuya la diferencia de altitud de la formación respecto a la carga útil en cada paso de tiempo ($R_{altitud} \in \mathbb{R}^+$).
- Una recompensa que busca reforzar los dos componentes anteriores, al evaluar el ángulo θ respecto a Z, buscando ángulos $\theta \approx 90°$, lo que indicaría que la formación está por debajo de la carga útil, siendo una especie de sombra ($R_\theta \in \mathbb{R}^+$).

Por lo tanto, la señal de recompensas asociada a la distancia está dada por la suma de sus distintos componentes:

$$R_{dist} = R_{dist,min} + R_{altitud} + R_\theta \tag{4.9}$$

En el detalle, cada componente de la Ec. (4.9) tiene una serie de hiperparámetros escalares que definen el valor de la recompensa asociada a cada señal, e.g., la señal $R_{dist,min}$ definida en la Ec. (4.17) contiene los hiperparámetros[6] $r_{d,min}$, $r_{\delta,max}$, r_{prox}, r_{zona}, $r_{alejar} \in \mathbb{R}$, para cada uno es necesario colocar un valor de su respectiva recompensa, lo cual puede ser un reto por la cantidad de hiperparámetros que hay en el presente trabajo, dado que existe un amplio rango de valores con los que se pueden experimentar.

Para solucionar dicho reto, se propone una alternativa a la Ec. (4.9) tal que en lugar de asignar un valor dentro de un rango amplio para cada hiperparámetro, mejor se le asigna una valor $\in [0, 1]$ que puede ser interpretado como el nivel de importancia dentro de la recompensa. Adicionalmente, se generan un par de vectores, uno contiene cada componente de R_{dist} y el otro contiene la ponderación de cada componente, y por último, se define un escalar que define el valor máximo de recompensa que se puede obtener por el componente R_{dist}, siendo éste el único escalara que se necesita definir para R_{dist}. A continuación, se definirán dichos vectores y escalares.

El vector $\boldsymbol{r_{dist}}$ está compuesto por los distintos componentes de R_{dist} con la restricción de que cada componente tiene que estar en un rango $\in [0, 1]$ tal que:

$$\boldsymbol{r_{dist}} = [R_{dist,min}, R_{altitud}, R_\theta] \in \mathbb{R}^3 : 0 \leq R_{dist,min}, R_{altitud}, R_\theta \leq 1. \tag{4.10}$$

Asimismo, se define un vector de pesos $\mathrm{w}_{\boldsymbol{dist}}$ asociados a cada componente de $\boldsymbol{r_{dist}}$, ponderándolos según su nivel de importancia, con las restricciones de cada componente tiene que estar en un rango $\in [0, 1]$ y que la suma de sus componentes tiene que ser igual a 1, tal que:

$$\mathrm{w}_{\boldsymbol{dist}} = [w_1, w_2, w_3] \in \mathbb{R}^3 : 0 \leq w_i \leq 1 \text{ para } i = 1, 2, 3; \sum_{i=1}^{3} w_i = 1 \tag{4.11}$$

[6]Los hiperparámetros son definidos más adelante, únicamente se colocan en esta sección para fines ilustrativos.

Por último, se define un escalar $r_{max,dist} \in \mathbb{R}^+$ que es la recompensa máxima que puede recibir la formación de UAV's por la componente de R_{dist}. El producto punto entre los vectores $\mathrm{w}_{\boldsymbol{dist}} \cdot \boldsymbol{r}_{\boldsymbol{dist}} \in [0, 1]$ genera un escalar que representa el porcentaje de la recompensa máxima $r_{max,dist}$ que la formación de UAV's puede obtener por R_{dist} considerando los distintos componentes así como su nivel de importancia (e.g., si $\mathrm{w}_{\boldsymbol{dist}} = [0.5, 0.2, 0.3]$ significa que el componente que incentiva a la formación a acercarse a la carga útil lo más rápido posible ($R_{dist,min}$) es la de mayor importancia y atribución respecto a la recompensa R_{dist}). La siguiente ecuación describe la alternativa a la Ec. (4.9):

$$R_{dist} = r_{max,dist}(\mathrm{w}_{\boldsymbol{dist}} \cdot \boldsymbol{r}_{\boldsymbol{dist}}), \tag{4.12}$$

donde $r_{max,dist}$ y $\mathrm{w}_{\boldsymbol{dist}}$ son hiperparámetros del modelo, mientras que $\boldsymbol{r}_{\boldsymbol{dist}}$ es calculado en función de cada uno de sus componentes que se irán describiendo en las siguientes secciones.

Los componentes descritos en la Ec. (4.9) tienen varios hiperparámetros escalares $\in \mathbb{R}^+$ asociados para definir un valor de recompensa, pero como se mencionó existe otros componentes aparte de R_{dist} que también tienen sus propios hiperparámetros $\in \mathbb{R}$, con un total de 17 hiperparámetros (Tabla 4.1). Para cada uno es necesario definir un valor escalar, lo cual es un reto dado que al no estar acotado su valor pueden tomar una infinidad de valores haciendo la búsqueda de los mejores hiperparámetros asociados a la recompensa todo un reto.

La ventaja utilizar la Ec. (4.12) está en que en lugar de definir un valor escalar $\in \mathbb{R}$ para cada hiperparámetro ahora se acotan dichos hiperparámetros en $\in [0, 1] \in \mathbb{R}$, teniendo ahora una interpretación de los hiperparámetros como un porcentaje o pesos, siendo más sencillo asignar dichos valores. Adicionalmente, permite definir un único escalar que representa la recompensa máxima $r_{max,dist}$ por el componente R_{dist}, y un vector de pesos $\mathrm{w}_{\boldsymbol{dist}}$ que contiene el nivel de importancia de cada componente, siendo algo más intuitivo a la hora de definir los valores. A continuación se describirán el cálculo e intuición de cada uno de los componentes mencionados en la Ec. (4.10).

Distancia mínima

El componente $R_{dist,min}$ tiene como objetivo incentivar a la formación a acercarse lo más rápido posible a la carga útil, definiendo como métrica la distancia euclidiana entre la formación de UAV's y la carga útil en el plano XY. Para ello, se definieron un par de zonas con el objetivo de incentivar distintos comportamientos según la zona, una zona de rescate que está definida como una geometría alrededor de la carga útil (e.g., un círculo de 3 metros de radio, donde la carga útil es el centro del círculo) y la zona de búsqueda que es el espacio restante.

La zona de búsqueda procura incentivar que la formación se acerque a la zona de rescate lo más rápido posible, para ello se define una distancia euclidiana mínima global $d_{min} \in \mathbb{R}^+$ que se va actualizando cada vez que se logra una mejor distancia

mínima global $d_{min} \leftarrow d_t$ si $d_t < d_{min}$, buscando así atraer la formación a la carga útil. Cada vez que la formación de UAV's mejora la distancia mínima global d_{min} y está en la zona de búsqueda, obtiene una recompensa definida por el hiperparámetro $r_{d,min} \in \mathbb{R}^+$.

Adicionalmente, se busca incentivar que la formación se acerque lo más rápido posible a la zona de rescate, por lo que se agregó un componente que calcula la diferencia en la que se mejora la distancia mínima global, tal que:

$$\delta = d_{min} - d_t, \tag{4.13}$$

donde d_t es la distancia euclidiana entre la formación de UAV's y la carga útil en el paso de tiempo actual t. Este componente busca darle mayor importancia a una mejora en diferencias más grandes (i.e., es mejor acercarse a ritmos de 5 m/s que 1 m/s). Para ello se define una diferencia máxima global δ_{max} que se va actualizando cada vez que se logra una mejor diferencia máxima global $\delta_{max} \leftarrow \delta_t$ si $\delta_t > \delta_{max}$, donde δ_t es la diferencia calculada en el paso de tiempo actual t. En caso de que se mejore d_{min} pero no δ_{max} se da una recompensa proporcional δ_t/δ_{max}, dicho de otra forma, si se mejora la diferencia máxima global se obtiene toda la recompensa asociada con este componente, en caso contrario se obtiene una recompensa proporcional según el valor de δ_{max}. La recompensa asociada a dicho incentivo está definida por el hiperparámetro $r_{\delta,max} \in \mathbb{R}^+$.

Asimismo, para incentivar un poco más la aproximación de los UAV's a la zona de rescate, se agrega un término que otorga una recompensa proporcional a la cercanía de la zona de rescate, de acuerdo con la relación de la distancia asociada a la zona de rescate d_{zona} y la distancia obtenida d_t, es decir d_{zona}/d_t. La recompensa asociada a dicho incentivo está definida por el hiperparámetro $r_{prox} \in \mathbb{R}^+$. La función de recompensas en la zona de búsqueda considerando los incentivos mencionados se expresa como:

$$R_{dist,busqueda} = r_{d,min} + \frac{\delta_t}{\delta_{max}} r_{\delta,max} + \frac{d_{zona}}{d_t} r_{prox} \quad \text{si} \quad d_t < d_{min}; d_t > d_{zona}, \tag{4.14}$$

donde $r_{d,min}, r_{\delta,max}, r_{prox}, d_{zona}$ son hiperparámetros del modelo. Los primeros tres están asociados a los valores de recompensas que se obtienen al mejorar la distancia mínima global, mejorar la diferencia máxima y acercarse a la zona de rescate respectivamente. Mientras que la última es la distancia a partir de la carga útil que se forma la zona de rescate, estando en un rango de $(0, r_{max}]$ metros.

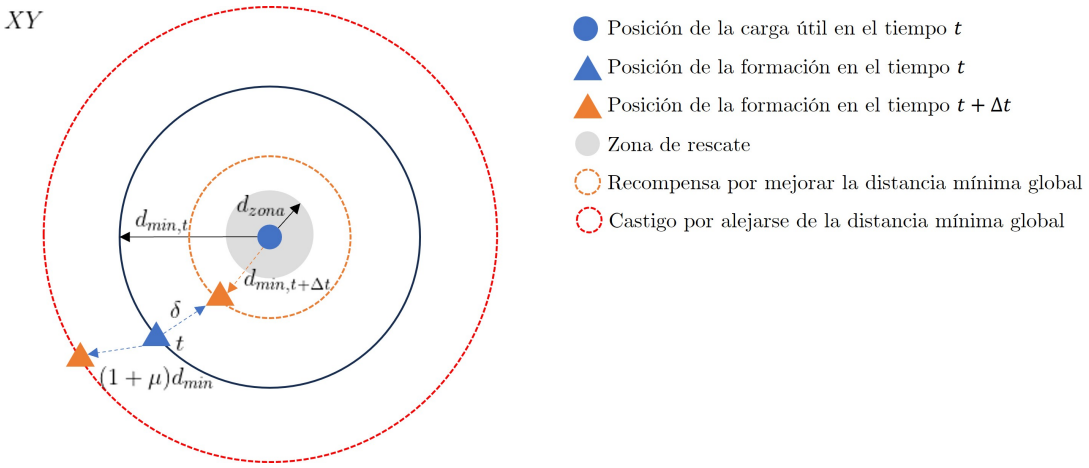


Figura 4.3: Esquema de las recompensas y castigos en el componente de distancia.

La zona de rescate está definida por una geometría de cierta longitud d_{zona} que rodea a la carga útil, en dicha zona se procura que los UAV's se acerquen lo más posible pero ya no se incentiva que sean con las diferencias δ_t más altas, esto es porque el enfoque ahora es tratar de realizar la captura lo más precisa posible. Dicho de otra forma, se mantiene la recompensa por mejorar la distancia mínima global $r_{d,min}$ pero ya no se considera la recompensa asociada a mejorar δ_{max}. Para no eliminar las recompensas asociadas por mejorar δ_{max} y acercarse a la zona de rescate r_{prox} de la Ec. (4.14), se agregó una recompensa por estar en la zona de rescate r_{zona} que está conformada por ambas recompensas, es decir, $r_{zona} = r_{\delta,max} + r_{prox}$. Por lo tanto, la función de recompensas en la zona de rescate se expresa como:

$$R_{dist,rescate} = r_{d,min} + r_{zona} \quad \text{si} \quad d_t \leq d_{min}; d_t \leq d_{zona}. \tag{4.15}$$

Adicionalmente, se agrega un término que castiga a la formación con un valor definido por el hiperparámetro $r_{alejar} \in \mathbb{R}^-$ por alejarse del mejor mínimo global de la distancia d_{min} en una cierta proporción definida por $(1+\mu)$, incentivando a los UAV's a mejorar esa distancia, que en una forma de pensarlo en es incentivar a que aprendan a seguir la trayectoria de la carga útil, tal que:

$$R_{dist,alejar} = -r_{alejar} \quad \text{si} \quad d_t \geq (1+\mu)d_{min}, \tag{4.16}$$

donde r_{alejar} y $\mu \in [0,1] \subset \mathbb{R}$ son hiperparámetros del modelo, siendo el primero el valor con el que se castiga a la formación de UAV's por alejarse y el segundo determina a partir de que proporción de d_{min} se aplica el castigo a los UAV's por alejarse. Por lo tanto, la expresión general del componente de la recompensa por distancia R_{dist} queda expresada en función de sus partes como:

$$R_{dist,min} = \begin{cases} r_{d,min} + \dfrac{\delta_t}{\delta_{max}} r_{\delta,max} + \dfrac{d_{zona}}{d_t} r_{prox} & \text{si} \quad d_t < d_{min}; d_t > d_{zona}, \\ r_{d,min} + r_{zona} & \text{si} \quad d_t \leq d_{min}; d_t \leq d_{zona}, \\ -r_{alejar} & \text{si} \quad d_t \geq (1+\mu)d_{min}. \end{cases} \tag{4.17}$$

La Ec. (4.12) restringe a que el componente $R_{dist,min} \in [0,1] \subset \mathbb{R}^+$, por lo tanto se tienen las siguientes restricciones sobre los hiperparámetros asociados a los distintos incentivos:

$$0 \leq r_{d,min} + r_{\delta,max} + r_{prox} \leq 1,$$
$$r_{d,min}, r_{\delta,max}, r_{prox}, r_{alejar} \in [0,1].$$

Con estas restricciones se definen los rangos permitidos de los hiperparámetros asociados a los incentivos, permitiendo darles una interpretación de importancia a cada uno según su ponderación, e.g., si $r_{d,min} = 0.7$ significa que en el componente de $R_{dist,min}$ es más importante aproximarse a la carga útil porque cada vez que se mejora la mínima distancia global d_{min} se obtiene el 70 % de la recompensa asociada a $R_{dist,min}$. Este factor es multiplicado por el producto punto de la Ec. (4.12) por su respectivo peso definido por el vector $\mathrm{w}_{\boldsymbol{dist}}$ de la Ec. (4.11), obteniendo así el porcentaje total otorgado a la recompensa por distancia R_{dist} por acercarse a la carga útil, pero si no se acerca este componente tendrá un valor de 0, el cual no sumará a la recompensa general. Esta intuición es la que está detrás de la forma alternativa descrita en la Ec. (4.12), en lugar de tener a los hiperparámetros $\in \mathbb{R}$, se acotan para que estén en $\in [0,1] \subset \mathbb{R}^+$ interpretándolos como ponderaciones de cada uno de los incentivos en las distintas señales de recompensas.

Altitud

Las recompensas asociadas a la distancia mínima $R_{dist,min}$ están calculadas para el plano XY, pero también es necesario aproximarse en altitud a la carga útil para realizar la captura lo más pronto posible en el rango permitido, es por eso que también se generaron recompensas asociadas a la altitud.

El primer componente busca incentivar a los UAV's a mejorar la diferencia de altitud entre la carga útil y la formación ($\Delta h_t = h_{carga} - h_{UAV's}$) respecto al paso de tiempo anterior (Δh_{t-1}), obteniendo una recompensa definida por el hiperparámetro $r_{h,min} \in \mathbb{R}^+$ cuando $\Delta h_t \leq \Delta h_{t-1}$, i.e., cuando se disminuye la diferencia de altitudes respecto al paso de tiempo anterior.

Adicionalmente, se busca incentivar a la formación por estar siempre por debajo de la carga útil, dado que si se encuentra por arriba nunca podrán capturar la carga útil, por lo que, la formación de UAV's obtiene una recompensa de valor definido por el hiperparámetro $r_{h,pos} \in \mathbb{R}^+$ cuando se encuentra por debajo de la carga útil ($\Delta h_t \geq 0$) y un castigo $-r_{h,pos}$ cuando se encuentra por arriba de la carga útil ($\Delta h_t < 0$). Por lo tanto, la recompensa que se obtiene por el componente de altitud $R_{altitud}$ queda expresada como:

$$R_{altitud} = \begin{cases} r_{h,min} + sgn\left(\Delta h_t\right) r_{h,pos} & \text{si} \quad \Delta h_t \leq \Delta h_{t-1}, \\ sgn\left(\Delta h_t\right) r_{h,pos} & \text{si} \quad \Delta h_t > \Delta h_{t-1}, \end{cases} \tag{4.18}$$

donde la función signo (sgn) indica si se obtiene una recompensa en castigo según si la formación se encuentra por debajo o arriba de la carga útil. La Ec. (4.12) restringe

a que el componente $R_{altitud} \in [0,1] \subset \mathbb{R}^+$, por lo tanto se tienen las siguientes restricciones sobre los hiperparámetros asociados a los distintos incentivos:

$$0 \leq r_{h,min} + r_{h,pos} \leq 1,$$
$$r_{h,min}, r_{h,pos} \in [0,1].$$

Ángulos respecto a Z

Las recompensas asociadas a los ángulos respecto a Z $(\theta_{XZ}, \theta_{YZ})$ buscan incentivar a los UAV's a dirigirse a la carga útil y estar por debajo de la misma, siendo una forma de reforzar los dos componentes anteriores mencionados en las Ecs. (4.17, 4.18). En la Fig. 4.4 se muestra un esquema en un plano (e.g., XZ) donde se ilustra la posición de los UAV's y distintas posiciones posibles de la carga útil, cada posición posible tendrá un ángulo θ asociado, y a partir de éste se puede generar una recompensa según su valor.

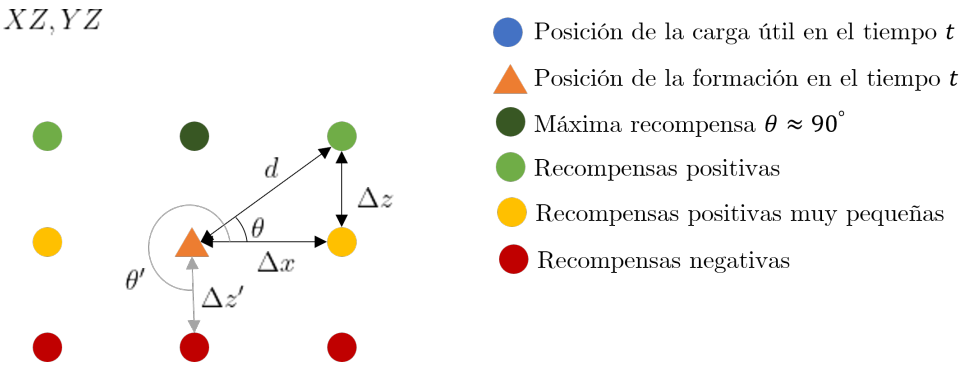


Figura 4.4: Esquema de las recompensas y castigos según el valor del ángulo θ.

El cálculo del ángulo θ se realiza para cada plano (XZ, YZ) con la función seno inversa, esto para que el valor de θ sea independiente a la posición en x o y, y solamente sea dependiente a la altitud Δz, y además esté acotado en los valores de $\in [0, 90]^\circ \subset \mathbb{R}^+$, tal que:

$$\theta = \sin^{-1}\left(\frac{\Delta z}{d + \epsilon}\right), \tag{4.19}$$

donde d es la distancia euclidiana entre la carga útil y los UAV's según el plano que se esté calculando y ϵ es un número muy pequeño que evita problemas con la división entre cero. Una vez calculado el ángulo θ, se genera una función de recompensa asociado a θ, tal que valores de $\theta < 0^\circ$ implican que la formación está por arriba de la carga útil, siendo un comportamiento no deseado porque no se podría hacer la captura, por lo que se castiga a la formación de UAV's.

Los valores de $0^\circ \leq \theta \leq 90^\circ$ generan recompensas positivas, pero en los valores cercanos a 0° implica que la formación está a la misma altitud $(\Delta z \approx 0)$, lo cual es bueno o malo según la posición de la formación respecto a la carga útil, pero en este componente de la recompensa por distancia R_{dist} lo que se busca es acercarse lo más

rápido posible a la zona de rescate, por lo tanto un ángulo $\theta \approx 0°$ no es deseado, porque de esa forma no se podría hacer la captura, y por lo tanto recibiría muy poca recompensa. En caso contrario, se busca que $\theta \approx 90°$ porque eso implicaría que la formación esté por debajo de la carga útil sobre la misma línea, siendo una especie de sombra, por lo tanto, obtendría una mayor recompensa al estar θ cerca de los valores de 90°. Para obtener esa recompensa se realiza una función paramétrica, donde el primer componente castigo por ángulos negativos, y el segundo componente genera una fracción lineal de la recompensa según el valor el θ.

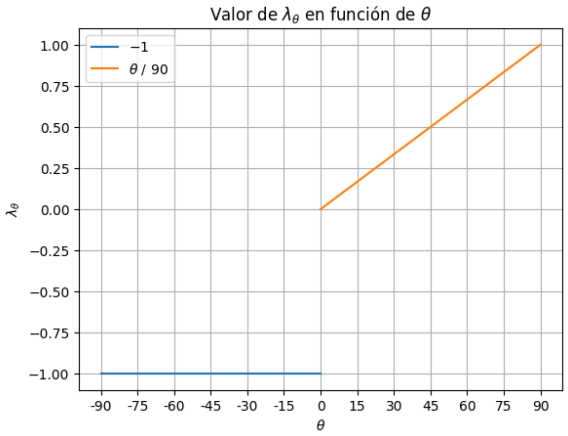


Figura 4.5: Valor de λ_θ en función de θ.

$$\lambda_\theta = \begin{cases} -1 & \text{si} \quad \theta < 0, \\ \dfrac{\theta}{90} & \text{si} \quad 0 \leq \theta \leq 90, \end{cases} \tag{4.20}$$

donde $\theta/90 \in [0, 1] \subset \mathbb{R}^+$, obteniendo así la máxima recompensa cuando $\theta = 90°$. Por lo tanto, la recompensa obtenida para cada plano se calcula como:

$$R_{\theta,XZ} = \lambda_{\theta,XZ} \cdot r_\theta, \tag{4.21}$$

$$R_{\theta,YZ} = \lambda_{\theta,YZ} \cdot r_\theta, \tag{4.22}$$

donde $r_\theta \in \mathbb{R}^+$ es un hiperparámetro del modelo y representa la recompensa que se obtiene según el valor de θ. La recompensa R_θ está compuesta por la suma de la recompensa asociada por los ángulos θ obtenidos para cada plano XZ y YZ, tal que:

$$R_\theta = R_{\theta_{XZ}} + R_{\theta_{YZ}} \tag{4.23}$$

La Ec. (4.12) restringe a que el componente $R_\theta \in [0, 1] \subset \mathbb{R}^+$, por lo tanto se tiene la restricción de que $r_\theta \in [0, 1]$, pero dado que r_θ se encuentra en ambos componentes de la suma la restricción es entonces $r_\theta \in [0, 0.5] \subset \mathbb{R}^+$.

4.3.2. Recompensas según la zona

Las recompensas según la zona buscan incentivar que los UAV's se acerquen a la carga útil por medio de recompensas intermedias que van incrementando por llegar

a ciertas zonas definidas por un vector de distancias $\boldsymbol{d_{zonas}} \in \mathbb{R}^n$, donde n es el número de zonas definidas. En la Fig. 4.6 se muestra un esquema donde cada círculo representa una zona y mientras más cerca esté la formación de UAV's de la carga útil obtendrá una recompensa asociada a la zona en la que se encuentre.

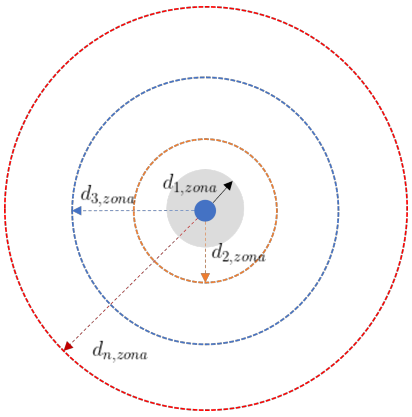


Figura 4.6: Esquema de las distintas zonas definidas en el vector $\boldsymbol{d_{zonas}}$.

Para calcular las recompensas asociadas por zona es necesario definir cuatro hiperparámetros. El primero es un escalar $r_{max,zona} \in \mathbb{R}^+$ que define el valor máximo de la recompensa que puede obtener la formación de UAV's respecto al componente R_{zona}, el segundo es un vector de distancias $\boldsymbol{d_{zonas}} \in \mathbb{R}^{+n}$ que contiene las distancias que definen cada zona tomando como centro la posición de la carga útil, el tercero es un vector de pesos $\mathrm{w}_{\boldsymbol{zonas}} \in \mathbb{R}^{+n}$ que contiene el porcentaje de la recompensa $r_{max,zona}$ que se obtiene por zona tal que:

$$\mathrm{w}_{\boldsymbol{zonas}} = [w_1, w_2, ..., w_n] \in \mathbb{R}^n : 0 \leq w_i \leq 1 \text{ para } i = 1, 2, ..., n. \tag{4.24}$$

Por último, el hiperparámetro $dim \in \{2, 3\}$ hace referencia al número de dimensiones en el que se realizará el cálculo. Si $dim = 2$ significa que el cálculo de las distancias se hará sobre el plano XY con círculos de radio $\boldsymbol{d_{zonas}}$, mientras que si $dim = 3$ significa que el cálculo de las distancias se hará con medias esferas en la parte de abajo con centro en la carga útil, esto es porque no se busca incentivar que la formación se encuentre por arriba de la carga útil.

El valor de la recompensa según la zona se calcula con la siguiente ecuación:

$$R_{zona} = \lambda_{zona} r_{max,zona}, \tag{4.25}$$

donde λ_{zona} es el porcentaje de la recompensa $r_{max,zona}$ que corresponde al i-ésimo valor de $\mathrm{w}_{\boldsymbol{zonas}}$ cuando la formación de UAV's se encuentra en la i-ésima zona definida por la distancia respecto a la carga útil en $\boldsymbol{d_{zonas}}$, en caso de que la formación no esté en alguna de las zonas $\lambda_{zona} = 0$.

En la Fig. 4.6 se muestra un esquema donde cada círculo representa una zona definida por la distancia euclidiana de los componentes de $\boldsymbol{d_{zonas}}$, y según el círculo donde se encuentre la formación de UAV's obtendrá un porcentaje λ_{zona} de la recompensa máxima $r_{max,zona}$, dicho porcentaje está definido en el vector de pesos $\mathrm{w}_{\boldsymbol{zonas}}$. Por ejemplo, si $\boldsymbol{d_{zonas}} = [5, 10, 15, 20]$, $\mathrm{w}_{\boldsymbol{zonas}} = [1, 0.8, 0.6, 0.4]$ y la formación de UAV's se encuentra a 6 m de la carga útil, entonces $\lambda_{zona} = 0.8$, es decir, la formación de UAV's obtendrá el 80 % de la recompensa $r_{max,zona}$.

Asimismo, $r_{max,zona}$ es un hiperparámetro del modelo asociado a la recompensa máxima que los UAV's obtienen por estar dentro de la zona de rescate. La Ec. (4.12) restringe a que el componente $R_{zona} \in [0, 1] \subset \mathbb{R}^+$, por lo tanto se tiene la restricción de que $r_{max,zona} \in [0, 1] \subset \mathbb{R}^+$.

4.3.3. Recompensas según el rango en el marco legal

El componente de rango de captura (R_{rango}) busca incentivar a los UAV's a realizar la captura en un rango legal definido por las regulaciones para los UAV's en México, siendo un radio máximo $r_{max} = 457$ m del piloto y una altura máxima $h_{max} = 122$ m. La señal de recompensa de éste componente está compuesta por las recompensas de realizar la captura en el radio permitido ($R_{rango,horizontal}$) y en la altura permitida ($R_{rango,vertical}$), tal que:

$$R_{rango} = R_{rango,horizontal} + R_{rango,vertical}. \tag{4.26}$$

De manera análoga al componente de la recompensa por distancia R_{dist} descrita en la Ec. (4.9), la Ec. (4.26) puede escribirse en su forma alternativa. Para ello, se define el vector $\boldsymbol{r_{rango}}$ que está compuesto por los distintos componentes de R_{rango} con la restricción de que cada componente tiene que estar en un rango $\in [0, 1] \subset \mathbb{R}^+$ tal que:

$$\boldsymbol{r_{rango}} = [R_{rango,horizontal}, R_{rango,vertical}] \in \mathbb{R}^2 : 0 \leq R_{rango,horizontal}, R_{rango,vertical} \leq 1. \tag{4.27}$$

Asimismo, se define un vector de pesos $\mathrm{w}_{\boldsymbol{rango}}$ asociados a cada componente de $\boldsymbol{r_{rango}}$, ponderándolos según su nivel de importancia, con las restricciones de que cada componente tiene que estar en un rango $\in [0, 1] \subset \mathbb{R}^+$ y que la suma de sus componentes tiene que ser igual a 1, tal que:

$$\mathrm{w}_{\boldsymbol{rango}} = [w_1, w_2] \in \mathbb{R}^2 : 0 \leq w_i \leq 1 \text{ para } i = 1, 2; \sum_{i=1}^{2} w_i = 1. \tag{4.28}$$

Por último, se define un escalar $r_{max,rango} \in \mathbb{R}^+$ que es la recompensa máxima que puede recibir la formación de UAV's por el componente asociado al rango de captura R_{rango}. El producto punto entre los vectores $\mathrm{w}_{\boldsymbol{rango}} \cdot \boldsymbol{r_{rango}} \in [0, 1] \subset \mathbb{R}^+$ genera un escalar que representa el porcentaje de la recompensa máxima $r_{max,rango}$ que la formación de UAV's puede obtener por R_{rango} considerando los distintos componentes

así como su nivel de importancia. La siguiente ecuación describe la alternativa a la Ec. (4.26):

$$R_{rango} = r_{max,rango}(\mathrm{w}_{\boldsymbol{rango}} \cdot \boldsymbol{r}_{\boldsymbol{rango}}), \tag{4.29}$$

donde $r_{max,rango}$ y $\mathrm{w}_{\boldsymbol{rango}}$ son hiperparámetros del modelo, mientras que $\boldsymbol{r}_{\boldsymbol{rango}}$ es calculado en función de cada uno de sus componentes que se irán describiendo en las siguientes secciones.

Rango horizontal

Para calcular el componente de la recompensa por realizar la captura en un cierto rango horizontal, siendo el centro el punto de despegue $(0, 0, 0)$, se definen un par de hiperparámetros. El primero es un vector $\mathrm{w}_{\boldsymbol{rango,horizontal}}$ que contiene el porcentaje de la recompensa $r_{rango,horizontal} \in \mathbb{R}^+$ que obtendrán los UAV's según el rango horizontal donde realicen la captura, definido por los componentes $\mathrm{w}_{\boldsymbol{rango,horizontal}} = \{w_{r,dentro} \in [0,1] \subset \mathbb{R}^+, w_{r,fuera} \in [0,1] \subset \mathbb{R}^+\}$ donde $w_{r,dentro}$ es la proporción asociada a la recompensa si se realiza la captura dentro del límite horizontal establecido por regulación r_{max}, mientras que el componente $w_{r,fuera}$ es la proporción asociada a la recompensa si se realiza la captura fuera del rango permitido, tal que:

$$R_{rango,horizontal} = r_{rango,horizonal} \cdot \begin{cases} w_{r,dentro} & \text{si} \quad r_{captura} \leq r_{max}, \\ w_{r,fuera} & \text{si} \quad r_{captura} > r_{max}, \end{cases} \tag{4.30}$$

donde $r_{rango,horizonal}$ es un hiperparámetro del modelo y es el valor de la recompensa que se obtiene por realizar la captura en el rango horizontal (radio) permitido por la regulación delimitado por r_{max} con centro donde se realizó el despegue de los UAV's, y $r_{captura}$ es el radio en el plano XY donde se realizó la captura, siendo el centro el punto del despegue. La Ec. (4.29) restringe a que el componente $R_{rango,horizontal} \in [0,1] \subset \mathbb{R}^+$, por lo tanto se tiene la restricción de que $r_{rango,horizonal} \in [0,1] \subset \mathbb{R}^+$.

Rango vertical

Para calcular el componente de la recompensa por realizar la captura en un cierto rango de altitud, se definen un par de hiperparámetros. El primero es un vector $\mathrm{w}_{\boldsymbol{rango,vertical}}$ que contiene el porcentaje de la recompensa $r_{rango,vertical} \in \mathbb{R}^+$ que obtendrán los UAV's según el rango de altitud donde realicen la captura, definido por los componentes $\mathrm{w}_{\boldsymbol{rango,vertical}} = \{w_{\mathrm{v},arriba} \in [0,1] \subset \mathbb{R}^+, w_{\mathrm{v},dentro} \in [0,1] \subset \mathbb{R}^+, w_{\mathrm{v},abajo} \in [0,1] \subset \mathbb{R}^+\}$ donde $w_{\mathrm{v},arriba}$ es la proporción de la recompensa si se realiza la captura por arriba del límite establecido por la regulación h_{max}, $w_{\mathrm{v},abajo}$ es la proporción de la recompensa por realizar la captura por debajo de una altura mínima h_{min} definida como hiperparámetro del modelo, mientras que $w_{\mathrm{v},dentro}$ es la proporción de la recompensa si se realiza la captura en el rango $[h_{min}, h_{max}]$, tal que:

$$R_{rango,vertical} = r_{rango,vertical} \cdot \begin{cases} w_{\mathrm{v},arriba} & \text{si} \quad h_{captura} > h_{max}, \\ w_{\mathrm{v},dentro} & \text{si} \quad h_{min} \leq h_{captura} \leq h_{max}, \\ w_{\mathrm{v},abajo} & \text{si} \quad h_{captura} < h_{min}, \end{cases} \tag{4.31}$$

donde $r_{rango,vertical}$ es un hiperparámetro del modelo y es el valor de la recompensa que se obtiene por realizar la captura en una región de altitud definida por una altura máxima regulada h_{max} y un límite mínimo definido como hiperparámetro del modelo, $h_{captura}$ es la altitud en la que se realizó la captura. La Ec. (4.29) restringe a que el componente $R_{rango,vertical} \in [0,1] \subset \mathbb{R}^+$, por lo tanto se tiene la restricción de que $r_{rango,vertical} \in [0,1] \subset \mathbb{R}^+$.

4.3.4. Recompensas por realizar la captura

En esta sección se aborda el componente de la recompensa por realizar la captura, cuyo objetivo es guiar a la formación a realizar la captura de la carga útil lo más centrado posible en la red lo más suave posible (i.e., menor velocidad) dentro del rango horizontal y vertical permitido legalmente. Los componentes de la recompensa por realizar la captura son:

- Una recompensa por realizar con éxito la captura ($R_{exito} \in \mathbb{R}^+$).
- Una recompensa por realizar la captura lo más centrado posible en la red ($R_{zona,red} \in \mathbb{R}^+$).
- Una recompensa por realizar la captura con la menor velocidad de los UAV's posible, siendo un incentivo para que sea una captura suave ($R_{vel} \in \mathbb{R}^+$).

Por lo tanto, la señal de recompensas por realizar la captura está dada por la suma de sus distintos componentes:

$$R_{captura} = R_{exito} + R_{zona,red} + R_{vel}. \tag{4.32}$$

De manera análoga al componente de la recompensa por distancia R_{dist} descrita en la Ec. (4.9), la Ec. (4.32) puede escribirse en su forma alternativa. Para ello, se define el vector $\boldsymbol{r_{captura}}$ que está compuesto por los distintos componentes de $R_{captura}$ con la restricción de que cada componente tiene que estar en un rango $\in [0,1]$ tal que:

$$\boldsymbol{r_{captura}} = [R_{exito}, R_{zona,red}, R_{vel}] \in \mathbb{R}^3 : 0 \leq R_{exito}, R_{zona,red}, R_{vel} \leq 1. \tag{4.33}$$

Asimismo, se define un vector de pesos $\mathrm{w}_{\boldsymbol{captura}}$ asociados a cada componente de $\boldsymbol{r_{captura}}$, ponderándolos según su nivel de importancia, con las restricciones de que cada componente tiene que estar en un rango $\in [0,1] \subset \mathbb{R}^+$ y que la suma de sus componentes tiene que ser igual a 1, tal que:

$$\mathrm{w}_{\boldsymbol{captura}} = [w_1, w_2, w_3] \in \mathbb{R}^3 : 0 \leq w_i \leq 1 \text{ para } i = 1,2,3; \sum_{i=1}^{3} w_i = 1. \tag{4.34}$$

Por último, se define un escalar $r_{max,captura} \in \mathbb{R}^+$ que es la recompensa máxima que puede recibir la formación de UAV's por el componente de realizar la captura $R_{captura}$. El producto punto entre los vectores $\mathrm{w}_{\boldsymbol{captura}} \cdot \boldsymbol{r_{captura}} \in [0,1] \subset \mathbb{R}^+$ genera un escalar

que representa el porcentaje de la recompensa máxima $r_{max,captura}$ que la formación de UAV's puede obtener por $R_{captura}$ considerando los distintos componentes así como su nivel de importancia. La siguiente ecuación describe la alternativa a la Ec. (4.32):

$$R_{captura} = r_{max,captura}(\mathrm{w}_{\boldsymbol{captura}} \cdot \boldsymbol{r}_{\boldsymbol{captura}}), \tag{4.35}$$

donde $r_{max,captura}$ y $\mathrm{w}_{\boldsymbol{captura}}$ son hiperparámetros del modelo, mientras que $\boldsymbol{r}_{\boldsymbol{captura}}$ es calculado en función de cada uno de sus componentes que se irán describiendo en las siguientes secciones.

Captura realizada con éxito

La principal y más importante recompensa que se obtiene es cuando se realiza con éxito la captura, es decir, los UAV's obtienen dicha recompensa en el momento en que la carga útil hace contacto o colisiona con la cara interna de la red, tal que:

$$R_{exito} = r_{exito} \text{ si } \exists \text{ colisión}_{\text{carga-red}}, \tag{4.36}$$

donde $r_{exito} \in \mathbb{R}^+$ es un hiperparámetro del modelo. La Ec. (4.35) restringe a que el componente $R_{exito} \in [0, 1] \subset \mathbb{R}^+$, por lo tanto se tiene la restricción de que $r_{exito} \in [0, 1] \subset \mathbb{R}^+$. El valor de $r_{exito} = 1$ implica que al realizar la captura se tiene toda la recompensa asociada a este componente, si $r_{exito} = 0.5$ entonces únicamente se obtendría la mitad.

Zona de la red

La recompensa asociada según la zona de la red busca incentivar a que se capture la carga útil lo más centrado posible a la red, evitando problemas de que se capture en la orilla y se tenga el riesgo de que se salga de la red y, por lo tanto, no se realice la captura de manera exitosa.

De manera similar al componente de recompensas según la zona R_{zona} se definen una serie de hiperparámetros. El primero es un escalar $r_{zona,red} \in \mathbb{R}^+$ que define el valor máximo de la recompensa que puede obtener la formación UAV's por capturar la carga útil lo más centrado posible de la red, el segundo es un vector con las distancias relativas al radio de rescate $r_{rescate}$ definido en la sección 3.2 que delimitan las distintas zonas $\boldsymbol{d}_{\boldsymbol{rel,rescate}} \in \mathbb{R}^n$ donde n es el número de distancias relativas definidas en el vector.

Por ejemplo, si $r_{rescate} = 4$ m y $\boldsymbol{d}_{\boldsymbol{rel,rescate}} = [0.5, 0.8, 1]$ significa que hay 3 zonas, la primera zona estará en el rango de la mitad del radio de rescate definido es decir entre $[0, 2]$, siendo la posición más centrada y por lo tanto la ideal. La segunda zona estará definida entre la mitad y el 80 % de la red, es decir entre $(2, 3.2]$, siendo una zona de riesgo moderado, y la última zona será el sobrante es decir entre $(3.2, 4]$ siendo la zona de mayor riesgo. En la Fig. 4.7 se muestra un esquema de las distintas zonas

descritas en función de $\boldsymbol{d_{rel,rescate}}$.

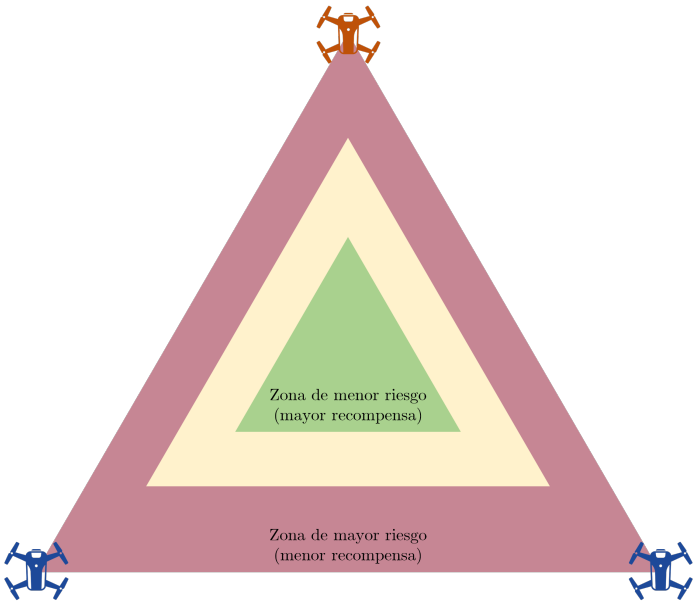


Figura 4.7: Esquema de las distintas zonas de riesgo en función de la proporción del radio de rescate.

El tercer hiperparámetro es un vector $\mathrm{w}_{\boldsymbol{zona,red}} \in \mathbb{R}^n$ que asocia el porcentaje de la recompensa máxima $r_{zona,red}$ que se puede obtener por la realizar la captura en una determinada zona, tal que:

$$\mathrm{w}_{\boldsymbol{zonas}} = [w_1, w_2, ..., w_n] \in \mathbb{R}^n : 0 \leq w_i \leq 1 \text{ para } i = 1, 2, ..., n. \tag{4.37}$$

Por ejemplo, si $\mathrm{w}_{\boldsymbol{zona,red}} = [1, 0.5, 0.1]$ significa que si se realiza la captura en la primera zona (ideal) se obtiene el 100 % de la recompensa, mientras que si la captura se realiza en la última zona (zona de alto riesgo) se obtiene el 10 % de la captura.

Esta recompensa se otorga cuando la carga útil cruza un plano XY triangular en el eje Z, que es la proyección de la red. Pero, para ser más laxos con la recompensa y agregar más recompensas intermedias se optó por una estrategia en la que se comienzan a generar estas recompensas cuando en la proyección de la carga útil se encuentra en cualquiera de las zonas, con la restricción de que se encuentre en una diferencia de altitud de la carga útil y los UAV's pequeña, generando un prisma triangular que se mueve con la formación de UAV's. Esto, con el objetivo de ayudar a los UAV's a centrar la red con la carga útil metros antes de realizar la captura en la etapa de rescate.

$$R_{zona,red} = \lambda_{zona,red} r_{zona,red} \quad \text{si} \quad -h_{zona,inf} \leq \Delta h_t \leq h_{zona,sup}, \tag{4.38}$$

donde $\lambda_{zona,red}$ es el valor del porcentaje de la recompensa asociada a la zona de la red en la que se encuentra la carga útil, en caso de que la carga útil se encuentre fuera

de las zonas definidas $\lambda_{zona,red} = 0$, y $h_{zona,inf}, h_{zona,sup} \in \mathbb{R}^+$ son hiperparámetros del modelo que define el umbral del diferencial de altitud entre la carga útil y los UAV's, teniendo la granularidad de definir el umbral inferior $h_{zona,inf}$ que es negativo porque la carga útil estará por debajo de los UAV's cuando se realice la captura, y un umbral superior $h_{zona,sup}$. La Ec. (4.35) restringe a que el componente $R_{zona,red} \in [0, 1] \subset \mathbb{R}^+$, por lo tanto se tiene la restricción de que $r_{zona,red} \in [0, 1] \subset \mathbb{R}^+$.

Velocidad de captura

El objetivo de este componente de la recompensa es incentivar a los UAV's a realizar la captura de la carga útil lo más suave posible, es decir, con la mínima velocidad posible. Para ello se propone una función para calcular un parámetro $\lambda_{vel} \in [0, 1] \subset \mathbb{R}^+$ que determina el porcentaje de la recompensa máxima $r_{vel,captura} \in \mathbb{R}^+$ que se puede obtener por este componente, siendo $r_{vel,captura}$ un hiperparámetro del modelo.

Dicha función busca incentivar a los UAV's a realizar la captura con la menor velocidad posible, es decir obtendrá la máxima recompensa cuando la magnitud de la velocidad de los UAV's sea muy cercana a 0, es decir, $\|\boldsymbol{v_{UAVs}}\| = v = 0$ y obtendrá la mínima recompensa cuando la velocidad de los UAV's sea igual a su velocidad máxima definida por $\langle v_{dron} \rangle = 5$ m/s, es decir $\|\boldsymbol{v_{UAVs}}\| = \langle v_{dron} \rangle = v_{max}$. En el presente trabajo se estableció una función lineal[7] para calcular λ_{vel}, tal que:

$$\lambda_{vel} = -\frac{v}{v_{max}} + 1, \tag{4.39}$$

donde v es la magnitud de la velocidad de los UAV's. La recompensa por la velocidad de captura es entonces una proporción (λ_{vel}) de la recompensa máxima por este componente ($r_{vel,captura}$), tal que:

$$R_{vel} = \lambda_{vel} r_{vel,captura}. \tag{4.40}$$

La Ec. (4.35) restringe a que el componente $R_{vel} \in [0, 1] \subset \mathbb{R}^+$, por lo tanto se tiene la restricción de que $r_{vel,captura} \in [0, 1] \subset \mathbb{R}^+$.

4.3.5. Penalizaciones

Los eventos que ocasionan se termine la simulación son:

- La colisión de algún UAV de la formación con cualquier otro agente (piso, carga útil, red) dentro del entorno de simulación.
- La colisión de la carga útil con el piso o algún UAV.
- El número de puntos de la trayectoria de la carga útil simulada se terminaron, i.e., no se realizó la captura de la carga útil.
- La captura exitosa de la carga útil, i.e., la carga útil hace contacto con la pared interna de la red.

[7] En futuros trabajos se puede explorar otras funciones, por ejemplo, funciones exponenciales.

Colisiones

Los primeros tres puntos se traducen en recompensas negativas o castigos, mientras que el último se abordó en la sección 4.3.4. Las colisiones se determinan extrayendo la información del entorno de simulación utilizando la API de *Unreal*, escribiendo en un archivo común las colisiones que se están generando en cada paso de tiempo, detectando así las colisiones que cada agente tiene, así como el agente con el que colisionó. Por lo tanto, el componente de la recompensa por colisión está dado por:

$$R_{colision} = R_{colision,UAV} + R_{colision,payload}, \tag{4.41}$$

donde $R_{colision,UAV}$ es el castigo que reciben los UAV's por colisionar con cualquier otro agente y $R_{colision,payload}$ es el castigo que se recibe si la carga útil tuvo colisión con el piso o con algún UAV, tal que:

$$\begin{aligned} R_{colision,UAV} &= r_{colision,UAV}, \\ R_{colision,payload} &= r_{colision,payload}, \end{aligned}$$

donde $r_{colision,UAV}, r_{colision,payload} \in [0, 1000] \subset \mathbb{R}^+$ son hiperparámetros[8] del modelo asociados a los castigos por colisión. Nótese que en la Ec. (4.8) se restan estas recompensas, por lo que los hiperparámetros tienen que ser positivos para lograr penalizar estos comportamientos.

Fin de la trayectoria de la carga útil

En la sección 3.1 se describe el proceso de cómo se incorpora la trayectoria de la carga útil en el entorno de simulación, en el cual se generan archivos CSV que contienen la información de la trayectoria de descenso de la carga útil simulada, donde cada fila equivale a un paso de tiempo en el entorno de simulación, con dicho archivo se puede determinar cuando finaliza la trayectoria, i.e., cuando ya se leyeron todos los puntos del archivo CSV y no se realizó la captura de la carga útil, cuando esto sucede se retorna una recompensa negativa asociada al hiperparámetro[8] $r_{fin,trayectoria} \in [0, 1000] \subset \mathbb{R}^+$ del modelo, tal que:

$$R_{fin} = r_{fin,trayectoria}. \tag{4.42}$$

Nótese que en la Ec. (4.8) se restan estas recompensas, por lo que los hiperparámetros tienen que ser positivos para lograr penalizar estos comportamientos.

Penalización por tiempo

Adicionalmente a todos los componentes descritos, se agrega un componente que penaliza a los UAV's por cada paso de tiempo que estén en el entorno de simulación,

[8]El rango propuesto en las distintas señales de recompensa es una primera iteración. La experimentación y ajuste en el valor de las distintas recompensas es una oportunidad para explorar en trabajos posteriores.

tratando de incentivar que realicen la captura de la carga útil lo más rápido posible, la magnitud de dicha penalización está asociada al hiperparámetro[8] $r_{tiempo} \in [0, 100] \subset \mathbb{R}^+$ del modelo, tal que:

$$R_{tiempo} = r_{tiempo}. \tag{4.43}$$

Nótese que en la Ec. (4.8) se restan estas recompensas, por lo que los hiperparámetros tienen que ser positivos para lograr penalizar estos comportamientos.

4.4. Proceso de entrenamiento

La selección de los algoritmos a utilizar en el presente trabajo están limitados a los algoritmos que se encuentran dentro de la paquetería *Stable Baselines3* en la versión que se está utilizando mostrados en la Fig. 4.1, adicionalmente se considera la capacidad que tienen para modelar el espacio de acciones, ya sea continuo (DDPG, SAC, A2C, HER, PPO), discreto o híbrido (A2C, HER, PPO). Por cuestiones en el tiempo de entrenamiento no fue posible iterar sobre cada uno de los algoritmos, por lo que se seleccionaron los algoritmos a probar según sus características y también basado en la literatura, identificando los algoritmos que son más utilizados para el entrenamiento de UAV's (DQN, PPO, SAC, A2C, A3C), así como sus resultados.

Los algoritmos seleccionados para la presente tesis fueron SAC y PPO. SAC para los espacios de acciones continuos dado que es un algoritmo de actor-crítico conocido por su eficacia en entornos con espacios de acciones continuos, utilizando métodos de optimización de la política y de la función de valor de manera conjunta, además incorpora una entropía máxima para incentivar la exploración, incentivando a que el agente aprenda todas las formas posibles de resolver la tarea.

El algoritmo PPO se entrenó para un espacio de acciones discreto y continuo. PPO ha demostrado ser eficaz en una variedad de entornos, con una capacidad de proporcionar un equilibrio robusto entre exploración y explotación. Además, tienen un enfoque en la estabilidad del aprendizaje, mitigando los riesgos asociados con actualizaciones de política demasiado agresivas, dado que se aseguran de que al actualizar la política nunca se aleje demasiado de la política anterior.

En el proceso de entrenamiento se dividió el banco de simulaciones donde se encuentran todos los archivos CSV con las distintas trayectorias de la carga útil en dos conjuntos, un conjunto de entrenamiento que es un directorio que contiene el 80 % de los archivos en el banco de simulaciones seleccionados aleatoriamente, y un conjunto de prueba que es un directorio que contiene el 20 % de los archivos restantes. El conjunto de prueba tiene el fin de probar la mejor política entrenada.

Para generar resultados significativos y poder compararlos, se realizaron 5 entrenamientos de los algoritmos SAC con el espacio de acciones continuos, PPO con el espacio de acciones discretos y PPO con el espacio de acciones continuos.

En la Tabla 4.1 se muestran los principales hiperparámetros utilizados para el entrenamiento, así como sus respectivos valores. Los hiperparámetros mostrados son los que se describieron a lo largo del presente trabajo, mientras que los hiperparámetros asociados a los algoritmos SAC y PPO son los que están por defecto en la biblioteca *Stable Baselines3*. Es importante mencionar que existe trabajo futuro en la exploración de estos valores para lograr obtener el comportamiento deseado.

Componente	**Subcomponente**	**Hiperparámetro**	**Valor**
Globales	Radio de rescate	$r_{rescate}$	4 m
	Velocidad máxima del UAV	v_{max}	5 m/s
	Rango horizontal	r_{max}	457 m
	Rango vertical	h_{max}	122 m
	FPS	FPS	30
Entrenamiento	Modelo	`model`	PPO/SAC
	Proximidad inicial	`move_to_proximity`	`teleport`
	Proximidad inicial	`proxy_vector`	[50,50,40]
	Espacio de acciones	`action_space_option`	`discrete_all/vel_continuos`
	Espacio de acciones	`discrete_vel`	5
	Número de pasos	`total_timesteps`	200,000
Recompensa R_{dist}	R_{dist}	$r_{max,dist}$	50
	R_{dist}	$\mathbf{w_{dist}}$	[0.5,0.2,0.3]
	R_{dist}	$r_{max,dist}$	50
	$R_{dist,min}$	$r_{d,min}$	0.7
	$R_{dist,min}$	$r_{\delta,max}$	0.2
	$R_{dist,min}$	r_{prox}	0.1
	$R_{dist,min}$	r_{zona}	0.3[1]
	$R_{dist,min}$	d_{zona}	5 m
	$R_{dist,min}$	r_{alejar}	-0.5
	$R_{dist,min}$	μ	5 %
	$R_{altitud}$	$r_{h,min}$	0.8
	$R_{altitud}$	$r_{h,pos}$	0.2
	R_{θ}	r_{θ}	0.5[2]
Recompensa R_{zona}	R_{zona}	$r_{max,zona}$	30
	R_{zona}	$\mathbf{d_{zonas}}$	[5,10,15,25,40] m
	R_{zona}	$\mathbf{w_{zonas}}$	[1,0.8,0.6,0.4,0.2]
Recompensa R_{rango}	R_{rango}	$r_{max,rango}$	10
	R_{rango}	$\mathbf{w_{rango}}$	[0.4,0.6]
	$R_{rango,horizontal}$	$r_{rango,horizontal}$	1
	$R_{rango,horizontal}$	$\mathbf{w_{rango,horizontal}}$	[1,0.5]
	$R_{rango,vertical}$	$r_{rango,vertical}$	1
	$R_{rango,vertical}$	$\mathbf{w_{rango,vertical}}$	[0.5,1,0.2]
	$R_{rango,vertical}$	h_{min}	5 m
Recompensa	$R_{captura}$	$r_{max,captura}$	10,000
	$R_{captura}$	$\mathbf{w_{captura}}$	[0.6,0.2,0.2]
	R_{exito}	r_{exito}	1
	$R_{zona,red}$	$r_{zona,red}$	1
	$R_{zona,red}$	$\mathbf{d_{rel,rescate}}$	[0.5,0.8,1]
	$R_{zona,red}$	$\mathbf{w_{zona,red}}$	[1,0.5,0.2]
	$R_{zona,red}$	$h_{zona,inf}$	3 m
	$R_{zona,red}$	$h_{zona,sup}$	3 m
	R_{vel}	$r_{vel,captura}$	1
Penalización R_{tiempo}	R_{tiempo}	r_{tiempo}	5*
Penalización puntos terminales	$R_{colision,UAV}$	$r_{colision,UAV}$	1,000*
	$R_{colision,payload}$	$r_{colision,payload}$	1,000*
	R_{fin}	$r_{fin,trayectoria}$	1,000*

Tabla 4.1: Valor de los distintos hiperparámetros utilizados en el entrenamiento. Los hiperparámetros fueron seleccionados de manera heurística.

[1] $r_{\delta,max} + r_{prox}$.

[2] Se calcula para el ángulo en el plano XZ y YZ.

* Los componentes por penalización en la Ec. (4.8) se encuentran restando, por lo tanto el valor de los hiperparámetros es positivo.

5 Resultados

En esta sección se analizarán los resultados obtenidos de tres escenarios distintos, y para cada escenario se realizaron 5 entrenamientos de 200,000 pasos para poder hacer la comparación entre los distintos escenarios. El primer escenario es el entrenamiento del algoritmo PPO con un espacio de acciones discreto dado por un conjunto de tres posibles acciones para cada componente de la dirección unitaria de la velocidad $\{-1, 0, 1\}$ y un factor $\lambda \in [0, 1]$ que discretiza un vector en intervalos que representan un porcentaje de la velocidad máxima de los UAV's v_{max}. El segundo y tercer experimento es el entrenamiento de los algoritmos PPO y SAC con un espacio continuo de acciones que está compuesto por un conjunto continuo de tres elementos, donde cada elemento define la dirección y proporción en un rango de [-1,1] de la velocidad máxima de los UAV's v_{max}. El espacio de acciones se describe con mayor detalle en la sección 4.1.

Los experimentos fueron realizados con un procesador i9-13900K y una tarjeta de video NVIDIA GeForce RTX 4090, con un tiempo de entrenamiento promedio por experimento para 200,000 pasos de 1.7 días para el entrenamiento de PPO con espacio de acciones discreto, 1.9 días para el entrenamiento de PPO con espacio de acciones continuas y 2.2 días para el entrenamiento de SAC con espacio de acciones continuas. Los resultados muestran que SAC tiene mejores resultados respecto a la recompensa promedio acumulada en comparación de PPO, con la desventaja de que es un poco más tardado el entrenamiento.

En la Fig. 5.1 se muestran los resultados para cada escenario respecto a la duración promedio de los episodios conforme avanzan los pasos de entrenamiento[1], en la que se observa una diferencia entre los distintos algoritmos en los primeros 50,000 pasos de entrenamiento, después se observa que los algoritmos van mejorando la longitud promedio de los episodios llegando a valores cercanos a una longitud de 500.

La interpretación de tener longitudes promedio de episodios más grande es que los UAV's están aprendiendo a no colisionar conforme se van entrenando, dado que cuando sucede una colisión el episodio termina. Por lo tanto, la formación de UAV's aprende más rápido a permanecer en el entorno sin colisionar utilizando el espacio de acciones continuo.

[1]Los pasos de entrenamiento es el número total de pasos ejecutados por la formación de UAV's para cualquier entorno.

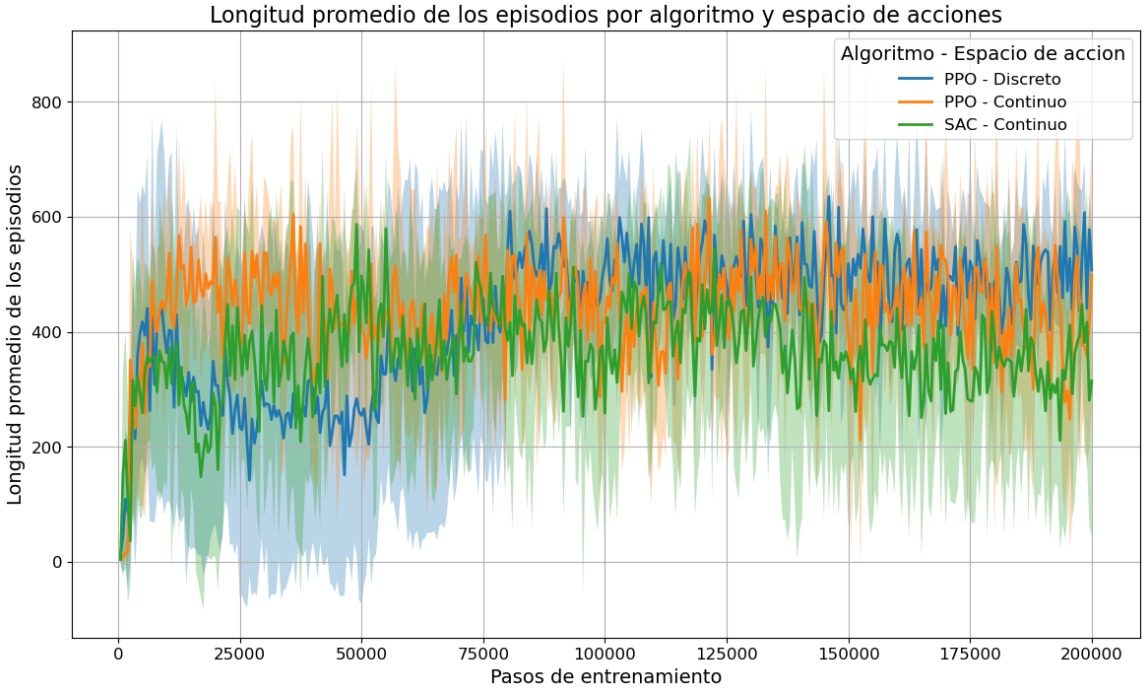


Figura 5.1: Longitud promedio de los episodios por pasos de entrenamiento durante el aprendizaje, para cada algoritmo y espacio de acciones.

En la Fig. 5.2 se muestra la recompensa acumulada promedio obtenido en función de los pasos de entrenamiento para cada escenario, en la que se observa que comparando el espacio de acciones para el algoritmo PPO se tiene un mejor desempeño al utilizar un espacio de acciones continuo, dado que a veces PPO puede tener dificultades para explorar eficientemente en espacio de acciones discretos. Además, se observa un comportamiento similar en ambos donde al principio del entrenamiento tienen relativamente altas recompensas, pero conforme avanza el entrenamiento las recompensas decrementan, inclusive a llegar a recompensas negativas (PPO con espacio de acciones discreto) o con valores cercanos a cero (PPO con espacio de acciones continuo).

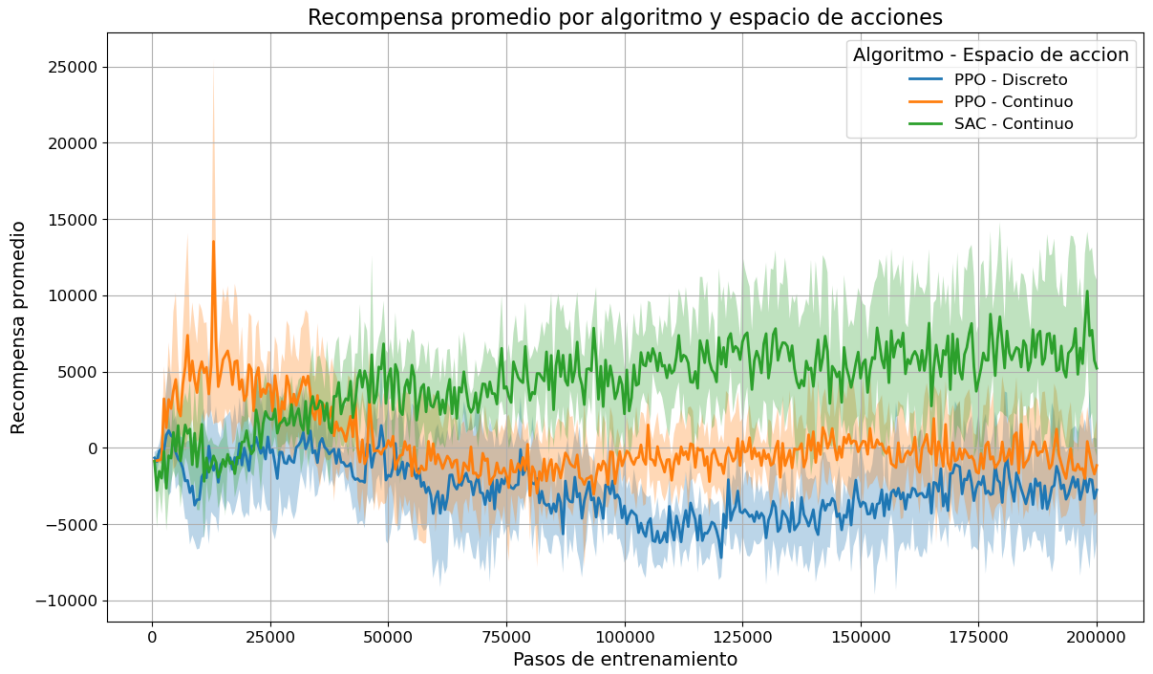


Figura 5.2: Recompensas acumuladas promedio por por pasos de entrenamiento durante el aprendizaje, para cada algoritmo y espacio de acciones.

Ahora bien, comparando los algoritmos de PPO y SAC con el espacio de acciones continuo se observa un mejor rendimiento del algoritmo SAC, adicionalmente, SAC va aprendiendo poco a poco, incrementando las recompensas promedio conforme se va entrenando la formación de UAV's, además se observa que conforme avanza el entrenamiento las bandas de confianza comienzan a abrirse, dando señales de que no está convergiendo, dado que hay experimentos donde obtuvo mucho más recompensas y otros en los que obtuvo mucho menores recompensas. Por lo que, mientras que SAC va mejorando sus acciones para maximizar la recompensa, en PPO se observa que, aunque después de un tiempo aprenden a no colisionar, no están consiguiendo las recompensas intermedias, sino que están recolectando recompensas negativas por alejarse de la carga útil.

Adicionalmente, se observa que las recompensas en SAC van incrementando y todavía no se observa una convergencia o un estado estable, indicando que posiblemente incrementar los pasos de entrenamiento puede mejorar el rendimiento del entrenamiento. El objetivo de capturar la carga útil en todos los experimentos para cada escenario no se cumple, es decir, la formación de UAV's no logró realizar la captura de la carga útil en ningún episodio, abriendo la oportunidad de iterar sobre distintos componentes como por ejemplo, incrementar el número de pasos de entrenamiento, las señales de recompensa, los hiperparámetros, los algoritmos, así como entender el comportamiento durante el entrenamiento para considerar mejoras futuras para lograr el objetivo.

Para validar si los distintos escenarios son estadísticamente diferentes se realizó una prueba Wilcoxon comparando cada uno de los escenarios. La prueba de Wilcoxon es una prueba no paramétrica[2] que se utiliza para comparar dos muestras relacionadas, comprobando si los valores medios de dos grupos dependientes (e.g., cuando los datos se obtienes a partir de mediciones repetidas) difieren significativamente entre sí. La hipótesis nula es que no existe diferencia (en términos de tendencia central) entre los dos grupos de la población, mientras que la hipótesis alternativa es que existe una diferencia entre los dos grupos de la población. En la Tabla 5.1 se muestra el p-valor comparando por pares cada uno de los escenarios, observándose que para cada comparación el p-valor < 0.05, por lo que se rechaza la hipótesis nula, i.e., tenemos suficiente evidencia para decir que existe una diferencia entre los distintos escenarios.

Tabla 5.1: Resultados p-valor de la prueba Wilcoxon comparando los distintos escenarios para definir si existe o no una diferencia entre los experimentos.

Comparación de escenarios	p-valor
PPO Discreto vs PPO Continuo	1.11e-146
PPO Discreto vs SAC Continuo	1.86e-290
PPO Continuo vs SAC Continuo	3.44e-153

Ahora bien, en la Tabla 5.2 se observa el valor promedio, la desviación estándar y la desviación estándar relativa de las recompensas acumuladas de los distintos experimentos de cada escenario en el último paso de entrenamiento, observando que el espacio de acciones continuo genera un mejor promedio en las recompensas promedio, y que SAC es el algoritmo con mejores resultados, pero tiene una alta incertidumbre, mostrando que aún no logra converger a alguna política, y que es necesario modificar el tiempo de entrenamiento, las señales de recompensa, los hiperparámetros y/o el POMDP.

Tabla 5.2: Recompensa acumulada promedio y desviación estándar de los experimentos en el último paso de entrenamiento (200,000) para cada algoritmo y espacio de acciones.

Algoritmo	Espacio de acciones	μ	σ	μ/σ
PPO	Discreto	-2,756.471	3,346.518	-0.824
PPO	Continuo	-1,152.355	3,025.394	-0.381
SAC	Continuo	5,219.572	5,773.804	0.904

En conclusión, los resultados indican un mejor rendimiento utilizando un espacio de acciones continuo y el algoritmo SAC, esta mejora podría atribuirse a la capacidad de SAC para manejar acciones continuas permitiendo movimientos más suaves, además tiene una estrategia de exploración eficiente que le permite explorar con mayor eficiencia el espacio de acciones por la maximización de la entropía, siendo una

[2]No es necesario que los datos se distribuyan normalmente.

ventaja en la captura de una carga útil en un espacio muy grande de búsqueda.

A continuación, se realiza un análisis más detallado sobre el comportamiento de las distintas señales de recompensa durante el entrenamiento del mejor experimento utilizando el algoritmo SAC con acciones continuas, tratando de generar información relevante que pueda ayudar a entender la influencia de cada componente, así como a formular mejoras futuras en próximos experimentos.

5.1. Análisis de recompensas

En la Fig. 5.3 se muestra el valor de la recompensa acumulada de los episodios, así como el promedio de la recompensa acumulada general R y de sus componentes por grupo de iteración, donde los grupos de iteración son la agrupación de los distintos episodios generados durante el aprendizaje, en este caso se dividieron 863 episodios en 50 grupos equitativamente y manteniendo la secuencia de entrenamiento, siendo el grupo 50 el que contiene los últimos episodios del entrenamiento.

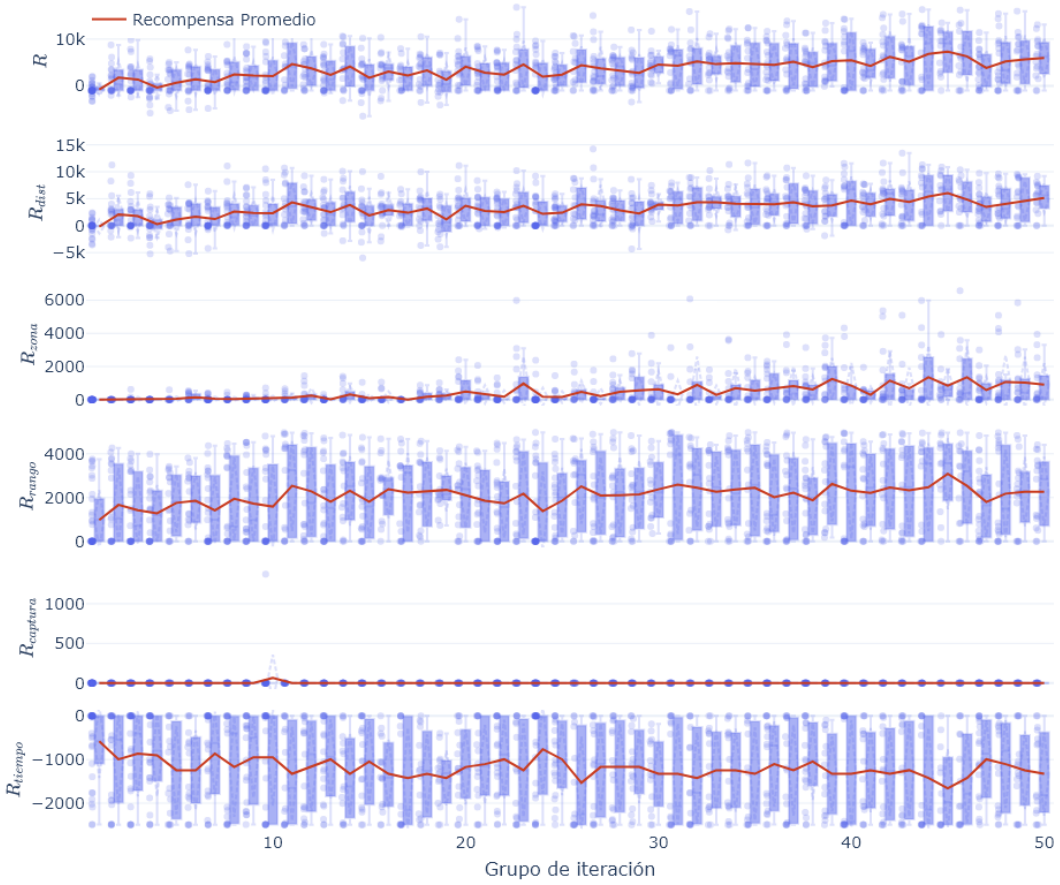


Figura 5.3: Recompensa acumulada de los episodios (*boxplot*) y el promedio de la recompensa acumulada general R (línea roja) y de sus componentes R_{dist}, R_{zona}, R_{rango}, $R_{captura}$, R_{tiempo} por grupo de iteración.

En la Fig. 5.3 se puede observar la dispersión en los valores de la recompensa acumulada en los episodios por agrupación, así como su evolución en el entrenamiento, observando que la tendencia de la recompensa general promedio va incrementado, así como la dispersión de los valores son cada vez mayores. También va disminuyendo la ocurrencia de los episodios que tienen recompensas acumuladas negativas, mostrando que conforme va avanzando el entrenamiento también mejora la recompensa, y que dentro de los pasos entrenados todavía no existía una convergencia, siendo una oportunidad el aumentar el número de pasos en próximos trabajos.

Adicionalmente, se puede observar la contribución de cada uno de los componentes sobre la recompensa general, siendo la recompensa por distancia R_{dist} la de mayor aportación seguida de la recompensa según el rango en el marco legal R_{rango}. La recompensa por distancia busca incentivar a la formación a acercarse lo más rápido

posible a la carga útil, en el que se observa un comportamiento similar a la general, con una tendencia positiva de recompensas conforme avanza el entrenamiento y además va tomando mayores valores de recompensa acumulada y disminuyendo los valores negativos.

La recompensa según la zona R_{zona} busca incentivar a que los UAV's se acerquen a la carga útil por medio de recompensas intermedias en función de la zona, definidas por una distancia sobre el plano XY a la que se encuentren los UAV's y la carga útil, observándose que los valores iniciales de esta recompensa son muy bajos pero conforme avanza el entrenamiento se comienzan a obtener valores cada vez más grandes en las recompensas acumuladas, lo que indica que los UAV's cada vez entran a zonas más cercanas a la carga útil.

Las recompensas según el rango en el marco legal R_{rango} buscan incentivar a que los UAV's estén dentro del rango legalmente permitido, observándose que en promedio se mantiene constante y no existe una evolución significativa en la dispersión de sus valores en el transcurso del entrenamiento, sumando en promedio la misma cantidad por episodio, pero es un componente importante para generar restricciones en el entrenamiento.

La recompensa asociada a realizar la captura $R_{captura}$ busca incentivar a los UAV's a capturar la carga útil lo más centrado posible en la red y lo más suave posible. Como se mencionó, en ninguno de los 2 experimentos se logró realizar la captura de la carga útil, aunque en un episodio se sumaron recompensas asociadas a la señal de captura, es importante mencionar que esta señal no sólo genera recompensas por realizar la captura, sino que genera recompensas intermedias cuando están muy cerca de la carga útil.

La señal de recompensa de captura detectada en un episodio muestra que es posible realizar la captura, pero es necesario realizar un análisis detallado de los componentes para detectar mejoras, así como la calibración de los hiperparámetros para lograr capturar la carga útil, e.g., al incrementar el número total de pasos existe la posibilidad de que se comience a conseguir la captura de la carga útil.

El castigo por tiempo R_{tiempo} busca incentivar a los UAV's a capturar la carga útil lo más rápido posible, observándose que el castigo acumulado también se incrementa conforme avanza el entrenamiento, mostrando que los UAV's aprendieron a no colisionar desde los primeros pasos, permitiendo estar más tiempo en el entorno, lo que se traduce en un incremento en el castigo por el tiempo.

5.1.1. Análisis de la recompensa por distancia R_{dist}

En la Fig. 5.4 se muestra el valor de la recompensa acumulada de los episodios, así como el promedio de la recompensa acumulada por distancia R_{dist} y de sus componentes por grupo de iteración. El componente $R_{altitud}$ busca incentivar a los UAV's

a acercarse lo más pronto posible en el rango permitido a la carga útil en términos de altitud, observándose un ligero incremento en la recompensa promedio conforme avanza el entrenamiento, también se observa que desde los primeros pasos existen episodios que alcanzan valores máximos, indicando que los UAV's fueron aprendiendo a mejorar la diferencia de alturas.

El componente $R_{dist,min}$ busca incentivar que los UAV's se acerquen lo más rápido posible a la carga útil al dar recompensas cada vez que se mejora la distancia mínima global, pero se observa que el promedio de las recompensas acumuladas es negativo. Para entender dicho comportamiento es necesario analizar con mayor detalle sus componentes: la búsqueda $R_{dist,busqueda}$ que procura acercarse lo más rápido posible a la carga útil, el rescate $R_{rescate}$ que procura estar en una zona que permita la captura lo más suave posible, y un castigo por alejarse $R_{dist,alejar}$.

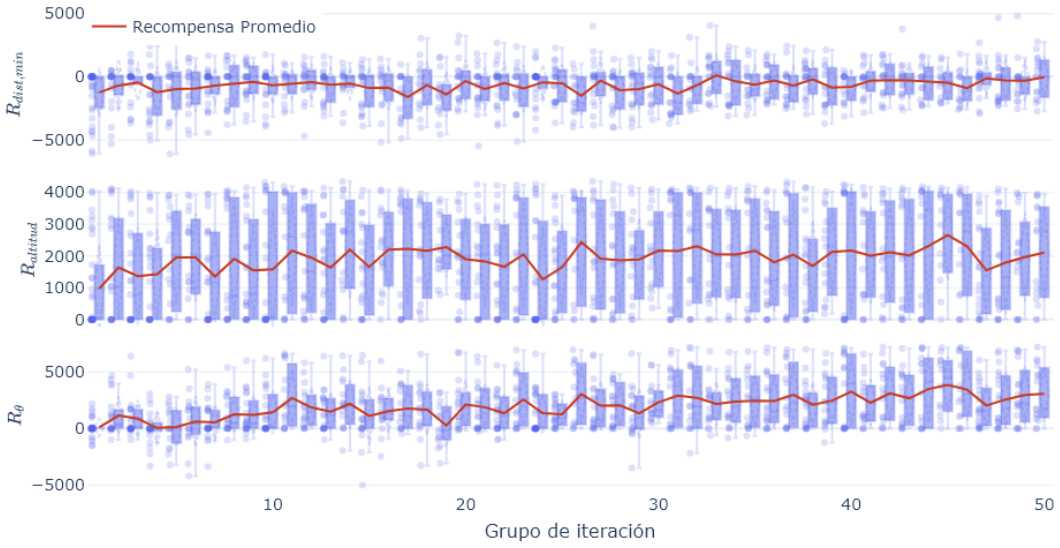


Figura 5.4: Recompensa acumulada de los episodios (*boxplot*) y el promedio de la recompensa acumulada por distancia R_{dist} (línea roja) y de sus componentes $R_{dist,min}$, $R_{altitud}$, R_θ por grupo de iteración.

En la Fig. 5.5 se muestra el detalle de los componentes de R_{dist}, donde se observa que la recompensa por estar en la etapa de búsqueda es mayor y además va incrementando en función del entrenamiento, mostrando que la señal de recompensa por mejorar la distancia a la carga útil en pasos más grandes ayuda en el entrenamiento ($R_{dist,busqueda}$), pero dichas recompensas se están cancelando con el castigo por alejarse en un cierto umbral de la distancia mínima ($R_{dist,alejar}$), siendo una posible mejora, ser más laxos con dicho castigo, permitiendo obtener más recompensas por acercarse. Adicionalmente, se observa que la etapa de rescate $R_{dist,rescate}$ comienza con señales

muy bajas de recompensa y conforme avanza el entrenamiento comienzan a generarse cada vez más episodios con valores superiores a cero para este componente, dicho comportamiento es el esperado dado que el espacio de búsqueda es muy grande, entonces es natural que la mayor parte del tiempo se encuentre en dicha etapa, por el otro lado, la etapa de rescate significa que los UAV's se encuentran relativamente cerca para poder realizar la captura, indicando así que conforme avanza el entrenamiento los UAV's aprenden a acercarse a la carga útil, aunque con ritmos lentos.

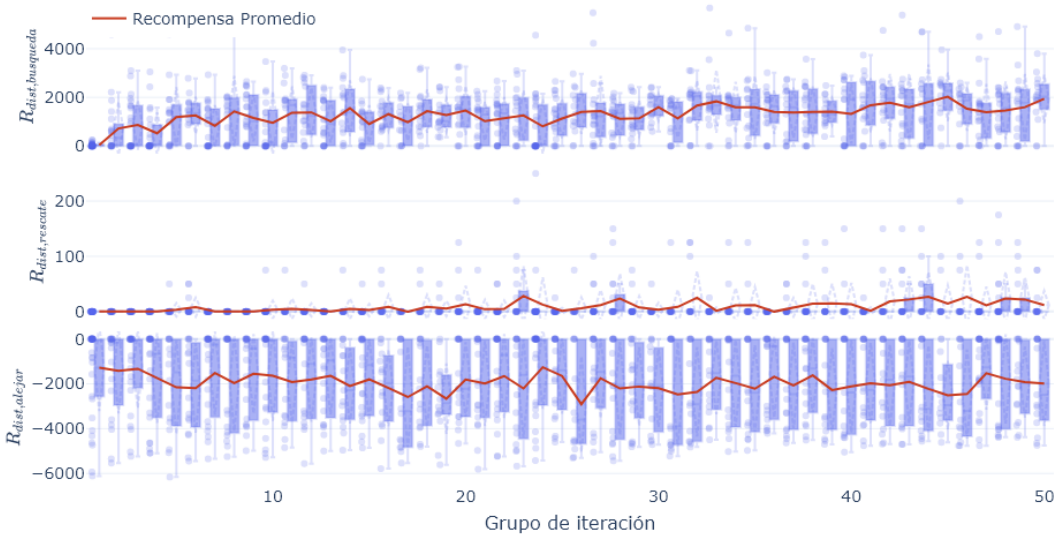


Figura 5.5: Recompensa acumulada de los episodios (*boxplot*) y el promedio de la recompensa acumulada por la distancia mínima $R_{dist,min}$ (línea roja) y de sus componentes $R_{dist,busqueda}$, $R_{dist,rescate}$, $R_{dist,alejar}$ por grupo de iteración.

En la Fig. 5.6 se observa el valor promedio de los coeficientes calculados para $R_{dist,min}$, dichos coeficientes son pesos asociados entre [0,1] de los distintos componentes. En la Tabla 4.1 se muestran los valores máximos para cada uno de los coeficientes, tal que $r_{d,min} = 0.7, r_{d,max} = 0.2, r_{prox} = 0.1, r_{zona} = 0.3$. En la Fig. 5.6 se observa que tanto el coeficiente $r_{d,min}$ y $r_{d,max}$ se acercaron a sus valores máximos y además tienen una tendencia positiva conforme avanza el entrenamiento, estos coeficientes están relacionados con que los UAV's se acerquen lo más rápido posible a la carga útil, mostrando que efectivamente son componentes que ayudan a guiar a los UAV's a realizar la captura. Por otro lado, los coeficientes r_{prox} y r_{zona} prácticamente son nulos, por lo que se podría incrementar el peso asociado a $r_{d,min}$ y $r_{d,max}$ y disminuirlo en r_{prox} y r_{zona}, guiando a los UAV's a realizar la captura.

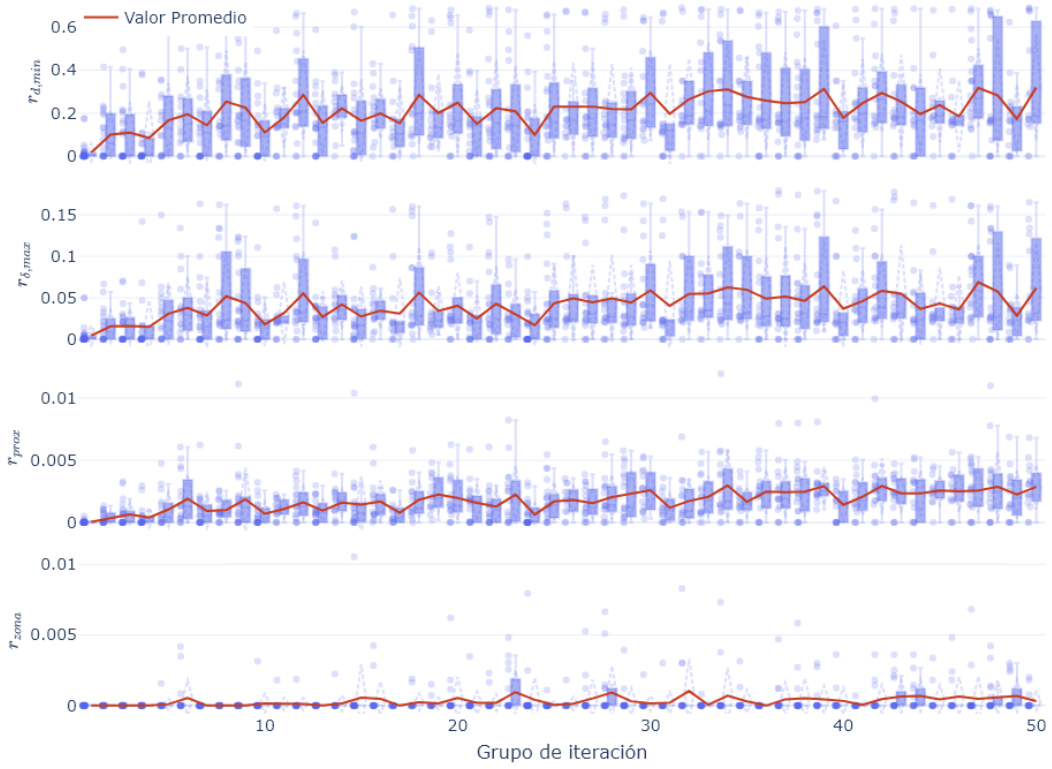
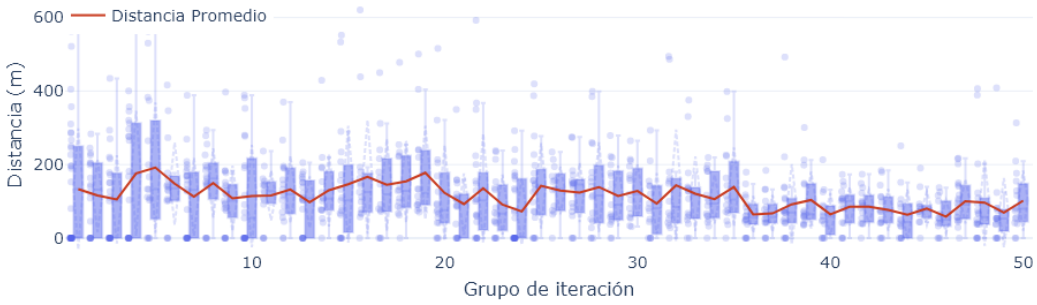


Figura 5.6: Valor promedio de los episodios (*boxplot*) y el valor promedio (línea roja) de los distintos coeficientes de R_{dist} por grupo de iteración.

Por último, se realiza un análisis más detallado respecto la distancia en el plano XY de los UAV's respecto a la carga útil. En la Fig. 5.7a se observa que conforme avanza el entrenamiento la distancia promedio también va disminuyendo al igual que la dispersión de los valores, disminuyendo los valores máximos promedio de distancia. Adicionalmente, se agruparon los episodios en 5 grupos por orden de entrenamiento, donde el primero contiene los primeros 180 episodios y el último contiene los últimos 143 episodios (720-863), permitiendo analizar con mayor granularidad las distancias para cada grupo. En la Fig. 5.7b se muestra la distancia promedio de cada grupo, observándose una clara tendencia en la disminución de la distancia, pero aún con valores promedio relativamente altos para cumplir con el objetivo.



(a) Distancia promedio por episodio y grupo de iteración.

(b) Distancia promedio por grupos.

Figura 5.7: Distancia promedio en el plano XY en los distintos grupos.

En la Fig. 5.8 se muestran los perfiles de velocidad en cada paso de tiempo para cada episodio dividido en los 5 grupos, el color azul muestra las trayectorias que tienen una distancia mínima entre los UAV's y la carga útil en XY mayor a 5 metros, mientras que el color rojo muestra las trayectorias que consiguieron una distancia mínima en XY menor o igual a los 5 metros, es decir, las que se acercaron más a la carga útil. La Fig. 5.8 permite visualizar la evolución de la distancia mínima XY a lo largo del entrenamiento, observando varios comportamientos interesantes:

- En el primer grupo se observa que se tienen trayectorias muy ruidosas que incluso terminan con valores mayores de distancia que en los primeros pasos.
- En los primeros pasos se marca una pendiente muy pronunciada hacia abajo, indicando que los UAV's buscan acercarse lo más rápido posible a la carga útil.
- La densidad de trayectorias en distancias menores a los 5 m va aumentando conforme avanza el entrenamiento.

- Del lado izquierdo, se muestran las trayectorias con una escala logarítmica en Y con el objetivo de visualizar de mejor manera las distancias más pequeñas obtenidas, observando que conforme avanza el entrenamiento, la densidad de trayectorias que alcanzan valores de distancia mínima en XY va aumentando conforme avanza el entrenamiento, es decir, los UAV's están aprendiendo a acercarse cada vez más a la carga útil.

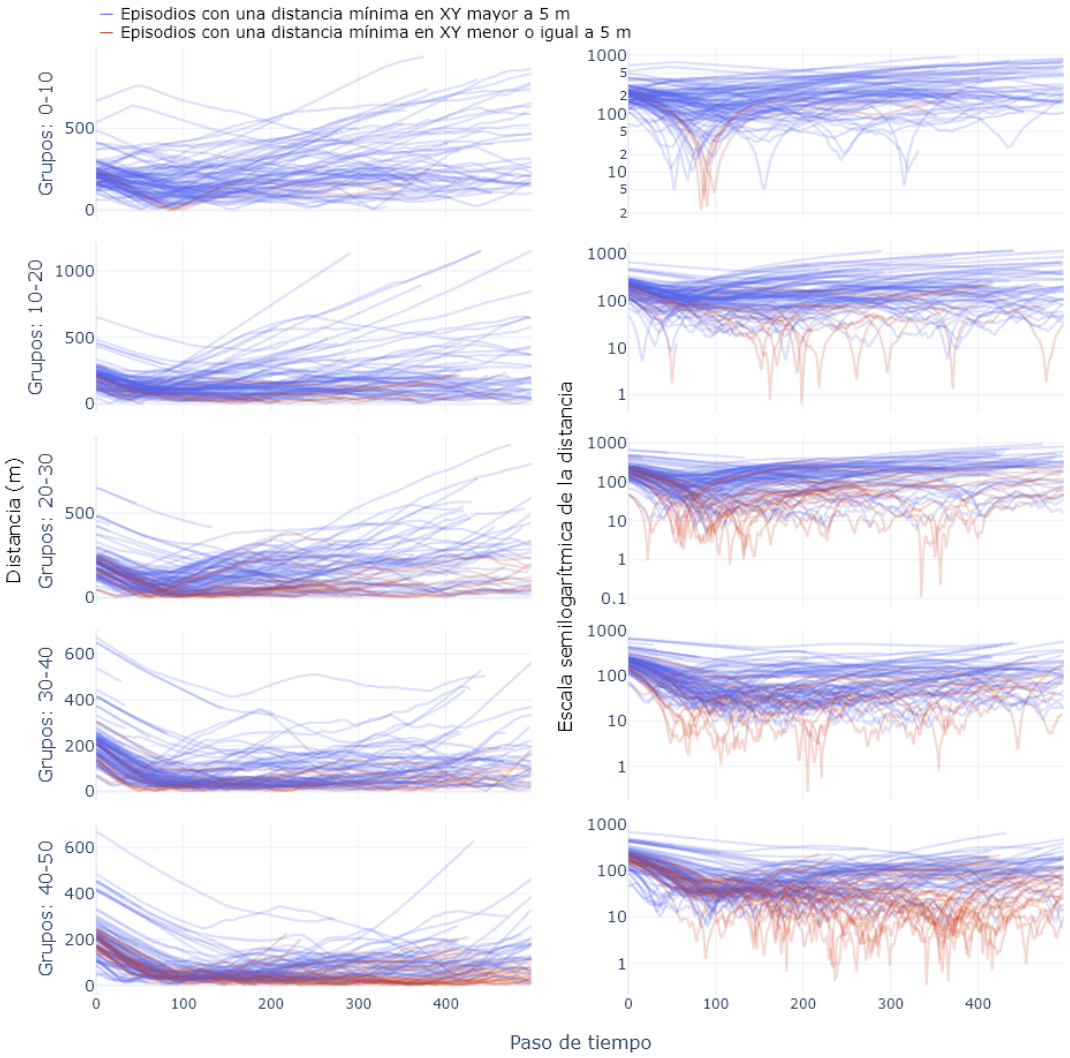


Figura 5.8: Perfil de distancias euclidianas en XY en cada paso de tiempo por grupos, observando que conforme avanza el entrenamiento la dispersión de valores de la distancia XY disminuye, convergiendo a valores pequeños.

La Fig. 5.9 muestra la distribución de la distancia XY para cada grupo, en la cual se observa que conforme avanza el entrenamiento la dispersión en los valores de la distancia en XY va disminuyendo, así como la distancia promedio, pero la distribución

de los últimos 2 grupos es muy parecida, lo que puede indicar una posible convergencia en la distancia y que tal vez no sólo sea necesario entrenar con un mayor número de pasos, sino que también podría ayudar la experimentación con algunas modificaciones en los hiperparámetros como aumentar el peso de la recompensa por distancia o ser más laxos en el castigo por alejarse.

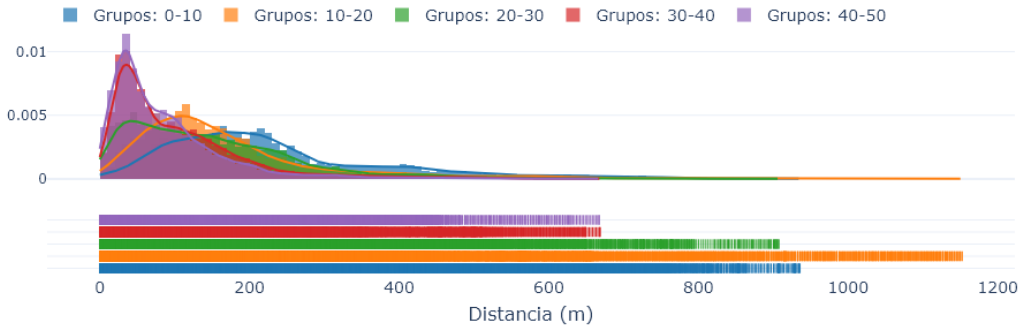
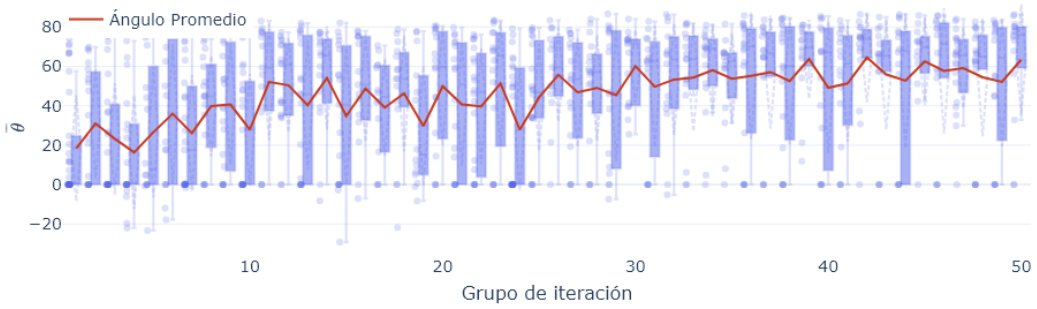


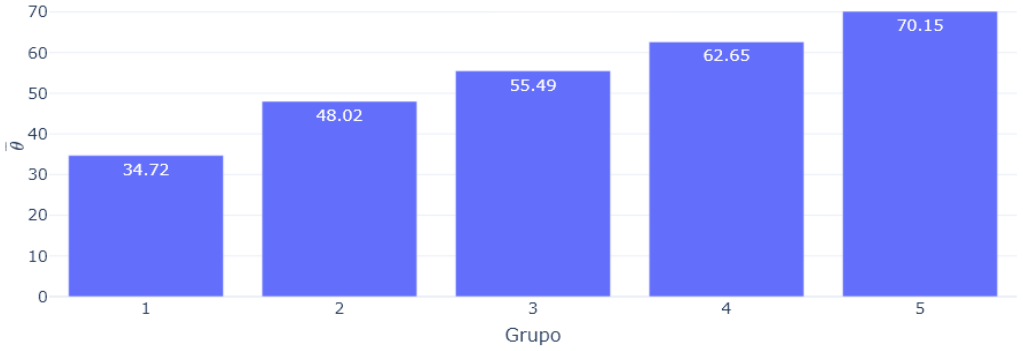
Figura 5.9: Distribución de los valores de la distancia euclidiana por grupo.

5.1.2. Análisis de la recompensa por ángulos R_θ

Las recompensas asociadas a los ángulos respecto a Z $(\theta_{XZ}, \theta_{YZ})$ buscan incentivar a los UAV's a dirigirse a la carga útil y estar por debajo de la misma, es decir, cuando $\theta \approx 90°$. En la Fig. 5.10a se muestra el valor promedio de cada episodio, así como el ángulo promedio $\overline{\theta} = (\theta_{XZ} + \theta_{YZ})/2$ para cada grupo de iteración, en la cual se observa una tendencia a mejorar el valor de $\overline{\theta}$ conforme avanza el entrenamiento, obteniendo valores altos $\overline{\theta} \approx 86°$ con mayor frecuencia y valores pequeños e inclusive negativos con menor frecuencia, estando la mayor parte de los valores en un rango de $\overline{\theta} \in [56°, 80°]$ en el último grupo de iteración. Adicionalmente, en la Fig. 5.10b se muestra $\overline{\theta}$ de cada grupo, afirmando la clara tendencia de ir aumentando $\overline{\theta}$ conforme avanza el entrenamiento, lo que significa que esta señal de recompensa ayuda a reforzar las señales de recompensa asociadas a la distancia R_{dist} y además a buscar a seguir a la carga útil como una sombra, es decir, estar por debajo de la carga útil para realizar la captura lo más centrado posible.



(a) Ángulo promedio $\overline{\theta}$ conformado de los ángulos θ_{XZ} y θ_{YZ} por episodio y grupo de iteración.



(b) Ángulo promedio $\overline{\theta}$ por grupos.

Figura 5.10: Ángulo promedio $\overline{\theta}$ en los distintos grupos.

En la Fig. 5.11 se muestran los perfiles de los ángulos θ_{XZ} (lado izquierdo) y θ_{XZ} (lado derecho) para cada episodio dividido en los 5 grupos, el color azul muestra las trayectorias que tienen una distancia mínima en XY mayor a 5 m, mientras que el color rojo muestra las trayectorias que consiguieron una distancia mínima en XY menor o igual a los 5 m.

La Fig. 5.11 permite visualizar la evolución del ángulo promedio $\overline{\theta}$ a lo largo del entrenamiento, observando que en los primeros grupos hay una alta dispersión en los valores de $\overline{\theta}$, teniendo una alta densidad de trayectorias en un rango de [50°, 90°], pero después de unos pasos de tiempo se comienza a abrir a un rango de [-90°, 90°] observándose una tendencia a disminuir los ángulos obtenidos en los primeros pasos. Dicho comportamiento se va modificando conforme avanza el entrenamiento, manteniendo cada vez más los ángulos de [50°, 90°], así como una tendencia en los primeros pasos a buscar el ángulos objetivo $\theta \approx 90°$, disminuyendo las trayectorias que comienzan a tomar rangos más grandes [-90°, 90°].

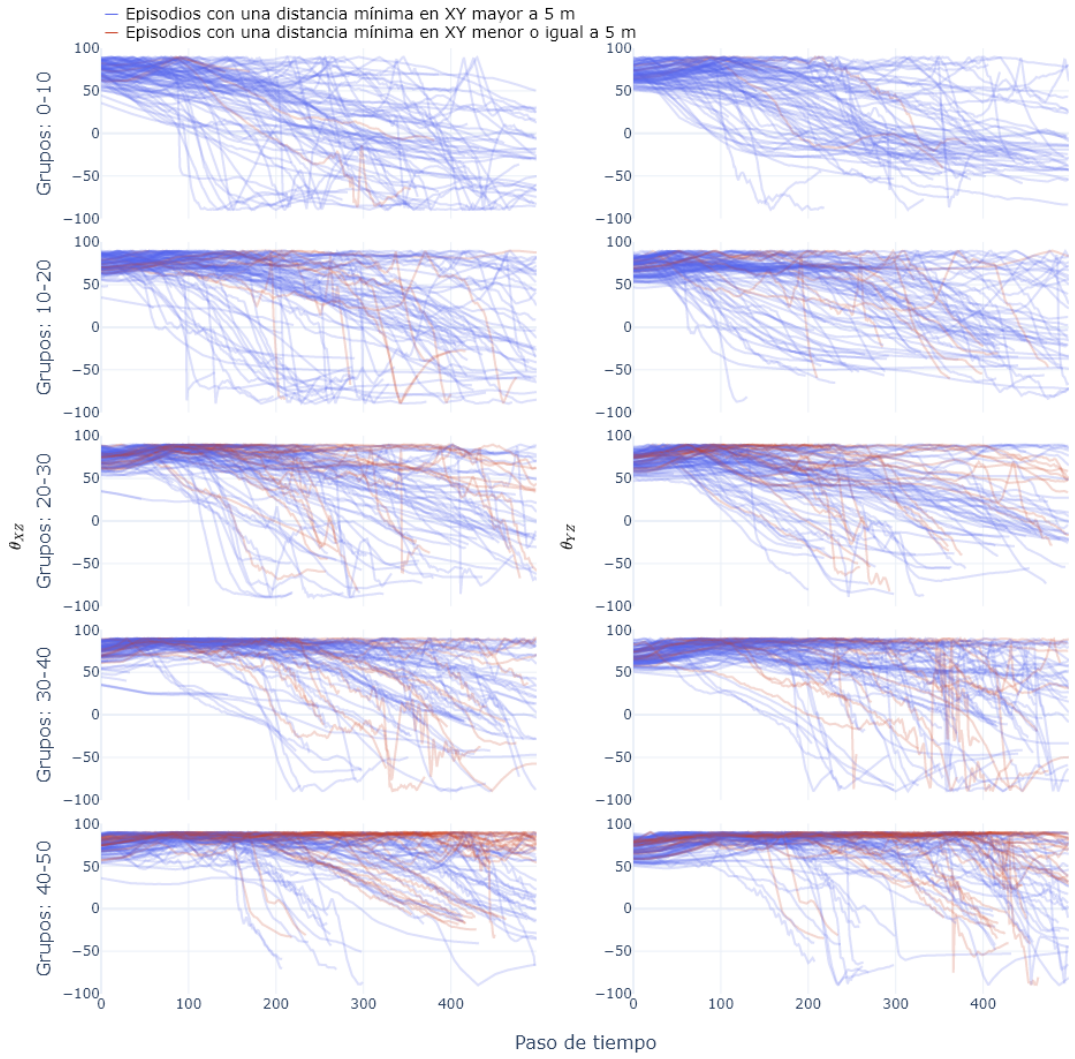


Figura 5.11: Perfil del ángulo promedio $\overline{\theta}$ en cada paso de tiempo por grupos, observando que conforme avanza el entrenamiento la dispersión de valores de los ángulos disminuye, convergiendo a valores objetivo de 90°.

La Fig. 5.12 muestra la distribución de los valores promedio del ángulo $\overline{\theta}$ para cada grupo, en la cual se observa que conforme avanza el entrenamiento la dispersión de los valores de los ángulos va disminuyendo, así como los valores promedio, observándose una distribución con una cola larga pero que se vuelve más angosta conforme se entrena, indicando que existe una mejoría en cada grupo. Como hipótesis, si se entrenara con un mayor número de episodios, la distribución de ángulos tendería a un promedio de $\overline{\theta} \approx 90°$ con una menor variación.

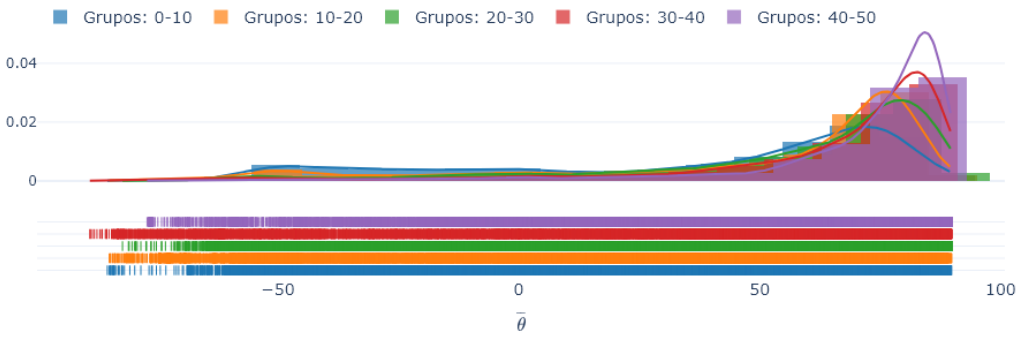


Figura 5.12: Distribución de los valores del ángulo promedio $\overline{\theta}$ por grupo.

5.1.3. Análisis de la recompensa por zona R_{zona}

Las recompensas según la zona buscan incentivar que los UAV's se acerquen a la carga útil por medio de recompensas según la zona en la que se encuentren los UAV's definidas por $\boldsymbol{d_{zonas}} = [5, 10, 15, 25, 40]$, donde cada valor es la zona definida por la distancia XY respecto al centro de la carga útil. En la Fig. 5.13 se muestra la proporción de veces que los UAV's estuvieron en ciertas zonas para cada grupo, mostrando que en las primeras iteraciones prácticamente siempre estuvieron a una distancia mayor a 40 m de la carga útil, pero esa proporción va disminuyendo conforme los UAV's van aprendiendo porque cada vez están en zonas más cercanas, mostrando que todavía existe movimiento entre las proporciones lo que indica que contar con un mayor número de pasos de entrenamiento puede ayudar a que los UAV's vayan mejorando la proporción de las zonas más cercanas hasta alcanzar alguna convergencia o algún estado estable, además se puede observar que este componente puede ayudar a dar recompensas intermedias para realizar la captura.

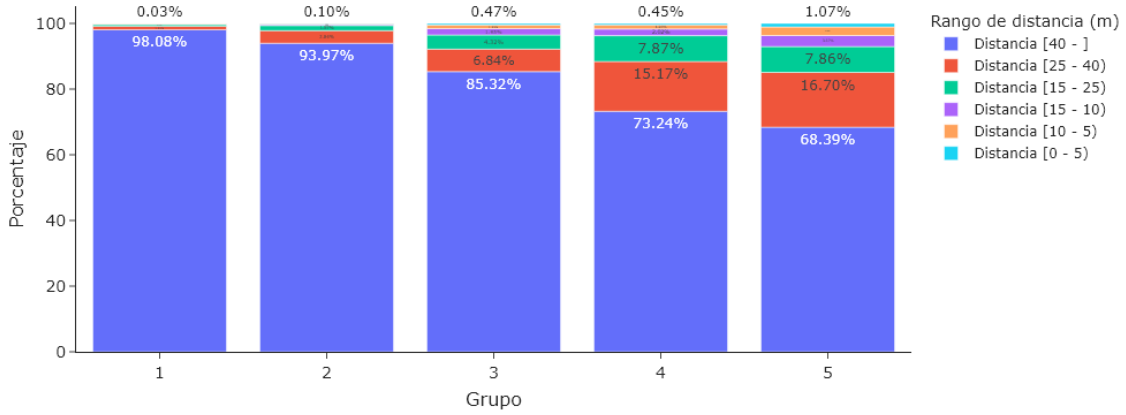


Figura 5.13: Proporción de veces en que la formación de UAV's se encuentra en las distintas zonas definidas por un radio y de centro la carga útil, por grupo.

5.1.4. Análisis de la recompensa por captura $R_{captura}$

Las recompensas asociadas para realizar la captura no solo generan señales de recompensa si se realizó la captura o no, sino que también busca incentivar a los UAV's a realizar la captura de la carga útil lo más centrado posible en la red lo más suave posible (i.e., con una velocidad baja). En la Fig. 5.14 se observa que en todo el entrenamiento se obtuvo una señal de recompensa por la captura de la carga útil en un episodio, siendo un episodio importante de revisar.

Figura 5.14: Recompensa acumulada de los episodios (*boxplot*) y el promedio de la recompensa acumulada por la señal de captura $R_{captura}$ (línea roja) por grupo de iteración.

En la Fig. 5.15 se muestra el valor de la recompensa de cada uno de los componentes de la señal $R_{captura}$ en cada paso de tiempo, siendo R_{exito} la señal asociada a que se realizó de manera exitosa la captura, $R_{zona,red}$ la señal asociada según la zona de

la red en la que se encuentra la carga útil y R_{vel} la señal que incentiva que se realice la captura lo más suave posible. En la Fig. 5.15 se observa que las señales $R_{zona,red}$ y R_{vel} únicamente se activaron en un solo paso de tiempo.

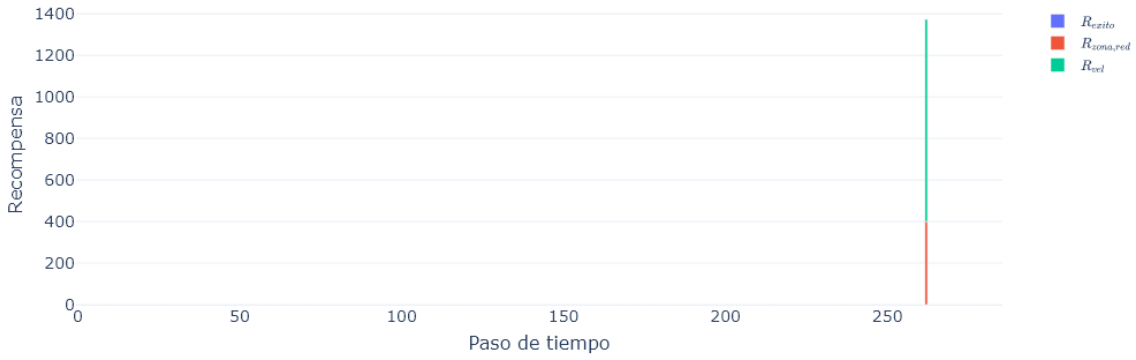
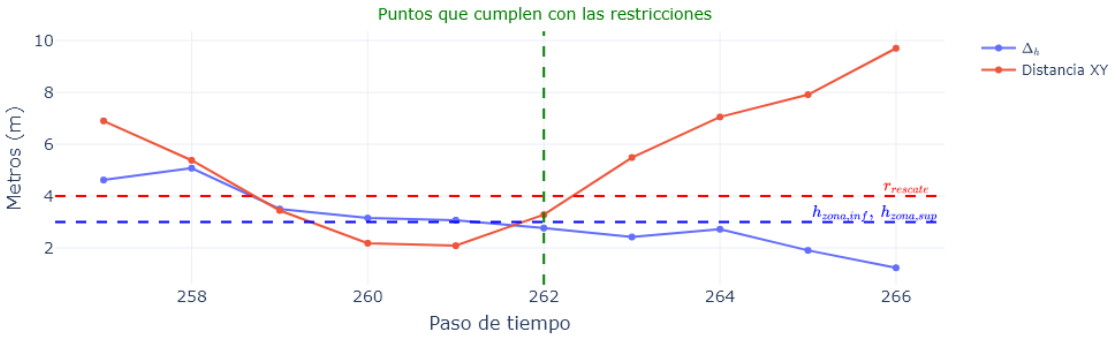
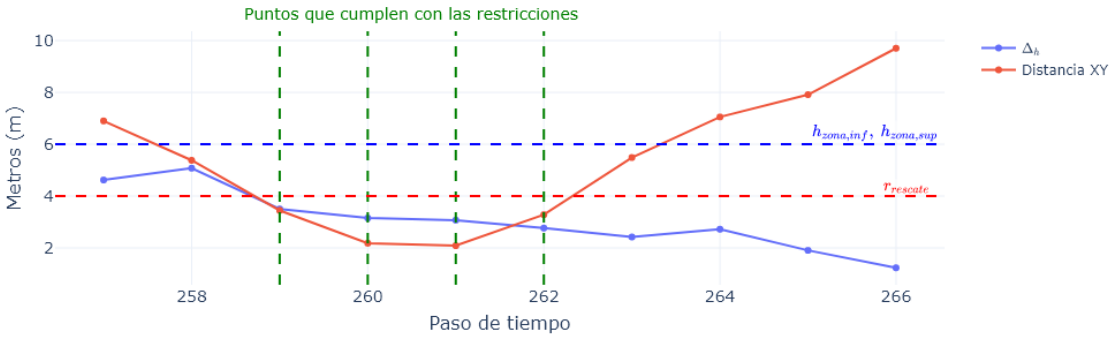


Figura 5.15: Recompensa promedio de los distintos componentes de la recompensa por captura $R_{captura}$ (R_{exito}, $R_{zona,red}$, R_{vel}) por grupo de iteración.

Las señales $R_{zona,red}$ y R_{vel} se activan cuando se cumplen un par de restricciones, cuando $-h_{zona,inf} \leq \Delta h_t \leq h_{zona,sup}$ y cuando la distancia XY $d_{XY} \leq r_{rescate}$, definidos en la Tabla 4.1. Es decir, estos componentes se activan cuando los UAV's están cercanos a la carga útil para generar recompensas intermedias y guiar a los UAV's a realizar la captura, siendo R_{exito} el único componente que se activa cuando se realiza la captura de manera exitosa. En la Fig. 5.16a se muestra un acercamiento en los pasos de tiempo cercanos a cuando se obtuvo dicha recompensa, observando que únicamente un paso de tiempo fue aquel que cumplió ambas restricciones, pero se observa que algunos pasos de tiempo anteriores también están muy cercanos con cumplir con las restricciones, por lo que si se relajaran un poco las restricciones como $h_{zona,inf}, h_{zona,sup}$ se podrían obtener más pasos de tiempo que generen dichas recompensas intermedias para ayudar a guiar a los UAV's a realizar la captura tal como se muestra en la Fig. 5.16b, siendo una posible modificación para mejorar en un siguiente experimento.



(a) Pasos de tiempo que cumplen con las restricciones actuales para obtener señales de $R_{zona,red}$ y R_{vel}.



(b) Pasos de tiempo que cumplen con restricciones más relajadas para obtener señales de $R_{zona,red}$ y R_{vel}.

Figura 5.16: Restricciones en los pasos de tiempo para calcular las señales de $R_{zona,red}$ y R_{vel}.

En la Fig. 5.17 se muestra un resumen de las trayectorias en el espacio y en el plano XY tanto de los UAV's como de la carga útil en cada paso de tiempo, donde los puntos rojos corresponden a los últimos 25 pasos de tiempo en la trayectoria de la carga útil, y adicionalmente se gráfica la distancia en el espacio entre los UAV's y la carga útil en cada paso de tiempo. En la Fig. 5.17 se observa que en cada paso de tiempo va disminuyendo la distancia, y también se puede observar que en los primeros pasos la formación de UAV's se va acercando a la carga útil, después se comienza a desviar pero eventualmente comienza a corregir la trayectoria para acercarse lo más posible a la carga útil, llegando a un punto donde estuvieron a 2.09 m de distancia en el plano XY a una diferencia de altitud de 3.07 m, después de ese punto los UAV's se pasaron de la carga útil alejándose poco a poco de la misma, incrementando así la distancia en los últimos pasos de tiempo. La cercanía a la que estuvo la formación respecto a la carga útil en ese episodio muestra que es posible realizar la captura si

se continúan con experimentos en esta dirección, siendo un trabajo a futuro.

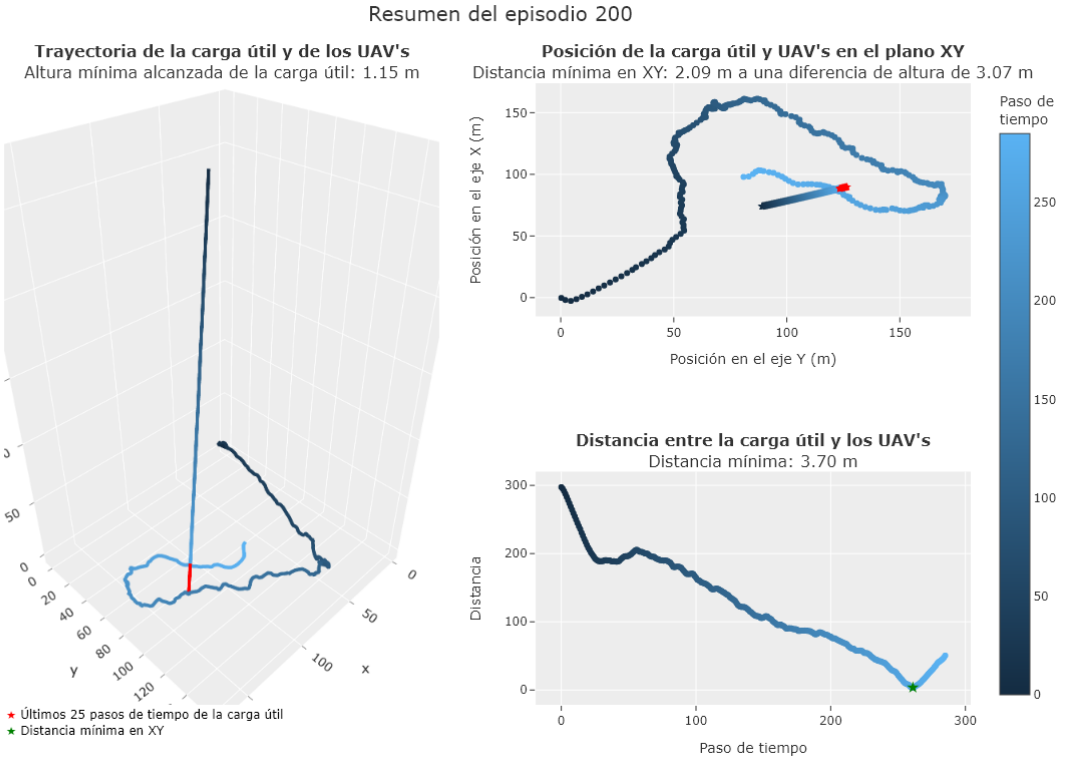


Figura 5.17: Resumen de la trayectoria de los UAV's y carga útil en el episodio donde se obtuvo la señal $R_{captura}$.

5.2. Análisis de los mejores episodios

En esta sección del análisis de resultados se tiene como objetivo ver con mayor detalle aquellos episodios que tuvieron un mejor desempeño respecto a la recompensa acumulada, así como los episodios que lograron una menor distancia en el espacio entre los UAV's y la carga útil.

5.2.1. Episodios con mayor recompensa acumulada

En la Fig. 5.18 se muestran los episodios con mayor recompensa acumulada durante la evaluación, en la cual se observa que no existe una diferencia grande entre todos, siendo un promedio de la recompensa acumulada de los 5 mejores episodios de 16,230. En las Figs. 5.19, 5.20, 5.21 se muestra un resumen de los 3 mejores episodios respecto a la suma acumulada, mostrando la trayectoria tanto en el espacio como en el plano XY de los UAV's como de la carga útil, adicionalmente se muestra la distancia entre ambos en el espacio en función del paso de tiempo, así como un perfil de las recompensas obtenidas por paso de tiempo en sus principales componentes R_{dist}, R_{zona}, R_{rango}, $R_{captura}$, R_{tiempo}.

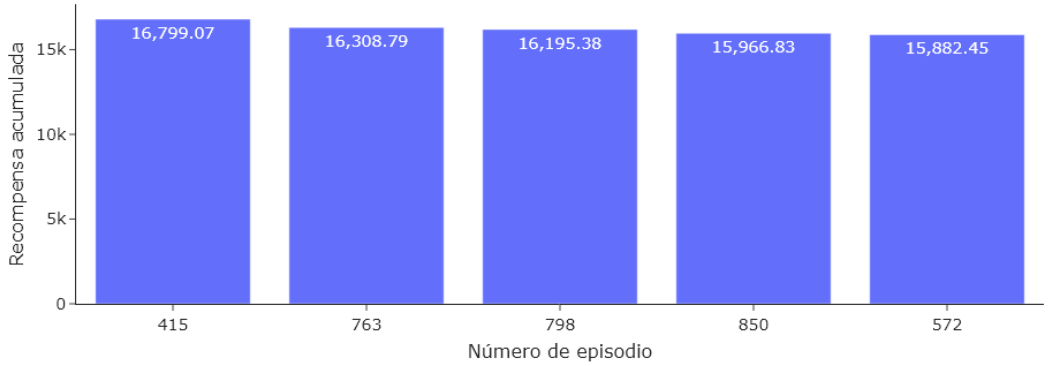


Figura 5.18: Episodios con mayor recompensa acumulada durante la evaluación.

En las Figs. 5.19, 5.20, 5.21 se observa que respecto a la distancia XY, los UAV's generaron una estrategia tal que buscan acercarse lo más rápido posible a la carga útil, en una forma casi de ir en línea recta hacia la carga útil, eventualmente llega un punto en el que comienza a ajustar su posición, siendo una trayectoria un poco más errática pero aun así buscando a la carga útil. Adicionalmente, se observa que existe la posibilidad de que se realice la captura dado que, si alcanza a la carga útil y además llega a pasar relativamente cerca de ella, siendo una oportunidad experimentar con los hiperparámetros para que eventualmente logre aprender a acercarse a la carga útil de manera suave y sobre todo capturarla.

El perfil de recompensas para los 3 eventos tiene un comportamiento similar, las principales recompensas obtenidas en los primeros pasos es por medio del componente de distancia R_{dist}, aunque llega un punto en el que comienzan a disminuir de manera considerable pero también comienzan a incrementar los valores de R_{zona}, observando que en los primeros pasos los UAV's buscan acercarse lo más rápido posible a la carga útil, ingresando cada vez a zonas de menor distancia con mayor recompensa, siendo cada vez más difícil mejorar la distancia mínima global, y por el contrario, siendo más fácil alejarse obteniendo en lugar de recompensas, castigos, como se observó en la Fig. 5.5. Respecto al componente de R_{rango} se observa que prácticamente casi siempre está recibiendo dicha recompensa, incentivando a que se encuentre dentro del área legalmente permitida. Por último, el componente R_{tiempo} es constante dado que siempre se obtiene ese castigo en cada paso de tiempo, buscando incentivar que se realice el objetivo lo más rápido posible.

Adicionalmente, en las gráficas mostradas con las trayectorias en el espacio tanto de los UAV's como de la carga útil se muestra la altura mínima alcanzada por la carga útil, esta puede ser alcanzada por 2 factores, el primero es que los UAV's colisionaron terminando así el episodio, y la segunda es cuando terminan su trayectoria, es decir, cuando los datos en los archivos CSV que contienen los datos de su trayectoria hayan terminado.

Los UAV's en los mejores episodios seleccionados en función de la recompensa acumulada y de la distancia mínima ya aprendieron a no colisionar con el suelo, por lo que el motivo en que se termine la trayectoria de la carga útil es el segundo factor. Ahora bien, al analizar dichos resultados se observan alturas mínimas que van de 1 m a valores de 125 m, siendo algo inesperado dado que dentro del algoritmo desarrollado, al seleccionar `move_to_proximity=teleport` lo que se hace es que la altitud se ajusta tal que la carga útil siempre llegue a 0, la razón detrás de esto es porque los UAV's no sabrán cuando termina la trayectoria y para simplificar el problema se puede suponer que las trayectorias terminan en 0, aunque en experimentos posteriores se puede modificar para incentivar a que realicen la captura lo más rápido posible.

Independientemente a la posición relativa de la altitud para simplificar el problema, se observó una oportunidad de mejora para los experimentos futuros, porque en este experimento los valores de la altitud mínima alcanzada de la carga útil se esperan valores cercanos a 0 y no es lo que se observa, y esto se debe a que la librería *Stable Baselines3* tiene un número máximo de pasos por episodio que son 500 pasos[3], hay trayectorias que tienen esa cantidad o un número menor de pasos, es decir, puntos de la trayectoria de la carga útil en los archivos CSV, pero hay otras trayectorias que tienen más puntos que el número máximo de pasos permitido, este último grupo entonces se interrumpe a los 500 pasos, tal que no terminan la trayectoria. De manera puntual, se debe ajustar este hiperparámetro para asegurar que todas las trayectorias de la carga útil en cada episodio se realicen.

En el episodio 415 (Fig. 5.19) se observa que los UAV's fueron ajustando su trayectoria para ir acercándose a la carga útil, pero la trayectoria de la carga útil terminó antes, aun así el comportamiento es prometedor. En el episodio 763 (Fig. 5.20) se observa la misma estrategia de acercarse lo más rápido posible en los primeros pasos por una línea recta, después los UAV's comienzan a ajustar su trayectoria, tal que se posicionan por debajo de la carga útil, teniendo un comportamiento como si estuvieran esperando a la carga útil, teniendo movimientos erráticos en esa etapa. Por último, en el episodio 798 (Fig. 5.21) se observa que los UAV's llegaron a esperar un tiempo la carga útil, que es el nudo que se observa un poco antes de que termine la trayectoria de los UAV's y después se alejaron, pero en caso de esperar en esa zona del nudo, era posible realizar la captura.

[3]El hiperparámetro de número máximo de pasos por episodios no está implementado de manera directa en la librería, por lo que se tienen que hacer envolventes para poder modificar estos valores.

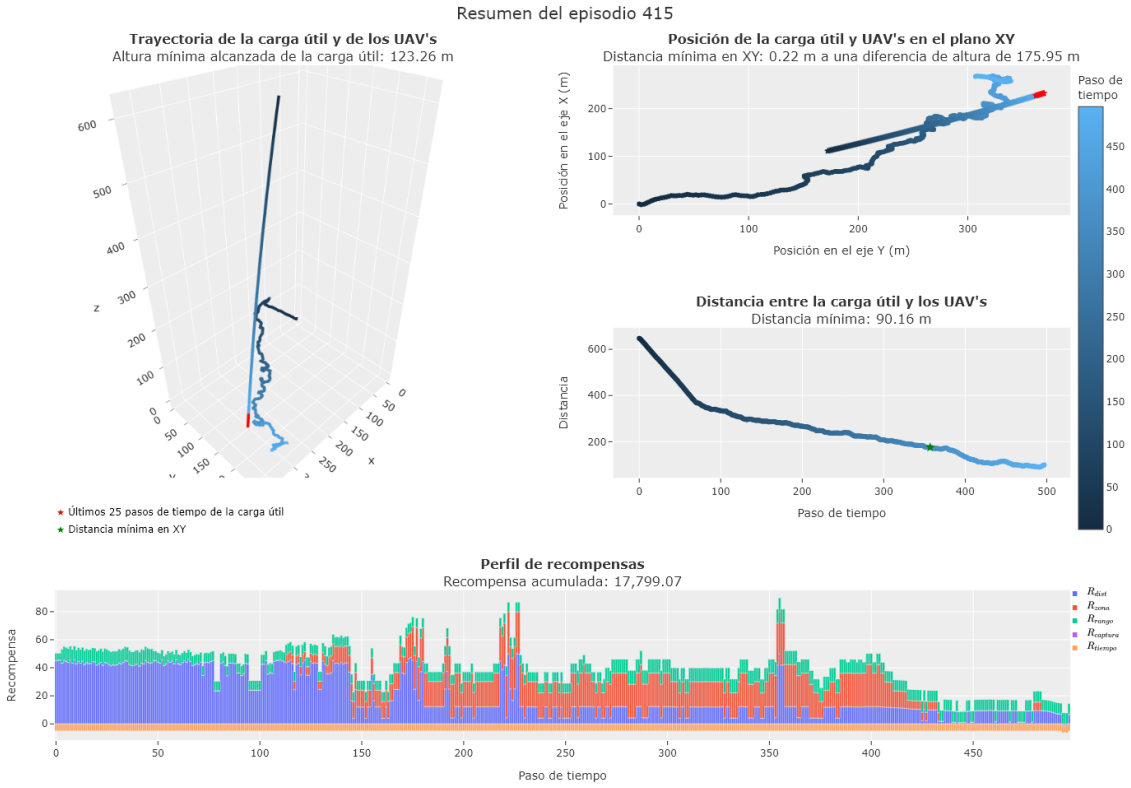


Figura 5.19: Resumen de la trayectoria de los UAV's y carga útil en el episodio 415, siendo el mejor episodio respecto a la recompensa acumulada.

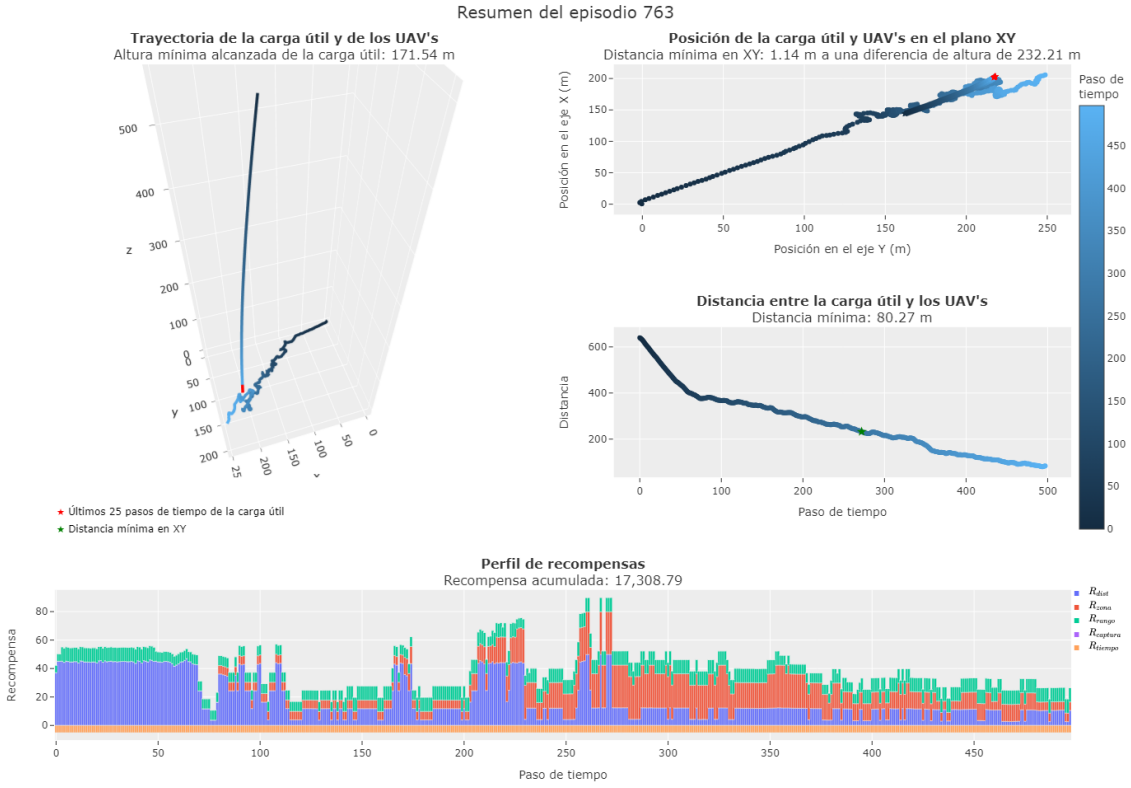


Figura 5.20: Resumen de la trayectoria de los UAV's y carga útil en el episodio 763, siendo el segundo mejor episodio respecto a la recompensa acumulada.

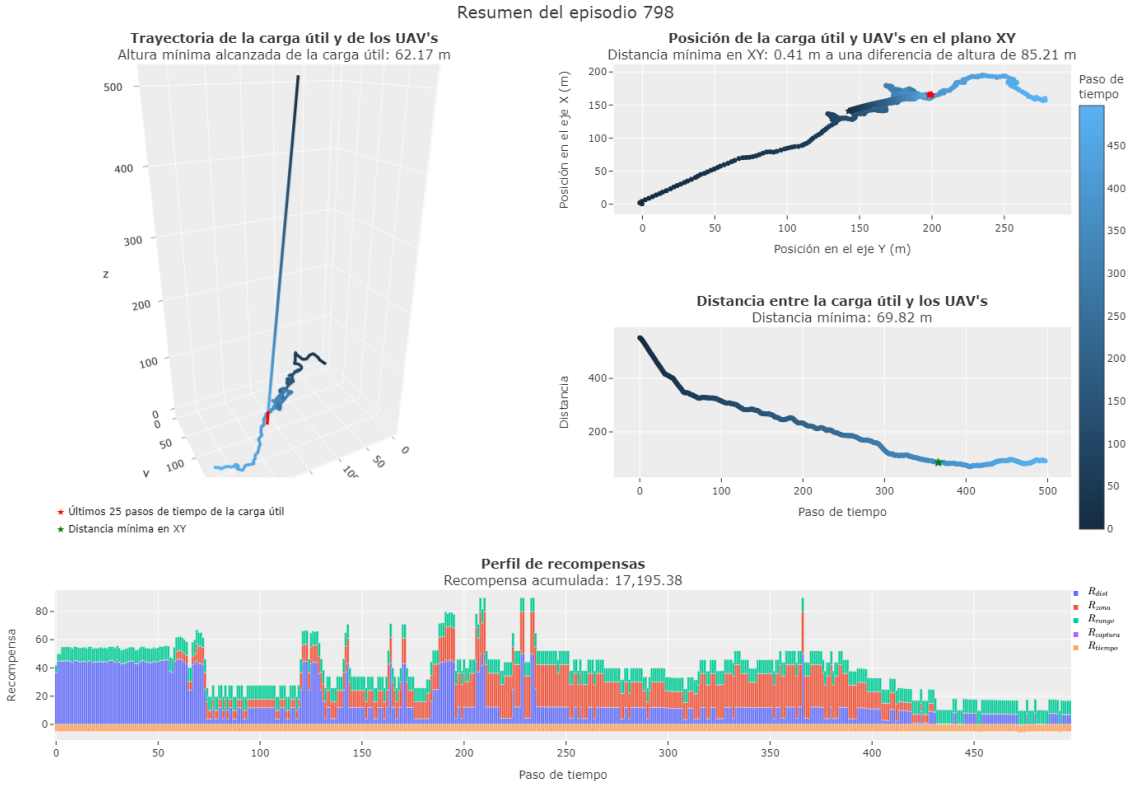


Figura 5.21: Resumen de la trayectoria de los UAV's y carga útil en el episodio 798, siendo el tercer mejor episodio respecto a la recompensa acumulada.

5.2.2. Episodios con menor distancia

En la Fig. 5.22 se muestran los episodios con la menor distancia en el espacio entre los UAV's y la carga útil, en la cual se observa que llegaron a estar a menos de 10 m entre ellos, siendo una buena señal, dado que como se ha mencionado a lo largo de la sección los UAV's si están aprendiendo a acercarse a la carga útil. De igual manera que en los episodios con mayor recompensa acumulada, se analizan los 3 mejores episodios que obtuvieron la distancia mínima mostrados en las Figs. 5.23, 5.24, 5.25. La recompensa acumulada del mejor episodio (200) es de 7,292.07, es decir, un poco menos de la mitad de las mejores recompensas acumuladas de los episodios detectados con anterioridad, por lo que, es importante analizarlos dado que se pensaría que al ser R_{dist} el componente de mayor peso, los episodios con menor distancia deberían tener recompensas acumuladas similares a las de los episodios revisados con anterioridad.

De manera general, en las Figs. 5.23, 5.24, 5.25 se observa que tienen una trayectoria más errática que las trayectorias revisadas en la subsección anterior, es decir, todavía no tienen la estrategia de acercarse lo más rápido posible a la carga útil, iterando y corrigiendo la trayectoria según pasa el tiempo, pero acercándose demasiado en algún punto. Además, en el perfil de recompensas se observa que aún no se tiene una estabilidad en las recompensas obtenidas por la distancia, es decir, todavía

se está penalizando por alejarse, o después de un tiempo comienzan a recibir esas recompensas, lo que indica que ajustaron su distancia para acercarse a la carga útil, obteniendo recompensas por estar en zonas cercanas a la carga útil, pero esto sucede en los últimos pasos de tiempo.

El episodio 200 (Fig. 5.23) es el que fue analizado en la sección del análisis de la recompensa por captura (5.1.4), es el único episodio que consiguió recompensas por el componente $R_{captura}$, esto es porque estuvo muy cerca de la carga útil, observándose que son recompensas muy grandes comparadas con las demás, pero que únicamente suceden un 1 paso de tiempo por las restricciones asociadas a dicho componente. En el episodio 304 (Fig. 5.24) se observa de manera clara el ajuste que fueron realizando los UAV's para acercarse a la carga útil, obteniendo así recompensas altas por R_{dist} y R_{zona} en los últimos pasos de tiempo que logró acercarse, estando a una distancia en XY de 2.84 m a una diferencia de altura de 4.55 m, es decir, estuvo muy cerca de estar en una zona para realizar la captura. En el episodio 304 (Fig. 5.25) se observan muchos ajustes en la trayectorias, pero en un momento estuvo a 4.74 m en XY, pero también después de estar muy cerca tiene un comportamiento el que se aleja de la carga útil.

En comparación de los episodios analizados, se observa que los episodios con mayor recompensa acumulada ya tienen una estrategia clara en los primeros pasos de tiempo, acercándose a la carga útil en línea recta tal que acceden a las recompensas por estar en zonas más cercanas a la carga útil en menos pasos de tiempo, mostrando trayectorias más suaves en comparación de los episodios con la distancia mínima. Una propuesta de mejora es entonces ajustar los hiperparámetros de los componentes de R_{dist} para que se pueda prolongar esa línea recta, o mejorar el movimiento errático que tiene la trayectoria después de un tiempo (i.e., que se suavice), específicamente cuando están en zonas relativamente cercanas a la carga útil.

En conclusión, en el análisis de los mejores episodios se observó que si bien, en ningún episodio se realizó la captura, existieron episodios que mostraron que es posible realizar la captura con la metodología planteada porque los UAV's se acercaron lo suficiente para pensar que se puede dar la captura.

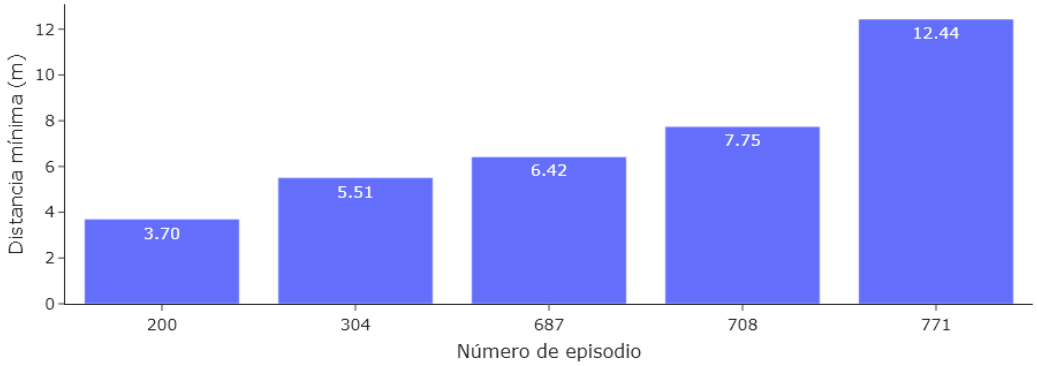


Figura 5.22: Episodios con la menor distancia entre los UAV's y la carga útil durante la evaluación.

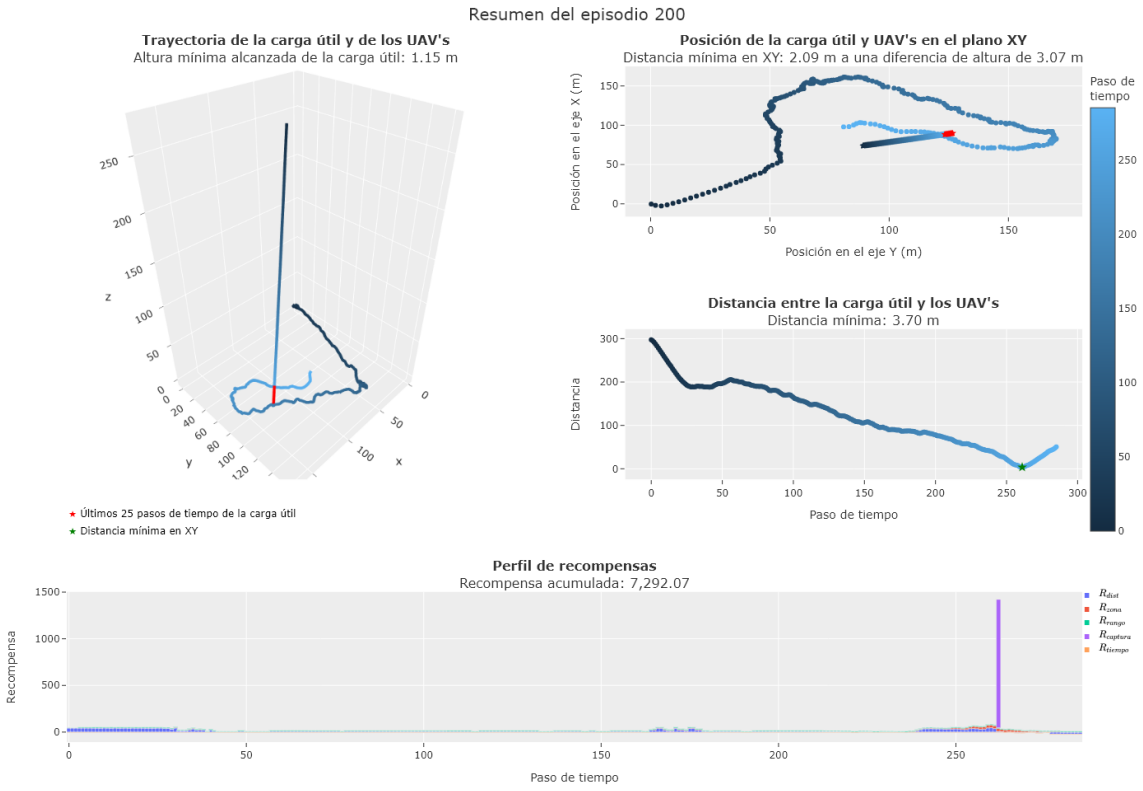


Figura 5.23: Resumen de la trayectoria de los UAV's y carga útil en el episodio 200, siendo el mejor episodio respecto a la distancia mínima.

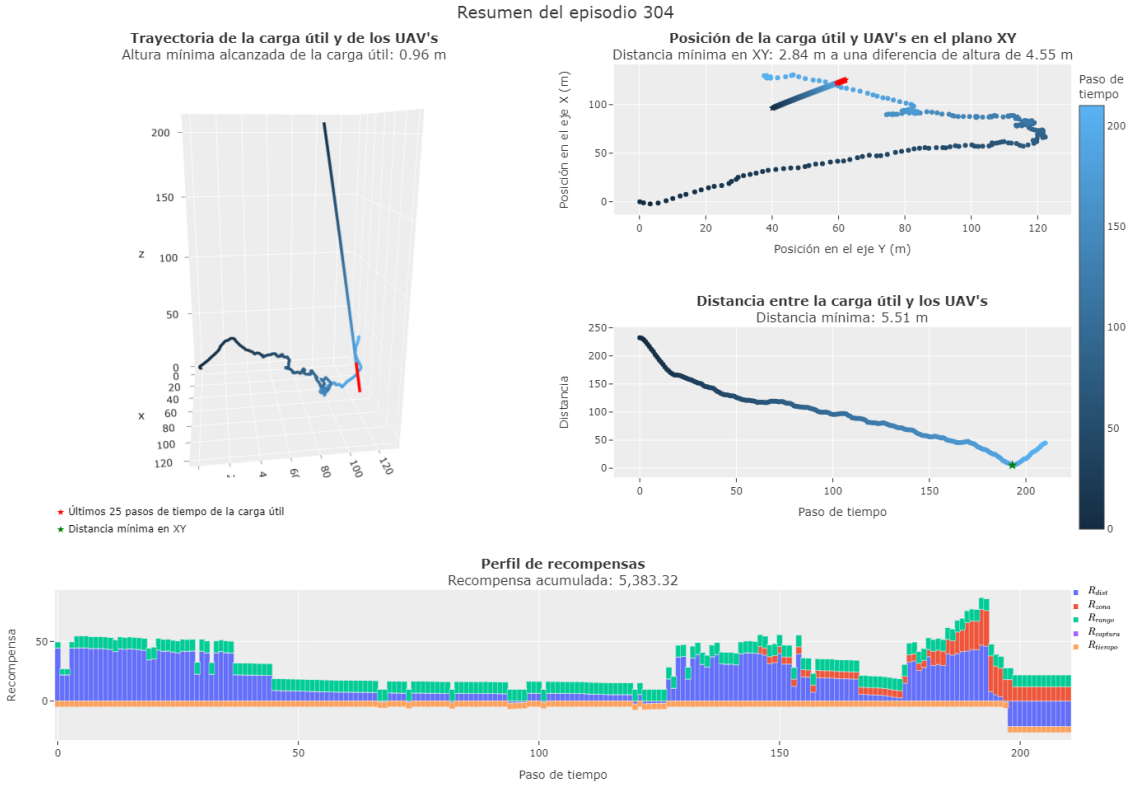


Figura 5.24: Resumen de la trayectoria de los UAV's y carga útil en el episodio 304, siendo el segundo mejor episodio respecto a la distancia mínima.

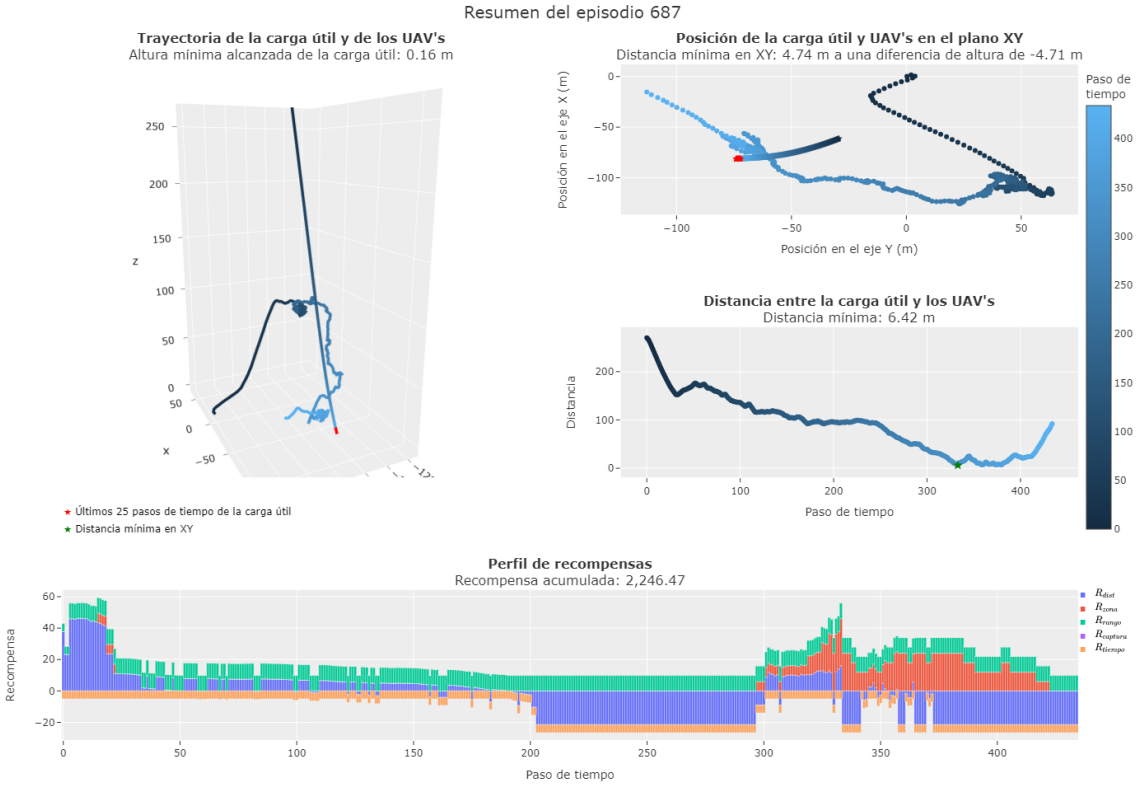


Figura 5.25: Resumen de la trayectoria de los UAV's y carga útil en el episodio 687, siendo el tercer mejor episodio respecto a la distancia mínima.

6 Conclusiones y trabajo a futuro

6.1. Conclusiones

El objetivo principal de la tesis es modelar un POMDP que permita la coordinación de un sistema multi-robots UAV's con el objetivo de búsqueda y rescate de la carga útil de un globo de gran altitud (HAB), resolviéndolo mediante algoritmos de aprendizaje por refuerzo. Dicho objetivo se cumplió de manera parcial, ya que se consiguió desarrollar el POMDP y un marco de trabajo permitiendo el entrenamiento de un par de algoritmos de aprendizaje por refuerzo (PPO y SAC) con distintos espacios de acciones (discreto y continuo), pero con la capacidad y flexibilidad de utilizar más algoritmos de aprendizaje por refuerzo.

Asimismo, el objetivo de realizar la búsqueda se cumple, porque los resultados muestran que los UAV's conforme se entrenaron, fueron aprendiendo estrategias para buscar cada vez con mayor agilidad la carga útil. El objetivo de realizar el rescate no se cumplió, es decir, los UAV's no consiguieron rescatar la carga útil en ningún momento con el mejor modelo entrenado (SAC con un espacio de acciones continuas), aunque los resultados muestran comportamientos prometedores que con la modificación de algunos hiperparámetros como el número de pasos de entrenamiento probablemente permitirán que los UAV's eventualmente logren generar las políticas necesarias para realizar la captura.

Adicionalmente se cumplió con el objetivo particular de generar un entorno de simulación para el entrenamiento de los algoritmos de aprendizaje por refuerzo, generando un marco de trabajo que emplea tecnologías existentes considerando la dinámica y control de UAV's, la física del entorno, las interacciones entre los multi-agentes (UAV's, carga útil, red, suelo). La dinámica de la carga útil se obtuvo prediciendo alrededor de 1,200 trayectorias de descenso en función del lugar, día y hora del lanzamiento del HAB con el software *Cambridge University Spaceflight Prediction*, guardando para cada trayectoria hora, latitud, longitud y altitud en un archivo CSV, generando así un banco de simulaciones para ejecutar el entrenamiento.

El marco de trabajo permite realizar acciones de los UAV's en el entorno, regresando las observaciones del estado, así como la señal de recompensa. El entorno de simulación fue desarrollado en *Unreal* con la capacidad de simular la física del entorno y las interacciones entre los múltiples agentes. La interacción del entorno de simula-

ción con los UAV's se realizó mediante APIs para controlar a los UAV's utilizando *AirSim*. El entorno con sus conexiones se envolvieron en un entorno de entrenamiento de *OpenAI Gym* para entrenar los algoritmos de aprendizaje por refuerzo de la librería *Stable Baseline3*[1]. Por lo tanto, el marco de trabajo puede servir para solucionar distintos problemas (e.g., vehículos autónomos, diseño de misiones), entrenando o probando algoritmos en ambientes virtuales para posteriormente llevarlos a ambientes reales.

El número de hiperparámetros del modelo es alto, algunos de ellos fueron definidos a lo largo del trabajo, otros fueron encontrados en las distintos experimentos realizados, específicamente el valor máximo de las recompensas ($r_{max,dist}$, r_{zona}, $r_{max,rango}$, $r_{max,captura}$, r_{tiempo}, $r_{colision}$, $r_{fin,trayectoria}$), así como el peso de cada uno de sus componentes. Los demás fueron definidos de manera heurística. Es importante mencionar que los experimentos previos no únicamente se variaron los hiperparámetros, sino que también las señales de recompensa, por lo que no se tuvo el tiempo suficiente para experimentar con el valor de los hiperparámetros del modelo presentado en el trabajo, siendo una mejora obligatoria en los siguientes pasos para mejorar el rendimiento del modelo.

Los algoritmos probados fueron PPO con un espacio discreto y continuo de acciones y SAC con un espacio continuo de acciones, dichos algoritmos fueron seleccionados en función de la literatura encontrada [1, 2, 3, 4, 5]. Para cada uno de los escenarios se ejecutaron 5 experimentos para comparar su desempeño, así como validar que los distintos grupos fueran significativamente diferentes entre sí utilizando la prueba Wilcoxon. Los resultados mostraron que utilizar el algoritmo SAC daba mejores resultados que PPO para resolver este problema en particular, debido a su capacidad de manejar espacios de acciones continuos y su estrategia de exploración eficiente mediante la maximización de entropía, y además uno de los puntos importantes que se observaron a lo largo de los resultados es que es posible que aumentando el número de pasos de entrenamiento se pudiera lograr una convergencia de los resultados e inclusive comenzar a realizar las capturas.

Los resultados muestran que los UAV's lograron aprender estrategias de acercamiento hacia la carga útil (búsqueda), lo que sugiere que aprendieron a acercarse lo más rápido posible a la carga útil. Adicionalmente se observó que para los episodios con la menor distancia en el espacio entre los UAV's y la carga útil se tienen trayectorias erráticas y recompensas acumuladas de la mitad que de los mejores episodios, pero a pesar de ello, hubo momentos que los UAV's estuvieron lo suficientemente cerca como para sugerir que el aprendizaje podría mejorar con ajustes en los hiperparámetros.

El análisis detallado de los componentes de la señal de recompensa reveló información valiosa sobre su impacto y posibles mejoras futuras. La recompensa basada en

[1]El objetivo particular es probar algoritmos, no desarrollarlos, por lo que se utilizó la librería *Stable Baseline3* que contiene una serie de algoritmos de aprendizaje por refuerzo ya implementados

la distancia resultó crucial para dirigir eficientemente a los UAV hacia la carga útil, generando recompensas significativas al acercarse y permanecer cerca. La recompensa por ángulos indicó una tendencia positiva en la mejora de los valores promedio, sugiriendo que los UAV aprendieron a posicionarse estratégicamente. La recompensa por rango demostró estabilidad, cumpliendo con las restricciones legales de vuelo. La recompensa por zona motivó a los UAV a aproximarse a la carga útil, con una progresiva mejora durante el entrenamiento. Aunque no se logró la captura, se identificaron posibles mejoras al relajar las restricciones asociadas a $R_{zona,red}$ y R_{vel}, potencialmente aumentando las recompensas intermedias y guiando a los UAV hacia el logro del objetivo. Sin embargo, la falta de convergencia en los valores señala la oportunidad de optimizar el entrenamiento.

En conclusión, a pesar de no haber logrado la captura exitosa, los resultados sugieren que el modelo de aprendizaje por refuerzo implementado ha permitido que los UAV's aprendan estrategias de búsqueda al acercarse cada vez más a la carga útil. La diversidad en las trayectorias y las oportunidades de mejora identificadas proporcionan una base sólida para experimentos futuros. Con ajustes cuidadosos en los hiperparámetros y restricciones, así como la consideración de mejoras en la lógica de la captura, se espera que el modelo alcance un rendimiento más consistente y logre la captura de la carga útil. Este trabajo sienta las bases para futuras investigaciones en el campo del control distribuido de UAV's para tareas de captura de objetos móviles.

6.2. Trabajo a futuro

Para el trabajo a futuro se proponen las siguientes recomendaciones, aunque algunas pueden tener mayor relevancia según el caso de estudio:

- Mejorar el modelo actual con las mejoras identificadas:
  - Incrementar el número de pasos de entrenamiento, basado en la observación de que algunas mejoras podrían ocurrir con más pasos de entrenamiento.
  - Modificar las restricciones asociadas a $R_{zona,red}$ y R_{vel} para generar mayores recompensas intermedias cuando los UAV's se encuentren muy cerca de la carga útil, guiando a los UAV's a realizar la captura lo más centrado y suave posible.
  - Modificar el número máximo de pasos por episodio, garantizando que todas las trayectorias de la carga útil se completen, evitando interrupciones prematuras.
  - Experimentar con los valores de $R_{dist,busqueda}$ y $R_{dist,alejar}$ porque actualmente se están anulando ambas señales, una opción podría ser relajar un poco la penalización por alejarse, permitiendo que el componente de búsqueda tome más influencia, pero podría ser el caso en que al no tener que vencer la penalización por alejarse no tenga algún incentivo para ir mejorando.

    - Ajustar los hiperparámetros del modelo.

- Validar el modelo POMDP, para entender qué tan bueno es el modelo para resolver el problema, adicionalmente entender si la señal de recompensas es la correcta o existen otras señales que deben implementarse. La metodología para hacerlo es generar un modelo base, e.g., desarrollar un modelo independiente a aprendizaje por refuerzo, que genere una trayectoria para que la formación de UAV's realice la captura, y con dicha trayectoria comparar las trayectorias generadas por las políticas aprendidas con aprendizaje por refuerzo, analizar las diferencias y proponer nuevas señales de recompensa.

- Probar con otros algoritmos de aprendizaje por refuerzo como HER que aborda el tema de las recompensas esparsas en entornos en los que la exploración no siempre es suficiente para alcanzar el objetivo, Dyna-Q que utiliza la experiencia real para construir y perfeccionar el modelo dinámico[2].

- Agregar al entorno de simulación múltiples objetos para que los UAV's aprendan a no colisionar con ningún objeto en el entorno, así como dotar a los UAV's de un sistema de percepción (e.g., LIDAR, cámara) para identificar los objetos cercanos y así tener información en retroalimentación para evitar dichas colisiones.

- Extender el entrenamiento a algoritmos de aprendizaje multi-agente (MARL), donde no se tenga el esquema líder-seguidor y que cada UAV tenga un modelo de aprendizaje, tal que aprendan a coordinar y cooperar entre ellos para lograr el objetivo.

- Implementar el mejor modelo entrenado en UAV's físicos para validar el rendimiento en la vida real.

- Ayudar en el entrenamiento al utilizar anotaciones de un experto, así como posibles trayectorias que permitan realizar la captura, agilizando el entrenamiento.

- Implementar técnicas de cómputo paralelo para agilizar el entrenamiento.

[2]Actualmente no se encuentra disponible en la librería *Stable Baseline3*

Bibliografía

[1] B. G. Maciel-Pearson, L. Marchegiani, S. Akcay, A. Atapour-Abarghouei, J. Garforth, and T. P. Breckon, "Online deep reinforcement learning for autonomous uav navigation and exploration of outdoor environments," *arXiv preprint arXiv:1912.05684*, 2019.

[2] A. P. Kalidas, C. J. Joshua, A. Q. Md, S. Basheer, S. Mohan, and S. Sakri, "Deep reinforcement learning for vision-based navigation of uavs in avoiding stationary and mobile obstacles," *Drones*, vol. 7, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2504-446X/7/4/245

[3] C. Chronis, G. Anagnostopoulos, E. Politi, A. Garyfallou, I. Varlamis, and G. Dimitrakopoulos, "Path planning of autonomous uavs using reinforcement learning," in *Journal of Physics: Conference Series*, vol. 2526, no. 1. IOP Publishing, 2023, p. 012088.

[4] E. Cetin, C. Barrado, G. Muñoz, M. Macias, and E. Pastor, "Drone navigation and avoidance of obstacles through deep reinforcement learning," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. IEEE, 2019, pp. 1–7.

[5] D. Villota Miranda, M. Gil Martínez, and J. Rico-Azagra, "A3c for drone autonomous driving using airsim," in *XLII Jornadas de Automática*. Universidade da Coruña, Servizo de Publicacións, 2021, pp. 203–209.

[6] Y. Tan, *Handbook of research on design, control, and modeling of swarm robotics*. IGI global, 2015.

[7] X. Xu and Y. Diaz-Mercado, "Swarm herding: A leader-follower framework for multi-robot navigation," *arXiv preprint arXiv:2101.07416*, 2021.

[8] P. G. F. Dias, M. C. Silva, G. P. Rocha Filho, P. A. Vargas, L. P. Cota, and G. Pessin, "Swarm robotics: A perspective on the latest reviewed concepts and applications," *Sensors*, vol. 21, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2062

[9] C. Carbone, O. Garibaldi, and Z. Kurt, "Swarm robotics as a solution to crops inspection for precision agriculture," 2018.

[10] A. R. Cheraghi, S. Shahzad, and K. Graffi, "Past, present, and future of swarm robotics," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2021, pp. 190–233.

[11] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," *Robotica*, vol. 31, no. 3, pp. 345–359, 2013.

[12] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics*, E. Şahin and W. M. Spears, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 10–20.

[13] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, 2020.

[14] A. Jevtić and D. Andina de la Fuente, "Swarm intelligence and its applications in swarm robotics," 2007.

[15] D. Albani, T. Manoni, A. Arik, D. Nardi, and V. Trianni, "Field coverage for weed mapping: toward experiments with a uav swarm," in *Proceedings of the 11th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT 2019)*. Pittsburg, US: EAI, 2019, pp. 1–16. [Online]. Available: http://laral.istc.cnr.it/saga/wp-content/uploads/2019/05/BICT2019.pdf

[16] V. Trianni, J. IJsselmuiden, and R. Haken, "The saga concept: swarm robotics for agricultural applications," Technical Report. 2016. Available online: http://laral. istc. cnr. it/saga ..., Tech. Rep., 2016.

[17] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *Intelligent Technologies for Environmental Monitoring Workshop, IEEE AVSS Conference*, 2017, pp. 1–6. [Online]. Available: http://laral.istc.cnr.it/saga/wp-content/uploads/2018/02/ITEMS2017-ID11.pdf

[18] T. Blender, T. Buchner, B. Fernandez, B. Pichlmaier, and C. Schlegel, "Managing a mobile agricultural robot swarm for a seeding task," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 6879–6886.

[19] A. G. Millard, R. Ravikanna, R. Groß, and D. Chesmore, "Towards a swarm robotic system for autonomous cereal harvesting," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2019, pp. 458–461.

[20] T. Minßen, C. Gaus, L. Urso, S. Hanke, J. Schattenberg, and L. Frerichs, "Robots for plant-specific care operations in arable farming-concept and technological requirements for the operation of robot swarms for plant care tasks," *EFITA/WCCA*, vol. 11, pp. 1–11, 2011.

[21] K. Spanaki, E. Karafili, U. Sivarajah, S. Despoudi, and Z. Irani, "Artificial intelligence and food security: swarm intelligence of agritech drones for smart agrifood operations," *Production Planning & Control*, pp. 1–19, 2021.

[22] P. Garcia-Aunon, J. J. Roldán, and A. Barrientos, "Monitoring traffic in future cities with aerial swarms: Developing and optimizing a behavior-based surveillance algorithm," *Cognitive Systems Research*, vol. 54, pp. 273–286, 2019.

[23] A. L. Alfeo, M. G. Cimino, and G. Vaglini, "Enhancing biologically inspired swarm behavior: Metaheuristics to foster the optimization of uavs coordination in target search," *Computers & Operations Research*, vol. 110, pp. 34–47, 2019.

[24] I. Lončar, A. Babić, B. Arbanas, G. Vasiljević, T. Petrović, S. Bogdan, and N. Mišković, "A heterogeneous robotic swarm for long-term monitoring of marine environments," *Applied Sciences*, vol. 9, no. 7, p. 1388, 2019.

[25] N. D. Griffiths Sànchez, P. A. Vargas, and M. S. Couceiro, "A darwinian swarm robotics strategy applied to underwater exploration," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–6.

[26] M. S. Innocente and P. Grasso, "Self-organising swarms of firefighting drones: Harnessing the power of collective intelligence in decentralised multi-robot systems," *Journal of Computational Science*, vol. 34, pp. 80–101, 2019.

[27] M. A. Limeira, L. Piardi, V. C. Kalempa, A. S. de Oliveira, and P. Leitão, "Wsbot: A tiny, low-cost swarm robot for experimentation on industry 4.0," in *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, 2019, pp. 293–298.

[28] Y. Liu, L. Wang, H. Huang, M. Liu, and C.-z. Xu, "A novel swarm robot simulation platform for warehousing logistics," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017, pp. 2669–2674.

[29] J. L. Farrugia and S. G. Fabri, "Swarm robotics for object transportation," in *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018, pp. 353–358.

[30] Pipebots. (2022) Pipebots. Último acceso: 2 de julio de 2022. [Online]. Available: https://pipebots.ac.uk/

[31] M. Kayser, L. Cai, C. Bader, S. Falcone, N. Inglessis, B. Darweesh, J. Costa, and N. Oxman, "Fiberbots: design and digital fabrication of tubular structures using robot swarms," in *Robotic Fabrication in Architecture, Art and Design*. Springer, 2018, pp. 285–296.

[32] N. Melenbrink and J. Werfel, "Local force cues for strength and stability in a distributed robotic construction system," *Swarm Intelligence*, vol. 12, no. 2, pp. 129–153, 2018.

[33] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[34] A. S. Mohamed Thaheer and N. A. Ismail, "Mission design and analysis of usm high-altitude balloon," *Journal of Mechanical Engineering (JMechE)*, vol. 14, no. 2, pp. 62–92, 2017.

[35] A. Friedli, "Steering an autonomous weather balloon," Master's thesis, Swiss Federal Institute of Technology Zurich, ETH, 2013.

[36] S. CAT. (2021) Hasp 2021 (high altitude student platform). Último acceso: 14 de septiembre de 2021. [Online]. Available: https://stratocat.com.ar/fichas/2021/FSU-20210914.htm

[37] J.-B. Renard, G. Berthet, V. Catoire, N. Huret, D. Lagoutte, F. Lefeuvre, J.-L. Pinçon, C. Robert, M. Tagger, E. Seran *et al.*, "Cobrat project: Long duration balloons for the study of high energy phenomena and consequence for stratospheric chemistry," in *19th ESA Symposium on European Rocket and Balloon Programmes and Related Research*, 2009, pp. 181–186.

[38] L. Nagpal and K. Samdani, "Project loon: Innovating the connectivity worldwide," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2017, pp. 1778–1784.

[39] S. Benjamin, "Exploration to exploitation: An industry analysis of suborbital space tourism," *New Space*, vol. 6, no. 1, pp. 87–98, 2018.

[40] J. M. López Urdiales, I. Peris Marti, and G. Girard, "Bloostar, the enabler for more efficient satellites in low earth orbit," in *2018 SpaceOps Conference*, 2018, p. 2468.

[41] M. Holzschuh, "Improving high altitude balloon payload recovery by developing an autonomous parafoil steering system," Universität der Bundeswehr Hamburg, Fakultät für Maschinenbau Professur für Mechatronik Holstenhofweg 85 22043 Hamburg, Germany, 2022-08. [Online]. Available: http://hdl.handle.net/10945/70551

[42] A. Kräuchi, R. Philipona, G. Romanens, D. F. Hurst, E. G. Hall, and A. F. Jordan, "Controlled weather balloon ascents and descents for atmospheric research and climate monitoring," *Atmospheric measurement techniques*, vol. 9, no. 3, pp. 929–938, 2016.

[43] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[44] K. Kersandt, "Deep reinforcement learning as control method for autonomous uavs," Master's thesis, Universitat Politècnica de Catalunya, 2018.

[45] F. Carton, "Exploration of reinforcement learning algorithms for autonomous vehicle visual perception and control," Ph.D. dissertation, Institut polytechnique de Paris, 2021.

[46] N. M. Ashraf, R. R. Mostafa, R. H. Sakr, and M. Rashad, "A state-of-the-art review of deep reinforcement learning techniques for real-time strategy games," *Applications of Artificial Intelligence in Business, Education and Healthcare*, pp. 285–307, 2021.

[47] E. Morales, "Aprendizaje por refuerzo profundo."

[48] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[49] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 2, pp. 201–212, 2011.

[50] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[51] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, 2010.

[52] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[53] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[54] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[55] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[56] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[57] M. S. Ausin, "Introducción al aprendizaje por refuerzo. parte 5: políticas de gradiente," Medium, 2020, Último acceso: 5 de noviembre de 2023. [Online]. Available: https://markelsanz14.medium.com/introducci%C%B3n-al-aprendizaje-por-refuerzo-parte-5-pol%C3%ADticas-de-gradiente-8e92725e9c8f

[58] D. Silver, "Lectures on reinforcement learning," https://www.davidsilver.uk/teaching/, 2015, Último acceso: 5 de noviembre de 2022.

[59] P. W. van Heeswijk, "Policy gradients in reinforcement learning explained," Medium, 2022, Último acceso: 5 de noviembre de 2023. [Online]. Available: https://towardsdatascience.com/policy-gradients-in-reinforcement-learning-explained-ecec7df94245

[60] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[61] S. Levine, "Deep reinforcement learning," https://rail.eecs.berkeley.edu/deeprlcourse-fa17/, 2017, Último acceso: 5 de noviembre de 2023.

[62] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.

[63] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[64] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[65] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*. PMLR, 2016, pp. 1329–1338.

[66] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," *arXiv preprint arXiv:1512.04455*, 2015.

[67] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," *arXiv preprint arXiv:1804.08617*, 2018.

[68] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[69] Sciforce, "Reinforcement learning and asynchronous actor-critic agent (a3c) algorithm, explained," Medium, 2021, Último acceso: 10 de noviembre de 2023. [Online]. Available: https://medium.com/sciforce/reinforcement-learning-and-asynchronous-actor-critic-agent-a3c-algorithm-/explained-f0f3146a14ab

[70] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[71] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *International conference on machine learning*. PMLR, 2018, pp. 1407–1416.

[72] P. W. van Heeswijk, "Proximal policy optimization (ppo) explained," Medium, 2022, Último acceso: 5 de noviembre de 2023. [Online]. Available: https://medium.com/towards-data-science/proximal-policy-optimization-ppo-explained-abed1952457b

[73] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[74] E. Escribá Pina, "Aprendizaje por refuerzo mediante deep learning para las ciudades inteligentes," Ph.D. dissertation, ETSI_Informatica, 2021.

[75] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *Advances in neural information processing systems*, vol. 30, 2017.

[76] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[77] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International conference on machine learning*. PMLR, 2017, pp. 1352–1361.

[78] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[79] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans, "Understanding the impact of entropy on policy optimization," in *International conference on machine learning*. PMLR, 2019, pp. 151–160.

[80] (2023) Deep reinforcement learning. nanodegree program. Udacity. Último acceso: 5 de noviembre de 2023. [Online]. Available: https://www.udacity.com/course/deep-reinforcement-learning-nanodegree

[81] J. Hui, "Rl — reinforcement learning algorithms quick overview," Medium, 2018, Último acceso: 5 de noviembre de 2023. [Online]. Available: https://jonathan-hui.medium.com/rl-reinforcement-learning-algorithms-quick-overview-6bf69736694d

[82] S. Causevic, "A structural overview of reinforcement learning algorithms," Medium, 2020, Último acceso: 10 de noviembre de 2023. [Online]. Available: https://towardsdatascience.com/an-overview-of-classic-reinforcement-learning-algorithms-part-1-f79c8b87e5af

[83] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[84] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 595–625, 2016.

[85] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[86] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 621–635.

[87] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proceedings of the 2020 Conference on Robot Learning*, 2021, pp. 1147–1157.

[88] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[89] ——, "Aerial Informatics and Robotics platform," Microsoft Research, Tech. Rep. MSR-TR-2017-9, 2017.

[90] K. C. Goh, R. B. Ng, Y.-K. Wong, N. J. Ho, and M. C. Chua, "Aerial filming with synchronized drones using reinforcement learning," *Multimedia Tools and Applications*, vol. 80, no. 12, pp. 18 125–18 150, 2021.

[91] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2992–2997.

[92] J. Jackson, G. Ellingson, and T. McLain, "Rosflight: A lightweight, inexpensive mav research and development tool," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 758–762.

[93] AirSim. (2021) Reinforcement learning in airsim. Último acceso: 1 de enero de 2021. [Online]. Available: https://microsoft.github.io/AirSim/reinforcement_learning/

[94] meteblue. Último acceso: 15 de septiembre de 2022. [Online]. Available: https://content.meteoblue.com/en

[95] S. de Información y Visualización de Estaciones Automáticas (SIVEA). Gobierno de México. Estaciones meteorológicas automáticas (emas). Último acceso: 20 de septiembre de 2022. [Online]. Available: https://smn.conagua.gob.mx/es/observando-el-tiempo/estaciones-meteorologicas-automaticas-ema-s

[96] U. of Cambridge. Cambridge university spaceflight landing prediction (cusf). Último acceso: 30 de septiembre de 2022. [Online]. Available: https://predict.sondehub.org/

[97] S. de Relaciones Exteriores (SRE). Gobierno de México. Requisitos para el ingreso de drones a méxico. Último acceso: 15 de septiembre de 2022. [Online]. Available: https://embamex.sre.gob.mx

[98] StratoFlights. (2023) Conjunto de globo estratosférico. Último acceso: 26 de diciembre de 2023. [Online]. Available: https://www.stratoflights.com/es/shop/conjunto-globo-estratosferico/

[99] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[100] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[101] B. Popkin, *An overview on the nutrition transition and its health: The bellagio meeting*. Pub. Health Nutrition, 2002.

[102] J. Bruinsma, *World agriculture: towards 2015/2030: an FAO perspective*. Routledge, 2017.

[103] P. Niloofar, S. Lazarova-Molnar, D. P. Francis, A. Vulpe, G. Suciu, and M. Balanescu, "Modeling and simulation for decision support in precision livestock farming," in *2020 Winter Simulation Conference (WSC)*, 2020, pp. 2601–2612.

[104] T. Van Hertem, L. Rooijakkers, D. Berckmans, A. P. Fernández, T. Norton, and E. Vranken, "Appropriate data visualisation is key to precision livestock farming acceptance," *Computers and electronics in agriculture*, vol. 138, pp. 1–10, 2017.

[105] P. F. Rubiano Pérez, E. Nieves Olmos, and A. D. Gregoryc Tatis, "Monitoreo de ganado con drones," *Encuentro Internacional de Educación en Ingeniería*, ago. 2019. [Online]. Available: https://acofipapers.org/index.php/eiei/article/view/241

[106] Martín Lima. (2020) Drones y vacas lecheras: manejo del ganado. Agro Avisos. Fuente: Tennessee State University. [Online]. Available: https://www.agroavisos.net/2020/07/27/drones-y-vacas-lecheras-manejo-del-ganado/

[107] X. Li and L. Xing, "Use of unmanned aerial vehicles for livestock monitoring based on streaming k-means clustering," *Ifac-Papersonline*, vol. 52, no. 30, pp. 324–329, 2019.

[108] A. Borshchev, "The big book of simulation modeling: Multimethod modeling with anylogic 6," 2013.

[109] S. Jung and K. B. Ariyur, "Strategic cattle roundup using multiple quadrotor uavs," *International Journal of Aeronautical and Space Sciences*, vol. 18, no. 2, pp. 315–326, 2017.

[110] X. Li, H. Huang, A. V. Savkin, and J. Zhang, "Robotic herding of farm animals using a network of barking aerial drones," *Drones*, vol. 6, no. 2, p. 29, 2022.

[111] G. Abdulai, M. Sama, and J. Jackson, "A preliminary study of the physiological and behavioral response of beef cattle to unmanned aerial vehicles (uavs)," *Applied Animal Behaviour Science*, vol. 241, p. 105355, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168159121001428

[112] J. G. A. Barbedo, L. V. Koenigkan, T. T. Santos, and P. M. Santos, "A study on the detection of cattle in uav images using deep learning," *Sensors*, vol. 19, no. 24, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/24/5436

[113] D. C. Bonilla Verdugo *et al.*, "Drones como apoyo para la ganadería a gran escala," 2017.

[114] D. I. Gómez Bedoya and R. Castrillón, "Reconocimiento automático de ganado bovino a partir de imágenes aéreas tomadas con drones: Un enfoque exploratorio." 2019.

[115] T. Wang, X. Xu, C. Wang, Z. Li, and D. Li, "From smart farming towards unmanned farms: a new mode of agricultural production," *Agriculture*, vol. 11, no. 2, p. 145, 2021.

[116] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field coverage," *arXiv preprint arXiv:1803.07250*, 2018.

[117] K. E. Jablonski, R. B. Boone, and P. J. Meiman, "An agent-based model of cattle grazing toxic geyer's larkspur," *PloS one*, vol. 13, no. 3, p. e0194450, 2018.

[118] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[119] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[120] S. D. Levy, "Robustness through simplicity: A minimalist gateway to neurorobotic flight," *Frontiers in Neurorobotics*, vol. 14, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2020.00016

[121] J. F. Hernandez-Garcia and R. S. Sutton, "Understanding multi-step deep reinforcement learning: A systematic study of the dqn target," *arXiv preprint arXiv:1901.07510*, 2019.