Trabajo Fin de Grado
Ingeniería de las Tecnologías Industriales

# Aplicación de algoritmos Deep Policy Iteration en vehículos acuáticos para el patrullaje autónomo de escenarios medioambientales.

Autor: Jose Maria Jurado Polvillo
Tutor: Daniel Gutiérrez Reina
Samuel Yanes Luis

**Dpto. Ingeniería Electrónica**
**Escuela Técnica Superior de Ingeniería**
**Universidad de Sevilla**

Sevilla, 2022

ACE-TI
Grupo de investigación
Ingeniería Electrónica

Trabajo Fin de Grado
Ingeniería de las Tecnologías Industriales

# Aplicación de algoritmos Deep Policy Iteration en vehículos acuáticos para el patrullaje autónomo de escenarios medioambientales.


Autor:
Jose Maria Jurado Polvillo

Tutor:
Daniel Gutiérrez Reina
Samuel Yanes Luis

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla



Sevilla, 2022

Proyecto Fin de Carrera: Aplicación de algoritmos Deep Policy Iteration en vehículos acuáticos para el patrullaje autónomo de escenarios medioambientales.

Autor: Jose Maria Jurado Polvillo

Tutor: Daniel Gutiérrez Reina

Samuel Yanes Luis

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El secretario del Tribunal

*A mi familia*

*A mis maestros*

*A los que están y estuvieron.*

# Agradecimientos

Este trabajo ha supuesto para mí un reto personal ya que el mundo del aprendizaje automático era casi completamente desconocido para mí. Cuando conocí el proyecto que había presente me fascinó la idea de que una investigación que se desarrollaba aquí en Sevilla tuviera una implantación en Paraguay, a más de 9000 kilómetros, y una aplicación tan importante en el control de la salud de un ecosistema en peligro.

Este proceso me ha ayudado a crecer internamente, gestionando mejor mi tiempo, dar mis primeros pasos en el lenguaje Python y fascinarme con las posibilidades que posee. Quiero agradecer a Daniel Gutiérrez y Samuel Yanes haberme brindado la oportunidad de participar en este proyecto tan bonito e interesante, el haberme guiado y acompañado en mis primeros pasos en el campo del Machine Learning y de su constante buena predisposición en ello.

A mi constante compañera de biblioteca Marisol y amigos que siempre han sido para mí una fuente de motivación y alegría junto con mis compañeros de trabajo. No hubiera podido desarrollar este trabajo sin todos ellos.

Quienes siempre me acompañar desde donde estén saben que este proyecto también es de ellos.

*Jose Maria Jurado Polvillo*

*Escuela Técnica Superior de Ingeniería de Sevilla*

*Sevilla, 2022*

# Resumen

En este trabajo se ha realizado un estudio de la implantación de la metodología *Deep Reinforcement Learning*, específicamente algoritmos *policy gradient* "*on policy*", en la resolución de un problema de explotación y toma de muestras en un entorno acuático definido como emplazamiento de experimentación. Se han aplicado algoritmos de esta familia como el REINFORCE y el PPO, donde tras realizar un proceso de sintonización de sus parámetros se ha extraído los modelos obtenidos en cada caso para su posterior simulación. Se han comprobado las planificaciones de ruta desarrolladas por cada algoritmo, así como una comparativa con otras técnicas más usualmente empleadas en navegación automática autónoma.

Este estudio trata de continuar una investigación presente de la Escuela Técnica Superior de Ingeniería de Sevilla para la evaluación y monitorización de los parámetros que caracterizan el estado biológico del lago Ypacaraí en la que se plantea el uso de técnicas de aprendizaje por refuerzo "*off_policy"*. Mediante el uso de un entorno que recrea las características físicas del medio real se aplicarán las técnicas comentadas para garantizar una cobertura, navegación y recogida de datos autónoma optimizadas.

# Abstract

In this work we have carried out a study of the implementation of the Deep Reinforcement Learning methodology, specifically "on-policy" policy gradient algorithms, in the resolution of an exploitation and sampling problem in an aquatic environment defined as an experimental site. Algorithms from this family, such as REINFORCE and PPO, have been applied, where, after a process of tuning their parameters, the models obtained in each case have been extracted for subsequent simulation. The route planning developed by each algorithm has been checked, as well as a comparison with other techniques more commonly used in autonomous automatic navigation.

This study is a continuation of a current research project of the *Escuela Técnica Superior de Ingeniería de Sevilla* for the evaluation and monitoring of the parameters that characterize the biological state of Lake *Ypacaraí*, in which the use of "off-policy" reinforcement learning techniques is proposed. Using an environment that recreates the physical characteristics of the real environment, the techniques will be applied to guarantee optimized coverage, navigation and autonomous data collection.

# Índice

# 1 INTRODUCCIÓN

*"Una hipótesis puede ser fructífera, no sólo para sus proponentes si no, aún más, para conducir a otros nuevos avances."*

*- William Ian Beardmore Beveridge-*

## 1.1. Antecedentes

El análisis de parámetros que pueden regir la calidad del agua y por tanto del estado natural del medio acuático es una tarea de vital importancia que forma parte en los planes de actuación, control y protección de las instituciones competentes sobre el medio ambiente a nivel global. En la actualidad, donde un crecimiento generalizado de la población mundial y de las actividades industriales se identifica una huella ecológica cuya evolución negativa sobre la naturaleza, existe una corriente de compromiso de preservación cada vez más importante que trata de contrarrestar la acción destructiva que el ser humano ejerce sobre el entorno. Gracias a la acción de asociaciones, movimientos, ONG' y gobiernos comprometidos con el medio ambiente a lo largo de los años se ha apreciado un cambio cultural en la sociedad, que cada vez más parece comprometida a la conservación del medio natural. Han aparecido muchas campañas de concienciación (campaña S.O.S Lago Ypacaraí[1]), marcos regulatorios y legislaciones (como la ley nº 5256/14[2]) con el fin de reducir la acción humana en la naturaleza.

Particularmente, el medio acuático es un medio muy sensible a las actividades humanas. Según organismos representativos, la contaminación acuática debido a contaminantes derivados de la acción del hombre supone un riesgo creciente para la salud humana y la conservación del medio ambiente.

[1] Fuente: https://www.efeverde.com/noticias/paraguay-aecid-rescatar-iconico-lago-ypacarai/
[2] Fuente: https://comunidad.paraguay.gov.py/poder-legislativo

**Figura 1.1:** Vertido de aguas residuales en torrente fluvial. Fuente: https://es.wikipedia.org/wiki/Contaminaci%C3%B3n_h%C3%ADdrica

Estas sustancias contaminantes suponen un riesgo para la especies animales y vegetales que ven como su ecosistema sufre una degradación que en algunos casos supone un peligro de supervivencia para muchas especies.

Algunos de los contaminantes causantes de estos problemas pueden ser catalogados como pesticidas, bacterias, desechos industriales, virus, bacterias, compuestos orgánicos e inorgánicos, etc. Estos agentes provocan efectos negativos como reducción del oxígeno disuelto en el agua, cambios en el PH, aumento del número de partículas en suspensión, proliferación de colonias de bacterias, alteración del estado microbiológico acuático, cambios en la composición salina. Existen muchas más consecuencias negativas que no han sido comentadas, pero lo que queda claro es que existe un reto muy importante que abordar para poder conservar el patrimonio natural.

Existen organismos locales que se encargan de realizar tareas de recogida de muestras y análisis químicos de las mismas para realizar un seguimiento del estado acuático de recursos hídricos y detectar cambios bruscos o accidentes puntuales.

**Figura 1.2:** Ejemplo de toma de muestras para su análisis químico. Fuente: https://blog.fibrasynormasdecolombia.com/muestreo-de-aguas-residuales-y-tipos-de-analisis-empleados/

En Paraguay, más concretamente entre las ciudades de Ypacaraí, Areguá, existe un cuerpo de agua llamado Ypacaraí. Este lago se encuentra rodeado de una orografía irregular compuesta por varios cuerpos montañosos y posee una superficie de unos 60km$^2$ con una profundidad superior a 2,5m de media en su superficie. Históricamente ese lago ha sido un lugar con una importancia cultural y turística muy destacable. Para la población local es muy valorado, ya que usualmente pueden realizarse muchas actividades recreativas tanto en sus aguas como en las infraestructuras colindantes. En 2020, sufre un descenso pronunciado en las lluvias y en el caudal de sus afluentes, por lo que su masa acuática descendió pronunciadamente y se produjeron problemas asociados como el aumento de concentración de microorganismos y como consecuencia olores

muy desagradables para la población local.

**Figura 1.3:** Imagen aérea del lago Ypacaraí. Fuente: https://media.ultimahora.com/p/a28c3e1019ea4247f2b42d8f04088cae/adjuntos/161/imagenes/009/176/0009176370/playa-ypacarai.png

Alrededor de este lago existe un tejido industrial desarrollado por el que se produce un vertido de desechos proveniente en gran medida de una actividad agroganadera. La problemática existente en el lago Ypacaraí consiste en la gran cantidad presente de sustancias catalogadas como nutrientes (nitratos, nitritos, fosforo, amonios, DBO…) provenientes de las aguas residuales de los núcleos de población cercanos. Este problema ha llegado a un punto crítico en varias ocasiones en los últimos 30 años alcanzado cotas en las que se ha dictado un cierre de acceso a sus aguas por riesgo biológico. Sobre 2017 se activaron medidas para la protección del lago y su conservación provocando que la calidad del agua aumentara durante unos años, pero actualmente se encuentra en una situación muy desfavorable por lo que se supone la implantación de nuevas medidas de actuación.

La proliferación de cianobacterias, residuos sólidos y el exceso de nutrientes suponen un riesgo directo a la fauna local y a los lugareños, ya que se han registrado problemas de eutrofización de gran importancia en sus aguas.

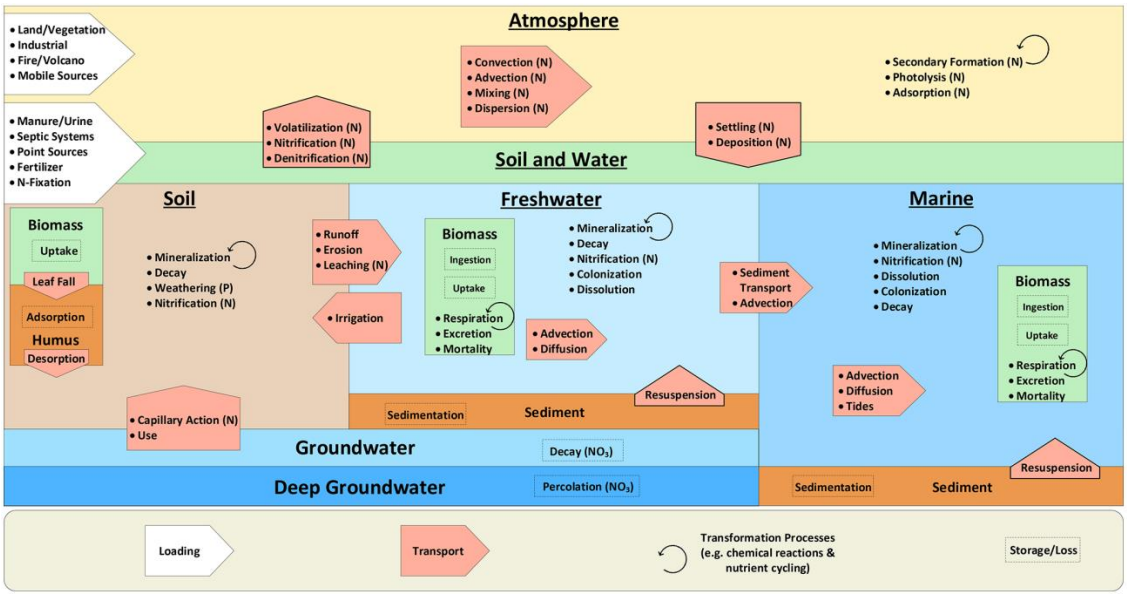



**Figura 1.4:** Ciclo natural de un ecosistema acuático. Fuente: https://pubmed.ncbi.nlm.nih.gov/30036050/

El proceso de eutrofización comienza cuando en un ecosistema equilibrado se produce un incremento descontrolado de nutrientes, normalmente a causa de los nitratos y fosfatos provenientes de diversas fuentes. Como consecuencia del ciclo de natural de descomposición de nutrientes y el crecimiento de la colonia de microorganismos y algas se reduce el oxígeno disuelto en la masa acuática y la aparición de sustancias toxicas para la fauna. Es un proceso en el que también se genera materia sedimentaria en los fondos y en suspensión pudiendo afectar al coeficiente de atenuación difusa del agua.

El efecto de las cianobacterias es la aparición y crecimiento de unas algas (CHAB) que durante su ciclo de vida son productoras de sustancias toxicas y dañinas para la vida natural y el ser humano. Son un problema global dado a que su presencia ha sido detectada en casi todas las superficies acuáticas de agua dulce. Cuando su concentración es notable, dicha agua deja de ser potable y por lo tanto el medio acuático deja de ser adecuado para el uso recreativo para la sociedad [1].

**Figura 1.5:** Efecto del crecimiento descontrolado de algas y colonias de cianobacterias. Fuente: http://climate.org/algae-cyanobacteria-blooms-and-climate-change/

Han surgido un número importante de investigaciones con el objetivo de estudiar la situación y proponer soluciones a esta problemática [2] [3]. Una línea de investigación muy interesante ha sido el empleo de vehículo de superficie no tripulados para la recogida de muestras de parámetros clave de manera autónoma [4] [5] [6]. El uso de este tipo de tecnologías supone una reducción de la interacción directa con el medio pudiendo evitar el posible contagio humano de sustancias nocivas y aumentar la eficiencia de estas tareas. Otro aspecto muy interesante resultante de estos proyectos sería el análisis de parámetros de manera remota e instantánea obteniendo estos resultados en tiempo real concluyendo en una monitorización del estado acuático de la zona.

**Figura 1.6:** Vista del entorno natural del lago Ypacaraí. Fuente: https://www.retema.es/noticia/drones-para-monitorizar-el-nivel-de-contaminacion-del-lago-ypacarai-en-paraguay-d6Kvv

Se han realizado propuestas técnicas muy interesantes para la aplicación de tecnologías relacionadas con el *Machine Learning* o inteligencia artificial para el estudio y optimización del proceso de recogida de muestras de este recurso hídrico [4]. Más específicamente el uso de algoritmos adaptativos de aprendizaje por refuerzo supone una línea activa y adecuada para el medio dada sus características dinámicas y naturales que parece abierta a un proceso de optimización y ejecución posterior en el emplazamiento.

El enfoque a abordar el problema de exploración del lago Ypacaraí ha supuesto una estrecha relación entre las autoridades paraguayas y la Agencia Española de Cooperación Internacional para el Desarrollo (AECID), a través de la cual se planea la actividad de monitorización del estado del lago en tiempo real mediante el uso de embarcaciones ASV (*automatic surface vehicle*).

Actualmente se está trabajando en la puesta a punto de un nuevo modelo ASV. Este modelo está realizando pruebas de monitorización en unas instalaciones de Sevilla en un entorno controlado, para optimizar el guiado autónomo del mismo teniendo en cuenta el escenario en el que trabaja y las comunicaciones intrínsecas del mismo a través de distintas tecnologías. Otro aspecto muy interesante que implementa este modelo es el guiado y localización a través de tecnología GPS con el fin de alcanzar una eficiencia y autonomía completa.


**Figura 1.7:** Vehículo ASV en entorno de pruebas, Sevilla

Este modelo consta de un conjunto de sensores que permite analizar parámetros destacables del agua (DBO, temperatura del fluido y ambiente, PH, concentración de algas y agentes nocivos) así como la recogida de muestras para su posterior estudio.

El sistema de control de estos vehículos consta de un módulo de control dinámico realimentado que trabaja en bajo nivel recibiendo las órdenes del generador de trayectorias encargado de seleccionar el camino óptimo a partir del estado que se actualiza a partir del GPS. El campo de trabajo en que se encuentra el grupo de investigadores encargado del proyecto sería que desarrollar un algoritmo de exploración o cobertura del mapa que sea lo más adecuado posible a las características del vehículo y del entorno del trabajo.

Se han implementado algoritmos de aprendizaje por refuerzo profundo tipo *off-policy* con unos resultados muy prometedores [7] [6] . La versatilidad de esta familia de algoritmos permite que mediante esta metodología se pueda abordar eficientemente el problema de exploración y recogida de muestras [5]. Tras una investigación al respecto se ha conseguido demostrar que abordar esta tarea con una automatización gobernada por inteligencia

artificial es más eficiente que otras técnicas más arraigadas en la robótica clásica, como algoritmos de exploración tipo cortacésped o expansivos que tratan de recorrer todo el espacio de la superficie de trabajo.


**Figura 1.8:** Vehículo ASV en fase de pruebas de la universidad de Sevilla. Fuente: https://www.us.es/actualidad-de-la-us/la-us-lidera-un-proyecto-con-drones-para-mejorar-la-gestion-ambiental-del-lago#lg=1&slide=1

El objetivo de esta investigación es la puesta en marcha de una flota de ASV en el entorno acuático Ypacaraí para el control y monitorización del impacto ambiental que tiene lugar en sus aguas.

**Figura 1.9:** Desarrollo de experimentación de flota en un entorno controlado.

## 1.2. Objetivos y justificación

Este trabajo pretende continuar una investigación sobre la aplicación de diversas técnicas de aprendizaje por refuerzo sobre el problema de exploración y toma de datos en el lago Ypacaraí. La idea principal es aplicar una serie de técnicas distintas a las estudiadas anteriormente para comprobar se puede conseguir una estrategia más eficiente a la hora de resolver este reto de exploración autónoma.

Lo que se pretende conseguir es una estrategia óptima para el problema de monitorización que se quiere resolver del vehículo en el entorno lo más fiel a la realidad posible mediante técnicas de *Deep Reinforcement Learning* con algoritmos tipo *on-policy*. Para ello, será necesario el diseño de un simulador que represente las características principales del lago como dimensiones, obstáculos, etc., y del vehículo que recoja todo el comportamiento posible de este.

El modelo desarrollado trabajaría como generador de rutas: tras recibir el mapa del entorno de trabajo particularizado en un formato adecuado será el encargado de generar una ruta a seguir por el vehículo para la recolección de medidas e información en el lago teniendo en cuenta la existencia de zonas con más o menos interés de visita. Debemos tener en cuenta esta cuestión ya que es importante tener en cuenta que un lago no es un ecosistema estático y en algunas zonas presentará condiciones más favorables para la sedimentación de nutrientes o la reproducción de organismos nocivos.

El objetivo de este trabajo consiste en la obtención de rutas óptimas que debe realizar un vehículo en la tarea de exploración y explotación en un entorno definido con anterioridad. Adicionalmente se pretende realizar:

- Revisar el estado actual del uso de técnicas de aprendizaje automático para tareas de autónomas de exploración.
- Obtención de un algoritmo adecuado para abordar eficientemente la exploración autónoma de un recurso hídrico previamente mapeado.
- Observar cómo se adaptan las técnicas empleadas para la búsqueda de una solución al problema planteado.
- Presentación de resultados y consideración de los aspectos principales obtenidos tras el proceso de experimentación.

Aspectos importantes a tener en cuenta en la elaboración de un algoritmo eficiente:

- Se propondrá un algoritmo parametrizable dependiendo de las necesidades de la monitorización.
- Se adaptará a las dimensiones del escenario, acumulando una penalización cuando el vehículo exceda de estas. Por supuesto, deben impedirse realizar desplazamientos a lugares que sean imposibles de alcanzar o que se dispongan imposibles de visitar.
- El algoritmo debe adaptarse a una regeneración a lo largo del tiempo de la importancia de la medida en los lugares visitados.
- Se debe premiar la toma de muestras a lo largo del recurso hídrico en función de la importancia de la zona a analizar y tratar de realizar una cobertura zonal lo más amplia posible.

Aunque no se ha tomado como premisa fundamental, se pretende realizar un algoritmo que sea adaptable al trabajo en diversos entornos acuáticos. Por ello se ha valorado la capacidad de adaptación a una modificación del entorno de trabajo respecto a la empleada inicialmente como entorno de trabajo. Se realizarán evaluaciones de los modelos obtenidos con unas condiciones específicas en otras distintas para estudiar su comportamiento.

## 1.3. Motivación

La automatización puede entenderse como el estudio y la implementación de una serie de técnicas y herramientas en un proceso que tiene como objetivo la realización de una tarea de manera autónoma y eficaz.

A lo largo de la historia, el desarrollo de la automatización en la industria ha sufrido una gran transformación llena de avances y mejoras tecnológicas que ha supuesto la implementación de sistemas cada vez más complejos. En sí mismo la industria ha sido impulsada por ella misma y ha supuesto el origen de grandes revoluciones y puntos de inflexión muy importantes.

En la actualidad se aprecia un aumento del uso de procesos automatizados en muchísimos campos de aplicación y pienso que aun teniendo en cuenta la gran cantidad de escenarios en los que tiene lugar esta tecnología, veremos que aún quedan por venir más avances importantes en esta área de la ingeniería.

Si observamos varios ámbitos de aplicación que consideramos importantes relacionados con este proyecto:

- **Recogida de datos sin interacción humana**

  Para la implantación de un sistema automatizado son necesarios la recogida de datos relativos al entorno donde se plantea actuación del proceso a desarrollar. Será de vital importancia el entendimiento del escenario a partir de sus características, ya que supondrán una fuente de información acerca de las limitaciones y especificaciones técnicas que rigen el comportamiento y la evolución tanto del entorno como del agente durante la ejecución de la tarea programada.

- **Procesamiento de datos y obtención de resultados**

  Otra característica muy importante en la realización de un sistema automatizado será el procesamiento de esos datos previamente recogidos y la adecuación de estos para un correcto aprovechamiento de la información que contengan. Este es un aspecto muy importante en el modelado de sistemas, ya que se trata de una operación muy delicada en la que un error ínfimo en los datos usados para su modelaje puede acarrear la obtención de un modelo no adecuado o erróneo. Por ello esta etapa en muy importante y es muy común que se realicen tareas de filtrado y corrección de datos antes de su uso. Por ejemplo, una técnica de programación como *Machine Learning* basa sus buenos resultados en gran medida al correcto procesamiento de datos para la obtención de buenos resultados.

- **Ámbito de actuación de vehículos no tripulados específicamente en ambientes náuticos.**

Un ámbito de aplicación de la automatización que ha supuesto una gran evolución en la industria y que aún parece que estará muy presente en el futuro ha sido la realización de tareas mediante dispositivos preprogramados como vehículos o robots. Eso ha supuesto, gracias a la participación de estos, que la eficacia de fabricación de manufacturas debido al aumento de la calidad y velocidad de fabricación. Un ejemplo de ello puede ser las máquinas de control numérico, robots articulados y los vehículos no tripulados.

Los vehículos no tripulados tuvieron su origen en investigaciones militares como suele suceder con la gran mayoría de los avances científicos. En 1916 fue desarrollado un vehículo aéreo que podía ser pilotado mediante ondas de radio, lo que supuso un avance revolucionario[3]. Desde entonces se han dado lugar en gran número y variedad diversos tipos de vehículos que no necesitasen directamente de interacción humana para su control.

Si se realiza una clasificación de ellos teniendo en cuenta las características del entorno donde desarrollan su función se pueden observar 3 tipos de vehículos:

UGV (*unmanned ground vehicle*), UAV (*unmanned aerial vehicle*) y USV (*unmanned surface vehicle*)

Este grupo de vehículos denominados como USV también son conocidos por las siglas ASV (*autonomous surface vehicle*), que es como será referido a lo largo de este documento.

Los ASV están presente en multitud de escenarios, investigaciones y son empleados para una gran variedad de tareas. La funcionalidad y posibilidades que ofrecen son impresionantes, ya que normalmente los entornos acuáticos tienen asociados una dificultad de acceso y operación. Actualmente existen dispositivos que cuentan con una autonomía de incluso tres o cuatro meses aprovechando fuentes de energía como la solar y la cinética que pueden aprovechar de las olas cuando trabajan en el mar [8].

**Figura 1.10:** Vehículo no tripulado de superficie Mahi Two.

Fuente: https://www.electricvehiclesresearch.com/articles/26394/atlantic-ocean-crossing-by-a-solar-electric-autonomous-vessel

Por ello muchas tareas de exploración, recogidas de datos, vigilancia y reconocimiento son realizadas por estos vehículos ya que ofrecen muchos beneficios respecto a otros sistemas de trabajo usados previamente.

Como en otro tipo de dispositivos, las posibilidades de explotación son muy numerosas en función del equipamiento que se le aporte a este dispositivo por lo que teniendo en cuenta la gran variedad de sensores y actuadores que pueden añadidos nos aporta una visión de las ventajas y aplicaciones de este tipo de tecnología y lo interesante que resulta su implantación.

[3] https://www.iwm.org.uk/history/a-brief-history-of-drones

- **Path planning o planificación de rutas en vehículos**

Un aspecto muy importante en la eficacia que ha venido asociada al uso de los vehículos autónomos es la planificación de rutas o *path planning*, ya que el rendimiento del dispositivo vendrá asociado en gran medida a un correcto desplazamiento y realización de tareas en su recorrido.

La planificación de rutas consiste normalmente en la generación de una serie de trayectorias con un grado de optimización que son generadas en un entorno que debe ser conocido en cierta medida.

Hoy en día la implantación en la navegación autónoma está muy desarrollada y el grado de complejidad que ha alcanzado es impresionante, permitiendo que vehículos se adapten automáticamente a entornos en plena circulación incluso, como ejemplo pueden ser los coches autónomos. La eficiencia de estos vehículos no para de incrementarse debido a la importante inversión en desarrollo que realizan muchísimas empresas asociadas por ejemplo al sector automovilístico, al sector aeronáutico, espacial y naval ya que se ha puesto de manifiesto la importancia en el futuro de estos dispositivos.

Lo que está claro es que cada día estamos rodeados más de este tipo de tecnología, el mejor ejemplo podemos observar en cómo se ha introducido en tareas tan rutinarias en cualquier casa como es la limpieza con dispositivos como *Roomba*.

- **Algoritmos**

Entre las técnicas más empleadas en la generación de trayectorias se pueden destacar el modelado del escenario, métodos de grafos, algoritmos de trayectorias aleatorias y algoritmos de trayectorias basados en la probabilidad.

Los algoritmos siempre han sido una gran herramienta para la resolución de problemas planteados. La complejidad de estos siempre vendrá marcada de la mano del grado de dificultad para resolver la tarea a la que se enfrenten. Desde un conjunto de operaciones a un conjunto de datos hasta algoritmos genéticos vemos que están presentes en todos los campos de la ciencia. Parece ser que la investigación y el desarrollo de estos puede seguir marcando los pasos a seguir en el proceso de resolver problemas que la humanidad aun es incapaz de solucionar actualmente. Por lo que estoy seguro de que seguiremos viendo grandes avances tecnológicos gracias a ellos.

- **Machine learning**

El aprendizaje automático o *Machine Learning* se basa en una serie de algoritmos para conseguir que los dispositivos realicen un aprendizaje autónomo en diversos escenarios y tareas propuestas.

Tras importantes avances en la capacidad de cálculo de los ordenadores, la cantidad de información que puede usarse el mismo tiempo que años atrás supone que pueden alcanzarse resultados en un tiempo mucho menor y siendo más adecuados u óptimos.

Es la base de la inteligencia artificial y esta tecnología está implantada en numerosísimos sectores como de la información, militar, económicos, políticos, etc.

Es muy interesante la posibilidad de crear modelos predictivos de casi cualquier sistema y poder predecir su comportamiento en un futuro más o menos lejano en el tiempo.

- **Deep learning**

El aprendizaje profundo o *Deep Learning* consiste en una serie o conjunto de algoritmos que están diseñados para un aprendizaje automatizado.

Un aspecto muy importante de este tipo de arquitecturas es el uso de las redes neuronales, que son muy importantes a la hora de afrontar tareas difíciles de resolver como por ejemplo reconocimientos de imagen (en cadenas de fabricación comprobar si algún producto está mal manufacturado), reconocimiento de voz y escritura, control de tráfico en ciudades inteligentes (*Smart cities*), estudios de toma de decisiones,

modelos de conducta y muchísimos más ámbitos.

Las redes neuronales convolucionales poseen unas capas denominadas *hidden layers* que mediante unos procesos llamados convoluciones en las que se realiza operaciones entre grupos de pixeles con un producto escalar de un *kernel*. Aunque en realidad habrá tantos *kernel* por convolución como filtros o matrices de salida de cada capa o *layer*.

La idea es aplicar varias convoluciones, una tras otra, para conseguir mejores resultados.

Normalmente los algoritmos de las redes neuronales son aplicados en técnicas de lo que se denomina aprendizaje profundo (*Deep Learning*) y por supuesto en el aprendizaje profundo por refuerzo (*Deep Reinforcement Learning*). Esta última es una técnica en la que se busca que el actor vaya aprendiendo automáticamente a tomar decisiones óptimas.

Para ello se usa una red neuronal que es quien define las posibilidades de aprendizaje de este agente. El agente abordará estas decisiones mediante una variable muy importante en el proceso que será la recompensa, proveniente de la ley de recompensa, que marcará la idoneidad de la decisión tomada por él.

Este es un campo de investigación en plena expansión debido a los prometedores resultados ya contrastados en el análisis de datos de diversa índole. Además, se pueden abordar problemas en los que otros métodos de cálculos exactos no pueden conseguir buenos resultados.

Es por ello por lo que me siento muy interesado en abordar un trabajo de fin de grado que incluye todas estas características anteriormente mencionadas. Se trata de un proyecto que anima mucho a profundizar en una rama de conocimiento que era desconocida para mí y tener la posibilidad de enfrentarme a un lenguaje de programación tan completo y que tiene tantas posibilidades como es Python y las librerías empleadas de este.

En este proyecto se realizará un análisis entre varias técnicas de aprendizaje por refuerzo para la resolución de un escenario en el que un ASV trabaje en un medio acuático con la misión de tomar medidas de varios componentes en el agua. Existe un proyecto en paralelo en el que se desarrolla este prototipo físico y me atrae mucho la idea de simular este proceso y comprobar el funcionamiento de la tarea idóneamente. Es un proyecto que incluye muchas ramas ingenieriles de conocimiento como ingeniería de control, de sistemas, hidráulica y electrónica.

Además, el proyecto plantea una futura investigación en la que se planea que se controle una flota de estos vehículos para que trabajen en paralelo.

### 1.4. Estructura del documento

Este documento ha sido redactado con la premisa de seguir una estructuración clara de los conceptos que aparecen en el tratando siempre de introducir al lector de manera rápida a una idea clara de los mismos. Tras una breve explicación de los motivos por los que se ha decidido realizar una experimentación y por lo tanto la redacción de este trabajo, se realizará una descripción del estado actual de la técnica relativa a los puntos que pueden ser referidos o relacionados con los equipos, técnicas o experimentación que engloba este proyecto. Como la complejidad del problema puede ser abrumadora a la hora de identificar elementos, código empleado, bibliotecas, etc., en el apartado definición del problema se plantea una visión global del software empleado, consideraciones oportunas respecto al uso de este para respetar el enfoque del proyecto. Posteriormente se realizará una visión generalizada y discreta de los conceptos clave relacionados con las técnicas empleadas en la resolución del problema abordado con lo que se pretende poco a poco explicar las decisiones tomadas con el fin de obtener unos resultados positivos y coherentes que puedan ser presentados.

# 2 ESTADO DEL ARTE

En este apartado se analizará el estado de la técnica sobre procedimientos y conceptos importantes que están relacionado en ámbito técnico con el alcance de este trabajo.

Se analizarán ejemplos de procesos automatizados de toma de datos mediante vehículos autónomos empleados para ese fin para continuar con una tarea que en muchos casos suele ser implementada a la anterior en muchos casos, el patrullaje automático de vehículos en unos escenarios determinados. Resulta interesante tener una visión sobre esta temática en este estudio teniendo en cuenta el objetivo de este. Para ello será necesario revisar el estado técnico sobre la construcción de entornos de simulación y trabajo, ya que trabajar con un entorno de simulación será un ingrediente básico en la búsqueda de la solución adecuada para el problema de exploración del lago Ypacaraí.

Posteriormente se analiza un aspecto fundamental en la tarea del diseño de vehículos autónomos, la aplicación y diseño de algoritmos de planificación de ruta (*path planning*) para finalizar revisando el estado de los vehículos autónomos de superficie tratando de aproximarnos al objeto de nuestro trabajo.

## 2.1. Vehículos autónomos de superficie (ASV)

Los vehículos autónomos de superficie o ASV's (*autonomous surface vehicles*) son una familia muy extensa de vehículos que poseen unas características particulares que definen tanto en el modo de funcionamiento como en el ecosistema para el cual está diseñado su movimiento operativo. Principalmente como se puede intuir a partir de dominación es una familia de vehículos que están diseñados para trabajar en superficies acuáticas, ya sea en medios fluviales o marinos. Existe una denominación más amplia de todos los vehículos de no tripulados conocida como USV (*unmanned surface vehicle*) que engloba a la categoría que se desarrollará en este apartado.

En un principio estos vehículos fueron diseñados para su uso en investigación y con fines militares. Su desarrollo se plantea como posible solución a la necesidad de tener una mayor capacidad operativa en un entorno no controlado y seguro para la interacción humana como son las superficies fluviales y marítimas. Son vehículos diseñados para operar con gran precisión y sin prácticamente generación de huella ecológica.

El número de tareas que han sido realizadas por esta categoría de vehículos es inmenso si además se observa la gran variedad de escenarios distintos y de naturalezas distintas. Algunos ejemplos innovadores, como la toma de análisis de diversos parámetros del agua automáticamente, realización de mapas topográficos submarinos, tareas de búsqueda y rescate en misiones humanitarias, localización de balizas, observación de fauna [26] etc.

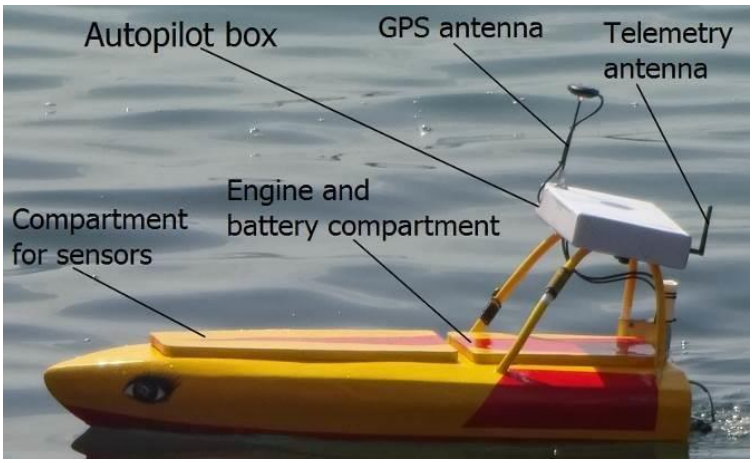



**Figura 2.1:** Vehículo autónomo de superficie ASV y hardware presente en dicho modelo [27].

La evolución que han experimentado los ASV a lo largo de tiempo ha sido proporcional a la de los problemas para los que se han diseñado y la tecnología presente en sus componentes [28]. Actualmente existen modelos que se encuentran en operación ininterrumpida durante meses gracias a células fotovoltaicas, y que están diseñados para evitar colisiones durante su trayecto[4].

Cuando hablamos de un vehículo no tripulado y autónomo es importante hablar de la navegación automática ya que es un aspecto crucial para que el vehículo pueda realizar su operación. El desarrollo de esta navegación es un campo importante cuyo desarrollo se trata de una ardua tarea. La navegación siempre ha sido uno de los aspectos más complicados a tratar en el diseño de un dispositivo autónomo que se desplace. Normalmente el uso de tecnología de geolocalización GPS (*global positioning system*) está presente en estos dispositivos. Mediante el uso del GPS se obtienen datos referidos al posicionamiento, dirección y velocidad muy importantes para la tarea de control de la navegación. Existen problemas asociados que han sido el foco de investigación de muchos grandes profesionales como por ejemplo los problemas de precisión y errores acumulados en la posición [29], perturbaciones, interferencias y problemas relacionados con la precisión de los dispositivos empleados [30]. A lo largo de los años han sido implementadas una gran diversidad de técnicas que han reducido o solucionado las limitaciones técnicas a las que se ha ido enfrentando la implantación de esta tecnología. Un ejemplo de ello ha sido el uso de técnicas diferenciales (*DGPS*) para la corrección de errores entre la posición real y la posición calculada [31]. El DGPS (*differential global positioning system*) consiste en un sistema de posicionamiento global que ofrece información sobre la posición objetivo a partir de los satélites con una corrección implícita y la duración para la que es válida dicha corrección, por lo que la precisión que aporta respecto al anterior GPS es muy interesante.

Junto a un dispositivo de localización, suele equiparse a este tipo de vehículos con diversos tipos de sensores como sensores de profundidad y visión junto con algún tipo de comunicación por radio para recibir y transmitir información.

El modelado de este tipo de embarcaciones suele abordarse con modelos no lineales de al menos segundo grado y de orden mayor a la hora de caracterizar la cinemática.

Para abordar la navegación en el pasado han sido usados métodos como control PID, en multicapas, control predictivo, técnicas de estimación, etc. En los últimos años, una parte de la comunidad científica ha centrado sus esfuerzos en aplicar nuevos métodos como el aprendizaje automático en la gestión del control de navegación de muchos vehículos no tripulados. "*Gracias a los avances en Deep Learning, en sensórica y en los algoritmos ha sido posible la implantación de técnicas de aprendizaje profundo para que problemas como la navegación y evitar obstáculos en entornos dinámicos"* [32].

Los ASV's al ser vehículos acuáticos añaden una complicación más y es que se tratan de entornos dinámicos que suponen un reto abordar. Es por lo que algoritmos como los denominados de aprendizaje por refuerzo obtienen tan buenos resultados, ya que son capaces de trabajar con entornos dinámicos o estáticos y pueden

[4] Fuente: http://www.sailbuoy.no/news-2/64-transatlantic-crossing

evadirse problemas a los que otros métodos anteriores son más sensibles como el ruido, cúmulo de errores, eficiencia y limitaciones de cálculo.

## 2.2. Recogida de datos automática por vehículos

La monitorización de parámetros acuáticos mediante vehículos autónomos es una realidad en una gran cantidad de entornos distintos. Gracias a la evolución tecnológica que ha tenido lugar en campos como la robótica, comunicaciones, sensórica y navegación se han podido asignar tareas de análisis y el seguimiento de parámetros que permiten una monitorización de los niveles de calidad del agua en muchos recursos hídricos.

Existen ejemplos de implantación de estos métodos en la recogida de muestras en puntos de vertidos, como residuales, donde el posible difícil acceso y la exposición a posibles agentes negativos para la salud humana suponen un escenario idóneo para el uso de esta tecnología[5].

La navegación autónoma de vehículos tanto aérea como acuática es un área en constante estudio y progreso debido a que dicha temática aprovecha el impulso de investigaciones relacionadas donde se desarrollan modelos y vehículos empleados militarmente.

**Figura 2.2:** Vehículo no tripulado especializado en la detección y neutralización de minas acuáticas[6].

También es destacable el uso de vehículos acuáticos en el control de la calidad del medio [9] y la participación en tareas de mantenimiento[7] en explotaciones de acuicultura donde es importante un constante estudio de los emplazamientos productivos.

El uso de sensores de última generación como las sondas multiparamétricas permiten la detección de variables como la temperatura, pH, conductividad, densidad, concentración de oxígeno disuelto, presencia de gases disueltos, turbidez, etc. Dicha información procesada adecuadamente se aprovecha para generar modelos predictivos del medio y extraerá datos de los procesos químicos que tienen lugar en el entorno. Si tenemos en cuenta la posibilidad de toma de muestras físicas por los vehículos para un posterior análisis junto a la capacidad de analizar variables importantes automáticamente, como las descritas anteriormente, puede entenderse la tendencia alcista del uso de esta tecnología en tareas automatizadas de análisis de parámetros acuáticos.

Desde la década de 1990 el lago Ypacaraí ha sufrido un proceso de degeneración de la calidad de sus aguas debido a la actividad humana y al crecimiento de la concentración de sustancias negativas para la vida acuática. Por ello se han realizado a lo largo de los años campañas de concienciación, el cierre de actividades ganaderas y la toma de medidas medioambientales para su conservación. En dicho enclave han sido desarrolladas varias investigaciones internacionales donde se han propuesto soluciones a la ineficiencia detectada en la cobertura de la detección de los picos de contaminación que tienen lugar a lo largo de su

[5] https://www.oceanalpha.com/application_cases/hk-water-supplies-department-introduces-unmanned-surface-vehicle-system/
[6] https://www.ecagroup.com/en/solutions/usv-mine-identification-and-neutralization
[7] https://www.marinetechnologynews.com/news/aquaculture-norwegian-researchers-revolutionizing-611137

superficie. Dicho lago ha sufrido altibajos durante varios periodos en los que sus aguas resultan inadecuadas para el recreo lo que pone de manifiesto que el control de los valores contaminantes es ineficaz.

Dichas investigaciones han supuesto un punto de partida para el desarrollo de medidas de actuación orientadas en la detección de estos valores anormales y el patrullaje del entorno [10] [6]. El uso de flotas autónomas supone un recurso muy útil para las autoridades a la hora de tomar medidas que afectan a la conservación de los entornos naturales, teniendo en cuenta además la evolución del pensamiento empático con la naturaleza que está siguiendo la sociedad.

## 2.3. Patrullaje automático en superficies definidas

El patrullaje consiste en la visita periódica de las zonas o regiones de un emplazamiento determinado. Normalmente la idea de patrullaje como tarea en sí suele asociarse con uno o más objetivos adicionales, como puede ser la observación o vigilancia. Es común aplicar técnicas de patrullaje a trabajos en los que se propone controlar el estado de una superficie, alertas y observación de anomalías, en entornos que se quieran monitorizar.

La navegación y la planificación de rutas son elementos clave tanto en el desarrollo como en la explotación de cualquier sistema autónomo. El sistema de navegación está compuesto por los elementos de control necesarios para que los vehículos puedan realizar desplazamientos a lo largo del entorno de trabajo correspondiente con las garantías necesarias para asegurar un correcto funcionamiento del dispositivo en el desempeño de tareas pertinentes de este. Es importante la implantación de sistemas de localización, detección de obstáculos, características del entorno, comunicación en tiempo real y un sistema de control que trabaje con un buen modelo del vehículo y/o entorno para conformar un sistema de navegación adecuado. La arquitectura empleada suele estar centralizada en alto nivel mediante un módulo de control, CPU, que gestiona la información de los sistemas de posicionamiento, condiciones del contorno (como profundidad, obstáculos, velocidad y dirección del viento, giróscopos, …), del sistema de propulsión y comunicaciones.

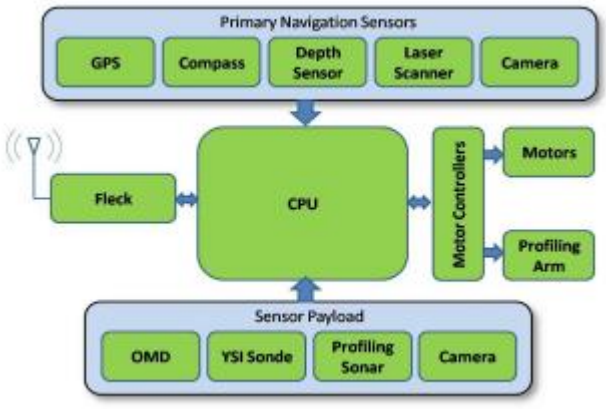



**Figura 2.3:** Arquitectura del sistema de control de ASV[8].

Cuando se planea abordar una tarea de exploración, es común que dicha tarea se defina como un problema de cobertura. Los problemas de cobertura condicionan el ejercicio de resolución a obtener un resultado óptimo tratando de visitar la mayor área posible de la superficie de trabajo, con el agente en cuestión, aprovechando

[8] Fuente: https://www.araa.asn.au/acra/acra2009/papers/pap155s1.pdf

los recursos disponibles, por lo que se suele premiar una conducta de visitas únicas a los puntos presentes en el entorno de explotación. Existen ejemplos de dichos criterios de exploración en la navegación basada en *lawn-mower* (cortacésped), donde dicho comportamiento de navegación resulta muy adecuado para la tarea a desarrollar [11].

**Figura 2.4:** Vehículo autónomo con generador de trayectoria *lawn-mower*[9].

Resulta un planteamiento interesante y adecuado en muchas situaciones, pero existen otros problemas en los que dicha directriz resulta insuficiente o inadecuada, por ejemplo, puede ser más interesante volver a ciertos puntos donde la recompensa por una visita reiterada es mayor que explorar zonas del mapa no visitadas previamente.

La planificación de rutas (path planning) consiste en el diseño de un conjunto de trayectorias a seguir, en este caso, por el vehículo. Si nos emplazamos en el campo de la robótica, el path planning afecta directamente a la planificación del movimiento del móvil. A través de información de la sensórica y entorno se encargará de diseñar un conjunto de trayectorias adecuado para que el desplazamiento a realizar sea adecuado adaptándose a los criterios que se tengan en cuenta en el proceso de decisión.

La evolución en los algoritmos empleados para la resolución de las trayectorias a realizar para la ejecución de un movimiento ha evolucionado juntamente con los avances en el campo de la robótica. Los algoritmos basados en la resolución de grafos buscan minimizar una función objetivo que está sujeta a restricciones propias del sistema. En [12] se evalúan diversos algoritmos, con una complejidad creciente, en la resolución de la navegación autónoma de un vehículo autónomo, y se refleja la evolución temporal de estos algoritmos.

En nuestro caso, trataremos de implementar una planificación de rutas para resolver el patrullaje informativo empleando el aprendizaje por refuerzo, sujeta a algunas restricciones de la masa acuática, donde tendrá bastante importancia estudiar el problema de cobertura y se verá cómo será preferible navegar en ciertas zonas ya transitadas previamente antes que explorar zonas inexploradas.

## 2.4. Caracterización de entornos para estudios de simulación.

Cuando se plantea un problema a resolver o necesidad en los que tiene importancia el entorno de trabajo, optimizaciones de rutas, tareas de exploración, etc., queda claro que si se pretende aplicar un algoritmo ya sea de aprendizaje automático, siguiendo la deriva temática, *Deep Learning* o cualquier otra familia es vital contar con información de ese entorno. Dicha información, estado o evolución del mismo, es otra variable más a optimizar para obtener la solución adecuada a el problema planteado ya que el sistema completo engloba al entorno y a todos los agentes adicionales participantes.

A la hora de abordar tareas de simulación y explotación en un entorno determinado, es muy importante la definición y adecuación de este con las características necesarias. Es importante que el entorno digital diseñado sea fiel a la condición real del modelo que quiere representarse, ya que obviamente la finalidad de esta

[9] https://www.researchgate.net/figure/Our-tractor-lawn-mowing-vehicle-autonomously-cutting-long-grass_fig1_249811362

actividad será el diseño de una experimentación que pueda realizarse en el entorno ficticio para una posterior implementación física adecuada. Por ello se ha visto una evolución en la técnica importante a la hora de crear entornos fieles a la realidad con características y restricciones que comprometen a dichos entornos cumplir ciertas reglas homónimas a las presentes en la realidad.

Si se analizan entornos empleados en investigaciones basadas en la aplicación de técnicas de aprendizaje automático, pueden encontrarse mapas conformados por varias capas de información y diversas dimensiones. En la literatura pueden encontrarse referenciada una problemática asociada al empleo de algoritmos *Deep Reinforcement Learning* con entornos donde se aplica una caracterización elevada de la zona de trabajo desarrollando problemas de eficiencia de cálculos y optimizaciones ineficientes de los gradientes oportunos.

Cuando se empezaron a usar entornos de trabajo para simulaciones la calidad de los mismos podría definirse como simplista, ya que su aspecto podría rememorar a escenarios extraídos de videojuegos o máquinas recreativas. Pero ciertamente la fortaleza de estos surge de este hecho, ya que resultan perfectos gracias a su robustez para el desarrollo y prueba de algoritmos.

Cuando se trabaja en entornos donde no se conocerá con certidumbre el estado actual y la evolución del mismo es importante diseñar un algoritmo de planificación de rutas preparado para gestionar con eficiencia la explotación de zonas conocidas, como la exploración de otras tantas para conseguir la mayor recompensa posible, ya que optimizar los resultados forzando que el agente trabaje en zonas donde se conoce el valor de la cobertura puede alcanzarse un mínimo local en la solución del problema. Por lo tanto, un buen algoritmo de path planning debe introducir una componente de aleatoriedad en la elección de las trayectorias para no "estancarse" en las mismas zonas y trabajar explorando regiones previamente no visitadas para conseguir una solución más interesante.

Ejemplos de trabajos realizado en entornos evolutivos a lo largo del tiempo pueden consultarse en [13] y [14].

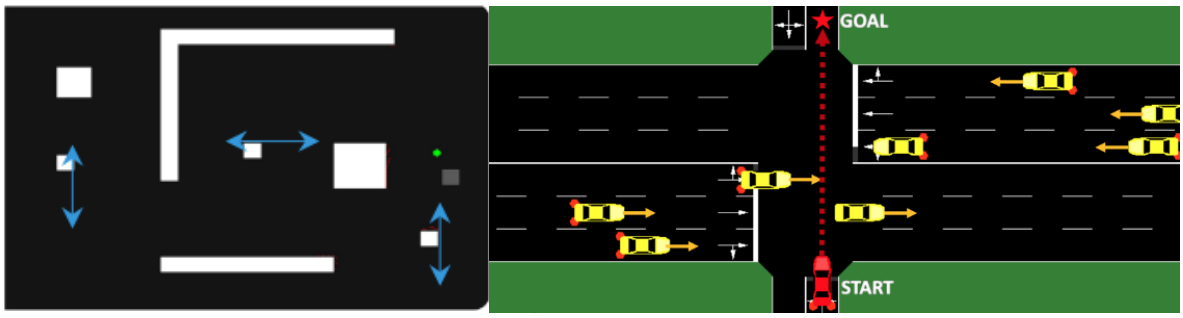



**Figura 2.5:** Ejemplos de entornos dinámicos presentes en estudios basados en la aplicación de técnicas de Reinforcement Learning [13] [14].

En la actualidad, los entornos suelen ser más específicos y ricos en detalles para la evaluación y entrenamiento de modelos ya que este hecho es intrínseco a la obtención de resultados mejores y más realistas. Por ejemplo, en [15] se realiza una investigación sobre la toma de decisiones automatizada sobre el comportamiento de un vehículo en un entorno de alto detalle. Se aprecia como se trabaja con un entorno simulado 3D que recrea el estado instantáneo de tráfico de una localización determinada, por lo que el modelo extraído tras el entrenamiento en dicho entorno parametrizado resulta muy realista.

## 2.5. Inteligencia artificial en tareas de patrullaje

Mediante el uso de inteligencia artificial se han abordado soluciones para muchos problemas propuestos relacionados con la navegación. Es cierto que la complejidad de la técnica empleada para la resolución vendrá en relación directa a dificultad del problema a abordar. Aunque tradicionalmente esta relación ha sido cierta, el nivel de especialización alcanzado en las técnicas desarrolladas ha cambiado este paradigma, ya que el uso de algoritmos complejos como el A3C no supone una carga computacional mayor que DQN por ejemplo.

El uso de *Machine Learning* en embarcaciones ha supuesto una herramienta, aún en desarrollo en gran multitud de casos, con muchas posibilidades de implantación que permiten optimizar tareas sin una supervisión humana completa. Existen distintos ejemplos que van desde una gestión óptima del ángulo sobre la horizontal de los módulos fotovoltaicos empleados como fuente de energía [16] hasta tareas complejas de patrullaje [17] con identificación de obstáculos y generación de mapa.

Tal y como se ha comentado anteriormente, la planificación de ruta automática es un aspecto muy importante y presente en los vehículos autónomos, y por consiguiente en la tarea de exploración y patrullaje en superficies de trabajo. El uso de sensores resulta necesario para poder recibir información del entorno. Aplicando técnicas de aprendizaje automático es necesario poder trabajar con dichos datos para que el módulo de control pueda tratarlos sean convertidos en información útil para el módulo de trazado de rutas, tarea necesaria para que el módulo de control del dispositivo realice su función adecuadamente. En vehículos acuáticos suele ser común el uso de sensores de posicionamiento global, al igual que en vehículos homónimos terrestres y aéreos, sensores de proximidad, sondas, visión por imagen, láseres, etc., necesarios para la correcta ejecución de tareas como el patrullaje. En dispositivos diseñados para realizar tareas más específicas, es plausible el uso de sensores de un uso mas particular como medidores de PH, densímetros, etc.

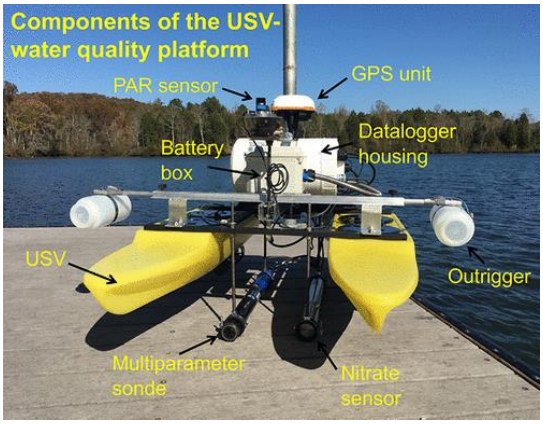



**Figura 2.6:** Ejemplo de dispositivo acuático de superficie y su equipamiento tecnológico[10].

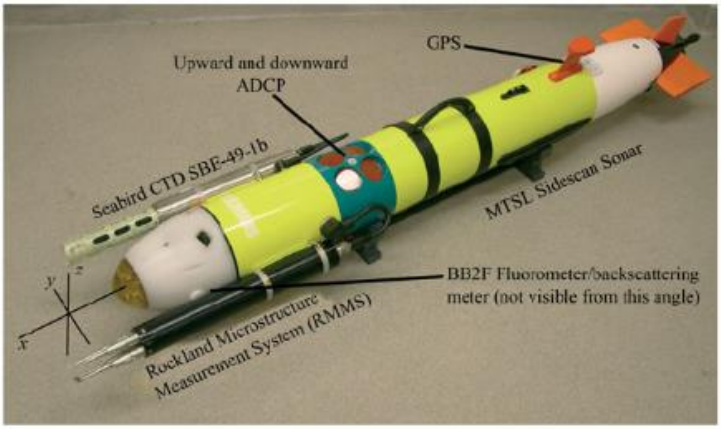



**Figura 2.7:** Ejemplo de dispositivo subacuático y su equipamiento tecnológico [18].

El uso del aprendizaje por refuerzo a la hora de afrontar tareas de exploración está muy implantada gracias a un gran número de investigaciones realizadas. Si hablamos de la navegación basada en una interacción visual con el entorno pueden encontrarse diversos métodos de trabajo como, por ejemplo, basado en la lectura de

[10] https://pubs.acs.org/doi/10.1021/acsestwater.1c00342

mapa [19]. También se desarrollan métodos de navegación en los que se va creando un mapa digital mientras el vehículo va navegando, ese mapa se va actualizando conforme se va tomando nueva información, y se explota por el algoritmo de path planning empleado [20].

Existe una investigación [21] donde se realiza una comparación entre un algoritmo de imitación y un algoritmo asíncrono *Deep Reinforcement Learning* cuando se plantea un problema de exploración sin mapa resolviendo como afecta a la obtención de una recompensa idónea un preentrenamiento previo.

En [22], se estudia como el A3C implementado a través de una GPU en un entorno de interior desconocido es capaz de realizar un mapeado mediante un sensor de escaneado laser y una cámara RGB-D en un dispositivo móvil. Como característica destacable, se introduce ruido gaussiano en las lecturas del sensor para hacer más real la experiencia del vehículo móvil.

Si nos referimos al entorno de nuestro lago Ypacaraí, una investigación realizada en el marco del departamento de ingeniería electrónica de la parte este trabajo [23] pone en marcha el estudio de técnicas de aprendizaje por refuerzo en dicho entorno, más específicamente *value based*, que ha servido para poner en marcha una experimentación real donde se trabaja la puesta en marcha de una flota de vehículos para la cobertura y toma de medidas en un emplazamiento acuático especifico. En dicha investigación se pone de manifiesto la bondad del uso de un algoritmo *Deep Q-Learning* en la tarea de exploración sin un modelado previo del entorno. Se tratará de estudiar el uso de algoritmos *on-policy* para posteriormente realizar una comparación entre ambos trabajos y comprobar que técnica ofrece mejores resultados.

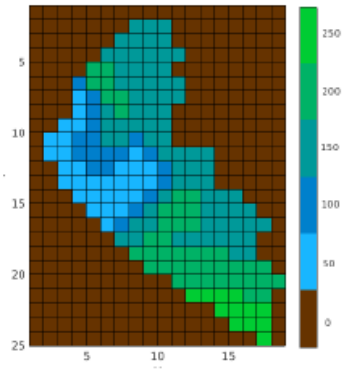



**Figura 2.8:** Caracterización del lago Ypacaraí en la investigación [5].

## 2.6. Path Planning

El planificado de rutas o de movimiento es una tarea muy importante de las tareas de navegación. El objetivo fundamental de este tema o tarea es resolver el reto de desplazar un sujeto desde un punto A hasta otro punto B en un entorno determinado, evadiendo obstáculos y cumpliendo unas directrices que tenga preestablecidas previamente, diseñando una serie de trayectorias a seguir hasta alcanzar el objetivo designado.

Es un término muy asociado a campos tecnológicos como la robótica, donde toma un papel muy importante. Todos los vehículos que se diseñan para desarrollar algún tipo de movimiento navegando sobre cualquier superficie o por el aire debe enfrentarse al problema de la generación de una ruta y posteriormente de trayectorias, si estamos hablando de un vehículo autónomo o semi autónomo. Suele caracterizarse como un trabajo de alto nivel, y dentro del bucle de control de cualquier vehículo, suelen implantarse estos algoritmos justo tras la recepción de datos del entorno, ya que para la resolución correcta del problema del movimiento autónomo se debe tener una imagen clara del estado del entorno.

Un aspecto muy importante que debe tenerse en cuenta es que esta técnica es muy sensible a discordancias y errores que pueden aparecer asociados a trabajar con un mal modelado. Es por tanto de vital importancia que se trabaje con un modelo preciso y bien calculado.

El desarrollo de algoritmos de *path planning* cada vez más robustos y el uso de tecnología de *Machine*

*Learning* para dicha tarea ha impulsado la navegación no tripulada a ser una de ramas de la robótica más prominente, ya que pueden abordarse escenarios muy complejos con características dinámicas destacables que antes eran inalcanzables. Este hecho ha provocado que las líneas de investigación y desarrollo centradas en el empleo de vehículos autónomos crezca exponencialmente desde los años 90.

Hoy en día pueden verse ejemplos en cualquier entorno y con diversas tareas que pueden abordarse con esta tecnología. Ello supone que muchas puedan ser automatizadas o realizadas para evitar situaciones de peligro a los seres humanos.

**Figura 2.9:** Robot de monitorización de cultivos vinícolas [24].

Esta técnica consiste en resolver un problema de optimización que normalmente se plantea como solución a trabajar sobre un modelo representativo de la superficie de trabajo. Como se ha descrito es muy importante que el modelo del vehículo esté muy bien realizado ya que podría inducir a un error en el sistema. El seguimiento de una ruta consiste en un problema de control conocido donde se busca reducir la variación de las trayectorias realizadas respecto a las planteadas por el planificador de rutas [25].

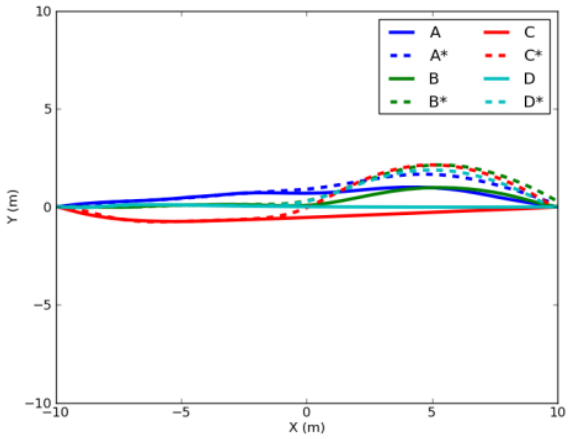



**Figura 2.10:** Ejemplo de variación de rutas realizadas respecto a las planteadas inicialmente en vehículo terrestre.

# 3 DEFINICIÓN DEL PROBLEMA

## 3.1. Planteamiento del problema

El problema que se plantea consiste en la exploración de un entorno acuático conocido previamente mediante un vehículo autónomo de superficie. Cuando nos referimos a que el entorno acuático es conocido previamente significa que en el sistema se incorpora las características suficientes y necesarias de este para poder trabajar los algoritmos en él. El enfoque que buscamos en este trabajo consiste en realizar un análisis de los resultados obtenidos tras la aplicación de varios algoritmos de la familia *Deep Reinforcement Learning* en la resolución del problema de patrullaje no homogéneo en la superficie del lago Ypacaraí.

Definimos el problema del patrullaje como un conjunto de subtareas a realizar:

- Cobertura de la superficie acuática para la recolección de datos necesarios.
- Actualización del estado del lago y la importancia de las zonas que lo conforman.
- Seguimiento del estado del vehículo y su ubicación.
- Optimización de la ruta de patrullaje teniendo en cuenta zonas de mayor importancia.

El problema descrito consiste es una versión particular del problema de cobertura por planificación de rutas. La planificación de rutas o *path planning* consiste en la tarea de crear una ruta, o secuencia de trayectorias, en la que se optimicen los recursos disponibles, como la batería del vehículo, y a la vez se produzca una toma da muestras lo más eficiente posible. En este caso el objetivo principal no consiste en una cobertura total del espacio de trabajo, sino optimizar y recoger el máximo número de muestras con la mayor importancia posible. Tal y como se entenderá la importancia en este trabajo consistirá en el valor numérico que evaluará el interés de ciertas zonas para el muestreo por parte del ASV. Un factor importante será la concentración de microorganismos, sustancias químicas en el medio acuático y la ubicación de puntos de vertido. La cobertura de una superficie se define como el proceso en el que se visitan las regiones que componen dicha superficie, mediante un vehículo en este caso, que tiene como objetivo cubrir la superficie para la visita y toma de datos.

Con dicha afirmación se deja entrever que en esta versión se valora la posible visita reiterada de zonas con valor destacable en favor de otras poco interesantes a lo largo del tiempo. Es por tanto necesario obtener un algoritmo que permita desarrollar un *path planning* con un enfoque de patrullaje particular cuyo objetivo sea optimizar los recursos energéticos del vehículo en la máxima toma de muestras con mayor importancia de recogida posible. Por tanto, la métrica que debe tomarse como referencia será la toma de resultados más elevada en importancia absoluta del escenario estudiado.

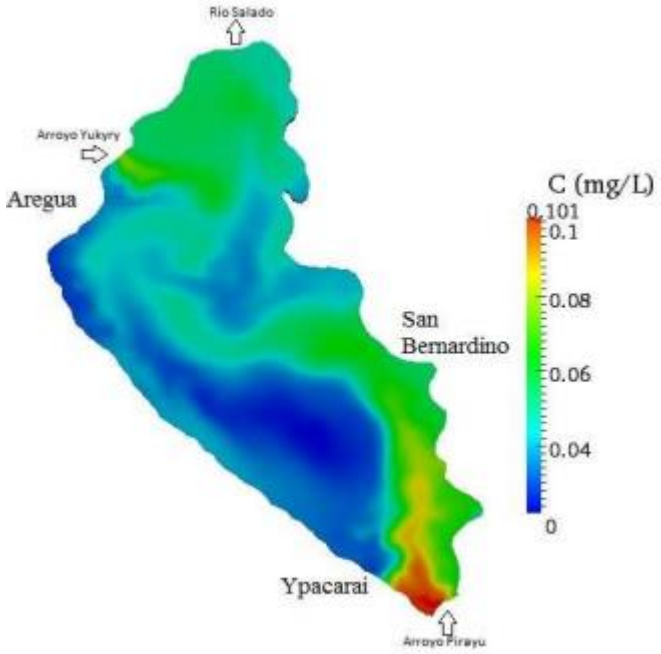



**Figura 3.1:** Concentración no homogénea de contaminantes en el lago Ypacaraí [33].

Si se observa la imagen anterior salta a la vista que la interpretación del problema planteado como cobertura total resulta poco adecuado, será interesante en varios casos la visita reiterada a distintas zonas. Se ha realizado previamente una visión del estado de la técnica y referenciado varias investigaciones de algoritmos de aprendizaje por refuerzo profundo en problemas similares y estudiado las condiciones de nuestro emplazamiento para resolver que parece adecuado abordarlo con dicha metodología. Particularmente resulta interesante el hecho de poder abordar un escenario dinámico con carácter estocástico con este tipo de técnica ya que resulta muy adecuada cuando partimos de un escenario conocido.

En el contexto de este trabajo el patrullaje informativo consiste en la operación de exploración o de la superficie definida en la que se recopila información necesaria para su posterior análisis. Es por tanto una tarea compuesta de dos subtareas, la exploración de la superficie y la toma de datos de la misma. En dicha superficie se encontrarán diversas zonas con distintas características o importancia. Si observamos la figura 3.1 apreciaremos como existen zonas con mayor o menor concentración de contaminantes y es importante comentar que para nuestro sistema o modelo será más interesante o importante visitar esas zonas donde la presencia por metro cuadrado de contaminantes es mayor. Es por ello que cuando nos referimos a la importancia de una zona, estamos evaluando la concentración o la importancia de la toma de muestras en dicha zona.

Este trabajo pretende convertirse en una investigación paralela a trabajos previos realizados sobre la aplicación de técnicas *Deep Reinforcement Learning* [5] específicamente con algoritmos *on-policy*. Es importante comentar la existencia de varias restricciones para tener en cuenta como la zona operativa que corresponderá a la superficie navegable del lago. Adicionalmente se tendrá en cuenta las características del ASV como vehículo eléctrico y la sensórica integrada en el mismo que definirán la autonomía y la superficie de influencia de este. Por todo ello será necesario el desarrollo de un entorno en el que definir dichas limitaciones y poder trabajar con una caracterización adecuada del entorno. Para la representación del mapa del lago en un entorno accesible para trabajar con él, se realizará una adecuación del mismo a las condiciones estandarizadas como entorno de entrenamiento. Esto supone que el uso de distintos algoritmos en el mismo entorno resulta muy adecuado para la tarea posterior de evaluación de resultados y comparativa correspondiente. En la sección correspondiente a resultados se realiza un análisis profundo de las características necesarias y restricciones que han sido consideradas oportunas para la correcta caracterización del entorno.

# 4 METODOLOGÍA

## 4.1. Deep learning

El *Machine Learning* o como también es conocido aprendizaje automático o automatizado es un campo de estudio en el que se busca mediante diversas técnicas que se produzca un aprendizaje autónomo. Este aprendizaje se desarrollará en un elemento con propiedades computacionales.

El objetivo de aplicar *Machine Learning* a un ordenador, por ejemplo, es conseguir diseñar una técnica con la que se extraiga una información correcta, de un análisis realizado correctamente, de unos datos para el posterior aprendizaje a partir de ellos. Hay muchas acciones posibles a realizar mediante estas técnicas como el reconocimiento de patrones, imágenes, realizar predicciones de comportamiento en sistemas, reconocimiento por voz, estudios de mercado y sociales, etc., pero una característica común en todas estas técnicas o procesos de aprendizaje es que el aprendizaje se realiza a partir de la experiencia.

El modelo resultante de este aprendizaje está directamente relacionado con la finalidad de este aprendizaje. Hay muchísimos tipos de con diversas características qué variarán en función de la complejidad de la tarea a resolver.

Anteriormente se ha comentado que el aprendizaje se realiza mediante diversas técnicas y estas serán recogidas en forma de algoritmo. Cómo se puede prever debido a la enorme cantidad existente de diversos tipos de algoritmos, a su vez el aprendizaje automático puede catalogarse en diversos tipos de aprendizaje que dependerán del algoritmo en el que estén basados.

Aquí se tratará especialmente un tipo de técnica denominada aprendizaje por refuerzo, que es una técnica que suele aplicarse en tareas o problemas complejos debido a su potencia.

El *Deep Learning* puede denominarse un subcampo del *Machine Learning* que enfrenta el aprendizaje automático en el tiempo de decisiones óptimas. El *Deep Learning* supone una evolución que surge del *Machine Learning* que parte de la inspiración en el aprendizaje humano y utiliza arquitecturas de redes neuronales para llevar a cabo su función. Además, supuso un gran avance en el campo científico y la inteligencia artificial.

En realidad, tanto el *Deep Learning* como el *Machine Learning* se sirven como modelo de aprendizaje el humano, solo que el aprendizaje profundo se asemeja más al humano debido a al uso de las redes neuronales anteriormente comentadas. En cambio, en *Machine Learning* suelen aplicarse arquitecturas como ramas o arboles de decisión, que resultan algo más simples, pero menos eficientes.

El término *Deep* aparece debido a que las neuronas empleadas en esta técnica estarán organizadas en capas, unas detrás de otras, y esta organización recibe el nombre de red neuronal profunda.

Con todo ello puede verse la evolución que sufre el campo del aprendizaje automático y vemos la tendencia de este en asemejarse al raciocinio humano.

El uso de las redes neuronales en esta familia de algoritmos se desarrolla en dos etapas importantes, el entrenamiento donde se busca obtener un modelo y la ejecución de este modelo para obtener resultados finales.

Durante el entrenamiento de la red neuronal se introducen los datos de entrenamiento a esta. Dicha información sufre una transformación a través de las capas neuronales que componen la red hasta alcanzar la última de ellas, este método se denomina *forward propagation*. Tras obtener un resultado de entrenamiento se compara con otro conocido para extraer la información necesaria para optimizar los valores de los pesos y sesgos de los parámetros internos de la red. Este proceso se denomina *backward propagation* y se realizará tantas veces como se defina en favor de alcanzar el resultado final esperado.

Dicha técnica obtiene resultados más adecuados y alcanza grados elevados de eficacia cuantas más iteraciones se realicen del proceso anterior, aunque no es adecuado prolongar dicho entrenamiento indefinidamente.

## 4.2. Reinforcement Learning

El *Reinforcement Learning* representa una modalidad de aprendizaje dentro del campo de *Machine Learning*. En el aprendizaje por refuerzo aparecen conceptos que pueden estudiarse y adoptar con el tiempo, con lo que pueden realizarse modelos a partir del tiempo y conseguir modelos dinámicos que pueden resolver tareas que antes difícilmente pudieran ser abordables.

El aprendizaje por refuerzo surge como solución a problemas que en principio pudieran resultar estáticos de entrada y salida pero que al sufrir una evolución en el tiempo no pueden ser afrontados con otras técnicas más primitivas.

En esta clase de aprendizaje un agente tiene que aprender a tomar y realizar acciones en un entorno que puede ser estático o dinámico. Para ello existe un concepto de recompensa muy importante en este tipo de aprendizaje y que puede ser positiva o negativa. El agente irá aprendiendo de su experiencia en el entorno y conocerá como de bueno o malo serán sus acciones en él en función de la recompensa que vaya obteniendo. Es importante que el aprendizaje por refuerzo lo que busca es alcanzar una recompensa lo más alta posible a lo largo del tiempo por lo que es posible que se pueda observar un comportamiento no eficiente del agente a corto plazo.

Otro aspecto muy importante es que deben tomarse todo tipo de decisiones, aunque su recompensa posterior sea positiva o negativa ya que deben conocerse todos lo resultados posibles que puedan darse porque tomar una decisión mala a corto plazo puede ser más beneficiosa en el futuro y en recompensa total.

Los métodos de aprendizaje por refuerzo pueden clasificarse por varios aspectos:

- Basados en modelo, o sin modelo.
- Basados en valor o política.
- Con política o sin política.

Cuando tenemos un método sin modelo significa que no se crea un modelo del entorno o la recompensa, sino que directamente se relaciona las acciones con las recompensas. En cambio, si tenemos un método que genera un modelo que tratará de realizar predicciones de las siguientes recompensas y las observaciones que tendrán lugar.

Cada una de las dos opciones tienen sus puntos fuertes y débiles, pero normalmente los métodos con generación de modelo suelen usarse cuando existen un entorno determinado y que posea reglas estrictas y bien definidas.

Si analizamos la segunda clasificación de métodos, un método basado en *value iteration* analizará con el agente como protagonista cada acción posible y su resultado de recompensa para siempre tomar el mejor valor. En cambio, el método basado en política, o *policy iteration* aproxima en cada caso la política del agente y por lo tanto la acción que debe realizar en cada instante. La política se representará como una distribución de probabilidades de tomar coda acción por el agente.

Para terminar, se comentará el ultimo tipo de clasificación, y es que un método on-policy como se caracteriza por ir actualizando y optimizando la política con la que se trabaja para obtener el mejor resultado donde el

agente capta la mejor política y la emplea para actuar. Por lo tanto, la política que se usa para actuar y actualizar el algoritmo es la misma. En cambio, en un método *off-policy*, que puede verse muy empleado en *Q-learning*, el agente aprende la política óptima usando una política tipo *greedy* para después comportarse con la política de otros agentes. Principalmente, aunque sea un poco confuso este método se considera *off-policy* porque la política que es actualizada en el algoritmo es distinta a la empleada para actuar.

Es muy común que la morfología de los problemas propuestos de resolución mediante el *Reinforcement Learning* sigan la morfología de los procesos de decisión de Markov (MDP *Markov Decision Process*). Al aplicar esta caracterización determinada a algún problema a resolver, se garantiza una posible resolución mediante la metodología de aprendizaje por refuerzo.

El MDP es la forma matemática ideal del problema de aprendizaje por refuerzo. Permite formalizar un problema de toma de decisiones donde las decisiones tomadas en el instante presente tienen influencia en las decisiones futuras. Resulta interesante aplicar este formato a nuestro problema en el lago Ypacaraí ya que sería adecuado plantear que las decisiones del vehículo influyesen en estados futuros, trayectorias, etc.

A continuación, se describirán algunos elementos importantes que tienen lugar en los MDP y la funcionalidad de los mismos.

### 4.2.1. Elementos del aprendizaje

- **Entorno**

El entorno es un espacio de trabajo conformado por un conjunto de reglas, normas y limitaciones que definen el comportamiento y evolución de este. Un entorno puede estar basado en un modelo o no, en función de la tarea que quiera realizarse. Lo que, si tienen en común todos los entonos, es que su tarea principal es de servir de apoyo para la realización de experimentos y operaciones encargándose de reproducir las acciones a realizar por el agente y calcular las consecuencias de estas en el universo ficticio que supone el entorno para el agente. Se encarga de transmitir la recompensa asociada a la acción a realizar por el agente a este mismo. A parte de las acciones y las recompensas, el entorno puede dar otro tipo de información denominada observaciones. Una observación interesante podría ser el estado del agente en el entorno, por ejemplo, o algún dato numérico o posicional relativo al agente. Las observaciones se tratan de información secundaria relativa al estado del agente en el entorno que también puede incluir información relacionada con la recompensa.

- **Agente**

El agente será la entidad que en cualquier problema o experimento es el encargado de tomar acciones y enviar información al entorno con sus decisiones. Además, es quien con su acción tomada es el responsable de cambiar de estado cada step del sistema y por lo tanto lo hace evolucionar. El entorno como ya hemos comentado enviará como mínimo información al agente acerca del estado y recompensa, pudiendo también encontrarse diversas observaciones transmitidas deliberadamente por el desarrollador en cada caso.

Principalmente puede definirse como el órgano responsable de mantener contacto con el entorno, realizar acciones transmitiendo y recibiendo información con él.

- **Recompensa**

La recompensa se trata de un valor numérico que evalúa la importancia tanto positiva como negativa de la ejecución de ciertas acciones en el entorno por el agente. Por lo tanto, definirá como de buena o mala sea realizar una determinada acción. Este escalar será proporcionado por el entorno tras recibir la acción que tome a cabo el agente mediante una interacción que se explicará un poco más adelante.

Independientemente de la técnica de aprendizaje o algoritmo empleado el funcionamiento del concepto de recompensa debe ser el descrito.

Lo más típico es que el comportamiento de la recompensa sea local, es decir que muestre el valor de éxito del agente instantáneo o en la actividad actual y no del éxito acumulado del sumatorio de experiencias anteriores. A veces es referido el concepto de recompensa a alguna variable donde se acumule los valores de éxito por

parte del agente para su uso en ciertos algoritmos donde sea necesario su uso o estudio.

En realidad, lo que buscamos son acciones que nos aporten estados con un valor lo más alto posible porque dichas acciones proporcionaran las mayores recompensas a lo largo del tiempo, pero como veremos será mucho más complicado obtener estos valores comentados que los valores de las recompensas.

La frecuencia con la que el agente reciba la recompensa será definida por el algoritmo diseñado.

- **Estado**

El estado puede definirse como una imagen instantánea del universo creado por el entorno y el agente dentro de éste. Refleja por completo la situación del entorno y del agente en un instante o step preciso.

Suele ser uno de los elementos o información que el agente recibe del entorno. El número de estados distintos que puede tener lugar dependerá de la simplicidad o no del entorno y del número posible de acciones a poder realizar por el agente, siendo usualmente un número bastante elevado.

Los estados tienen asociado un valor numérico que no debe confundirse con la recompensa. Es un poco confuso, pero básicamente decimos que el valor de un estado será el valor total de recompensa que el agente conseguirá a lo largo de pasos o *steps* partiendo desde el estado en sí mismo. Cuando hemos visto que la recompensa se caracteriza por ser un valor instantáneo.

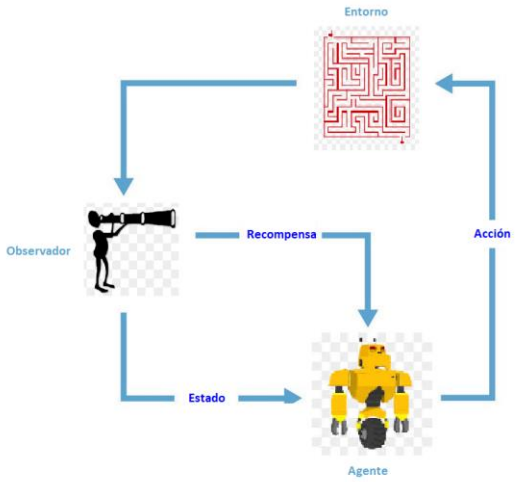



**Figura 4.1:** Elementos principales y sus interacciones del *Reinforcement Learning*.

- **Políticas**

Las políticas son las encargadas de dominar el comportamiento del agente. Son una especie de reglas que el agente deberá seguir en su funcionamiento. Gracias a las políticas se podrá condicionar el objetivo del aprendizaje automático, así como varios parámetros como la velocidad y estabilidad.

Es muy común que las políticas tengan carácter estocástico ofreciendo probabilidades a cada posible acción para que tenga lugar, por lo que a veces se suele referirse a ella como distribución de probabilidades.

Así como es número de estados podría ser muy elevado lo mismo ocurre con las políticas por lo que es muy difícil que una política sea igual a otra al igual que ocurre con los estados.

La política es un elemento muy importante en el aprendizaje por refuerzo, ya que normalmente la búsqueda de la política idónea lleva asociado la optimización del proceso que es este estudiando.

Podemos encontrarnos también políticas deterministas que vendrán definidas en función de la acción tomada en un estado concreto.

La nomenclatura que suele emplearse para referirse a ella en un instante t es:

$$\pi(a|s) = P[A_t = a|S_t = s]$$

Los diversos métodos de aprendizaje por refuerzo especificaran cómo irá variando o evolucionando la política a lo largo del tiempo o pasos aprendiendo la experiencia pasada y acumulada.

- **Exploración y explotación**

Cuando queremos implantar un aprendizaje no supervisado es muy importante que exista un balance entre explotación y exploración. El agente deberá aprender en determinar una estrategia eficaz que sea eficaz para la tarea de exploración del entorno.

Esta estrategia consiste en que el agente debe encontrar el equilibrio idóneo entre explorar nuevas situaciones o estados en los que obtener buenos resultados o recompensas altas y explotar resultados previamente alcanzados en estados que ya sabe que producen buenas recompensas que garanticen una recompensa acumulada alta.

El imposible realizar las dos acciones a la vez y por ello es importante que el agente emplee una estrategia adecuada que garantice un resultado con más o menos seguridad de buenos resultados.

Existen varias estrategias muy conocidas como la *ε-greedy*, métodos basados en la distribución de Boltzmann, métodos de estimación de intervalos y valores iniciales optimistas.

Es un problema dentro del Reinforcement *Learning* la combinación de la exploración y explotación ya que es importante que el agente siempre siga buscando nuevos estados, aunque ya haya alcanzado buenos resultados, ya que, si llegado un punto el algoritmo se dedica solo a explotar estados pasados en decremento de explorar, el agente dejará de aprender [33].

- **Interacción agente entorno recompensa**

En este apartado se ha ido comentando en cada uno de los elementos si forma parte de la comunicación entre agente y entorno.

La interacción entre estor dos elementos es muy importante ya que si no existiera el agente no sabría lo bien o mal que es la acción que acaba de realizar, ni que recompensa recibe esa acción o en qué estado se encuentra o acaba el entorno debido a esa acción. A la inversa ocurre lo mismo, tendríamos un entorno o universo en el que el agente no existe o no afecta en ningún aspecto a estado o evolución de este.

Una vez que se puede tener una idea de la importancia de esta relación, es importante tener en cuenta que la interacción se realizara con la frecuencia que exija el método de aprendizaje con el que estemos trabajando y que los elementos básicos que se transmiten en esta comunicación serán:

- Estado, s
- Recompensa, r
- Acción, a
- Observaciones, obs

La información podremos verla reflejada tanto en números escalares como la recompensa, la acción y diversas observaciones, como en matrices de puntos como puede ser representado el estado y varias observaciones.

Es por tanto de una gran importancia el hecho que exista esta interacción, ya que en su defecto no podría realizarse un aprendizaje por refuerzo al no tener información de episodios pasados y no poder aprender de la experiencia.

Simplemente se ha tratado de introducir una serie de conceptos de manera básica y entendible para que el lector pueda formarse una idea correcta y relativamente clara del contenido y alcance del proyecto.

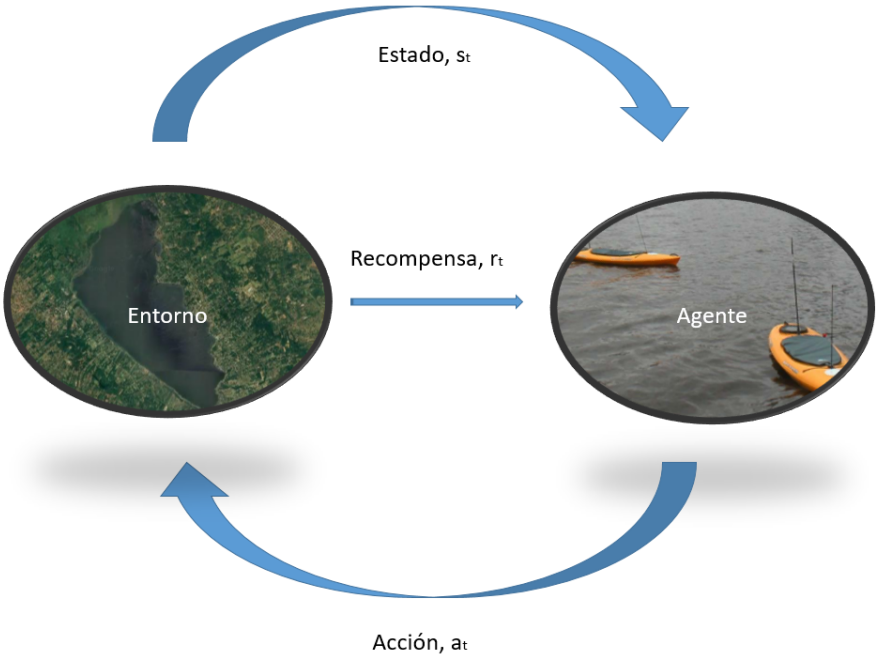



**Figura 4.2:** Arquitectura de la comunicación entre elementos de aprendizaje

## 4.3. Deep reinforcement learning

El aprendizaje por refuerzo profundo o *Deep Reinforcement Learning* puede considerarse una familia de métodos que surge al aplicar redes neuronales *deep neural networks* para trabajar con conceptos o elementos del aprendizaje por refuerzo *Reinforcement Learning*. La idea es obtener una aproximación empleando estas redes del modelo a partir de las funciones de estado y recompensa que gobiernan el sistema con el que se trabaja. En esta familia, existen métodos basados en la optimización de una política (*on-policy*) $\pi(a, s; \theta)$ y otros(*off-policy*) en los que también aparece el concepto de política, pero como se comenta en él apartado 4.3.1 no se emplea en sí para regir la actuación del agente del sistema. Independientemente del algoritmo estudiado, en esta familia aparecen los parámetros $\theta$ que son los pesos de las redes neuronales. Como puede comprobarse en la literatura, dichos pesos rigen los elementos que conforman las bases teóricas y prácticas de los algoritmos: *value function* $\mathrm{v}(\mathrm{s}, \theta)$, *policy* $\pi(s, \theta)$, etc.

La principal diferencia que existe entre el *Reinforcement Learning* y el *Deep Reinforcement Learning* consiste en el método de aproximación empleado. En el aprendizaje por refuerzo profundo se realiza mediante un actualización de los pesos internos de las redes neuronales, según una determinada tasa de refresco que dependerá del algoritmo empleado, que permitirá obtener una convergencia del modelo evaluando la bondad de los resultados obtenidos en las experiencias acumuladas

Usualmente, se plantea el uso del gradiente por descenso o ascenso a la hora de minimizar las pérdidas o maximizar las recompensas de la función objetivo que suelen ser estocástico en lugar de determinista, ya que es interesante trabajar con unas variables probabilísticas a la hora de afrontar problemas de exploración-explotación de soluciones.

Dentro de esta familia de algoritmos, suelen categorizarse estos mismos mediante las denominaciones *value based* y *policy based*, aunque existe una corriente que además define otra categoría más que recoge los métodos llamados *Actor Critic*. Dicha denominación podría ahorrarse ya que cómo se verá en el apartado 4.7 ya que estos algoritmos comparten arquitectura de operación con los policy based, pero resulta interesante comentar este aspecto.

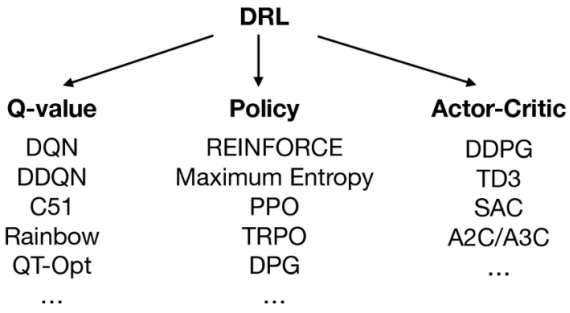



**Figura 4.3:** Clasificación de los principales algoritmos en subfamilias de *Deep Reinforcement Learning*. Fuente:https://www.researchgate.net/publication/356270871_A_survey_on_deep_learning_and_deep_reinforcement_learning_in_robotics_with_a_tutorial_on_deep_reinforcement_learning

Aunque no se ha comentado hasta ahora, en el apartado 4.5 se verán los elementos esenciales empleados en las metodologías de aprendizaje por refuerzo y aprendizaje por refuerzo profundo ya que se prefiere explicar dichos conceptos con claridad y la función que realizan en los algoritmos de esta familia solo una vez, y no en los dos apartados anteriores.

### 4.3.1. Técnicas de aprendizaje

En este apartado se comentará de manera muy superficial algunos de las técnicas de aprendizaje más extendidas y usadas para ser comentadas. Estas técnicas se definen en forma de algoritmo que desarrollarán el procedimiento de aprendizaje correspondiente.

- **Aprendizaje supervisado**

En primer lugar, tenemos el aprendizaje supervisado, puede ser el más conocido dentro de la gran gama de algoritmos empleados en aprendizaje automático. Consiste en un aprendizaje automático a partir de una base de datos ya catalogados o caracterizados perfectamente. Se denomina supervisado porque se va a prendiendo a partir de soluciones que ya se conocen por los datos que se tienen como referencia. Los datos o ejemplos pueden decirse que ya están categorizados.

- **Aprendizaje no supervisado**

En cambio, el aprendizaje no supervisado consiste en que los datos que se tienen no están categorizados y por lo tanto la tarea que se debe realizar sobre ellos es una búsqueda de patrones o relaciones que pueda extraerse de los mismos. Es una técnica que desde su descubrimiento se ha empleado más y más en campos como la biología, muchos aspectos relacionados con la visión, reconocimiento de patrones, etc.

Dentro de esta familia de algoritmos existen diversas técnicas u algoritmos que tienen mejores o peores cualidades frente a los problemas que se generan en el desarrollo de esta técnica. Un buen método creara grupos de datos que serán lo más distintos entre sí posible, y que los datos que clasifique en cada grupo se asemejen bastante.

- **Aprendizaje por refuerzo**

El aprendizaje por refuerzo se ha comentado anteriormente y por resumir conceptos y comentar superficialmente se definirá brevemente.

Esta familia de algoritmos plantea un aprendizaje en el que se evolucionará a partir de la experiencia en la exploración de un entorno controlado en más o menos nivel. Se hace uso de una entidad denominada agente que será el encargado de realizar una serie de acciones en un entorno. Al realizarse esta serie de acciones comentada el agente recibirá una serie de recompensas que pueden ser positivas o negativas y el algoritmo irá

aprendiendo a partir de estas experiencias obtenidas para intentar que las acciones que realice el agente sean las que proporcione la mayor recompensa total.

Como en las otras técnicas comentadas existen una gran variedad de algoritmos o métodos dentro de la familia de Reinforcement *Learning*.

Los procesos de decisión de Markov (MDP) y la ecuación de Bellman son empleados en muchos algoritmos de esta familia para la búsqueda de soluciones óptimas y el planteamiento de un problema de manera adecuada. Estos conceptos pueden encontrarse debidamente explicados en este documento.

- **Método cross-entropy**

El método de entropía cruzada es un método mucho menos usado que otros como los englobados en los tipos *value iteration* y *policy iteration*, pero posee algunas características reseñables que hace interesante que sea comentado.

- **Value iteration**

Los algoritmos del tipo *value iteration* pueden ser una de las familias más extendidas dentro del *Machine Learning*.

Tenemos un MDP que deberá resolverse para obtener la máxima recompensa final. Sin llegar a profundizar mucho más diremos que habrá que evaluar cada una de las acciones que llevaran al agente de un estado a otro y observar la recompensa que se obtiene en cada momento y proceso a través de la función de valor con una política determinada. Si se maximiza la recompensa encontrando la mejor secuencia de acciones y extraemos a posteriori una política, podemos confiar en que la política será la óptima para la resolución del proceso de decisión.

**Value Iteration, for estimating $\pi \approx \pi_*$**

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \text{argmax}_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

Como podemos ver en cada bucle de ejecución del algoritmo existe un procedimiento de evaluación de cada acción-estado que no para de recorrer todas las posibles situaciones hasta que el método converja para posteriormente extraer la política relacionada con la trayectoria óptima realizada por el agente.

Es un método que conlleva una carga computacional elevada. Si tenemos como ejemplo un MPD con 4 posibles estados y 3 acciones supone que tendremos como mínimo 24 movimientos del agente que evaluar con sus correspondientes estados a interacciones con el entorno. Todo ello debe evaluarse siguiendo la política anteriormente calculada y teniendo en cuenta además al valor anterior con el coeficiente de recuerdo para luego extraer la combinación que produce la máxima recompensa y tras ellos extraer la nueva política óptima.

Existen varias adaptaciones del método que reducen el tiempo de convergencia del algoritmo que pueden ser aplicadas.

Es un método robusto que puede afrontar problemas complejos asegurando la resolución de estos gracias al uso de capas neuronales que permiten aproximar mediante el uso de filtros, funciones de activación, optimizador y un proceso de resolución matemático conocido como inducción hacia atrás. Así este método es capaz de estudiar todas las posibles situaciones que puedan darse en un escenario de trabajo y analizar la mejor solución en función del estado actual del mismo.

- **Policy iteration**

Si hablamos de algoritmos tipo *policy iteration*, nos referiremos a una familia de algoritmos con una funcionalidad concreta y que podría asemejarse en cierta forma a los tipos *value iteration* previamente comentados.

Este tipo de algoritmo suele comenzar a trabajar con una política arbitraria que se irá mejorando tras ser evaluada para que vuelva a sufrir el mismo proceso una y otra vez hasta que se obtenga la política idónea y el proceso converja.

En el método pueden identificarse dos procesos fundamentales, la mejora de la política y la evaluación de la política, que son los mismos que podemos encontrar en *value iteration*. Existen varias diferencias entre los métodos y cada uno de ellos posee ciertas cualidades que hacen que su explotación sea idónea en función del problema a resolver.

**Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$**

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\left[r + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   *old-action* $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Con el *policy iteration* se busca una rápida convergencia hacia una política idónea. Este método cumple que, tras iniciar una iteración en la resolución con una política arbitraria, la siguiente, obtenida tras la función de valor de la política anterior, será mejor. Esto supone que las políticas irán variando poco a poco de la anterior y por lo tanto lo mismo ocurrirá con las funciones de valor así que la carga computacional asumiblemente leve del proceso asegura unas iteraciones rápidas y por lo tanto la convergencia del MPD se alcanzará rápidamente. Es común que muchas situaciones el problema converja en varias iteraciones mediante este método.

Este método se centra directamente en la política y por lo tanto en el comportamiento del agente en el entorno y por ello se necesita información proveniente de éste.

Los procesos de mejora de política y evaluación de política están directamente relacionados entre sí. La mejora de la política se realiza respecto a la función de valor que a su vez se evalúa a partir de una determinada política elegida como comportamiento.

Cuando los dos procesos dejen de sufrir cambios y hayan alcanzado una determinada estabilidad, podremos decir que se ha alcanzado la política idónea y por lo tanto la función de valor asociada será óptima.

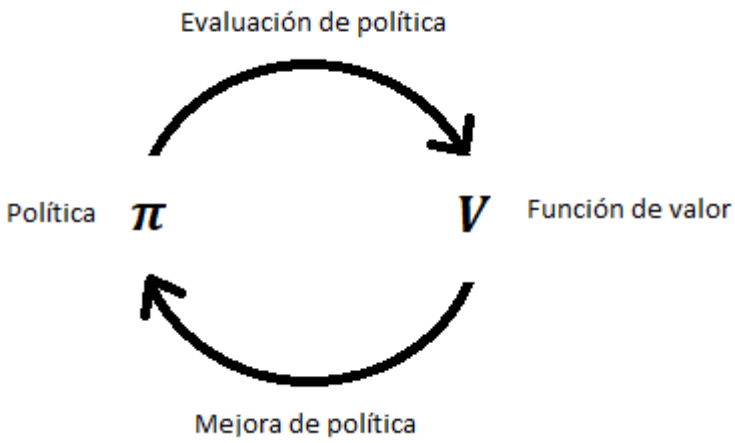



**Figura 4.4:** Representación del modelo de funcionamiento del método *policy iteration*.

La política empleada en los distintos algoritmos de este método es de carácter estocástica, es decir que posee un comportamiento no determinista donde aparece en cierta medida un componente aleatorio. Esto es así porque en la familia *Reinfocement Learning* también es importante la exploración de acciones no idóneas o no adecuadas ya que es posible que tras una acción no óptima a largo plazo se alcance un resultado idóneo. Por lo tanto, es conveniente que la política sea estocástica y realizar un aprendizaje de las decisiones "buenas y no tan buenas".

Aunque este método requiera de más interacción con el entorno que otros como el *value iteration*, necesita de menos datos anteriores para su funcionamiento por lo que la carga computacional es menor, pero en su defecto el uso de un gran número de datos por parte del método *value* proporciona una eficiencia muy alta de resolución, aunque conlleva más carga computacional. También en este método solo se precisa acceder a la red neuronal una sola vez por iteración en la ejecución.

En el siguiente apartado se describirá en más detalle el funcionamiento de los métodos basados en gradiente de política.

## 4.4. Métodos policy gradient

Los métodos *policy gradient* tienen como objetivo alcanzar el ajuste idóneo de los parámetros de la red neuronal mediante la técnica del gradiente ascendente (*gradient ascent*) en la búsqueda de la política idónea. Esta técnica consiste en estudiar la variación del gradiente que se devuelve en la función que obtenemos a la salida de la red neuronal que obviamente tiene que ser evaluada por el algoritmo.

Lo que se obtiene al evaluar este gradiente, es la dirección en la que deben modificarse los parámetros de la red neuronal y la variación en mayor o menor medida. Con esta técnica aparecen ciertas inestabilidades si no se limita la variación que se aplica al gradiente en cada step por lo que suele saturar su valor. En un ejercicio de practica se obtendrá el equilibrio deseado entre velocidad y efectividad de resultados para así obtener un buen valor de saturación del gradiente.

La técnica *gradient ascent* consiste en una actualización del gradiente buscando una maximización de la función objetivo. El parámetro $\alpha$ se denomina *step size*.

$$\theta_j \leftarrow \theta_j + \alpha \nabla_\theta J(\theta)$$

En cambio, el descenso del gradiente (*gradient descent*) tiene como objetivo la minimización de la función en cuestión.

$$\theta_j \leftarrow \theta_j - \alpha \nabla_\theta J(\theta)$$

El uso de estas técnicas en *Machine Learning* dependerá de si el tipo de función es convexa o cóncava.

Cuando la función de coste es cóncava, usaremos el método del gradiente ascendente buscando un máximo local y en cambio cuando sea convexa usaremos el descendente para buscar un mínimo local.

Como en los métodos *policy gradient* buscamos optimizar la recompensa, usaremos el ascenso del gradiente para obtener el máximo de la función de recompensa.

$$J(\theta) = \sum_{\tau \sim \pi_\theta} P(\tau, \theta) R(\theta) = E_\pi[R(\tau)]$$

La función objetivo a maximizar $J(\theta)$, nos informará de como de idónea es la política empleada al calcular la recompensa esperada asociada a una política particular.

El gradiente de esta función es lo que se conoce como gradiente de política o *policy gradient*.

Cuando se emplean políticas estocásticas, las redes neuronales ofrecen a su salida un vector de probabilidades que nos indica la probabilidad de realizar cada una de las acciones categorizadas. Por ello al estudiar la probabilidad que tiene asociada cada una de las posibles acciones a realizar, el paso del gradiente nos dirá la máxima combinación de $P(\tau, \theta)R(\theta)$ al usar el gradient ascent.

A diferencia de una política determinista, que conllevará la elección de una acción concreta $\pi(s) = a(s)$, la estocástica nos devolverá la probabilidad de llevarse a cabo una determinada acción $\pi(a|s) = P(a|s)$.

En los métodos *policy gradient* y *policy based*, como hemos comentado, la salida de la red neuronal será una distribución de probabilidad de acción. Esto es muy interesante ya que permite realizar un estudio de cada componente del vector para clasificarlas y así puede realizarse un ajuste más preciso de los parámetros de la red neuronal a lo largo de las iteraciones que tienen lugar durante el proceso.

Una de las características más significativa de esta familia de métodos, es que no necesita apoyarse en una función de valor para evaluar cada una de las posibles acciones a tomar por el agente. Evidentemente se puede usar una función de valor para que los parámetros de la política que se vayan obteniendo sean optimizados, pero la elección de la acción no se sostendrá en ninguna evaluación de valores entre las acciones.

A diferencia de los métodos *Q-learning* y derivados como DQN, no se trabaja con los datos obtenidos de una política anterior. Por ellos estos métodos pueden converger muy rápido si comparamos con la otra familia comentada, pero para ello se debe realizar un contacto más continuo con el entorno para obtener la información y datos necesarios.

A continuación, van a comentarse varios aspectos técnicos que deben resolverse o tener claros a la hora de la implementación de esta metodología.

Si analizamos las políticas estocásticas, existen principalmente dos tipos que suelen usarse en el aprendizaje por refuerzo. El uso de una u otra dependerá del tipo de espacio de acción con el que se trabaje. Una política que emplee un espacio de acción discreto se llamará política categórica (*Categorical policy*) y si en cambio tenemos que usar una política con un espacio continuo, se usan las políticas gaussianas (*Gaussian policy*).

Esta clasificación se ha comentado porque la implementación de la probabilística en un algoritmo no será la misma si estamos tratando con acciones discretas o continuas como las matemáticas y estadísticas que se emplean tampoco lo serán.

Como se ha comentado, se trabajará con probabilidades de tomar distintas acciones a la salida de la red neuronal. Si tuviésemos un espacio de acciones discretas, se analizarán cada una de las variables categóricas que tenemos a la salida de la red neuronal mediante el uso de regresión logística[11].

Es importante el uso del ratio probabilístico logarítmico (*Likelihood*) ya que nos da una visión de los resultados obtenidos respecto a los esperados.

$$\log(a|s) = \log\left[P_\theta(s)\right]_a$$

Este ratio se emplea principalmente por motivos de eficiencia computacional ya que es más rápido y ocupa menos memoria calcular desviaciones que diversas sumas de valores gigantescas por lo que supone una herramienta muy eficaz y una alternativa ideal en la interpretación de probabilística.

[11] https://en.wikipedia.org/wiki/Logistic_regression

Se denomina $P_\theta(s)$ de un estado $s$ determinado, al vector de salida de probabilidades que corresponde a la última capa de la red.

La suma de las evaluaciones de todas estas trayectorias en cada uno de los experimentos realizados conforma la recompensa de la interacción del agente con el entorno.

Un problema que surge en este tipo de algoritmos de gradiente es que tiene lugar o se va acarreando una fuerte varianza en los gradientes obtenidos. Como se verá en algunas de las técnicas que se van a comentar, suele aplicarse el concepto de *baseline* [34].

$Var(A) = \sum_{m-1}^{k} Var\,(A_m) = E[(A - E[A])'A - E[A])] = E[(A - E[A])^2]$ , donde A es un vector

$$A = (A_1, A_2, A_3, \ldots\ldots, A_k$$

El uso de este concepto heurístico consiste en aplicar una serie de predicciones con las que compararemos los valores obtenidos de dichos algoritmos. Cuando no se emplea esta técnica estaremos entrenando un modelo que posee relativamente poca capacidad de optimización ya que no tiene una referencia de como de bueno puede ser un resultado obtenido al compararse consigo mismo en otro experimento o iteración por lo que converjan óptimamente es más difícil. Una denominación simple podría ser como que es el estado de algo en un instante de tiempo en un lugar determinado.

Aunque se ha comentado anteriormente, no se ha mostrado como sería el gradiente de la función objetivo. Partimos de la función

$$J(\theta) = \sum_{\tau \sim \pi_\theta} P(\tau, \theta) R(\theta) = E_\pi[R(\tau)]$$

También vimos cómo se iría actualizando la política aplicando el método del gradiente ascendente.

$$\theta_j = \theta_j + \alpha \nabla_\theta J(\theta)|_j$$

Y que mediante el uso del ratio *Log-Likelihood* se puede obtener la distribución de probabilidades de la distribución categórica a la salida de la red neuronal y derivando:

$$\nabla_\theta P(\tau, \theta) = P(\tau, \theta) \nabla_\theta log P(\tau, \theta)$$

Si se estudia el gradiente del último término de esta ecuación tenemos que:

$$\nabla_\theta log P(\tau, \theta) = \nabla_\theta log P(s_0) + \sum_{t=0}^{T} \left( \nabla_\theta log P(s_t, a_t) + \nabla_\theta log \pi_\theta(s_t, a_t) \right) = \sum_{t=0}^{T} \nabla_\theta log \pi_\theta(s_t, a_t)$$

Así que el gradiente de la función objetivo puede simplificarse o quedaría como:

$$\nabla_\theta J(\theta) = \nabla_\theta E_\pi[R(\tau)] = \int_\tau \nabla_\theta P(\tau, \theta) R(\tau) = \int_\tau P(\tau, \theta) \nabla_\theta log P(\tau, \theta) R(\tau) = E_\pi \left[ \sum_{t=0}^{T} \nabla_\theta log \pi_\theta(s_t, a_t) R(\tau) \right]$$

Veremos más adelante como se implementa el concepto de *baseline* en la ecuación del gradiente.

- **REINFORCE**

Los algoritmos tipo *REINFORCE* pertenecen a la familia de algoritmos *policy gradient*. En sí mismo se trata de un procedimiento o método característico que también puede ver se referenciado como Mont-Carlo *policy gradient*. Tiene muchas similitudes con el método *cross-entropy* por lo que a veces podemos encontrarnos con que también se refieran a este método como tal.

Para comenzar, hay que tener en cuenta que aquí se trata de un algoritmo de gradiente ascendente $\theta_j = \theta_j + \alpha \nabla_\theta J(\theta)|_j$, pero veremos otra nomenclatura más idónea para explicar la idea en este caso.

Partimos de la ecuación del gradiente:

$$\nabla_\theta J(\theta) = E_\pi[\sum_a q_\pi(s, a) \nabla \pi(a|s, \theta)]$$ [35]

Vemos como la aproximación del gradiente se define como la suma de todas las experiencias posibles sujetas a una política sujeta a unas acciones en función del estado y las variables paramétricas internas del algoritmo y también aparece una función $q_\pi(s, a)$, que es una función de valor que valorará la recompensa esperada desde un determinado estado $\epsilon$ [0,1] y sujeta a una determinada política.

Si denominamos $S_t, A_t$ como la suma de todos los valores aleatorios posibles de $s$ y $a$, para también introducir otra variable llamada $G_t$ representa la recompensa esperada y parametrizamos la ecuación con t en lugar de j tenemos que:

$$\nabla_\theta J(\theta) = E_\pi \left[ G_t \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \right] \text{[35]}$$

Y que la actualización del gradiente:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \text{[35]}$$

Donde $G_t$ representa la recompensa esperada. Si observamos la expresión con algo de detenimiento veremos que la misma tiene sentido. Aparece un producto entre la recompensa esperada y el gradiente de la probabilidad de tomar una acción entre la probabilidad de tomar una acción. Esta división forma un vector que devuelve la dirección que aumenta en mayor cantidad la probabilidad de que se repita la acción $A_t$ en el espacio de los parámetros $S_t$. Al actualizar vemos que el valor será proporcional a la recompensa esperada e inversamente proporcional al valor de la probabilidad de tomar dicha acción.

El algoritmo *Reinforce* tendría la siguiente forma:

1. Inicializamos la red neuronal con los pesos $\theta$ aleatorios.
2. Se ejecutan N episodios y se guardan las transiciones (s, a, r, s').
3. Luego se calcula para cada t de todos los episodios la recompensa total por descuento:

   $q_t = \sum_{j=0} \gamma^j r_j(s, a)$ donde $\gamma$ representa la tasa de descuento.
4. A continuación, se calcula la función de pérdida del sumatorio total de las transiciones que se han producido:

   $$Loss = -\sum_{k,t} q_{k,t}(s, a) \log \left(\pi\left(a_{k,t}, s_{k,t}\right)\right)$$
5. Se aplica el SGD (descenso por gradiente estocástico) para una actualización de los pesos de la red o función y así se disminuye el $Loss$ previamente calculado.
6. Por último, se vuelve al paso 2 hasta que converjan los resultados.

Básicamente este puede ser un algoritmo genérico tipo *REINFORCE* y la arquitectura común que puede identificarse en algoritmos de la familia.

Como los algoritmos de gradiente de política es un método que suele converger rápidamente, pero para ello es necesario de un gran número de interacciones con el entorno.

Un problema que aparece en la ejecución será la aparición de un considerable valor de varianza influyendo en los valores de los gradientes y parámetros importantes de la red neuronal, etc. influyendo en el comportamiento del algoritmo y en su entrenamiento volviéndolo inestable.

- **REINFORCE CON BASELINE**

El concepto de *baseline* consiste en introducir en la expresión del gradiente una función que dependerá del estado en que se encuentre el sistema y que pueda realizar una valoración o estimación numérica del mismo.

Pueden verse muchos ejemplos de *baselines* en la literatura, puede emplearse una función de valor que nos del valor de un determinado estado, también puede emplearse la media de la recompensa por descuento, algunos valores constantes a los que tienda nuestro sistema, etc…

Idealmente una función de valor $V(s_t)$

En la práctica, si empleamos una función de valor para ejercer de baseline en nuestro algoritmo tenemos que ser conscientes de será una tarea imposible obtener el *valor* exacto, por tanto, se trabajará con una aproximación.

Mediante el uso de una *baseline* ($b$) se representa una expectativa por parte del agente de obtener una determinada recompensa por lo que si se obtiene lo que se espera no debe corregirse ningún error acumulado.

$$b(s)\nabla\sum_a \pi(a|s,\theta) = b(s)\nabla 1 = 0 \text{ [35]}$$

$$\theta_{t+1} = \theta_t + \alpha(G_t - b(S_t))\frac{\nabla\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \text{ [35]}$$

Una forma generalizada de la ecuación del gradiente podría ser:

$$\nabla J = E_{(s,a\to\pi)}[\sum_t \nabla log\pi(a_t|s_t)(\sum_{t'=t} R(s_{t'}, a_{t'}, s_{t'+1}) - b(s_t))]$$ [12]

A continuación, se describirá un algoritmo *policy gradient* muy simple con el que se ha trabajado en los primeros resultados.

- **DISCOUNTED REWARD**

El algoritmo conocido como *Discounted Reward* consiste en un algoritmo tipo *REINFORCE* que trabaja con gradiente que optimiza la política empleando un estimador que evalúa la recompensa a partir del uso de una función que nos devuelve este valor al aplicar de una acción determinada.

Consiste por tanto en una versión del algoritmo *Reinforce policy gradient* en la que se aplica una estimación del gradiente particular.

$$\nabla J = E_{(s,a\to\pi)}\left[\sum_t \nabla log\pi(a_t|s_t)(\sum_{t'=t} R(s_{t'}, a_{t'}, s_{t'+1}))\right]$$

Donde $\sum_{t'=t} R(s_{t'}, a_{t'}, s_{t'+1}) = Q_t$ es un estimador de una política particular sobre la función $Q^{\pi_\theta}$.

Uno de los de las carencias más importantes que aparecen en este tipo de algoritmos es una inestabilidad que aparece en el gradiente al trabajar en reiteradas iteraciones donde se va acarreando una diferencia entre el resultado de este y el valor esperado. Por ello como se ha comentado anteriormente es muy usual aplicar como reducción de esta varianza mediante la técnica del *baseline*.

- **ACTOR CRITIC**

*Actor Critic* es una denominación para la familia de métodos o algoritmos que contienen características de los métodos *policy gradient* y se consideran una versión dista de estos. Son referidos como una diferencia temporal del método *policy gradient*.

Los algoritmos *Actor Critic* trabajan con dos elementos clave el actor y el crítico donde el primero elegirá la acción a tomar y el crítico informará de la bondad de la acción y como modificarla para obtener el mejor resultado posible. Como método *policy gradient* que es, el aprendizaje del actor estará basado en el ajuste de una política óptima, pero el otro participante de este método es quien considera el valor de la decisión tomada por el actor mediante una función de valor.

[12] Ecuación basada en fuente https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html#deriving-the-simplest-policy-gradient.

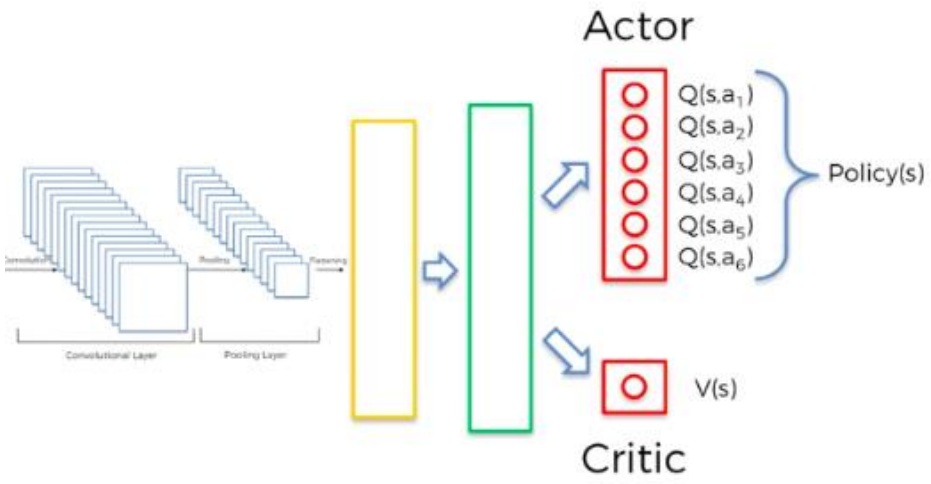



**Figura 4.5:** Modelo representativo de un algoritmo tipo Actor Critic[13].

Podemos encontrar muchas similitudes con el *REINFORCE* con *baseline* en las expresiones matemáticas que rigen al *Actor Critic*.

$$A(s_t, a_t) = \sum_{t\prime=0}^{T-1} r' - b(s_t)$$

La función de *Advantage* donde se refleja el valor de la recompensa obtenida menos $b(s_t)$ que representa el valor de recompensa del estado actual.

$$b(s_t) = V_{\pi_\theta}(s_t)$$

Por lo que la ecuación de la *Advantage* aplicando una determinada política quedaría finalmente:

$$A_{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)$$

Obteniendo la ecuación del gradiente siguiente:

$$\nabla J = E_{(s,a\to\pi)}\left[\sum_t \nabla log\pi(a_t|s_t) A_{\pi_\theta}(a_t|s_t)\right] =$$

$$= E_{(s,a\to\pi)}\left[\sum_t \nabla log\pi(a_t|s_t) r(s_t, a_t) + V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)\right]$$

El flujo generalizado de los algoritmos pertenecientes a esta familia de algoritmos sería el siguiente:

1. Se toma una muestra del estado y acción relativa $s_t$ y $a_t$.
2. Evaluaremos la función de ventaja anteriormente mostrada ().
3. Se evalúa la ecuación del gradiente y se actualizan los parámetros de la política.

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta)$$

4. Se actualizan los pesos de la red para que la parte crítica emplee una función de valor actualizada.

$$w = w + \alpha\, \delta$$

[13] Fuente: https://medium.com/@shagunm1210/implementing-the-a3c-algorithm-to-train-an-agent-to-play-breakout-c0b5ce3b3405

5. Se repetirá el proceso hasta obtener la política correspondiente al mínimo absoluto del gradiente y por lo tanto la más adecuada.

Este método realiza un estudio del error producido respecto a la función de valor conocido como diferencia temporal para reducir la varianza que se produzca en la evolución de la ejecución del algoritmo.

Existen muchos algoritmos interesantes como el A2C y el A3C que son muy potentes y estables y pueden trabajar en entornos muy dispares y con problemas arduos.

- **PROXIMAL POLICY OPTIMIZATION**

Este algoritmo conocido también por su acrónimo PPO (Optimización de políticas próximas). Se trata de una familia de métodos que se sirve de algunas técnicas para que trabajando con la base de datos se consiga realizar pasos o *steps* trabajando con una misma política sin que se produzca una divergencia o problemas de optimización. Trabaja con una optimización de primer orden.

Al igual que los algoritmos que hemos estado viendo se trata de un algoritmo tipo *policy iteration* y también fue diseñado para que pudiera emplearse en entornos continuos y discretos.

Pueden destacarse dos clases importantes a la hora de clasificar las variantes de este algoritmo.

En uno de ellos aparece el parámetro PPO-*Penalty* mediante el cual se busca solucionar la divergencia de Kullback-Leiber (DKL o KL-*divergenc*e), que nos dice la distancia estadística entre dos distribuciones de probabilidad, que son las que se recogen de la red neuronal, mediante una serie de restricciones y que representará la divergencia existente entre ellas. Mediante el ajuste de este parámetro en el entrenamiento de la red se sintonizará adecuadamente la reducción de la varianza en el gradiente.

La otra clase comúnmente denominada PPO-Clip, o *clipping*, que en este caso no realiza restricciones a la divergencia, actúa sobre la función objetivo impidiendo que en el desarrollo del algoritmo se obtenga una política que sea muy diferente a la anterior mediante un proceso de clipeado o como se denomina en lenguaje anglosajón *clipping*. El *clipping* consiste en realizar una reducción o recorte en el ratio de probabilidad, que consiste en una variable que nos da información de como de favorable o desfavorable será una política en función de otra anterior.

$$r_t(\theta) = \frac{\pi_\theta(a,s)}{\pi_{\theta_{old}}(a,s)}$$

Vamos a centrarnos en esta clase de algoritmo PPO en el desarrollo de nuestro trabajo por lo que se profundizará las ecuaciones o fórmulas que rigen el funcionamiento de este algoritmo.

La función objetivo a maximizar sería la siguiente:

$$L^{CLIP}(\theta) = E[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t]$$

Donde $\varepsilon$ consiste en un hiperparámetro del algoritmo y $A_t$ una función de estimación antes vista. Tras esto veremos como quedaría la actualización de parámetros de la política.

$$\theta_{t+1} = argmax_\theta\ E[L(s,a,\theta,\theta_t]$$

Si volvemos a la ecuación objetivo anterior de $L^{CLIP}(\theta)$ vemos que varía en función del valor positivo o negativo de la función de estimación $A_t$. Si tenemos que $A_t > 0$ el ratio $r_t$ se clipeará o recortará para $1+\varepsilon$ y si tenemos que $A_t < 0$ clipeamos en $1+\varepsilon$, finalmente se tomará el valor menor del objetivo cuando se somete al recorte y cuando no se le realiza para obtener la función objetivo con la que trabajaremos.

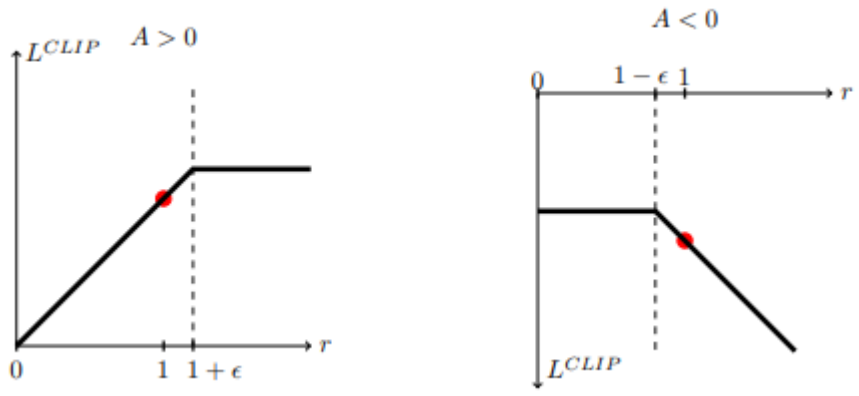



**Figura 4.6:** Descripción gráfica del proceso de *clipping*[14].

El algoritmo PPO obtiene políticas estocásticas a partir de la experiencia recopilada durante el entrenamiento. Empezamos con una política que se puede caracterizar como aleatoria debido a que los parámetros obtenidos de las condiciones iniciales no provienen de un proceso de optimización y suelen ser bastante malos. A partir de la exploración y las acciones a realizar sujetas a las políticas presentes en cada paso de iteración, se va obteniendo una mejor política con parámetros más idóneos que nos devuelve mejores valores de recompensa. Surge un problema asociado a este proceso y es que a medida que vamos obteniendo una mejor política, dejamos de dar "grandes saltos" en los parámetros para obtener una nueva, por lo que si se alcanza un mínimo local con una política determinada es difícil salir de él para alcanzar el mínimo absoluto.

El flujo del algoritmo PPO sería el siguiente:

1. Inicializamos la red neuronal con los parámetros $\theta$ aleatorios, así como a la función de evaluación
2. Bucle for i=0 : 1: Nmáx
    a. Se almacenarán los resultados de la política i obtenidos en el entorno, así como las trayectorias y acciones $D_i$=[ $\tau$].
    b. Se calculará el término $R_t$ de recompensa acumulada.
    c. Se calcula el termino $A_t$ mediante algún método de estimación basado en la función de valor.
    d. Actualizamos la política siempre priorizando maximizar el clipeado y su función objetivo:

$$\theta_{i+1} = argmax_\theta \, E[L(s, a, \theta, \theta_t]$$

$$\theta_{i+1} = argmax_\theta \frac{1}{D_i T} \sum_{\tau \in D_i} \sum_{0}^{T} \min \left( \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_i}(a_t, s_t)} A^{\pi_{\theta_i}}(a_t, s_t), clip(\frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_i}(a_t, s_t)} A^{\pi_{\theta_i}}(a_t, s_t), 1-\varepsilon, 1 + \varepsilon) A_t \right)$$

    e. Ajustamos la función de valor respecto a la recompensa acumulada usando el descenso del gradiente para minimizar la diferencia:

[14]Fuente: https://arxiv.org/pdf/1707.06347v2.pdf

$$\emptyset_{i+1} = argmin_{\emptyset} \frac{1}{D_i T} \sum_{\tau \in D_i} \sum_{0}^{T} (\mathrm{V}_{\emptyset}(s_t - R_t))^2$$

3. Cerramos el bucle for.

En el apartado 2, dentro del bucle (punto d) debemos trabajar con el gradiente que se irá actualizando estocásticamente mediante la técnica del gradiente ascendente como comentamos al buscar un máximo para nosotros de recompensa acumulada.

En el punto "e" se emplea la fórmula del error cuadrático medio para ajustar la función ya que es bastante simple y fácil de trabajar con ella, pero puede trabajarse con otra diferencia.

### 4.5. Aplicación de Deep Reinforcement Learning en la planificación de rutas.

Tras analizar los tipos de algoritmos, métodos y familias de algoritmos, se diseña el empleo de los algoritmos *REINFORCE* y PPO para afrontar la tarea de aprendizaje automático del ASV, ya que, debido a las características del entorno de trabajo, a priori se esperan los mejores resultados posible de este último. Se selecciona el algoritmo *REINFORCE* conociendo previamente unas limitaciones de este frente al PPO debido a que es interesante analizar la respuesta al problema que se obtiene mediante un algoritmo algo simple tipo *on-policy* para después analizar la respuesta de otro mucho más potente y actualizado a tareas complejas como la que se plantea.

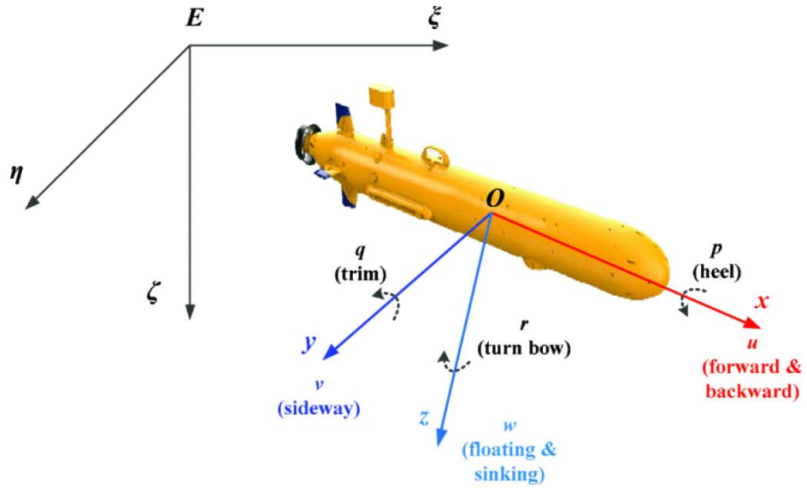



**Figura 4.7:** Planificación de rutas de AUV mediante el uso de algoritmo PPO [36].

Se ha realizado una introducción superficial a los métodos basados en *policy gradient*. Comenzando desde el término *Machine Learning*, como se conoce al conjunto de técnicas en las que se basa el aprendizaje automático, se ha presentado una visión generalizada de unas subcategorías que nos han llevado a alcanzar la temática más particular en torno a la que se centra este trabajo. Es cierto que existen otros algoritmos menos conocidos o variaciones de estos que no se han recogido en este capítulo, pero mediante esta sección se puede observar el funcionamiento de los algoritmos con los que hemos trabajado y experimentado para afrontar la tarea de exploración y toma de muestras en el lago Ypacaraí.

Particularmente, el problema que se pretende abordar con este tipo de técnicas puede denominarse como de exploración orientada por motivación intrínseca. Cuando la interacción del agente con su entorno se ve

recompensada positiva o negativamente y se pueden detectar los elementos de un MDP de dicho sistema, resulta bastante apropiado el uso de técnicas DRL en la resolución de dicho problema ya que tiene un enfoque muy adecuado para afrontar problemas de exploración y explotación. Mediante métodos basados en gradiente de política abordaremos la resolución del sistema con unos puntos destacables.

Para comenzar se debe optimizar una política parametrizable que definirá la estrategia más adecuada del agente para la resolución del problema de exploración durante la fase de entrenamiento. La modificación de estos parámetros($\theta$) vendrá asociada a un proceso iterativo de aprendizaje mediante la experimentación del agente en su entorno. El criterio de búsqueda de una política más adecuada vendrá definido por la maximización de la recompensa obtenida por el agente en el ejercicio desarrollado, tiene sentido que por ello se pretenda conseguir una política que permita que el vehículo obtenga una recompensa mayor.

$$\theta_j \leftarrow \theta_j + \alpha \nabla_\theta J(\theta)$$

Dicha actualización de parámetros y política emplea hace uso del gradiente de la función de estimación $J(\theta)$ que consiste en la evaluación del grado de optimalidad de los parámetros empleados o una función de valor. Mediante el uso del gradiente aseguramos que los nuevos parámetros y por lo tanto política asociada irá mejorando tras la mencionada actualización. Como se ha mencionado anteriormente, existen varias técnicas de optimización en el DRL y por lo tanto varias pequeñas familias de algoritmos particulares, pero todos comparten la mecánica de optimización mediante el uso de distintas técnicas de derivación aplicados en funciones de valor.

Las políticas generadas a partir del uso de las conocidas redes neuronales, y los parámetros previamente comentados, pueden adaptarse a problemas con un alto grado de complejidad ya que dichos elementos son adaptables y configurables para dicha tarea. En el siguiente apartado se describirán brevemente la tipología de las mismas. Un aspecto por ello muy importante en la resolución de tareas de exploración mediante *Deep Reinfocement Learning* por tanto será el diseño de una ley de recompensa adecuada que se adapte a las características del sistema y que recoja todas las particularidades que puedan tener lugar durante el desarrollo de cualquier episodio durante el entrenamiento o ejecución de la red neuronal o el modelo. Dicha ley representará el comportamiento del agente y del sistema en función del estado del sistema tras diversas actuaciones por parte del agente que vendrán devenidas por la acción seleccionada de las que provee la red.

Los algoritmos de esta familia proponen la mejora de la política realizando una optimización de esta mediante la comparación de la misma con una versión anterior directamente o con la comparación de la función de ventaja de la misma manera habiendo dos grandes ramas, *policy based o value based*. En este trabajo se emplean algoritmos de la primera familia como se verá más a delante en la sección de resultados.

Es una técnica muy potente que es capaz de trabajar con entornos de diversas características, continuos o discretos y de un grados de complejidad alto. Pero es importante tener en cuenta que pueden producirse episodios de una duración “casi interminables” debido a una mala planificación del algoritmo adecuado y el grado de precisión empleados.

Si seguimos viendo los aspectos del uso de la técnica DRL en problemas de exploración, es importante comentar que la aparición de errores asociados a la varianza del gradiente y en las predicciones de los estados que surgen tras el hecho de realizar una acción y producirse un *step* en el sistema. Por ellos es muy común el uso de elementos que reducen estas componentes como por ejemplo el *baseline*, *clipping* o reductores de divergencia KL. Dichos elementos han sido comentados previamente en el punto 4.3.1 en el algoritmo que lo presente en su funcionamiento.

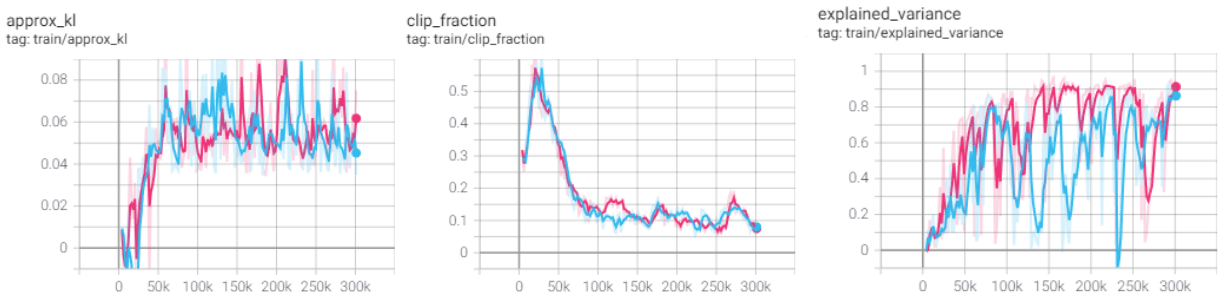



**Figura 4.8:** Ejemplos de la presencia de errores en sistemas abordados mediante tecnicas DRL.

La metodología empleada por tanto es realizar un entrenamiento lo más adecuado posible para obtener la mejor política posible y el modelo que se extrae ejecutarse convenientemente en diversas simulaciones. Como su nombre indica el aprendizaje por refuerzo trata de aprender de experiencias pasadas para ir mejorando, aunque a veces ocurra que computacionalmente sea poco eficiente. Para eso el avance de la técnica está en desarrollo ya que el uso de estos algoritmos cada vez se aplica en problemas más complejos.

A continuación, nos referiremos a la tarea de experimentación realizada tras la selección e implementación de los algoritmos seleccionados, así como la presentación de conceptos y elementos importantes de los mismos.

### 4.5.1. Redes Neuronales Convolucionales

Una red neuronal tiene la capacidad de trabajar con un tensor de datos. Se trata de una ampliación respecto a una red neuronal más básica y veremos en que consiste. Principalmente su uso está relacionado con el tratamiento y procesamiento de imágenes, matrices de datos $M \times N$, en diversas canales que podrían ser 3 como en los RGB uno para cada color.

La red convolucional está conformada por una serie de capas de neuronas semejantes a las redes básicas más otros dos tipos de capas:

- **Capas de convolución:** básicamente se trata de aplicar un filtro enmascarándola. En estas capas se tomar grupos de datos (pixeles) que estén cercas y realizar un producto escalar con otra matriz llamada *kernel*. Tras esta operación se obtiene otra matriz que puede ser otra vez procesada por más capas de convolución o a la siguiente capa del proceso. Estas capas junto a las de *pooling* suelen denominarse filtros y suelen ser empleados en el reconocimiento de formas y objetos en imágenes.

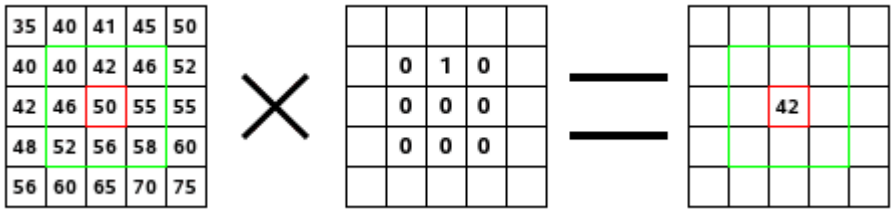



**Figura 4.9:** Ejemplo básico del funcionamiento de los filtros *kernel*.

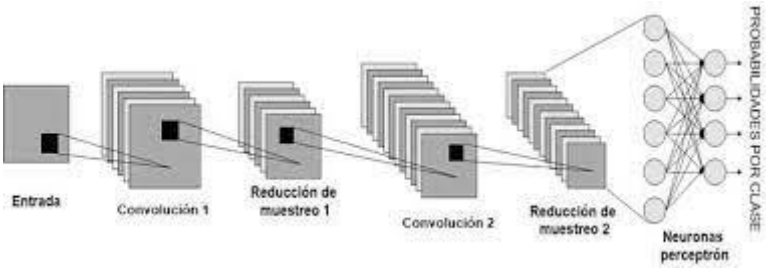



**Figura 4.10:** Ejemplo de estructuración interna de capas neuronales en una red convolucional[15].

- **Capas de *pooling*:** estas capas suelen situarse a posteriori de las convolutivas. Al igual que el concepto de *pooling*, la reducción dimensional, estas capas reciben las matrices salientes del *kernel* y se encargan de reducir su número de elementos y como consecuencia de ello habrá una menor tarea de cálculo en capas a posteriori.
  El *max-pooling* por ejemplo se encarga de obtener el máximo valor de cada subregión de las matrices de entrada al quedar dividida en menos regiones.

  Un problema derivado que puede darse lugar del *pooling* es que puede perderse información de la ubicación de los elementos tras varias etapas de este tipo de capas.

Entre las capas de convolución, entrada y salida, se aplican las llamadas funciones de activación. Dichas funciones se encargan de aplicar un filtro a la información que entra y sale de dichas capas. La información que transmitirán será la combinación de pesos y filtros *kernel* o entradas que es necesario transformar para que la red desarrolle un modelo no lineal y por tanto su uso permitirá abordar problemas más complejos. Funciones muy comúnmente empleadas pueden ser la Relu o lineal, escalón, sigmoide, tangente hiperbólica, etc.

Existen tipos de redes capaces de trabajar con entornos tridimensionales y con unas entradas de tamaño considerable. En este trabajo se hará uso de redes tipo *Actor Critic*, un tipo de red empleada para entornos de trabajo 2D complejos y el uso de algoritmos tipo A2C y PPO.

En nuestro algoritmo PPO emplearemos una red neuronal convolucional tipo *Actor Critic* con una función de activación tipo Tanh y optimizador Adam. Como entrada tomará el estado del entorno de trabajo en diversas capas de información, ya que el mismo posee varias capas para ello. La última capa de la red con ocho salidas correspondientes a las componentes del vector de acciones que se referirán a la probabilidad de tomar cada una de las direcciones identificables como las cardinales y las diagonales a las mismas.

### 4.5.2. Optimizador Adam

En la tarea de entrenar una red neuronal, se debe realizar una optimización de la función de coste asociada al problema a resolver. Para ello se realizan una serie de entrenamientos de la red neuronal donde se busca obtener una configuración de los pesos con los que trabaja adecuados.

Mediante un optimizador es como se realiza el cálculo de dichos pesos. El optimizador se encarga de resolver el gradiente de la función a optimizar por cada parámetro o peso que contenga la red neuronal minimizando el error presente y obteniendo un mejor resultado o recompensa (*gradient ascent*).

La actuación del optimizador se desarrollará en cada paso del algoritmo donde se emplee y está estrechamente relacionado al uso del parámetro *learning rate* ya que se debe tener en cuenta las experiencias o configuraciones de pesos anteriores.

[15]Fuente: https://riull.ull.es/xmlui/bitstream/handle/915/10422/Usando%20redes%20neuronales%20convolucionales%20para%20convertir%20caracteristicas%20visuales%20en%20estimulos%20sonoros.pdf?sequence=1

En nuestro caso emplearemos el optimizador Adam, que aplica una tasa de aprendizaje escalada a cada peso de la red neuronal. Dicho escalado se realiza a partir del primer momento del gradiente acumulado, o media, obtenido por *step* y también a partir del segundo momento limitando la varianza.

La formulación sería la siguiente[16]:

$$m = \beta_1 * m + (1 - \beta_1) * \Delta w$$
$$v = \beta_2 * v + (1 - \beta_2) * \Delta w^2$$
$$w = w - \frac{\alpha * m}{\sqrt{v + \varepsilon}}$$

Los términos $m$ y $v$ representan los dos momentos anteriormente comentados encargados de modelar la media de los gradientes y su varianza. $\beta_2$ $y$ $\beta_1$ $\in [0\ ,1]$ son dos parámetros encargados de incrementar la acción de los momentos sobre los pesos automáticamente.

El uso del optimizador Adam (*Adaptative Moment Estimation*) en algoritmos de *Reinforcement Learning* está muy extendido, ya que supone un avance importante respecto a los optimizadores anteriormente empleados como el SGD (*Stochastic Gradient Descent*), el Adagrad o el RMSprop.

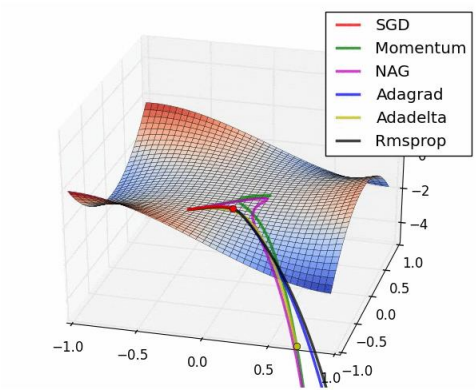



**Figura 4.11:** Resultados de convergencia del optimizador Adam frente a otros[17].

En nuestro caso emplearemos el uso del optimizador Adam en los distintos algoritmos actualizará los valores del parámetro *learning rate* a lo largo del rango del buffer de datos de cada *epoch* para que el gradiente realice los *steps* más adecuados en el aprendizaje o entrenamiento de la red neuronal.

### 4.5.3. Ley de recompensa

La ley de recompensa consiste en una función donde se definirán las condiciones, criterios y/o comportamiento de la recompensa generada en cada uno de los estados posibles que pueden darse lugar a lo largo de los entrenamientos o experimentos en el entorno diseñado. Dicha ley reconocerá una bonificación positiva al realizarse la exploración o visita de alguna zona transitable del mapa y en cambio se generará una recompensa negativa en el caso que pretenda alcanzar alguna zona no transitable del mapa donde trabajemos.

Es evidente que para lograr una buena función que modele el comportamiento de la recompensa en los distintos escenarios posibles, se debe trabajar con un escalado numérico a la hora de designar como de positivo o negativo será una acción que realizar. No será igual de positivo realizar la visita a una casilla recientemente

[16] Fuente: https://datasmarts.net/es/que-es-un-optimizador-y-para-que-se-usa-en-deep-learning/#
[17] Fuente: https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-pr%C3%A1ctico-819b39a3eb5

visitada como a una que no lo haya sido.

Como la recompensa dependerá de la acción realizada por el agente y del estado del escenario veremos como al igual que aspectos comentados anteriormente como la función de valor y la política, puede definirse en función de dichas variables.

$$F_R = R(s, a)$$

Es cierto que el uso de una función de recompensa es muy útil a la hora de realizar entrenamientos mediante algún método de aprendizaje por refuerzo porque estamos condicionando al algoritmo para que tenga en cuenta en mayor o menor medida como comportarse y trabajar. Como condicionamiento positivo le decimos como de bueno será realizar un movimiento hacia cierta posición positiva y como de malo será realizar otro hacia una posición negativa. El problema que surge de esto es que también se está condicionando al algoritmo dejar de visitar casillas desfavorables limitando el proceso de exploración por lo que se debe diseñar esta función con precaución sin determinar recompensas extremas, en cualquier caso. Estos casos pueden darse principalmente en lugares próximos a las zonas no transitables del mapa ya que la experiencia recopilada en dicha zona puede convertirse en muy desfavorable y por tanto no ser interesante para ser visitada por el modelo entrenado a través de dicho algoritmo.

Teniendo en cuenta las características de nuestro entorno, considerando como anteriormente ha sido comentado, queda claro que un lago no es un ecosistema homogéneo. Existen distintas propiedades a distinta profundidad y ubicación, pueden existir zonas donde se favorezca la concentración de distintos microorganismos o sustancias, por lo que resulta interesante trabajar con un mapa "de importancia no homogénea", para la visita del vehículo.

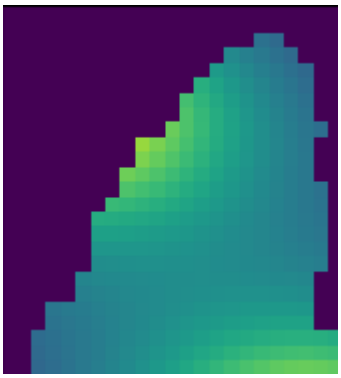


**Figura 4.12:** Detalle de mapa de importancia no homogénea.

En la figura 4.11 puede apreciarse como diversas zonas son más interesantes para la visita que otras representando dicha situación con distintos niveles de intensidad.

Para generar esta característica en el mapa, trabajaremos con una matriz de importancia o *idleness* que almacenará la información relacionada a la importancia de las características introducidas en el escenario. En otra matriz se almacenará un mapa de las características donde se almacenará la posición de estas y el estado del mapa $T$,. En resumen, tenemos una matriz $C(i,j)$ cuyas casillas almacenarán el estado instantáneo del entorno y además cuyos valores $\in [0,255]$. Dicha matriz estará compuesta por otras dos, una ya comentada donde se almacenará $T$ y otra donde se reflejará la importancia de las casillas $I$ con valores $\in [0,1]$.

$$C(i,j) = T(i,j) * I(i,j)$$

La matriz $C(i,j)$ trabajará tras cada actualización del entorno con una representación numérica del estado de este, donde a través del producto de las dos matrices según la formula anterior se realizará un escalado de los

valores de las casillas de $T$ entre 0 y 1. También es importante comentar que se actualizará el valor de las casillas en función de los movimientos realizados por el agente. Cuando una casilla ha sido visitada, su valor de *idleness* se reduce a cero ya que no tiene interés volver a visitarla a corto plazo. Por supuesto también debe a haber una tasa de recuperación de dicha importancia a lo largo del tiempo tratando de representar las circunstancias reales de trabajo en campo.

Como característica adicional, se ha querido introducir un pérdida constante de intensidad de los valores de importancia de las características del mapa en la ley de recompensa de un 20% de las casillas al ser visitadas.

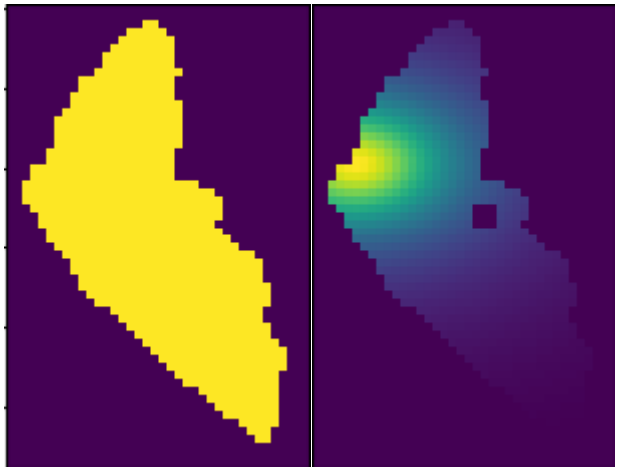


**Figura 4.13:** Representación de las matrices $T$ $y$ $C$.

A continuación, se detallarán los detalles de la ley de recompensa desglosados en función de las limitaciones que se le requieren imponer para ser adecuada al entorno de trabajo.

Se ha estimado que la autonomía máxima del vehículo es de 30 kilómetros, aunque esa variable puede modificarse, y:

$$nº\max movimientos = \frac{2 * autonomía(km)}{30} * nº\ casillas\ del\ mapa$$

La batería del vehículo se contabilizará en una fracción porcentual [0,100] en función del número máximo de movimientos.

| Detalles de ley de recompensa | |
|---|---|
| Tipo de casilla | Recompensa |
| No visitable (colisión) | -10 |
| Visitable por primera vez | $R_{máx}$ |
| Visitada anteriormente | $R_{mín}$ |

**Tabla 4.1**: Detalles de la ley de recompensa.

La tasa de recuperación de la importancia de las casillas también dependerá inversamente proporcional al

número máximo de movimientos posibles a realizar.

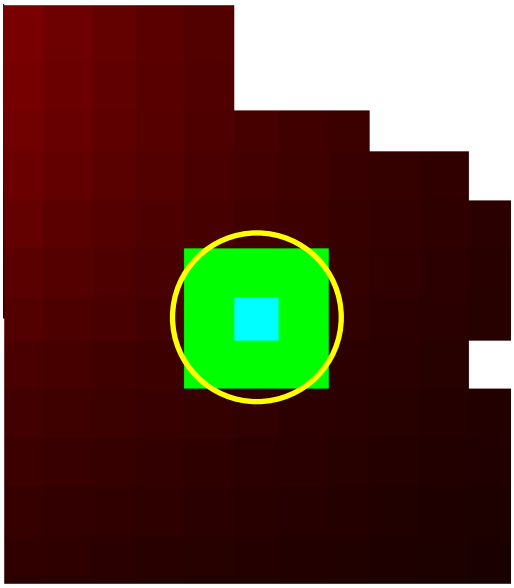


**Figura 4.14:** Zona de influencia (verde) de la toma de datos del vehículo.

La recompensa obtenida será igual a la suma de la importancia de las casillas alcanzadas por la zona de influencia del vehículo dividido entre la longitud de la circunferencia de representa.

$$R = \frac{Suma\ de\ importancia\ perimetral}{2\pi * radio\ circunferencia},$$

donde la suma de importancia descrita será la suma de los 8 términos de las casillas perimetrales al vehículo cuyos valores serán $= \epsilon\ [0,1]$.

Con dichas consideraciones, entendemos cómo se adoptan ciertos comportamientos por el algoritmo a aplicar y como se tenderá a que el vehículo ocupe casillas no visitadas previamente a corto plazo y a realizar movimientos perpendiculares (NE,NO,SE,SO) ya que así se visitan un mayor número de casillas nuevas por el área de influencia y por lo tanto se conseguirá una recompensa mayor. Por supuesto lo que queremos evitar son las colisiones que se producirían con las zonas no transitables, ya que sería la situación más desfavorable.

### 4.5.4. Algoritmos empleados

En este trabajo basado en el aprendizaje por refuerzo profundo, tras analizar el estado de la técnica y trabajos previos relativos a la toma de muestras mediante un protocolo autónomo basado en *Machine Learning*, se ha decidido estudiar el uso de unos algoritmos pertenecientes a la familia de los *policy gradient* dentro del *Reinforcement Learning*.

- **Discounted Reward:** es un algoritmo tipo *REINFORCE* muy sencillo basada en el uso de optimización de gradiente que emplea el uso de políticas, que son actualizadas en un proceso de mejora iterativo, para tratar de obtener el máximo valor de recompensa posible a partir de la experiencia obtenida cuyo pseudocódigo se muestra a continuación[18].

[18] https://stjohngrimbly.com/model-free-RL/

**Algorithm 1** REINFORCE

**procedure** REINFORCE
  *Start with policy model* $\pi_\theta$
  **repeat:**
    *Generate an episode* $S_0, A_0, r_0, \ldots, S_{T-1}, A_{T-1}, r_{T-1}$ *following* $\pi_\theta(\cdot)$
    **for** $t$ *from* $T-1$ *to* 0:
      $G_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_k$
    $L(\theta) = \frac{1}{T} \sum_{t=0}^{T-1} G_t \log \pi_\theta(A_t|S_t)$
    *Optimize* $\pi_\theta$ *using* $\nabla L(\theta)$
**end procedure**

- **PPO:** consiste en un algoritmo de la misma familia que el anterior basado en derivación del gradiente que busca la optimización de la política presente mediante su actualización de forma iterativa. Como peculiaridad, la actualización de la política respecto a la anterior se realiza en pasos bastante cercano mediante el uso de integración de primer orden consiguiendo una carga computacional bastante leve teniendo en cuenta la complejidad de las tareas. A continuación, se muestra el pseudocódigo de este algoritmo[19].

**Algorithm 1** PPO-Clip

Input: initial policy parameters $\theta_0$, initial value function parameters $\phi_0$
**for** $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function $V_{\phi_k}$.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left( V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
**end for**

Se considera que el uso de un algoritmo bastante simple como el *REINFORCE*, dentro de la familia, y posteriormente otro más desarrollado como el PPO, puede ofrecer una visión interesante acerca de los resultados de afrontar el problema planteado con un método basado en optimización de políticas para luego analizar dichos resultados con los obtenidos con un algoritmo más potente como el PPO del que se ha hablado.

[19] https://spinningup.openai.com/en/latest/algorithms/ppo.html

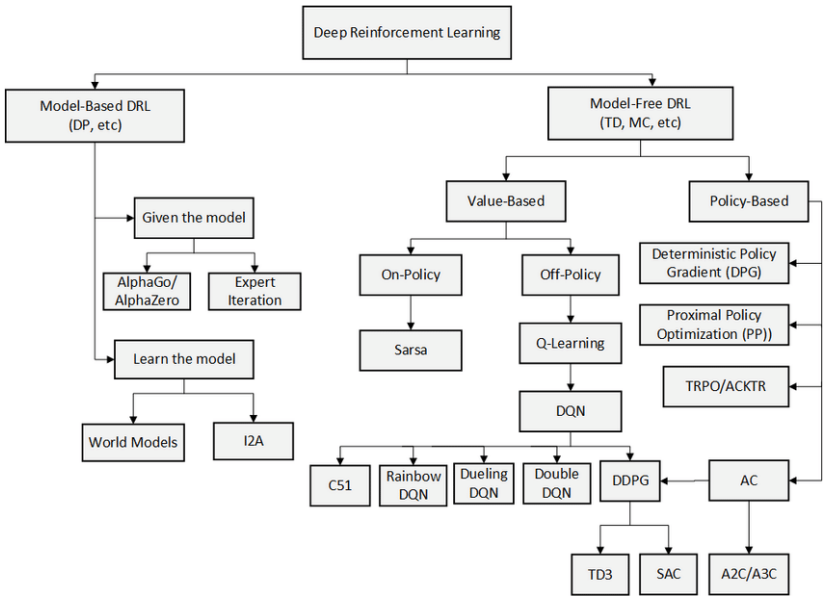



**Figura 4.15:** Esquema estructuración algoritmos de la familia de aprendizaje por refuerzo profundo [44].

Adicionalmente se han diseñado 3 algoritmos adicionales con el fin de crear una base desde la que se pueda analizar la eficacia o la no eficacia de la aplicación de los anteriores en el sistema.

- **Trayectoria *lawn-mower* concéntrica:** este algoritmo pertenece a una familia, *lawn-mower*, bastante conocida y empleada en áreas técnicas como la robótica y navegación autónoma. Las características principales que pueden destacarse son una arquitectura simple y eficiente, la búsqueda de una cobertura elevada de la superficie de trabajo y una implementación muy estable en distinto dispositivos móviles como vehículos no tripulados. Particularmente planteamos un algoritmo que desarrolla una trayectoria en forma de espiral partiendo desde un punto central en la que se trata de que el móvil o agente, o su zona de influencia, visite al menos una vez toda la superficie disponible.
- **Trayectoria *lawn-mower* longitudinal:** en este caso se diseña una algoritmo, también *lawn-mower*, que promueve los desplazamientos longitudinales del agente. Al igual que la versión concéntrica previamente descrita se trata de un algoritmo simple que analiza pocos características del entorno.

Los algoritmos *lawn-mower* son adaptativos al entorno de trabajo, un input clave serán las dimensiones de la superficie donde trabajan los móviles en los que el *path planning* está gobernado por esta familia. Se encuentra muy presente en tareas de exploración donde no se pretende valorar zonas o regiones que tengan una importancia mayor a otras. Comúnmente conocidos como algoritmos cortacésped pueden encontrarse ejemplos de su aplicación en estos dispositivos autónomos y rastreo de zonas en conflictos donde el rastreo de dispositivos explosivos supone un riesgo al ser humano como ejemplo de uso militar. Las limitaciones que presenta esta técnica son la evaluación del medio limitada por lo que el índice de eficiencia de los dispositivos que trabajan con dicha generación de rutas suele ser usado como técnicas alternativas o como referencia en tareas de desarrollo de técnicas más eficientes y especializadas.

- **Trayectoria aleatoria:** se pretende que las trayectorias generadas por este algoritmo sean casi totalmente aleatorias. Específicamente las acciones que realiza el móvil donde se implemente este *path planning* serán generadas mediante un algoritmo aleatorio que no tiene en cuenta la posición de este pero que puede estar sujeto de ciertas leyes o normas a las que debe regirse para una correcta adaptación a la superficie de trabajo en la que se encuentre el vehículo. Dichas limitaciones se introducen al sistema mediante dispositivos analógicos de detección de obstáculos como sensores fotoeléctricos o de presencia para garantizar la integridad del vehículos o del entorno de trabajo.

### 4.5.5. Estructuración del código

Para el desarrollo de este trabajo tal y como se ha comentado anteriormente, se ha priorizado la buena organización y portabilidad del código. Por ello se han creado varios scripts independientes:

- Desarrollo del código que conforma el entorno de simulación.
- Creación de la red neuronal.
- Entrenamiento de la red neuronal mediante el uso del algoritmo *Discounted Reward* (*REINFORCE*).
- Entrenamiento de la red neuronal mediante el uso del algoritmo PP0.
- Pruebas a los modelos obtenidos previamente en episodios controlados y presentación de resultados.
- Representación de datos experimentales.

# 5. RESULTADOS

Con el objetivo de obtener un modelo adecuado del agente realizando la tarea abordada, se plantea el entrenamiento de una red neuronal mediante el uso de distintos algoritmos de aprendizaje por refuerzo. Se busca obtener un agente entrenado que obtenga la máxima recompensa posible en un entorno definido en el que se aplicarán distintas restricciones.

El problema que se aborda es la exploración de un medio acuático interior mediante un vehículo acuático.

En este apartado se mostrarán los resultados obtenidos y el comportamiento de nuestro sistema que trata de recrear el lago Ypacaraí y un vehículo ASV realizando tareas de recogida de datos. Para ello es necesario trabajar en un entorno que trate de recrear lo más fielmente posible las características y condiciones presentes en dicho emplazamiento. Para lidiar con esa necesidad se ha desarrollado un entorno virtual donde se recrean diversas condiciones, reglas y normas para que un vehículo móvil, que representará al ASV, consiga recrear un comportamiento adecuado que pueda representar al hipotético comportamiento que tendría lugar en el entorno real simulado.

En este apartado se plantearán los siguientes puntos:

- Descripción del entorno
- Estructuración de resultados
- Resultados obtenidos
- Análisis de los resultados
- Comparación entre metodologías

Mediante esta estructuración se pretende mostrar paso a paso el trabajo realizado.

## 5.1. Descripción del entorno

Una pieza clave a la hora de entrenar un modelo es el entorno de trabajo. Dicho entorno determinará lo buena o mala en función de la recompensa obtenida es una acción y por tanto las experiencias recopiladas por el agente. En este caso se pretende trabajar en un emplazamiento físico acuático correspondiente al lago Ypacaraí y se debe caracterizar varios aspectos de este como el perímetro de la zona navegable, posibles obstáculos presentes, etc. Como nuestro emplazamiento consiste en una superficie plana correspondiente a un lago, consideraremos como la opción más adecuada trabajar con un mapa de dos dimensiones. La idea es caracterizar la superficie del lago mediante una matriz que a través del valor de sus celdas variables permita discretizar el estado del entorno que se pretende representar. El rango de dicha matriz será un punto de importancia que hay tratar con delicadeza ya que un número muy elevado de celdas puede generar una precisión elevada del entorno físico pero una carga computacional importante.

Será necesario que en el mismo se especifiquen las condiciones de trabajo en las que el vehículo simulado debe encontrarse.

Es interesante tomar una consideración más simplificada, número de celdas menor y una caracterización con menor precisión, para obtener un modelo entrenado más eficiente siempre buscando un equilibrio entre precisión y resultados que genere resultados correctos.

Para ello el código del entorno realizará una actualización de su propio estado donde se actualizará el valor de todas las casillas y en las visitadas se producirá una regeneración constante en cada step hasta alcanzar su valor inicial.

Es usual contar con una componente aleatoria en los valores de las casillas cuando se quiere trabajar con un emplazamiento del que se desconoce perfectamente es estado real. Dicho esto, no supone una particularidad negativa, ya que una característica de la técnica de aprendizaje por refuerzo consiste en la adaptabilidad de las soluciones que ofrece a una variabilidad de las características de las zonas donde se explotan sus modelos obtenidos.

Particularmente, trabajaremos con un entorno dinámico donde se plantea la presencia de obstáculos en la zona de trabajo y una evolución de la variable "importancia" de visita en todas las casillas transitables por el vehículo explorador.

Cuando se trabaja con un algoritmo de aprendizaje por refuerzo dicho entorno debe poseer varias características que serán descritas a continuación:

- Mapa: El mapa describirá las características físicas del medio que pretende simularse. En este caso se parte de una matriz de valores que forma una imagen puede observarse en la figura 5.1 cuya dimensión consiste en (57, 38) píxeles o celdas matriciales. Esta matriz de valores pretende discretizar los puntos del lago Ypacaraí en forma de celda mediante un valor numérico.

  La superficie navegable queda descrita con un valor positivo para diferenciarse del espacio no visitable por el agente donde dichas casillas contendrán un valor nulo. Es cierto que esta caracterización del entorno natural puede realizarse con una precisión más o menos fina empleando una matriz mayor o menor para cada caso. En primer lugar, puede razonarse que cuanto mejor sea la representación del lago más real serán los experimentos, pero existe una contraindicación a esta rápida acepción y es que la carga computacional de trabajar con un entorno muy preciso puede ser muy importante.

  Como consecuencia de dicha situación los entrenamientos resultarían muy pesados ya que el escenario debe actualizarse a cada paso del agente, comprobar si la acción ha sido legal, almacenar información de las casillas de éste, etc., y se prefiere que se pueda trabajar más eficientemente con los algoritmos y buscar un modelo más adecuado a través de la sintonización de los parámetros de los mismos.

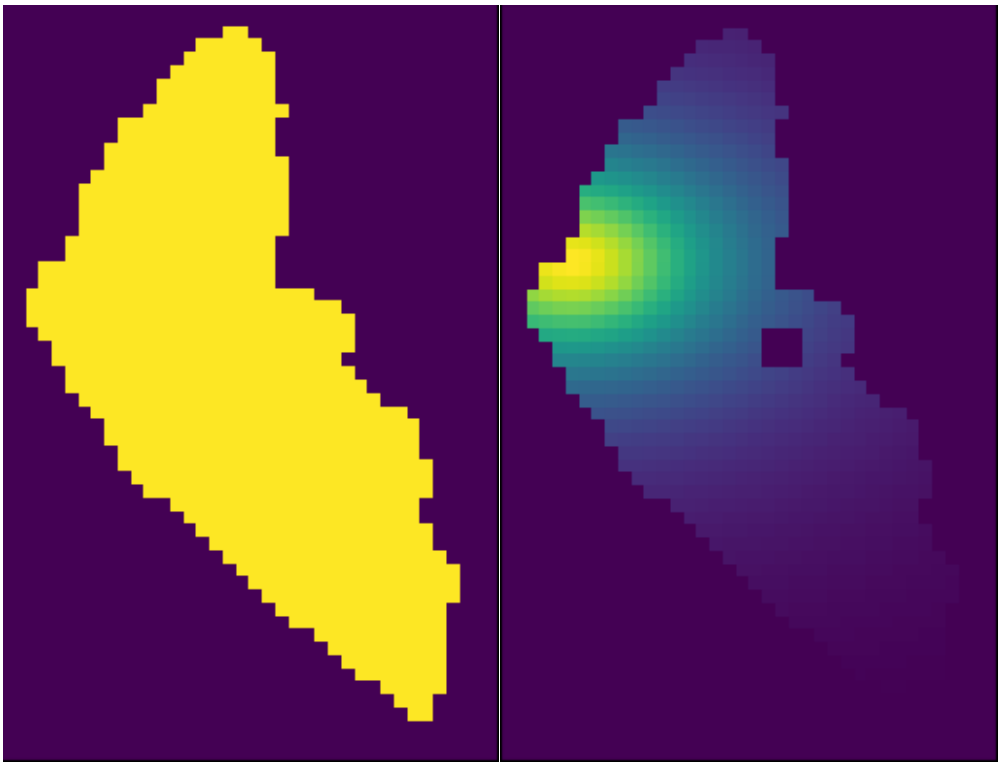


**Figura 5.1:** Vistas del entorno de simulación empleado para el aprendizaje y experimentación.

La penalización que obtendrá el vehículo al visitar zonas no navegables, como la orilla de la superficie será de -10.

El área de influencia del vehículo será de dos casillas, por lo que contabilizará la recompensa de la casilla donde se encuentre más las casillas que se encuentren en contacto a esta.

Para obtener el mapa con importancia no homogénea se emplea un método basado en la función de Shekel en el que se ha definido que haya 4 niveles de intensidad en la importancia de las casillas. En la figura 5.1 puede apreciarse dicha sensibilidad.

Se ha definido el parámetro de recuperación de la importancia en casilla visitada como $\frac{2}{nº\ máx\ desplazamientos}$ que en nuestra configuración del entorno corresponderá a $\frac{1}{58}$.

Tras la visita de una casilla por parte del vehículo se ha definido que el valor de importancia de la misma sea mínimo en ese instante. En siguientes actualizaciones el valor de dicha casilla sufriría la regeneración de su valor según el parámetro anteriormente comentado.

- Direcciones de desplazamiento: En este entorno se han definido 8 posibles direcciones de desplazamiento para el agente: norte, sur, este, oeste, noreste, sureste, suroeste y noroeste. Con ello se pretende representar todos los posibles desplazamientos del vehículo de manera eficiente y adecuada. Es cierto que, al trabajar sobre una matriz de datos, como se ha reflejado anteriormente, el consumo energético del vehículo no puede ser el mismo al desplazarse en trayectorias cortas como norte, sur, este y oeste que cuando se produzca un movimiento con dirección diagonal, ya que físicamente el desplazamiento es mayor. Para ello se define que, si un desplazamiento vertical y horizontal lleva asociado a un consumo unitario, los diagonales vienen definidos por la siguiente fórmula.

$$Consumo\ diagonal = \sqrt{2}\ Consumo\ horizontal\ o\ vertical = 1{,}414\ x$$

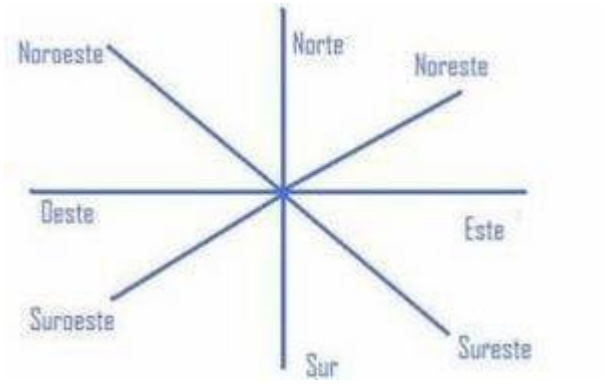



**Figura 5.2:** Direcciones de desplazamiento en el escenario.

La zona de influencia del vehículo será variable en el entorno, aunque inicialmente será igual a dos. Dicha zona de influencia se representará como si se tratase del radio de un circunferencia, aunque en el mapa sea imposible trabajar así ya que es discretizado. En la figura 4.13 puede apreciarse un detalle de este tema.

En esta configuración simulamos que la autonomía del vehículo es de 30 km navegables. Con ello el número de casillas máximo que se podrá visitar con el mismo será de 116. En función de las direcciones tomadas dicha cantidad puede ser menor.

- Estado del escenario: El entorno en el que se trabaja mantendrá una comunicación con el algoritmo y agente en el que hará *feedback* de las variables definidas por un proceso de decisión de Markov. La idea es que el agente envíe una acción a realizar y se reciba información del estado del entorno, recompensa obtenida, trayectoria y otro tipo de información importante. Es interesante comprobar cómo puede obtenerse una imagen global del escenario de trabajo y como se ubica en cada instante el vehículo móvil en dicha perspectiva. Puede darnos una información bastante completa del estado y la evolución del sistema.

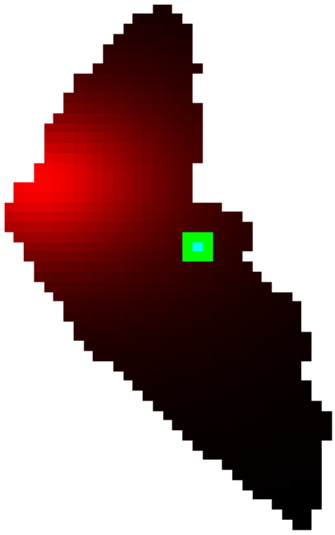


**Figura 5.3:** Detalle del vehículo y su zona de influencia.

Para almacenar el estado del entorno se empleará un vector de 4 componentes:

  - Estado [1]: matriz donde se almacena la posición del agente en el entorno.
  - Estado [2]: matriz donde se almacena la zona navegable del entorno.

- Estado [3]: matriz donde se almacena el valor de importancia de las casillas navegables del entorno.
- Estado [4]: matriz donde se almacena la zona y posición de influencia del agente en el entorno.

- Estructura de red: Se ha empleado un tipo de red neuronal convolucional a la hora de trabajar en los entrenamientos y episodios necesarios en nuestro entorno. Dichas redes deberán estar acondicionadas al algoritmo y entorno donde sea necesario usar el modelo obtenido. En nuestro trabajo se decide usar redes convolucionales y de tipo *Actor Critic* para el caso de uso del PPO, ambas *feedforward*. Son redes diseñadas para trabajar en entornos 2D como el mapa del entorno, y se ha tomado como condición común para todas ellas que las dimensiones de las capas de entrada y de salida sean iguales.
    - *Input Layer*: a la entrada será necesario tomar una imagen del estado.
    - *Hidenn Layer*:
        - *REINFORCE*: se empleará la función de activación lineal (Relu) y 2 capas convolucionales con distinto número de entradas 16 y 32, salidas 16 y 32 y filtros *kernel* 5 y 3.
        - PPO: se emplea la función tangente hiperbólica (Tanh) como función de activación. Tendremos dos capas con un tamaño de 64.
    - *Output Layer*: a la salida se obtendrá un vector de dimensión igual al número de acciones posibles (8) con distribución categórica.

  Se empleará el optimizador Adam en las redes neuronales de ambos algoritmos (apartado 4.8).
- Ley de recompensa: Se ha dispuesto que el escenario sea el encargado de evaluar y transmitir la recompensa obtenida por el agente tras recibir la acción a realizar por parte de este. Dicha evaluación se basa en seguir unos criterios definidos en la ley de recompensa [5]. En dicha ley se bonifica y se amonestan ciertos tipos de comportamiento que pueden ser considerados como positivos o negativos al supuesto comportamiento efectivo del ASV en la superficie de trabajo. Por ejemplo, se evalúa negativamente que la embarcación intente visitar casillas o zonas externas al lago ya que podría asemejarse a un encallamiento en la realidad. En cambio, se considerará positivamente el visitar casillas o zonas que no lo hayan sido previamente o que hayan regenerado su importancia, ya que la recompensa obtenida por ello será máxima en dicha ubicación. En apartado 4.9 se explica en más detalle la ley de recompensa que se aplica en este proyecto.

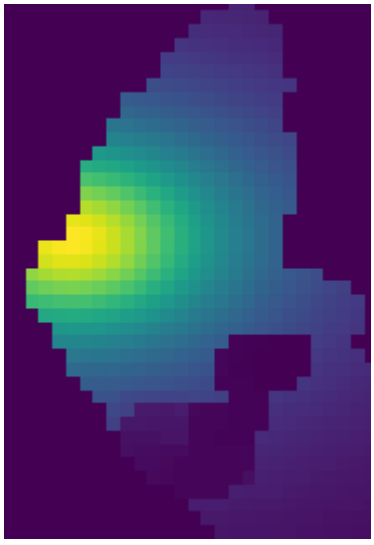


**Figura 5.4:** Evolución de la importancia en casillas visitadas.

Es importante destacar la portabilidad de este escenario a otros ámbitos al estar programado a conciencia para poder emplearse como los escenarios que pueden encontrarse en la herramienta investigadora de Gym (freeware). OpenAI es un sistema de entrenamiento abierto basado en código Python en la que muchos investigadores pueden realizar pruebas y experimentos de algoritmos y arquitecturas de aprendizajes con condiciones simuladas pudiendo obtener visualizaciones y datos de dichos experimentos aplicados en diversos escenarios preprogramados. La idea es este entorno sea accesible por más investigadores que quieran poner en práctica sus algoritmos y técnicas de aprendizaje sobre este entorno y así puedan alcanzarse diversas soluciones al problema de exploración definido en el mismo para poder ponerse en práctica la solución más idónea en campo.

La finalización del episodio queda sujeta al estado de la batería del vehículo o agente. Cuando en la batería quede menos de la energía necesaria para realizar un desplazamiento se considerará que el vehículo no puede cambiar de posición y se terminará el episodio en cuestión.

### 5.5.1. Bibliotecas empleadas

Para la realización de este trabajo se han empleado una serie de recursos técnicos que se desglosan a continuación:

- **Numpy:** es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas[20].
- **Pytorch:** es una biblioteca de aprendizaje automático3 de código abierto basada en la biblioteca de Torch, utilizado para aplicaciones que implementan cosas como visión artificial y procesamiento de lenguajes naturales[21]. Permite el uso y construcción de redes neuronales mediante funciones ya implementadas.
- **Tensorflow:** es una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos[22]. Esta biblioteca puede ejecutarse en varios núcleos de procesamiento (GPUs y CPUs) permitiendo su uso también en CUDA que es la unidad de procesamiento existente en las tarjetas gráficas, por lo que su uso en tareas gráficas puede ser más eficiente.
- **Matplotlib:** es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy[23].
- **StableBaselines3:** Stable Baselines3(SB3) es un conjunto de implementaciones robustas de algoritmos de aprendizaje por refuerzo con PyTorch[24] como framework de optimización. Mediante el uso de esta biblioteca se ha podido implementar el algoritmo PPO en el proceso de aprendizaje del ASV.

## 5.2. Estructuración de resultados

En este apartado se comentará como irán desglosados los resultados obtenidos a través de las experiencias obtenidas tras aplicar distintos métodos de exploración en el entorno.

[20] Fuente: https://es.wikipedia.org/wiki/NumPy
[21] Fuente: https://es.wikipedia.org/wiki/PyTorch
[22] Fuente: https://es.wikipedia.org/wiki/TensorFlow
[23] Fuente: https://es.wikipedia.org/wiki/Matplotlib
[24] Fuente: https://stable-baselines3.readthedocs.io/en/master/

La idea es mostrar con claridad y objetivamente los datos de una manera amena sin que la introducción de estos suponga confusión y que el lector no se pierda la lectura.

En primer lugar, se pretende comentar los hiperparámetros presentes en los algoritmos con los que se trabaja junto con una pequeña explicación individual. Se pretende clarificar la función de cada uno de ellos y cómo interactúan en el algoritmo que estén trabajando. Es necesario un proceso de sintonización de los mismos para alcanzar el comportamiento deseado de los modelos que se empleen en solucionar el problema propuesto. Para ello es necesario la realización de una serie de episodios y un análisis de los resultados extraídos. Este análisis será desarrollado en el subcapítulo correspondiente acompañado de la sensibilidad de los parámetros estudiados y como afectan las variaciones de estos en los modelos obtenidos.

Se ha estructurado los resultados para que aparezcan ordenados de manera que se presenten directamente los obtenidos con cada algoritmo. En este apartado se comentarán los parámetros que aparecen en cada uno de ellos, su efecto y los resultados que se recogen en distintas simulaciones realizadas. Posteriormente se plantea la evolución del agente-recompensa en la exploración del escenario cuando el agente realiza distintas trayectorias durante el ejercicio.

A la hora de mostrar las consecuencias de los episodios, se mostrarán mediante distintas figuras la evolución de los ejercicios representando aspectos importantes que sirvan para la evaluación de los experimentos.

Tras la presentación de los resultados se abarcará la comparativa también de los distintos métodos o algoritmos empleados en el entrenamiento de las redes neuronales y el movimiento autónomo correspondiente que realizará el agente cuando decidimos explotar los modelos recopilados.

### 5.2.1. Plan de validación de resultados

Para que los resultados obtenidos posean una catalogación de fiabilidad es necesario que para la toma de los mismos se sigan unos criterios de validación adecuados para asegurar la validez de los mismos.

Como criterio fundamental en la toma de datos experimentales, se busca que el entorno y versión del entorno de aprendizaje y experimentación sea el mismo para todos los algoritmos estudiados como en cada prueba de aprendizaje y pruebas ejecutadas. Así se garantiza que los datos obtenidos lo son partiendo de un punto departida universal.

Es importante también que los procesos de entrenamiento trabajen con el mismo script de creación de la red neuronal, porque, aunque evidentemente la red neuronal es parametrizable, la creación de estas por medio de otras librerías o funciones puede generar variaciones en el comportamiento del modelo creado respecto al método principal elegido.

Otro aspecto para tener en cuenta sería la representación de los datos obtenidos en las mismas magnitudes y que se tomen las mismas referencias, ya que cuando se realice alguna comparación de estos pueda garantizarse la validez de las conclusiones.

Para resumir, es importante cumplir

- Entorno de simulación fijo para todo tipo de algoritmos, aprendizajes y experimentaciones.
- Uso de los mismos scripts y funciones auxiliares.
- Establecer referencias absolutas a la hora de tomar resultados en las magnitudes y unidades.

A la hora de evaluar los resultados obtenidos, se valorarán varios aspectos que se consideran principalmente importantes:

- Recompensa acumulada
- Número de movimientos del vehículo (optimización de la batería)
- Estabilidad de la curva de aprendizaje
- Varianza en el gradiente (durante el entrenamiento)

- *Loss* en el gradiente (durante el entrenamiento)

## 5.3. Resultados obtenidos

A la hora de abordar la experimentación en el entorno comentado, se ha partido de datos presentes en diversas fuentes bibliográficas en las que se ha realizado una experimentación previa con los algoritmos que se emplean en este trabajo [37] [38].

### 5.3.1. Parámetros de los algoritmos

En los algoritmos *Deep Reinforcement Learning* existen parámetros que pueden considerarse comunes entre una gran mayoría de algoritmos. En nuestro caso podemos destacar aquellos que tienen consideración en los algoritmos que estudiaremos. Se comentarán aquellos que, suponemos previamente al estudio de su comportamiento, pueden afectar en mayor medida a la evolución de los entrenamientos y creación de los modelos del sistema como resultado.

Se propone desglosar dichos hiperparámetros en los apartados siguientes cuando se presenten los relacionados con cada algoritmo que estudiaremos.

#### 5.3.1.1. Parámetros de entrenamiento algoritmo Discounted Reward

A continuación, desglosaremos los valores de los hiperparámetros que aparecen en el algoritmo *Discounted Reward(REINFORCE)*. Es necesario un proceso de sintonización de estos para garantizar que el algoritmo se adapte lo mejor posible a las condiciones planteadas en el problema de exploración. Como consecuencia, veremos cómo los resultados obtenidos ofrecerán un mejor modelizado y por tanto unas recompensas mejores a lo largo de este proceso.

Como referencia para la interpretación de la virtud del valor de un parámetro se tomará como criterio la recompensa media obtenida y el número de desplazamientos realizado por el vehículo en los experimentos.

Seguidamente se describirá la función de cada uno de ellos y como tras varios experimentos, como han afectado al algoritmo y por ende al modelo obtenido asociado a distintos valores de estos.

- **Learning Rate:** este parámetro es el encargado de definir la tasa de actualización de los pesos de la red neuronal por iteración. Si trabajamos con una tasa de actualización muy baja tendremos problemas a la hora del aprendizaje ya que el aprendizaje puede estancarse en un mínimo local y no obtendríamos un modelo óptimo, además sería un entrenamiento lento. En cambio, si se selecciona un valor muy alto, este parámetro se define entre 0 y 1, podríamos no detectar el mínimo global.
- **Epochs:** también conocido como numero de ciclos, episodios o épocas, este parámetro define el número de veces que se realiza el entrenamiento de la red neuronal. La red neuronal trabajará sobre el conjunto de datos y actualizará sus parámetros internos tras la finalización de cada episodio. A menudo es común emplear, sobre todo en aprendizaje por refuerzo varios episodios para que la red neuronal estudie el conjunto de datos desde diversas perspectivas y por lo tanto se generen diversos modelos por lo que la red neuronal extraerá modelos distintos en diferentes episodios.
- **Batch size:** en este caso *Batch size* define el número de experiencias (acción-trayectorias-estado) con los que trabajará la red neuronal en cada actualización de los pesos. Puede verse referenciado con el término lote y es un parámetro muy ligado al de episodios o *epochs* ya que el tamaño del lote define el número de experiencias o *mini-batches* que se le pasa a la red por cada *epoch*.
- **Hidden sizes:** establece el número de neuronas de las capas ocultas de la red neuronal que estarán conectadas con todas las demás de la siguiente capa. Por defecto se trabaja con un valor igual a 512,

aunque más adelante se comenta cómo evolucionan los resultados obtenidos al realizar una modificación de este parámetro.

#### 5.3.1.2. Parámetros del algoritmo PPO

En este apartado se analizarán los valores de los hiperparámetros que conforman al algoritmo PPO. Mediante un proceso de sintonización de estos buscamos obtener una configuración óptima para que el algoritmo consiga alcanzar los mejores resultados posibles. Para no volver a definir ciertos hiperparámetros anteriormente descrito, cuando se produzca la aparición de alguno de ellos se describirá algún aspecto del mismo en su importancia en el algoritmo actualmente analizado.

Existen un número elevado de hiperparámetros en el PPO y que definen su funcionamiento. Algunos afectarán en mayor o menor medida al comportamiento en función de distintas variables. Por ejemplo, si estamos trabajando en un tipo de entorno estático o dinámico, las características de los datos con los que se trabaje, si preferimos que se alcancen resultados a corto o largo plazo o incluso del modelo o red neuronal que se emplee.

- **Learning_rate:** variable asociada a la tasa de actualización de los pesos de la red neuronal por iteración para el algoritmo PPO. Realiza la misma función que en el algoritmo *REINFORCE*.
- **Clip_range:** este parámetro normalmente representado mediante la letra Є, se emplea para realizar un recorte al objetivo de la política. Una nueva política puede obtener beneficios partiendo de otra antigua, pero se llegará a un punto que se evalúa mediante un valor de clip en el que por mucho que continue no podrá seguir mejorando los valores de la función de valor.
- **Clip_range_vf:** seleccionando esta variable activamos el proceso de clipeado de la función de pérdida. Si estuviese igual a cero se bloquea el recorte de la política y determinará en qué medida actuará sobre la función de valor el valor de *clip range* antes seleccionado.
- **Target_KL:** limita la divergencia producida entre pasos del entrenamiento. Cuando se supere el valor dispuesto se dejará de entrenar no se actualizará el valor del gradiente y el valor de la tasa de aprendizaje se actualizará a cero. Esta limitación se incluye debido a que el proceso de clipeado puede realizar actualizaciones muy grandes e incluir mucha divergencia en el modelo que estamos entrenando.
- **Max_grad-norm:** Este coeficiente determina el valor máximo del gradiente en el proceso de clipeado. A continuación, se muestra cómo se modeliza la función objetivo de este algoritmo.

$$L_t^{CLIP+VF+S}(\theta) = E_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

- **Ent_coef:** Con este valor lo que se consigue es que multiplicando el máximo valor de entropía posible y añadiéndolo al *loss* o pérdida conseguimos que una acción predominante defina la política y la exploración sea descartada automáticamente. En la fórmula anterior se representa como c2.
- **Vf_coef:** Este coeficiente regula la acción del error cuadrático de la diferencia entre el resultado de la función de valor y el valor idónea del mismo que buscamos.

$$L_t^{VF}(\theta) = (V_\theta(s_t) - V_t^{targ})^2$$

- **N_epochs:** número de episodios a realizar por el algoritmo antes de la actualización de los pesos de la red neuronal.
- **N_steps:** variable que define el número de actualizaciones de la política a realizarse. Con un valor elevado de esta variable se realizará un gran número de iteraciones en la búsqueda de una política óptima. Si seleccionamos un *n_steps* considerable a veces es contraproducente ya que tras muchas iteraciones la mejora sustancial de una nueva política puede ser inapreciable para una carga computacional mayor.
- **Gamma:** Gamma se encarga de determinar el factor de escala de la función de valor. Como hemos

visto estamos trabajando con un estimador de ventaja generalizado. Este estimador se cataloga dentro de los métodos de diferencia temporal.

Aquí se encuentra el factor de descuento, γ llamado gamma que como se puede apreciar en la ecuación de la ventaja determina la importancia de los valores futuros para el cálculo de la ventaja actual.

- **Gae_lambda:** para compensar la diferencia entre la varianza que se produce y el sesgo temporal se emplea el parámetro lambda. Este parámetro oscila entre cero y uno, pero realmente los mejores resultados aparecen cuando se trabaja con valores de entre 0,9 y 1.

  A diferencia de gamma, lambda solo introduce sesgo cuando la función de valor no devuelve valores adecuados o inexactos.

- **Batch_size:** *Batch* o lote se define como el número de muestras de información con el que trabaja el modelo antes de actualizar sus parámetros internos y calcular el error entre la recompensa obtenida y la esperada.

  Así mismo si tenemos un gran número de muestras en cada *batch*, el entrenamiento será más lento y se habrán analizado más experiencias del agente, pero esto no significa que sea mejor ya que es posible que sea más favorable que el modelo actualice sus pesos más a menudo [37].

Como referencia para caracterizar como de bueno será un parámetro, se toma como criterio la recompensa media obtenida y el número de movimientos realizados por el agente en los experimentos.

A continuación, se describirá la función de cada uno de ellos y como tras varios experimentos, como han afectado al algoritmo y por ende al modelo obtenido asociado a distintos valores de estos.

- **Learning_rate**

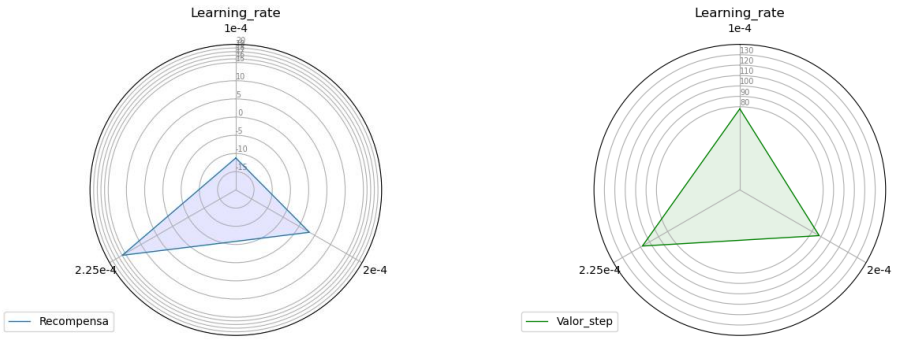



**Figura 5.5:** Resultados del entrenamiento respecto al parámetro learning_rate (PPO).

Este método trabaja con un optimizador Adam donde un parámetro muy importante y presente será el learning rate. También conocida como tasa de aprendizaje es el parámetro de nos va a marcar el tamaño de los pasos que realizará el gradiente a la hora de minimizar la función de coste obteniendo un mínimo local o global. Para valores superiores a 2.25e-4 el algoritmo no converge bien.

- **Clip_range**

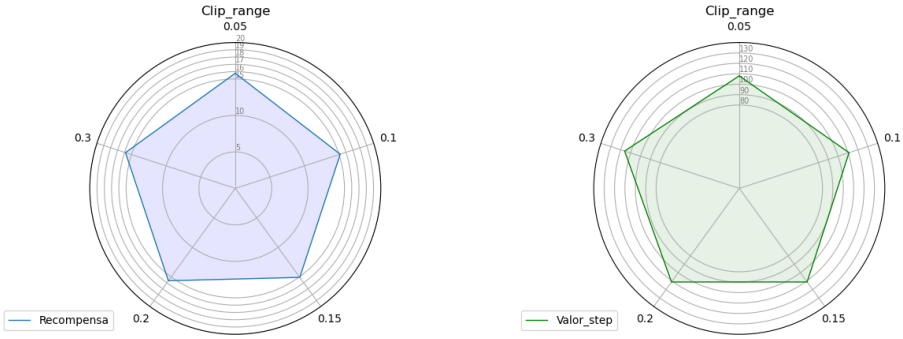



**Figura 5.6:** Resultados del entrenamiento respecto al parámetro clip_range (PPO).

Vemos como las experiencias recopiladas en varios experimentos muestran un comportamiento generalmente similar en los resultados obtenidos, donde el valor más adecuado resulta el 0.3 ya que se obtiene un valor más elevado de recompensa y un número de desplazamientos más alto.

- **Clip_range_vf**

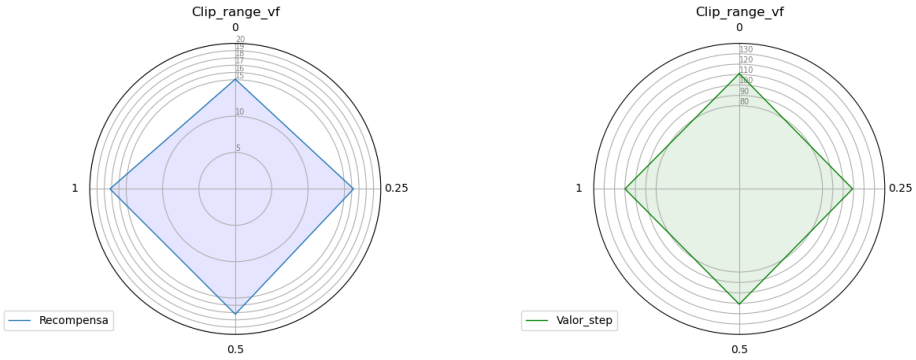



**Figura 5.7:** Resultados del entrenamiento respecto al parámetro clip_range_vf (PPO).

Se aprecia como aplicar el proceso de clipeado a nuestro algoritmo sin saturar nos genera la mejor configuración de este parámetro.

- **Target KL**

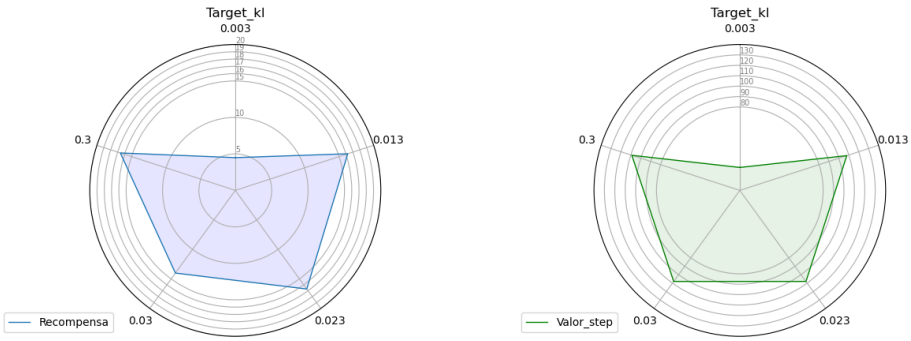



**Figura 5.8:** Resultados del entrenamiento respecto al parámetro target_kl (PPO).

En los resultados correspondientes a la sintonización del parámetro Target KL se observa como un actuación débil sobre la divergencia que se produce entre los *steps* del algoritmo provocan que el modelo resulta inadecuado e inestable. Para valores entre [0.01, 0.3] se obtienen modelos con un comportamiento acorde a lo requerido como solución con carácter convergente.

- **max_grad_norm**

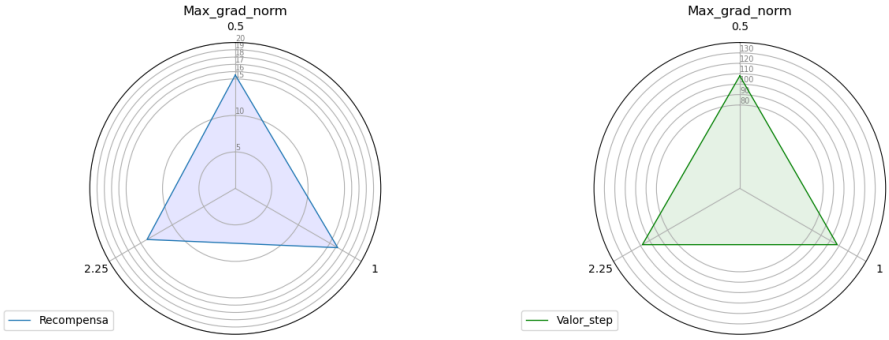



**Figura 5.9:** Resultados del entrenamiento respecto al parámetro max_grad_norm (PPO).

Un valor superior a la unidad genera modelos del sistema poco eficientes como puede observarse al obtener recompensas inferiores para un número mayor de desplazamientos del vehículo. Tiene sentido ya que un paso elevado de este parámetro puede provocar que el resultado del gradiente se salte el máximo absoluto de la función que estamos maximizando.

- **ent_coef**

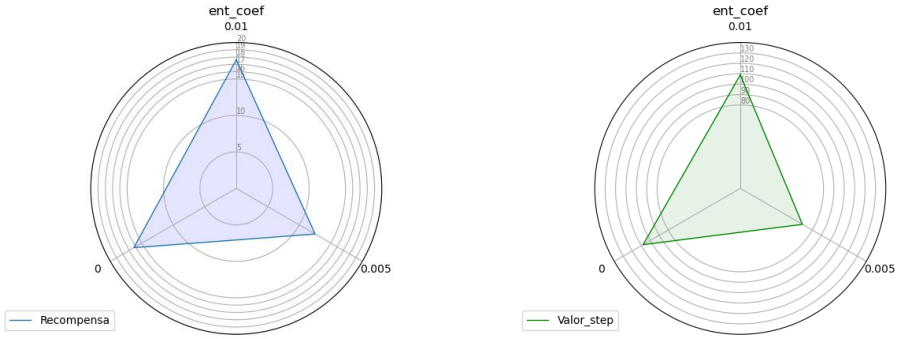



**Figura 5.10:** Resultados del entrenamiento respecto al parámetro ent_coef (PPO).

En este parámetro se aprecia la importancia del carácter estocástico de la técnica aplicada ya que el entrenamiento de un modelo con una influencia determinista presenta unos resultados mediocres. Nos basamos para realizar esta afirmación en los resultados obtenidos para valores en torno a una década superior ([0.1-1]) que no se han representado en las figuras anteriores debido a que las recompensas obtenidas eran muy pequeñas y negativas no pudiendo ser representadas adecuadamente.

- **vf_coef**

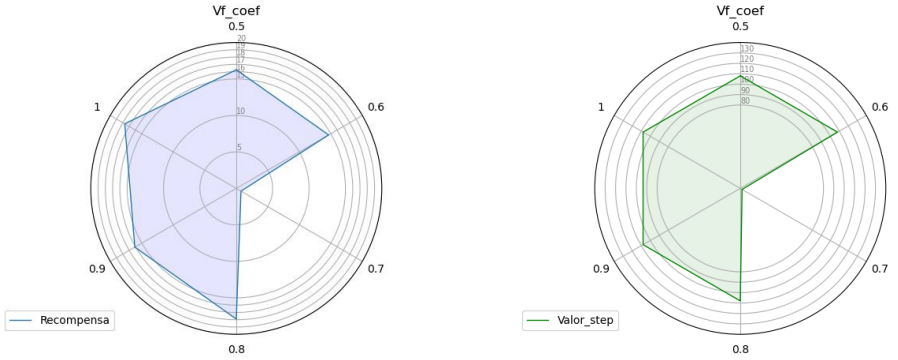



**Figura 5.11:** Resultados del entrenamiento respecto al parámetro vf_coef (PPO).

El parámetro vf_coef aporta información sobre la poca influencia que aplica su sintonización en la recompensa de los distintos modelos realizados con sus distintos valores.

- **n_epochs**

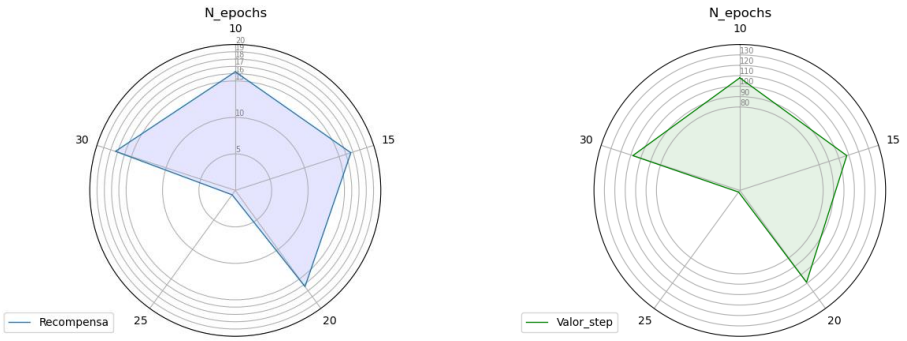



**Figura 5.12:** Resultados del entrenamiento respecto al parámetro n_epochs (PPO).

Vemos como se ha obtenido un modelo deficiente para el valor de $\mathrm{n\_epochs} = 25$ y como el aumento de los episodios supone una tendencia de la recompensa alcista y como se van obteniendo mejores modelos al trabajar con un valor alto de este parámetro.

- **n_steps**

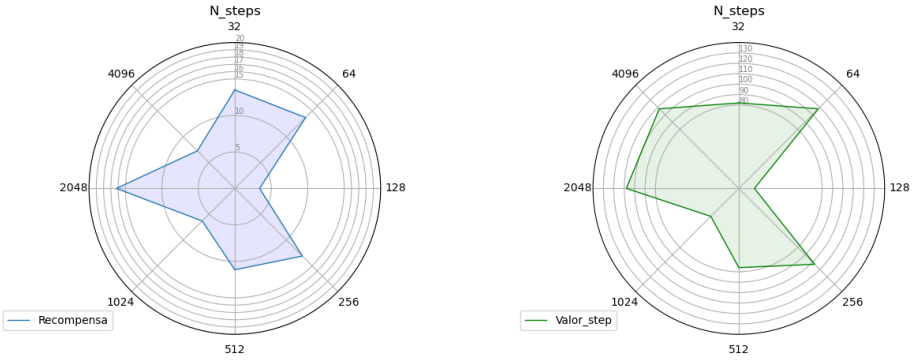



**Figura 5.13:** Resultados del entrenamiento respecto al parámetro n_steps (PPO).

Los resultados obtenidos derivados a la sintonización de este parámetro nos hacen ver la sensibilidad del algoritmo frente a la variación de este. Los mejores resultados obtenidos corresponden con el valor igual a 2048. Se aprecia como trabajando con un número pequeño de iteraciones en la optimización de la política se obtienen entrenamientos deficientes. Es necesario trabajar con un valor considerable debido a la complejidad asociada a nuestro entorno de trabajo.

- **gamma**

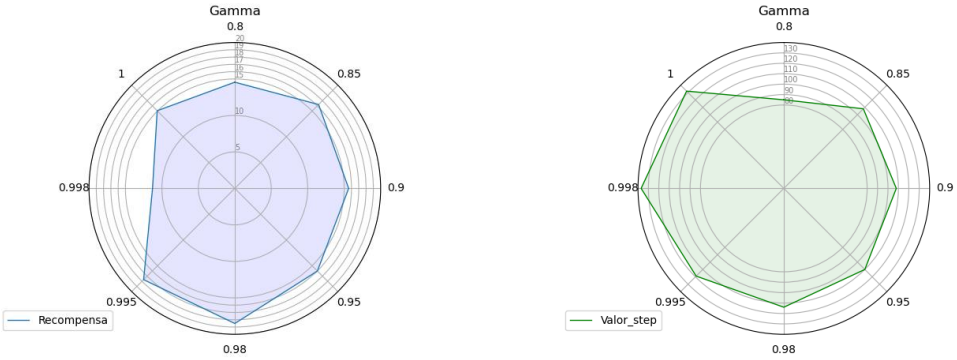



**Figura 5.14:** Resultados del entrenamiento respecto al parámetro gamma (PPO).

En esta experimentación el mejor modelo obtenido corresponde a $gamma = 0.98$ ya que se dónde se obtiene una mayor recompensa y un numero menor de desplazamientos del agente. Si se observa la evolución de la influencia de $\gamma$ en los distintos modelos vemos como es importante trabajar con una previsión de recompensas futura.

- **gae_lambda**

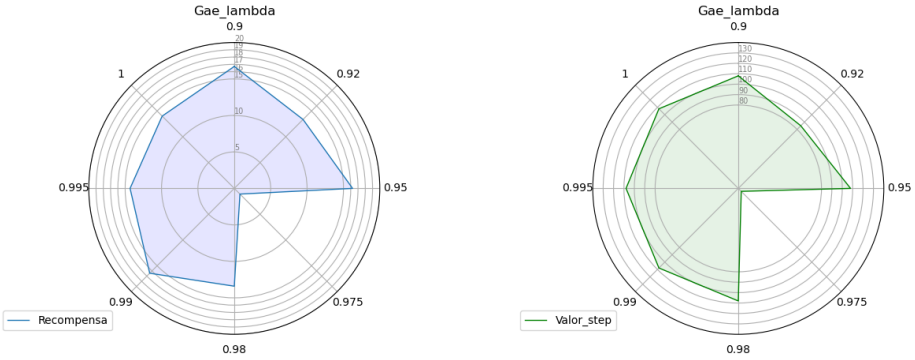



**Figura 5.15:** Resultados del entrenamiento respecto al parámetro gae_lambda (PPO).

Se ha entrenado un modelo deficiente para el valor $gae\_lambda = 0.975$ . El valor más adecuado de este parámetro se alcanza para $gae\_lambda = 0.99$

- **batch_size**

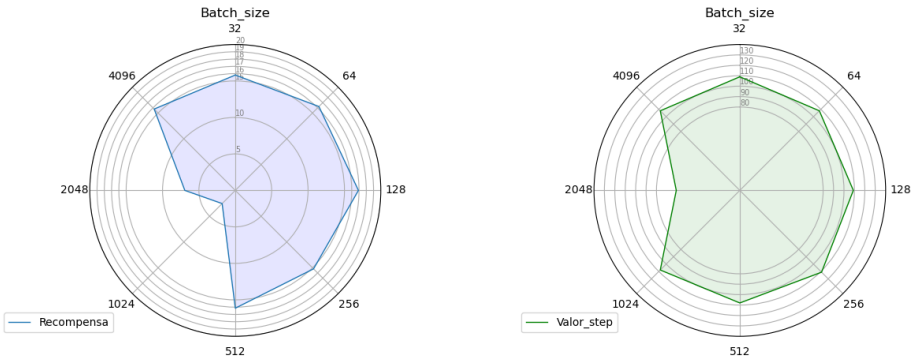



**Figura 5.16:** Resultados del entrenamiento respecto al parámetro batch_size(PPO).

El número de lotes que genera un modelo más adecuado teniendo en cuenta la métrica de nuestro trabajo es el correspondiente a $Batch\_size = 128$. Para valores superiores del mismo supone una tendencia exploratoria inadecuada del agente en detrimento de una explotación justa. Esto ocurre porque se evalúan demasiados escenarios o estados distintos y por tanto los datos provocan que la configuración de pesos de la red neuronal sea inadecuada.

Puede ser interesante realizar una visita al estudio presente en esta línea de la bibliografía [40], para tener una visión del comportamiento de este algoritmo y una comparativa con otros como el A2C, SAC y TD3.

| Hiperparámetros | Valores asociados a mejores resultados individuales |
|---|---|
| **Learning_rate** | 2.25e-4 |
| **Clip_range** | 0.3 |
| **Clip_range_vf** | 0.5 |
| **Target_kl** | 0.023 |
| **Max_grad_norm** | 1 |
| **Ent_coef** | 0.01 |
| **Vf_coef** | 1 |
| **N_epochs** | 30 |
| **N_steps** | 2048 |
| **Gamma** | 0.99 |
| **Gae_lambda** | 0.99 |
| **Batch_size** | 128 |

**Tabla 5.1:** Valores individuales de hiperparámetros con mejores resultados(PPO).

#### 5.3.1.3. Sensibilidad de los hiperparámetros

A la hora de buscar una sintonización correcta de los hiperparámetros del PPO se ha seguido paso a paso una estrategia basada en la recogida de experiencias obtenidas tras el entrenamiento de las redes neuronales. Como se ha podido comprobar, este algoritmo posee una gran cantidad de hiperparámetros que afectan a la realización de su labor por lo que la sintonización de estos para obtener un resultado óptimo no es una tarea trivial.

Se ha combinado por optimizar los parámetros individualmente empezando por los que sus variaciones podrían afectar en mayor medida al comportamiento del algoritmo y como consecuencia en la creación del modelo, para partir de un ajuste más global a uno más fino.

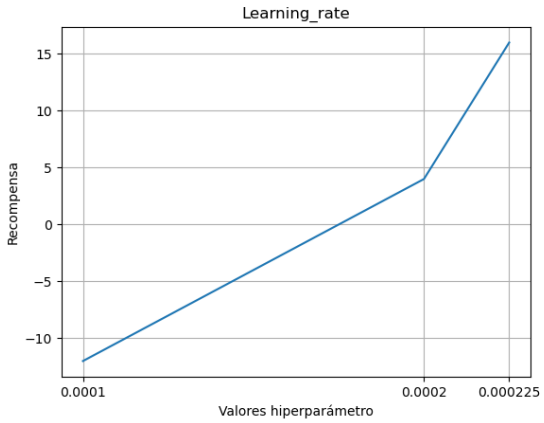
Como es natural tenemos que partir de la tasa de aprendizaje, continuando con los parámetros que afectan directamente al proceso de clipeado, gradiente, configuración de la red neuronal, para finalizar con las constantes que definen la longitud de los episodios y entrenamientos.

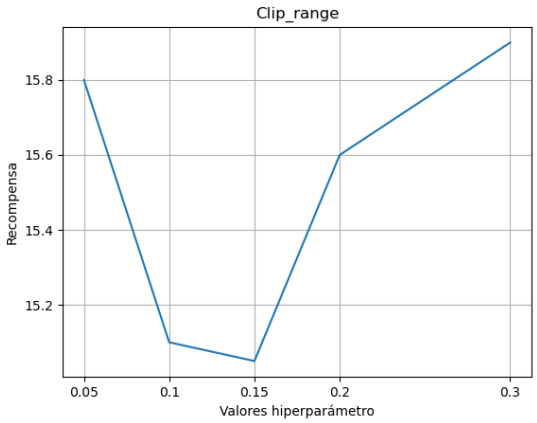
Realizando entrenamientos y episodios de prueba se recogen unos resultados de los que posteriormente se extraen unas figuras en las que se aprecia la sensibilidad de los mismos en función de los valores tomados en cada hiperparámetro durante estos entrenamientos. Si analizamos estas figuras puede apreciarse la tendencia de los resultados obtenidos en función de cada hiperparámetro del algoritmo. Al seguir este rumbo podemos valorar si estamos tomando el sentido correcto a la hora de modificar los valores y buscar el máximo de estos individualmente. Es importante también evaluar la huella o importancia que posee cada parámetro en el algoritmo observando cómo afecta una variación del mismo en la evolución del sistema. Es un proceso importante y clave para el diseño de un algoritmo eficiente que definirá el comportamiento del mismo por lo que el conocimiento de la sensibilidad de estos hiperparámetros será una herramienta muy útil para la

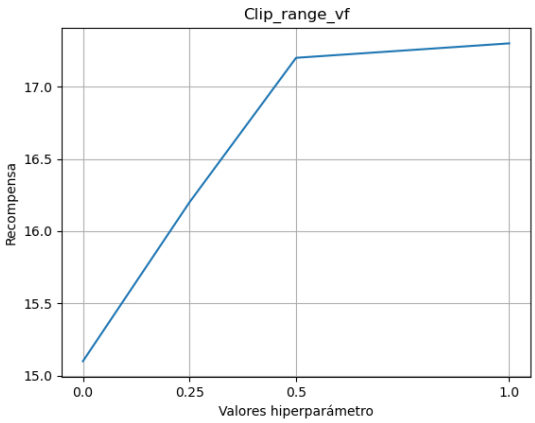
sintonización.

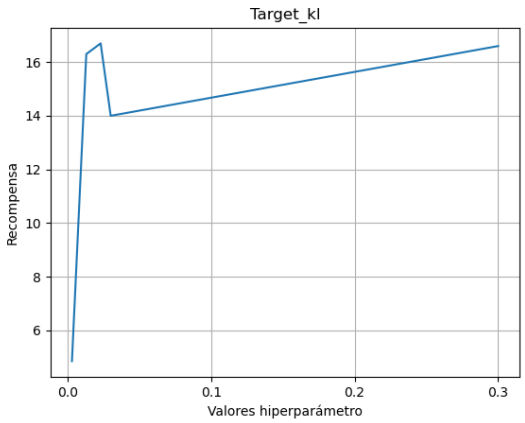
Como punto de partida se consideran los valores que se puedan encontrar en la literatura, pero es importante tener en cuenta que esta información solo puede ayudarnos en eso, en empezar el proceso de optimización.

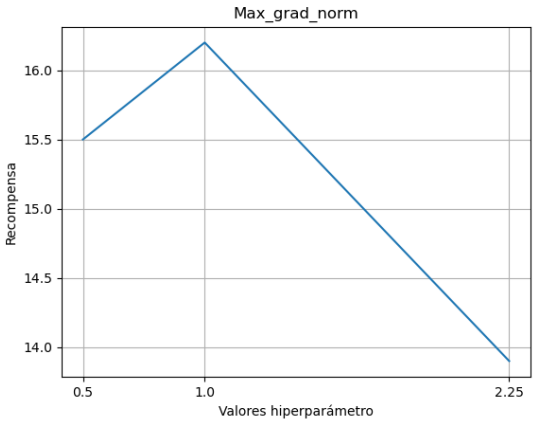
A continuación, se muestran las sensibilidades obtenidas normalizadas de los experimentos realizados.

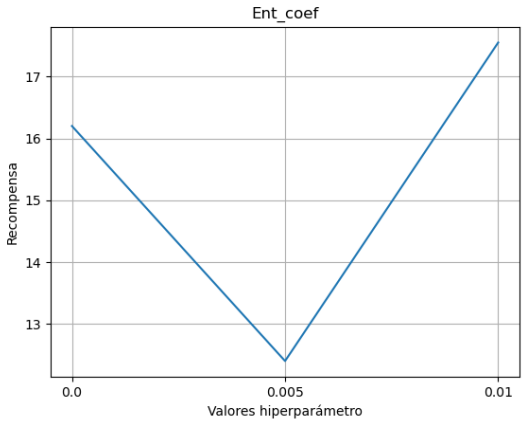


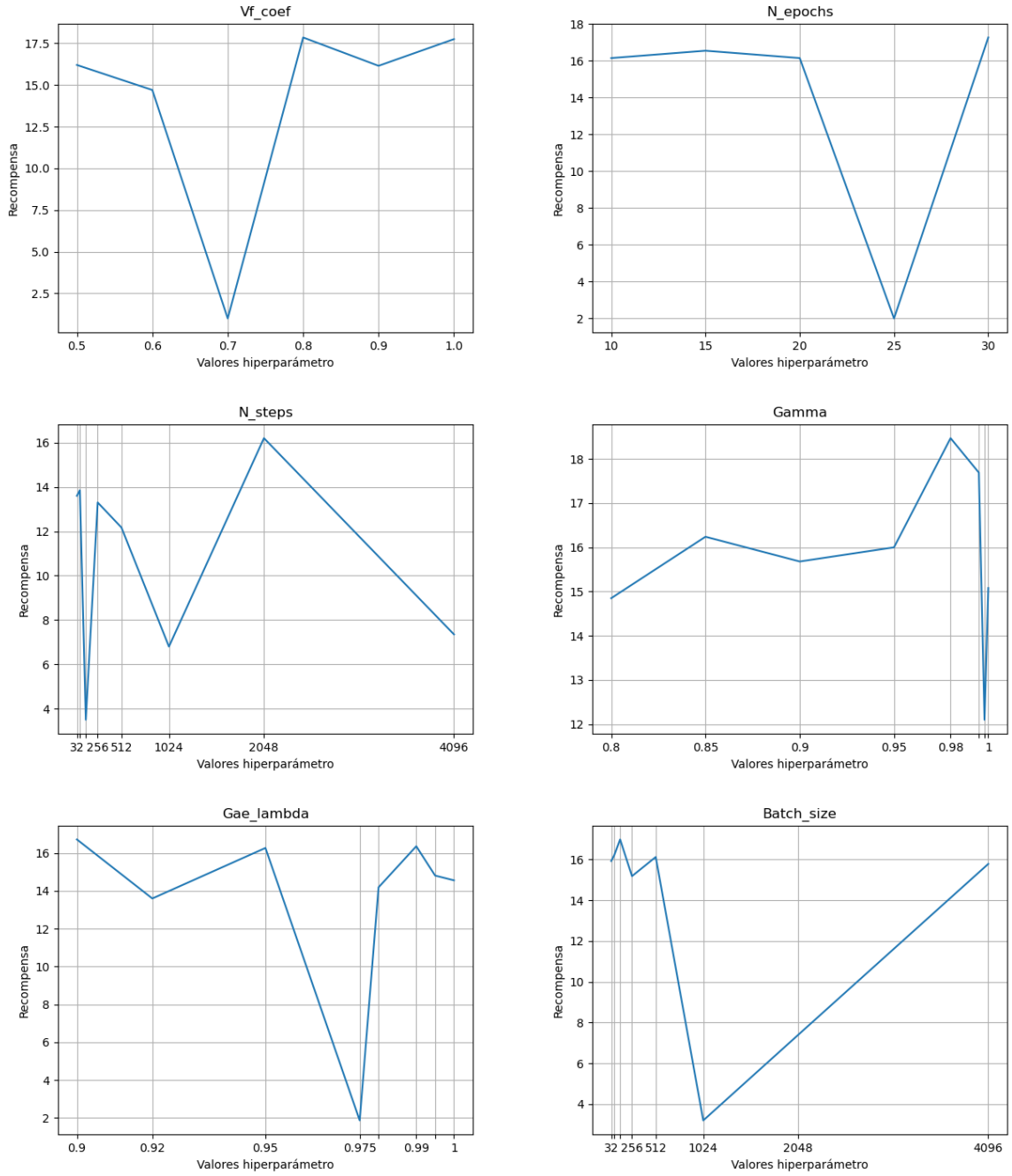



**Figura 5.17:** Sensibilidad de los hiperparámetros estudiados del algoritmo PPO.

En las curvas anteriores puede extraerse una visión relativa a la elección prevista de los hiperparámetros, pero para un correcto modelizado también se ha comprobado que la mejor configuración de parámetros no es el valor óptimo obtenido individualmente, si no que algunos de ellos tienen relación y por lo tanto hay que realizar un estudio más profundo evaluando la interacción relativa de ellos.

Para solucionar la consideración anterior, se parte de una configuración de parámetros óptima, individualmente hablando, y se van realizando más pruebas modificando los hiperparámetros en un rango individual de buen funcionamiento para posteriormente evaluar los resultados. Es un proceso bastante tedioso, ya que necesita de un estudio intrínseco de las características del algoritmo, entorno, red neuronal empleada, optimizador, etc.

Tras realizar este proceso iterativo de mejora se obtienen unos resultados que mejoran los recogidos en el proceso de optimización de hiperparámetros individual.

En el siguiente apartado se presentará la mejor configuración obtenida y los resultados correspondientes.

### 5.3.2. Resultados para superficie de importancia homogénea

**Figura 5.18:** Ejemplo de escenario de importancia homogénea.

Tras realizar los modelos de aprendizaje, será interesante comprobar los resultados que nos aportan estos para evaluar el comportamiento de los mismos y las virtudes y defectos que presentan. Para ello se ha diseñado un entorno controlado en el que trabajaran en las mismas condiciones que otros agentes obligados a seguir unas trayectorias más simples.

Compararemos los resultados de los agentes que son gobernados por las políticas y modelos obtenidos en los entrenamientos con otros que siguen trayectorias aleatorias y de tipo *lawn-mower* "cortacésped". Si los entrenamientos han sido realizados adecuadamente deberíamos observar cómo se mejora la recompensa por el agente que trabaja con trayectoria basados en algoritmos predictivos respectos a los más simples.

En este apartado, el entorno con el que trabajemos será el mismo para todos los experimentos y de importancia homogénea, por lo que todos los puntos visitados aportarán la misma recompensa, con la excepción de que un punto que ya ha sido previamente visitado obviamente pierde parte de su valor ya que sus datos ya fueron recogidos.

Es importante destacar que todos los puntos de este entorno comienzan con un valor máximo de importancia y el mismo decrecerá si es visitado, para volver a incrementarse a lo largo del tiempo. Es importante tener en cuenta lo mismo ya que en los siguientes resultados puede inducir a error observar cómo puntos visitados por el agente aparecen con una importancia de visita máxima. Esto ocurre por las imágenes que se muestran serán las tomadas al final del experimento y por lo tanto los puntos visitados al principio de este ya tuvieron tiempo para ganar valor de importancia mientras el agente los abandonó.

A continuación, tras una serie de experimentos realizados de diversa índole, se muestran los resultados medios obtenidos donde podemos ver la evolución de la recompensa obtenida a través de los movimientos del vehículo no tripulado y el desplazamiento de este por el entorno de experimentación.

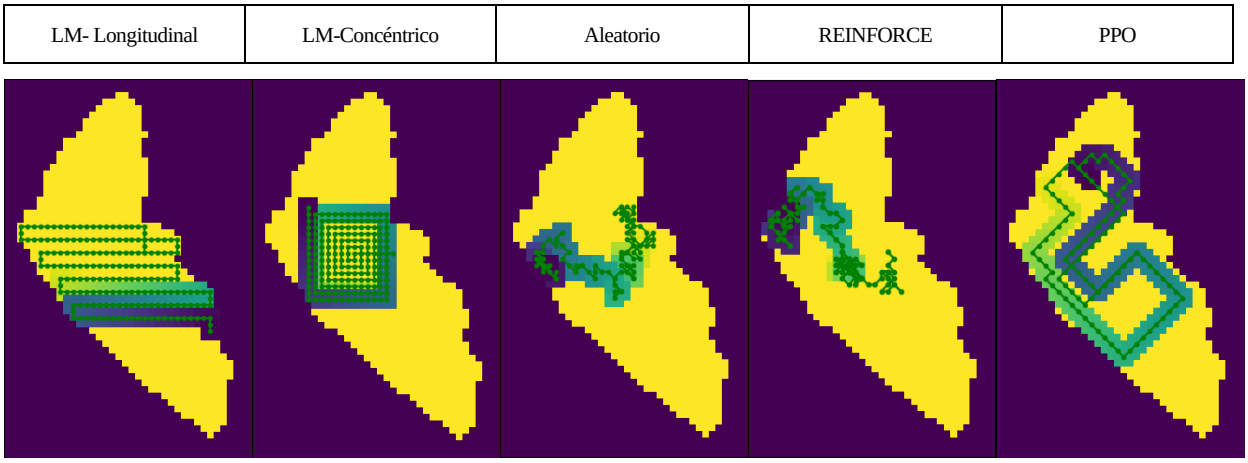



**Figura 5.19:** Exploración autónoma empleando distintos algoritmos en escenario homogéneo.

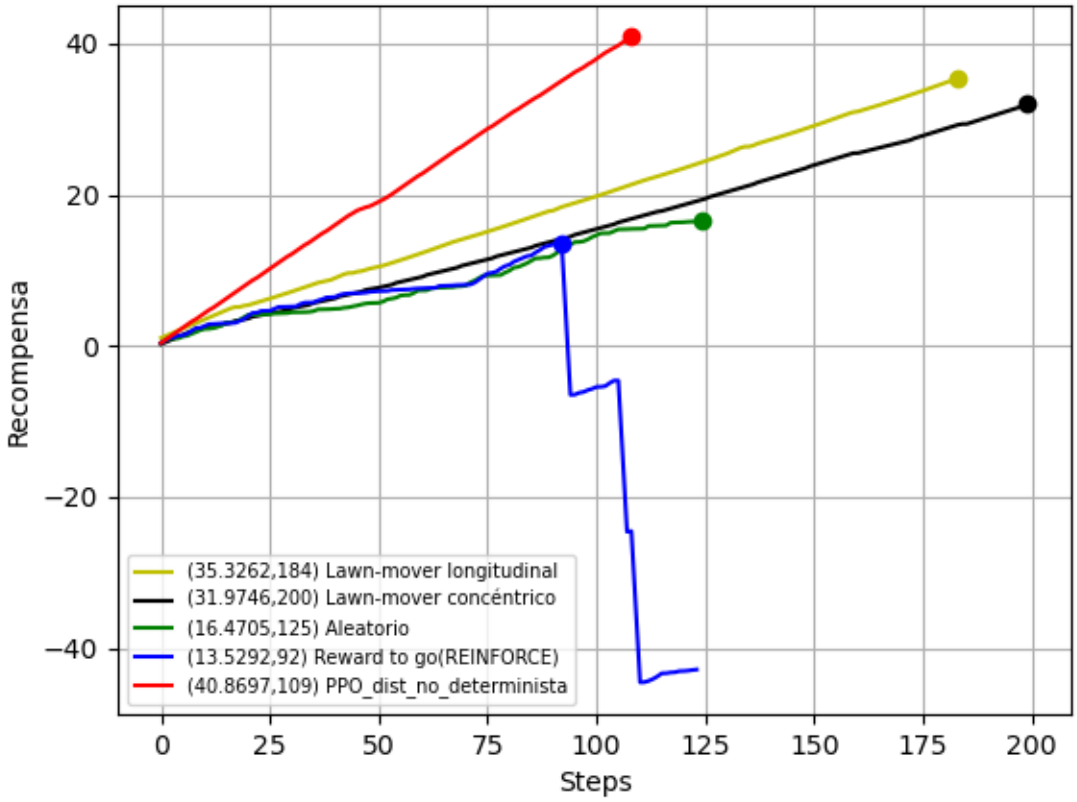



**Figura 5.20:** Evolución de la recompensa en la exploración autónoma empleando distintos algoritmos en escenario homogéneo.

### 5.3.3. Resultados para superficie de importancia no homogénea

Hemos visto una serie de resultados obtenidos en un escenario en los que la toma de datos en cualquier punto aporta la misma recompensa en un escenario de importancia homogénea. Ha sido muy interesante ver distintos comportamientos y se pueden sacar conclusiones muy útiles, pero a la hora de buscar la validación de estos en el entorno real surgen dudas de si el entorno en el que trabajamos podría reflejar mejor las características del medio real.

A la hora de intentar replicar con más exactitud una situación o estado con la realidad del entorno se nos plantea la prerrogativa de la importancia de tomar medidas en el lago. Salta a la vista que, en la realidad la importancia de tomar medidas en distintos lugares de este no puede ser la misma. Pueden existir puntos de vertidos de distinto origen, lugares donde la dinámica acuática almacena mayor concentración de sustancias, etc. Por ello es importante que se pueda trabajar con un escenario donde la importancia de las casillas no sea uniforme, porque, aunque siga un poco alejado de la realidad se introduzca este aspecto.

Aquí veremos los resultados obtenidos en un escenario con importancia no homogénea que se alcanzan cuando se aplican distintas trayectorias.

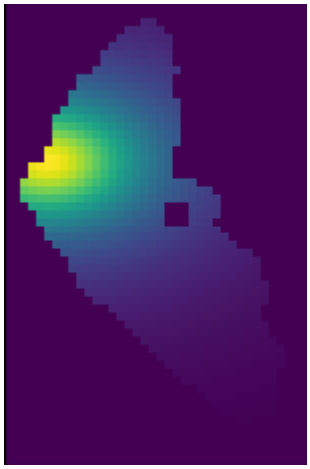


**Figura 5.21:** Ejemplo de escenario de importancia no homogénea

Al igual que en el apartado anterior se plantean diversas trayectorias de exploración para comparar la virtud de los resultados obtenidos. Se han realizado experimentos con trayectorias aleatorias, de exploración tipo *lawn-mower* "cortacésped" y autónomas de nuestros modelos de aprendizaje.

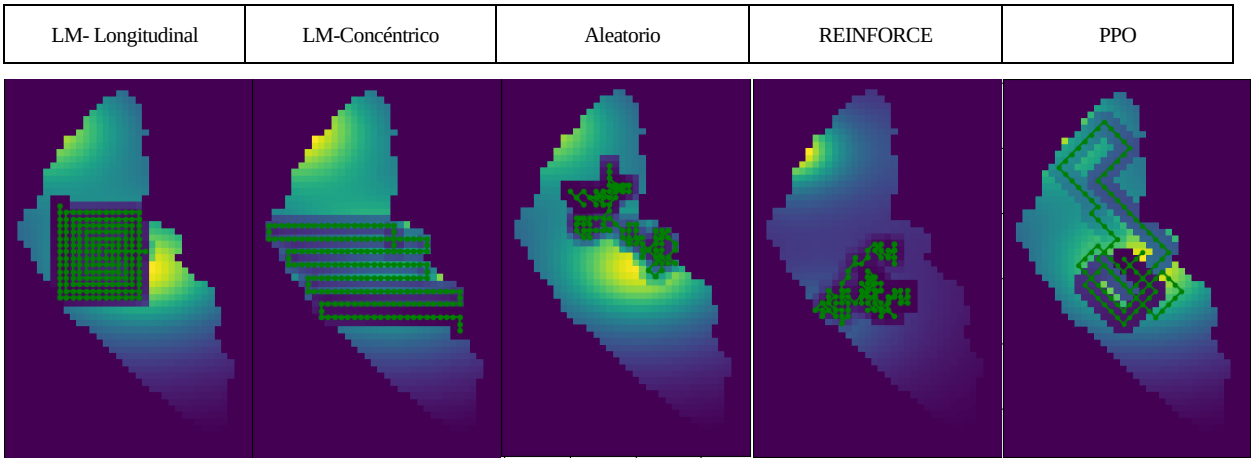



**Figura 5.22:** Exploración autónoma empleando distintos algoritmos en escenario no-homogéneo.

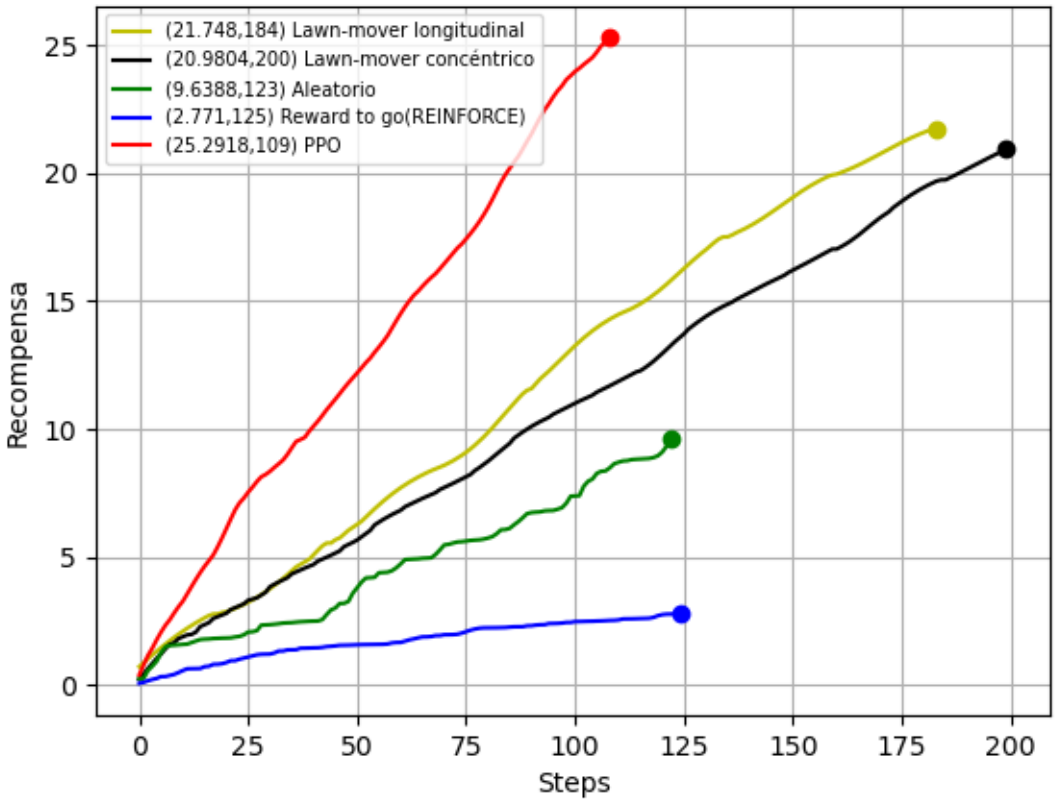



**Figura 5.23:** Evolución de la recompensa en la exploración autónoma empleando distintos algoritmos en escenario no-homogéneo.

**Trayectoria lawn-mower concéntrica**

Para analizar los resultados obtenidos y tener una referencia estable en torno a e estos, usamos la familia de algoritmos tipo *lawn-mower* o de barrido como también son conocidos. Los movimientos resultantes poseen características comunes que los hacen fáciles de identificar. Se producen modelos de movimiento regulares en los que el vehículo sigue unos patrones de trayectorias casi repetitivos para visitar todos los puntos posibles al menos una vez.

Es cierto que estos métodos son poco eficientes si los comparamos con otros en los que se analiza el terreno o superficie al no evaluar el medio por el que se desplaza, pero también es útil por la misma razón en entornos en los que no es necesario y se busca un trazado de rutas sencillo y algo simple.

En un primer lugar se analiza un movimiento de barrido concéntrico hacia el exterior en el que vemos como se visitan todos los puntos y existe un poco de redundancia, pero es forzada al querer emplear el mismo punto de inicio de movimiento en todos los experimentos.

**Trayectoria lawn-mower longitudinal**

Aquí se emplea una trayectoria *lawn-mower* longitudinal más típica para la exploración y más conocida. Ejemplos de la misma puede identificarse en robots domésticos como el “roomba” y en la industria, sobre todo en centros automatizados. Se aprecia como aquí el área de influencia de nuestro vehículo también realiza un barrido visitando las casillas al menos una vez como ocurría en la trayectoria concéntrica anterior. Es cierto que los resultados obtenidos son mejores debido a que existe menos visitas repetidas en las casillas por lo que este movimiento es bastante mejor que el previamente descrito.

**Trayectoria aleatoria**

En este caso se pueden analizar los resultados obtenidos al aplicar un algoritmo de decisiones aleatorias al agente. Como cabía de esperar, los resultados muestran que un movimiento errático y concentrado donde el agente visita de nuevo algunas casillas se traduce en una recompensa bastante inferior.

Aunque se trate de un algoritmo que genera desplazamientos aleatorios, en realidad sí que sigue algunas de ellas. Se le aplicaron algunas restricciones, como por ejemplo que la zona de trabajo se limitase a las casillas que representan el entorno acuático. Se ha seguido este criterio para obtener un resultado relativamente aceptable en el experimento con este algoritmo, ya que si no se hubiesen aplicado las mismas serían muy inferiores o negativos.

**Trayectoria autónoma con algoritmo Discounted Reward**

| Hiperparámetros | Valores |
|---|---|
| **Learning_rate** | 1e-3 |
| **Clip_range** | 0.5 |
| **N_epochs** | 10000 |
| **Batch_size** | 64 |
| **Hidden_size** | 512 |

**Tabla 5.2:** Hiperparámetros del algoritmo *Discounted Reward(REINFORCE)* en escenario de importancia uniforme.

Analizando el resultado del algoritmo *Discounted Reward(REINFORCE)* en un mapa de importancia homogénea vemos que el vehículo visita reiteradamente muchas casillas y tiende a permanecer en zonas específicas. Teniendo en cuenta que el entrenamiento de la red neuronal fue realizado en el mismo escenario, este comportamiento revela que la red neuronal ha sido entrenada de manera deficiente o que el algoritmo no es el adecuado para trabajar en las condiciones que se le exige por las características del entorno y el ejercicio.

Teniendo en cuenta que el resultado mostrado se trata de unas de las mejores recompensas alcanzadas por este algoritmo y los hiperparámetros empleados en el entrenamiento del modelo, se llega a la conclusión de que no es un algoritmo adecuado para la tarea. En un entrenamiento con un valor tan alto del parámetro *n_epochs* es de esperar que el algoritmo converja absolutamente y no localmente como parece que ocurre. El episodio ha finalizado tras un poco más de 120 desplazamientos con un comportamiento que consideramos inadecuado respecto al esperado.

Cuando ejecutamos este tipo de experimentos, se suele obtener una curva de la recompensa ascendente durante toda la simulación. En cambio, analizando las figuras anteriores vemos como el vehículo, tras comenzar bien, tiende a permanecer en regiones y tratar de obtener el máximo de recompensa local en detrimento de una recompensa mayor en el futuro. La disminución de la recompensa se debe a la pérdida de batería en desplazamientos ineficientes explorando casillas con intensidad o importancia nula y el choque con los límites del mapa. En un análisis macroscópico vemos como la red con la que trabajamos representa un modelo cuyo aprendizaje ha sido divergente y por lo tanto inadecuada, este modelo es imposible que alcance un máximo absoluto en este entorno. Parece ser que la complejidad de este son las causantes que un algoritmo tan simple como el *Discounted Reward(REINFORCE)* sea inadecuado o no cumpla con las expectativas planteadas. Parece que el horizonte temporal que se tiene en cuenta es bastante corto y por tanto el agente carece de la información necesaria para desempeñar correctamente su tarea. Este hecho es una consecuencia de los problemas asociados a este algoritmo que se han comentado previamente. En este MPD la varianza que se produce entre todas las trayectorias obtenidas a partir del gradiente que a su vez genera las políticas no se va cancelando, por lo que ese error se va incrementando a lo largo de los episodios de entrenamiento y se van obteniendo resultados cada vez más alejados de la realidad [41]. Si se tienen en cuenta todas estas consideraciones, el resultado obtenido tiene sentido y es acorde al comportamiento que se merece recoger de

esta simulación.

De todas formas, puede extraerse información útil para evaluar otras alternativas estudiadas. Se puede ir extrayendo la relación directa existente entre la cobertura del mapa, movimientos no repetitivos, amplitud de trayectorias con el valor de la curva de recompensa y por tanto con una eficiente recogida de muestras.

**Trayectoria autónoma con algoritmo PPO**

| Hiperparámetros | Valores |
|---|---|
| **Learning_rate** | 2.25e-4 |
| **Clip_range** | 0.15 |
| **Clip_range_vf** | 0.5 |
| **Max_grad_norm** | 1 |
| **Ent_coef** | 0 |
| **Vf_coef** | 1 |
| **N_epochs** | 30 |
| **N_steps** | 2048 |
| **Gamma** | 0.998 |
| **Target_kl** | 0.023 |
| **Gae_lambda** | 0.95 |
| **Batch_size** | 64 |
| **Hidden_size** | 1024 |

**Tabla 5.3:** Hiperparámetros del algoritmo PPO en escenario de importancia uniforme.

En este punto se muestran las figuras obtenidas de los episodios realizados con un modelo construido a través del entrenamiento de la red neuronal aplicando un algoritmo tipo PPO. Se han ejecutado dos simulaciones exactamente iguales con una salvedad, y es que en la primera de ellas el agente trabaja con una distribución determinista de acciones y en la segunda no. La diferencia de comportamiento entre ambas consiste en que en el primer caso cuando el agente recibe el vector de probabilidades de la red neuronal, este selecciona la acción con un mayor valor. En cambio, en la segunda esto no sucede así por lo que en este caso existe una pequeña incertidumbre. Esta variación de comportamiento refleja un resultado algo distinto como se puede observar en las trayectorias y en las curvas de recompensas de las dos experiencias. Es cierto que en este escenario la diferencia del área de influencia y trayectorias es pequeña debido a que el proceso ha sido muy bien modelizado, pero quizá en otro escenario con otras características en los que la importancia de visita difiera en mayor medida, existan obstáculos, etc. pueda apreciarse una variación mayor.

Aquí, a diferencia de con el *Discounted Reward(REINFORCE)*, si vamos cancelando la presencia del error asociado a la varianza y se entrena un modelo más acorde a realidad del escenario planteado. Como consecuencia puede verse como se afronta el reto de exploración con una metodología más adecuada y un comportamiento estable [42]. El agente realiza unas trayectorias amplias visitando más eficientemente las

casillas presentes, desplazamientos más largos, se producen menos visitas reincidentes y parece que el ASV es plenamente consciente de las dimensiones del espacio de trabajo que aporta recompensas positivas. Si analizamos la evolución de los pasos del gradiente, puede destacarse que los resultados que va ofreciendo siguen una dirección de convergencia total o absoluta y el agente no permanece en regiones determinadas trabajando repetidamente en buscar un máximo local, lo que no tiene sentido en un mapa donde la importancia de todas las casillas sin visitar es máxima.

Si volvemos a analizar los dos experimentos realizados con este algoritmo podemos sacar en conclusión que es interesante mantener una pequeña arbitrariedad en la elección de decisiones en favor de que una exploración más libre. Es cierto que la metodología determinista puede ser más eficiente en casos en los que sea interesante obtener buenos resultados con algo más de rapidez en detrimento de un mejor valor final, pero en este caso no se busca dicha solución. Estas afirmaciones son consecuentes con las bases del aprendizaje por refuerzo y los procesos de decisión de Markov, por lo que viene a constatar las características esenciales de esta familia de métodos de optimización.

- **Resultados de distintas distribuciones de acciones para el algoritmo PPO**

Como sabemos, el algoritmo PPO trabaja con una distribución de acciones estocástica por lo que el conjunto de acciones que el agente puede tomar se extrae del modelo entrenado en un vector de probabilidades. Cuando seleccionamos que la elección de la acción tenga carácter determinista estamos obligando que la elección que tome el agente a partir de ese vector de probabilidades sea la siempre la acción con el valor más alto. En cierta manera reduce la componente exploratoria del sistema, ya que el modelo siempre seleccionara la acción óptima.

Si realizamos varias simulaciones con esta característica activada veremos como en todas se seguirán las mismas trayectorias y la evolución del estado del entorno será igual en todas ellas. Esto se produce porque como hemos dicho siempre el modelo elegirá la acción óptima. Todo ello queda sujeto claro está a que las condiciones iniciales de todos los experimentos sean idénticas, si no los resultados y evolución de los sistemas serán distintos entre sí.

Se muestra a continuación los resultados obtenidos en la distintas distribuciones (estocástica o determinista) en las dos versiones del escenario estudiadas. Para ello es necesario el entrenamiento por separado del algoritmo con los mismos valores de hiperparámetros, ya que realizar una explotación del modelo obtenido en escenarios tan distintos provoca que el vehículo o agente no tiene una representación fiel a su realidad.

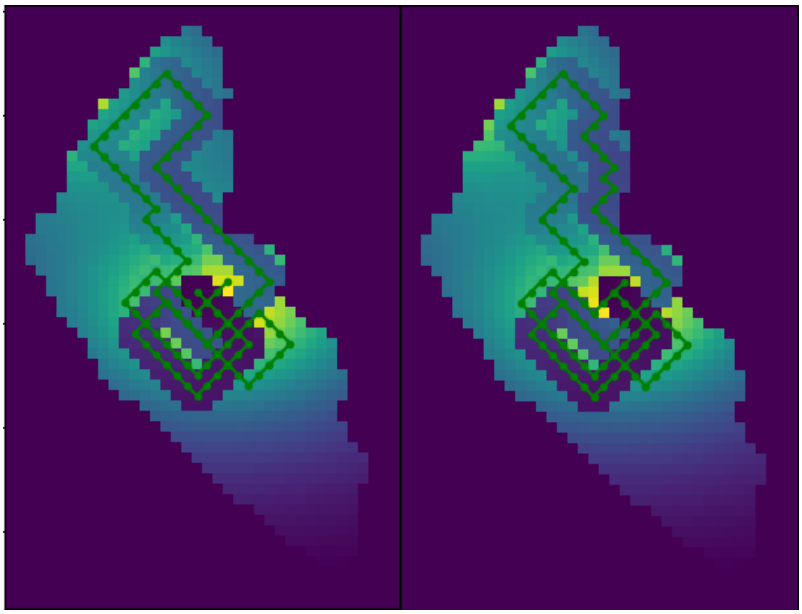


**Figura 5.24:** Exploración autónoma empleando algoritmo PPO con distribución de acciones no determinista (izquierda) y determinista (derecha) en escenario no-homogéneo.

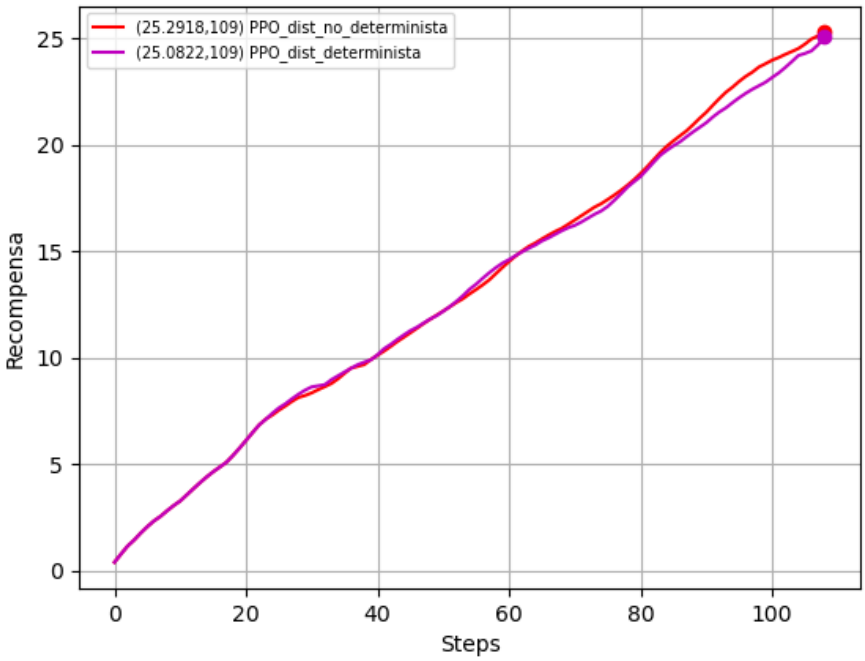



**Figura 5.25:** Evolución de la recompensa en la exploración autónoma empleando algoritmo PPO con distribución de acciones no determinista o estocástica y determinista en escenario no-homogéneo.

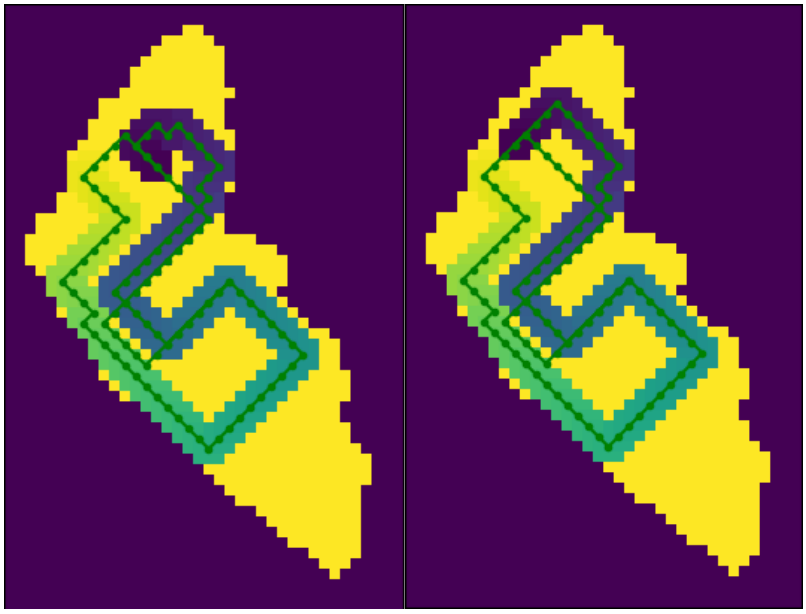


**Figura 5.26:** Exploración autónoma empleando algoritmo PPO con distribución de acciones no determinista (izquierda) y determinista (derecha) en escenario homogéneo.

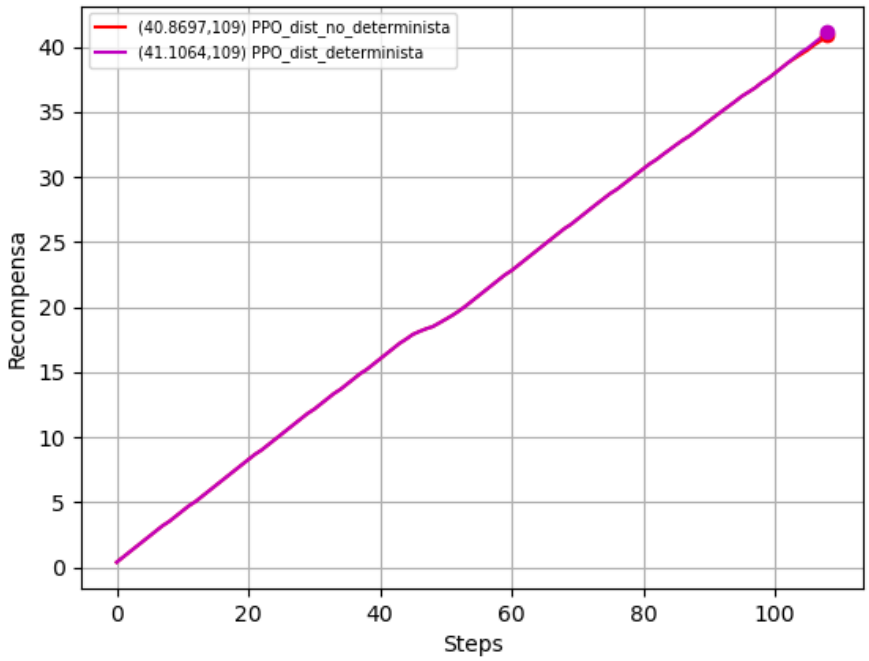



**Figura 5.27:** Evolución de la recompensa en la exploración autónoma empleando algoritmo PPO con distribución de acciones no determinista o estocástica y determinista en escenario homogéneo.

Si analizamos brevemente los dos resultados de las figuras anteriores salta a la vista que las diferentes opciones de distribución de acciones de los algoritmos son más determinantes en función de las características del medio de trabajo. Tiene sentido que en un escenario de importancia homogénea afecte menos la pérdida de aleatoriedad en la toma de direcciones ya que es una superficie en la que importa menos donde se encuentre el punto de trabajo. Por ello es algo menos importante el carácter exploratorio de nuestro algoritmo en favor de la explotación. En cambio, ocurre lo contrario en el escenario de importancia no-homogénea, donde vemos que existen zonas claves y que existe mayor importancia en la zona de trabajo, se aprecia como la diferencia entre las dos distribuciones se hace algo más destacable, aunque sigue siendo pequeña.

Es interesante ver como con esta técnica, PPO el agente tiende a visitar menos área que con las anteriores. Por supuesto la cobertura de más área o menos depende en buen grado del nivel de batería del agente y del gasto asociado a cada desplazamiento, pero al comparar en las mismas condiciones con otro experimento asociado a otro algoritmo es una característica que salta a la vista.

Los resultados obtenidos cuando trabajamos con acciones no deterministas muestran lo comentado anteriormente. Estas no serán siempre las óptimas ya que la elección de la acción a tomar estará basada en la confianza del modelo y posee una componente de aleatoriedad, pero como se explicó en el escenario con importancia homogénea ofrece mejor resultado gracias a esa pequeña arbitrariedad. La similitud de los resultados obtenidos empleado una distribución de acciones estocástica o determinista nos informa que el modelo entrenado a partir del mismo algoritmo PPO es muy bueno.

En estos experimentos, se aprecia como el método PPO ofrece un rendimiento muy oportuno observando cómo evoluciona la trayectoria del vehículo realizando desplazamientos diagonales en su mayoría y visitas poco reiteradas a las casillas que atraviesa. Es interesante ver como las virtudes de este algoritmo frente a los demás empleados destacan más en este escenario de importancia no homogénea, ya que en este caso la importancia de un modelo bien entrenado es más importante que en el escenario anterior porque todas las casillas presenten no poseen el mismo valor de importancia.

Al igual que en la versión de mapa de importancia homogéneo, aquí se ha realizado el experimento de un episodio con los hiperparámetros que ofrecen una recompensa media más favorable y un número de desplazamientos menor. Este segundo criterio de elección puede ser representativo del grado de optimización

de recursos alcanzado por el sistema de exploración-recogida de muestras.

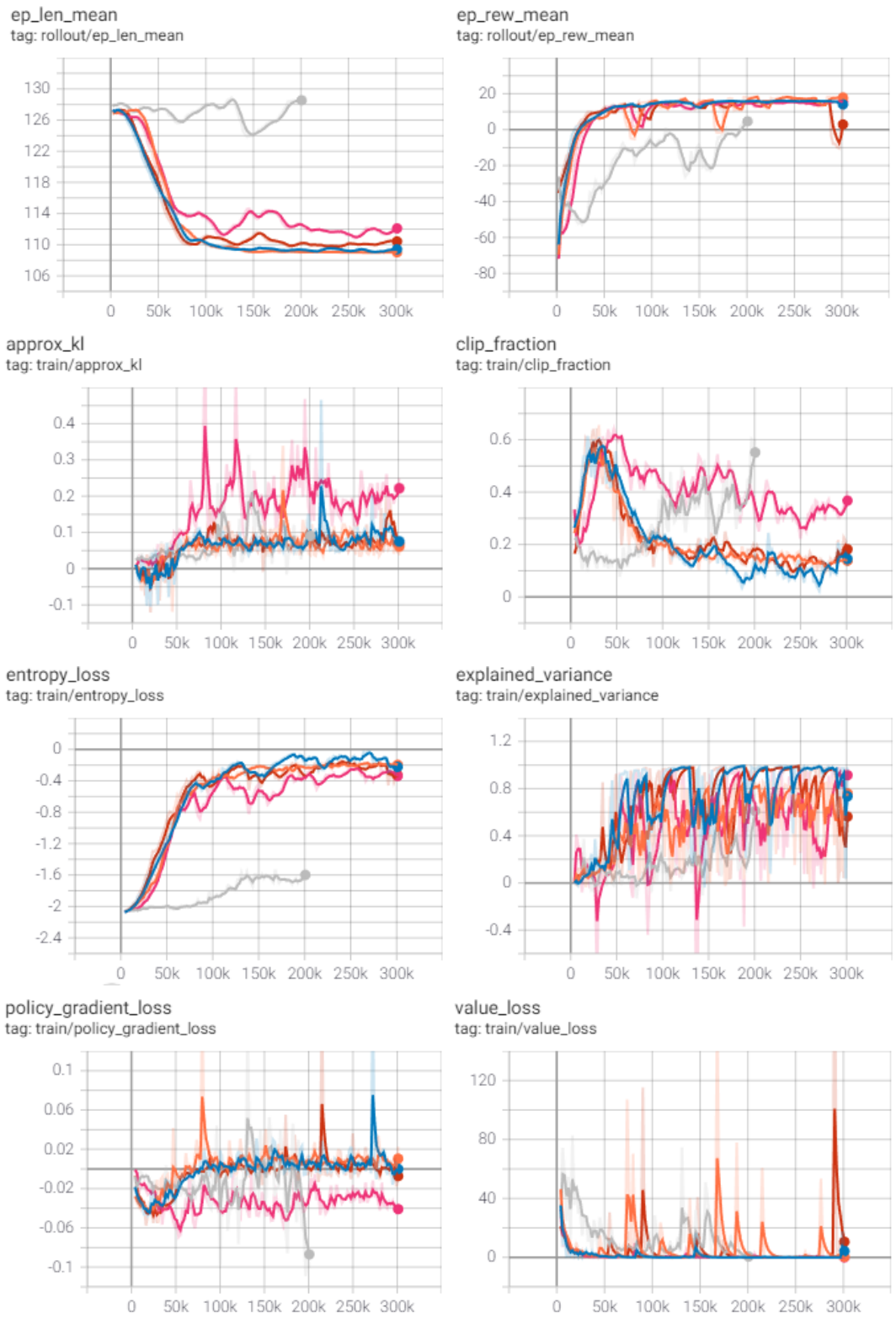



**Figura 5.28:** Características de varios modelos obtenidos a partir del algoritmo PPO durante la sintonización.

En la figura anterior se muestran varios modelos obtenidos durante la fase de sintonización del algoritmo PPO. Debido al gran número de experimentos realizados se han mostrado algunos con el fin de reflejar la evolución a través de este proceso y el buen comportamiento de los entrenamientos.

- **ep_len_mean:** refleja la longitud media de los episodios en el entorno de simulación.
- **ep_rew_mean:** aquí puede observarse la recompensa media acumulada durante los episodios.
- **approx_kl:** este parámetro aporta la diferencia que tiene lugar entre una política respecto a la anterior mediante una aproximación de la divergencia de Kullback-Leibler. Este valor tenderá a estabilizarse tras una considerable número de episodios y la estabilidad de esta curva será un indicador de un buen entrenamiento.
- **clip_fraction:** muestra la fracción del clipeado que se realiza en una nueva política respecto a la anterior. Este valor debe calcularse en cada búsqueda de una nueva política buscando un aprendizaje estable.
- **entropy_loss:** las pérdidas de la entropía reflejan en comportamiento del sistema ya que cuando el valor entero de este variable es elevado, el agente en el sistema empieza a comportarse deficientemente. En esta familia es importante conservar e carácter aleatorio del sistema, pero conservando una lógica u orden.
- **explained_variance:** este parámetro refleja si la función de valor empleada realiza una buena predicción respecto a la recompensa futura. Para valores positivos estaremos trabajando con una función de predicción adecuada.
- **policy_gradient_loss:** aquí se representa las pérdidas que se producen en el gradiente y por tanto el paso que se tomará en las iteraciones del algoritmo. Los valores ideales serán los más cercanos a 0 pero es interesante observar la estabilidad de la curva para analizar un buen entrenamiento respecto a otro más deficiente.
- **value_loss:** media de las pérdidas que se producen en la actualización de la función de valor respecto a la anterior. Cuando este valor aumenta se está produciendo una progresión positiva en el entrenamiento y cuando la recompensa tienda a estabilizarse debe reducirse.

La planificación de experimentos y obtención de resultados planteaba una serie de experimentos que están ordenados siguiendo un criterio de complejidad y buena solución al afrontar el problema planteado a resolver en sentido ascendente. Como se ha explicado, el proceso de sintonización ha ofrecido una configuración de hiperparámetros que consideramos óptima, pero durante este se ha observado una peculiaridad interesante que se comentará a continuación y que ofrece unos resultados singulares.

En el apartado 4.5 se comenta las características de una la red neuronal convolucional, que es la empleada en el algoritmo PPO y *Discounted Reward(REINFORCE)*. Como puede concluirse tras la lectura del apartado descrito anteriormente, se trata de un elemento muy importante a la hora de generar un modelo del sistema en el que estamos trabajando con el vehículo que también puede ser un elemento configurable en busca de obtener una mayor recompensa final. La configuración de esta no ha sido estudiada previamente al igual que los parámetros de los algoritmos empleados, pero no por ello ha sido un capítulo tratado trivialmente.

Tras a búsqueda de los parámetros idóneos de los algoritmos, se procede a obtener una configuración también idónea de número de capas y neuronas presentes en las redes empleadas.

Este estudio se decide realizar en el escenario donde se trabaja con una importancia no homogénea en las casillas de la superficie de trabajo, ya que se comprueba que los resultados son similares si se realizan en la otra versión más simple estudiada anteriormente.

- **Resultados de distintas configuraciones de red neuronal para el algoritmo PPO**

A continuación, se mostrará y analizará como un cambio de parámetros en la red neuronal afecta al comportamiento del agente. Las siguientes figuras muestran unos experimentos que no corresponden a episodios mostrados con anterioridad, ya que han sido realizado con un modelo distinto y se busca es realizar un análisis independiente a otros parámetros estudiados con anterioridad.

Para empezar, nos interesaba obtener el comportamiento del vehículo en el escenario empleando una configuración de *hidden_layers*=512 y con el mismo parámetro de *hidden_layers*=1024.

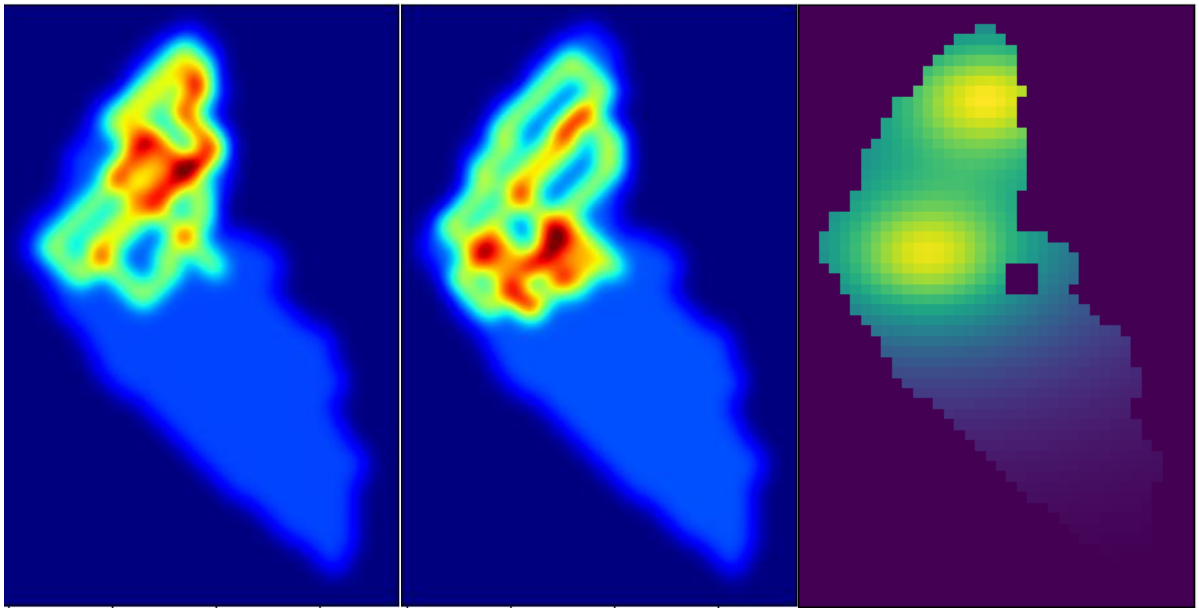


**Figura 5.29:** Mapa de calor del escenario tras el desarrollo de un episodio *hidden_layers*=512 (izquierda) y con *hidden_layers*=1024 (centro).

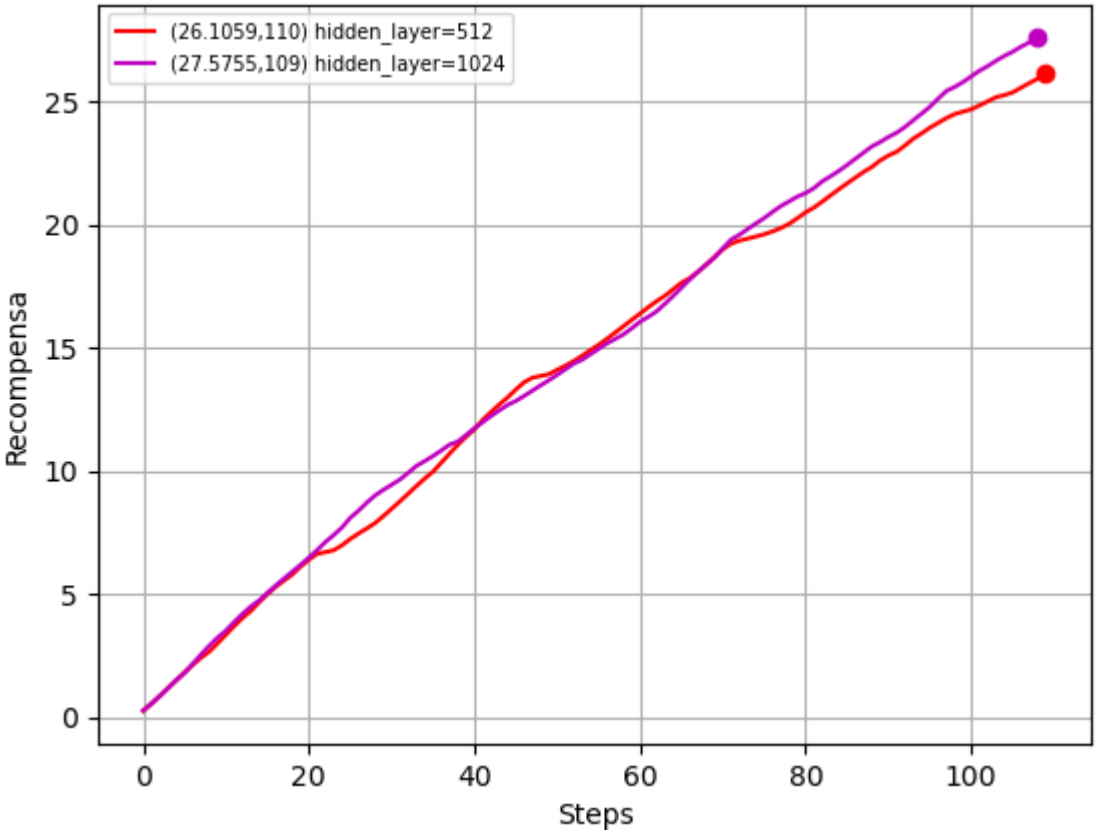



**Figura 5.30:** Evolución de la recompensa en la exploración tras el desarrollo de un episodio *hidden_layers*=512 (izquierda) y con *hidden_layers*=1024 (centro).

Si observamos el mapa de calor generado por las trayectorias del vehículo y contrastamos las zonas donde se genera unos focos de incidencia mayor con las zonas del mapa con mayor importancia, comprobamos que pueden corresponderse muchas de las mismas, pero también se concluye que el vehículo incide en zonas que se pueden considerar poco eficientes a la hora de tomar muestras.

Si observamos la Ilustración anterior, concretamente la figura de la derecha vemos que las zonas donde el vehículo incide más en su movimiento son más características de una serie de desplazamientos más eficiente al consultar el mapa de importancia no homogénea. Es cierto que se visita menos la zona norte con una importancia remarcable, pero eso no significa que sea menos idónea esta decisión, ya que a tenor de las recompensas obtenidas se aprecia que es más eficiente no visitarla y adecuar una exploración más ordenada de la zona central.

Las anteriores figuras se obtienen de varias simulaciones recogidas de la exploración del agente durante un episodio en un escenario idéntico al que se acostumbra durante el entrenamiento, por lo que se puede afirmar que una red neuronal entrenada para esa versión del escenario.

Un estudio interesante consiste en ver si se obtienen los mismos resultados tras el desarrollo de unos episodios en un escenario distinto al que se emplea para el entrenamiento de las redes.

A continuación, se muestra el resultado de explotar el modelo obtenido a partir de una red neuronal con la configuración de *hidden_layers*=512 y de nuevo con *hidden_layers*=1024 en un escenario distinto al que emplea para entrenarse.

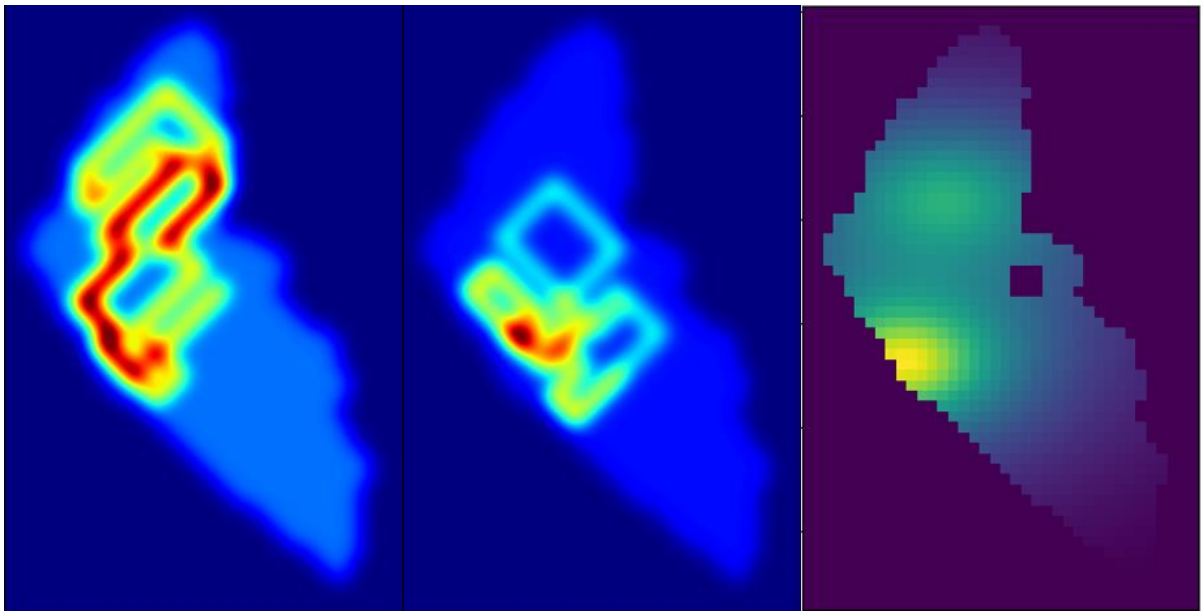


**Figura 5.31:** Mapa de calor del escenario tras el desarrollo de un episodio *hidden_layers*=512 (izquierda) y con *hidden_layers*=1024 (centro).

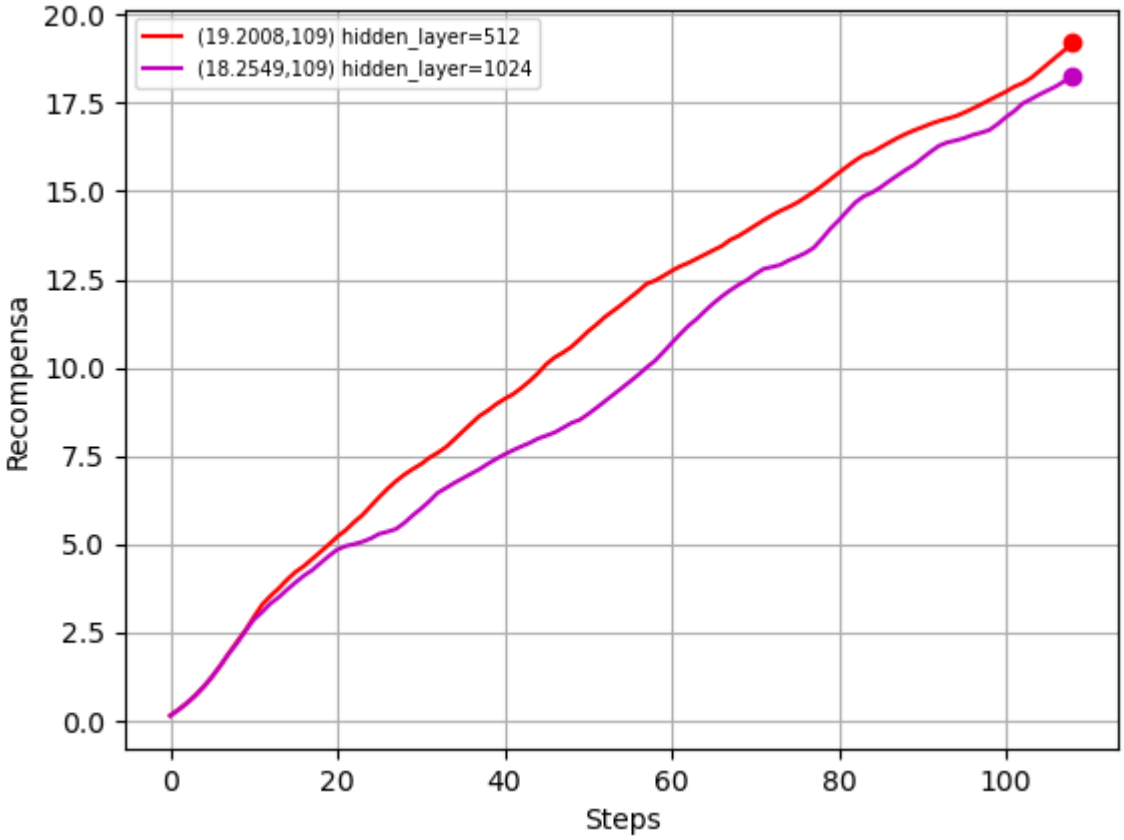



**Figura 5.32:** Evolución de la recompensa en la exploración tras el desarrollo de un episodio *hidden_layers*=512 (izquierda) y con *hidden_layers*=1024 (centro).

Vemos como el modelo se adapta aceptablemente bien a las nuevas características del escenario al que se enfrenta y centra sus movimientos en torno a la zona de importancia más destacable y por lo tanto más interesante de ser visitada. Puede apreciarse con claridad donde se centran los esfuerzos del ASV a la hora de trabajar e identificarse esa zona casi exactamente con las más importante del escenario.

En las figuras anteriores se concluyó que una red neuronal con un número superior de capas y neuronas adapta mejor el comportamiento del agente al ambiente donde se encuentre al generarse un modelo más complejo del mismo, pero como se comprobará a continuación esto no ocurre cuando el vehículo se encuentra trabajando en un escenario para el que no ha sido entrenado previamente, aunque se obtengan buenos resultados se consideran algo peores que los correspondientes a un modelo algo más simple y rápido computacionalmente hablando.

Como vemos en esta figura los desplazamientos también se realizan casi exclusivamente alrededor de la zona de más alta importancia, pero es menos adecuada a la obtenida con una red neuronal más simple ya que la exploración es más extensa y menos reiterada.

Se extrae como conclusión que el modelo obtenido con una red neuronal más compleja es más eficiente al trabajar en un escenario para el cual ha sido entrenado previamente, pero es menos adecuado al tener que trabajar en un emplazamiento desconocido previamente.

## 5.4. Comparación entre metodologías

A partir de los resultados obtenidos podemos comprobar como el algoritmo que obtiene mejores resultados a la hora de abordar el problema de exploración y análisis del recurso hídrico simulado es el PPO. Hay que tener en cuenta como se ha comentado que una variable del problema a tener en cuenta es la batería de la que

dispone el vehículo autónomo y que por lo tanto es un recurso que también hay que optimizar. En otro escenario en el que no se tuviera en cuenta este factor es posible que otros algoritmos en largos periodos de exploración obtengan una recompensa mayor ya que se pueden visitar todas las casillas.

Pero ajustándonos a los resultados obtenidos en el escenario que trabajamos, vemos que la pendiente de la curva recompensa es mayor en el PPO, así como la final obteniéndose en menos movimientos del vehículo, por lo que se puede afirmar que genera trayectorias más eficientes.

Es destacable que el vehículo suele realizar trayectorias diagonales ya que así es como se abarca más superficie con su área de influencia en cada movimiento. Este comportamiento también puede apreciarse en las trayectorias generadas por el modelo entrenado con el algoritmo *Discounted Reward(REINFORCE)* pero también llama la atención como su comportamiento hace que el vehículo tienda a priorizar zonas determinadas en la búsqueda de una posible recompensa óptima cuando hemos visto que no es así. Al intentar optimizar sus resultados en zonas localizadas vemos como pierde mucho dejando de visitar otras casillas colindantes en las que la zona de influencia del vehículo no repetiría tantas visitas y por tanto obtener una recompensa mayor.

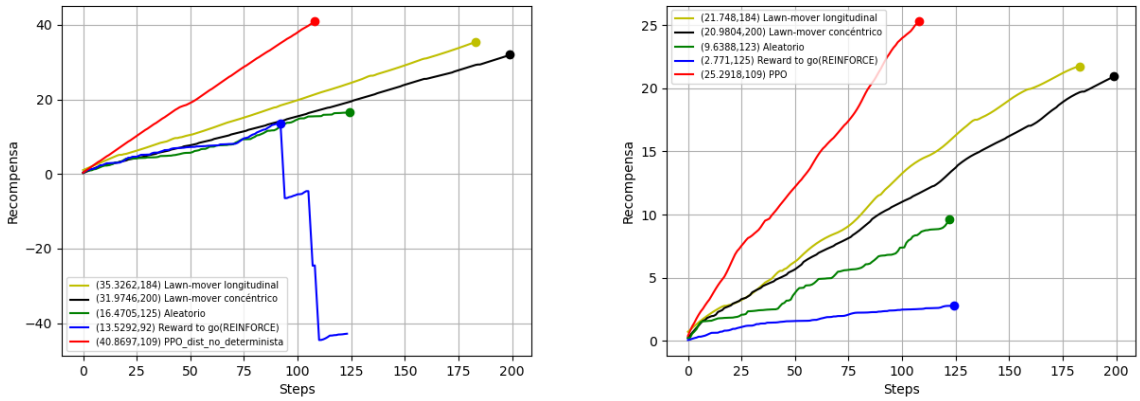



**Figura 5.33:** Recompensas obtenidas mediante diversos algoritmos durante la experimentación.

| | Recompensa media | Duración media de los episodios | Divergencia media entre experimentos |
|---|---|---|---|
| Lawn mower longitudinal | 20,49 | 188,1 | 1.2 |
| Lawn mower concéntrico | 20,56 | 179,8 | 0.9 |
| Aleatorio | -25.19 | 124,3 | 15.7 |
| Reward to Go | 5,94 | 126,7 | 5.8 |
| PPO | 25,96 | 116,4 | 2.7 |

**Tabla 5.4:** Resultados medios obtenidos en la experimentación con escenario de importancia no-homogénea.

Como era de esperar, examinando los resultados de las trayectorias realizadas por los algoritmos *lawn-mower*

vemos como la pendiente de la evolución de la recompensa en los experimentos parece casi estática, sobre todo en el mapa homogéneo. Esto es así porque las trayectorias han sido diseñadas específicamente para no superponer las áreas de influencia en el movimiento del vehículo, aunque en el caso del *lawn-mower* concéntrico si se superponen, ya que la implantación de dicho algoritmo teniendo en cuenta las dimensiones del mapa no ha sido muy adecuada. He de resaltar que, en el caso del escenario con importancia no homogénea, podría haberse diseñado una trayectoria más adecuada para esta semilla del entorno, pero no ha sido así porque se prefirió realizar una versión genérica para cualquier semilla que se trabaje. Además, no tendría lógica diseñar una trayectoria *lawn-mower* para cada punto de inicio del recurso hídrico distinto cuando lo que se busca que el vehículo sea autónomo casi en su totalidad.

Si comparamos los resultados de los algoritmos de barrido con los obtenidos del PPO, salta a la vista que este sigue siempre óptimo incluso en el mapa homogéneo, y esta comparación es la que más información nos aporta respecto a la eficacia del modelo y políticas obtenidas en los entrenamientos.

Estudiando las recompensas y trayectorias realizadas por el ASV cuando se le aplica una aleatoriedad en sus acciones, se aprecia que son bastante nefastas como cabía esperar. La única norma impuesta a este algoritmo es que el vehículo no pueda salir del escenario, por lo que los resultados estarían condicionados a esa norma y si dicha regla no existiera probablemente estaríamos observando unos resultados peores aún. Dichos experimentos nos aportan un punto de inicio a la hora de elaborar una buena idea sobre la virtud de los resultados obtenidos con otros algoritmos, por eso se han planteado añadirlos en este banco de resultados.

## 5.5. Análisis de los resultados

Los resultados obtenidos han recopilados tras la realización de una serie de experimentos ejecutados en unas condiciones controladas y homogéneas para así garantizar la verosimilitud de la información que pueda extraerse de los mismos.

Es cierto que, dada las experiencias recopiladas mediante diversos métodos de generación de trayectorias, previamente a la presentación de los resultados, se podía prever un comportamiento inadecuado o resultados un poco decepcionantes. En el caso que se ha tenido claro desde un principio que podía definirse con las anteriores características se ha tomado la decisión de presentar los mejores resultados obtenidos, ya que actuando así incluso se ve que el resultado con dicho algoritmo no podría acercarse incluso en virtud del resultado obtenido por el mejor algoritmo estudiado. Se ha tomado esta decisión para resaltar aún más el buen comportamiento de los modelos y sus trayectorias correspondientes que si son adecuados para la tarea de exploración de nuestro entorno. Estas consideraciones serán comentadas en el siguiente apartado con mayor profundidad.

A la hora de evaluar las experiencias, se muestran los recorridos realizados y las curvas de recompensa donde se muestra la evolución de ésta a lo largo del tiempo y en que instante se obtiene el mayor valor. Es una buena forma de conocer la convergencia de los algoritmos propuestos a la hora de entrenar los modelos y el comportamiento de los mismos a lo largo de un episodio propuesto.

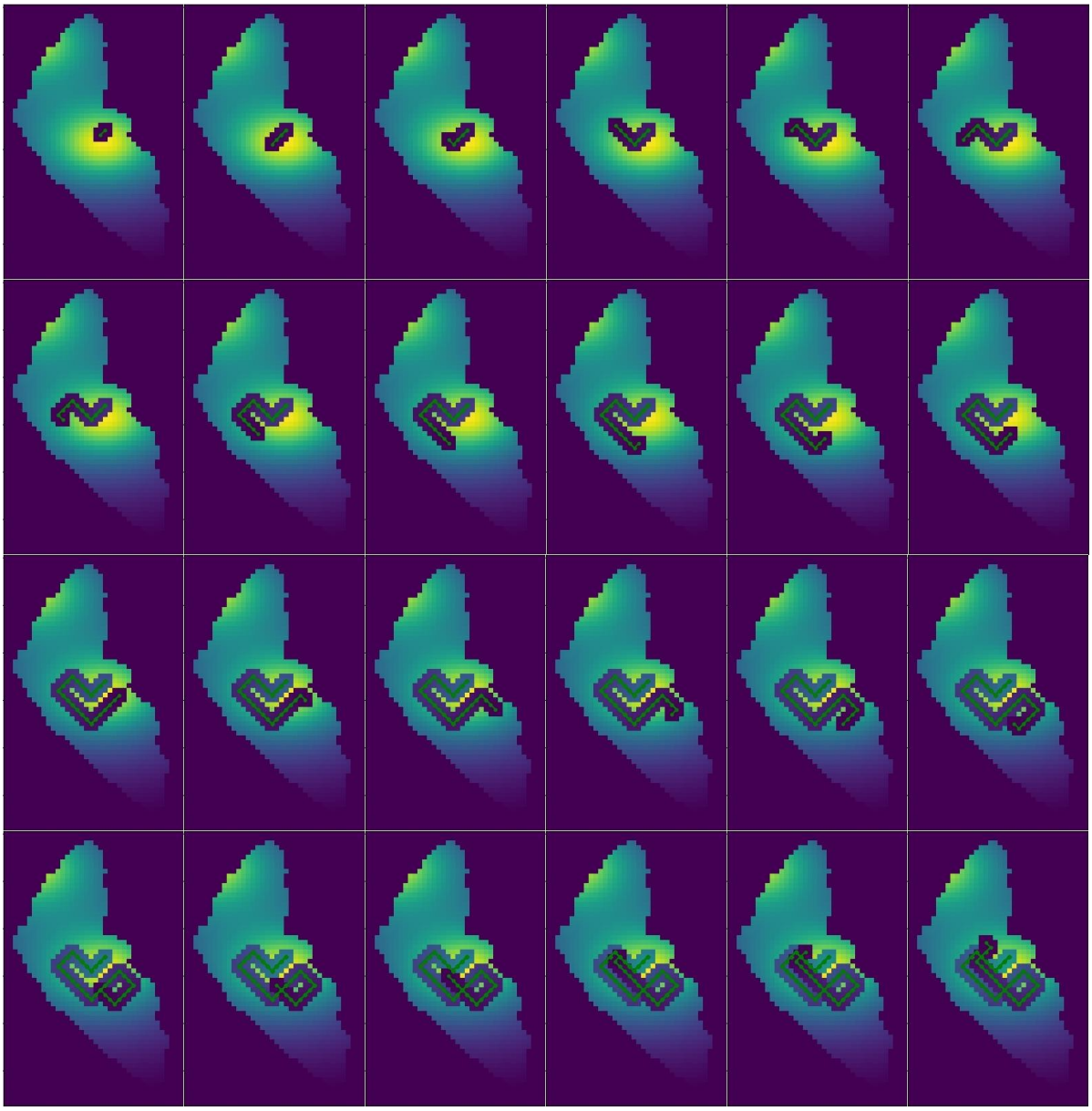
Aunque el análisis de los episodios puede aportar mucha información en determinados instantes críticos, como por ejemplo cuando las trayectorias siguen camino que pueden cruzarse o donde se alcanzan límites del mapa, donde es interesante analizar la carga computacional, la longitud del horizonte predictivo de los algoritmos y varias consideraciones interesantes, observando estas imágenes finales con detenimiento se pueden ver muchas de estas características importantes en la evolución de los entrenamientos.

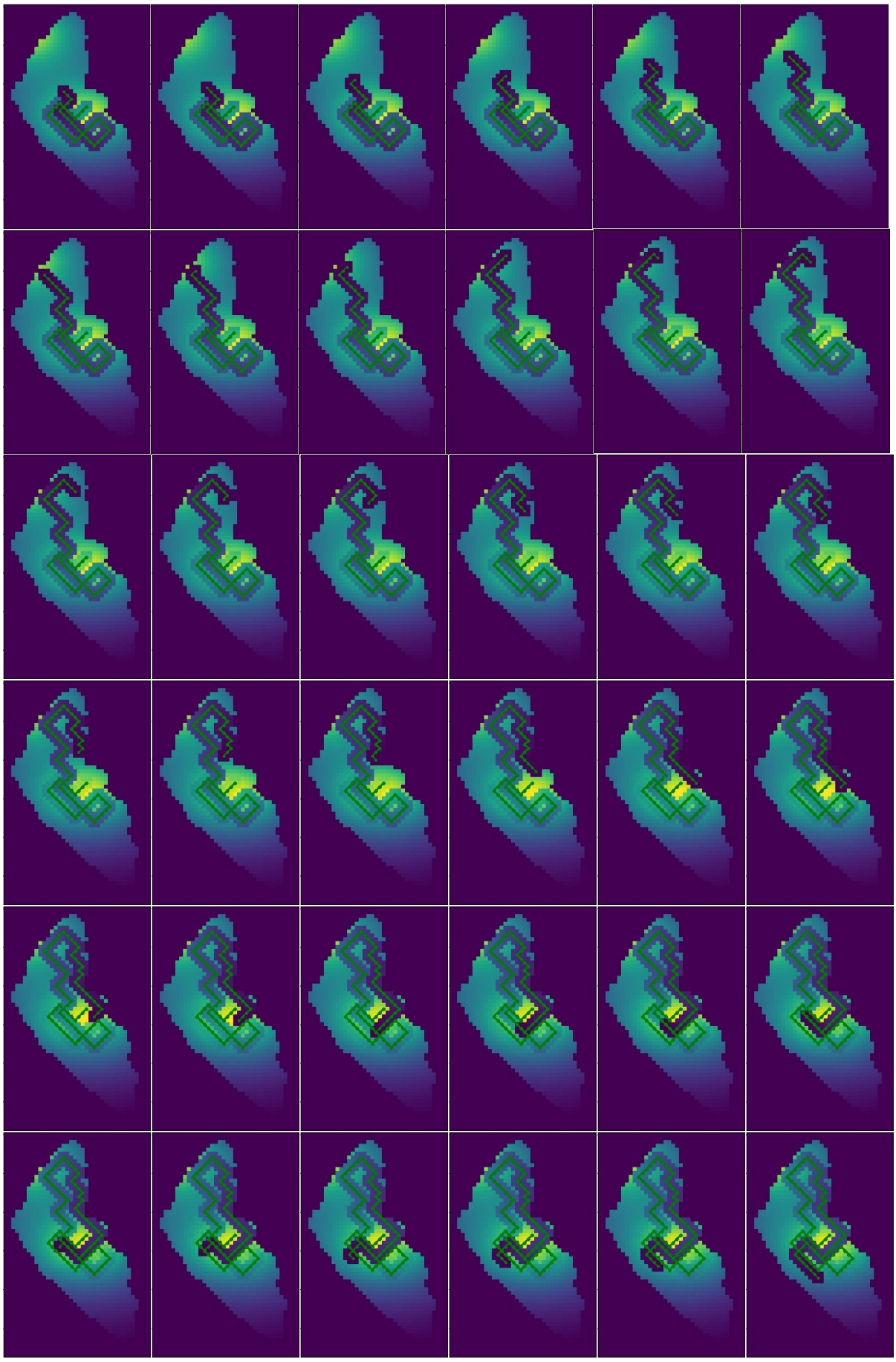
No se ha destacado mucho la importancia de la cobertura total en estos experimentos. Aunque es un a aspecto muy importante normalmente en ejercicios de exploración y navegación, para nosotros si estamos trabajando en la región cuyas casillas no mantienen una importancia homogénea y constante es más importante estudiar si las regiones más interesantes son visitadas y los resultados obtenidos en un limitado número de desplazamientos que nos condiciona la batería del vehículo. No se quiere decir que no es importante estudiar si se visita repetidamente ciertas casillas con el área de influencia del vehículo, ya que este hecho suele incurrir en una disminución de la eficiencia del ejercicio, solo se afirma que los inconvenientes asociados a la visita repetida de una casilla pueden ser suavizados o eliminados por la regeneración automática de la importancia en casillas definida en el entorno. Por tanto, la cobertura no es una característica que aporte información

absolutamente concluyente.

Se considera que los resultados obtenidos aportan una visión, de los experimentos realizados y del comportamiento de los modelos que han sido entrenados previamente mediante varias metodologías distintas, suficientemente compleja para que se puedan analizar los puntos fuertes y débiles de estos para afrontar la tarea propuesta. En el apartado siguiente se analizará técnicamente el uso de los diversos algoritmos empleados, así como sus detalles para poder justificar la elección del uso de la mejor metodología basándonos en el *feedback* de los episodios realizados en el entorno.

A continuación, se muestra una simulación paso a paso del vehículo empleando el modelo entrenado mediante el algoritmo PPO en el escenario de importancia no homogénea.





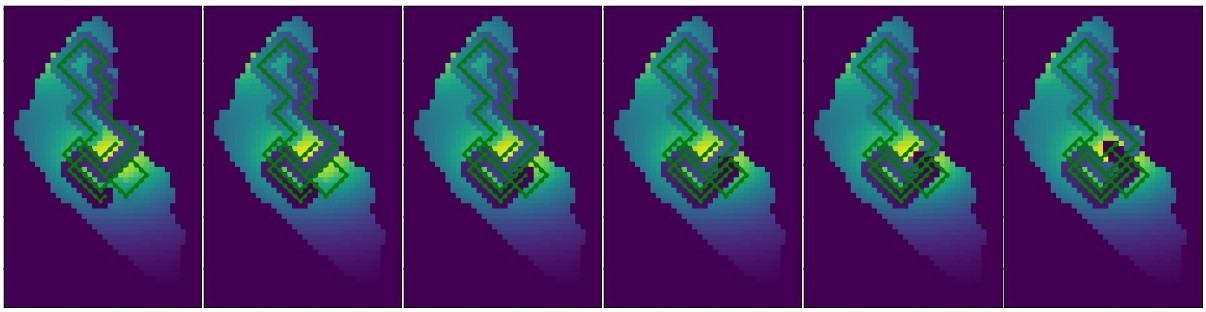


**Figura 5.34:** Simulación paso a paso del vehículo durante un episodio de exploración con algoritmo PPO.

Aquí se han mostrado las imágenes de un episodio completo del modelo generado por el algoritmo PPO trabajando con decisiones deterministas en el entorno de importancia no homogénea. Creemos que puede ser la simulación que puede aportar un mejor comportamiento en la simulación de la exploración de nuestro vehículo en el recurso hídrico natural, por lo que quiere mostrarse visualmente como trabajaría paso a paso el ASV y las trayectorias generadas a lo largo de su desplazamiento. Es interesante ver en esta serie de imágenes como se optimiza la recompensa optimizando la cobertura máxima del mapa teniendo en cuenta la regeneración del valor de visita en ciertas casillas a lo largo del tiempo. Este factor nos dice mucho de la potencia de aprendizaje a través del método de aprendizaje por refuerzo, capaz de optimizar varias variables y su evolución a lo largo del tiempo en un entorno complejo.

# 6. CONCLUSIONES Y LÍNEAS FUTURAS

## 6.1. Conclusiones

En este documento se han analizado como la aplicación de técnicas y metodologías de aprendizaje automático pueden ofrecer una solución a problemas y retos que previamente necesitasen ser abordados mediante el uso de otras vías más complejas que normalmente conlleven el uso de una cantidad de recursos menos eficientes.

El *Machine Learning* hoy en día se ha convertido en una herramienta presente en casi todos los aspectos de la actualidad con la que se ha podido automatizar tanto procesos y la eficiencia de estos como el estudio y modelizado de problemas que la humanidad no ha podido afrontar en el pasado.

Un campo muy interesante que ha sufrido avances muy generosos en poco tiempo gracias al análisis de datos y su interpretación ha sido de la genética. La posibilidad de poder analizar automáticamente y con eficiencia una cantidad ingente de imágenes ha permitido por ejemplo a detección temprana de enfermedades que pueden resultar letales para el ser humano en una etapa más desarrollada como los tumores. Actualmente se está trabajando con una inteligencia artificial basada en algoritmos de aprendizaje por refuerzo que es capaz de detectar melanomas en la piel gracias a una base de datos aportada por ciertas entidades médicas. Lo interesante de este sistema es que cuanto más trabaje más desarrollara su capacidad de detección y la precisión de la misma por sí misma, además de aportar la posibilidad de acceder a este tipo de herramientas a personas que no puedan visitar a personal médico específico para un primer diagnóstico.

El uso de esta tecnología permite desarrollar estrategias de resolución y optimización frente a tareas que plantean escenarios de trabajo con muchos grados de libertad y multitud de soluciones posibles gracias a la búsqueda de políticas eficientes que permiten la adaptación de los recursos presentes a los escenarios que se le plantee buscar una solución. La versatilidad, adaptación y eficiencia computacional son los puntos fuertes de esta familia de métodos que ya forman parte de los pilares sobre los que se asientan los avances científicos de nuestra generación.

Si hablamos de la experimentación realizada, se ha comprobado que mediante el uso de algoritmos pertenecientes a la familia de aprendizaje por refuerzo profundo se ha podido generar modelos entrenados capaces de poder optimizar las soluciones al problema planteado de exploración del recurso hídrico Ypacaraí de manera eficiente. Dada la complejidad del problema, se ha conseguido desarrollar una metodología automatizada mediante la que el vehículo autónomo de superficie es capaz de generar trayectorias adaptativas al escenario en el que trabaje obteniendo mejores resultados que los que podría generarse mediante el uso de otros algoritmos de generación de trayectorias más simples y conocidas en muchos sectores tecnológicos como el guiado automático en robótica. En estudios realizados sobre la aplicación de algoritmos de aprendizaje automático en la detección e interpretación de diversos tipos de cáncer se han obtenido resultados muy esperanzadores. Por ejemplo, el uso de dicha tecnología en la detección del cáncer de pecho, a partir de un paquete de datos para enseñar al algoritmo, permite analizar si se trata de un tumor maligno o benigno con una precisión de un 94% [43]. Las consecuencias de poder analizar clínicamente más eficiente y rápidamente a la

población aumentarían la tasa de detección y posible cura de estas enfermedades tan graves, debido a que está demostrado que una detección precoz aumenta en gran medida la tasa de posible recuperación frente a estas enfermedades.

Los resultados obtenidos tras emplear el algoritmo PPO han demostrado que su uso consigue que el agente pueda mejorar la recompensa alcanzada mediante técnicas *lawn-mower* en un 126,049% y en un 786.47% si partimos de los resultados obtenidos con el algoritmo *Discounted Reward(REINFORCE)* debido a que este algoritmo presenta una deficiencia importante al trabajar con la complejidad del escenario estudiado.

Como se ha comentado es interesante que los algoritmos que se han obtenido sean adaptables a otros entornos acuáticos con otra morfología y que un estudio más exhaustivo de campo pueda permitir la sintonización de los hiperparámetros oportunos para obtener resultados en un medio físico. Se abre por tanto un camino de mejora particular a este estudio a corto plazo y como se comentará a continuación a lo largo de un periodo de tiempo en el futuro que permitirá la experimentación con otro tipo de algoritmos interesantes y especificaciones más complejas del problema planteado.

## 6.2. Líneas futuras

Tras la finalización de este trabajo se muestra muy interesante la continuación de la tarea de experimentación y búsqueda de otro tipo de soluciones posibles de emplear. La evolución de la técnica de aprendizaje por refuerzo ha traído la aparición de nuevos algoritmos incluso dentro de la familia off-policy que podrían merecer un estudio, como el DDPG o el TRP también adaptados para poder trabajar en medios continuos por lo que parece muy adecuado su posible uso en este mismo escenario. También parecen muy oportunos los algoritmos de la familia *Actor Critic*, que son algoritmos que trabajan al igual que el PPO con métodos de optimización de políticas, destacando el posible comportamiento de los *Advantage Actors Critic*(A2C) y los *Asynchronous Advantage Actors Critic*(A3C). Estos últimos algoritmos comentados aparecen mucho en la experimentación reciente ya que permiten un aprendizaje óptimo y si ya se hablaba previamente de la adaptabilidad de la técnica *Deep Reinforcement Learning*, el desarrollo del A3C ha supuesto una evolución notable ya que la robustez y adaptabilidad a entornos discretos o continuos en diversas investigaciones lo colocan en una posición muy interesante para ser abordado en este proyecto.

Teniendo en cuenta que en un siguiente paso a la investigación de la implementación de algoritmos de aprendizaje por refuerzo en el comportamiento autónomo de un ASV sería poder conseguir una flota multi-vehículo que consiga trabajar simultáneamente, parece la mejor opción a estudiar ya que la complejidad del problema a estudiar y su solución no son triviales. Por lo tanto, se figura como protagonista a esta familia de algoritmos para abordar dicha línea futura de investigación.

**Figura 6.1:** Flota de vehículos acuáticos con operatividad simultánea [45].

A diferencia del problema enfocado en este trabajo, cuando se trabaje con más vehículos se añadirá otra tarea a resolver. Adicionalmente a optimizar las tareas de exploración individuales, se necesitará resolver el colisionamiento de los vehículos y un abordamiento multiobjetivo de la función a realizar por la flota para

tratar de obtener finalmente la mejor recompensa media entre todos los vehículos.

Un enfoque muy interesante para próximas actualizaciones sería trabajar con un mapeado de los entornos acuáticos más precisos tras un estudio sobre el terreno para poder trabajar con una caracterización más fiel a la realidad del entorno. La recogida de datos climáticos, morfométricos, así como hidroquímicos serían vitales para un enfoque más profundo de los proyectos a realizar.

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# TABLA DE ACRÓNIMOS

A2C: Advantage actor critic

A3C: Asynchronous advantage actor critic

Adam: Adaptive moment estimation

ASV: Autonomous Surface Vehicle

AUV: Autonomous Unmmaned Vehicle

CPU: Central processing unit

DBO: Demanda bioquímica de oxigeno

DDQN: Dueling deep Q networks

DDPG: Deep deterministic policy gradient

DGPS: Differential global positioning system

DKL: Deep kernel learning

DQN: Double Q Learning with neural Network

DPG: Deterministic policy gradient

GPS: Global positioning system

GPU: Graphics processing unit

MDP: Markov decision process

ONG: Organización no gubernamental

PID: Proportional – integral - derivative controller

PPO: Proximal policy optimization

QT-OPT: Distributed Q-learning algorithm

RELU: Rectified linear unit

SAC: Soft actor-critic

SB3: StableBaselines3

TD3: Twin delayed DDPG

TRPO: Trust region policy optimization

UGV: Unmanned ground vehicle

UAV: Unmanned aerial vehicle

USV: Unmanned surface vehicle)

# 7. BIBLIOGRAFÍA

[1] M. a. L. S. a. L. J. Cheung, «Toxin-producing cyanobacteria in freshwater: A review of the problems, impact on drinking water safety, and efforts for protecting public health,» *Journal of microbiology (Seoul, Korea)*, vol. 51, nº 10.1007/s12275-013-2549-3, pp. 1-10, 2013.

[2] G. López Moreira M., L. Hinegk, A. Salvadore, G. Zolezzi, F. Hölker, R. Monte Domecq S., M. Bocci, S. Carrer, L. De Nat, J. Escribá, C. Escribá, G. Benítez, C. Ávalos, I. Peralta, M. Insaurralde, F. Mereles, J. Sekatcheff y Wehrl, «Eutrophication, Research and Management History of the Shallow Ypacaraí Lake (Paraguay),» *Sustainability 2018*, vol. 10, p. 2426.

[3] C. C. H. D. N. S. O. G. B. C. Á. C. M. R. E. J. A. Inocencia Peralta, Estudio de sistemas de fitodepuración con islas flotantes de Typha SP (TOTORA) en el lago Ypacaraí, Universidad Nacional de Asunción - Centro Multidisciplinario de Investigaciones Tecnológicas- San Lorenzo; Universidad Nacional de Asunción - San Lorenzo, 2014.

[4] A. R. Balbuena, Estudio de la contaminación del lago Ypacaraí, Sevilla: Departamento de Ingeniería Química, Escuela Politécnica Superior de Ingeniería, Universidad de Sevilla, 2019.

[5] S. Yanes Luis, Aplicación de técnicas de Deep Reinforcement Learning a misiones de exploración para vehículos autónomos en escenarios lacustres, (Trabajo Fin de Máster Inédito). Universidad de Sevilla, Sevilla, 2020.

[6] D. G. R. S. L. T. M. Samuel Yanes Luis, «A multiagent deep reinforcement learning approach for path planning in autonomous surface vehicles: The Ypacaraí lake patrolling case,» *IEEE Access*, vol. 9, pp. 17084-17099, 2021.

[7] D. G. R. S. L. T. M. Samuel Yanes Luis, «A Deep Reinforcement Learning Approach for the Patrolling

Problem of Water Resources Through Autonomous Surface Vehicles: The Ypacarai Lake Case,» *IEEE Access*, vol. 8, pp. 204076-204093, 2020.

[8] J. E. a. H. G. Manley, Unmanned Surface Vessels (USVs) as tow platforms: Wave Glider experience and results, OCEANS 2016 MTS/IEEE Monterey, 2016, pp. 1-5.

[9] D. S. a. D. H. a. F. O. a. M. L. a. S. Sargento, «A Platform of Unmanned Surface Vehicle Swarms for Real Time Monitoring in Aquaculture Environments,» *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[10] S. Y. D. G. R. S. T. Federico Peralta, «Monitoring Water Resources through a Bayesian Optimization-based Approach using Multiple Surface Vehicles: The Ypacarai Lake Case Study,» de *IEEE Congress on Evolutionary Computation (CEC)*.

[11] N. a. L. N. a. A. W. M. H. Ahmad, «Autonomous Lawnmower using FPGA implementation,» *IOP Conference Series: Materials Science and Engineering*, vol. 160, p. 12112, 2016.

[12] D. G. R. Iván Cabezas Moraga, Aplicación de Path Planning de un Robot, 2017.

[13] Z. Z. P. D. Xiaoyun Lei, «Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning,» *Journal of Robotics, vol. 2018*, 2018.

[14] D. a. R. R. a. C. A. a. S. K. a. F. K. Isele, «Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning,» *Computer and information sciences*, 2017.

[15] X. Z. J. S. Yingjun Ye, «Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity simulation environment,„» *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 155-170, 2019.

[16] S. a. A. a. S. A. a. P. S. S. a. P. V. G. a. K. P. P. V, Solar-Powered Trash Collecting Boat with Solar Power Prediction using Machine Learning and Human-Computer Interface, 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021, pp. 262-269.

[17] H.-C. a. H. Y.-L. a. H. S.-S. a. O. G.-R. a. W. J.-R. a. H. C. Chang, «Autonomous Water Quality Monitoring and Water Surface Cleaning for Unmanned Surface Vehicle,» *Sensors*, vol. 21, pp. 1424-8220, 2021.

[18] Z. a. G. L. Wang, «Evolution of the spatial structure of a thin phytoplankton layer into a turbulent field,» *Marine Ecology-progress Series - MAR ECOL-PROGR SER*, vol. 374, pp. 57-74, 2009.

[19] J. a. K. Y. Borenstein, «Real-time obstacle avoidance for fast mobile robots,» *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, nº 5, pp. 1179-1187, 1989.

[20] T. M. B. U. P. C. Feras Dayoub, «Vision-only autonomous navigation using topometric maps,» de *Intelligent Robots and Systems (IROS), 2013*, 2013.

[21] M. a. S. S. a. T. M. a. C. C. a. K. A. a. S. R. a. N. J. Pfeiffer, «Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations,» *IEEE Robotics and Automation Letters*, vol. 3, nº 4, pp. 4423-4430, 2018.

[22] C. J. R. M. F. M. H. E. a. M. A. Hartmut Surmann, Deep Reinforcement learning for real autonomous mobile robot navigation in indoor environments, Computer Science Department, University of Applied Science Gelsenkirchen, 2020.

[23] D. G.-R. S. T. M. Samuel Yanes Luis, «A dimensional comparison between evolutionary algorithm and deep reinforcement learning methodologies for autonomous surface vehicles with water quality sensors,» *Sensors*, vol. 21, nº 8, p. 2862, 2021.

[24] L. a. N. D. S. F. a. M. J. a. F. N. a. L. J. a. M. R. a. C. P. Santos, «Path Planning for Automatic Recharging System for Steep-Slope Vineyard Robots,» *Advances in Intelligent Systems and Computing*, pp. 261-272, 2018.

[25] S. M. a. P. Corke, «Path planning using surface shape and ground properties,» *Proceedings of the 2011 Australasian Conference on Robotics and Automation (ACRA 2011)*, pp. 1-7, 2011.

[26] P. B. E. V. H. J. a. I. V. Tokekar, «Tracking aquatic invaders: Autonomous robots for monitoring invasive fish,» *IEEE Robotics & Automation Magazine*, vol. 20(3), pp. pp.33-41, 2013.

[27] V. a. D. M. a. S. I. Nikishin, «Autonomous Unmanned Surface Vehicle for Water Surface Monitoring,» *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 14, pp. 853-858, 2020.

[28] J. E. Manley, ««Unmanned Surface Vehicles, 15 Years of Development»,» *IEEE Oceanic Engineering Society*, 2008.

[29] I. Y. J. S. a. C. C. B. Chung, «"Error compensation of GPS using sensor fusion in intelligent vehicle,",» *4th International Conference on Autonomous Robots and Agents*, pp. 278-283, 2009 , 2009.

[30] A. M.-A. L. D. F. A.Pozo-Ruz, Sistema de posicionamiento global (GPS): Descripción, Análisis de Errores, Aplicaciones y Futuro, E.T.S. Ingenieros de Telecomunicación. Universidad de Málaga.

[31] J. A. C. J. J. L. a. P. M. N. 2. .. 2. p. 3.-3. M. R. Benjamin, «"Navigation of unmanned marine vehicles in accordance with the rules of the road,",» *IEEE International Conference on Robotics and Automation, 2006*, pp. 3581-3587, ICRA 2006.

[32] R. &. L. J. &. W. L.-F. &. M. N. Lambert, « Robust ASV Navigation Through Ground to Water Cross-Domain Deep Reinforcement Learning.,» *Frontiers in Robotics and AI.*, 2021.

[33] M. L. |. M. A. |. A. G. |. M. C. I. Errecalde, ««Aprendizaje por Refuerzo aplicado a la resolución de problemas no triviales,»,» *II Workshop de Investigadores en Ciencias de la Computación*, mayo 2000.

[34] E. a. B. P. L. a. B. J. Greensmith, «Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning,» *Journal of Machine Learning Research*, vol. 5, 2004.

[35] B. Richard S.Sutton and Andrew G, Reinforcement learning: an introduction, 2018.

[36] Z. a. H. Z. a. Y. Y. a. Y. Y. Wang, «Research on PPO algorithm in solving AUV path planning problem,» 2021 2nd International Seminar on Artificial Intelligence,» *Networking and Information Technology (AINIT)*, pp. 73-79, 2021.

[37] J. a. W. F. a. D. P. a. R. A. a. K. O. Schulman, Proximal Policy Optimization Algorithms, arXiv, 2017.

[38] T. a. P. G. a. L. M. a. Y. K. Blum, «RL STaR Platform: Reinforcement Learning for Simulation based Training of Robots,» 2020.

[39] J. S. a. P. M. a. S. L. a. M. J. a. P. Abbeel, «High-Dimensional Continuous Control Using Generalized

Advantage Estimation,» 2018.

[40] A. a. S. F. Raffin, «Generalized State-Dependent Exploration for Deep Reinforcement Learning in Robotics,» 2020.

[41] A. T. a. D. D. C. a. S. Mannor, «Learning the Variance of the Reward-To-Go,» *Journal of Machine Learning Research*, vol. 17, nº 13, pp. 1-36, 2016.

[42] Y. R. S. E. L. Q. S. L. L. a. K. L. Y. Guan, «Centralized Cooperation for Connected and Automated Vehicles at Intersections by Proximal Policy Optimization,» *IEEE Transactions on Vehicular Technology*, vol. 69, nº 11, pp. 12597-12608,, Nov 2020.

[43] S. a. A. A. a. C. T. Sharma, «Breast Cancer Detection Using Machine Learning Algorithms,» *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pp. 114-118, 2018.

[44] A. a. K. A. a. M. N. a. I. H. a. F. Z. a. K. M. a. A. A. a. B. B. a. K. A. a. H. I. a. C. G. Azar, «Drone Deep Reinforcement Learning: A Review,» *Electronics*, vol. 10, nº 10.3390/electronics10090999, 2021.

[45] M. a. C. J. a. L. J. a. N. P. Benjamin, «Navigation of unmanned marine vehicles in accordance with the rules of the road,» *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 3581-3587, 2006.

[46] J. L. N. M. Reeve Lambert, Robust ASV Navigation Through Ground to Water Cross-Domain Deep Reinforcement Learning, 2021.

[47] Z. a. H. Z. a. Y. Y. a. Y. Y. Wang, «Research on PPO algorithm in solving AUV path planning problem,» *2021 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pp. 73-79, 2021.

[48] A. R. Balbuena, Estudio de la contaminación del lago Ypacaraí, Escuela Politécnica Superior de Ingeniería, Universidad de Sevilla: Departamento de Ingeniería Química, 2019.