ISSN 1688-2784

Universidad de la República
Facultad de Ingeniería

Control adaptativo para veleros autónomos

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Agustín Rieppi y Florencia Rieppi

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero en Computación.

Tutores

Gonzalo Tejera ..... Universidad de la República
Mercedes Marzoa ..... Universidad de la República

Tribunal

Martín Llofriu ..... Universidad de la República
Luis Chiruzzo ..... Universidad de la República
Jorge Visca ..... Universidad de la República

Montevideo
miércoles 12 octubre, 2022

Control adaptativo para veleros autónomos, Agustín Rieppi y Florencia Rieppi.

ISSN 1688-2784

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 71 páginas.
Compilada el miércoles 12 octubre, 2022.

Resumen

En los últimos tiempos los veleros autónomos no tripulados han tomado gran relevancia. Esto se debe principalmente a las ventajas que los veleros tienen sobre las demás embarcaciones acuáticas en cuanto a la eficiencia energética, dado que es el viento la mayor fuente de energía que posibilita su navegación por períodos indefinidos. Uno de los grandes desafíos a los que se enfrentan los veleros autónomos es lograr navegar en diversas condiciones climáticas. Es por ello, que uno de los problemas que se encuentra al producir este tipo de veleros es la creación de un controlador que sea capaz de enfrentarse a distintas condiciones climáticas y que sea adaptable a diferentes veleros y al desgaste de estos.

Para lograr una solución a este problema, en este proyecto se quiere encontrar alternativas al uso de controladores estáticos programados a priori. Para ello, se busca generar un controlador adaptativo utilizando aprendizaje por refuerzo combinado con aprendizaje por imitación, haciendo uso de un simulador para su entrenamiento y evaluación.

Como resultado de los distintos experimentos realizados se obtienen dos controladores capaces de navegar en múltiples condiciones ambientales, teniendo un mejor desempeño que el controlador estático utilizado como referencia. Se destaca que en el entrenamiento de estos controladores se utiliza únicamente aprendizaje por refuerzo dado que el aprendizaje por imitación no resulta beneficioso en este proyecto.

Tabla de contenidos

Resumen I

Índice de figuras V

Índice de tablas VI

1. Introducción **1**
1.1. Presentación del problema y contexto 1
1.2. Estructura del documento 4

2. Estado del arte **7**
2.1. Simuladores 7
2.1.1. Comparativa de simuladores 7
2.1.2. Unmanned Surface Vehicle Simulator with Realistic Environmental Disturbances 8
2.2. Controladores 10
2.2.1. Comparativa entre controladores 10
2.2.2. Control of Autonomous sailboats 11
2.2.3. Reinforcement Sailing 12

3. Marco teórico **13**
3.1. Aprendizaje por Refuerzo 13
3.1.1. Algoritmos de aprendizaje por refuerzo 16
3.2. Aprendizaje por imitación 18
3.2.1. Generative Adversarial Imitation Learning 19
3.2.2. Adverserial Inverse Reinforcement Learning 19
3.3. Robot Operating System 20

4. Solución propuesta **21**
4.1. Definición y configuración del entorno de aprendizaje 21
4.1.1. Biblioteca de aprendizaje por refuerzo y aprendizaje por imitación 21
4.1.2. Interfaz entre el entorno y el aprendizaje por refuerzo 22
4.2. Definición e implementación del controlador 23
4.2.1. Tarea a realizar y su contexto 23
4.2.2. Espacio de observación y acción 23
4.2.3. Ejecución de acción y obtención de observación 24
4.2.4. Función de recompensa 25
4.2.5. Fin de episodio 26
4.3. Arquitectura de la solución 26

Tabla de contenidos

5. Experimentación **29**
5.1. Métricas . . . 29
5.2. Escenarios . . . 30
5.2.1. Generación automática . . . 30
5.2.2. Generación manual . . . 32
5.3. Trayectorias del controlador manual . . . 34
5.4. Pruebas pre-experimentales . . . 34
5.4.1. Pruebas en uso de aprendizaje por imitación . . . 34
5.4.2. Pruebas sobre función de recompensa . . . 36
5.4.3. Otras pruebas . . . 37
5.5. Pruebas experimentales . . . 38
5.5.1. Entrenamiento y evaluación cuantitativa entre controladores . . . 38
5.5.2. Evaluación cualitativa entre controladores . . . 41
5.5.3. Evaluación cambiando la configuración del velero . . . 43
5.5.4. Otras pruebas . . . 45

6. Conclusiones y trabajos a futuro **47**
6.1. Conclusiones . . . 47
6.2. Trabajos a futuro . . . 48

Referencias **51**

Glosario **55**

Anexo **57**
6.3. Gráficas con los resultados de las pruebas de recompensa . . . 57
6.4. Gráficas con los resultados de las pruebas de imitación . . . 59

Índice de figuras

1.1. Partes del velero 3
1.2. Rumbos del barco respecto al viento 4

2.1. Simulación velero - Enfoque superior 9
2.2. Simulación velero - Enfoque lateral 9
2.3. Arquitectura USVSim 10

3.1. Diagrama de interacción entre el agente y el ambiente 15

4.1. Arquitectura de la solución propuesta 27

5.1. Escenarios - Generación automática 31
5.2. Escenarios - Controlador manual 32
5.3. Escenarios - Pruebas pre-experimentales 33
5.4. Escenarios - Evaluación cualitativa 33
5.5. Evaluación cuantitativa - Distancia acumulada promedio 40
5.6. Evaluación configuración del velero - Distancia acumulada promedio . 44

6.1. Función de recompensa uno - Distancia acumulada promedio 57
6.2. Función de recompensa dos - Distancia acumulada promedio 58
6.3. Función de recompensa tres - Distancia acumulada promedio 58
6.4. Entrenamiento RL - Distancia acumulada promedio 59
6.5. Entrenamiento GAIL - Distancia acumulada promedio 59
6.6. Entrenamiento AIRL - Distancia acumulada promedio 60
6.7. Entrenamiento GAIL + RL - Distancia acumulada promedio 60
6.8. Entrenamiento AIRL + RL - Distancia acumulada promedio 61

Índice de tablas

2.1. Comparación de simuladores 8
2.2. Comparación de controladores 11

5.1. Resultados aprendizaje por imitación 35
5.2. Función de recompensa - Cantidad de episodios 37
5.3. Evaluación cuantitativa - Resultados 39
5.4. Evaluación cualitativa - Recompensa final 41
5.5. Evaluación cualitativa - Distancia acumulada 42
5.6. Evaluación cualitativa - Llega al objetivo 42
5.7. Evaluación configuración del velero - Resultados 44

Capítulo 1

Introducción

1.1. Presentación del problema y contexto

En la actualidad existe un gran interés por entender la evolución y estado actual del clima con el objetivo de combatir el cambio climático. Para ello es necesaria la obtención e investigación de distintos datos atmosféricos. La extracción de muestras del océano y mares es un área donde existen grandes dificultades para la obtención de datos de forma sistematizada. Los buques oceanográficos se han utilizado para recopilar estos datos, pero no resultan muy eficientes, ya que tienen un alto costo de operación y su disponibilidad es muy limitada.

Dada la necesidad de la obtención de datos del océano y los grandes avances tecnológicos en las ciencias de la computación es que surge la motivación del uso de vehículos acuáticos autónomos no tripulados para realizar esta tarea, buscando que la extracción de datos se realice de forma automática, con la intención de reducir los costos. Dentro de la gran variedad de vehículos acuáticos autónomos, el velero se destaca por sobre otros por el hecho de que no necesita una fuente de energía adicional al viento para la propulsión y como consecuencia es más ecológico, presentando como desventaja una mayor complejidad para la navegación del mismo.

Aún cuando la construcción del hardware necesario para controlar estos veleros continúa siendo un desafío importante, el problema principal en la automatización de los veleros es cómo diseñar un controlador robusto que sea capaz de manejar los múltiples controles de un velero y adaptarse al proceso de desgaste natural de la embarcación, como el estiramiento de la vela o la degradación del timón, entre otros componentes. Esto es desafiante si se espera que el velero navegue en mar abierto por largos períodos de tiempo, en el que se enfrentará a condiciones adversas y deberá ser capaz de soportar tormentas, grandes olas, fuertes vientos y otras adversidades climáticas.

La creación de veleros autónomos tiene una gran variedad de aplicaciones posibles, algunas de ellas son:

- Recolectar datos relevantes para la investigación sobre el cambio climático o el calentamiento global.
- Obtener muestras de agua para medir su calidad.
- Adquirir datos para realizar la estimación de la posible falta de agua dulce.

- Obtener información relevante acerca de catástrofes naturales como son las inundaciones en ríos y lagos.
- Controlar prácticas ilegales que puedan afectar el medio ambiente o ser contraproducentes para el país.

En el año 2021 se comienza un proyecto en conjunto con universidades de Brasil, Chile, Francia y Uruguay, donde se busca crear un velero autónomo que sea capaz de navegar durante semanas, e incluso meses, sin la necesidad de la intervención humana. Si bien existe una gran variedad de aplicaciones, el objetivo principal del proyecto interuniversitario es la recolección de datos de utilidad para la investigación del cambio climático y la obtención de muestras de agua de forma autónoma. El proyecto se compone de varias partes, desde trabajo más relacionado con el hardware hasta trabajo puramente de software. La Universidad Federal de Río Grande del Norte de Brasil, se centra en el hardware, creando el velero con todos los sensores y actuadores necesarios para lograr la automatización de su control. En nuestro proyecto se presenta un controlador adaptativo para un velero autónomo, el cual es responsable de conducir al velero a un punto objetivo y permanecer en él, siendo capaz de navegar en distintas condiciones climáticas.

El controlador propuesto se basa en un modelo de aprendizaje por refuerzo, este es entrenado utilizando aprendizaje tradicional combinado con aprendizaje por imitación, ambos entrenamientos se realizan en un entorno simulado. El aprendizaje por imitación tiene la finalidad de utilizar conocimiento de expertos para acelerar el entrenamiento. En este proyecto se busca comprobar si la utilización de este tipo de aprendizaje favorece el entrenamiento. El objetivo del uso de aprendizaje por refuerzo es crear un controlador que sea capaz de generalizar y adaptarse a los distintos entornos y al desgaste del velero, de forma que pueda ser de utilidad a lo largo del tiempo para una misma embarcación, e idealmente que sea capaz de adaptarse a distintos tipos de veleros. El uso de un simulador es fundamental para poder llevar a cabo este tipo de aprendizaje, dado que se requiere un gran volumen de datos para entrenar el modelo. La obtención de esos datos del mundo real conlleva un costo muy elevado, debido a que es necesario la creación del agente, sumado a la posible ruptura de este debido a los errores cometidos durante el entrenamiento. Además del elevado costo, el tiempo de aprendizaje en un entorno real es mayor que en la simulación, dado que en esta última se puede acelerar el tiempo e incluso paralelizar el aprendizaje. Es por estas razones que el aprendizaje por refuerzo sin el uso de un simulador resulta inviable.

Para poder entender muchas de las decisiones tomadas y los impactos que estas tienen, es necesario entender algunos aspectos básicos de la navegación de los veleros y sus componentes principales, donde también resulta relevante entender consideraciones específicas de los veleros autónomos no tripulados. A continuación se detallan aspectos claves de estos temas.

Existe una inmensa variedad de veleros con distintas características, sin embargo se pueden mencionar algunos elementos importantes de la embarcación que son comunes e indispensables para todos ellos, estos son:

- El casco
- Las velas
- El timón
- La quilla u orza

Los componentes mencionados se pueden observar en la figura 1.1.

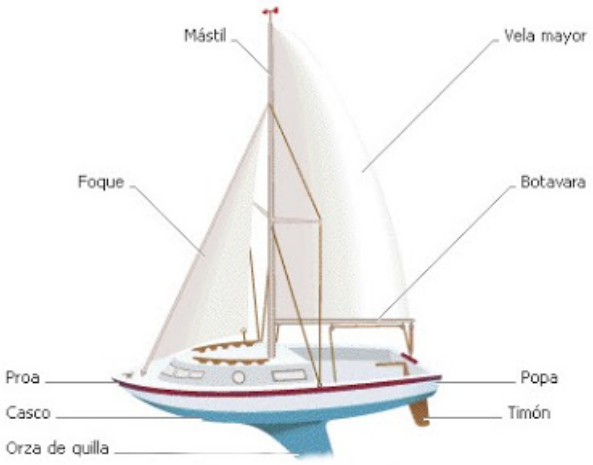

Figura 1.1: Partes del velero, imagen original obtenida de [1]

El casco se encuentra en contacto directo con el agua y le proporciona flotabilidad a la embarcación. En un velero autónomo, el casco debe ser cerrado para evitar que los componentes electrónicos tengan contacto con el agua. Las velas generan la propulsión de la embarcación y junto al timón definen el rumbo del velero. Por último, la quilla u orza genera estabilidad evitando que el velero se desvíe. La orza tiene la ventaja de ser removible, por lo tanto, los veleros que cuentan con orza en vez de quilla tienen la capacidad de poder navegar por zonas llanas o rocosas.
Los veleros logran el movimiento por el accionar del viento sobre su/s vela/s. El viento genera una presión superior sobre uno de los lados de la vela, lo que provoca que se genere una fuerza desde la alta presión hacia la baja. El componente lateral de esa fuerza es absorbido en su mayoría por la quilla u orza, provocando que el movimiento del velero sea mayormente hacia adelante.

Es importante considerar los distintos rumbos que la embarcación puede tomar, estos se definen respecto al viento aparente, es decir, el viento percibido desde la embarcación. Se considera un círculo en el que el ángulo de cero grados representa la dirección de donde viene el viento. El rumbo de proa es un rumbo no navegable y abarca entre los 40 y 45 grados hacia un lado y otro del círculo. Si se desea alcanzar un lugar en esta dirección, se debe navegar realizando zigzags en el rumbo de ceñida. Este rumbo es el más cercano en el que se logra avanzar contra el viento, aproximadamente a los 45 grados. Otro de los rumbos relevantes de conocer es el de través, donde el velero se posiciona perpendicular al viento. Es de los primeros rumbos que se enseñan al comenzar a navegar. El rumbo de largo se logra posicionando el velero a 130 grados aproximadamente del viento hacia ambos lados. Por último, el rumbo de popa es cuando el viento proviene por detrás de la embarcación, a 180 grados. Los rumbos mencionados se pueden visualizar en la figura 1.2

La posición de la vela y del timón es en gran parte lo que determina el rumbo del velero, por lo que para lograr avanzar en una dirección determinada es de gran importancia poseer el control de ambos.

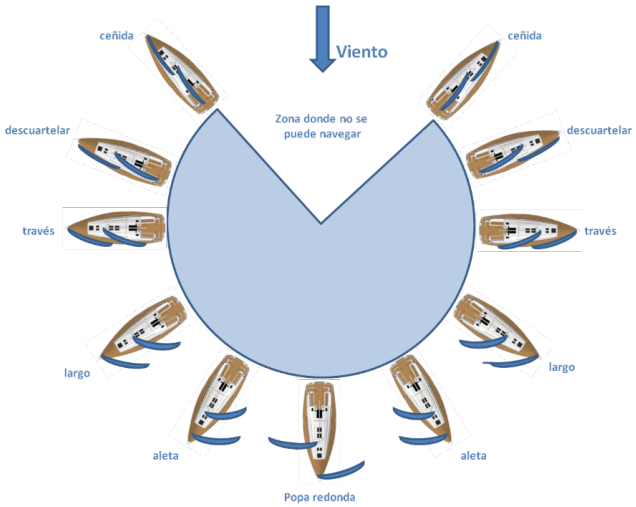

Figura 1.2: Rumbos del barco respecto al viento, imagen original obtenida de [2]

Los veleros autónomos no tripulados han tomado gran relevancia en los últimos tiempos, en gran parte por las ventajas de los veleros sobre las demás embarcaciones acuáticas, las cuales se mencionaron anteriormente. Dentro de los usos más usuales se encuentran las carreras marítimas [3] y la toma de muestras de agua [4, 5]. Los proyectos de veleros autónomos utilizados para carreras marítimas buscan optimizar la precisión y la velocidad de estos para un desafío determinado. En cambio, los proyectos relacionados con la toma de muestras de agua se esfuerzan en mejorar los veleros para que sean capaces de recorrer largas distancias sin la necesidad de la intervención humana, con el fin de aumentar la cantidad de muestras obtenidas y lograr un mejor control de los estados de los océanos.

1.2. Estructura del documento

Este documento se compone de seis capítulos. Los primeros tres están relacionados a la investigación y diseño de la solución propuesta y los últimos tres a la implementación y análisis de la solución propuesta.

El documento queda organizado de la siguiente forma:

- Capítulo 1: Se detalla el problema a abordar, su contexto y la estructura del documento.
- Capítulo 2: Se presenta el estado del arte, donde se describen las principales soluciones actuales en la simulación y control de veleros.
- Capítulo 3: En este capítulo se presentan aspectos teóricos que enmarcan el trabajo, detallando las herramientas más importantes utilizadas y su funcionamiento.
- Capítulo 4: Aquí se presenta la solución propuesta, describiendo el trabajo realizado junto a las justificaciones de las decisiones tomadas.
- Capítulo 5: Este capítulo detalla los experimentos realizados y los resultados obtenidos.

- Capítulo 6: Se presentan las conclusiones finales sobre los resultados obtenidos en la experimentación y se discuten las principales líneas de investigaciones a seguir a partir de lo realizado en este proyecto.

Capítulo 2

Estado del arte

En este capítulo se presenta un resumen del estado del arte realizado en el marco de este proyecto [6], donde se describen diversos trabajos y herramientas relacionados a la temática abordada. Para cada uno de estos se detalla un breve análisis y las conclusiones.

El estado del arte realizado se compone de dos grandes partes, la primera se centra en la simulación de vehículos acuáticos, haciendo especial hincapié en los veleros. Esta es un área clave para el trabajo presentado, ya que el entrenamiento y la experimentación se realiza a través de un simulador, por lo que el análisis de estos busca entender qué tan fiel es respecto a la realidad y cuánta información provee.
En la segunda área se realiza un relevamiento de los controladores existentes para vehículos acuáticos, donde también se hace especial hincapié en controladores de veleros y en aquellos controladores de vehículos acuáticos que utilizan aprendizaje por refuerzo. En esta sección se busca entender las soluciones actuales a la tarea que se quiere abarcar en este proyecto, con el propósito de tener una base de conocimiento en las dificultades y problemáticas que esta conlleva. Otro de los objetivos es la búsqueda de un controlador ya implementado para utilizar en el aprendizaje por imitación y tener como línea de base en las comparaciones finales.

2.1. Simuladores

En esta sección se describen brevemente las soluciones existentes para las simulaciones de vehículos acuáticos y las características de cada uno de ellos, las cuales permiten entender cuán fiel es la simulación respecto a un bote real, además es un buen punto de partida para elegir el que más se adecua para una determinada tarea.
Luego de presentada la comparativa se procede a hacer una descripción en profundidad sobre el simulador que los autores de este documento creen que se destaca por sobre el resto en la simulación de veleros.

2.1.1. Comparativa de simuladores

Existen varios simuladores disponibles, algunos con especial énfasis en la simulación de vehículos subacuáticos [7,8], mientras que otros se centran en vehículos acuáticos [9,10]. Para la comparativa se define una serie de características relevantes para facilitar la tarea de ver diferencias y similitudes. Se excluye el dato de si es de código abierto y si es interoperable con la herramienta Robot Operating System (ROS), la cual es

presentada en la sección 3.3, ya que todos los proyectos mencionados a continuación lo son. Características relevantes:

- Olas: puede simular (✓), no puede simular(✗)
- Flotabilidad: se mueve en un solo plano (✗), se mueve en un solo plano y en plano vertical (✓), se mueve en ambos planos y además incluye rotación (✓✓).
- Corrientes: no se simulan corrientes de agua (✗), se simulan corrientes de agua constante (✓), se simulan corrientes de agua variables (✓✓).
- Vientos: no se simula el viento (✗), se simula el viento de forma constante (✓), se simula el viento de forma variable (✓✓).
- Dinámica de fluidos: no se simula la dinámica de fluidos (✗), se simula la dinámica de fluidos (✓).
- Veleros: tiene la capacidad de simular veleros (✓) o no (✗).

Simulador	**Olas**	**Flotabilidad**	**Corrientes**	**Viento**	**Dinámica de fluidos**	**Veleros**
UWSim [7]	✗	✓	✗	✗	✗	✗
FreeFloating-Gazebo [8]	✗	✓	✓	✗	✗	✗
Robotx [10]	✓	✓✓	✗	✓	✗	✗
USVSim [9]	✓	✓✓	✓✓	✓✓	✓✓	✓
Autonomous Sailboat [11]	✗	✓	✗	✓	✗	✓
Sailing Robot [12]	✗	✗	✗	✓	✗	✓
SailBoatROS [13]	✗	✗	✗	✓✓	✗	✓

Tabla 2.1: Comparación de simuladores. Se resalta en verde cuando cuenta con la mayor capacidad en la categoría, en rojo la menor y en amarillo se representa un valor medio.

En la tabla comparativa 2.1 se logra ver que muchos de los simuladores evaluados no cumplen con una característica fundamental que es la simulación de veleros. Dentro de los otros cuatro se logra ver una clara superioridad del simulador USVSim por sobre el resto de las demás características evaluadas, es por ello que a continuación se realiza una profundización de este simulador. Otras características que no se evaluaron en un principio, pero que tienen gran importancia para la utilización de cualquier herramienta, son la documentación y si los autores se encuentran activos en la respuesta de preguntas. Es por ello que se cree importante mencionar que el simulador USVSim cuenta con una buena documentación que hace posible entender su funcionamiento y uso de forma relativamente rápida y además se encuentran respuestas a las consultas realizadas por la comunidad, por lo que se considera que los autores se encuentran activos.

2.1.2. Unmanned Surface Vehicle Simulator with Realistic Environmental Disturbances

El simulador Unmanned Surface Vehicle Simulator with Realistic Environmental Disturbances (USVSim) [9] se crea en base a la necesidad de disponer de un simulador para vehículos acuáticos que sea lo más fiel posible a la realidad, incluyendo la mayor cantidad de fuerzas naturales, por ejemplo el viento y las corrientes marítimas. Además, ofrece distintos modelos de vehículos acuáticos. El trabajo incorpora y mejora diversos proyectos existentes relacionados con la simulación de vehículos acuáticos.

Otras de las características a destacar es que cuenta con seis grados de libertad para distintos tipos de vehículos acuáticos sin tripulación, y la simulación, entre otras variables, incluye condiciones del medioambiente como viento, olas y corrientes.

Características:

El simulador ofrece cuatro modelos de embarcaciones, estos son un hidrodeslizador, un bote diferencial, un bote con un solo propulsor y un timón, y por último ofrece un velero que se compone de una vela, una quilla y un timón. Todas las embarcaciones tienen la capacidad de subdividir el casco en distintas cantidades, por defecto se dividen en seis partes. Esta subdivisión del casco genera que la flotación del barco sea más realista, dado que cada parte es afectada de distinta manera.
Un aspecto que hace destacar este simulador por sobre otros es la cantidad de variables del ambiente que se toman en cuenta para la simulación, algunas de las más trascendentes son el viento, las corrientes marítimas, las olas y la flotabilidad.
Cuenta con una gran variedad de sensores facilitados por el uso de Gazebo, herramienta de simulación la cual es descrita en la sección 3.3, por ejemplo, sensores de temperatura, de pH, de corrientes marítimas, de posición y medidas inerciales de la embarcación y sensores de distancia, entre otros.

A continuación se presentan imágenes ilustrativas del simulador seleccionado.

Figura 2.1: Simulación del velero enfocado desde arriba con el simulador USVSim

Figura 2.2: Simulación del velero enfocado de costado con el simulador USVSim

A continuación se encuentra un diagrama de la arquitectura del simulador, mostrando los principales componentes de USVSim y sus relaciones, muchas de las relaciones son basadas en el uso de la herramienta ROS, presentada en la sección 3.3.

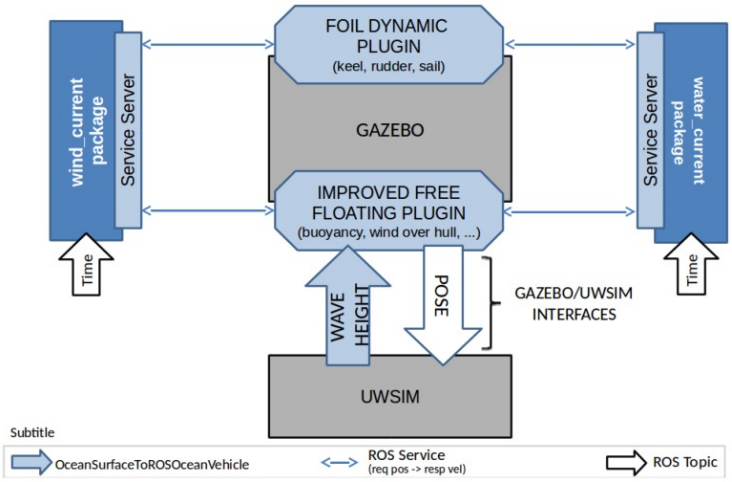

Figura 2.3: Arquitectura USVSim, imagen original obtenida de [9]

Gazebo [14] es utilizado como motor principal para la simulación y UWSim [7] para la visualización. Al núcleo principal compuesto por Gazebo se le agregan dos complementos, el primero es uno existente llamado Free Floating [8] al cual se le incorporan mejoras y el segundo, llamado Foil Dynamic, fue desarrollado como parte del proyecto USVSim.
Gazebo se encarga de comunicar a UWSim la posición del robot haciendo uso de los complementos. Como información de entrada se utiliza el viento y las corrientes de agua, la cual se obtiene a partir de dos servicios. Se utiliza también la altura de las olas recibidas a partir de UWSim. Con la información mencionada se calcula la posición tomando en cuenta todas las fuerzas aplicadas sobre el bote.

2.2. Controladores

En esta sección, al igual que se hizo para los simuladores, se detalla brevemente las soluciones existentes de controladores de vehículos acuáticos y las características de cada uno de ellos.
Luego de presentada la comparativa se procede a hacer una descripción en profundidad sobre dos de los controladores que los autores de este documento concluyen que son de mayor aporte para el proyecto.

2.2.1. Comparativa entre controladores

En esta comparativa se muestran distintos tipos controladores, algunos enfocados en el control de botes a motor [15–17] y otros para el control de veleros. Para estos últimos algunos presentan técnicas de control adaptativo [18–20] y otros no [21–24].

A continuación se definen una serie de características relevantes para facilitar la tarea de comparar diferencias y similitudes en la tabla 2.2:

- Veleros: controlador de velero (✓), controlador de otro vehículo acuático(✗).
- Timón: no lo controla o no se menciona(✗), lo controla estáticamente(✓), lo controla adaptativamente (✓✓). Si corresponde se agrega el algoritmo utilizado.
- Vela/Motor: no lo controla o no se menciona (✗), lo controla estáticamente(✓), lo controla adaptativamente (✓✓). Si corresponde se agrega el algoritmo utilizado.
- Planificación local: no realiza una planificación local o no se menciona (✗), realiza una planificación estática (✓), realiza una planificación adaptativa (✓✓). Si corresponde se agrega el algoritmo utilizado.
- Planificación global: no realiza una planificación global o no se menciona (✗), realiza una planificación estática (✓), realiza una planificación adaptativa (✓✓). Si corresponde se agrega el algoritmo utilizado.
- ROS: no integrable con ROS o no se menciona y no se cuenta con el código disponible por lo que no es posible determinarlo (✗), integrable con ROS (✓).
- Código libre: no se encuentra el código disponible (✗), se encuentra el código disponible (✓). En caso de que se cuente con la información, se adiciona el lenguaje de programación utilizado.

Controlador	Veleros	Timón	Vela/Motor	Planificación Global	Planificación Local	ROS	Código libre
Ctrl: [21]	✓	✓	✓ (Función lineal)	✗	✓ (SR, SL)	✓	✓ (Python)
Ctrl: [22]	✓	✓ (PID)	✓ (Función lineal)	✗	✓ (SL)	✗	✗
Ctrl: [23]	✓	✗	✗	✗	✓ (SL)	✗	✗
Ctrl: [24]	✓	✓ (P, PD, SC)	✓ (Función lineal)	✗	✓ (SL)	✗	✗
Ctrl: [18]	✓	✓ (PID)	✓ (Función lineal)	✓✓ (AQ)	✓	✓	✗
Ctrl: [19]	✓	✓ (PD)	✓✓ (AD)	✓ (Distancia euclidiana)	✓ (SR)	✗	✗ (C)
Ctrl: [20]	✓	✓✓ (AQ)	✓✓ (AQ)	✗	✓ (SR)	✗	✗
Ctrl: [15]	✗	✓✓ (AQRNP, GPDP)	✓✓ (AQRNP, GPDP)	✗	✓ (SL)	✗	✓ (Python)
Ctrl: [16]	✗	✓	✓	✗	✓✓ (MCPPEM)	✗	✗ (Matlab)
Ctrl: [17]	✗	✓	✓	✗	✓✓ (AQRNP)	✗	✗

Tabla 2.2: Comparación de controladores. SR (Seguidor de rumbo), SL(Seguidor de Línea), AQRNP (Aprendizaje Q con Redes Neuronales Profundas), GPDP (Gradiente de Política Determinista Profunda), AD(Árboles de Decisión), PD(Proporcional-Diferencial), PID(Proporcional-Integral-Derivada), AQ(Aprendizaje Q), SC(Seno Controlador), MCPPEM (Modelo de Control Predictivo Probabilístico con Eficiencia de Muestra). Se resalta en verde cuando cuenta con la mayor capacidad en la categoría, en rojo la menor y el amarillo representa un valor medio.

2.2.2. Control of Autonomous sailboats

El trabajo [21] tiene como objetivo crear un sistema de control para veleros, buscando ser una base para la investigación y enseñanza en veleros autónomos. Otro foco de interés es el filtrado de la información recibida de los sensores.

Sensores: El sistema hace uso de cuatro sensores, un sensor de viento, un sistema de posicionamiento global (GPS), una unidad de medición inercial (IMU) y una cámara.
Actuadores: Hace uso de dos actuadores, un timón y una vela mayor.

Implementan dos formas de dirigir el bote hacia el punto objetivo, donde se asume una trayectoria sin obstáculos. La primera es llamada seguidor de rumbo, donde se calcula de forma matemática la posición de la vela respecto a la dirección del viento, y la dirección del timón en base al ángulo entre el curso actual y el objetivo.
La segunda es llamada seguidor de línea, consiste en trazar una recta entre el punto destino y el de partida, que se utiliza como un vector de atracción, por lo que el bote ajusta su curso para ir lo más cercano a esa línea. La estrategia del seguimiento de línea es un balance entre permanecer cerca de la línea y de seguir el rumbo de esta.

2.2.3. Reinforcement Sailing

En el trabajo [20] se aplican distintas técnicas de aprendizaje por refuerzo para el control de un velero, el cual tiene como objetivo navegar en un rumbo dado con un viento constante. Este objetivo se diferencia con el presentado en este proyecto, el cual busca navegar hacia un punto específico con condiciones ambientales cambiantes.

Se utiliza la velocidad direccional para la función de recompensa. Las acciones son la posición de la vela definiendo el ángulo de esta, la cual puede tomar un valor entre $\frac{-\pi}{4}$ y $\frac{\pi}{4}$, y el control del timón, el ángulo de este puede tomar un valor entre $\frac{-\pi}{2}$ y $\frac{\pi}{2}$. El estado es representado por la velocidad del velero, el viento relativo y el error direccional.

Sensores: No se especifican explícitamente, pero es posible deducir que al menos se cuenta con un velocímetro, con un compás o una brújula para obtener el rumbo y con un sensor de velocidad y dirección del viento.
Actuadores: Se utilizan dos actuadores, un timón y una vela.

Para dirigir y controlar al velero se implementan las siguientes técnicas.

- Controlador codificado a mano
- Controlador utilizando método discreto
- Controlador utilizando método discreto adaptativo
- Aprendizaje por refuerzo en estado continuo

La última técnica mencionada, de aprendizaje por refuerzo en estado continuo, es la que presenta mayor punto de contacto con este proyecto debido a que utiliza aprendizaje por refuerzo utilizando redes neuronales. Este trabajo plantea como una de las conclusiones que al utilizar el método discreto y el método discreto adaptativo no se obtienen buenos resultados debido a la discontinuidad en las acciones, generando que el velero no pueda navegar en un curso predefinido. Por otro lado, el método de aprendizaje por refuerzo obtuvo buenos resultados, algunos incluso mejores que los obtenidos por el controlador a mano, ya que lograba navegar de forma más directa al objetivo.

Capítulo 3

Marco teórico

Para el desarrollo del controlador se hace uso de distintos algoritmos de aprendizaje por refuerzo y aprendizaje por imitación, por lo que resulta relevante entender la teoría en la que estos se basan. En este capítulo se realiza una breve introducción sobre el aprendizaje por refuerzo y los algoritmos que se utilizan, de igual manera, para el aprendizaje por imitación. Luego, se presenta información acerca de ROS, una herramienta utilizada para el desarrollo del controlador del velero.

3.1. Aprendizaje por Refuerzo

El aprendizaje por refuerzo es un área del aprendizaje automático [25], es uno de los tres paradigmas básicos del aprendizaje automático, junto con el aprendizaje supervisado y el aprendizaje no supervisado. A grandes rasgos, aprendizaje por refuerzo es el estudio de agentes y como estos aprenden a través de prueba y error. El agente es recompensado o castigado de acuerdo al impacto en el entorno de las acciones que toma, lo que hace más probable que repita o no ese comportamiento en el futuro. Para poder proceder con una descripción más detallada es necesario comprender terminología clave que forma parte del aprendizaje por refuerzo, la cual es descrita a continuación.

Estado y espacio de observación: Estado se le denomina a la descripción completa del estado del mundo en el cual vive el agente, no se omite información. Una observación es una descripción parcial de un estado, la cual puede omitir información. La forma más común de representar los estados y las observaciones es a través de vectores o matrices de valores reales.

Espacio de acción: Es el conjunto de acciones válidas, las cuales varían según el entorno y el agente. Las acciones pueden ser discretas o continuas, en las primeras existe una cantidad finita de movimientos disponibles para el agente, en las segundas las posibles acciones, a priori, son infinitas.
Esta distinción tiene importantes consecuencias, ya que algunas familias de algoritmos solo pueden ser aplicadas en algunos casos y en otros no es posible hacerlo de forma directa.

Política: La política es la estrategia que utiliza el agente para tratar de conseguir el objetivo planteado, es la regla por la cual se decide qué acciones tomar en cada estado, esta puede ser determinista o estocástica. En el aprendizaje por refuerzo se manejan políticas parametrizadas, donde la salida de la política se calcula a través de

funciones que dependen de un conjunto de parámetros, los cuales pueden ajustarse haciendo uso de un algoritmo de optimización.

Trayectorias: Las trayectorias, también denominadas episodios, son una secuencia de estados y acciones en el mundo. Se denomina transición a lo que sucede entre un paso y el siguiente y depende de la última acción realizada, la cual es determinada por la política.

Recompensa: La definición de la función de recompensa es de gran relevancia para el aprendizaje por refuerzo, dado que aquí se busca definir el comportamiento del agente. Depende del estado actual, la acción tomada y el siguiente estado. La recompensa es la retroalimentación que recibe el agente del entorno, la cual representa que tan bien se adecua la última acción a realizar la tarea objetivo.

Retorno: Existen distintos tipos de retorno, uno de ellos es el retorno de horizonte finito sin descuento. Es la suma de las recompensas obtenidas en un conjunto de pasos determinados. Otro tipo de retorno es el retorno de horizonte infinito con descuento. Es la suma de todas las recompensas obtenidas por el agente con un factor de descuento para priorizar las recompensas obtenidas recientemente. Cualquiera sea la opción elegida, el objetivo del aprendizaje por refuerzo es seleccionar la política que optimice el retorno esperado.

Funciones de valor: Esta función indica que es bueno en el largo plazo, a diferencia de la recompensa, la cual indica que es bueno en un sentido inmediato. Es de utilidad conocer el valor del estado, o del par estado-acción, dado que indica la conveniencia a largo plazo de ese estado, para ello se tiene en cuenta los estados que son posibles seguir y las recompensas de ellos. El valor de un estado es el retorno esperado actuando según una política determinada a partir del estado actual. Existen distintos tipos de funciones de valor que determinan la forma de calcular ese retorno.

Exploración y explotación: Estos dos términos refieren a la forma en que se determinan las acciones. Exploración se le denomina a elegirlas de forma aleatoria, por otro lado, explotación refiere a tomar una acción según cuán valiosa es esa acción en ese estado basado en el conocimiento adquirido hasta el momento. Uno de los problemas en el aprendizaje por refuerzo es encontrar un equilibrio entre estas dos técnicas.

Habiendo definido la terminología procedemos a definir la configuración general de un problema de aprendizaje por refuerzo. Esta se define como un proceso de control estocástico de tiempo discreto, donde el agente interactúa con su entorno de la siguiente manera: el agente comienza en un estado arbitrario del entorno, $s_0 \in S$, obteniendo una observación inicial $w_0 \in \Omega$. En cada paso del tiempo t el agente toma una acción en $a_t \in A$, teniendo tres consecuencias: (i) el agente obtiene una recompensa $r_t \in R$, (ii) se genera la transición del estado s_t al $s_{t+1} \in S$, (iii) el agente obtiene una observación $w_{t+1} \in \Omega$. Esta interacción queda reflejada gráficamente en la imagen 3.1

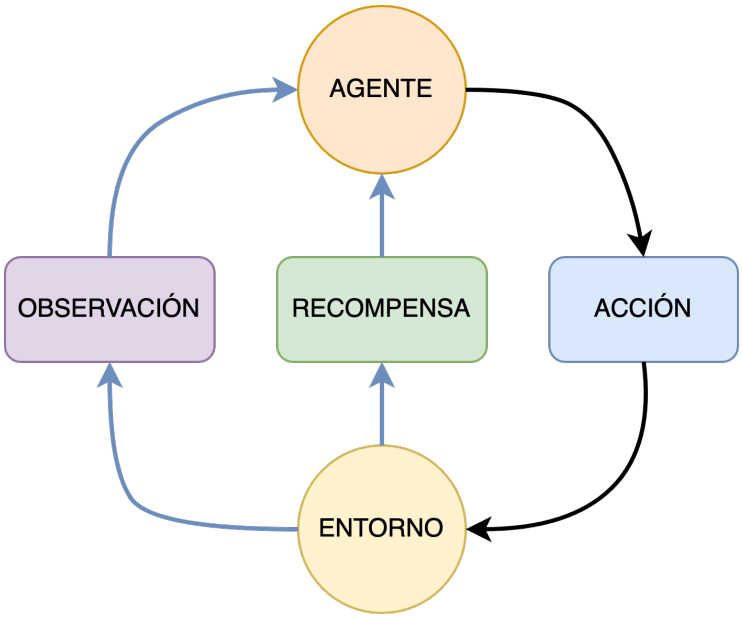

Figura 3.1: Diagrama de interacción entre el agente y el ambiente

Como fue mencionado, el objetivo principal del agente en un problema de aprendizaje por refuerzo es generar una política que maximice la recompensa esperada. Existen distintos caminos para lograr este objetivo, ya que existen diversas taxonomías de los problemas de aprendizaje por refuerzo, a continuación se detallan algunas de las clasificaciones más importantes.

Tabular vs Aproximados: La primera distinción, que es general a cualquier tipo de problema, se basa en la complejidad que presenta. Los problemas más simples donde el espacio de acción y de observación son lo suficientemente chicos para que la función valor-acción pueda ser representada en un arreglo o tabla se les llama tabular, en general se puede encontrar la solución óptima y exacta al problema. En los problemas más complejos, que presentan espacios de observación y acción más grandes, utilizar un arreglo o tabla para mapear cada par estado-acción resulta imposible, no solo computacionalmente sino también debido al tiempo y la cantidad de datos necesarios para completar la tabla. Es por eso que se debe generalizar, es decir, a partir de una muestra limitada de datos, lograr generar una buena política para una muestra mucho más amplia. A esta generalización se le denomina función de aproximación, la cual toma ejemplos de la función deseada e intenta construir una aproximación de la función completa.

Dentro de la política vs Fuera de la política: Esta clasificación también es general para todo problema de aprendizaje por refuerzo, los métodos dentro de la política intentan evaluar y mejorar la política que está siendo utilizada para tomar decisiones, mientras que los métodos fuera de la política buscan evaluar y mejorar una política diferente a la usada para generar los datos. Cuando se utiliza fuera de la política resulta sencillo aprender de trayectorias que no fueron obtenidas por la misma política, al contrario pasa con los métodos dentro de la política donde no resulta ni sencillo ni efectivo.

Sin modelo vs Basado en modelo: Esta es una de las clasificaciones más im-

portantes, refiere a la pregunta de si un agente tiene acceso o aprende un modelo del entorno, donde el modelo del entorno es una función que predice las transiciones de estados y su correspondiente recompensa.
La principal ventaja de tener un modelo es que el agente puede planificar pensando en futuros pasos, cuando se cuenta con esta posibilidad la cantidad de datos necesarios para obtener una buena política es sustancialmente menor. Pero como gran desventaja es que normalmente no se cuenta con un modelo del entorno, por lo que el agente si quiere hacer uso de un modelo debe aprenderlo directamente de la experiencia, esta tarea es extremadamente compleja y muchas veces el sesgo que se introduce al aprender el modelo es utilizado por el agente para obtener mejores resultados, tendiendo a tener muy malos resultados en un entorno real.
Los algoritmos sin modelo son menos eficientes en su aprendizaje, pero tienden a ser más sencillos de implementar y ajustar. Según la categoría en la que se encuentre el algoritmo se determina cómo y qué es lo que se quiere aprender.

Para los algoritmos basados en modelo no existe una separación tan clara de las opciones como si se tiene para los sin modelo. Dado que el enfoque del proyecto se basa en algoritmos sin modelo, se omite la descripción de dichas opciones. Se deja como referencia el trabajo [26] para profundizar en esta temática en particular.
En los algoritmos sin modelo existen dos grandes enfoques detallados a continuación.

Aproximación de política vs Aproximación de valor: Esta división se encuentra dentro de los problemas sin modelo, aquí se determina que se va a aprender para poder realizar la tarea.
Los métodos basados en aproximación de valor buscan aprender la función de valor, para poder saber el valor de los estados o del par acción-estado. Es utilizada para controlar las acciones, primero se aprenden los valores de los pares acción-estado los cuales son utilizados para implementar una política (por ejemplo ε-greedy) con la cual se seleccionan las acciones.
En cambio, los métodos basados en aproximación de política en principio no buscan aprender a calcular los valores de los estados, sino que la política es representada directamente con una función parametrizada con sus propios pesos. La política aproximada se modifica a lo largo del entrenamiento ajustando los pesos de la función. Por lo que aquí se busca aprender una política y tomar acciones en base a esta política y no directamente usando los valores estimados de cada estado.
Si bien las ventajas y desventajas dependen de cada problema, una de las fortalezas de la aproximación de política es que en principio se busca optimizar directamente lo que se quiere lograr, tendiendo a hacerlo más estable y confiable. En contraste, la aproximación de valor optimiza de forma indirecta el rendimiento del agente, esto conlleva muchos tipos de problemas en el aprendizaje por lo que tienden a ser más inestables. Sin embargo, estos últimos cuando funcionan de forma correcta requieren una cantidad sustancialmente menor de datos, ya que pueden reusar las muestras de forma más efectiva que los métodos de aproximación de política. Es importante mencionar que también existen algoritmos híbridos que utilizan ambas técnicas.

3.1.1. Algoritmos de aprendizaje por refuerzo

A continuación se realiza una breve descripción de los algoritmos de aprendizaje por refuerzo considerados para este proyecto, cuatro algoritmos sin modelo de aproximación de política. Dos algoritmos dentro de la política, Proximal Policy Optimization (PPO) y Advantage Actor Critic (A2C), y los restantes dos fuera de la política, Soft Actor Critic (SAC) y Twin Delayed Deep Deterministic Policy Gradients (TD3). Estos

son considerados como parte del estado del arte en aprendizaje por refuerzo para el control continuo, con un buen rendimiento en una gran variedad de tareas [27–30].

Advantage Actor Critic

A2C es una variante sincrónica y determinista del algoritmo Asynchronous Advantage Actor Critic (A3C [31]) con la que se han logrado resultados similares. Tanto A2C como SAC y TD3 se basan en la misma técnica llamada Crítico-Actor, este concepto maneja estructuras separadas para tener la política separada de la función de valor. La estructura de la política se la conoce como actor, ya que es el encargado de seleccionar las acciones, y a la estimación de la función de valor se la conoce como crítico, porque critica las acciones realizadas por el actor. El aprendizaje siempre es dentro de la política, el crítico debe aprender y criticar cualquier política que esté siguiendo actualmente el actor, para hacerlo busca estimar la función de valor y así poder determinar si el resultado de la acción que tomó el actor fue mejor o peor a lo esperado. El resultado de esta comparación es la entrada que recibe el actor para ajustar su política.

A2C se basa en la arquitectura de Crítico-Actor, donde el crítico para determinar que tan buena fue la acción del actor utiliza la función de valor estimada y calcula el valor de la ventaja, es decir, cuánto mejor es tomar una acción específica en comparación con la acción general promedio en el estado dado.

Proximal Policy Optimization

Este algoritmo busca resolver uno de los problemas que se tiene en el aprendizaje por refuerzo, la inestabilidad de la información obtenida durante el entrenamiento debido a que es generada interactuando con el entorno mientras se realiza el propio entrenamiento. Otra dificultad del aprendizaje por refuerzo que se quiere resolver es la hipersensibilidad a la configuración de parámetros.
Por lo tanto, el propósito del desarrollo de Proximal Policy Optimization (PPO) [32] es encontrar un balance entre una fácil implementación, que sea eficiente en términos del aprovechamiento de datos, y que los hiperparámetros resulten fáciles de ajustar. Para ello combina ideas del algoritmo A2C y al igual que TRPO [33] utiliza una región de confianza para mejorar el agente. La idea principal es que la nueva política no difiera mucho con la anterior en cada actualización. Además, para que el algoritmo sea más eficiente con las muestras generadas, se realiza la actualización de la política utilizando múltiples veces un mismo conjunto datos.

Twin Delayed Deep Deterministic Policy Gradients

Twin Delayed Deep Deterministic Policy Gradients (TD3) [34] es un algoritmo fuera de la política. Se busca solucionar un problema común del algoritmo Deep Deterministic Policy Gradient (DDPG) [35]. DDPG es un algoritmo que busca aprender de forma concurrente la función de valor y la política utilizando una arquitectura Crítico-Actor, y como lo indica el nombre es determinista, la política toma acciones específicas y no retorna una distribución de probabilidad. El mayor problema que presenta es la función de valor, en la cual se sobrevaloran las estimaciones. Para poder solucionar esto TD3 se basa en DDPG, manteniendo la arquitectura Crítico-Actor y propone tres modificaciones importantes:

- TD3 aprende dos funciones de valor en vez de una y utiliza el mínimo valor estimado para la actualización de la política.

- Actualiza la política menos frecuentemente que la función de valor, la publicación original recomienda actualizar la política por cada dos actualizaciones de la función de valor.
- Por último, al actualizar la política, TD3 agrega ruido a la acción objetivo, lo que hace que sea menos probable que la política explote acciones con estimaciones de alto valor.

En conjunto, estas tres modificaciones resultan en una mejora sustancial en el rendimiento respecto a DDPG.

Soft Actor Critic

Soft Actor Critic (SAC) [36] es un algoritmo que busca crear un puente entre la política estocástica y los enfoques DDPG, no es un sucesor de TD3 pero incorpora una de las modificaciones donde se utilizan dos funciones de valor para usar el mínimo de estas. Uno de los problemas que presentan muchos de los métodos basados en el enfoque de DDPG es la gran sensibilidad a los hiperparámetros, requiriendo mucho trabajo en el ajuste de ellos para lograr converger. Uno de los problemas que se busca combatir en el algoritmo SAC es la fragilidad en la convergencia.
Una pieza central de SAC es la regularización de la entropía, la política es entrenada para maximizar un equilibrio entre el retorno esperado y la entropía. La entropía se utiliza como una medida de aleatoriedad en la política, esto tiene una gran conexión con el equilibrio de exploración y explotación, donde un aumento en la entropía desencadena en un aumento en la exploración, ayudando a acelerar el proceso en el futuro y también previniendo que la política converja de forma prematura a un óptimo local malo.

3.2. Aprendizaje por imitación

El aprendizaje por imitación busca copiar el comportamiento de un experto para una determinada tarea. El agente es entrenado para ejecutar una tarea a partir de demostraciones, aprendiendo el mapeo entre observaciones y estados. Este tipo de aprendizaje está ganando popularidad, ya que facilita enseñar tareas complejas sin la necesidad de tener un conocimiento experto en ella.

Existen dos enfoques tradicionales para realizar esta tarea, el primero es *Behavioral Cloning*(BC) [37] que aprende la política como un problema de aprendizaje supervisado utilizando pares estado-acción obtenidas de las trayectorias del experto, la segunda es *Inverse Reinforcement Learning* (IRL) [38, 39] donde el objetivo es que el agente aprendiz encuentre una función de recompensa que pueda explicar el comportamiento del experto a partir de los datos de este.
Ambos enfoques tienen algunos problemas que son grandes limitantes a la hora de la aplicación. BC, que es muy simple e intuitivo, solo resulta exitoso si se cuenta con grandes cantidades de datos. IRL por su parte si bien ha tenido buenos desempeños en varias aplicaciones tiene como problema que solamente aprende la función de costo y no que acción tomar, por lo que en cada paso se requiere hacer un bucle interno para decidir qué acción tomar, esto resulta extremadamente costoso cuando se enfrentan entornos grandes y complejos en su dinámica. Otro problema que presenta es que asume que el comportamiento del cual se aprende es óptimo, esto es una suposición muy fuerte que no siempre es cierta.

En el marco de los problemas que presentan las técnicas mencionadas es que surgen

nuevos enfoques para solucionarlos, por lo que a continuación se detallan dos nuevas técnicas de aprendizaje por imitación. Es importante mencionar que ambas hacen uso de un algoritmo de aprendizaje por refuerzo, lo que resulta interesante es la posibilidad de continuar el entrenamiento del algoritmo usado a través de aprendizaje tradicional, dando lugar a combinar ambas técnicas para entrenar un modelo.

3.2.1. Generative Adversarial Imitation Learning

Generative Adversarial Imitation Learning (GAIL) [40] se inspira en una idea similar a IRL. Como el nombre lo sugiere está basado en Generative Adversarial Networks (GANs) [41]. GAIL puede ser definido como un algoritmo de imitación sin modelo.
En una GAN se tienen dos redes neuronales, generador y discriminador, el generador cumple el rol de generar nuevos datos a través de aprender la distribución del conjunto de datos recibido como entrada, el discriminador es el encargado de determinar si un dato es generado por el generador o es parte del conjunto de datos original.
Con la combinación de los conceptos IRL y GAN, GAIL puede aprender a partir de un número chico de trayectorias del experto. Siendo el objetivo principal de GAIL entrenar el generador para presentar un comportamiento similar al del experto, mientras que el discriminador sirve como una función de recompensa que juzga cuánto se parece el generador al experto.
Una propiedad fundamental de aplicar GAN al aprendizaje por refuerzo es que el generador nunca es expuesto al entrenamiento del mundo real, solamente el discriminador tiene contacto con el mundo real. Esto permite minimizar el problema de trasladar la trayectoria del experto al dominio en el cual se encuentre el agente. Por diversas razones, GAIL no es exactamente un IRL, principalmente por el hecho de que GAIL aprende una política y no una función de recompensa.

GAIL ha presentado resultados prometedores solucionando diversas tareas, teniendo mejores resultados que BC, y algunas veces mejor que la de los expertos. Esto debido a que utiliza aprendizaje por refuerzo y no se encuentra limitado a estar siempre lo más cercano al experto. A pesar de los resultados prometedores, este algoritmo es deficiente al intentar aprender una buena política sobre demostraciones de muchos expertos diferentes, asumiendo implícitamente que todas las demostraciones vienen de un solo experto. Algunos autores [42] también mencionan que GAIL falla al generalizar las dinámicas de diferentes entornos y como consecuencia no resulta tan robusto a cambios del entorno.

3.2.2. Adverserial Inverse Reinforcement Learning

Adversarial Inverse Reinforcement Learning (AIRL) [42] propone un algoritmo IRL basado en aprendizaje adversario, el cual posibilita aprender de forma simultánea la función de recompensa y la función de valor. A diferencia de trabajos anteriores como lo es GAIL, el cual aprende solamente la política, aquí se busca recuperar también la función de recompensa como se hace en IRL, teniendo como motivación que la política es menos robusta a transferencias del aprendizaje.
Al igual que GAIL, se utiliza una GAN donde en el generador se entrena una política que busca copiar el comportamiento del experto y un discriminador para clasificar pares estado-acción determinando si era generado por el generador o por el experto. Una de las dos grandes diferencias es que el generador se ajusta utilizando una función de recompensa y no con la salida del discriminador como se realiza en GAIL, la segunda diferencia es la presencia de una función de recompensa estimada, esta se ajusta utilizando la salida del discriminador.

Como resultado de experimentaciones realizadas [42] se concluye que AIRL tiene un mejor rendimiento que algoritmos previos de aprendizaje por imitación e IRL. La comparación contra GAIL muestra que ambas presentan resultados similares cuando los datos del experto y el escenario de entrenamiento emplean el mismo entorno, pero cuando estos entornos cambian AIRL muestra una mejora sensible.

3.3. Robot Operating System

Robot Operating System (ROS) es un meta-sistema operativo de código abierto que proporciona un conjunto de herramientas para la creación de software para robots. ROS brinda la mayoría de los servicios esperables de un sistema operativo, incluyendo la abstracción de hardware y el control de los dispositivos. Está compuesto por una serie de herramientas y paquetes con el objetivo de facilitar el desarrollo de comportamientos complejos y robustos en robots.

ROS se compone de nodos siguiendo una arquitectura de grafos, donde cada uno representa código ejecutable. Los nodos pueden ubicarse en una misma computadora o estar distribuidos entre computadoras y robots. La ventaja de la estructura utilizada es que cada nodo puede encargarse de un único aspecto del sistema. El principal mecanismo de comunicación entre los nodos es a través del envío y recepción de mensajes, estos son transmitidos sobre canales llamados tópicos. Normalmente, un nodo publica mensajes en un tópico y otros nodos se suscriben a este mismo tópico, cada vez que un mensaje es publicado en él, todos los nodos suscritos lo reciben. Existe una segunda forma de comunicación que, a diferencia de los mensajes que son una interacción unidireccional, es bidireccional. Esta comunicación se llama servicios y la interacción nace desde un nodo el cual hace un pedido de información a otro y luego recibe la respuesta.

Gazebo es una herramienta de simulación, que si bien no es parte de ROS, se puede utilizar a través de ROS. Esta herramienta permite generar simulaciones altamente realistas con una gran cantidad de opciones para modificar, por ejemplo, brinda la posibilidad de agregar y configurar sensores. Además permite modificar el entorno de la simulación.

Capítulo 4

Solución propuesta

La solución se compone de dos grandes partes, las cuales son detalladas en profundidad en las secciones a continuación. Dado que el controlador se basa en un modelo de aprendizaje por refuerzo, el primer paso de la solución propuesta consiste en la elección de una biblioteca de aprendizaje por refuerzo y la creación de una interfaz para hacer posible la comunicación entre la biblioteca y el entorno donde se encuentra el velero. La segunda parte refiere a la solución del problema, en primer lugar se define la tarea que el agente debe ejecutar para cumplir los objetivos esperados por el controlador, luego se procede a diseñar la solución.

4.1. Definición y configuración del entorno de aprendizaje

En esta sección se describen los pasos previos necesarios para poder desarrollar una solución, a grandes rasgos, se requiere tener configurado un entorno que permita aplicar aprendizaje por refuerzo al control del velero.

4.1.1. Biblioteca de aprendizaje por refuerzo y aprendizaje por imitación

La elección de la biblioteca tiene un gran impacto en el resto de la solución, por lo que resulta relevante entender los puntos claves sobre los cuales se toma esta decisión.
El primer punto es si la biblioteca ofrece los algoritmos que componen el estado del arte, más específicamente los cuatro algoritmos descritos en el marco teórico: PPO, A2C, TD3 y SAC. Otro punto es que tan buena es la documentación oficial, debido a que una buena documentación puede acelerar el proceso de poner en uso la biblioteca y facilitar la resolución de posibles inconvenientes que se presenten, para esto último también es de relevancia considerar si es utilizada por la comunidad. Un dato no excluyente pero si importante es si provee algoritmos de aprendizaje por imitación, en especial los detallados en el marco teórico: AIRL y GAIL. En caso de no contar con dichos algoritmos, evaluar la complejidad de integrarle una biblioteca externa que si los provea. Como último punto para la evaluación de las bibliotecas se busca entender cuan complejo resulta intercomunicar la biblioteca con entornos propios, definidos de forma manual. Una característica importante es si la biblioteca es de código abierto, sin embargo, este punto no se toma en cuenta en la comparación debido a que se consideran únicamente bibliotecas que lo sean. Se toma esta decisión debido a que la utilización de herramientas de código abierto es un requisito del proyecto y además

por las restricciones y problemas que implican las bibliotecas que no cumplen con esta característica. Uno de los tantos problemas es la imposibilidad de revisar el código que se utiliza limitando el entendimiento y la resolución de posibles problemas.

Teniendo en cuenta las características mencionadas se realiza una búsqueda de las bibliotecas que cumplen con la mayor cantidad de requisitos esperados, esta búsqueda se centra en artículos no académicos dado que no se encuentran trabajos formales sobre esta comparativa. En base a los artículos [43–45] se reduce la elección a tres posibles bibliotecas, Stable Baseline 3 [46], Tensorforce [47] y TF Agents [48]. De estas se decide utilizar Stable Baseline 3, ya que ésta ofrece una mejor documentación por sobre las otras, y si bien no ofrece los algoritmos de imitación mencionados anteriormente, existe una biblioteca de aprendizaje por imitación, llamada Imitation [49], la cual los implementa y es diseñada para ser usada en conjunto con Stable Baseline 3.

4.1.2. Interfaz entre el entorno y el aprendizaje por refuerzo

Una vez definida la biblioteca, es necesario generar una interfaz con el entorno donde se encuentra el velero. En este proyecto el entorno se compone de la simulación realizada por el simulador USVSim, descrito en el estado del arte.

Stable Baseline 3 utiliza como interfaz la estructura propuesta por OpenIA Gym [50] la cual determina la comunicación entre el entorno y el algoritmo de aprendizaje por refuerzo. Esta estructura fue diseñada con el objetivo de facilitar la estandarización de métricas y así poder comparar diferentes soluciones y algoritmos de aprendizaje por refuerzo, facilitando la posibilidad de recrear trabajos realizados.
Esta idea de OpenIA Gym fue adoptada por la gran mayoría de las bibliotecas actuales, por lo que las bibliotecas se crean sobre una interfaz llamada gimnasio que tiene que cumplir una serie de requisitos. La implementación de esta interfaz depende de cada problema y queda en manos del usuario de la biblioteca. Obviando algunos detalles menores, el gimnasio requiere implementar dos métodos y definir dos variables, estos requisitos son descritos a continuación:

- *observation_space*: Se debe definir la estructura de datos que se va a utilizar para representar el espacio de observación, además es necesario definir los posibles valores que puede tomar.
- *action_space*: Refiere a la definición de la estructura de datos que se va a usar para representar el espacio de acción, es necesario definir los posibles valores que puede tomar.
- *reset*: El gimnasio debe implementar este método, el cual se ejecuta al inicio de cada episodio. Este método se debe encargar de todo lo necesario para reiniciar la simulación.
- *step*: Por último se define el método más importante, siendo el responsable de prácticamente toda la interacción entre el algoritmo y el entorno. Se ejecuta cuando se toma una acción, el primer paso es decodificar la acción del espacio de acción a una acción razonable para el entorno, y así poder realizarla, una vez realizada debe obtener los datos del entorno y completar los datos del espacio de observación. Luego, partiendo de la nueva observación, se debe determinar el valor de la recompensa inmediata y si la simulación relacionada a ese episodio terminó. Esta información se retorna junto a la observación y son utilizados por el modelo de aprendizaje por refuerzo para realizar el entrenamiento.

4.2. Definición e implementación del controlador

La creación del controlador se compone en gran parte en la implementación del gimnasio descrito, al definir los métodos y variables se determina qué se quiere aprender y cómo. Previamente a la descripción de los métodos implementados es necesario especificar la tarea que el velero debe realizar y el contexto de esta.

4.2.1. Tarea a realizar y su contexto

Para definir la tarea que el velero quiere realizar es necesario partir del objetivo inicial de este proyecto. Simplificándolo podemos decir que consiste en lograr que un velero autonomo pueda navegar hasta un punto deseado y tomar allí diversas muestras. Aunque no sea un requisito excluyente resulta interesante que el velero logre mantenerse tan cerca como sea posible del punto objetivo a la hora de tomar las muestras para lograr obtenerlas en el punto elegido.

Con este objetivo presente se determina que la tarea del velero es la siguiente: partir de un punto inicial y llegar a un punto objetivo de la forma más rápida posible, luego se espera que se mantenga tan cerca como sea posible de dicho punto hasta que la tarea finalice.

Es necesario definir las variables del entorno que son tomadas en cuenta y modificadas en cada ejecución. Estas variables quedan definidas por la información que es posible obtener y modificar del entorno y por la importancia de estas en la tarea a realizar. En este trabajo las variables del entorno que se definen para cada ejecución son: el punto objetivo, la velocidad y dirección del viento, la velocidad y dirección de la corriente de agua y la velocidad y orientación del velero. Dado que el velero precisa del viento para su propulsión y maniobrabilidad, es que se fija un valor mínimo para la velocidad del viento, el cual se determina en dos metros por segundo. Para su definición se toma como referencia la escala [51] generada por una universidad de Canadá, la cual aplica la escala de fuerzas del viento Beaufort a barcos pequeños y en la que se describe que a partir de los dos metros por segundo se logra una lenta navegación. Por otro lado, se cree importante fijar un valor máximo para la velocidad del viento y de la corriente del agua, las cuales se determinan en cinco con cinco metros por segundo y en cero con cuatro metros por segundo respectivamente. Se toma como referencia los escenarios creados por quienes generaron el simulador utilizado [9] en los cuales se fijan dichos valores para la simulación de un escenario con grandes disturbios ambientales.

4.2.2. Espacio de observación y acción

La correcta elección del espacio de observación es crucial para lograr un aprendizaje exitoso, si el estado no contiene datos relevantes para tomar una buena decisión es probable que no se logre aprender una buena política, en cambio si la cantidad de datos es demasiado grande el aprendizaje puede no converger o requerir demasiado tiempo para hacerlo. Es por esto que el objetivo al definir qué datos integran el espacio de observación es que contenga la información mínima y suficiente del entorno sobre la cual sea posible determinar la mejor acción a realizar. En el caso de este trabajo en el que se utiliza un simulador es importante no perder de foco que toda la información utilizada para generar la observación debe ser posible obtenerla desde un velero real, debido a que el simulador proporciona información que no es posible obtener en un entorno real. La definición del espacio de acción en general es más sencilla, se debe

determinar qué acciones puede tomar el controlador y los posibles valores de esas acciones.

Espacio de observación: Luego de analizar las distintas variables a observar del entorno se decide incluir las siguientes:

- Distancia al objetivo: Se calcula la distancia entre el punto donde se encuentra el velero y el punto objetivo.
- Error direccional: Se calcula el ángulo que existe entre la orientación del velero y el lugar a donde debería apuntar para orientarse directamente al punto objetivo.
- Dirección de la velocidad del velero: Es la dirección hacia donde se dirige el velero respecto a la orientación de este.
- Velocidad del velero: Refiere a la velocidad con la que navega el velero.
- Dirección del viento: Dirección del viento respecto a la orientación del velero.
- Velocidad del viento: Es el módulo del vector de velocidad del viento.
- Dirección de la corriente: Dirección de la corriente respecto a la orientación del velero.
- Velocidad de la corriente: Es el módulo del vector de velocidad de la corriente.

Es importante remarcar que todos los datos que conforman el espacio de observación son posibles de obtener con sensores reales que un velero podría tener.

Espacio de acción: Este espacio cuenta con dos variables: el ángulo del timón y el ángulo de la vela.

Por la importancia que supone normalizar los datos al utilizar redes neuronales [52] y por recomendación de la biblioteca de aprendizaje por refuerzo seleccionada es que los espacios definidos son normalizados para el aprendizaje. El espacio de observación es normalizado entre [0,1] y el espacio de acción entre [-1,1].

4.2.3. Ejecución de acción y obtención de observación

La obtención de los datos para la observación queda condicionada por el entorno donde el agente se encuentre. En este proyecto todos los datos son obtenidos a través de ROS, algunos de ellos son utilizados directamente, sin embargo, para otros es necesario aplicar algunos cálculos. En particular, se quiere expresar el viento, la corriente y la velocidad del velero en el marco de referencia de la orientación del velero. Para ello es necesario realizar una rotación de los vectores obtenidos. En un principio se toma en consideración la utilización del paquete tf de ROS, de gran uso en el campo de la robótica, pero debido a la alta complejidad que implicaba su incorporación al proyecto sumado al mínimo uso que se le quería dar es que se decide utilizar la biblioteca SciPy [53] para realizar dicha rotación.

Por otro lado, se debe ejecutar la acción que se recibe, por como se define el espacio de acción, el mapeo hacia la posición de los actuadores es directo. Lo único que resta realizar es comunicar esa acción al simulador, pero luego de esto surge un problema complejo, este es la definición del tiempo que se debe esperar entre que se realiza la acción y la obtención de la observación. Si se espera poco tiempo puede suceder que los actuadores estén todavía moviéndose a la posición correcta o que aún no haya generado un impacto en el entorno, en cambio si se espera mucho se pierde información del entorno que puede ser relevante para lograr controlar el velero adecuadamente. Esto

impacta de gran manera en el aprendizaje, en especial para el control de un velero, donde las consecuencias de una acción y sus tiempos son muy variables y dependen fuertemente de las condiciones del entorno. Por ejemplo, existen posiciones de la vela relativas a la dirección del viento que resultan muy costosas de realizar, en cuanto a tiempo, o directamente no es posible realizar ese movimiento. Existe un trabajo [54] en el cual se identifica este problema y se propone una solución novedosa, en este proyecto no se utiliza esta técnica por la complejidad que adiciona al proyecto y se plantea como trabajo a futuro. Se decide utilizar un tiempo de espera arbitrario determinado a través de experimentación. La prueba consiste en tomar mil acciones aleatorias en un escenario con bajos disturbios ambientales para determinar la posición de la vela y el timón, se calcula el tiempo que le toma al velero mover los actuadores a una posición cercana a la objetivo y se calcula el tiempo promedio de esas mil acciones, sin considerar los tiempos de espera mayores a dos segundos, ya que se asume que son posiciones inalcanzables para el velero. El promedio que se obtuvo es cercano a 0.5 segundos de espera, por lo cual es el valor que se toma como tiempo de espera en la experimentación de este proyecto.

4.2.4. Función de recompensa

La función de recompensa es donde se determina el comportamiento que se quiere alcanzar, a través del castigo o de la penalización de cada acción se busca enseñar al agente a lograr una tarea específica. El valor de recompensa puede ser cualquier número, cuanto más alto mejor y cuanto más bajo peor. Este valor es calculado en base a la observación resultante de la acción previa, se trata de reflejar que tan buena es la acción tomada respecto al objetivo deseado. El valor se calcula de forma totalmente arbitraria y condicionada al problema a resolver, pero existen algunas guías para su definición.
Existen dos extremos a la hora de diseñar la recompensa, mantener lo más simple posible el cálculo o complejizarlo. Por lo general en el primero se da un valor positivo si se realiza una acción buena y uno negativo si la acción es mala, por ejemplo [-1, 1]. En la segunda generalmente se utilizan rangos continuos de valor según alguna función específica, por ejemplo retornar la velocidad a la que avanza el robot. La recompensa más simple tiene como beneficio que no se necesita especial conocimiento de la tarea a realizar y generalmente asegura un buen comportamiento. Como desventaja puede ser más lento el aprendizaje, ya que la información de la recompensa es relativamente baja. Por el otro lado, una recompensa compleja requiere tener un grado de conocimiento alto de la tarea a realizar y tiene como ventaja que es posible acelerar en gran medida el aprendizaje. Pero si la información que trasmite la función de recompensa no es correcta, el algoritmo puede no converger o converger a un óptimo local malo.

Se requiere de mucha experiencia y una profunda investigación para comprender la dinámica de la navegación y lograr formular una función de recompensa compleja que represente el objetivo deseado. Por esta razón y teniendo como referencia lo detallado, se decide utilizar funciones de refuerzo relativamente sencillas. Se realizan experimentos con el fin de evaluar distintas funciones de recompensa. El detalle y análisis de estos experimentos realizados son expuestos en el siguiente capítulo. Aquí se explicita la función de recompensa elegida por sobre las restantes.

Como la tarea definida es lograr que el velero vaya hacia un punto y se mantenga en él, la función de refuerzo elegida, a grandes rasgos, busca premiar cuando se acerca y castigar cuando se aleja. Teniendo eso presente se definen tres escenarios con sus recompensas asociadas:

- *Recompensa* = 1: Cuando la distancia al objetivo es menor a la observación previa.
- *Recompensa* = 0: Cuando la distancia al objetivo es igual a la observación previa.
- *Recompensa* = −1: Cuando la distancia al objetivo es mayor a la observación previa.

Se considera que el velero avanza o se aleja del punto objetivo si la diferencia entre la distancia anterior y la actual es mayor a un épsilon determinado. Si esta diferencia es inferior, se considera que el velero se encuentra a la misma distancia que en la observación anterior.

4.2.5. Fin de episodio

En un principio se consideró definir el episodio de horizonte variable donde la ejecución se termina debido a alguna condición [55], para esto se determinó que el episodio finalizara una vez que el velero llegara al objetivo. Luego se decidió realizar una modificación debido a que el objetivo no es solo alcanzar el punto objetivo, sino permanecer en él. Es por esto que la ejecución de cada episodio se termina luego de una cantidad arbitraria de pasos, cantidad que se determina para que el velero sea capaz de cumplir con el objetivo. A esto se le denomina episodio de horizonte fijo [55], donde el episodio termina luego de una cantidad fija de pasos.

La determinación de la cantidad de pasos debe ser por lo menos suficiente para que realizando una navegación razonablemente correcta sea posible alcanzar el punto objetivo. Para determinar esto se calcula cuántos pasos le toma al controlador manual navegar en rumbo de ceñida para alcanzar un punto que se encuentra contra el viento y permanecer en él al menos unos segundos, la navegación en ceñida es uno de los rumbos más lentos para navegar por eso la elección de este. Una vez determinada la cantidad de pasos que le toma llegar a dicho punto, se considera el doble de los que le tomó para dar tiempo a permanecer en el punto y tener la seguridad de que la cantidad de pasos es suficiente.

4.3. Arquitectura de la solución

En esta sección se detalla la arquitectura de la solución propuesta, identificando los componentes y sus relaciones. La arquitectura se compone principalmente de dos grandes componentes: el controlador adaptativo construido y el simulador elegido [9].

El componente controlador adaptativo cuenta con el modulo de entrenamiento en el cual se determina entre otras configuraciones el tipo de entrenamiento a realizar: si se utiliza aprendizaje por imitación o no y se determina el algoritmo de aprendizaje por refuerzo. También es posible definir si se desea utilizar el controlador manual. Este modulo se comunica con un entorno en el cual se implementan las funciones requeridas por la interfaz del gimnasio descrita en la sección 4.1.2. Como puede verse en la figura 4.1, la comunicación se realiza a través de las funciones 'step' y 'reset' descritas anteriormente.
El gimnasio al comienzo de cada episodio es responsable de configurar las variables que determinan el entorno del episodio. Para luego en cada paso realizar una acción y retornar la información requerida por la función 'step', esto incluye obtener la información del entorno, procesarla y retornar una observación, retornar la recompensa de la

acción tomada y si el episodio finalizó. Para esto se comunica con la interfaz del simulador, la cual proporciona la información necesaria. La comunicación con dicha interfaz se realiza principalmente a través de las funciones: 'set_up_env', 'set_action", "get_obs'. Como el nombre lo sugiere estas funciones responden respectivamente a: configurar las condiciones iniciales del entorno antes de comenzar un episodio, configurar el ángulo del timón y la vela en cada paso, obtener información de las condiciones del entorno en cada paso.
Para poder proporcionar la información requerida, la interfaz se debe comunicar con el simulador. Esta comunicación se realiza a través de tópicos y servicios utilizando ROS. En particular, para la obtención de información acerca del velero como la posición y la velocidad de este se utiliza el tópico 'sailboat/state'. La determinación de la corriente marina en cada episodio y la obtención de esta información en cada paso se realiza a través del tópico 'gazebo/current', y para el viento se usa el servicio 'windCurrent'. La transmisión del ángulo en el que deben posicionarse los actuadores en cada paso se realiza a través del tópico 'sailboat/joint_setpoint'. Para configurar el viento al comienzo de cada episodio se usa un servicio provisto para ello. Para configurar la orientación y posición inicial del velero en cada episodio se usa el servicio 'gazebo/set_model_state'. No se entra en detalle acerca de la comunicación dentro del simulador debido a que es descrita en el capítulo 2.

Por otro lado, se desea guardar información del entrenamiento, para ello se crea un tópico denominado 'sailboat/metrics' en el cual en cada paso se publica información de: la observación, la recompensa, las acciones y el episodio.

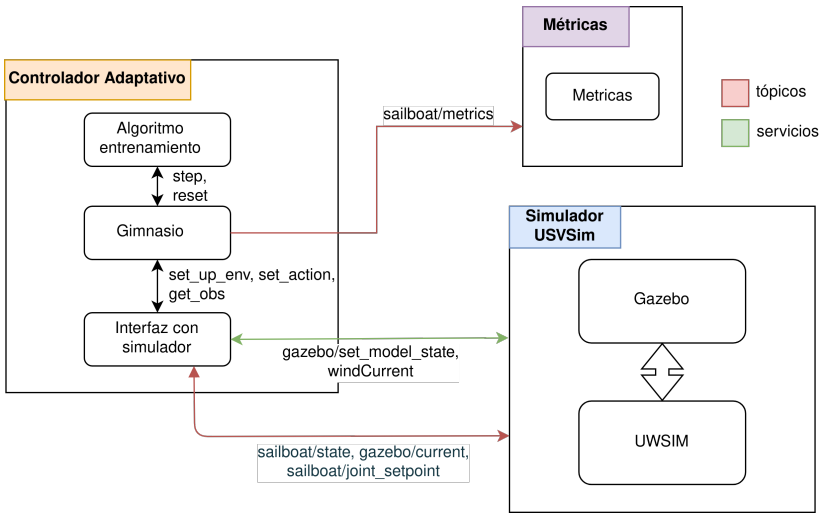

Figura 4.1: Arquitectura de la solución propuesta

Capítulo 5

Experimentación

Una vez diseñada e implementada la solución, se realizan pruebas para comparar y analizar los distintos métodos y enfoques. Debido a la naturaleza propia de la técnica de aprendizaje por refuerzo, en la que en cada paso se interactúa con el entorno para obtener un nuevo dato, el aprendizaje conlleva un tiempo considerable. En el contexto de este proyecto el problema planteado es menor, ya que el entrenamiento se desarrolla en un simulador, donde ejecutando una serie de modificaciones y ajustes en el mismo se logra obtener una simulación que transcurre a mayor velocidad que el tiempo real, aproximadamente diez veces más rápido en la simulación respecto al tiempo real. Sin embargo, a pesar de esta mejora, los tiempos para lograr una convergencia en el aprendizaje resultan extremadamente largos pudiendo llevar varios días. A causa de este problema resulta muy costoso probar todas las ideas y enfoques en un experimento final, por lo que se opta por hacer pruebas pre-experimentales donde se generan escenarios simplificados para validar y descartar ideas antes de pasar a la experimentación final.

Este capítulo queda separado en cinco secciones, en la primera se detallan las métricas, la siguiente se corresponde con los escenarios, luego se dedica una sección a la experimentación con el controlador manual, en la sección cuatro se plantean las pruebas pre-experimentales y la última se dedica a las pruebas experimentales.

La experimentación se realizó en una Laptop MSI PS63 Modern 8RC con las siguientes especificaciones:

- Sistema Operativo: Ubuntu 18.04
- Versión de ROS: Melodic
- Memoria RAM: 16GB DDR4-2666
- CPU: Intel Core i7-8565U
- GPU: Nvidia GeForce GTX 1050 Ti Max-Q

5.1. Métricas

En esta sección se definen tres métricas para comparar los resultados de los controladores, dependiendo del tipo de prueba que se realice se elige un subconjunto de estas o la totalidad. Esta diferencia es detallada en la descripción de cada prueba.

La primera métrica es la recompensa promedio. Luego, teniendo en cuenta el objetivo del proyecto, es que se definen dos métricas adicionales: la primera es el promedio de la distancia acumulada al objetivo y la segunda es la cantidad de episodios en los que el velero llegó al objetivo.

Recompensa promedio: Esta métrica se calcula como el promedio de la recompensa obtenida por el agente en cada paso. Esta métrica es la más tradicional del aprendizaje por refuerzo y no refiere de forma directa a la tarea a realizar sino a la función de refuerzo definida. Valores altos de esta métrica indican que tan bien logra comportarse el agente respecto a la función de refuerzo definida. Es importante esta distinción porque puede que la definición de la función de recompensa no refleje correctamente la tarea a realizar.
Sea n la cantidad de episodios y r_i la recompensa final del episodio i la métrica utilizada es la siguiente:

$$recompensa\_promedio = (\sum_{i=1}^{n} r_i)/n$$

Cantidad de episodios: La segunda métrica es la cantidad de episodios en los que el velero, en algún momento de todo el episodio, se encuentra a una distancia menor a un umbral determinado del punto objetivo. El umbral elegido es dos veces y media la longitud del velero. Aquí se busca entender si el velero fue capaz de llegar al objetivo por lo menos una vez, sin importar si antes o después se alejó mucho del mismo.

Distancia acumulada promedio: Para cada episodio se suma la distancia euclidiana desde el centro del velero al punto objetivo en cada uno de sus pasos. Luego se realiza el promedio de todos los episodios. Esta métrica es expresada en metros. A continuación se define dicha métrica con notación matemática.
Sea d_{ij} la distancia entre el velero y el punto objetivo en el paso i del episodio j, n la cantidad de episodios en los que se evalúa y m la cantidad de pasos de cada episodio.

$$distancia\_acumulada\_promedio = (\sum_{j=1}^{n} \sum_{i=1}^{m} d_{ij})/n$$

Esta métrica busca entender qué tan cercano al punto objetivo a lo largo de todo el episodio se encuentra el velero.

5.2. Escenarios

Al igual que en la sección anterior aquí se definen distintos conjuntos de escenarios que se utilizan en las distintas pruebas a realizar. Para los escenarios existen dos formas de generación: manual y automática. Cada escenario comienza con el velero en una posición inicial fija y queda definido por la tupla (Punto objetivo, Orientación inicial, Corriente, Viento).

5.2.1. Generación automática

Este tipo de generación se utiliza para los entrenamientos largos de al menos un millón de pasos y para las evaluaciones cuantitativas. Los escenarios se generan de forma aleatoria, a continuación se explica como se genera cada elemento de la tupla y que restricciones se utilizan.

- Punto objetivo: el punto objetivo se determina con la dupla (x, y) que representa las coordenadas en el marco global. Para seleccionar los valores de esta dupla se utiliza el punto de inicio del velero como centro de referencia, del cual se definen un radio mínimo y máximo determinando un anillo, con el objetivo de variar la distancia pero que todos los puntos sean alcanzables en la cantidad de pasos que determinan al episodio. Los valores x e y se seleccionan de forma aleatoria con la restricción de que pertenezcan al anillo. El radio mínimo y máximo se definen teniendo como referencia la distancia de los puntos elegidos en las trayectorias del controlador manual y considerando que la diferencia entre el máximo y mínimo pueda ser recorrida en una cantidad pequeña de pasos, de forma que no afecte el cumplimiento de la tarea.
- Orientación inicial: este valor es un número perteneciente al rango $[-\pi, \pi]$, representando la orientación en radianes del velero al inicio del episodio. En este caso se selecciona un valor aleatorio del rango válido.
- Corriente: la corriente del agua se expresa como un vector (x, y), para generarlo se elige una velocidad de forma aleatoria entre los valores de velocidad de corriente máxima y mínima definidos en el capítulo previo, y un ángulo entre el rango $[-\pi, \pi]$. Luego la velocidad se descompone en los valores x e y.
- Viento: Este valor es análogo al de corriente, pero utilizando los valores de velocidad máxima y mínima de la velocidad del viento.

Como ejemplo de los puntos objetivos generados aleatoriamente se presenta la figura 5.1 en los cuales el velero parte desde el centro del anillo.

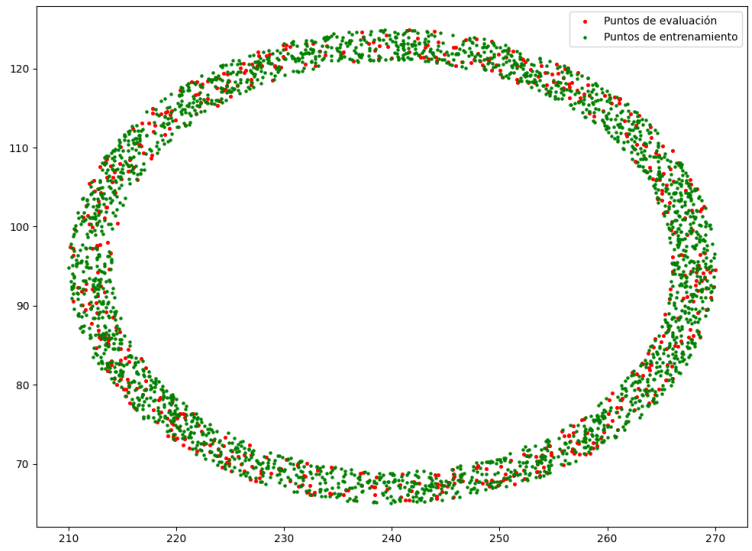

Figura 5.1: Ejemplo de 3000 puntos objetivos generados aleatoriamente, en color verde se representan los utilizados en el entrenamiento y en color rojo los de evaluación.

5.2.2. Generación manual

La generación de estos escenarios se realiza utilizando la misma tupla con las mismas restricciones que en los escenarios generados de forma automática, con la diferencia de que los valores se eligen de forma manual.

Escenarios para las trayectorias del controlador manual

Se busca generar trayectorias de escenarios diversos pero donde el controlador manual tenga buen desempeño. Teniendo en cuenta lo anterior, se generan dieciocho escenarios tratando de cubrir todos los rumbos de navegación, exceptuando la zona muerta y el rumbo en ceñida. Los rumbos utilizados se pueden observar en la figura 5.2.
Se utilizan dos vientos moderados, la mitad de los escenarios utilizan un viento con una velocidad de dos metros por segundo y la otra mitad utilizan un viento de tres metros por segundo. Se determinan dichos valores debido a que el controlador manual es capaz de navegar correctamente en esas condiciones. En todos los escenarios se utiliza una corriente nula por el hecho de que el controlador manual no toma en cuenta esta información y no logra llegar al punto objetivo si existe una corriente considerable.

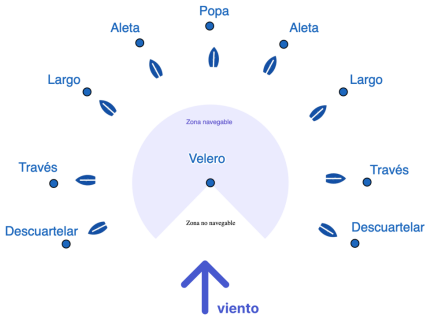

Figura 5.2: Rumbos elegidos para las trayectorias del controlador manual

Escenarios para las pruebas pre-experimentales de imitación

Debido a que la cantidad de pasos de entrenamiento es pequeña, la generación de escenarios de estas pruebas se realiza de forma manual, de forma de lograr simplificar para posibilitar un aprendizaje en menor cantidad de tiempo.
En esta generación de escenarios se toma como referencia los puntos elegidos para las trayectorias del controlador manual, añadiendo algunos puntos entre ellos con el fin de obtener más variedad, en particular se añade el rumbo de ceñida. Se establecen quince puntos objetivos. Por otro lado, se plantean tres opciones de corriente marítima: sin corriente, 0.1 m/s de corriente sobre el eje x y 0.1 m/s de corriente sobre el eje y. Se agrega la corriente marítima con el objetivo de incorporar a los escenarios de entrenamiento una variable que el controlador manual no tome en cuenta, pero sin complejizar demasiado. Para determinar el escenario se toma uno de los quince puntos y una de las tres corrientes establecidas, la velocidad del viento y la orientación inicial del velero se mantiene constante en todos ellos.
Los escenarios propuestos para la evaluación de estas pruebas se forman con cuatro puntos, los cuales representan el rumbo de aleta y el rumbo de través para cada uno de los lados. Además, se plantean dos posibilidades para la corriente: sin corriente y con 0.05 m/s en ambos ejes. De esta forma se generan ocho escenarios diferentes.

En la figura 5.3 se pueden observar los distintos rumbos utilizados en el entrenamiento y en la evaluación.

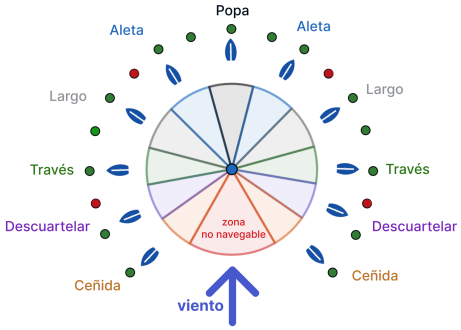

Figura 5.3: Rumbos elegidos en las pruebas pre-experimentales de aprendizaje con imitación. En color verde se presentan los puntos objetivos utilizados en el entrenamiento y en color rojo los utilizados para la evaluación.

Escenarios para las pruebas finales cualitativas

Estos escenarios son generados con el fin de observar cómo navega el velero en distintas situaciones. Para ello se proponen seis puntos objetivos, tres de ellos en los que no debería ser muy complicado navegar, estos son: a favor del viento y rumbo de largo hacia ambos lados. Y los otros tres suponen un desafío mayor, estos son: contra el viento y rumbo de ceñida hacia ambos lados. En la figura 5.4 se pueden observar los rumbos mencionados.
A su vez se plantean dos situaciones climáticas bastante distintas, una de ellas representando un momento tranquilo para navegar y la otra representando un momento con altos disturbios. Para el primer escenario se determina una corriente casi nula, de 0.05 metros por segundo y un viento moderado, la velocidad de este es de tres metros por segundo. Para el segundo escenario se determina una corriente importante de cero con dos metros por segundo y un viento fuerte de cinco metros por segundo.
Por cada punto objetivo se generan dos escenarios, uno con cada una de las situaciones climáticas mencionadas. En total se determinan doce escenarios.

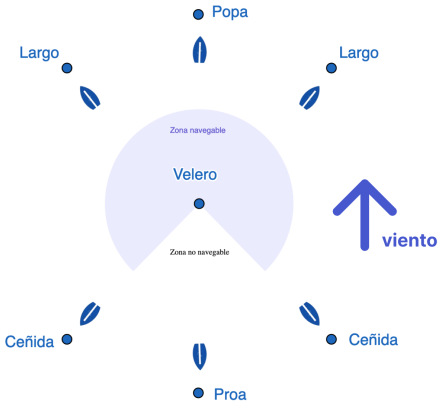

Figura 5.4: Rumbos elegidos para la evaluación cualitativa

5.3. Trayectorias del controlador manual

Para las técnicas de aprendizaje por imitación GAIL y AIRL, es necesario disponer de datos de trayectorias realizadas por el controlador manual. Los datos de cada trayectoria contienen por cada paso la acción tomada, la observación y la recompensa obtenida. La biblioteca por imitación elegida provee las funcionalidades para recolectar esta información, por lo que solamente se debe determinar sobre qué escenarios se van a obtener las trayectorias. A la hora de definir los escenarios es importante que las trayectorias reflejen una secuencia de buenas acciones, ya que a partir de estas se intenta aprender. Con base en esto se definen los escenarios para las trayectorias del controlador manual descritos en la sección 5.2.2.

5.4. Pruebas pre-experimentales

Como se menciona anteriormente, la motivación de estas pruebas pre-experimentales es validar ideas con el objetivo de tener menor cantidad de combinaciones para las pruebas finales. Estas tienen dos grandes objetivos, el primero es definir la función de recompensa y el segundo, relacionado al uso de aprendizaje por imitación, es comprobar si tanto el algoritmo GAIL como AIRL pueden ser beneficiosos para el aprendizaje en este proyecto. Sumado a estos objetivos se describen otros experimentos realizados como búsquedas de caminos alternativos o posibles modificaciones a la idea final de la solución propuesta.
Si bien ambas pruebas son pre-experimentales, se encuentran ciertas diferencias entre ellas respecto a la cantidad de tiempo que se les dedica y a los escenarios empleados para su entrenamiento y evaluación. Las pruebas sobre la función de recompensa se realiza en un escenario más cercano al final, ya que la definición de esta tiene impacto en las siguientes pruebas, mientras que las pruebas de aprendizaje por imitación son de un carácter más exploratorio.

5.4.1. Pruebas en uso de aprendizaje por imitación

Estas pruebas tienen como objetivo validar si presenta mejoras incorporar el aprendizaje por imitación en las pruebas finales.

Para esta comparación se definen seis agentes, para cada uno se prueba con los cuatro algoritmos de aprendizaje por refuerzo previamente definidos. Las pruebas consisten en entrenar 400.000 pasos y evaluar los modelos en dieciséis episodios. La cantidad de pasos de entrenamiento se determina debido a que eran suficientes para que el controlador lograra alcanzar algunos puntos y no se deseaba realizar pruebas que llevaran mucho tiempo en esta etapa. Para entrenar y evaluar se utilizan los escenarios para las pruebas pre-experimentales de imitación descritos en la sección 5.2.2, y como métricas se utilizan la distancia acumulada promedio y la cantidad de episodios. A continuación se detallan los cinco agentes definidos:

- RL: Este agente realiza la totalidad del entrenamiento solo con aprendizaje por refuerzo.
- GAIL: Este agente realiza la totalidad del entrenamiento solo con aprendizaje por imitación utilizando la técnica GAIL.
- AIRL: Este agente realiza la totalidad del entrenamiento solo con aprendizaje por imitación utilizando la técnica AIRL.

- GAIL + RL: Este agente realiza los primeros 200.000 pasos con aprendizaje por imitación utilizando la técnica GAIL y luego continúa los restantes 200.000 pasos usando aprendizaje por refuerzo.
- AIRL + RL: Este agente realiza los primeros 200.000 pasos con aprendizaje por imitación utilizando la técnica AIRL y luego continúa los restantes 200.000 pasos usando aprendizaje por refuerzo.

Todos los agentes, a excepción de AIRL y AIRL + RL, se prueban con los cuatro algoritmos descritos en el marco teórico (SAC, PPO, A2C, TD3). Los agentes AIRL y AIRL + RL se prueban utilizando A2C y PPO, ya que el algoritmo AIRL no es compatible con SAC y TD3.

A continuación se plantean, en forma de tabla, los resultados obtenidos en las pruebas realizadas para las dos métricas descritas anteriormente. Para ello se utiliza únicamente el valor final de la métrica distancia acumulada promedio. Para ver el detalle de esta métrica y observar la evolución de la misma a lo largo de los pasos, se pueden consultar las gráficas que se encuentran en el anexo 6.4.

Agente		Métricas	
Algoritmo	Entrenamiento	Distancia acumulada promedio [m]	Cantidad de episodios
A2C	RL	8504.75	0
	GAIL	8504.11	0
	AIRL	8653.42	0
	GAIL + RL	8503.48	0
	AIRL + RL	**8502.56**	0
PPO	RL	8515.20	0
	GAIL	**3673.72**	**13**
	AIRL	8595.47	0
	GAIL + RL	5113.34	7
	AIRL + RL	8104.93	0
SAC	RL	2231.63	**16**
	GAIL	**2114.26**	**16**
	GAIL + RL	3796.94	12
TD3	RL	**3952.24**	**12**
	GAIL	8504.59	0
	GAIL + RL	5706.33	4

Tabla 5.1: Resultados de las pruebas de aprendizaje por imitación.

En los resultados obtenidos se puede observar que no hay una clara ventaja en el rendimiento del entrenamiento al utilizar algoritmos de aprendizaje por imitación. Los resultados dependen fuertemente sobre qué algoritmo se analicen. En particular con el algoritmo PPO se obtienen mejores resultados al utilizar el agente GAIL + RL e incluso mejor aún al utilizar únicamente el agente GAIL, en cambio, al analizar TD3 hay una clara superioridad en utilizar RL respecto a las técnicas con imitación. Con SAC si bien los resultados obtenidos con el agente RL son mejores, se obtuvieron resultados muy similares con los agentes que utilizan imitación, por último A2C no se toma en cuenta en el análisis por los resultados extremadamente negativos en todos los agentes.

Esta paridad da pie a seguir investigando en las pruebas finales el beneficio de utilizar

aprendizaje por imitación, sin embargo al observar los resultados obtenidos con los algoritmos de aprendizaje por imitación AIRL y GAIL en sus distintas combinaciones, se puede determinar la superioridad de GAIL por sobre AIRL. Por esta razón y por el hecho de que con dos de los algoritmos no es posible utilizar el algoritmo AIRL se decide solo utilizar GAIL y GAIL + RL en las pruebas finales.

5.4.2. Pruebas sobre función de recompensa

Estas pruebas tienen como objetivo definir la función de recompensa a utilizar, para esto se proponen tres funciones diferentes. Todas tienen en común la siguiente idea: cuando el velero se aleja del objetivo se retorna menos uno, cuando se acerca al objetivo se retorna uno y en caso de que no se haya alejado o acercado se retorna cero. La variación se encuentra en el valor retornado una vez que el velero llega al objetivo. Se le denomina llegar al objetivo a encontrarse dentro de un radio determinado con centro en el punto al que se quiere llegar.
En un principio se definió otra función de refuerzo, la cual no forma parte de estas pruebas, donde se utilizó como retorno la métrica *Velocity Made good to Course* (VMC) [56], la cual es la velocidad ganada con respecto a un rumbo o una marca. La VMC se calcula multiplicando la velocidad del velero por el coseno del ángulo entre la dirección del punto objetivo y el rumbo del velero. Esta función de recompensa es descartada no solo por los malos resultados, sino que no se adapta al objetivo de este proyecto debido a que la tarea no es solamente llegar a un punto objetivo sino mantenerse en él.

Para esta prueba se generan 1000 escenarios diferentes, 800 para el entrenamiento y 200 para la evaluación utilizando el método automático, detallado en la sección 5.2.1. Como métricas se utilizan la distancia acumulada promedio y la cantidad de episodios.

Las tres funciones a probar retornan distintos valores cuando el velero se encuentra en el objetivo:

- Función 1: Esta función retorna diez siempre que se encuentre en el objetivo.
- Función 2: Esta función retorna uno siempre que se encuentre en el objetivo.
- Función 3: Esta función mantiene el mismo cálculo que se realiza cuando se está fuera del objetivo, retornando uno si se acerca, menos uno si se aleja y cero si no varía.

Cada una de las funciones se prueba para los cuatro algoritmos descritos en el marco teórico (SAC, PPO, A2C, TD3). Las pruebas consisten en realizar un entrenamiento de 1.000.000 de pasos y luego evaluar el modelo en 200 episodios, donde cada episodio consta de 750 pasos.

Al igual que en la sección anterior se muestran los resultados obtenidos en forma de tabla utilizando las métricas mencionadas anteriormente. Si se desea observar la evolución de la métrica distancia acumulada promedio se pueden consultar las gráficas relacionadas en la sección anexo 6.3.

Agente		Métricas	
Algoritmo	Función	Distancia acumulada promedio [m]	Cantidad de episodios
A2C	Función 1	42968.94	**16**
	Función 2	43110.95	15
	Función 3	**42153.71**	15
PPO	Función 1	**17048.40**	**150**
	Función 2	47485.22	13
	Función 3	20612.99	138
SAC	Función 1	25416.65	109
	Función 2	16648.85	**135**
	Función 3	**15642.66**	128
TD3	Función 1	27085.93	71
	Función 2	20442.26	**122**
	Función 3	**16648.85**	120

Tabla 5.2: Resultados de las pruebas de función de recompensa.

En la tabla 5.2 al considerar la métrica distancia acumulada promedio se observa que la función tres obtiene mejor resultado en tres de los algoritmos. Si bien en el algoritmo PPO no se obtiene el mejor resultado, la distancia acumulada promedio se encuentra apenas por encima del mejor resultado, el cual se logra con la función uno. En cuanto a la cantidad de episodios, se puede observar que tanto la función uno como la función dos obtienen los mejores resultados en dos de los algoritmos y la función tres no obtiene el mejor resultado en ninguno de ellos. Sin embargo, logra resultados no tan alejados de los mejores en todos los algoritmos, por lo que se considera que el rendimiento de esta función es más uniforme que el de las otras.
Luego de analizar los datos obtenidos, se considera que la mejor opción es utilizar la función de recompensa tres para realizar las pruebas finales.

5.4.3. Otras pruebas

En el proceso de definición de la solución propuesta se exploraron y probaron varios caminos que por diferentes razones y por no realizar las pruebas de forma rigurosa no forman parte de la experimentación documentada. A continuación se describen algunas de estas y por qué fueron descartadas.

Pruebas añadiendo la acción dentro de la observación: Como se mencionó en la solución propuesta, la espera de la acción es un problema. Con el fin de investigar si al persistir la acción se podía disminuir dicha espera, se incorporó la acción previa en el estado de observación y se bajó el tiempo de espera entre una acción y otra al mínimo. Luego de realizar algunas simples pruebas se observó que disminuir tanto el tiempo de espera seguía afectando negativamente, por lo que no solucionaba este problema. Es por eso que no se continuó con este estado de observación.

Pruebas de terminación anticipada: en el marco del proyecto se exploró la incorporación de la técnica de terminación anticipada, esto significa que aunque el horizonte sea fijo se determina una condición especial que finaliza la ejecución. A diferencia de las otras condiciones de finalización de episodios, esta no se relaciona directamente a la tarea que se quiere realizar o al objetivo deseado, sino que busca terminar la ejecución cuando el agente se encuentra en estados irrecuperables para lograr el objetivo. Tiene

como finalidad evitar que se recolecten datos que poco contribuyen al aprendizaje, principalmente en las primeras etapas de este.
Esta técnica fue probada, pero no se obtuvieron beneficios al utilizarla. Como condición de terminación anticipada se utilizó una cantidad arbitraria de pasos consecutivos en los que el velero se alejaba del punto objetivo. Se cree que una de las razones por las que no se notaron mejoras utilizando esta técnica se debe a que el velero tiene la posibilidad de recuperarse a pesar de alejarse del punto objetivo por una cantidad considerable de pasos consecutivos. Esta técnica tiene más beneficios en problemas en que el agente llega a estados que le es imposible recuperarse. Un ejemplo donde la aplicación es directa y tiene validez es el trabajo DeepMimic [57] en el cual se plantea como objetivo que un robot humanoide logre caminar y realizar acrobacias. El agente una vez que se cae no tiene posibilidad de volver a ponerse en pie, por lo que la caída del agente es un buen ejemplo de condición para terminación anticipada. Dado que no se obtuvieron buenos resultados y además, como se menciona en la técnica anterior, los episodios de largo variable no favorecen el aprendizaje por imitación, es que se decide desestimar esta incorporación al proyecto.

Pruebas de optimización de parámetros: Un aspecto importante para lograr un entrenamiento exitoso es la configuración de los hiperparámetros de los algoritmos de aprendizaje por refuerzo empleados. Para definirlos se decide utilizar una herramienta específica para ello, denominada Optuna [58]. Este software brinda la posibilidad de realizar una búsqueda automática de los hiperparámetros óptimos para cada algoritmo. Para ello se procedió a realizar la integración de esta herramienta al proyecto.
El objetivo de esta herramienta es realizar una serie de entrenamientos utilizando distintas configuraciones de parámetros. Por el alto costo que supone cada entrenamiento se utilizaron únicamente veinte combinaciones por algoritmo. Cada entrenamiento constó de 400.000 pasos utilizando diferentes escenarios y se evaluaron las distintas configuraciones utilizando diez escenarios distintos.
Debido al alto costo computacional que supone su utilización de forma correcta, es decir, utilizar una cantidad importante de combinaciones diferentes de hiperparámetros y realizar un entrenamiento y evaluación significativa, es que no se logra obtener beneficio alguno de su utilización. Sin embargo, se cree que es una herramienta de gran ayuda para la configuración de parámetros que debe ser considerada como parte de los trabajos a futuro.

5.5. Pruebas experimentales

Una vez finalizadas las pruebas pre-experimentales se realizan las pruebas finales, donde se entrenan los controladores en escenarios complejos, utilizando una gran cantidad de pasos, para luego evaluar su desempeño utilizando distintas métricas.
Esta sección queda dividida en cuatro partes, en la primera se entrenan los controladores y se evalúan de forma cuantitativa entre ellos y el controlador manual, en la siguiente parte se encuentra una evaluación cualitativa entre los mejores controladores de la primera parte y el controlador experto. Luego se encuentra otra evaluación cuantitativa de los mejores controladores, donde la dimensiones del velero se modifican para evaluar la adaptación de los controladores entrenados en la primera parte. Por último, se muestran pruebas exploratorias realizadas para proponer trabajos a futuro.

5.5.1. Entrenamiento y evaluación cuantitativa entre controladores

En esta sección se describen los detalles del entrenamiento de cada uno de los controladores, para luego analizar con un enfoque cuantitativo el desempeño de cada uno.

Se suma a dicha comparación el controlador manual.

Tomando como insumo los resultados de las pruebas pre-experimentales se definen tres tipos de entrenamientos aplicados a tres algoritmos (SAC, PPO, TD3), donde es importante destacar que el algoritmo A2C no es tomado en cuenta para las pruebas finales por los malos resultados y por el costo computacional de incluirlo en ellas. Además, la técnica de aprendizaje AIRL no es tomada en cuenta por lo argumentado en las conclusiones de las pruebas de imitación. Ya definidos los algoritmos a utilizar se definen los tres tipos de entrenamiento a utilizar:

- RL: Este agente realiza la totalidad del entrenamiento solo con aprendizaje por refuerzo.
- GAIL: Este agente realiza la totalidad del entrenamiento solo con aprendizaje por imitación utilizando la técnica GAIL.
- GAIL + RL: Este agente realiza la mitad de pasos con aprendizaje por imitación utilizando la técnica GAIL y luego continúa los restantes pasos usando aprendizaje por refuerzo.

Para estas pruebas se generan 3000 escenarios diferentes utilizando el método automático, el cual es detallado en la sección 5.2.1, 2400 se utilizan para el entrenamiento y 600 para la evaluación. Como métricas se utilizan todas las definidas previamente, estas son: distancia acumulada promedio, cantidad de episodios y recompensa promedio.

Habiendo definido los nueve agentes, los escenarios y las métricas, se realiza el entrenamiento que se compone de dos millones de pasos y luego para la evaluación se realizan 600 episodios.

Agente		Métricas		
Algoritmo	Entrenamiento	Recompensa promedio	Distancia acumulada promedio [m]	Cantidad de episodios
TD3	RL	-586.56	90576.18	55
	GAIL	-485.77	**48400.30**	58
	GAIL + RL	**-480.16**	49985.42	**81**
SAC	RL	**249.05**	**11834.66**	**445**
	GAIL	-468.75	47607.80	58
	GAIL + RL	-31.54	26718.31	174
PPO	RL	**-53.03**	**19244.34**	**418**
	GAIL	-304.29	43372.99	156
	GAIL + RL	-230.85	28368.07	354
Manual	-	-275.85	24188.99	402

Tabla 5.3: Resultados de la evaluación cuantitativa sobre 600 episodios

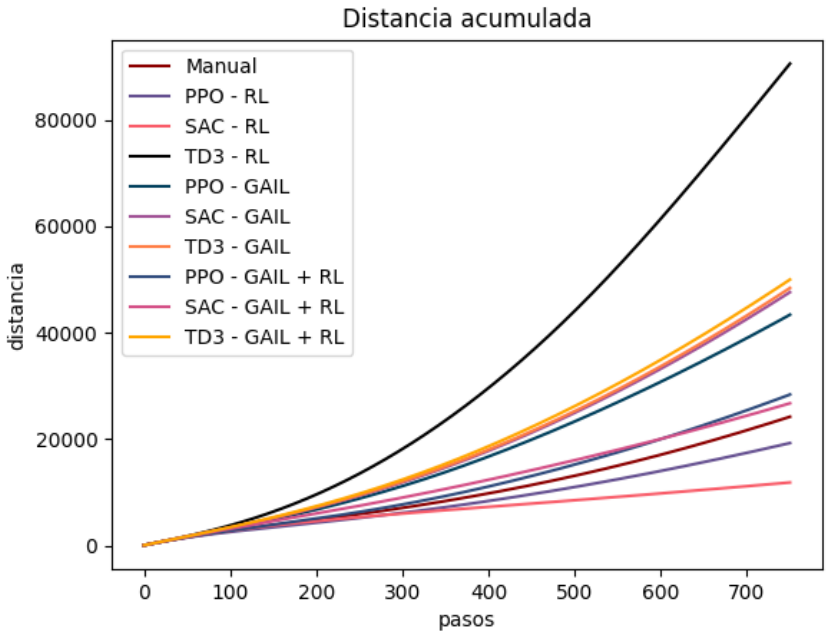

Figura 5.5: Resultados de la evaluación cuantitativa utilizando la métrica distancia acumulada promedio la cual es expresada en metros.

Al observar la tabla 5.3 es posible apreciar que la técnica de aprendizaje por imitación no obtiene buenos resultados con ninguno de los algoritmos utilizados. Con dos de los algoritmos, PPO y SAC, los resultados empeoran tanto al utilizar el agente GAIL + RL como al utilizar el agente GAIL, siendo aún peores utilizando este último. Esto no es así con el algoritmo TD3, utilizando este algoritmo se obtienen resultados similares con los tres agentes, no se nota desventaja o ventaja de la utilización del aprendizaje por imitación con este algoritmo. Si bien los resultados obtenidos con aprendizaje por imitación fueron malos, no significa necesariamente que estas técnicas no sean buenas, sino que hay múltiples factores que pueden afectar su desempeño. En este caso específico, dos factores que pueden haber afectado son el controlador manual, que es tomado como el experto para la imitación, y la cantidad y variedad de trayectorias utilizadas. Si bien al observar el comportamiento del velero utilizando el controlador manual en algunos escenarios sin grandes disturbios da a entender que se comporta bastante bien, la decisión de haberlo tomado como experto puede no haber sido acertada debido a que el resultado de la evaluación en los 600 escenarios utilizando la métrica de recompensa promedio es bastante pobre.

Poniendo en foco solo los entrenamientos que utilizan el agente RL se puede apreciar que TD3 no logra buenos resultados, mientras que PPO y SAC si logran buenos resultados, mejores que los obtenidos con el controlador manual al compararlos con las tres métricas utilizadas. En la comparación de estos últimos se puede ver que SAC tiene un desempeño ampliamente superior en la recompensa promedio y distancia acumulada promedio, sin embargo, en la cantidad de veces que se llega al objetivo PPO no queda muy lejos de SAC.
Al observar la gráfica 5.6 la cual muestra la métrica distancia acumulada promedio

se puede apreciar que SAC se destaca sobre el resto al mantenerse cerca del objetivo luego de los 400 pasos. El resto de los agentes, a partir de este punto, comienzan a aumentar la distancia acumulada promedio considerablemente, esto puede indicar que SAC logra aprender a mantenerse cerca del objetivo a lo largo del episodio y no solo en una parte de este.

5.5.2. Evaluación cualitativa entre controladores

Habiendo comparado los controladores de manera cuantitativa se seleccionan los dos que mejor resultado obtuvieron en todas las métricas para ser usado en las pruebas que comprenden esta sección. Por lo analizado en la sección anterior estos son los controladores que utilizan los algoritmos SAC y PPO junto con la técnica RL, a los cuales los llamaremos por el nombre del algoritmo utilizado de ahora en más. La idea de las pruebas cualitativas es poder analizar el desempeño de los controladores en escenarios arbitrarios, para así intentar comprender qué capacidades de navegación generaron.

Además de los controladores seleccionados se añade el controlador manual para la evaluación. Los escenarios utilizados son los doce definidos para las pruebas finales cualitativas detallados en la sección 5.2.2 y la evaluación se compone de un episodio por cada uno. Para el análisis de la evaluación se utilizan las tres métricas descritas. Dado que se muestran los resultados por cada episodio el promedio no cobra sentido. A continuación se presentan tres tablas con los resultados obtenidos en cada uno de los escenarios evaluados con las distintas métricas utilizadas. En ellas se identifican los puntos objetivos de acuerdo al rumbo y al entorno, este se divide de acuerdo a las dos condiciones climáticas como altos disturbios y bajos disturbios.

Punto objetivo		Recompensa final		
Entorno	Rumbo	SAC	PPO	Manual
Alto Disturbio	Ceñida Izq.	**313**	-561	-751
	Proa	**305**	-751	-736
	Ceñida Der.	**-52**	-432	-302
	Largo Der.	**158**	-476	-499
	Popa	**298**	-577	-279
	Largo Izq.	**220**	-551	-265
Bajo Disturbio	Ceñida Izq.	93	-163	**126**
	Proa	**271**	-11	-90
	Ceñida Der.	**443**	-154	106
	Largo Der.	**432**	-82	9
	Popa	**741**	-495	-68
	Largo Izq.	**328**	-324	77

Tabla 5.4: Resultados de las pruebas cualitativas con la métrica recompensa final.

Punto objetivo		Distancia acumulada [m]		
Entorno	Rumbo	SAC	PPO	Manual
Alto Disturbio	Ceñida Izq.	**17208.94**	23886.35	61399.29
	Proa	**26338.63**	35120.63	47893.62
	Ceñida Der.	33446.41	111902.65	**21370.36**
	Largo Der.	**12340.63**	42079.17	40042.23
	Popa	**10941.31**	13813.81	35792.36
	Largo Izq.	19518.06	41155.45	**17271.61**
Bajo Disturbio	Ceñida Izq.	9290.22	6592.39	**6586.11**
	Proa	20099.34	**10022.44**	25597.88
	Ceñida Der.	13626.66	8259.97	**6375.34**
	Largo Der.	12404.20	7746.01	**5642.47**
	Popa	14963.87	8721.61	**4967.55**
	Largo Izq.	20046.97	7840.99	**7065.70**

Tabla 5.5: Resultados de las pruebas cualitativas con la métrica distancia acumulada.

Punto objetivo		Paso en el que llega al objetivo		
Entorno	Rumbo	SAC	PPO	Manual
Alto Disturbio	Ceñida Izq.	-	-	-
	Proa	-	-	-
	Ceñida Der.	-	-	-
	Largo Der.	120	**67**	70
	Popa	239	**78**	-
	Largo Izq.	182	**83**	320
Bajo Disturbio	Ceñida Izq.	312	221	**196**
	Proa	-	**320**	-
	Ceñida Der.	-	273	**193**
	Largo Der.	527	288	**89**
	Popa	712	101	**91**
	Largo Izq.	712	145	**117**

Tabla 5.6: Resultados de las pruebas cualitativas indicando el número de paso en el que por primera vez se encuentra en el punto objetivo, en los casos que no se llega al punto objetivo se marca con un guión.

Al ver los resultados, se puede observar que existe una diferencia en los desempeños dependiendo la métrica observada. Específicamente se puede ver una diferencia en los resultados mirando la recompensa o mirando las métricas restantes, dado que SAC tiene una ventaja considerable en la recompensa, pero esta ventaja desaparece y en algunos casos obtiene peor desempeño para las otras métricas.

En el caso de la recompensa como se menciona previamente, SAC es sustancialmente superior a PPO y a Manual, donde estos últimos dos tienen un desempeño similar. Por lo que se podría decir que SAC es el agente que mejor aprendió a hacer la tarea de acuerdo a la función de recompensa definida. Estos resultados no se mantienen si se observa la métrica distancia acumulada, donde SAC sigue teniendo mejores resultados en los escenarios de alto disturbio, pero para los escenarios de bajos disturbios PPO y Manual lo superan, en especial este último obtiene los mejores resultados para este

tipo de condiciones. Por último al analizar la última tabla, se terminan de revertir los resultados, ya que SAC obtiene los peores resultados en la cantidad de puntos que alcanza el objetivo y en la velocidad a la que llega a los alcanzados, en cambio PPO en este caso tiene los mejores resultados tanto en la cantidad de puntos que alcanza como en la velocidad que lo hace.

Al analizar los rumbos a los que logran llegar los distintos controladores se puede observar que todos ellos alcanzan el punto objetivo en un entorno de bajos disturbios y a favor del viento, es decir, rumbos de largo y popa. Algo similar sucede en un entorno de altos disturbios, en los que los tres controladores logran llegar a los puntos objetivos en el rumbo de largo y todos menos el manual logran el rumbo de popa. Se da un panorama totalmente distinto al analizar los rumbos de ceñida y de proa, rumbos muy difíciles de navegar. Ninguno de los controladores logra alcanzarlos cuando se navega en un entorno de altos disturbios, sin embargo, el controlador PPO logra alcanzar todos ellos cuando se enfrenta a un entorno de bajos disturbios. Alcanzando incluso realizar la navegación en zigzag, necesaria para alcanzar el punto que se encuentra contra viento. Esta apreciación es posible realizarla al observar los videos de las pruebas cualitativas, los cuales se encuentran disponibles en el repositorio del proyecto [59].

A modo de resumen se entiende que SAC obtiene muy buena recompensa acercándose lentamente al objetivo, llegando tarde o directamente no llegando, una de las causas de este comportamiento es que mantenerse en el punto no es una tarea trivial y en muchos casos más compleja que navegar hacia un punto. Por lo que en términos de la tarea que se busca lograr, el comportamiento de SAC no es el buscado. Por otro lado PPO no obtiene buenos resultados en la recompensa, pero al analizar su navegación tiene un mejor desempeño. Si bien SAC supera a PPO en cantidad de puntos alcanzados en las pruebas cuantitativas, resultado reflejado en la tabla de dichas pruebas 5.3, al observar como se alcanzan algunos puntos, se cree que PPO tiene una navegación más realista o más acorde al objetivo planteado en este proyecto.

5.5.3. Evaluación cambiando la configuración del velero

Por último se realiza una tercera evaluación, utilizando los mismos controladores que en la evaluación cualitativa. En esta evaluación se intenta evaluar la adaptación de los controladores al cambio de alguno de los componentes del velero. Para esto se modifican los siguientes tres componentes: vela, timón y quilla.
Dado que el simulador para calcular las fuerzas aplicadas a cada componente las multiplica por el área del componente en cuestión, es que se decide modificar el área para simular diferentes configuraciones del velero. Se toma como base las áreas utilizadas en el entrenamiento, a partir de estas se aumenta el valor para analizar cómo repercute este cambio en el desempeño del controlador.

A continuación se detallan las configuraciones del velero a evaluar:

- Dimensión original: Vela = $1{,}44m^2$, Quilla = $1{,}00m^2$, Timón = $1{,}00m^2$
- Dimensión 1: Vela = $1{,}80m^2$, Quilla = $1{,}25m^2$ Timón = $1{,}25m^2$
- Dimensión 2: Vela = $2{,}16m^2$ Quilla = $1{,}50m^2$, Timón = $1{,}50m^2$

Los escenarios para esta prueba son los mismos 600 utilizados para la evaluación cuantitativa realizada previamente, donde también se utilizan las tres mismas métricas.

Por lo que se evalúan los controladores sobre 600 escenarios, variando en cada prueba las dimensiones ya descritas.

Dimension	Controlador	Recompensa promedio	Distancia acumulada promedio [m]	Cantidad de episodios
Original	SAC	**259.05**	**11834.66**	**445**
	PPO	-53.03	19244.34	418
	Manual	-275.85	24188.99	402
Dimensión 1	SAC	**240.75**	**14175.04**	**432**
	PPO	-103.15	26276.04	414
	Manual	-260.53	25805.71	415
Dimensión 2	SAC	**204.73**	**16696.89**	**408**
	PPO	-131.03	31471.47	399
	Manual	-279.86	28201.31	**408**

Tabla 5.7: Resultados de las pruebas modificando la configuración del velero

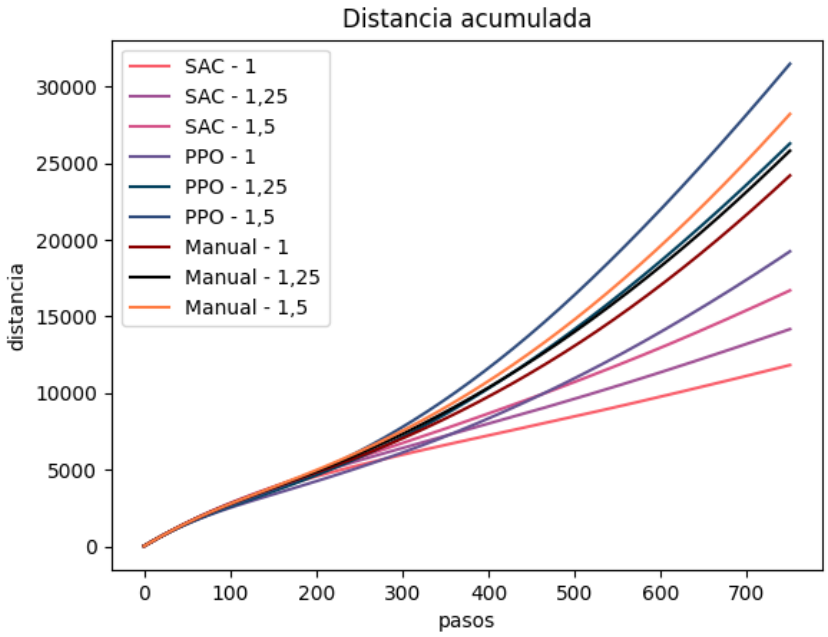

Figura 5.6: Resultados de las pruebas modificando la configuración del velero utilizando la métrica distancia acumulada promedio, la cual es expresada en metros.

Al analizar los resultados de las pruebas adaptativas se puede apreciar que tanto SAC como PPO empeoran a medida que se agrandan las dimensiones de las partes del velero, sin embargo, esta baja en el desempeño no se ve en el controlador Manual. A pesar de este declive de los controladores de aprendizaje por refuerzo es importante mencionar que SAC se mantiene con mejores resultados que el controlador Manual en todos los casos y PPO en general también lo supera.

Estos resultados son un indicador de que, si bien los resultados de los controladores entrenados siguen siendo superiores al manual, existe un espacio para mejorar ajustando y explorando otras técnicas al momento de cambiar el velero que se controla y para soportar el desgaste del velero. Existen muchos caminos para atacar este problema, ejemplos de estos son el aprendizaje por transferencia o la continuación del entrenamiento en el nuevo velero o en el mismo velero luego de un tiempo de navegación en el que las partes de este se encuentren desgastadas.

5.5.4. Otras pruebas

Dado que los resultados obtenidos no fueron excelentes se decide realizar algunas pruebas para poder evaluar posibles soluciones para trabajos futuros y así obtener un controlador con mayor capacidad de adaptación a las diferentes situaciones climáticas y al desgaste del velero.

Pruebas de horizonte variable: se decide explorar si al utilizar horizonte variable el aprendizaje para navegar hacia un punto resulta más sencillo. Para ello se realiza el mismo entrenamiento y evaluación que el utilizado en las pruebas finales con los algoritmos PPO y SAC. Dado que el horizonte es variable la métrica que se considera relevante para comparar es la cantidad de episodios.
Se decide modificar la recompensa para premiar el llegar rápido al punto objetivo, durante el recorrido a éste la recompensa es la misma que se plantea en las pruebas anteriores, una vez alcanzado el punto objetivo se realiza la resta entre la cantidad de pasos límite y los pasos que le tomó llegar al objetivo y se multiplica por dos.
El controlador generado con el algoritmo PPO logra llegar a 489 puntos objetivos, 71 puntos más que el controlador generado en el proyecto con este algoritmo. Por otro lado, al considerar el algoritmo SAC el resultado es de 418 puntos objetivos, 27 puntos objetivos menos que el controlador generado con este algoritmo. Si bien en este caso el controlador no mejora en la cantidad de puntos objetivos, al observar su comportamiento, se considera que mejora la forma de navegación en los rumbos que es más fácil navegar, avanzando rápidamente al punto objetivo.
Los resultados de estas pequeñas pruebas no son contundentes, sin embargo, se considera que esta modificación puede ser un buen camino a seguir para lograr mejores resultados.

Pruebas de alargamiento de entrenamiento: se desea evaluar si los controladores logran un mejor rendimiento al continuar el entrenamiento, dado que la cantidad de pasos de entrenamiento junto a la configuración de los hiperparámetros pueden ser unos de los motivos por los cuales no se logre el mejor rendimiento en el aprendizaje. Como es mencionado anteriormente, obtener la óptima configuración de hiperparámetros resulta muy costoso en cuanto a cantidad de tiempo, es por eso que se decide explorar el alargamiento del entrenamiento como posible solución.
Para estas pruebas se utiliza el controlador PPO seleccionado y se continúa su entrenamiento por dos millones de pasos más bajo las mismas condiciones en que se realizó su entrenamiento previo. Luego se realiza su evaluación utilizando los 600 episodios con los que se evalúa en las pruebas cuantitativas.
El controlador obtenido luego de este entrenamiento obtiene peores resultados que el controlador PPO antes de dicho entrenamiento en las tres métricas utilizadas. La distancia acumulada promedio aumenta de 19,244, 34m a 23,839, 81m, la recompensa promedio obtenida disminuye de $-53,03$ a $-72,28$ y la cantidad de episodios disminuye de 418 a 390. Este resultado da a entender que aumentar la cantidad de pasos

no es uno de los caminos a continuar para lograr mejores controladores.

Capítulo 6

Conclusiones y trabajos a futuro

6.1. Conclusiones

El objetivo de este proyecto se basa en la creación de un controlador adaptativo para veleros autónomos utilizando las técnicas de aprendizaje por refuerzo y de aprendizaje por imitación. Luego de haber trabajado en ello se considera que parte del objetivo es cumplido. Si bien los controladores generados no logran navegar en todas las condiciones climáticas planteadas, se logra un controlador capaz de navegar en mayor cantidad de escenarios que el controlador manual, logrando incluso navegar en rumbos contra el viento. Es por estas razones que si bien se entiende que los controladores generados no son excelentes se va por el camino correcto para lograr generar un controlador que logre navegar en mayor cantidad de situaciones y rumbos.

En cuanto a la utilización del aprendizaje por imitación se puede decir que no fue beneficioso en este proyecto, incluso se considera que resultó contraproducente en algunos entrenamientos. Sin embargo, esto no significa que esta técnica no tenga éxito en otros trabajos, incluso en trabajos similares realizando algunas modificaciones, como puede ser la función de recompensa o lograr conseguir otro controlador para ser considerado como experto.

La navegación es una tarea de gran complejidad, y a pesar de haber leído y hablado con expertos en el tema, requiere conocerla aún más. Analizando los resultados, se visualiza que un posible mejor camino sería considerar de forma separada los comportamientos de navegar hacia un punto objetivo y mantenerse en un punto determinado. Se cree que esta opción de entrenar distintos controladores para realizar diferentes comportamientos puede simplificar y solucionar algunos de los problemas que surgieron en este trabajo. Esta es una de las razones por la cual se realiza una prueba de entrenar un controlador modificando el objetivo a navegar hacia un punto objetivo, notando que es un camino que tiene sentido considerar en trabajos futuros.

Otra conclusión a resaltar es respecto a la validez de las pruebas pre-experimentales en este tipo de proyecto, ya que sus resultados no fueron congruentes con los de las pruebas experimentales. Una de las posibles causas es por la naturaleza misma del aprendizaje automático, donde los modelos necesitan converger hacia un óptimo para lograr buenos resultados. Al realizar pruebas cortas esa convergencia no se logra ni se está cerca, dando lugar a resultados no representativos de lo que se podría lograr continuando el aprendizaje. Con esto como precedente, utilizar pruebas pre-experimentales

no parece ser una buena herramienta para determinar los caminos a seguir, siendo tal vez una mejor opción elegirlos basándose en documentación teórica y trabajos previos.

6.2. Trabajos a futuro

Desde el inicio de este proyecto surgen gran cantidad de caminos alternativos a los tomados, a los que se suman varias ideas para continuarlo. Por la naturaleza misma del proyecto surgen distintas formas de resolver el mismo problema, siendo necesario dejar muchos caminos pendientes por el tiempo que insume cada uno de ellos. A continuación se detalla una serie de trabajos que se creen pertinentes para explorar:

- En principio se plantea como trabajo a realizar la utilización de los mejores controladores generados en un velero real, pudiendo evaluar realmente el comportamiento de estos. Esta tarea si bien incluye enormes desafíos, es la más interesante a realizar, ya que el objetivo final al crear el controlador es poder controlar un velero autónomo real.
- Como es mencionado en este documento, uno de los problemas que se enfrentó es la decisión del tiempo óptimo de espera entre que se realiza la acción y se obtiene la observación, esto se cree que tiene un gran impacto en el proceso de aprendizaje por refuerzo. Por eso resulta interesante incorporar la técnica planteada en el trabajo [54], el cual intenta solucionar este problema.
- Algo ya mencionado a lo largo de este documento es la búsqueda en la mejora del desempeño de los controladores entrenados al cambiar de velero o modificar las dimensiones del mismo. Para esto existen varios caminos posibles, uno de los que resulta más interesante es aplicar la técnica de aprendizaje por transferencia.
- Otro trabajo a realizar es la utilización de forma correcta de la herramienta Optuna. La correcta configuración de hiperparámetros tiene un enorme impacto en el aprendizaje por refuerzo, pudiendo generar una mejora sustancial tanto en la velocidad del aprendizaje como en los resultados finales de este. Como es mencionado previamente, esta búsqueda es de un costo computacional muy alto y por este motivo no se pudo utilizar correctamente, ya que se debería realizar un entrenamiento y evaluación con una cantidad de datos significativa. Habiendo reducido los agentes donde se busca optimizar los hiperparámetros, y con un mayor tiempo disponible o mayor capacidad de cómputo, es posible realizar esta optimización de forma correcta.
- Plantear el objetivo de este trabajo como dos por separado, es decir, tener dos tareas diferentes: una donde se busca navegar hacia un punto y otra donde se busca permanecer en un punto específico. Ya que ambas tareas por separado presentan una alta complejidad y no resulta trivial diseñar una función de recompensa que contemple ambas. Por lo que se entrenaría un controlador para cada tarea y luego se combinarían.
- Explorar el diseño de una función de recompensa compleja, incorporando el conocimiento de expertos, intentando que se adecúe de mejor forma al buen control de un velero en términos de navegación.
- Un camino que resulta muy interesante de transitar es incorporar al controlador la capacidad de evitar obstáculos. Esto implicaría varios cambios, en especial se debería agregar la información necesaria al estado de observación y ajustar la función de recompensa para penalizar cualquier tipo de colisión.
- Algo que se destaca del velero como vehículo autónomo, es la capacidad de propulsión a través del viento, sin la necesidad de ninguna fuente de energía externa.

Sin embargo, si es necesaria una fuente de energía externa para el movimiento de los actuadores, por eso un trabajo a futuro puede ser la penalización del movimiento de los actuadores, dado que realizar el movimiento de estos requiere de energía que sería deseable minimizar. Tal vez esto derive en otros beneficios, ya que en grandes embarcaciones también se busca mover lo menos posible la vela porque cada movimiento disminuye la velocidad adquirida.

- Incorporar la técnica de validación cruzada para entrenar y evaluar los controladores. Esta técnica fue definida para utilizar al principio de este trabajo, pero tuvo que ser descartada, ya que su aplicación adiciona un tiempo muy grande a la experimentación. Si bien el costo en tiempo es alto, se cree que sería beneficioso el uso de ella para obtener resultados más contundentes a la hora de sacar conclusiones.

Referencias

[1] Vela, partes de un velero. https://recursosparaeldeporte.blogspot.com/2010/04/vela-partes-de-un-velero.html.

[2] Singladuras náuticas. Navegación a vela. http://singladurasnauticas.yolasite.com/navegacionvela.php, visitado en 2021-5-15.

[3] The Microtransat Challenge. The microtransat challenge.

[4] OceanAlpha. Autonomous water sampling & monitoring boat.

[5] Pasan Prempraneerach, P. & Kulvanit. Autonomous robot boat for water sampling and environmental data acquisition tasks. 2010.

[6] Analisis de estado del arte. https://gitlab.fing.edu.uy/agustin.rieppi/pg_velero_autonomo/-/blob/master/Documents/Informe_Proyecto_Grado_Estado_Del_Arte.pdf, visitado en 2022-03-14.

[7] J. Javier Fernandez Mario Prats, Javier Perez and Pedro J. Sanz. An open source tool for simulation and supervision of underwater intervention missions. 2012.

[8] Olivier Kermorgant. A dynamic simulator for underwater vehicle-manipulators. 2014.

[9] Marcelo Paravisi, Davi H. Santos, Vitor Jorge, Guilherme Heck, Luiz Marcos Gonçalves, and Alexandre Amory. Unmanned surface vehicle simulator with realistic environmental disturbances. 2019.

[10] Brian Bingham, Carlos Aguero, Michael McCarrin, Joseph Klamo, Joshua Malia, Kevin Allen, Tyler Lum, Marshall Rawson, and Rumman Waqar. Toward maritime robotic simulation in gazebo. October 2019.

[11] Alexandre EVAIN. Simulation of an autonomous sailboat with ros. 2021.

[12] Sébastien Lemaire, Yu Cao, Thomas Kluyver, Daniel Hausner, Camil Vasilovici, Zhong-Yuen Lee, Umberto José Varbaro, and Sophia M. Schillai. Adaptive Probabilistic Tack Manoeuvre Decision for Sailing Vessels. In Sophia M. Schillai and Nicholas Townsend, editors, *Robotic Sailing 2018*, pages 95–103, 2018.

[13] Christophe VIEL Ulysse VAUTIER. Ros nodes for plymouth sailboat project. 2020.

[14] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004.

[15] Rodrigo Pereira Abou Rejaili and Jonathas Marcelo Pereira Figueiredo. Deep reinforcement learning algorithms for ship navigation in restricted waters. *Mecatrone*, 3(1), Dec. 2018.

[16] Yunduan Cui, Shigeki Osaki, and Takamitsu Matsubara. Reinforcement learning ship autopilot: Sample efficient and model predictive control-based approach, 2019.

[17] Xinyu Zhang, Chengbo Wang, Yuanchang Liu, and Xiang Chen. Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning. *Sensors*, 19(18), 2019.

[18] Andouglas Gonçalves da Silva Junior, Davi Henrique dos Santos, Alvaro Pinto Fernandes de Negreiros, João Moreno Vilas Boas de Souza Silva, and Luiz Marcos Garcia Gonçalves. High-level path planning for an autonomous sailboat robot using q-learning. *Sensors*, 20(6), 2020.

[19] Theophil Ruzicka. Model based design of a sailboat autopilot. 2017.

[20] Gillian Hayes Philip Sterne. Reinforcement sailing. 2006.

[21] Corentin Jegat. Control of autonomous sailboats. 2019.

[22] Shaolong Yang, Chuan Liu, Ya Liu, Jinxin An, and Xianbo Xiang. Generic and flexible unmanned sailboat for innovative education and world robotic sailing championship. *Frontiers in Robotics and AI*, 8:27, 2021.

[23] Benoit Clement. Control algorithms for a sailboat robot with a sea experiment. 2013.

[24] Jon Melin. Modeling, control and state-estimation for an autonomous sailboat. 2015.

[25] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[26] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

[27] Ching-Chang Wong, Shao-Yu Chien, Hsuan-Ming Feng, and Hisasuki Aoyama. Motion planning for dual-arm robot based on soft actor-critic. *IEEE Access*, 9:26871–26885, 2021.

[28] Luckeciano Carvalho Melo and Marcos Ricardo Omena Albuquerque Máximo. Learning humanoid robot running skills through proximal policy optimization. In *2019 Latin american robotics symposium (LARS), 2019 Brazilian symposium on robotics (SBR) and 2019 workshop on robotics in education (WRE)*, pages 37–42. IEEE, 2019.

[29] Chengmin Zhou, Bingding Huang, and Pasi Fränti. An advantage actor-critic algorithm for robotic motion planning in dense and dynamic scenarios. *arXiv preprint arXiv:2102.03138*, 2021.

[30] PB Khoi, NT Giang, and H van Tan. Control and simulation of a 6-dof biped robot based on twin delayed deep deterministic policy gradient algorithm. *Indian J. Sci. Technol*, 14:2460–2471, 2021.

[31] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[33] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[34] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018.

[35] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[36] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[37] Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[38] Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*, 05 2000.

[39] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 101–103. ACM Press, 1998.

[40] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[41] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[42] Justin fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. 10 2017.

[43] Mauricio Fadel Argerich. 5 frameworks for reinforcement learning on python. `https://towardsdatascience.com/5-frameworks-for-reinforcement-learning-on-python-1447fede2f18`, visitado en 2022-03-14.

[44] Piotr Januszewski Vladimir Lyashenko. The best tools for reinforcement learning in python you actually want to try. `https://neptune.ai/blog/the-best-tools-for-reinforcement-learning-in-python`, visitado en 2022-03-14.

[45] Thomas Simonini. On choosing a deep reinforcement learning library. `https://blog.dataiku.com/on-choosing-a-deep-reinforcement-learning-library`, visitado en 2022-03-14.

[46] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. `http://jmlr.org/papers/v22/20-1364.html`, visitado en 2022-03-14.

[47] Alexander Kuhnle, Michael Schaarschmidt, and Kai Fricke. Tensorforce: a tensorflow library for applied reinforcement learning. Web page. `https://github.com/tensorforce/tensorforce`, visitado en 2022-03-14.

[48] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow, 2018. `https://github.com/tensorflow/agents`, visitado en 2022-03-14.

[49] Steven Wang, Sam Toyer, Adam Gleave, and Scott Emmons. The `imitation` library for imitation learning and inverse reinforcement learning. `https://github.com/HumanCompatibleAI/imitation`, 2020.

[50] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[51] Samantha James Roland Stul. Beaufort wind force scale.

[52] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3):1464–1468, 1997.

[53] The SciPy community. The scipy community.

[54] Baiming Chen, Mengdi Xu, Zuxin Liu, Liang Li, and Ding Zhao. Delay-aware multi-agent reinforcement learning for cooperative and competitive environments. *arXiv preprint arXiv:2005.05441*, 2020.

[55] Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. 2017.

[56] Gianandrea Mannarini, Rita Lecci, and Giovanni Coppini. Introducing sailboats into ship routing system visir. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–6, 2015.

[57] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic. *ACM Transactions on Graphics*, 37(4):1–14, Aug 2018.

[58] Optuna. Optimize your optimization.

[59] Agustín Rieppi and Florencia Rieppi. Pg_velero_autonomo. `https://gitlab.fing.edu.uy/agustin.rieppi/pg_velero_autonomo`, 2021.

[60] Author. Title. *Journal*, 10(2):1–2, 2001.

[61] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[62] International Robotic Sailing Conference. World robotic sailing championship.

[63] Jose Antonio Martin H. Estudios sobre sistemas adaptativos con aplicaciones en la robótica autónoma y los agentes inteligentes. *Archivo Digital UPM*, 2913, 03 2009.

[64] Ahmed Hussein, Mohamed Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 50, 04 2017.

[65] Wim Meeussen Tully Foote, Eitan Marder-Eppstein. tf. `http://wiki.ros.org/tf`.

Glosario

SAC: del inglés *Soft Actor Critic*, algoritmo de aprendizaje por refuerzo.

PPO: del inglés *Proximal Policy Optimization*, algoritmo de aprendizaje por refuerzo.

TD3: del inglés *Twin Delayed Deep Deterministic Policy Gradients*, algoritmo de aprendizaje por refuerzo.

A2C: del inglés *Advantage Actor CriticProximal Policy Optimization*, algoritmo de aprendizaje por refuerzo.

GAIL: del inglés *Generative Adversarial Imitation Learning*, algoritmo de aprendizjae por imitación.

AIRL: del inglés *Adverserial Inverse Reinforcement Learning*, algoritmo de aprendizaje por imitación.

RL: del inglés *Reinforcenment learning*, refiere a realizar todo el entrenamiento utilizando aprendizaje por refuerzo.

RL + GAIL o RL + AIRL: Refiere a realizar la mitad del entrenamiento utilizando aprendizaje por imitación y la otra mitad utilizando aprendizaje por refuerzo.

Anexo

6.3. Gráficas con los resultados de las pruebas de recompensa

A continuación se encuentran las gráficas con los resultados de las pruebas pre-experimentales en las que se comparan distintas funciones de recompensa con el objetivo de seleccionar una de ellas para las pruebas finales.

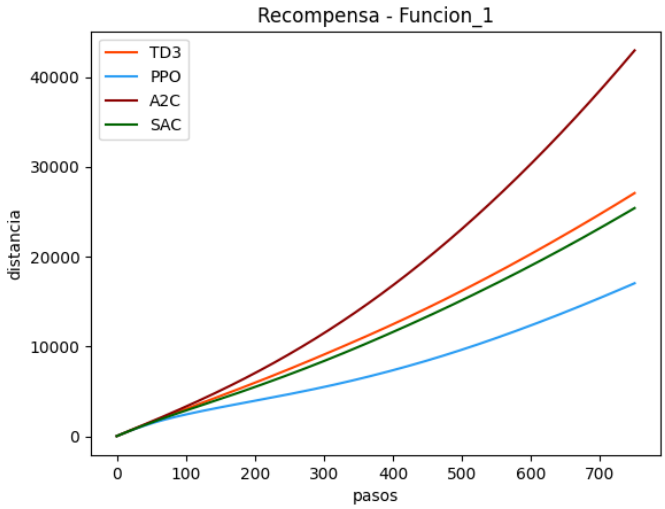

Figura 6.1: Resultados utilizando la función uno con la métrica distancia acumulada promedio, la cual es expresada en metros.

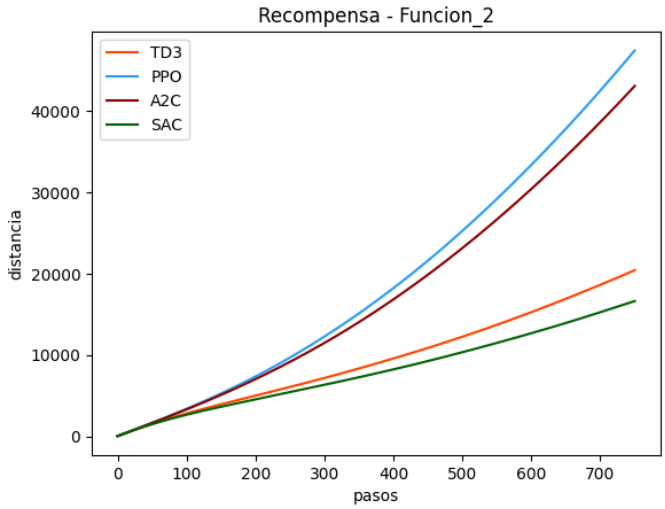

Figura 6.2: Resultados utilizando la función dos con la métrica distancia acumulada promedio, la cual es expresada en metros.

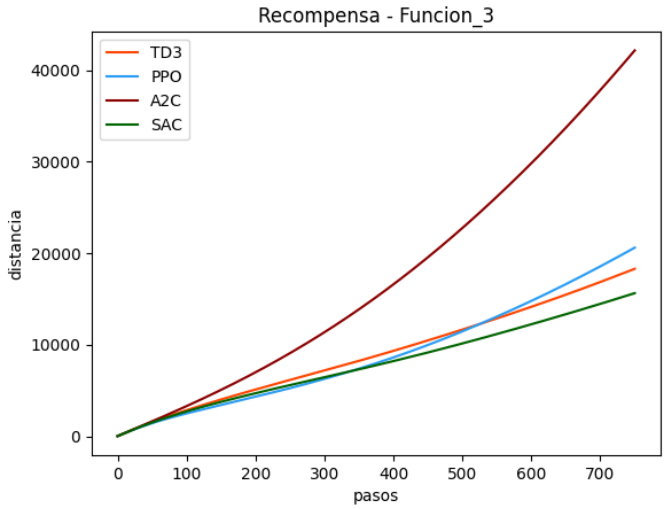

Figura 6.3: Resultados utilizando la función tres con la métrica distancia acumulada promedio, la cual es expresada en metros.

6.4. Gráficas con los resultados de las pruebas de imitación

A continuación se encuentran las gráficas con los resultados de las pruebas pre-experimentales en las que se comparan distintos entrenamientos utilizando o no algoritmos de imitación para decidir si realizar entrenamientos con ellos en las pruebas finales.

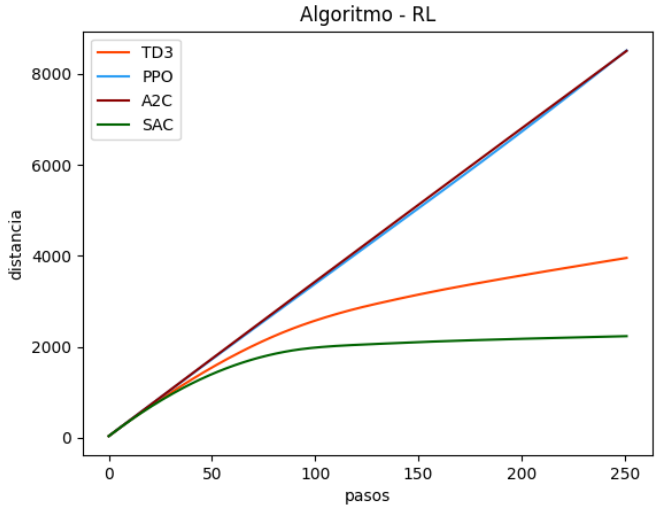

Figura 6.4: Resultados de entrenar sin algoritmos de imitación con la métrica distancia acumulada promedio, la cual es expresada en metros.

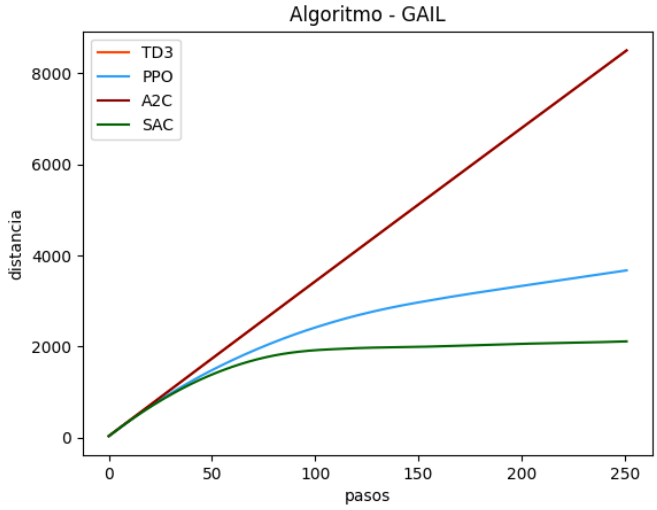

Figura 6.5: Resultados de utilizar algoritmo GAIL con la métrica distancia acumulada promedio, la cual es expresada en metros.

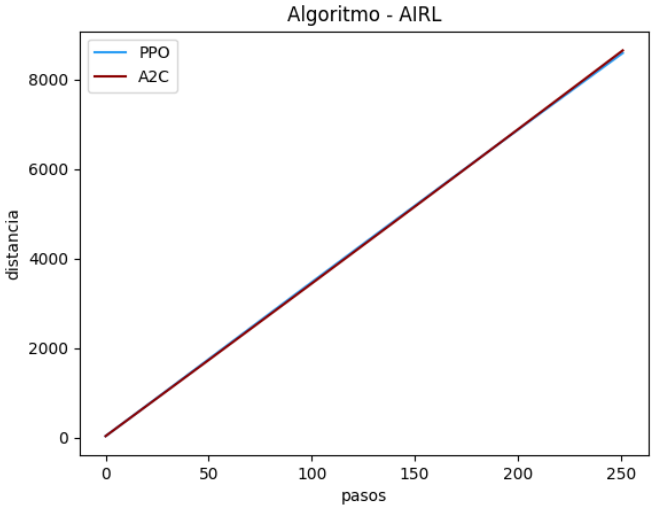

Figura 6.6: Resultados de utilizar algoritmo AIRL con la métrica distancia acumulada promedio, la cual es expresada en metros.

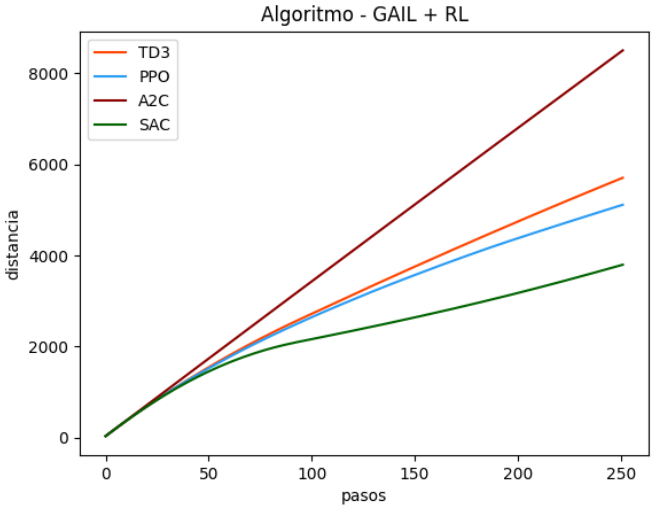

Figura 6.7: Resultados de utilizar algoritmo GAIL y luego entrenar sin algoritmo de imitación con la métrica distancia acumulada promedio, la cual es expresada en metros.

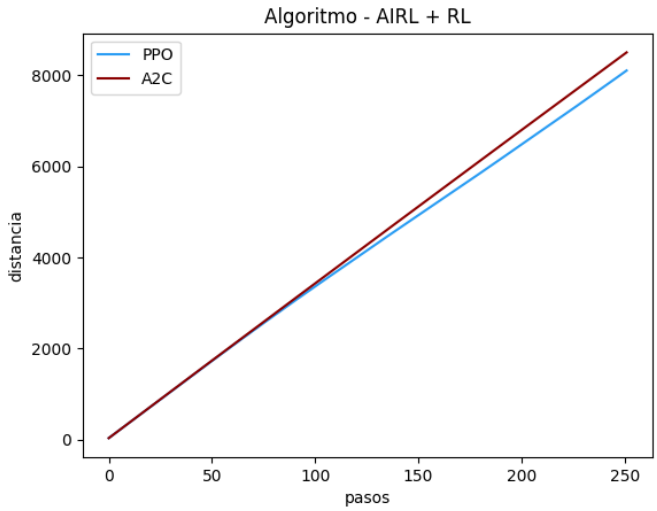

Figura 6.8: Resultados de utilizar algoritmo AIRL y luego entrenar sin algoritmo de imitación con la métrica distancia acumulada promedio, la cual es expresada en metros.

Esta es la última página.
Compilado el miércoles 12 octubre, 2022.