Deusto

Facultad de Ingeniería
Ingeniaritzako Fakultatea
Faculty of Engineering

Estudio de la planificación de trayectorias en entornos dinámicos para manipuladores industriales mediante aprendizaje por refuerzo

Tesis doctoral presentada por IGNACIO FIDALGO ASTORQUIA
dentro del programa de doctorado en INGENIERÍA PARA LA SOCIEDAD DE LA INFORMACIÓN Y DESARROLLO SOSTENIBLE

Doctorando

Firmado por Ignacio Fidalgo Astorquia el día 28/10/2025 con un certificado emitido por Ignacio Fidalgo Astorquia

Los directores

TELLAECHE IGLESIAS, ALBERTO (AUTENTICACIÓN)

Firmado digitalmente por TELLAECHE IGLESIAS, ALBERTO (AUTENTICACIÓN)
Fecha: 2025.10.29 09:44:40 +01'00'

Firmado digitalmente por VAZQUEZ GOMEZ JUAN IGNACIO - 30669432J
Fecha: 2025.10.28 19:07:44 +01'00'

A mis compañeros de trabajo,
por compartir este camino.

Resumen

Esta tesis estudia cómo los robots industriales pueden aprender a planificar sus movimientos de manera más autónoma y eficiente. El trabajo se centra en los manipuladores, que son brazos robóticos diseñados para realizar tareas de precisión en entornos industriales, y explora nuevas formas de generar trayectorias de movimiento que les permitan alcanzar objetivos de manera segura y flexible.

A lo largo de la investigación se revisan primero las bases del problema: cómo se describen los movimientos de un robot, qué limitaciones físicas y geométricas condicionan su comportamiento, y qué métodos se han utilizado tradicionalmente para calcular trayectorias. Estos métodos clásicos, aunque probados y fiables, muestran limitaciones cuando se aplican en entornos dinámicos, cambiantes o poco estructurados, como ocurre en la industria moderna.

La propuesta de la tesis consiste en aprovechar técnicas de aprendizaje automático que permiten a los robots adquirir experiencia en simulaciones antes de enfrentarse a situaciones reales. De este modo, el robot no depende únicamente de algoritmos programados de antemano, sino que puede aprender a moverse a partir de la práctica acumulada. Para lograrlo, se ha diseñado un proceso que combina entrenamiento en entornos virtuales, incorporación de ejemplos provenientes de métodos tradicionales y una estrategia de aprendizaje progresivo, en la que el robot comienza con tareas sencillas y avanza hacia retos más complejos.

Los resultados muestran que este enfoque permite obtener movimientos más consistentes y seguros, que además pueden ejecutarse

en robots reales sin necesidad de largos tiempos de ajuste. En las pruebas realizadas, el sistema ha demostrado no solo ser capaz de alcanzar los objetivos planteados, sino también de adaptarse mejor que los métodos clásicos cuando se introducen restricciones adicionales o cuando los objetivos cambian de posición.

Las principales aportaciones de la tesis incluyen: una metodología de aprendizaje que acelera la adquisición de habilidades de planificación, un proceso de validación que asegura que las trayectorias aprendidas son seguras y viables en robots reales, y una comparación detallada que evidencia las ventajas de este enfoque frente a los planificadores más utilizados en la industria.

Finalmente, se plantean posibles líneas de investigación futura. Entre ellas destacan la posibilidad de transferir el conocimiento adquirido a diferentes robots y entornos sin necesidad de volver a entrenar desde cero, la integración con técnicas de optimización que permitan movimientos aún más precisos, y la exploración de sistemas multimodales en los que convivan distintas estrategias de planificación para aprovechar lo mejor de cada una.

Abstract

This dissertation explores how industrial robots can be endowed with greater autonomy and adaptability when planning their motions. The focus is placed on robotic manipulators—mechanical arms widely used in manufacturing—and on developing strategies that enable them to generate safe and efficient trajectories while coping with changing and unstructured environments.

The study begins by revisiting the fundamentals of motion planning: how robot movements are represented, what physical and geometric constraints must be considered, and how conventional algorithms address these challenges. While these classical approaches are reliable, they often fall short in modern industrial contexts that demand flexibility and real-time adaptability.

The proposed approach leverages machine learning techniques that allow robots to acquire experience through simulation before deployment in the real world. Instead of relying solely on predefined algorithms, the robot improves its performance by practicing and refining its strategies. To this end, a training process was designed that integrates three elements: large-scale virtual experimentation, guidance from traditional planners, and a progressive learning scheme that gradually increases task complexity.

Experimental results show that this method produces trajectories that are more consistent, safer, and readily transferable to physical robots without extensive fine-tuning. The learned policies not only

achieve the specified goals but also demonstrate superior adaptability compared to classical methods when dealing with additional constraints or moving targets.

The main contributions of this work include: a learning methodology that accelerates the acquisition of planning skills, a validation framework that ensures safe deployment on real robots, and a comprehensive benchmark highlighting the advantages of the proposed approach over widely used industrial planners.

Finally, the dissertation outlines several promising research directions, such as transferring learned knowledge across different robots and environments, integrating with optimization techniques for greater precision, and developing multimodal planning frameworks that combine complementary strategies.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que han hecho posible la realización de esta tesis.

En primer lugar, a mis directores de tesis, por su guía constante, por su rigor científico y por la confianza depositada en mí a lo largo de estos años. Sus orientaciones han sido fundamentales para mantener el rumbo de esta investigación y para crecer como investigador.

A mis compañeros y compañeras de DeustoTech y de la Universidad de Deusto, por el apoyo diario y el buen ambiente de colaboración y trabajo que hicieron de este recorrido una experiencia enriquecedora.

A la Universidad de Deusto, por brindarme el entorno académico y los recursos necesarios para llevar a cabo este trabajo, así como a los proyectos que lo han financiado, sin los cuales esta investigación no hubiera sido posible.

En el plano personal, a mi familia, por su paciencia, cariño incondicional y aliento en los momentos más exigentes. A mis amigos y amigas, por recordarme la importancia de equilibrar la vida académica con la personal.

Finalmente, a todas las personas que, de una u otra manera, han contribuido a este camino: mi gratitud es también para ellas. Esta tesis es tanto un logro personal como el resultado de un esfuerzo colectivo.

Gracias,
Ignacio Fidalgo Astorquia

Índice general

Índice de figuras

Índice de tablas

Acrónimos

ABIT*	Advanced BIT*
API	Application Programming Interface
ATD*	Anytime Truncated D*
BKPIECE	Bidirectional KPIECE
BiEST	Bidirectional Expansive Space Trees
BIT*	Batch Informed Trees Star
BiTRRT	Bidirectional Transition-based Rapidly-exploring Random Tree
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
DAgger	Dataset Aggregation
DDPG	Deep Deterministic Policy Gradient
DH	Denavit-Hartenberg
DRL	Deep Reinforcement Learning
EST	Expansive Space Trees
FK	Forward Kinematics
FMT*	Fast Marching Tree

GAIL	Generative Adversarial Imitation Learning
GDL	Grado de Libertad
GPMP	Gaussian Process Motion Planner
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HER	Hindsight Experience Replay
IK	Inverse Kinematics
KOMO	K-Order Markov Optimization
KPIECE	Kinodynamic Planning by Interior-Exterior Cell Exploration
LTL	Linear Temporal Logic
MDP	Markov Decision Process
MJCF	MuJoCo XML Configuration Format
MPNet	Motion Planning Networks
OMPL	Open Motion Planning Library
PDST	Path-Directed Subdivision Tree
PID	Controlador Proporcional-Integral-Derivativo
PPO	Proximal Policy Optimization
PRM	Probabilistic Roadmap
PRM*	Probabilistic Roadmap star
PROJEST	Projection-based EST
RABIT*	Regionally Accelerated BIT*
RRT	Rapidly-exploring Random Tree

RRT*	Rapidly-exploring Random Tree Star
RRT-Connect	Rapidly-exploring Random Tree Connect
RL	Reinforcement Learning
ROS	Robot Operating System
ROS-I	ROS-Industrial
SAC	Soft Actor-Critic
SBL	Single-Query Bi-directional Lazy PRM
SPARS	Sparse Roadmap Spanner
SPARS2	Sparse Roadmap Spanner 2
STOMP	Stochastic Trajectory Optimization for Motion Planning
STRIDE	Search Tree with Resolution Independent Density Estimation
SST	Stable Sparse-RRT
TD3	Twin Delayed Deep Deterministic Policy Gradient
TD3	Twin Delayed Deep Deterministic Policy Gradient
TOTG	Time-Optimal Trajectory Generation
TOPPRA	Time-Optimal Path Parameterization via Reachability Analysis
TrajOpt	Trajectory Optimization
TRRT	Transition-based Rapidly-exploring Random Tree
UR	Universal Robots
URDF	Unified Robot Description Format

La motivación es lo que te pone en marcha, el hábito es lo que hace que sigas.

Jim Ryun

CAPÍTULO

1

Introducción

Contenido del Capítulo

La creciente demanda de soluciones robóticas flexibles, adaptables y capaces de operar en entornos industriales dinámicos ha impulsado una evolución significativa en los enfoques de planificación de trayectorias. Tradicionalmente, la planificación geométrica se ha abordado mediante técnicas basadas en muestreo o en optimización, con buenos resultados en entornos estáticos y bajo condiciones controladas. Sin embargo, estos enfoques encuentran limitaciones en entornos industriales donde el determinismo y la replicabilidad son cruciales, y donde las condiciones pueden cambiar rápidamente. En este contexto, la planificación de trayectorias para manipuladores robóticos industriales se enfrenta a retos significativos, como la necesidad de generar trayectorias eficientes y seguras que puedan adaptarse a condiciones cambiantes, la integración de restricciones cinemáticas y dinámicas, y la capacidad de operar en tiempo real con metas definidas dinámicamente.

Este capítulo introduce los fundamentos conceptuales y el marco de referencia sobre el cual se construye toda la investigación de esta tesis doctoral. Dado que constituye la apertura del manuscrito, su propósito es doble: por un lado, contextualizar los desafíos actuales de la robótica industrial en el marco de la Industria 4.0, y por otro, presentar la hipótesis central, los objetivos de investigación y la metodología que guiarán el desarrollo del trabajo. De este modo, el capítulo establece las bases teóricas, técnicas y metodológicas necesarias para comprender la motivación y la orientación del estudio.

En primer lugar, se aborda la motivación y el contexto, analizando cómo la transición hacia sistemas de producción más flexibles, adaptativos y personalizados demanda nuevas aproximaciones a la planificación de trayectorias en manipuladores industriales. Se destacan las limitaciones de los métodos clásicos de planificación, cuya rigidez contrasta con la creciente necesidad de adaptabilidad en entornos dinámicos. Este análisis pone de manifiesto la pertinencia de explorar alternativas basadas en técnicas de inteligencia artificial, en particular, el aprendizaje por refuerzo profundo, como vía para superar dichas limitaciones.

Posteriormente, se formula la hipótesis de investigación, que plantea la posibilidad de diseñar un proceso general basado en aprendizaje por refuerzo pro-

fundo capaz de generar políticas deterministas, modulares y escalables para la planificación geométrica de trayectorias. A partir de esta hipótesis, se definen tanto el objetivo principal como un conjunto de objetivos específicos que orientan el desarrollo del trabajo. Estos objetivos abarcan desde la caracterización del problema y la construcción del entorno experimental hasta la validación en robots reales, garantizando una cobertura completa de las fases de análisis, diseño, experimentación y evaluación.

A continuación, se expone la metodología de investigación, concebida como un proceso iterativo y reflexivo. Este enfoque metodológico se apoya en ciclos sucesivos de exploración, desarrollo y evaluación experimental, que permiten afinar progresivamente las decisiones de diseño en función de los resultados obtenidos. Además, se enfatiza la diseminación temprana de resultados parciales en foros científicos, lo que no solo aporta retroalimentación externa, sino que refuerza la relevancia de la línea de trabajo.

Finalmente, se presentan las contribuciones científicas principales, que incluyen comparaciones sistemáticas entre métodos clásicos y modernos, el desarrollo de un proceso de entrenamiento general y modular para políticas deterministas, y el análisis detallado de técnicas de aceleración del aprendizaje y de generalización. Estas aportaciones consolidan la validez del enfoque y lo posicionan como una alternativa robusta y aplicable a los retos de la automatización industrial contemporánea.

La estructura del capítulo es la siguiente: en la Sección 1.1 se presenta la motivación y el contexto de la investigación; en la Sección 1.2 se plantea la hipótesis junto con el objetivo principal y los objetivos específicos; la Sección 1.3 describe el marco metodológico seguido durante la investigación; y finalmente, en la Sección 1.4 se enumeran las contribuciones científicas alcanzadas. La Sección 1.5, que cierra el capítulo, introduce la organización general de la memoria.

1.1 Motivación y contexto

La industria manufacturera actual, impulsada por el paradigma de la Industria 4.0, exige sistemas robóticos cada vez más flexibles y adaptables a dinámicas cambiantes del mercado (Lu, 2017). La visión de una producción personalizada hasta lotes unitarios ha dejado de ser teórica y se acerca a la realidad gracias al advenimiento de robots inteligentes y otras tecnologías avanzadas (Monostori et al., 2016). En este contexto, los robots industriales deben reconfigurarse rápida y eficazmente para acomodar productos y tareas variadas, incluso cuando los objetivos de dichas tareas no se conocen hasta el momento de la operación. Por ejemplo, en entornos de ensamblado flexible, logística de almacenes o colaboración humano-robot, la meta específica (la posición de una pieza a montar, el destino de un objeto a colocar o la interacción con un operario) puede definirse en tiempo de ejecución y no antes. La capacidad de adaptarse a objetivos definidos sobre la marcha se ha vuelto, por tanto, un requerimiento clave para la próxima generación de sistemas de automatización industrial (Sultanov et al., 2022).

Aun cuando los robots han demostrado ser altamente efectivos en tareas repetitivas bajo condiciones predecibles (típico en la producción en serie tradicional), adaptar un sistema robótico a escenarios cambiantes sigue siendo un desafío importante. El creciente número de variantes de producto y la tendencia hacia lotes de fabricación más pequeños demandan células robotizadas más flexibles tanto a nivel de hardware como de software. Sin embargo, la reprogramación y reconfiguración de dichas células ante cambios en el proceso sigue siendo un proceso lento, no estandarizado y que requiere especialistas en robótica industrial. En la práctica, los programadores deben anticipar todas las posibles variaciones y codificarlas por adelantado, lo cual limita la flexibilidad del sistema a un conjunto finito de casos predefinidos (Sultanov et al., 2022). Este enfoque rígido resulta poco viable cuando las tareas y sus metas pueden cambiar de forma impredecible. De hecho, en operaciones complejas como el ensamblado industrial, la falta de adaptabilidad en la automatización actual conlleva que muchas tareas todavía dependan en gran medida de operadores humanos, quienes aportan la destreza y flexibilidad necesarias para manejar la variabilidad del

entorno (Monostori et al., 2016). Esto evidencia la necesidad de nuevas técnicas de control y planificación más versátiles, que permitan a los robots asumir tareas con objetivos dinámicos, reduciendo la intervención humana y los tiempos muertos en entornos de producción cambiantes.

La planificación de trayectorias para manipuladores industriales de 6 Grado de Libertad (GDL) es un problema fundamental en robótica: consiste en generar una secuencia de configuraciones articulares que lleve al robot desde un estado inicial hasta un estado meta, evitando colisiones y respetando las restricciones cinemáticas. Durante décadas, este problema se ha abordado con éxito mediante técnicas de planificación clásicas, apoyadas en modelos geométricos y algoritmos deterministas o probabilísticos. Estas técnicas han demostrado su utilidad en numerosos escenarios controlados (Kavraki et al., 1996), especialmente cuando la configuración del entorno y la meta final se encuentran bien definidas a priori. No obstante, presentan limitaciones importantes en contextos dinámicos o inciertos. En general, los planificadores tradicionales suponen entornos estáticos: típicamente funcionan apoyados en un mapa fijo del espacio de trabajo y dividen la planificación en subproblemas (planificación global y local por separado), lo que dificulta su adaptación a entornos no estructurados o con cambios súbitos. En situaciones donde la meta se especifica sobre la marcha o donde el entorno varía, estos algoritmos deben recomputar la ruta desde cero ante cada cambio significativo, careciendo de mecanismos para incorporar experiencia previa o aprender de iteraciones pasadas. En otras palabras, sus capacidades de reacción y aprendizaje son muy limitadas: no poseen inteligencia adaptativa, ofrecen pobre desempeño en planificación dinámica y no aprovechan retroalimentación para mejorar sucesivamente (Lavalle, 2006). Estas limitaciones reducen la eficacia de los métodos clásicos cuando se requiere una planificación reactiva en tiempo real frente a objetivos y condiciones cambiantes.

En años recientes, los avances en aprendizaje automático han propiciado un cambio de paradigma en la planificación robótica, liderado por el aprendizaje por refuerzo profundo (DRL). A diferencia de los métodos convencionales, un agente de DRL aprende estrategias de movimiento mediante la experimentación iterativa: a través de prueba y error, ajusta sus acciones en función de una señal

de recompensa, desarrollando así una política capaz de generalizar a situaciones no vistas y de adaptarse a variaciones del entorno en tiempo real (Sutton and Barto, 2018). Este enfoque basado en los datos integra en una sola arquitectura el proceso de decisión global y local del planificador, optimizando el movimiento gracias a la retroalimentación continua del entorno (Wang et al., 2020). La capacidad de los agentes de DRL para adaptarse a entornos cambiantes, especialmente aquellos con obstáculos dinámicos o metas móviles, ha sido un factor decisivo en su adopción creciente para tareas de planificación en robótica (Tai et al., 2017). De hecho, presentan un comportamiento significativamente más reactivo y flexible que otras técnicas de control automático, lo que ha impulsado un marcado aumento de la investigación en este ámbito durante la última década. Desde aproximadamente 2017 en adelante se observa un auge en las publicaciones científicas sobre DRL aplicado a la planificación de trayectorias, coincidiendo con la madurez de las redes neuronales profundas y su integración con algoritmos de refuerzo, lo cual ha permitido abordar entornos y problemas mucho más complejos de lo que era posible previamente (Sultanov et al., 2022). Este auge del DRL en robótica pone de manifiesto el interés tanto académico como industrial por desarrollar sistemas de control más autónomos y adaptativos.

Cabe destacar que muchas investigaciones iniciales validan sus métodos de planificación con DRL en escenarios simplificados sin obstáculos externos, centrándose únicamente en las restricciones intrínsecas del robot (evitar la autocolisión entre eslabones y la colisión con el entorno inmediato, como el suelo). Este enfoque controlado de validación permite aislar el desempeño del algoritmo de planificación sin la incertidumbre añadida de obstáculos impredecibles, facilitando el análisis de la solución aprendida. Siguiendo esa línea, el presente trabajo se centra en la planificación de trayectorias para un manipulador industrial de 6 grados de libertad en un entorno despejado de obstáculos externos, considerando como restricciones principales la prevención de colisiones del robot consigo mismo y con el suelo. La meta a alcanzar no se proporciona hasta el momento de la ejecución, reflejando así casos de uso como el ensamblado flexible de distintas piezas, la preparación de pedidos en logística, o ciertas tareas colaborativas, en los cuales el robot debe generar rutas factibles hacia objetivos que van variando sobre la marcha. Este escenario pone a prueba la capacidad

de adaptación del planificador: el desafío y motivación central es explorar cómo las técnicas modernas basadas en DRL pueden superar las limitaciones de los métodos clásicos en contextos de meta dinámica, habilitando robots industriales más autónomos y adaptativos frente a la incertidumbre en la definición de sus objetivos.

1.2 Hipótesis y objetivos

El presente trabajo parte de la premisa de que los métodos tradicionales de planificación geométrica presentan limitaciones relevantes en entornos industriales donde la meta debe definirse dinámicamente durante la ejecución. En este contexto, se plantea la siguiente hipótesis de investigación como base conceptual del desarrollo:

> *Es posible diseñar un proceso general basado en aprendizaje por refuerzo profundo que permita obtener una política determinista capaz de resolver la planificación geométrica de trayectorias para manipuladores robóticos industriales, alcanzando cualquier meta definida en tiempo de ejecución, y que pueda ser aplicada de forma modular y escalable a diferentes tareas de planificación.*

Esta hipótesis descansa sobre tres propiedades fundamentales que caracterizan el proceso que se pretende obtener: el determinismo, la modularidad y la escalabilidad. En primer lugar, un proceso determinista es aquel que, ante una misma configuración del sistema, siempre produce la misma acción como salida. Esta propiedad es especialmente deseable en robótica industrial, donde la repetibilidad y la previsibilidad del comportamiento son claves para garantizar la seguridad y la integración con otros procesos. Esto es especialmente relevante en entornos industriales reales, donde la lectura de sensores puede provocar pequeñas variaciones en el estado del sistema, pero el proceso debe ser capaz de generar una acción consistente y predecible. En segundo lugar, la modularidad implica que el resultado del proceso puede ser utilizado como componente elemental dentro de estructuras de planificación más amplias. Por ejemplo, si

es capaz de generar un movimiento eficiente hacia una meta puntual, entonces puede integrarse en un planificador jerárquico que divida una trayectoria compleja en una secuencia de metas intermedias. Finalmente, la escalabilidad se refiere a la posibilidad de aplicar el mismo proceso a manipuladores distintos o a tareas con restricciones adicionales, sin necesidad de rediseñar completamente el sistema. Esto permitiría, por ejemplo, reutilizar el proceso en un brazo robótico distinto siempre que se respeten ciertos principios de representación del entorno y del espacio de acción.

El objetivo principal de esta tesis es desarrollar un proceso de entrenamiento general, basado en técnicas de aprendizaje por refuerzo profundo, que permita obtener una política determinista capaz de resolver la planificación geométrica de manipuladores robóticos articulados. Esta política deberá ser capaz de alcanzar metas definidas en tiempo de ejecución, cubriendo de forma robusta todo el espacio de trabajo, y concebida para ser escalable, modular y reutilizable como bloque base dentro de planificadores de trayectorias más complejos.

A partir del objetivo general planteado, esta tesis se estructura en torno a un conjunto de objetivos específicos que guían cada una de las etapas del trabajo. Estos objetivos abordan desde la caracterización del problema y el diseño del entorno de aprendizaje, hasta la validación experimental de la política resultante y su comparación con métodos clásicos. Cada uno de ellos contribuye al desarrollo progresivo del enfoque propuesto y a la evaluación rigurosa de su aplicabilidad, robustez y generalización.

OE1. Analizar las limitaciones de los métodos clásicos de planificación geométrica, basados en muestreo y optimización local, en contextos donde las metas se definen en tiempo de ejecución, identificando sus principales cuellos de botella en términos de adaptabilidad, eficiencia y reutilización.

OE2. Formular el problema de planificación geométrica como un entorno de aprendizaje por refuerzo profundo, definiendo una representación abstracta y general del estado y la meta, un espacio de acción aplicable a distintos manipuladores, y una función de recompensa que incentive la convergencia rápida, la seguridad y la suavidad del movimiento.

OE3. Diseñar y entrenar una política determinista y generalizable mediante técnicas de aprendizaje por refuerzo profundo.

OE4. Validar empíricamente la capacidad de generalización de la política en escenarios no vistos durante el entrenamiento, verificando su capacidad para alcanzar metas arbitrarias distribuidas por el espacio de trabajo y para operar bajo restricciones articulares y estructurales realistas.

OE5. Evaluar la modularidad y escalabilidad del enfoque propuesto, aplicando la política como bloque funcional dentro de planificadores jerárquicos y tareas multimodales.

OE6. Comparar el rendimiento del enfoque con técnicas clásicas de planificación tradicionales, utilizando métricas adecuadas.

OE7. Validar la aplicabilidad de la política en un entorno real mediante su ejecución en un robot físico, evaluando la capacidad de transferencia *sim-to-real*.

1.3 Metodología de investigación

La metodología de esta tesis sigue un enfoque iterativo y reflexivo, estructurado en torno a ciclos de exploración, desarrollo y evaluación experimental. Este enfoque permite generar conocimiento aplicable de forma progresiva, revisando y mejorando el diseño en función de los resultados obtenidos, hasta alcanzar un nivel de madurez suficiente para su diseminación científica.

La Figura 1.1 resume el proceso metodológico seguido, que se articula en las siguientes fases:

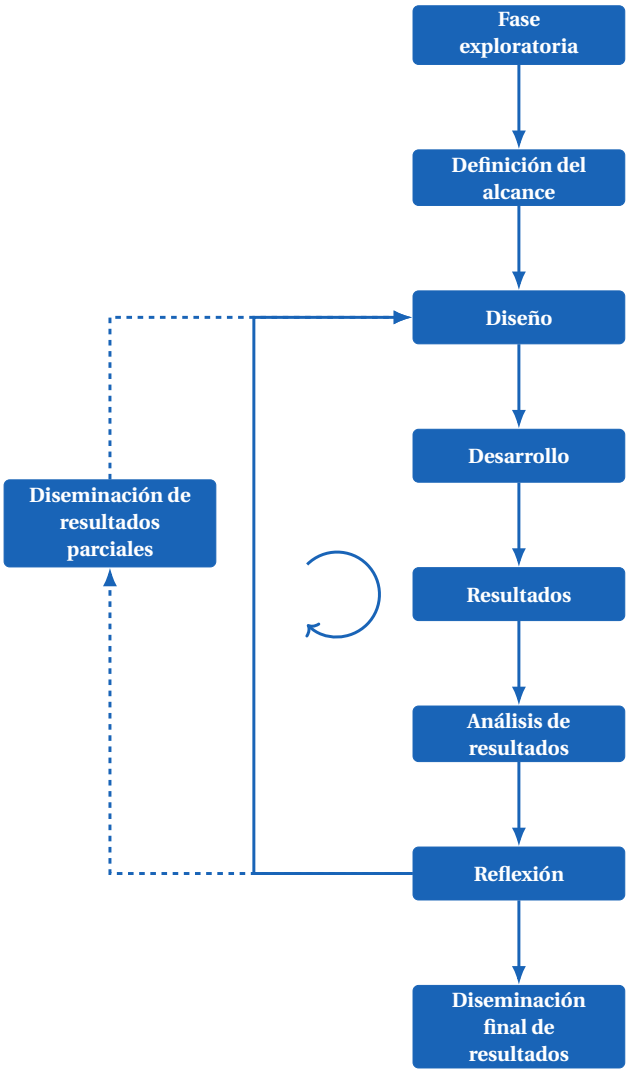

Figura 1.1: Representación del ciclo metodológico seguido.

El trabajo se inició con una fase de exploración conceptual, orientada a comprender las limitaciones actuales en la planificación de trayectorias para manipuladores industriales. Esta etapa incluyó una revisión crítica del estado del arte, que permitió identificar las limitaciones de los enfoques tradicionales. Como resultado de esta revisión, se definió con claridad el alcance de la investigación, formulando una hipótesis de partida centrada en el diseño de un proceso de planificación generalizable.

Una vez definidos los objetivos, se inició el diseño y desarrollo de un entorno de simulación que sirviera como banco de pruebas para validar, de forma progresiva, el proceso propuesto. Durante esta fase se evaluaron distintas plataformas y alternativas de implementación, explorando múltiples configuraciones de espacio de observación, espacio de acción, funciones de recompensa y estructuras de entrenamiento. Este enfoque permitió analizar de forma iterativa el comportamiento del sistema bajo diferentes estrategias de modelado, configuraciones generales y variantes de entrenamiento, con el fin de identificar aquellas combinaciones más prometedoras.

Tras varias iteraciones de mejora, se obtuvieron resultados parciales que permitieron evaluar en detalle el comportamiento del sistema en distintas configuraciones y situaciones. Esta evaluación se apoyó en criterios objetivos, como la capacidad de alcanzar metas distribuidas en el espacio de trabajo o la suavidad de las trayectorias generadas, así como en pruebas prácticas de validación en el robot real. Los resultados obtenidos se analizaron con detalle, identificando fortalezas, limitaciones y posibles líneas de mejora.

Durante este proceso iterativo, algunos de los resultados parciales obtenidos fueron suficientemente relevantes y estables como para ser compartidos con la comunidad científica. Estos avances intermedios permitieron generar contribuciones publicadas en congresos, facilitando el contraste con otros enfoques, recibiendo retroalimentación externa y enriqueciendo el desarrollo posterior del sistema. La diseminación temprana también contribuyó a validar la pertinencia de la línea de investigación seguida y a reforzar su impacto potencial en el ámbito de la planificación robótica.

El análisis de los resultados sirvió como base para una fase de reflexión, desde la cual se reformularon parcialmente algunos aspectos del diseño inicial. En esta etapa se tomó la decisión de reforzar ciertos elementos clave del sistema, como su estabilidad frente a configuraciones no vistas y su capacidad de generalización. Esta retroalimentación condujo a nuevas fases de desarrollo y validación, manteniendo así un ciclo continuo de refinamiento progresivo.

Finalmente, una vez consolidado el conocimiento generado, se procedió a la diseminación de los resultados mediante la redacción de esta memoria doctoral y la elaboración de publicaciones científicas. Este proceso no solo ha permitido

validar empíricamente la hipótesis de partida, sino también contribuir al avance del estado del arte en la planificación flexible para robots industriales.

En conjunto, la metodología seguida ha permitido avanzar desde una concepción inicial teórica hasta una solución práctica validada, siguiendo un enfoque estructurado y reflexivo, alineado con los principios del método científico aplicado en ingeniería.

1.4 Contribuciones científicas

La presente tesis doctoral contribuye al avance del conocimiento en el campo de la planificación de trayectorias para manipuladores robóticos industriales mediante técnicas de aprendizaje por refuerzo profundo. En particular, se ha abordado el problema de generar políticas generalizables y eficientes que permitan alcanzar metas definidas en tiempo de ejecución, comparando enfoques clásicos y modernos, y analizando el impacto de distintas estrategias de entrenamiento.

Las principales contribuciones científicas de este trabajo son las siguientes:

1. **Comparación sistemática entre métodos clásicos de planificación geométrica y enfoques basados en aprendizaje por refuerzo profundo.** Se ha llevado a cabo un estudio cuantitativo y cualitativo que contrasta planificadores tradicionales (basados en muestreo y optimización) con políticas entrenadas mediante aprendizaje por refuerzo. Este análisis ha permitido identificar fortalezas y debilidades de cada enfoque en términos de eficiencia, capacidad de generalización, modularidad y aplicabilidad en tiempo de ejecución.

2. **Generación de conocimiento sobre estrategias de aceleración del aprendizaje y mejora de la generalización en entornos simulados.** A partir de múltiples experimentos controlados, se ha profundizado en cómo diferentes configuraciones del entorno, de la representación del estado y de las metas afectan a la velocidad de convergencia y a la estabilidad del entrenamiento.

3. **Desarrollo de un proceso de entrenamiento general para obtener políticas deterministas reutilizables.** Se ha diseñado y validado un proceso estructurado, basado en aprendizaje por refuerzo profundo, que permite obtener políticas deterministas capaces de resolver la planificación geométrica para manipuladores robóticos industriales, alcanzando metas arbitrarias en tiempo de ejecución. Este proceso ha sido concebido para ser modular y escalable, lo que facilita su integración en arquitecturas más complejas y su adaptación a distintas tareas.

4. **Análisis del impacto de técnicas específicas de aprendizaje profundo en la estabilidad y robustez de las políticas.** Se ha estudiado de forma detallada cómo afectan técnicas como el *curriculum learning*, el uso de experiencia experta o la regularización a la capacidad de generalización y al comportamiento de las políticas en configuraciones no vistas. Este análisis proporciona pautas prácticas para mejorar la eficacia de entrenamientos en entornos continuos de alta dimensionalidad.

5. **Evaluación de la modularidad y escalabilidad del enfoque propuesto.** Se ha demostrado que la política aprendida puede utilizarse como bloque base en planificadores jerárquicos, permitiendo la composición de trayectorias a partir de puntos objetivo encadenados. Asimismo, se ha analizado su aplicación en tareas multimodales, con restricciones específicas, y su viabilidad en manipuladores distintos al caso base, cuando las condiciones de simulación lo permiten.

En conjunto, estas contribuciones refuerzan la viabilidad del uso de aprendizaje por refuerzo profundo como alternativa competitiva a los métodos clásicos de planificación, ofreciendo un marco generalizable, eficiente y modular para su aplicación en entornos industriales reales.

1.5 Organización de la memoria

La presente memoria doctoral se estructura en siete capítulos, organizados de manera progresiva desde la contextualización del problema hasta la validación experimental y la discusión de futuras líneas de investigación. Cada capítulo aborda una fase específica del trabajo, contribuyendo de forma coherente al cumplimiento de la hipótesis y los objetivos planteados en la Sección 1.2. A continuación, se resume el contenido de cada uno:

- **Capítulo 1: Introducción.** Establece el contexto y la motivación de la investigación, presentando la hipótesis de partida, el objetivo principal y los objetivos específicos. Asimismo, expone la metodología seguida y las contribuciones científicas alcanzadas, ofreciendo una visión global del trabajo realizado.

- **Capítulo 2: Fundamentos de la planificación de trayectorias.** Recoge los fundamentos conceptuales y matemáticos necesarios para comprender la planificación geométrica de trayectorias y el aprendizaje por refuerzo profundo.

- **Capítulo 3: Planificación de trayectorias.** Analiza en detalle los métodos clásicos de planificación geométrica, tanto los basados en muestreo como los de optimización local. Se identifican sus limitaciones en contextos de objetivos definidos en tiempo de ejecución, sentando las bases para la necesidad de enfoques alternativos. Incluye una revisión crítica de la literatura, abarcando desde métodos clásicos de planificación hasta los principales algoritmos empleados en DRL.

- **Capítulo 4: Diseño del proceso de planificación.** Presenta la construcción del conjunto de datos diseñado, desarrollado y utilizado en la investigación. En este se describe la generación de un amplio conjunto de trayectorias, que sirve como referencia para el entrenamiento y la validación de las políticas.

- **Capítulo 5: Entrenamiento de la nuestra solución basada en aprendizaje por refuerzo.** Constituye el núcleo experimental de la tesis. Describe la formulación del problema, el diseño del entorno de entrenamiento, la definición de estados, acciones y recompensas, así como las técnicas de aceleración aplicadas. Además, se presenta la comparación entre algoritmos de DRL y se selecciona la política final.

- **Capítulo 6: Evaluación y validación del enfoque propuesto.** Evalúa la política propuesta en escenarios de simulación no vistos durante el entrenamiento y en el entorno real. Se incluyen métricas, así como un análisis detallado de la transferencia *sim-to-real*. La validación cubre distintos modos de ejecución, demostrando la robustez del enfoque.

- **Capítulo 7: Conclusiones y líneas futuras.** Integra los resultados alcanzados y valida la hipótesis de investigación. Se discuten los objetivos cumplidos, se resumen las contribuciones principales y se proponen las líneas de trabajo futuro. Este capítulo cierra la memoria, destacando la proyección del trabajo hacia nuevas oportunidades de investigación y aplicación industrial.

En conjunto, esta estructura ofrece una progresión lógica y coherente desde los fundamentos conceptuales hasta la validación práctica en un robot real, asegurando que cada capítulo contribuya de manera explícita a la construcción y validación del enfoque propuesto.

La formulación adecuada de un problema es más importante que su solución.

Albert Einstein

CAPÍTULO

2

Fundamentos de la planificación de trayectorias

Contenido del Capítulo

ESTE CAPÍTULO desarrolla los fundamentos necesarios para comprender y modelar matemáticamente a los manipuladores robóticos industriales, estableciendo el puente entre la descripción geométrica de sus componentes y la formulación abstracta utilizada en planificación de trayectorias. El objetivo es presentar de manera estructurada los conceptos espaciales, cinemáticos y computacionales que constituyen la base de cualquier sistema de control y planificación en robótica manipuladora.

En primer lugar, se introducen los conceptos espaciales esenciales en robótica, como la representación de posición y orientación, el espacio de trabajo y el espacio de configuración. Estas nociones permiten describir rigurosamente tanto la localización del efector final como el conjunto de configuraciones admisibles del robot, distinguiendo entre regiones alcanzables, libres de colisión y objetivos factibles. A continuación, se aborda la definición y clasificación de los manipuladores, considerando aspectos como el número de grados de libertad, la redundancia, la destreza y la relación entre diseño mecánico y capacidad operativa.

Posteriormente, se introduce el modelado cinemático, que constituye el núcleo matemático del análisis de manipuladores. Se estudian tanto la cinemática directa, basada en transformaciones homogéneas y en la parametrización sistemática mediante la convención de Denavit–Hartenberg, como la cinemática inversa, cuyo carácter no lineal y frecuentemente no unívoco plantea retos analíticos y computacionales de gran relevancia.

Finalmente, el capítulo conecta estos fundamentos con la simulación y ejecución de trayectorias en robots reales. Se introducen las herramientas de software que permiten validar trayectorias en entornos virtuales y su traducción posterior a comandos ejecutables en hardware, estableciendo así el ciclo completo que une el modelado teórico con la práctica experimental.

La estructura del capítulo es la siguiente: en la Sección 2.1 se presentan los conceptos espaciales básicos en robótica y la clasificación de manipuladores según sus grados de libertad y capacidades. La Sección 2.1.5 aborda el modelado cinemático directo e inverso, así como el análisis de singularidades y redundancias. Finalmente, en la Sección 2.2 se introducen los fundamentos matemáticos

que enlazan el modelado geométrico con la planificación de trayectorias, sirviendo de base para los capítulos experimentales posteriores.

2.1 Modelado de manipuladores robóticos

Un manipulador robótico es un sistema mecánico articulado compuesto por una cadena de eslabones unidos mediante articulaciones móviles, cuyo objetivo es posicionar y orientar un efector final dentro de un entorno tridimensional (Spong et al., 2006). La mayoría de los manipuladores industriales tienen estructura de cadena abierta, donde un extremo está anclado a una base fija y el otro corresponde al efector final.

2.1.1 Grados de libertad

El número de GDL de un manipulador es el número de parámetros independientes necesarios para definir su configuración. En general, para alcanzar una pose arbitraria en el espacio tridimensional se requieren al menos 6 GDL. Muchos manipuladores modernos implementan precisamente 6, pero otros sistemas incorporan GDL adicionales (manipuladores redundantes) para mejorar la flexibilidad, evitar obstáculos o cumplir tareas secundarias.

El diseño del manipulador influye directamente en aspectos como:

- **Destreza**: Capacidad para generar movimientos suaves y precisos en el entorno.
- **Redundancia**: Capacidad de cumplir una tarea de múltiples formas, lo que permite optimización secundaria (evitar colisiones, minimizar energía, etc.).
- **Rigidez y carga útil**: Relacionados con la estructura mecánica y los actuadores utilizados.

2.1.2 Representación de posición y orientación

La representación precisa de la posición y orientación de cuerpos rígidos en el espacio tridimensional es fundamental en robótica, especialmente en el análisis y control de manipuladores. En términos generales, la pose de un cuerpo rígido se define como una combinación de su posición y orientación con respecto a un sistema de referencia. Esta información es necesaria para describir la localización de cada eslabón del manipulador, así como del efector final con respecto a su base.

2.1.2.1 Representación de la posición

La posición de un punto en el espacio tridimensional se representa mediante un vector columna en coordenadas cartesianas:

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \tag{2.1}$$

Este vector indica la localización del origen de un sistema de coordenadas móvil (por ejemplo, asociado al efector final) con respecto a un sistema de referencia fijo (normalmente el sistema base del manipulador).

2.1.2.2 Representación de la orientación

La orientación de un sistema de referencia móvil respecto a uno fijo se puede representar de diversas formas. A continuación, se presentan las más comunes:

a) **Matriz de rotación**: Una orientación puede representarse mediante una matriz ortogonal 3×3 con determinante 1:

$$R \in SO(3) = \left\{ R \in \mathbb{R}^{3\times 3} \mid R^T R = I,\ \det(R) = 1 \right\} \tag{2.2}$$

Esta matriz transforma vectores expresados en el sistema móvil al sistema fijo y viceversa. Además, cada columna de R representa un eje del sistema

móvil en coordenadas del sistema fijo. Las matrices de rotación son ampliamente utilizadas en cinemática debido a su simplicidad algebraica y su integración directa en transformaciones homogéneas.

b) **Ángulos de Euler**: Alternativamente, la orientación puede expresarse mediante una secuencia de tres rotaciones elementales sobre ejes coordenados. Existen múltiples convenciones (por ejemplo, ZYX, ZYZ, XYZ, etc.), lo que puede inducir ambigüedades si no se especifica claramente la convención adoptada.

   Para la convención Z-Y-X (rotación primero sobre z, luego sobre y, y finalmente sobre x), la matriz de rotación resultante es:

$$R = R_x(\theta_3)\, R_y(\theta_2)\, R_z(\theta_1) \tag{2.3}$$

   Donde θ_1, θ_2 y θ_3 son los ángulos de rotación y R_x, R_y, R_z son las matrices de rotación elemental alrededor de los ejes correspondientes. Aunque esta representación es intuitiva, sufre de singularidades cinemáticas, particularmente cuando dos ejes de rotación se alinean, lo que da lugar al fenómeno de *gimbal lock* (Craig, 2004).

c) **Cuaterniones**: Los cuaterniones unitarios proporcionan una representación compacta, continua y libre de singularidades de la orientación. Un cuaternión unitario se define como:

$$q = q_0 + q_1 i + q_2 j + q_3 k \in \mathbb{H} \tag{2.4}$$

   Donde los coeficientes reales (q_0, q_1, q_2, q_3) satisfacen:

$$\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{2.5}$$

   La conversión de un cuaternión unitario a una matriz de rotación se realiza mediante:

$$R(q) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{2.6}$$

Los cuaterniones son especialmente útiles en simulación y control, pues permiten interpolación suave de orientaciones (*slerp*) y evitan ambigüedades inherentes a los ángulos de Euler (Siciliano et al., 2009).

2.1.2.3 **Transformaciones homogéneas**

Para combinar posición y orientación en una sola estructura algebraica, se utilizan las matrices de transformación homogénea 4×4:

$$T = \begin{bmatrix} R & p \\ 0^T & 1 \end{bmatrix} \in SE(3) \tag{2.7}$$

Donde $R \in SO(3)$ representa la rotación y $p \in \mathbb{R}^3$ la traslación. Estas matrices pertenecen al grupo especial euclídeo $SE(3)$ y permiten encadenar transformaciones mediante multiplicación matricial. Su uso es estándar en cinemática directa e inversa, planificación de trayectorias y simulación de movimientos en espacios tridimensionales.

2.1.3 **Espacio de configuración**

El espacio de configuración, denotado como C, es una abstracción fundamental en planificación de movimientos para robots manipuladores. Se define como el conjunto de todas las configuraciones posibles que puede adoptar el sistema robótico, es decir, todas las combinaciones de valores articulares que determinan su postura completa (Lavalle, 2006).

En el caso de un manipulador de n grados de libertad, cada configuración puede representarse como un vector:

$$q = [q_1, q_2, \ldots, q_n] \in \mathbb{R}^n \tag{2.8}$$

Donde cada q_i representa el valor de la i-ésima articulación (angular para articulaciones rotacionales, lineal para prismáticas). Por tanto, el espacio de configuración completo se modela como un subconjunto:

$$C \subseteq \mathbb{R}^n \tag{2.9}$$

Este espacio puede presentar una geometría compleja, especialmente cuando se incorporan restricciones físicas, límites articulares, y posibles colisiones con el entorno o consigo mismo.

Dentro de C, se definen tres subconjuntos fundamentales para la planificación de trayectorias:

- C_{obs}: Conjunto de configuraciones en las que alguna parte del manipulador colisiona con un obstáculo del entorno o consigo mismo.
- C_{free}: Conjunto de configuraciones válidas, libres de colisiones, en las que el manipulador puede operar de forma segura:

$$C_{\text{free}} = C \setminus C_{\text{obs}} \tag{2.10}$$

- C_{goal}: Conjunto de configuraciones que satisfacen el objetivo deseado (por ejemplo, alcanzar una posición y orientación específicas del efector final).

La dificultad de modelar C_{obs} crece significativamente con el número de grados de libertad del manipulador y la complejidad del entorno. Por ello, muchos algoritmos utilizan representaciones implícitas o aproximadas, como mapas ocupacionales.

La mayoría de los algoritmos de planificación modernos, como RRT (Rapidly-exploring Random Trees) o PRM (Probabilistic Roadmaps), operan directamente en C_{free}, debido a que este espacio evita explícitamente las configuraciones inválidas. La planificación consiste entonces en encontrar una curva continua:

$$\tau : [0,1] \rightarrow C_{\text{free}}, \quad \text{con } \tau(0) = q_{\text{start}},\ \tau(1) = q_{\text{goal}} \tag{2.11}$$

La factibilidad de esta trayectoria depende de la continuidad, la ausencia de colisiones y el cumplimiento de restricciones adicionales (como límites dinámicos o cinemáticos).

2.1.4 Espacio de trabajo

En robótica manipuladora, el espacio de trabajo (*workspace*) es el subconjunto del espacio euclídeo tridimensional $\mathbb{R}^3$ que puede ser alcanzado por el efector final del manipulador. Este concepto resulta esencial para la planificación de movimientos, la validación de trayectorias y el análisis de la capacidad operativa del robot dentro de su entorno.

Sea $q \in \mathcal{Q}$ una configuración articular válida del manipulador, donde $\mathcal{Q} \subset \mathbb{R}^n$ representa el espacio de configuraciones articulares admisibles considerando los límites físicos de cada articulación. La función de cinemática directa $f : \mathcal{Q} \rightarrow \mathbb{R}^3 \times SO(3)$ permite calcular la pose del efector final a partir de una configuración articular:

$$x = f(q) = \begin{bmatrix} p(q) \\ R(q) \end{bmatrix} \tag{2.12}$$

Donde $p(q) \in \mathbb{R}^3$ representa la posición y $R(q) \in SO(3)$ la orientación del efector final. El espacio de trabajo, denotado como $\mathcal{W}$, se define como el conjunto de todas las posiciones y orientaciones del efector final que pueden alcanzarse a través de configuraciones articulares válidas:

$$\mathcal{W} = \left\{ (p, R) \in \mathbb{R}^3 \times SO(3) \;\middle|\; \exists q \in \mathcal{C},\ f(q) = (p, R) \right\} \tag{2.13}$$

En otras palabras, $\mathcal{W}$ contiene todos los puntos del espacio a los que el efector puede llegar en al menos una orientación válida.

2.1.4.1 Clasificación del espacio de trabajo

En general, se distinguen los siguientes subconjuntos dentro del espacio de trabajo:

- **Espacio alcanzable** (*reachable workspace*): Conjunto de posiciones que el efector puede alcanzar en al menos una orientación válida.
- **Espacio diestro** (*dexterous workspace*): Subconjunto del espacio alcanzable donde el robot puede alcanzar todas las orientaciones posibles del efector final (o al menos todas las requeridas para una tarea específica).

- **Espacio operable**: Región en la que el efector puede ejecutar un conjunto específico de tareas con restricciones cinemáticas y dinámicas (por ejemplo, soldadura o ensamblaje).

2.1.4.2 Factores que afectan la forma del espacio de trabajo

La geometría, estructura y limitaciones físicas del manipulador influyen directamente en la forma y el volumen del espacio de trabajo. Algunos de los factores determinantes incluyen:

- **Tipo y disposición de articulaciones**: Por ejemplo, robots cartesianos generan un espacio cúbico, mientras que robots antropomórficos tienen un espacio más esférico o acampanado.
- **Longitud de los eslabones**: Afecta el alcance máximo del efector.
- **Límites articulares**: Imponen restricciones sobre las regiones accesibles.
- **Colisiones internas**: En algunos robots, ciertas configuraciones válidas desde el punto de vista cinemático pueden provocar colisiones físicas entre eslabones.
- **Singularidades**: En ciertas zonas del espacio, el manipulador pierde grados de libertad efectivos, afectando su capacidad para controlar la orientación. Normalmente esto ocurre cuando dos o más ejes de rotación se alinean, lo que limita la capacidad del robot para alcanzar ciertas orientaciones (Siciliano et al., 2009).
- **Herramientas y accesorios**: La forma y tamaño del efector final también influyen en el espacio de trabajo efectivo, ya que pueden limitar la capacidad de alcanzar ciertos puntos o realizar tareas específicas.

2.1.4.3 Visualización del espacio de trabajo

La representación gráfica del espacio de trabajo es una herramienta útil para el análisis de la capacidad de cobertura del robot. Se suele estimar mediante simulaciones, discretizando el espacio articular y evaluando la función de cinemática

directa para cada muestra. En algunos casos, se emplean mapas de densidad para visualizar la frecuencia con la que se alcanza cada punto del espacio, o nubes de puntos para visualizar el volumen operativo.

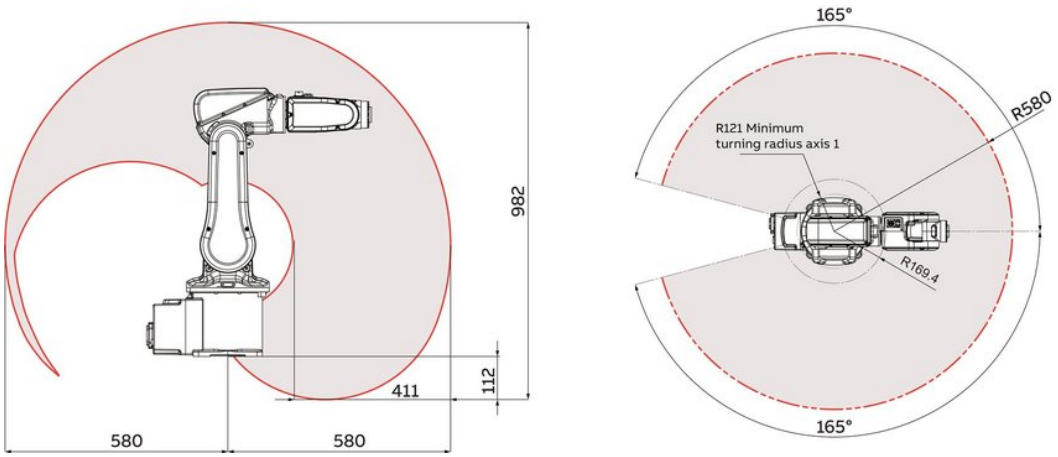

Figura 2.1: Visualización del espacio de trabajo del manipulador industrial ABB IRB120. (Bhatt et al., 2019)

2.1.5 Modelado cinemático de manipuladores

El modelado cinemático de un manipulador robótico consiste en establecer la relación matemática entre los valores articulares del robot y la posición y orientación de su efector final. Este análisis constituye la base para tareas como control, planificación de trayectorias, simulación y análisis de singularidades.

Existen dos problemas fundamentales en cinemática: la cinemática directa, que computa la pose del efector a partir de una configuración articular dada, y la cinemática inversa, que resuelve los valores articulares requeridos para alcanzar una determinada pose.

2.1.5.1 Cinemática directa

La cinemática directa (Forward Kinematics (FK)) permite calcular la pose del efector final, representada por una transformación homogénea ${}^0T_n$, a partir de los valores articulares $q = [q_1, q_2, \ldots, q_n]^T$. Para ello, se modela la cadena cine-

mática del manipulador mediante transformaciones homogéneas (sección 2.1.2.3) sucesivas entre eslabones.

El método más comúnmente utilizado para este propósito es la convención de parámetros Denavit-Hartenberg (DH) (Craig, 2004), que consiste en asignar un sistema de coordenadas a cada eslabón de forma sistemática. Cada articulación se describe mediante cuatro parámetros:

- θ_i: Ángulo de rotación alrededor del eje z_{i-1} (variable si la articulación es rotacional).
- d_i: Desplazamiento a lo largo de z_{i-1} (variable si la articulación es prismática).
- a_i: Longitud del eslabón medida sobre el eje x_i.
- α_i: Ángulo entre los ejes z_{i-1} y z_i, medido alrededor de x_i.

La transformación homogénea de un eslabón i al anterior se expresa como:

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

La pose total del efector respecto al sistema base se obtiene como el producto encadenado:

$$^0T_n = A_1 A_2 \dots A_n = \prod_{i=1}^{n} A_i \tag{2.15}$$

Este modelo permite calcular la posición p y orientación R del efector final como:

$$^0T_n = \begin{bmatrix} R & p \\ 0^T & 1 \end{bmatrix} \tag{2.16}$$

De la misma manera, se pueden obtener las coordenadas de cualquier eslabón intermedio i respecto al sistema base 0:

$$^0T_i = A_1 A_2 \dots A_i = \prod_{j=1}^{i} A_j \tag{2.17}$$

2.1.5.2 **Cinemática inversa**

El problema de la cinemática inversa (Inverse Kinematics (IK)) consiste en encontrar los valores articulares $q = [q_1, q_2, \ldots, q_n]^T$ que permiten al efector final alcanzar una pose deseada en el espacio operacional, definida por una transformación homogénea objetivo:

$$T_d = \begin{bmatrix} R_d & p_d \\ 0^T & 1 \end{bmatrix} \in SE(3) \tag{2.18}$$

Desde un punto de vista funcional, este problema puede plantearse como la operación inversa a la cinemática directa. Sin embargo, a diferencia de la cinemática directa, que está bien definida para cualquier $q \in \mathbb{R}^n$, la operación inversa no puede formularse, en general, como una función matemática en sentido estricto. Esto se debe a varias razones fundamentales:

- **No inyectividad**: Para manipuladores no redundantes ($n = 6$), puede haber múltiples configuraciones q que produzcan exactamente la misma pose T_d, es decir:

  $$f(q_1) = f(q_2) = T_d \quad \text{con } q_1 \neq q_2 \tag{2.19}$$

  Esto implica que $f^{-1}(T_d)$ no está unívocamente definido.

- **No sobreyectividad**: Existen poses en $SE(3)$ que no son alcanzables por el manipulador debido a restricciones geométricas o articulares. En estos casos, no existe ningún q tal que $f(q) = T_d$.

- **No linealidad**: La función $f(q)$ involucra productos de senos y cosenos, lo que la hace altamente no lineal. Esto complica la resolución analítica y la estabilidad de métodos numéricos.

Así, el problema de cinemática inversa se formula como la búsqueda del conjunto de soluciones q tal que:

$$\text{Encontrar } q \in \mathcal{Q} \subseteq \mathbb{R}^n \text{ tal que } f(q) = T_d \tag{2.20}$$

donde $\mathcal{Q}$ representa el espacio de configuraciones válidas, respetando límites articulares y evitando colisiones.

La cardinalidad y la naturaleza del conjunto de soluciones depende de:

- El número de GDL del manipulador:
  - Si $n < 6$: el sistema está subdeterminado, y no siempre existe solución.
  - Si $n = 6$: puede haber varias soluciones discretas.
  - Si $n > 6$: el sistema está sobredeterminado (manipulador redundante) y existen infinitas soluciones que forman una variedad continua.
- La geometría del robot: Ciertas configuraciones permiten factorizar el problema en coordenadas independientes.

Los métodos para resolver la cinemática inversa se clasifican en dos grandes categorías:

1. **Métodos analíticos (o geométricos)**: Basados en la descomposición algebraica y trigonométrica del modelo cinemático. Estos métodos producen soluciones exactas y explícitas, pero solo son aplicables a manipuladores con estructuras particulares.

   Un caso clásico es el análisis de manipuladores con tres ejes rotacionales consecutivos que se cruzan en un punto, permitiendo desacoplar posición y orientación del efector. Este tipo de estructura fue estudiado por Pieper en su tesis doctoral (Pieper, 1968).

2. **Métodos numéricos**: Empleados para manipuladores de geometría arbitraria. Formulan el problema como una minimización del error entre la pose deseada y la generada por una configuración tentativa:

$$\min_{q \in \mathcal{Q}} \left\| f(q) - T_d \right\|^2 \tag{2.21}$$

   La norma utilizada depende de cómo se representen posición y orientación (métricas en $SE(3)$, error en cuaterniones, etc.). Entre los algoritmos comunes se incluyen:

- **Descenso del gradiente**: Ajusta q iterativamente en dirección del gradiente del error.
- **Newton-Raphson**: Utiliza la aproximación lineal basada en el Jacobiano.
- **Método de la pseudo-inversa**: Resuelve la versión diferencial del problema mediante:

$$\dot{q} = J^{\dagger}(q)\,\dot{x} \tag{2.22}$$

  Integrando $\dot{q}$ en el tiempo se obtiene una trayectoria hacia q_{sol}.

Estos métodos requieren una estimación inicial q_0 próxima a la solución, y pueden verse afectados por mínimos locales o por zonas de baja manipulabilidad (cercanas a singularidades).

En el caso de manipuladores redundantes ($n > 6$), la cinemática inversa admite soluciones infinitas. En estos casos, se pueden introducir criterios de optimización secundaria (por ejemplo, evitar obstáculos, minimizar energía o evitar singularidades) para seleccionar una solución deseable mediante técnicas como la proyección ortogonal en el espacio nulo del Jacobiano:

$$\dot{q} = J^{\dagger}(q)\,\dot{x} + \left(I - J^{\dagger}(q)J(q)\right) z \tag{2.23}$$

donde z es un vector arbitrario que permite explotar la redundancia del manipulador.

2.2 Fundamentos matemáticos de la planificación de trayectorias

2.2.1 Trayectorias

En el contexto de manipuladores robóticos, una trayectoria describe la evolución temporal de la configuración del robot o de su efector final a lo largo del tiempo. La planificación de trayectorias constituye un proceso fundamental en

la ejecución de tareas como ensamblado, soldadura, pintura o manipulación de objetos, y se vincula estrechamente con la cinemática y dinámica del sistema.

Sea C el espacio de configuración del manipulador, entonces una trayectoria en el espacio de configuración es una función continua del tiempo:

$$\tau : [0, T] \to C, \quad \tau(t) = q(t) \tag{2.24}$$

donde $q(t)$ representa el vector de configuraciones articulares en el instante de tiempo $t \in [0, T]$.

Alternativamente, una trayectoria en el espacio de trabajo es una función:

$$\gamma : [0, T] \to \mathbb{R}^3 \times SO(3), \quad \gamma(t) = (p(t), R(t)) \tag{2.25}$$

donde $p(t)$ es la posición y $R(t)$ la orientación del efector final. Esta trayectoria debe ser consistente con la cinemática del manipulador, es decir, existir $q(t)$ tal que $f(q(t)) = \gamma(t)$ para todo t.

Es fundamental distinguir entre dos tipos de trayectorias según su nivel de parametrización:

- **Trayectoria geométrica o camino (*path*)**: es la curva descrita en el espacio sin referencia explícita al tiempo. Por ejemplo, el camino que sigue el efector final en el espacio cartesiano, sin importar a qué velocidad lo recorre. Se define como:

  $$\mathscr{C} = \{q \in C \mid q = \tau(s),\ s \in [0, 1]\} \tag{2.26}$$

- **Trayectoria cinemática (*trajectory*)**: es una trayectoria parametrizada en función del tiempo, que incluye velocidad y aceleración. Es la función $\tau(t)$ utilizada para ejecutar físicamente el movimiento sobre el robot. Comúnmente se expresa simplemente como trayectoria.

  La relación entre ambas está mediada por el perfilado temporal, donde se asigna una ley de evolución $s(t)$ que transforma una trayectoria geométrica en una trayectoria cinemática: $\tau(t) = \tau(s(t))$.

2.2.2 Espacio de planificación

Las trayectorias pueden planificarse tanto en el espacio de configuración (apartado 2.1.3) como en el espacio de trabajo (apartado 2.1.4):

- **Espacio de configuración** (C): La planificación se realiza directamente sobre los valores articulares. Este enfoque evita singularidades y garantiza que las soluciones sean realizables físicamente, aunque no proporciona directamente información sobre el camino seguido por el efector final del robot.
- **Espacio de trabajo** ($\mathbb{R}^3 \times SO(3)$): Permite definir trayectorias intuitivas en el entorno operativo (por ejemplo, una línea recta en el espacio cartesiano), pero requiere resolver la cinemática inversa para su ejecución y puede llevar a trayectorias inviables debido a singularidades o límites articulares.

Una trayectoria bien definida debe cumplir varias condiciones según la tarea:

- **Continuidad**: Al menos en posición (C^0), pero deseable también en velocidad (C^1) o aceleración (C^2) para evitar movimientos abruptos.
- **Límites articulares**: Respeto de los límites articulares, de velocidad, aceleración y par impuestos por los actuadores y la estructura del robot.
- **Prevención de colisiones**: Con el entorno y con el propio robot (autocolisiones).
- **Sincronización**: Todos los actuadores del robot deben alcanzar sus posiciones finales al mismo tiempo, evitando desincronizaciones que puedan causar daños o errores en la tarea.

2.2.3 Planificación geométrica

La planificación geométrica de trayectorias es un subproblema de la planificación de movimiento en robótica, que se enfoca en encontrar una trayectoria

continua que conecte dos configuraciones dadas del robot sin considerar explícitamente el tiempo, ni las restricciones dinámicas (como velocidades, aceleraciones o fuerzas). El único criterio es la factibilidad geométrica, es decir, que el robot no colisione durante el movimiento.

Sea C el espacio de configuración del robot, y $C_{\text{free}} \subset C$ el subconjunto libre de colisiones. Dados:

- $q_{\text{start}} \in C_{\text{free}}$: Configuración inicial.
- $q_{\text{goal}} \in C_{\text{free}}$: Configuración objetivo.

El problema de planificación geométrica consiste en encontrar una trayectoria continua:

$$\tau : [0,1] \to C_{\text{free}}, \quad \tau(0) = q_{\text{start}}, \quad \tau(1) = q_{\text{goal}} \tag{2.27}$$

tal que $\tau(s)$ no cruce ninguna configuración en C_{obs} (zona de colisión).

Esta planificación se realiza típicamente en el espacio de configuración debido a que:

- Permite representar explícitamente las restricciones articulares y de colisión/autocolisión.
- La factibilidad geométrica se define de forma directa en este espacio.
- Es más general y aplicable a robots con geometría y cinemática arbitrarias.

Sin embargo, este espacio suele tener una topología compleja y dimensionalidad elevada, lo que dificulta la exploración sistemática mediante técnicas analíticas.

La planificación geométrica requiere verificar que una trayectoria propuesta τ permanezca completamente dentro de C_{free}. Esto se logra mediante un módulo de detección de colisiones, que evalúa:

- Si una configuración $q \in C$ resulta en colisión con el entorno o consigo mismo.

- Si el segmento de trayectoria entre dos configuraciones discretas atraviesa zonas de colisión.

Para ello se emplean modelos geométricos simplificados del robot (cubos, cilindros, mallas convexas) y algoritmos jerárquicos (árboles AABB) que aceleran las comprobaciones.El coste computacional del módulo de comprobación de colisiones es un factor determinante en la eficiencia de los algoritmos de planificación.

La mayoría de los algoritmos de planificación comparten una estructura modular compuesta por:

1. **Módulo de muestreo**: Genera configuraciones aleatorias o dirigidas en C.
2. **Evaluador de colisiones**: Determina si una configuración o segmento es válido.
3. **Estrategia de conexión**: Intenta unir configuraciones cercanas.
4. **Criterio de finalización**: Se detiene al encontrar un camino válido o tras agotar recursos computacionales.

Este marco general da lugar a una amplia familia de algoritmos de planificación que se abordarán en la siguiente subsección.

2.2.4 Fundamentos teóricos de los algoritmos de planificación de trayectorias

Los algoritmos de planificación de trayectorias son métodos computacionales diseñados para encontrar una secuencia continua de configuraciones válidas que conecten un estado inicial con un objetivo, evitando colisiones y respetando restricciones geométricas. Su estudio se fundamenta en geometría computacional, teoría de grafos, teoría de muestreo y análisis de complejidad.

El objetivo de estos algoritmos es resolver el problema de planificación geométrica planteado en el apartado 2.2.3, es decir, encontrar una trayectoria τ que conecte q_{start} con q_{goal} dentro del espacio de configuración libre C_{free}.

Dado que C puede ser de alta dimensionalidad (6D o más en manipuladores), los algoritmos exactos basados en representaciones explícitas del espacio de configuración son computacionalmente inviables. Por ello, se recurre a métodos de aproximación, principalmente mediante técnicas de muestreo.

2.2.4.1 **Principales algoritmos de planificación**

En la planificación de trayectorias para manipuladores robóticos, los principales enfoques están basados en técnicas de muestreo y búsqueda en grafos. Estos métodos permiten explorar el espacio de configuración de forma eficiente, evitando la necesidad de representar explícitamente todo el espacio. La alta dimensionalidad del espacio de configuración y la complejidad geométrica de los entornos hacen que algoritmos basados en búsqueda exhaustiva como A* (Hart et al., 1968) o Dijkstra (Dijkstra, 1959) sean inviables en la práctica. En su lugar, se utilizan algoritmos probabilísticos y de muestreo que permiten explorar el espacio de forma más eficiente.

Estos algoritmos de muestreo se exploran exahustivamente en el apartado 3.2, pero a continuación se ofrece una breve introducción a los conceptos clave y las principales familias de algoritmos.

Existen dos grandes familias de algoritmos de planificación basados en muestreo. Por una parte, los basados en *roadmaps*, que construyen un grafo de configuraciones libres conectadas localmente. El precursor de esta familia es el algoritmo PRM (Kavraki et al., 1996) que se explora en el apartado 3.2.1.1.1. El algoritmo 1 ilustra el funcionamiento base de esta familia. Por otra parte, los basados en árboles, que expanden árboles hacia muestras aleatorias. El algoritmo más representativo de esta familia es el RRT (LaValle, 1998), que se explora en el apartado 3.2.1.1.2. El algoritmo 2 ilustra el funcionamiento base de esta familia.

Si bien ambas familias de algoritmos son capaces de encontrar trayectorias válidas, difieren en su enfoque y aplicabilidad. Además, cada una tiene sus propias variantes y mejoras que se adaptan a diferentes escenarios y requisitos de planificación. Algunas de las características clave que se evalúan al comparar estos algoritmos incluyen:

Algoritmo 1: Probabilistic Roadmaps (PRM)

Input: Espacio de configuración C, subconjunto libre C_{free}, número de muestras N, número de vecinos k

Output: Grafo G que conecta configuraciones libres

```
1  V ← ∅ ;                    // Inicializar conjunto de nodos
2  E ← ∅ ;                    // Inicializar conjunto de aristas
3  for i ← 1 to N do
4      q_aleat ← Muestra aleatoria en C;
5      if q_aleat ∈ C_free then
6          V ← V ∪ {q_aleat};
7          N_k ← k vecinos más cercanos a q en V;
8          foreach q_vecino ∈ N_k do
9              if SEGMENTO[q_aleat, q_vecino] ⊂ C_free then
10                 E ← E ∪ {(q_aleat, q_vecino)};
11 return G = (V, E);
```

1. Determinismo: Si el algoritmo produce siempre la misma solución para las mismas entradas.

2. Complejidad computacional: Medida de la eficiencia del algoritmo en términos de tiempo y recursos necesarios.

3. Capacidad de encontrar soluciones óptimas: Si el algoritmo garantiza encontrar la mejor solución posible según un criterio de coste definido.

4. Capacidad de adaptarse a cambios en el entorno: Si el algoritmo puede replanificar o ajustar la trayectoria en tiempo real ante cambios en el entorno o en las restricciones del problema.

5. Reutilización de soluciones: Si el algoritmo puede aprovechar soluciones previamente calculadas para responder a nuevas consultas de planificación sin necesidad de recalcular todo desde cero.

6. Capacidad de manejar restricciones adicionales: Si el algoritmo puede incorporar restricciones cinemáticas, dinámicas o de otro tipo en la planificación de trayectorias.

Algoritmo 2: Rapidly-exploring Random Tree (RRT)

Input: Espacio de configuración C, subconjunto libre C_{free}, configuración inicial q_{start}, número máximo de iteraciones N
Output: Árbol T que conecta configuraciones libres

```
T ← {q_start}; // Inicializar árbol con la configuración inicial
for i ← 1 to N do
    q_aleat ← Muestra aleatoria en C;
    q_cercano ← Nodo más cercano a q_aleat en T;
    q_nuevo ← Expande desde q_cercano hacia q_aleat;
    if q_nuevo ∈ C_free then
        T ← T ∪ {q_nuevo};
        Añadir arista (q_cercano, q_nuevo) a T;
return T;
```

Esto implica que la elección del algoritmo adecuado depende del contexto específico de la tarea, las características del manipulador y las restricciones del entorno. Incluso siendo viable la incorporación de ambos enfoques en un mismo sistema, ya que cada uno puede ser más adecuado para diferentes tipos de problemas o fases del proceso de planificación.

Para estos métodos probabilísticos, la calidad y distribución de las muestras es crítica para lograr una cobertura eficiente del espacio. En la práctica, se utilizan dos enfoques principales para el muestreo: El muestreo uniforme, que genera muestras aleatorias distribuidas uniformemente en el espacio de configuración, y el muestreo guiado, que prioriza regiones de interés o zonas con alta probabilidad de éxito.

2.2.5 **Parametrización temporal de trayectorias**

Una vez resuelto el problema de planificación geométrica, ya sea mediante técnicas deterministas o probabilísticas, el resultado suele ser un camino continuo en el espacio de configuración, comúnmente descrito como una curva $\tau(s)$ con $s \in [0,1]$. Para que dicho camino sea ejecutable en un manipulador robótico, es necesario parametrizarlo temporalmente, es decir, definir cómo progresa el robot a lo largo de ese camino en función del tiempo real t.

Este proceso, también conocido como generación de perfiles temporales, transforma una trayectoria geométrica en una trayectoria completa mediante una función de reparametrización $s(t)$:

$$q(t) = \tau(s(t)), \quad \text{con} \quad t \in [0, T],\ s(t) \in [0, 1] \tag{2.28}$$

donde $s(t)$ es una función creciente que describe el avance del robot a lo largo del camino geométrico en función del tiempo. La parametrización temporal debe respetar las restricciones cinemáticas y dinámicas del manipulador, garantizando que el movimiento sea seguro y eficiente. Existen diferentes enfoques para la parametrización temporal de trayectorias. A continuación se presentan los más representativos:

2.2.5.1 **Parametrización temporal analítica**

Este enfoque genera directamente trayectorias $q(t)$ expresadas como funciones analíticas del tiempo. Estas funciones pueden derivarse, evaluarse e implementarse de forma eficiente en controladores industriales, sin necesidad de interpolación posterior. Los métodos analíticos más comunes incluyen (Siciliano et al., 2009):

- **Interpolación polinómica:** Se utilizan polinomios de orden 3 (cúbicos) o 5 (quínticos) para conectar varias configuraciones, garantizando continuidad en posición, velocidad y (en el caso quíntico) aceleración.

- **Perfiles trapezoidales de velocidad:** Trayectorias con fases definidas de aceleración constante, velocidad máxima constante, y desaceleración simétrica. Muy extendidas en entornos industriales por su simplicidad.

Estos métodos son rápidos y fáciles de implementar, pero tienen limitaciones en cuanto a la complejidad de las trayectorias que pueden representar. A partir de la trayectoria geométrica, es posible interpolar punto a punto cada configuración, generando una trayectoria temporalmente parametrizada $q(t)$ que cumple con las restricciones cinemáticas del robot.

2.2.5.2 **Parametrización temporal basada en integración**

A diferencia de los métodos analíticos, este enfoque, aun partiendo de una trayectoria geométrica $\tau(s)$, no define directamente una función $q(t)$. En su lugar, se calcula una ley temporal $s(t)$ que recorre el camino $\tau(s)$, generando así una trayectoria cinemática $q(t) = \tau(s(t))$. Alguno de los métodos más destacados en este ámbito son:

- **TOPPRA**: Genera un perfil temporal óptimo cumpliendo restricciones de velocidad y aceleración a lo largo del camino (Pham and Pham, 2018).

- **Time-Optimal Trajectory Generation (TOTG)**: Es una implementación simplificada del concepto de parametrización temporal, basada en el algoritmo de Bobrow (Bobrow et al., 1985). Se utiliza por defecto en el framework MoveIt (Kunz and Stilman, 2013).

Estos métodos permiten generar trayectorias temporales óptimas, minimizando el tiempo de ejecución mientras se cumplen las restricciones cinemáticas del robot. La parametrización temporal se realiza mediante integración numérica, calculando la evolución de $s(t)$ a partir de las restricciones impuestas. Su principal limitación es que el resultado no se expresa como una función cerrada $q(t)$, sino que requiere muestreo y discretización de la trayectoria, lo que puede aumentar la complejidad computacional.

2.2.5.3 **Optimización directa de trayectorias**

En contraste con los métodos secuenciales de planificación y posterior perfilado, los enfoques de optimización directa consideran la trayectoria completa como una variable a optimizar. Estos métodos representan la trayectoria como un conjunto de puntos de control $q_1, q_2, \ldots, q_N$ definidos sobre una malla temporal y minimizan una función de coste global sujeta a restricciones cinemáticas y dinámicas.

La optimización directa permite integrar la planificación y la parametrización temporal en un único proceso, lo que resulta en trayectorias más suaves y

eficientes. La formulación general del problema es:

$$\min_{q_1,q_2,\ldots,q_N} J(q_1,q_2,\ldots,q_N) \quad \text{tal que} \quad \begin{cases} \|\dot{q}_i\| \leq \dot{q}_{\max} \\ \|\ddot{q}_i\| \leq \ddot{q}_{\max} \\ q_i \in C_{\text{free}} \\ q_1 = q_{\text{start}},\ q_N = q_{\text{goal}} \end{cases} \tag{2.29}$$

donde J es una función de coste que puede incluir términos como distancia recorrida, suavidad de la trayectoria, tiempo total, etc. La principal limitación de estos métodos es la necesidad de una buena inicialización, ya que pueden converger a mínimos locales. Sin embargo, su capacidad para planificar trayectorias complejas y restricciones avanzadas los hace muy atractivos en aplicaciones industriales. Estas técnicas se exploran en detalle en el apartado 3.3.

2.2.6 Proceso general de planificación de trayectorias

A modo de resumen, el proceso de planificación de trayectorias para manipuladores robóticos puede dividirse en varias etapas secuenciales, cada una de las cuales transforma el problema en una representación más cercana a la implementación real. Este proceso modular permite una clara separación de responsabilidades y facilita la reutilización de componentes. Este proceso que abarca desde la especificación de los estados inicial y final hasta la obtención de una trayectoria cinemáticamente factible y lista para su ejecución en hardware. Este proceso se compone típicamente de varias etapas secuenciales, cada una de las cuales transforma el problema en una representación más cercana a la implementación real.

2.2.6.1 Definición de la tarea

La tarea se especifica mediante una configuración inicial q_{start} y una configuración objetivo q_{goal}, ambas pertenecientes al subconjunto libre de colisiones del espacio de configuración C_{free}. Alternativamente, en algunos casos se especifica como una pose deseada del efector final, en cuyo caso es necesario resolver la cinemática inversa para obtener la configuración articular correspondiente.

2.2.6.2 **Planificación geométrica**

A partir de la definición del espacio de configuración libre C_{free}, se aplica un algoritmo de planificación geométrica que genere una trayectoria continua $\tau(s)$ que conecte q_{start} y q_{goal}, tal que $\tau(s) \in C_{\text{free}} \ \forall s \in [0,1]$. Esta etapa emplea típicamente métodos de muestreo como PRM o RRT (ver apartado 2.2.4).

2.2.6.3 **Parametrización temporal**

Una vez validada geométricamente la trayectoria $\tau(s)$, se procede a su conversión en una trayectoria cinemática $q(t) = \tau(s(t))$ que incluya una evolución temporal compatible con las capacidades del robot. Este paso se explora en detalle en el apartado 2.2.5.

2.2.6.4 **Generación del perfil de control**

La trayectoria parametrizada se traduce en una secuencia de consignas que serán interpretadas por los controladores de bajo nivel del robot. Esto incluye:

- Muestreo de la trayectoria a alta frecuencia.
- Cálculo de velocidades y aceleraciones deseadas.
- Interfaz con controladores, típicamente de tipo Controlador Proporcional-Integral-Derivativo (PID) (Åström and Hägglund, 1995).

2.2.6.5 **Ejecución y realimentación**

Finalmente, la trayectoria se ejecuta sobre el manipulador físico. Durante la ejecución, se emplean sensores para supervisar la evolución del sistema y detectar posibles desviaciones. Este lazo de realimentación permite implementar estrategias de control adaptativo, detección de errores o incluso replanificación dinámica si fuese necesario.

Este esquema modular permite una clara separación de responsabilidades, favorece la reutilización de componentes y facilita la comparación entre distintas técnicas de planificación. En sistemas complejos, cada una de estas etapas puede incorporar optimizaciones, aprendizaje automático o interacción con humanos, lo cual se explora en capítulos posteriores.

2.3 Fundamentos del aprendizaje por refuerzo profundo

El Reinforcement Learning (RL) es un paradigma del aprendizaje automático en el cual un agente aprende a tomar decisiones a través de la interacción con un entorno, recibiendo retroalimentación en forma de recompensas. A diferencia del aprendizaje supervisado, donde se proporciona una etiqueta por cada entrada, en RL el agente debe descubrir por sí mismo qué acciones producen los mejores resultados a largo plazo (Sutton and Barto, 2018).

Este proceso se modela formalmente mediante un proceso de decisión de Markov (Markov Decision Process (MDP)), definido como una tupla:

$$\mathscr{M} = (\mathscr{S}, \mathscr{A}, P, R, \gamma) \tag{2.30}$$

donde:

- $\mathscr{S}$ es el conjunto de estados del entorno.
- $\mathscr{A}$ es el conjunto de acciones que puede ejecutar el agente.
- $P(s|s,a)$ es la función de transición, que define la probabilidad de pasar al estado s al ejecutar la acción a desde el estado s.
- $R(s,a)$ es la función de recompensa, que asigna una señal escalar inmediata tras ejecutar a en s.
- $\gamma \in [0,1]$ es el factor de descuento, que pondera la importancia de las recompensas futuras.

El agente interactúa con el entorno en una secuencia de pasos discretos: en cada paso t, observa el estado s_t, ejecuta una acción a_t, recibe una recompensa r_t y transita a un nuevo estado s_{t+1}. La figura 2.2 ilustra este proceso clásico de interacción. El objetivo del agente es aprender una política $\pi(a|s)$ que maximice la recompensa acumulada esperada:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2.31}$$

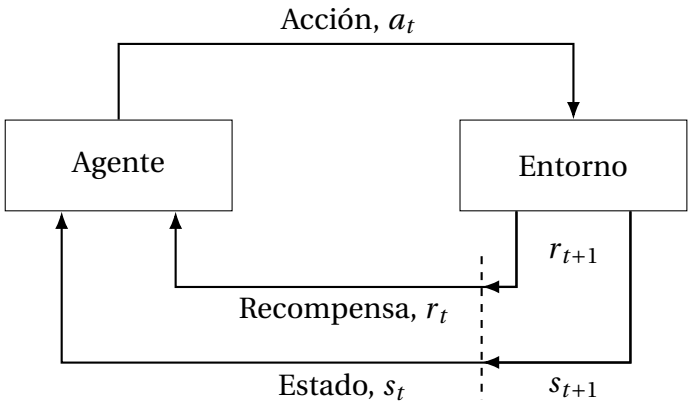

Figura 2.2: Esquema clásico de interacción entre agente y entorno en aprendizaje por refuerzo. (Sutton and Barto, 2018)

Dos funciones fundamentales guían el proceso de aprendizaje:

- **Función valor de estado**: $V^{\pi}(s) = \mathbb{E}[G_t \mid s_t = s, \pi]$
- **Función valor de acción**: $Q^{\pi}(s,a) = \mathbb{E}[G_t \mid s_t = s, a_t = a, \pi]$

El objetivo del aprendizaje es encontrar la política óptima π^* que maximice el valor esperado de retorno desde cualquier estado:

$$\pi^* = \arg\max_{\pi} V^{\pi}(s), \quad \forall s \in \mathcal{S} \tag{2.32}$$

2.3.1 Aprendizaje profundo aplicado al aprendizaje por refuerzo

El aprendizaje por refuerzo profundo (DRL) surge de la integración del aprendizaje por refuerzo clásico con redes neuronales profundas, que actúan como aproximadores de funciones para manejar espacios de estados y acciones continuos o de alta dimensión. Este enfoque ha permitido resolver tareas previamente intratables, como juegos complejos, control robótico de precisión o toma de decisiones en entornos parcialmente observables (Mnih et al., 2015a; Lillicrap et al., 2015).

El uso de redes neuronales permite aproximar funciones de valor $V^{\pi}(s)$, funciones $Q(s,a)$ o directamente políticas $\pi(a|s)$. Según el componente que se entrene mediante redes neuronales, los algoritmos se clasifican en:

- **Métodos basados en valor**: Estos métodos buscan aprender una función de valor de acción $Q(s,a)$ que estima el retorno esperado al ejecutar una acción a en un estado s, y seguir una política π posteriormente:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right] \tag{2.33}$$

  La política se obtiene de forma implícita como la acción que maximiza el valor estimado:

$$\pi(s) = \arg\max_a Q(s,a) \tag{2.34}$$

- **Métodos basados en política**: En este enfoque, se parametriza directamente la política como una función diferenciable $\pi_\theta(a|s)$, generalmente modelada como una red neuronal que genera una distribución de probabilidad sobre las acciones. El objetivo es maximizar la recompensa esperada:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{2.35}$$

  Para ello se utiliza el gradiente de política, que actualiza los parámetros en la dirección que incrementa la expectativa del retorno:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) \cdot Q^{\pi}(s,a)\right] \tag{2.36}$$

- **Métodos actor-crítico**: Los métodos actor-crítico combinan característica de ambos enfoques anteriores: mantienen dos redes separadas, una para la política (el actor) y otra para la función de valor (el crítico). El actor produce acciones $a \sim \pi_\theta(a|s)$, mientras que el crítico estima $Q^{\pi}(s,a)$ o $V^{\pi}(s)$ y guía la actualización del actor mediante gradientes más estables.

  Este enfoque permite aprendizaje continuo, eficiente en muestras, y admite tanto políticas deterministas como estocásticas. Estos métodos actor-crítico constituyen el enfoque dominante en aplicaciones robóticas, gracias a su flexibilidad, estabilidad y eficiencia en entornos complejos.

Los algoritmos de DRL han demostrado ser especialmente eficaces en tareas de control continuo, donde los espacios de acción no pueden representarse de forma discreta. Sin embargo, estos enfoques presentan retos significativos:

- **Inestabilidad en el entrenamiento**: Las redes neuronales pueden sobreajustarse o no converger debido a la correlación entre datos consecutivos. Para mitigar esto, se utilizan técnicas como la integración de *replay buffers*, el *target network* y la normalización.
- **Exploración eficiente**: En entornos de control continuo, la exploración aleatoria puede ser ineficaz. Para abordar esto, se pueden emplear estrategias de exploración guiada, como la integración de ruido a las acciones o el uso de políticas estocásticas.
- **Dependencia intensiva de datos**: Muchos algoritmos requieren millones de interacciones con el entorno, lo cual es costoso o inviable en robots físicos. Esto motiva el uso de simulación, aprendizaje offline o aprendizaje a partir de demostraciones.

2.3.2 Formulación del problema de planificación como un MDP

Para aplicar algoritmos de aprendizaje por refuerzo a la planificación de trayectorias en robótica, es necesario modelar el entorno como un MDP. La formulación como MDP permite aplicar algoritmos de aprendizaje por refuerzo profundo para encontrar una política óptima $\pi^*(a|s)$ que maximice el retorno esperado:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{2.37}$$

2.3.2.1 Representación del estado

El estado representa una descripción completa del sistema en un instante de tiempo. En el contexto de un manipulador robótico, el estado puede ser o incluir combinaciones de los siguientes elementos:

- **Configuración articular**: Vector de posiciones q y/o velocidades articulares $\dot{q}$.
- **Pose del efector final**: Posición y orientación expresadas en coordenadas cartesianas o como matriz de transformación homogénea.
- **Distancia al objetivo**: Vector de error entre la pose actual del efector y el objetivo.
- **Estado del entorno**: Presencia de obstáculos, posición de objetos a manipular, contactos con el entorno.
- **Percepción sensorial**: Imágenes RGB/D, nubes de puntos, entradas de cámaras o sensores de fuerza.

La elección del estado debe ser tal que contenga toda la información relevante para la toma de decisiones del agente. Esto implica que el estado debe ser suficiente para predecir las consecuencias de las acciones futuras, permitiendo al agente aprender una política óptima. Esta elección depende del tipo de tarea y del nivel de abstracción requerido.

Una mala elección del estado puede llevar a un aprendizaje ineficiente o a la incapacidad de resolver la tarea, ya que el agente no tendría acceso a la información necesaria para tomar decisiones informadas.

2.3.2.2 **Espacio de acción**

Una acción define la decisión que toma el agente en cada estado. En robótica, las acciones pueden modelarse a diferentes niveles:

- **Espacio articular**: Acciones como velocidades o incrementos angulares en las articulaciones $\dot{q}$ o Δq.
- **Espacio cartesiano**: Velocidades lineales y angulares del efector final (twist), o desplazamientos deseados en el espacio operativo.
- **Acciones discretas**: Seleccionar entre un conjunto finito de movimientos (o incrementos) predefinidos.

- **Acciones jerárquicas o simbólicas**: Como “agarrar”, “mover hacia”, “soltar”, útiles en entornos de planificación de alto nivel.

En tareas de planificación de trayectorias, es común que el espacio de acciones sea continuo, por lo que los algoritmos de DRL utilizados deben estar diseñados para operar sobre $\mathbb{R}^n$.

2.3.2.3 **Diseño de la función de recompensa.**

La función de recompensa cuantifica el desempeño del agente en función de sus acciones. Un buen diseño de la recompensa es esencial para guiar el aprendizaje hacia comportamientos deseados. En planificación de trayectorias, se suelen emplear:

- **Recompensa final binaria**: por alcanzar la configuración deseada. Por ejemplo $r = 1$ si $\|q(t) - q_{\text{goal}}\| < \varepsilon$, $r = 0$ en otro caso.
- **Recompensa densa**: basada en la distancia al objetivo, del tipo $r = -\|f(q) - x_{\text{goal}}\|^2$.
- **Penalización por colisiones**: recompensas negativas si el robot entra en C_{obs}.
- **Penalización energética o de suavidad**: para evitar trayectorias abruptas, por ejemplo $r = -\|\dot{q}\|^2$.

El diseño de la recompensa influye significativamente en la eficiencia del entrenamiento y en la calidad de la solución aprendida. Además, se pueden incorporar técnicas de *reward shaping* para proporcionar recompensas adicionales por avanzar hacia el objetivo, evitar zonas peligrosas o completar subtareas. Esto ayuda a acelerar el aprendizaje y guiar al agente hacia comportamientos más eficientes.

2.3.2.4 **Evolución del proceso de decisión**

El agente opera según el siguiente bucle:

1. Observa el estado actual s_t.
2. Selecciona una acción $a_t \sim \pi(a|s_t)$.
3. Ejecuta a_t en el entorno y transita al nuevo estado $s_{t+1} \sim P(s|s_t, a_t)$.
4. Recibe una recompensa $r_t = R(s_t, a_t)$.
5. Repite el proceso, actualizando su política para maximizar el retorno acumulado.

Este ciclo continúa durante múltiples episodios, permitiendo al agente aprender una política que produzca trayectorias válidas. Si bien la política opera sin información explícita de los modelos cinemáticos o dinámicos del robot, suele ser habitual que opere sobre una simulación del entorno, donde estos modelos están implementados. Esto permite al agente aprender de forma segura y eficiente, evitando el desgaste físico del hardware y acelerando el proceso de entrenamiento.

Esta formulación permite resolver tareas de planificación de movimiento como un problema de decisión secuencial bajo incertidumbre, integrando percepción, control y optimización de forma unificada.

2.3.3 **Desafíos de la transferencia a entornos reales**

Una de las principales ventajas del DRL en robótica es la posibilidad de entrenar políticas en entornos simulados, donde los episodios pueden ejecutarse a gran velocidad, sin riesgo físico, y con control total sobre la dinámica del entorno. Sin embargo, una política entrenada íntegramente en simulación rara vez se transfiere directamente con éxito al mundo real debido a discrepancias entre ambos dominios. Este fenómeno se conoce como el problema de *sim-to-real gap* (Jakobi et al., 1995; Zhao et al., 2020).

Las principales diferencias que dificultan la transferencia pueden clasificarse en:

- **Modelos físicos imperfectos**: La simulación puede no capturar fricciones, contactos, flexibilidad o dinámicas no lineales del robot.
- **Sensórica y ruido**: La sensórica utilizada en los entornos reales suele sufrir ruido en la medida y errores de calibración que no se replican en simulación.
- **Diferencias en el tiempo de ejecución**: La simulación puede ser determinista y estable, mientras que el sistema real sufre latencias y *jitter* en la ejecución.
- **Errores de calibración**: Diferencias entre el modelo geométrico del robot o su cinemática real.

Estas discrepancias provocan que políticas entrenadas exitosamente en simulación puedan generar comportamientos fallidos o inesperados al desplegarse en el hardware real.

Para abordar este problema, se han desarrollado diversas estrategias que buscan mejorar la generalización de las políticas entrenadas:

- **Randomización de dominios (Domain Randomization)**: Consiste en entrenar la política en múltiples variaciones del entorno simulado (ruido en sensores, variación en masas, texturas, iluminación, etc.), de forma que la política aprenda a ser robusta ante variaciones desconocidas (Tobin et al., 2017a).
- **Ajuste fino en el entorno real**: Se realiza una adaptación posterior al entrenamiento simulado, ajustando la política directamente en el hardware, generalmente con datos limitados.
- **Aprendizaje híbrido**: Mezclar demostraciones reales con episodios simulados para reducir la dependencia exclusiva de la simulación.

En planificación de trayectorias, donde la precisión espacial, la prevención de colisiones y el cumplimiento de restricciones físicas son esenciales, la brecha *sim-to-real* puede comprometer seriamente la seguridad del sistema. Una política que minimiza la distancia al objetivo en simulación puede colisionar con obstáculos mal modelados en el mundo real. Por ello, los enfoques más exitosos combinan simulación masiva con validación y refinamiento en entornos controlados reales.

Perdona siempre a tus enemigos; nada les molesta más.

Oscar Wilde

CAPÍTULO

3

Planificación de trayectorias

Contenido del Capítulo

ESTE CAPÍTULO ofrece una revisión crítica de la evolución de la planificación de trayectorias para manipuladores industriales, situando el problema en el contexto de más de seis décadas de investigación y desarrollo. Desde los primeros métodos deterministas de enseñanza por guiado y programación punto a punto, hasta los algoritmos probabilísticos, la optimización local y las aproximaciones híbridas más recientes, el recorrido histórico revela cómo las necesidades de la industria han impulsado soluciones cada vez más eficientes, escalables y adaptativas.

El objetivo central de este capítulo es proporcionar una visión completa de los fundamentos teóricos y prácticos que sustentan la planificación de trayectorias, con especial atención a las limitaciones de los métodos clásicos frente a los entornos dinámicos y no estructurados que caracterizan a la industria. En este marco, se exploran las principales familias de planificadores: los métodos basados en muestreo, que constituyen el estándar de facto en bibliotecas como Open Motion Planning Library (OMPL); los métodos de optimización local, centrados en refinar trayectorias iniciales para mejorar suavidad y viabilidad dinámica; y los enfoques híbridos, que buscan combinar lo mejor de ambos paradigmas. Finalmente, se introduce el cambio de paradigma impulsado por el aprendizaje automático, y en particular por el aprendizaje por refuerzo profundo (DRL), que ha abierto la puerta a sistemas de planificación más autónomos y reactivos.

Esta revisión no solo sienta las bases conceptuales para los capítulos experimentales posteriores, sino que también destaca la necesidad de explorar alternativas a los planificadores probabilísticos clásicos. En particular, se subraya cómo el aprendizaje profundo por refuerzo representa una oportunidad única para superar limitaciones de adaptabilidad, consistencia y eficiencia, lo que motiva directamente la propuesta metodológica desarrollada en esta tesis.

La estructura del capítulo es la siguiente: en la Sección 3.1 se presenta una panorámica de la evolución histórica de la planificación de trayectorias, desde los primeros enfoques deterministas hasta la estandarización en frameworks como MoveIt. La Sección 3.2 analiza en detalle los métodos basados en muestreo probabilístico, mientras que la Sección 3.3 aborda los métodos de optimización local. En la Sección 3.4 se estudian las soluciones híbridas que combinan muestreo y optimización, y en la Sección 3.5 se revisan las tendencias emergentes

basadas en aprendizaje automático, con énfasis en el aprendizaje por refuerzo profundo aplicado a la planificación geométrica.

3.1 Evolución histórica de la planificación de trayectorias

La planificación de trayectorias para manipuladores industriales de seis grados de libertad ha pasado por tres grandes etapas históricas: métodos offline deterministas basados en enseñanza por puntos, planificadores automatizados offline y técnicas online adaptativas.

3.1.1 Primeros brazos articulados y programación por guiado

La década de 1960 marcó el nacimiento de los robots industriales articulados de seis ejes, iniciando la necesidad de planificar sus movimientos. En 1961 se instaló en una fábrica de *General Motors* el *Unimate*, considerado el primer brazo robot industrial, diseñado por George Devol y Joseph Engelberger (Gasparetto and Scalera, 2019). Estos primeros robots realizaban tareas repetitivas con movimientos predefinidos punto a punto. Dado que la capacidad de cómputo era muy limitada, la forma predominante de "programar" estas máquinas era de manera determinista y manual, mediante el método conocido como *teaching by showing* o guiado por demostración: un operario movía físicamente el robot a través de cada posición deseada y se registraban las coordenadas articulares de toda la trayectoria para reproducir la secuencia. Este enfoque, era simple y no requería ordenadores, por lo que se difundió ampliamente en los años 60 y 70 (Lozano-Perez, 1983).

Sin embargo, sus limitaciones pronto se hicieron evidentes: las "trayectorias" eran secuencias rígidas de puntos sin posibilidad de adaptación ni lógica condicional. No se podían incluir bucles, condiciones ni reacciones a sensores en estos programas primitivos, lo cual era suficiente para tareas sencillas como soldadura por puntos o pintura, pero inadecuado para tareas más complejas (ensamblajes, manipulación con variaciones) que requerían decisiones basadas en sensores (Lozano-Perez, 1983).

3.1.2 **Lenguajes de programación y cálculo automático de interpolaciones**

La industria empezó a demandar mayor flexibilidad y precisión en la ejecución, así como formas más rápidas de reprogramar los robots para nuevas tareas. A mediados de los años 1970, con el abaratamiento de las computadoras, surgieron los primeros lenguajes de programación de robots que permitieron una programación offline más sofisticada y algorítmica. Un ejemplo pionero fue WAVE, desarrollado en la Universidad de Stanford (1973) como el primer lenguaje de programación de robots de propósito general (Srihari and Deisenroth, 1988).

WAVE permitía especificar movimientos del manipulador en coordenadas cartesianas y planificar trayectorias suavizadas de forma automatizada, aunque su ejecución aún implicaba calcular la trayectoria por adelantado en un computador central (Lozano-Perez, 1983). Poco después aparecieron otros lenguajes como AL (sucesor de WAVE, con sintaxis tipo ALGOL), AML de IBM y VAL de *Unimation*, entre finales de los 70 e inicios de los 80 (Srihari and Deisenroth, 1988). Estas plataformas introdujeron estructuras de programación convencionales (condicionales, bucles) y comandos específicos para mover robots, integrando la lógica basada en información de diferentes sensores en los programas. Por ejemplo, el lenguaje VAL (usado en los robots PUMA) ofrecía instrucciones de movimiento tanto en espacio articular como en espacio cartesiano (comandos *MOVE* para movimiento articulado y *MOVES* para movimiento lineal del efector final) (Haviland and Corke, 2024).

Un avance crucial fue que los propios controladores de los robots empezaran a encargarse del cálculo de la trayectoria entre posiciones guardadas. En lugar de requerir que el programador definiera manualmente todos los puntos intermedios o aceleraciones, el controlador podía interpolar automáticamente con perfiles de velocidad predefinidos (por ejemplo, curvas trapezoidales o polinomios cúbicos) para generar movimientos suaves y deterministas. VAL, por ejemplo, implementaba un intérprete en tiempo real que calculaba las trayectorias “sobre la marcha”, eliminando la necesidad de cálculo previo *offline* y facilitando la interacción con el usuario.

Este paso de la interpolación *offline* a la generación automática de trayectorias en línea fue posible gracias a los microprocesadores más potentes de los años 80, y respondía a necesidades industriales de mayor rapidez en la programación y recalculado de trayectorias, reduciendo tiempos muertos en el cambio de tareas. Para finales de los años 1980, la investigación en robótica empezó a abordar de forma sistemática el problema de la planificación automática de trayectorias libres de colisión.

3.1.3 **Planificación geométrica: descomposición en celdas y campos potenciales**

Un hito teórico fundamental fue la introducción del concepto del espacio de configuración (*configuration space*) (Lozano-Perez, 1983). Esta formulación abstrae todas las posibles posiciones de un robot (sus ángulos articulares) como puntos en un espacio numérico de N dimensiones (siendo N los GDL); dentro de este espacio, los obstáculos del entorno se mapean a una región prohibida llamadas espacio de obstáculos (*obstacle space*), que es un subconjunto del espacio de configuración (Lozano-Pérez, 1983). De este modo, el problema geométrico de mover un brazo articulado sin colisionar se podía analizar computacionalmente buscando caminos en el espacio de configuración que eviten dichas regiones prohibidas.

Apoyados en este concepto, florecieron los primeros algoritmos de planificación geométrica. Por un lado, los métodos de descomposición en celdas dividían el espacio libre en regiones simples (celdas) y construían grafos de conectividad para encontrar rutas factibles (Lozano-Pérez, 1983). Por otro lado, se exploraron técnicas de campos potenciales artificiales para la planificación libre de colisiones en tiempo real (Khatib, 1990). En el enfoque de Khatib, el robot se modela como una partícula bajo influencias de un "potencial" atractivo en la meta y repulsivo en las cercanías de obstáculos, generando fuerzas virtuales que lo conducen por una trayectoria segura de forma reactiva. Estos algoritmos geométricos deterministas supusieron las primeras herramientas de planificación deliberativa: la controladora del robot (o un computador externo) podía calcular automáticamente una ruta libre de colisiones en su espacio de trabajo, en lugar

de depender enteramente de trayectorias registradas por el usuario. No obstante, su aplicabilidad práctica estaba limitada por la complejidad computacional.

Por una parte, los métodos de descomposición en celdas eran exhaustivos y podían encontrar trayectorias óptimas, pero su complejidad crecía exponencialmente con el número de grados de libertad del robot y la cantidad de obstáculos en el entorno. Por otra parte, los campos potenciales eran rápidos y reactivos, pero podían quedar atrapados en mínimos locales o generar trayectorias subóptimas si no se ajustaban adecuadamente los parámetros del potencial. En resumen, aunque estos primeros algoritmos sentaron las bases teóricas y prácticas para la planificación automática de trayectorias, su uso estaba limitado a entornos relativamente simples y robots con pocos grados de libertad. A finales de los años 80 y principios de los años 90, la investigación comenzó a buscar métodos más eficientes y escalables que pudieran manejar la creciente complejidad de los robots industriales modernos.

3.1.4 Introducción de métodos probabilísticos: PRM y RRT

A mediados de los años 90 se produjo un salto significativo gracias a la introducción de métodos probabilísticos de planificación, que sacrificaban la exhaustividad determinista a cambio de eficiencia en espacios complejos. El método Probabilistic Roadmap (PRM), (Kavraki et al., 1996), ejemplifica esta nueva clase de algoritmos. En PRM, se generan aleatoriamente muchas configuraciones válidas (sin colisión) del robot y se construye con ellas un grafo de rutas conectando dichas configuraciones mediante interpolaciones sencillas; ese grafo probabilístico sirve como "mapa" del espacio de configuración libre sobre el cual se busca un camino entre el inicio y la meta.

Poco después surgieron los Rapidly-exploring Random Tree (RRT) (LaValle, 1998), árboles aleatorios que exploran rápidamente el espacio de configuración mediante muestras aleatorias sucesivas. Los algoritmos RRT demostraron gran eficacia para planificar movimientos en robots de muchos grados de libertad e incluso con estructuras cinemáticas complicadas, al ampliar un árbol de posibles movimientos desde el estado inicial hasta alcanzar la meta. Estos enfoques basados en el muestreo probabilístico marcaron un cambio de paradigma: en

lugar de dividir sistemáticamente el espacio o de seguir gradientes artificiales, exploran el espacio de manera estocástica.

La industria comenzó a beneficiarse de estos avances a finales de los años 90 y 2000, incorporándolos en herramientas de programación offline de robots. Por ejemplo, los simuladores y paquetes de programación *offline* podían calcular rutas automáticamente evitando colisiones en entornos complejos, reduciendo drásticamente el tiempo de programación manual. Si bien los métodos probabilísticos no garantizan encontrar la ruta óptima ni probar la inexistencia de camino, en la práctica cumplen con las necesidades industriales de encontrar trayectorias válidas en tiempos razonables, incluso en escenarios con muchos obstáculos o múltiples articulaciones. Esto respondía a la creciente exigencia de cálculo casi en tiempo real y replanificación rápida ante cambios en el entorno.

3.1.5 **Consolidación de los métodos probabilísticos**

Hacia la década de 2010, la comunidad robótica consolidó numerosas de estas técnicas dentro de *frameworks* de *software* reutilizables, facilitando su adopción tanto en investigación como en entornos productivos. Un ejemplo destacado es la Open Motion Planning Library (OMPL) (Şucan et al., 2012). Esta librería es un *framework* de código abierto que implementa una amplia gama de algoritmos de planificación de movimiento basados en muestreo, desde PRM y RRT hasta variantes óptimas, con una interfaz simple para que el usuario pueda resolver problemas complejos de planificación con mínimo esfuerzo. Con el nacimiento del Robot Operating System (ROS) en 2007, OMPL se integró como parte del ecosistema de ROS (a través de MoveIt), convirtiéndose en el estándar de facto para planificación de trayectorias en robots móviles y manipuladores. OMPL no solo proporciona implementaciones eficientes de algoritmos probados, sino que también permite a los desarrolladores crear nuevos planificadores personalizados sin tener que preocuparse por la complejidad subyacente de la gestión del espacio de configuración y las comprobaciones de colisión. Esta integración ha permitido que los investigadores y desarrolladores se enfoquen en resolver problemas específicos de sus aplicaciones robóticas, reutilizando componentes robustos y optimizados. La adopción de OMPL ha sido masiva, permitiendo que

robots industriales y móviles accedieran a algoritmos de planificación avanzados mediante componentes estandarizados (Zhang et al., 2024). La aparición de OMPL y plataformas similares refleja un auge de la planificación automatizada: en lugar de re-desarrollar algoritmos *ad hoc* para cada robot, la industria dispone de herramientas genéricas y optimizadas que se pueden aplicar a distintos brazos de 6 GDL, reduciendo los tiempos de desarrollo de aplicaciones robóticas.

3.1.6 **Nacimiento de los métodos basados en optimización**

En paralelo, durante la década comprendida entre 2010 y 2020 surgieron nuevos métodos de optimización de trayectorias que complementaron a los planificadores tradicionales. Mientras que PRM/RRT y sus variantes se enfocan en hallar alguna ruta libre, estos métodos de optimización toman una trayectoria (es decir, una ruta inicial calculada o una secuencia de puntos) y la refinan para mejorar su calidad según ciertos criterios (longitud, suavidad, tiempo de ejecución, distancias a obstáculos, etc). Un pionero fue Covariant Hamiltonian Optimization for Motion Planning (CHOMP) (Ratliff et al., 2009), que aplica gradientes funcionales para ajustar de manera iterativa una trayectoria inicial y minimizar un costo que refleja colisiones y falta de suavidad. CHOMP demostró que era posible partir de una trayectoria factible cualquiera (incluso subóptima) y converger hacia una trayectoria localmente óptima más suave y libre de colisiones, todo mediante cálculo numérico eficiente. Posteriormente, Stochastic Trajectory Optimization for Motion Planning (STOMP) (Kalakrishnan et al., 2011), adoptó un enfoque estocástico: genera múltiples perturbaciones aleatorias de la trayectoria y promedia sus efectos para reducir el coste, logrando escapar de mínimos locales y manejar restricciones generales. Por su parte, Trajectory Optimization (TrajOpt) (Schulman et al., 2014) incorporó directamente las colisiones como restricciones en un esquema de optimización secuencial convexa, logrando planificar desde cero trayectorias libres optimizando un coste con restricciones de colisión y dinámica. TrajOpt mostró ventajas notables en velocidad de cómputo (resolvía problemas de brazos de 7 GDL en del orden de 0.1–1 s) y ca-

lidad de las soluciones, superando tanto a planificadores de muestreo (OMPL) como a enfoques previos como CHOMP en varios escenarios.

En paralelo, empezaron a surgir las variantes óptimas de los planificadores probabilísticos, como Rapidly-exploring Random Tree Star (RRT*) y Probabilistic Roadmap star (PRM*) (Karaman and Frazzoli, 2011), que garantizan encontrar una trayectoria óptima global en el espacio de configuración al aumentar el número de muestras. Estos métodos combinan la exploración estocástica con la optimización, logrando trayectorias más cortas y suaves que sus predecesores.

A diferencia de los métodos de optimización local, que requieren una trayectoria inicial, los planificadores óptimos pueden encontrar caminos óptimos desde cero, a costa de tiempos de cómputo más largos. Por ejemplo, RRT* puede tardar varios segundos en encontrar una trayectoria óptima en entornos complejos, mientras que PRM* puede requerir minutos dependiendo de la densidad de obstáculos y el número de muestras. Sin embargo, ambos han demostrado ser efectivos en aplicaciones industriales donde se requiere alta precisión y suavidad en los movimientos.

El impacto industrial de estos métodos de optimización ha sido mejorar la suavidad y eficiencia de las trayectorias: por ejemplo, trayectorias más suaves reducen las vibraciones y el desgaste mecánico, trayectorias más cortas o con menos aceleraciones bruscas reducen tiempos de ciclo y consumo energético, todos factores cruciales en producción. Además, estas técnicas permiten incorporar restricciones de la tarea (mantener cierta orientación de la herramienta durante el movimiento, evitar zonas prohibidas, etc) directamente en el proceso de planificación, aumentando la precisión y confiabilidad de los movimientos generados.

3.1.7 **Enfoques híbridos: *Pipelines* de planificación y optimización**

Finalmente, una tendencia reciente ha sido combinar lo mejor de ambos mundos mediante enfoques híbridos de planificación. En entornos reales complejos, un planificador puramente global (como RRT) puede encontrar un camino, pero

quizá no sea el más suave; a su vez, un optimizador local como CHOMP mejora una trayectoria dada, pero no puede encontrar caminos si la inicial se aleja mucho de una solución válida.

La solución híbrida consiste en encadenar un planificador global con un optimizador local, típicamente usando el primero para obtener rápidamente una trayectoria factible y luego refinándola con el segundo. Por ejemplo, en aplicaciones basadas en ROS con MoveIt se emplea un planificador de muestreo de OMPL para generar una ruta inicial entre el punto de partida y el objetivo, y acto seguido se ejecuta CHOMP o STOMP para optimizar esa ruta respecto a longitud y distancia a obstáculos (Liu and Liu, 2022).

En este trabajo reportan que este tipo de pipeline mejora significativamente la calidad de la trayectoria (cuanto menor sea la longitud, los movimientos se realizan de forma más suave) a coste de un ligero aumento en el tiempo de cómputo, cumpliendo así con las exigencias de trayectorias óptimas y seguras en entornos industriales. La planificación híbrida también brinda robustez ante cambios dinámicos: el planificador global puede replanificar si surge un obstáculo nuevo, mientras que el optimizador ajusta localmente la trayectoria. Esto es especialmente útil en células de manufactura flexibles o con interacción humano-robot, donde se requieren trayectorias reactivas y recalculadas al vuelo sin sacrificar la suavidad.

3.1.8 **Tendencias actuales: Aprendizaje automático en planificación de trayectorias**

En la actualidad (2020s), impulsados por la necesidad de mayor autonomía y rapidez, han emergido enfoques basados en aprendizaje automático aplicados a la planificación de movimientos.

En lugar de diseñar algoritmos de planificación completamente a mano, se entrena a modelos de inteligencia artificial (redes neuronales, sistemas de aprendizaje por refuerzo, etc) para que asistan o sustituyan al planificador tradicional (Wang et al., 2021a). Por ejemplo, se han desarrollado redes neuronales que predicen directamente configuraciones libres cercanas a una ruta óptima, guiando

la exploración de algoritmos de muestreo para que converjan más rápido en soluciones de calidad.

Otros trabajos emplean aprendizaje por refuerzo para que un agente (el robot virtual) aprenda a moverse en su espacio de trabajo evitando obstáculos, sin cálculo explícito de caminos, generando políticas reactivas de movimiento o en tiempo real ante objetivos cambiantes.

También destaca el aprendizaje a partir de demostraciones: el robot puede generalizar nuevas trayectorias imitando ejemplos de movimiento dados por un operador humano, técnica útil para programar rápidamente tareas complejas sin modelado geométrico exhaustivo.

Estas tendencias aprovechan datos y experiencia previa para lograr planificaciones más rápidas y adaptativas: métodos recientes han mostrado resolver problemas de planificación de brazos robóticos con obstáculos en milisegundos tras un periodo de entrenamiento, superando con amplitud los tiempos de planificadores tradicionales en escenarios similares (Wang et al., 2021a). Además, los algoritmos de aprendizaje pueden continuamente mejorar su desempeño al exponerse a nuevas situaciones, alineándose con la demanda industrial de robots cada vez más inteligentes, capaces de replanificar al vuelo frente a imprevistos en entornos no estructurados o de optimizar sus trayectorias con múltiples criterios (tiempo, energía, seguridad).

Si bien aún es un campo en desarrollo, la incorporación de técnicas de inteligencia artificial en la planificación de trayectorias augura robots industriales más autónomos, con movimientos óptimos calculados en tiempo real y mayor capacidad de interacción con entornos cambiantes.

En resumen, la planificación de trayectorias para robots industriales de seis GDL ha evolucionado en sincronía con las necesidades de la industria y los avances tecnológicos. Se pasó de simples secuencias punto a punto programadas manualmente en los años 60–70, a sistemas deterministas de interpolación polinomial y lenguajes de programación dedicados en los 80, que dieron al programador herramientas para incorporar lógica y mejorar la precisión de los movimientos. Los 90 trajeron algoritmos automáticos de planificación geométrica y probabilística que permitieron enfrentar la complejidad de entornos reales con eficiencia, respondiendo a la exigencia de ciclos de planificación más rápidos

y configuraciones más complejas. En las dos últimas décadas, la disponibilidad de *frameworks* como OMPL y MoveIt, junto con nuevos algoritmos de optimización de trayectorias y estrategias híbridas, han proporcionado soluciones para obtener trayectorias cada vez más óptimas, seguras y ajustadas a restricciones industriales (minimizando tiempos de ciclo, energía consumida, desgaste, etc), sin sacrificar la viabilidad computacional en tiempo real. Finalmente, la integración de técnicas de aprendizaje automático representa la frontera actual, apuntando a una planificación de movimientos más inteligente y adaptativa, capaz de satisfacer las demandas de flexibilidad y reactividad de la Industria 4.0 (Luo and Li, 2024).

3.2 Planificación basada en muestreo probabilístico

Los planificadores basados en muestreo han sido fundamentales en el avance de la planificación de movimiento para manipuladores robóticos, especialmente por su eficacia en espacios de configuración de alta dimensión y entornos complejos. Algoritmos como RRT (LaValle, 1998), PRM (Kavraki et al., 1996) y sus múltiples variantes se han convertido en el estándar para generar trayectorias factibles cuando las soluciones exactas son inviables. Estos planificadores construyen un grafo o árbol de búsqueda muestreando incrementalmente el espacio de configuración y conectando puntos muestreados mediante primitivas de planificación local. Su atractivo radica en la garantía probabilística de encontrar una solución, la aplicabilidad generalista y la mínima dependencia de heurísticas específicas de la tarea. A diferencia de los métodos deterministas, que garantizan encontrar una solución óptima cuando existe un muestreo lo suficientemente denso o una formulación convexa, los algoritmos probabilísticos renuncian a garantías exactas a cambio de escalabilidad y flexibilidad.

3.2.1 Clasificación de algoritmos probabilísticos de planificación

Los algoritmos de planificación de trayectorias basados en muestreo aleatorio (*sampling-based*) se dividen clásicamente en planificadores geométricos y planificadores kinodinámicos. Los primeros buscan caminos geométricamente fac-

tibles en el espacio de configuración (ignorando las características dinámicas del robot), mientras que los segundos incorporan explícitamente restricciones cinemáticas y dinámicas (operando en el espacio de estado, que incluye velocidad y otras variables dinámicas). En la figura 3.1 se muestra una taxonomía de los principales algoritmos de planificación basados en muestreo probabilístico.

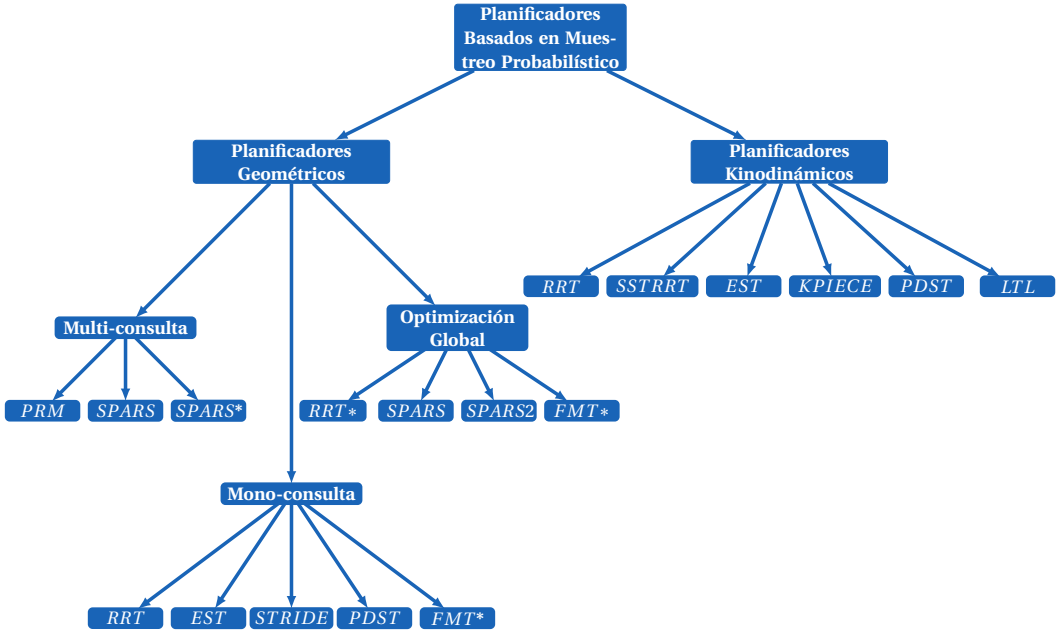

Figura 3.1: Taxonomía de los principales algoritmos de planificación basados en muestreo probabilístico. Se omiten los numerosos planificadores derivados de RRT y PRM, así como sus variantes optimizadas (marcadas con *). Los planificadores multi-consulta pueden emplearse también en modo mono-consulta, aunque su fortaleza radica en la reutilización del roadmap para múltiples problemas de planificación.

En los siguientes subapartados, se presenta una clasificación detallada de los planificadores probabilísticos de muestreo según estas categorías, basada en literatura científica revisada por pares, incluyendo sus subcategorías principales, características, ventajas, limitaciones y usos típicos.

3.2.1.1 **Planificadores geométricos**

Los planificadores geométricos asumen que el robot puede ejecutar cualquier trayectoria geométrica, es decir, que la transición entre dos puntos consecuti-

vos de la trayectoria es interpolable numéricamente. Por ello, solo consideran la geometría del problema (obstáculos y configuraciones alcanzables) en el espacio de configuración. Esto permite ignorar las ecuaciones de movimiento, simplificando el problema de búsqueda de un camino libre de colisiones. Dentro de este grupo general se distinguen subcategorías según el modo de uso y los objetivos de optimización.

3.2.1.1.1 Planificadores multi-consulta

Los métodos multi-consulta construyen una estructura de datos global (generalmente un grafo de caminos llamados *roadmap*) en una fase de preprocesamiento, que luego puede responder eficientemente múltiples consultas de ruta en el mismo entorno. El representante clásico es el Probabilistic Roadmap (PRM), introducido por (Kavraki et al., 1996).

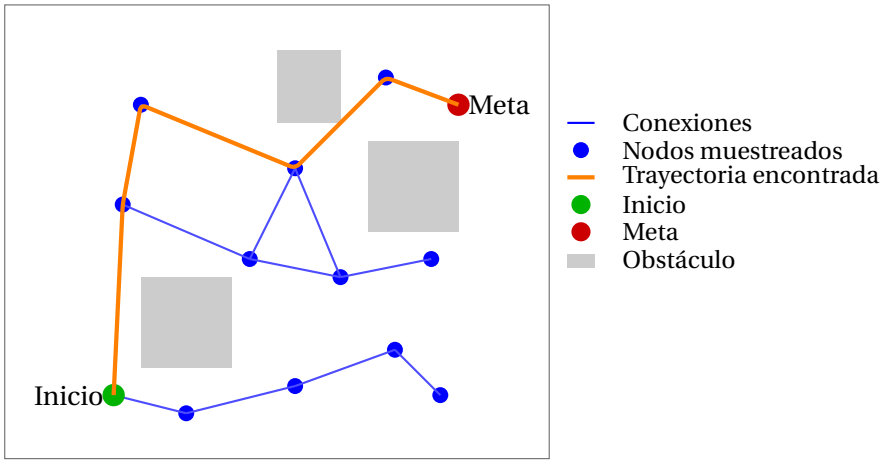

Figura 3.2: Ilustración esquemática del método PRM: se muestrean nodos libres (azul), se conectan formando un grafo, y se encuentra una trayectoria (naranja) entre el inicio (verde) y la meta (rojo) evitando obstáculos (gris).

PRM procede en dos fases: una fase de exploración donde se muestrean aleatoriamente configuraciones libres y se conectan mediante un planificador local para formar un grafo disperso, y una fase de consulta donde el inicio y meta se conectan al grafo y se busca un camino en él.

Debido a que el *roadmap* se construye independientemente de una consulta específica, puede reutilizarse para responder rápidamente muchas consultas di-

ferentes en el mismo espacio de trabajo. PRM es probabilísticamente completo (encuentra una solución si existe, dado suficiente muestreo), pero no garantiza trayectorias óptimas y tiende a requerir muchos nodos y aristas para cubrir adecuadamente espacios complejos.

Para mejorar la eficiencia de PRM, se han propuesto variantes que reducen el tamaño del grafo manteniendo la capacidad de responder consultas. Un ejemplo es Sparse Roadmap Spanner (SPARS), propuesto por (Dobson and Bekris, 2014), que construye un subgrafo disperso (*spanner*) del espacio de configuración garantizando soluciones de calidad cercana a la óptima con muchas menos aristas. SPARS ofrece una optimización asintótica: a medida que el muestreo aumenta, la calidad del camino hallado converge a no más de un factor constante de la longitud óptima. Además, provee un criterio de terminación significativo: la probabilidad de que nuevos nodos deban añadirse al roadmap tiende a cero al incrementarse las iteraciones.

Una extensión denominada Sparse Roadmap Spanner 2 (SPARS2) elimina la necesidad de construir primero un grafo denso; SPARS2 logra las mismas propiedades teóricas directamente con menor sobrecarga de memoria, manteniendo la convergencia óptima asintótica sin tener que almacenar un *roadmap* completo de densidad infinita.

Estas técnicas son especialmente útiles en escenarios donde se requiere responder muchas consultas en entornos estáticos, ya que el coste de preprocesamiento se amortiza con las múltiples reutilizaciones del *roadmap*.

3.2.1.1.2 Planificadores mono-consulta

En contraste con los anteriores, los métodos mono-consulta (o de consulta única) construyen una estructura de búsqueda específicamente para cada trayectoria solicitada. Típicamente, estos algoritmos exploran el espacio de configuración mediante la expansión incremental de árboles desde el estado inicial (y a veces también desde el objetivo), sin realizar un preprocesamiento global. Su fortaleza radica en resolver rápidamente una consulta en entornos donde un preprocesamiento exhaustivo sería prohibitivo o innecesario. Sin embargo, deben reiniciarse desde cero para cada nueva consulta o cambio en el entorno.

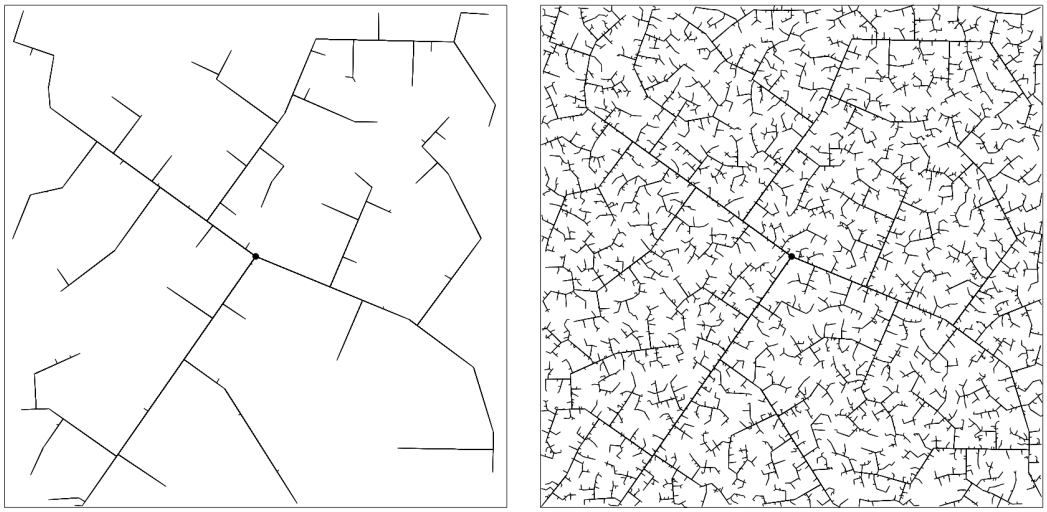

Figura 3.3: Ejemplo de un árbol RRT creciendo en un espacio de configuración de dos dimensiones. A la izquierda se muestran las primeras iteraciones, donde el árbol se expande aleatoriamente hacia nuevas regiones. A la derecha, el árbol ha crecido significativamente, cubriendo una gran parte del espacio de configuración (Lavalle, 2006).

El algoritmo más conocido de esta categoría es RRT, propuesto por (LaValle, 1998). RRT construye iterativamente un árbol de búsqueda que se expande aleatoriamente a través del espacio de configuración, favoreciendo la exploración de regiones inexploradas gracias a su sesgo incremental hacia áreas no cubiertas por el árbol. Esto le permite cobertura rápida de espacios de alta dimensión, ofreciendo gran eficiencia para hallar alguna ruta factible con rapidez. Estos algoritmos son probabilísticamente completos pero producen soluciones subóptimas. Es decir, aunque encuentra un camino muy rápidamente, este suele ser más largo o menos eficiente que el óptimo. Aún así, RRT ha demostrado ser extremadamente útil en robótica gracias a su simplicidad y eficacia.

Su facilidad de implementación y éxito práctico la han convertido en una herramienta básica en planificación de movimiento. Otro algoritmo mono-consulta destacado es Expansive Space Trees (EST), introducido por (Hsu et al., 1997). EST también construye un árbol aleatorio, pero aplica una estrategia de muestreo que premia la expansión hacia aquellas regiones del espacio de configuración que han sido menos exploradas hasta el momento (regiones expansivas). Me-

diante una medida de expansión o cobertura local, EST selecciona nodos frontera y dirige la exploración a áreas escasamente muestreadas, lo que incrementa la probabilidad de atravesar pasajes estrechos en el espacio de configuración.

Al igual que RRT, EST está orientado a hallar rápidamente una ruta factible no óptima, y típicamente genera también un único árbol por consulta. Además de RRT y EST, se han desarrollado variaciones enfocadas en mejorar la eficiencia de la exploración mono-consulta en diferentes situaciones. Por ejemplo, Path-Directed Subdivision Tree (PDST) (Ladd and Kavraki, 2005) es un planificador arbóreo que elimina la dependencia de una métrica de distancia explícita. PDST emplea una subdivisión recursiva del espacio en celdas y mantiene un registro de la densidad de trayectorias simuladas en cada celda. La expansión del árbol se guía hacia celdas con baja densidad de muestras, asegurando que el árbol cubra progresivamente todas las regiones alcanzables.

Por su parte, Search Tree with Resolution Independent Density Estimation (STRIDE), presentador por (Gipson et al., 2013), es un algoritmo más reciente, inspirado en EST, que introduce una forma eficiente de estimar la densidad de muestras directamente en el espacio de configuración sin depender de proyecciones de baja dimensión. STRIDE emplea una estructura de acceso geométrico a vecindades para determinar las regiones menos exploradas y focalizar allí la expansión del árbol. Esto resulta especialmente ventajoso en sistemas de alta dimensión, donde las proyecciones lineales tradicionales (usadas en EST para medir cobertura) pueden perder información; STRIDE puede detectar vacíos en el espacio completo de estados y guiar el muestreo hacia ellos.

En evaluaciones comparativas se ha demostrado que STRIDE mantiene un buen balance entre exploración global y explotación local, mejorando la eficiencia en escenarios de dimensión elevada donde otros métodos podrían requerir muchas más iteraciones.

Finalmente, cabe mencionar Fast Marching Tree (FMT*), un algoritmo mono-consulta óptimo propuesto por (Janson and Pavone, 2016) para mejorar la calidad de las soluciones sin sacrificar demasiada eficiencia. En FMT* no crece un árbol incrementando una muestra por iteración como RRT, sino que realiza una expansión “en oleadas”: conecta progresivamente los nodos empezando desde el inicio, explorando primero las conexiones de menor costo acumulado. Se ha

demostrado que FMT* es asintóticamente óptimo, pero converge hacia soluciones cercanas al óptimo con menos muestras y en menos tiempo, especialmente en espacios de alta dimensión o cuando la evaluación de colisiones es costosa (Janson and Pavone, 2016).

Una limitación inherente de estos métodos es que, al no reutilizar información entre consultas, pueden resultar computacionalmente costosos si se deben resolver muchas consultas en el mismo entorno; en tales casos, un enfoque multi-consulta (como PRM) podría ser más apropiado. No obstante, debido a que no requieren preprocesamiento, los métodos mono-consulta son muy flexibles y pueden aplicarse incluso cuando el espacio de configuración no es completamente conocido de antemano o cambia durante la operación del robot.

3.2.1.1.3 Planificadores de optimización global

Los planificadores orientados a optimización, no solo tienen como objetivo encontrar algún camino factible, sino garantizar que, al incrementarse el tiempo de planificación, las soluciones convergerán hacia la óptima global (mínima distancia u otra función de coste definible). Estos métodos se basan en los anteriores (multi o mono consulta) pero incorporan estrategias para refinar y mejorar la calidad de las trayectorias. Estos planificadores son considerados asintóticamente óptimos, es decir, que la calidad de su solución tiende al óptimo global cuando el número de muestras tiende a infinito.

El pionero en este ámbito fue RRT*, presentado por (Karaman and Frazzoli, 2011) como una modificación de RRT que implementa un proceso de re-enlace de nodos para mantener el árbol cercano al mínimo coste. Sin embargo, esta garantía tiene un coste computacional: RRT* requiere considerar muchos vecinos de cada muestra para actualizar conexiones óptimas, lo que aumenta significativamente el tiempo de planificación y hace más lenta la convergencia inicial hacia una solución utilizable.

En la práctica, RRT* suele necesitar más muestras que RRT para obtener un camino, aunque ese camino será de calidad creciente. Por esta razón, RRT* puede ser poco eficiente en situaciones donde se necesita una respuesta muy rápida y un camino subóptimo es aceptable; su fortaleza radica en aplicaciones donde la calidad final de la trayectoria es crucial.

En la categoría multi-consulta, la versión óptima es PRM* (también introducida por (Karaman and Frazzoli, 2011)), que modifica PRM conectando cada nuevo punto con un número creciente de vecinos en lugar de un número fijo, asegurando que la longitud de los caminos converge a la óptima. PRM* hereda de PRM su capacidad multi-consulta, pero también su principal desventaja: para lograr esta optimización asintótica debe seguir añadiendo nodos y aristas indefinidamente. Como se ha discutido anteriormente, esto resulta en *roadmaps* muy densos y costes de consulta más altos.

Precisamente este problema motivó la aparición de algoritmos como SPARS y SPARS2 ya descritos, que encuentran un balance entre optimización y eficiencia. Estudios han mostrado que, aunque SPARS/SPARS2 no siempre alcanzan la misma calidad que PRM*, en la práctica logran caminos muy próximos al óptimo con muchas menos muestras, alcanzando en ciertos escenarios la misma calidad relativa con mayor rapidez. Por tanto, son alternativas atractivas cuando se busca un compromiso entre calidad de solución y recursos limitados. Por su parte, para los algoritmos mono-consulta, ya se mencionó FMT* como un algoritmo orientado a optimización. FMT* ha demostrado ser asintóticamente óptimo (Janson and Pavone, 2016), superando a RRT* en rapidez de convergencia hacia buenas soluciones.

En resumen, los planificadores de esta subcategoría aseguran mejores trayectorias a costa de un mayor esfuerzo de cómputo. La elección entre un método óptimo (como RRT* o PRM*) y uno tradicional (como RRT o PRM) dependerá de los requisitos: si la calidad del camino es crítica se justifica usar estos algoritmos óptimos. En cambio, si se necesita rapidez o las condiciones del entorno son muy dinámicas, puede preferirse un método subóptimo pero computacionalmente más eficiente.

3.2.1.2 **Planificadores kinodinámicos**

Los planificadores kinodinámicos extienden la planificación por muestreo al caso en que el robot tiene restricciones impuestas por sus modelos cinemático y dinámico. Esto significa que el planificador debe encontrar no solo una ruta geométrica válida, sino también un trayecto temporalmente factible: una secuencia

de estados alcanzable mediante órdenes de control que respeten las velocidades, aceleraciones, fuerzas y otras limitaciones del robot. En lugar de trabajar solo en el espacio de configuración, estos métodos operan en el espacio de estado (espacio de configuración y sus derivadas), y las transiciones entre estados se generan a través de la integración de las ecuaciones de movimiento del robot (típicamente ecuaciones diferenciales). La incorporación de estas restricciones hace la planificación más compleja, ya que el espacio a explorar es de mayor dimensión y no todas las conexiones geométricas son viables físicamente.

Un resultado clave de (LaValle and Kuffner, 1999) fue demostrar que el enfoque RRT podía adaptarse eficazmente a la planificación con dinámica. En el algoritmo de *Randomized Kinodynamic Planning* se utiliza un RRT sobre el espacio de estado, donde en cada iteración se elige una acción de control aleatoria y un tiempo de aplicación, simulando progresivamente el movimiento del sistema desde un nodo existente. Si la trayectoria resultante es válida (sin colisiones y dentro de los límites), se agrega al árbol. Este procedimiento genera un árbol de trayectorias dinámicas que explora rápidamente las posibilidades del sistema físico. Una ventaja crucial es que RRT no necesita una función de transición analítica que conecte dos estados; en su lugar, se apoya únicamente en simulaciones incrementales, algo relativamente sencillo de implementar para muchos robots. Sin embargo, al igual que su contraparte geométrica, la implementación kinodinámica de RRT no garantiza soluciones óptimas y tiende a producir trayectorias subóptimas.

De forma similar, el concepto de EST fue llevado al dominio kinodinámico por (Kindel et al., 2000), aplicando muestreo expansivo para un robot con dinámica y hasta consideraron obstáculos en movimiento. En esencia, esta aplicación también simula acciones de control aleatorias desde estados menos explorados, guiándose por proyecciones adecuadas del espacio de estado. Esto puede implicar, por ejemplo, proyectar el estado (posición, velocidad) solo en la posición para evaluar qué áreas geométricas no han sido cubiertas, combinándolo con técnicas para diversificar las velocidades exploradas. Aunque conceptualmente similar al caso geométrico, la dificultad radica en elegir proyecciones y métricas que capturen la accesibilidad dinámica de ciertas regiones.

La planificación kinodinámica motivó también algoritmos específicos diseñados para explorar de manera más eficiente el espacio de estados bajo dinámicas complejas. Dos ejemplos notables son PDST y Kinodynamic Planning by Interior-Exterior Cell Exploration (KPIECE). PDST, ya mencionado anteriormente, además de su enfoque en la densidad de trayectorias, incorpora restricciones dinámicas al definir cómo se pueden conectar los nodos en el espacio de estado. Kinodynamic Planning by Interior-Exterior Cell Exploration (KPIECE), propuesto por (Şucan and Kavraki, 2010), utiliza una estrategia de exploración basada en celdas que divide el espacio de estado en regiones y busca caminos a través de ellas, respetando las restricciones cinemáticas del robot. KPIECE ha demostrado ser efectivo para robots con restricciones complejas, aunque su implementación puede ser más complicada que RRT o EST.

Otro enfoque interesante es Stable Sparse-RRT (SST) presentado por (Li et al., 2014), que introduce dos ideas principales: mantener el árbol actualizado eliminando regularmente nodos redundantes cercanos entre sí y usar una estrategia de selección de nodos basada en dos conjuntos: un conjunto activo (de "testigos" representativos del espacio) y el conjunto completo de estados del árbol. Con esto, SST garantiza que el árbol no crece innecesariamente en regiones ya bien exploradas. En la práctica, SST converge mucho más rápido que RRT* a caminos de alta calidad, y su complejidad computacional es competitiva con la de RRT simple.

Una extensión interesante de la planificación kinodinámica es integrar objetivos de alto nivel expresados a través de una lógica temporal (por ejemplo, alcanzar regiones en cierto orden, visitar periódicamente ciertas zonas, evitar condiciones hasta que se cumplan otras, etc). La Linear Temporal Logic (LTL) se ha utilizado para formalizar estos objetivos. (Plaku, 2012) presentaron un enfoque donde se combina la búsqueda discreta en un autómata finito con la exploración continua en el espacio de estados del robot.

En esta arquitectura, a grandes rasgos, el modelo LTL se convierte en un autómata que supervisa el cumplimiento de la tarea, y un planificador probabilístico (por ejemplo RRT) debe encontrar una trayectoria que no solo alcance una meta geométrica, sino que satisfaga la secuencia de requisitos lógicos impuesta por la fórmula. Para ello, se explora el producto cartesiano entre el espacio de

estados del robot y el autómata, buscando un camino que conduzca a un estado de aceptación del autómata, lo cual equivale a cumplir la especificación temporal (Muhayyuddin et al., 2017).

Fundamentalmente, el planificador genera trayectorias físicas válidas que satisfacen las subtareas en el orden adecuado. Estos planificadores kinodinámicos basados en LTL (también llamados *sampling-based motion planning with temporal goals*) permiten abordar problemas de planificación de alto nivel en robótica autónoma, es decir, cualquier tarea cuyo objetivo no sea solo alcanzar una posición final, sino ejecutar una secuencia de acciones definidas por una lógica temporal definida.

En general, estos algoritmos tienden a ser más lentos y complejos que sus equivalentes geométricos debido a la necesidad de simular y respetar restricciones diferenciales. Por ello, en la práctica se suele optar por planificadores geométricos cuando el robot cuenta con un buen controlador que realiza la transición entre configuraciones, reservando los planificadores kinodinámicos para casos donde las dinámicas no pueden ignorarse. Incluso existe la posibilidad híbrida: planificar geométricamente primero y luego corregir la trayectoria con controladores (aunque esto no garantiza factibilidad si la ruta geométrica es muy demandante dinámicamente). Los avances como SST indican que es posible lograr cierto grado de optimización también bajo dinámicas complejas, pero a un costo computacional mayor. Asimismo, la integración con LTL muestra cómo la planificación por muestreo se está extendiendo para cubrir no solo cómo moverse, sino qué secuencia de acciones cumplir, aportando mayor autonomía a sistemas robóticos.

3.2.2 **Variantes y extensiones de los planificadores**

A lo largo de las últimas décadas se han propuesto numerosas mejoras sobre los algoritmos RRT, PRM y sus derivados, con el fin de aumentar su eficiencia, robustez o calidad de soluciones. Entre las principales técnicas destacan la búsqueda bidireccional, la evaluación diferida de colisiones (*lazy collision checking*), los *roadmaps* dispersos (*sparse*) y los métodos de optimización global de

trayectorias dentro del propio algoritmo de muestreo. A continuación se describen estas variantes y extensiones clave, junto con sus beneficios en el contexto de manipuladores de 6 GDL y otros sistemas robóticos.

3.2.2.1 **Búsqueda bidireccional**

Las técnicas de búsqueda bidireccional consisten en explorar el espacio de configuración simultáneamente desde el estado inicial y desde la meta, haciendo que dos árboles o grafos crezcan y traten de encontrarse en un punto medio. Este enfoque, implementado clásicamente en Rapidly-exploring Random Tree Connect (RRT-Connect) por (Kuffner and LaValle, 2000), consiste en construir dos RRTs independientes en paralelo, uno arraigado en el inicio y otro en la meta, que avanzan aleatoriamente pero también se guían uno hacia el otro gracias a una heurística pre-establecida.

La figura 3.4 ilustra este proceso: los árboles crecen desde ambos extremos y se conectan cuando se encuentran, formando una trayectoria completa. Esta técnica reduce significativamente el tiempo de planificación al disminuir la cantidad de muestras necesarias para conectar dos puntos distantes en el espacio de configuración.

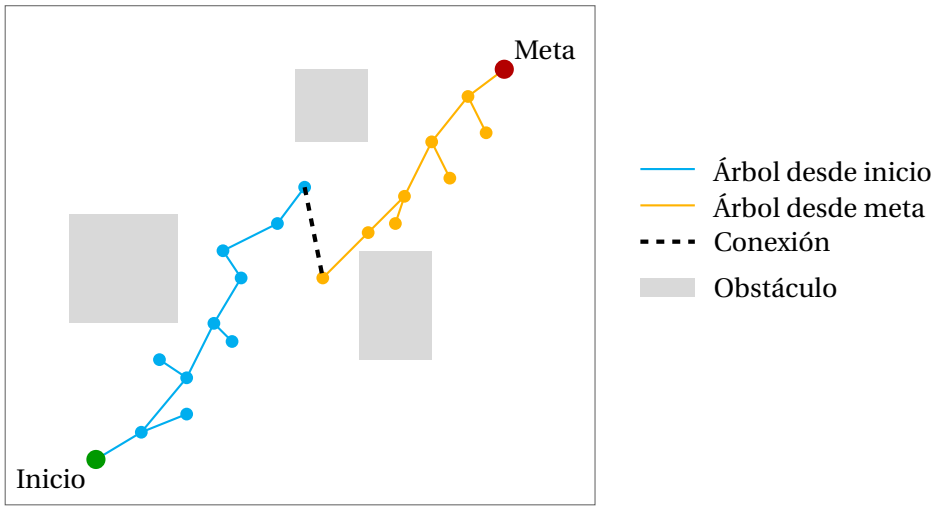

Figura 3.4: Ejemplo de búsqueda bidireccional con RRT-Connect. Dos árboles crecen desde el inicio (verde) y la meta (rojo), intentando conectarse en un punto común (línea discontinua). Los obstáculos se muestran en gris claro.

La estrategia de "conexión" implica extender uno de los árboles en dirección a una muestra aleatoria y, tras cada extensión, intentar unirlo directamente con el otro árbol (creciendo sin limitar el paso, a diferencia de la extensión incremental estándar). Cuando ambos árboles logran enlazarse, se obtiene una trayectoria completa que soluciona el problema. La búsqueda bidireccional reduce drásticamente el tiempo de planificación en consultas individuales, especialmente en espacios amplios o con "pasillos estrechos", pues divide la exploración desde ambos extremos en lugar de hacerlo desde uno solo. RRT-Connect ha demostrado ser un método sumamente eficiente para planificar en espacios de alta dimensionalidad, resolviendo rápidamente movimientos de brazos de 6–7 GDL y otros problemas complejos de manipulación (Kuffner and LaValle, 2000). Gracias a su simplicidad e impresionante rendimiento, variantes bidireccionales de RRT (y otros árboles aleatorios) se han convertido en estándar en la práctica; por ejemplo, muchos frameworks de robótica utilizan RRT-Connect o esquemas similares como planificador por defecto en robots industriales de seis ejes, al obtener soluciones válidas en pocos milisegundos en entornos heterogéneos.

3.2.2.2 **Evaluación diferida de colisiones**

La evaluación diferida de colisiones (*lazy collision checking*) es una optimización transversal a muchos planificadores, se centra en retrasar al máximo las comprobaciones de colisión para ahorrar tiempo computacional. En un algoritmo tradicional, cada vez que se genera un nuevo nodo o conexión, se valida inmediatamente que esté libre de colisión, lo cual puede ser muy costoso cuando hay que hacer miles de verificaciones geométricas. Esta técnica propone construir inicialmente la estructura de búsqueda asumiendo de entrada que todas las muestras y conexiones son válidas, y posponer las validaciones hasta que realmente hagan falta.

La figura 3.5 ilustra este concepto. En lugar de verificar cada conexión al momento de crearla, se construye un grafo con nodos y aristas sin comprobar colisiones. Luego, se busca una ruta candidata que conecta el inicio y la meta. Solo cuando se detecta una colisión en alguna de las aristas de esa ruta, se eliminan esas conexiones y se busca una nueva ruta válida.

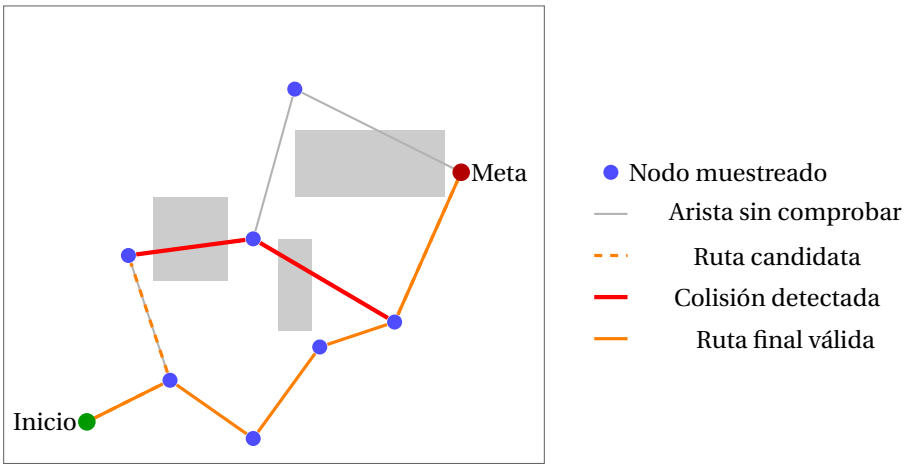

Figura 3.5: Ejemplo de evaluación diferida de colisiones. Inicialmente, se construye un grafo con nodos y aristas sin verificar colisiones (en gris). Luego, se busca una ruta candidata (línea discontinua naranja) que conecta el inicio y la meta. Al detectar colisiones en ciertas aristas (en rojo), estas se eliminan y se busca una nueva ruta válida (en naranja sólida).

Un ejemplo es Lazy PRM (Bohlin and Kavraki, 2000). En este método, se muestrean nodos aleatorios y se conectan formando un grafo denso sin verificar colisiones en ese momento; luego, se busca en dicho grafo la ruta más corta desde el inicio a la meta y solo entonces se chequea si los nodos y aristas de esa ruta están libres de colisión. Si alguna colisión es detectada, se elimina ese tramo del grafo y se vuelve a buscar una ruta alternativa, iterando hasta hallar una trayectoria completamente libre. Al evitar comprobaciones inútiles, la planificación con evaluación diferida de colisiones logra acelerar enormemente la obtención de una solución válida.

Estudios posteriores confirmaron que el *lazy collision checking* reduce la carga computacional en la planificación basada en muestreo, ya que las pruebas de colisión suelen ser el cuello de botella principal en problemas de alta complejidad geométrica (Kim et al., 2018). Esta filosofía de "evaluar solo cuando es necesario" se ha incorporado también en otros algoritmos: por ejemplo, existen variantes *lazy* de RRT y de planificadores óptimos, donde las colisiones de las nuevas ramas o grafos se comprueban de forma diferida durante la expansión o incluso tras hallar un candidato de camino. Asimismo, la idea *lazy* se combinó con la búsqueda bidireccional en planificadores como Single-Query Bi-

directional Lazy PRM (SBL) de (Sánchez and Latombe, 2003), obteniendo algoritmos muy eficientes para consultas únicas en espacios complejos. En entornos con robots manipuladores de múltiples articulaciones, estas técnicas resultan especialmente ventajosas, pues minimizan la evaluación repetitiva de colisiones entre el brazo y obstáculos mientras exploran configuraciones, concentrándose solo en verificar aquellas rutas candidatas que realmente conectan inicio y fin.

3.2.2.3 **Construcción de grafos dispersos**

A medida que los planificadores probabilísticos fueron incorporando objetivos de optimización (apartado 3.2.1.1.3), se hizo evidente que los métodos tradicionales como PRM y RRT, aunque efectivos para hallar caminos válidos, no eran ideales para entornos complejos. En particular, se observó que estos algoritmos tendían a generar demasiados nodos y aristas, lo que impactaba negativamente en la memoria y velocidad de consulta. Para abordar este problema, surgieron técnicas dedicadas a mantener las estructuras de búsqueda lo más dispersas o *sparse* posible sin perder la capacidad de encontrar caminos cortos.

En el caso de PRM, destaca el método SPARS y SPARS2, ya introducidos en el apartado 3.2.1.1.3. En la práctica, SPARS y su variante mejorada SPARS2 logran datos muy compactos y respuestas rápidas: se ha mostrado en simulaciones que los *roadmaps* dispersos conservan soluciones de calidad óptima aproximada utilizando muchísimos menos vértices, lo que acelera la planificación en línea (Dobson and Bekris, 2014). En el dominio de brazos industriales, esto permite preprocesar mapas de movimiento de manera eficiente; por ejemplo, un robot puede aprender un *roadmap* global del entorno con miles de configuraciones menos que un PRM tradicional, y aun así responder consultas punto a punto casi instantáneamente al reutilizar ese grafo reducido. Para planificadores tipo árbol (RRT), también se han desarrollado técnicas de dispersión de muestras como SST. Gracias a esta poda activa, SST genera un árbol más ligero y estable, mejorando con ello la eficiencia y facilitando incluso cierta optimización del camino encontrado. En resumen, la introducción de criterios de dispersión de datos en los planificadores de muestreo ataca el problema del crecimiento descontrolado de la estructura de búsqueda, logrando un equilibrio entre la calidad de las trayectorias generadas y la eficiencia computacional.

3.2.2.4 **Algoritmos óptimos y últimas tendencias**

Durante las últimas dos décadas, los planificadores basados en muestreo han evolucionado significativamente, incorporando no solo la capacidad de encontrar caminos válidos, sino también de optimizar esos caminos según criterios específicos. En el apartado 3.2.1.1.3 se ha visto cómo los algoritmos RRT* y PRM* introdujeron la idea de optimización asintótica, garantizando que, al aumentar el tiempo de cómputo, las soluciones convergerían hacia la óptima. De este modo, a largo plazo, el árbol explora no solo en busca de alguna ruta, sino que va mejorando continuamente las rutas halladas. PRM* aplica un principio similar conectando cada nuevo punto a un número creciente de vecinos en lugar de un número fijo, asegurando que el grafo resultante contenga eventualmente caminos de coste óptimo.

El coste de estas mejoras es un aumento significativo en el número de muestras y cálculos necesarios; RRT*, por ejemplo, suele requerir más iteraciones que RRT para devolver una solución inicial (debido al sobrecoste de intentar optimizar mientras explora). Por este motivo, se han propuesto técnicas para acelerar la convergencia hacia trayectorias de bajo costo sin sacrificar la optimización. Un enfoque efectivo es el muestreo informado introducido en Informed RRT* (Gammell et al., 2014), que mejora la eficiencia de RRT* al restringir el espacio de muestreo a una región elipsoidal alrededor del inicio y la meta. En lugar de muestrear aleatoriamente todo el espacio, Informed RRT* se enfoca en áreas donde es más probable encontrar soluciones óptimas, lo que reduce drásticamente el número de muestras necesarias para converger a un camino corto (Gammell et al., 2018).

Otra estrategia destacada son los métodos de búsqueda por lotes guiados por heurísticas, como Batch Informed Trees Star (BIT*), que combinan ideas de algoritmos A* y RRT*: BIT* realiza expansiones en lotes de muestras calculando cotas de coste y enfocándose primero en conectar aquellas que potencialmente minimizan la distancia (Gammell et al., 2015). En pruebas comparativas, BIT* y variantes similares han mostrado convergencia más veloz hacia soluciones casi óptimas, superando a RRT* en escenarios de alta dimensionalidad o con evaluaciones de colisión costosas.

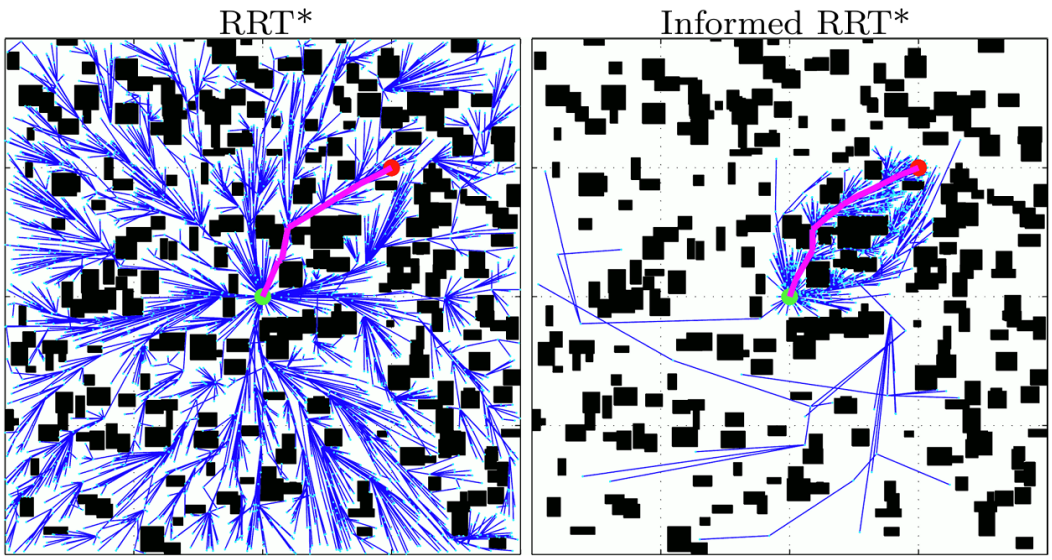

Figura 3.6: Comparación entre RRT* e Informed RRT* para un espacio de configuracion simple. Mientras que RRT* explora todo el espacio de configuración, Informed RRT* se enfoca en una región elipsoidal alrededor del inicio y la meta, acelerando la convergencia hacia soluciones óptimas. (Gammell et al., 2014)

En los últimos años, estos algoritmos se han seguido refinando y extendiendo, incorporando técnicas de muestreo adaptativo, optimización de trayectorias en línea y mejoras en la eficiencia de búsqueda. Gran ejemplo de ello es Advanced BIT* (ABIT*) que incorpora al ya existente BIT* una búsqueda avanzada en grafos en modo *anytime* (Strub and Gammell, 2020). Basándose en métodos como Anytime Truncated D* (ATD*), ABIT* incorpora mecanismos de inflación y truncamiento del árbol de búsqueda que permiten intercalar muestreo y búsqueda heurística de forma más agresiva (Aine and Likhachev, 2013). Esto permite encontrar soluciones en menos tiempo y aplicar las estrategias de convergencia óptima de forma continua.

Finalmente, es común que muchas implementaciones prácticas incluyan un postprocesado de optimización sencillo a cualquier trayectoria encontrada por muestreo: técnicas de atajado (*shortcutting*) y suavizado *spline* pueden reducir la longitud y curvatura del camino sin requerir un algoritmo complejo (Hauser and Ng-Thow-Hing, 2010; Geraerts and Overmars, 2007). Esto significa que incluso planificadores no óptimos como RRT-Connect suelen producir en la práctica trayectorias razonablemente cortas tras unos segundos adicionales de re-

finamiento. No obstante, cuando la exigencia de calidad es alta o se planifican movimientos repetitivos donde cada segundo ahorrado es crítico, los planificadores probabilísticos óptimos (RRT*, PRM*, FMT*, BIT*, etc.) y sus mejoras son herramientas valiosas.

En resumen, la introducción de criterios de optimización en la planificación basada en muestreo ha permitido cerrar la brecha entre la rapidez estocástica y la calidad determinista, ofreciendo trayectorias libres de colisión cada vez más seguras, cortas y adaptadas a métricas de costo relevantes en entornos industriales.

3.2.3 **Implementaciones en entornos reales**

Gran parte de los avances en planificación por muestreo se han desarrollado en el contexto académico, pero su impacto ha trascendido a la robótica industrial y comercial. Los planificadores basados en muestreo se han convertido en una herramienta esencial para resolver problemas de planificación de movimiento en entornos reales, desde brazos robóticos hasta vehículos autónomos.

En el ámbito académico, la biblioteca OMPL ha sido fundamental para estandarizar y facilitar la implementación de estos algoritmos. OMPL proporciona una amplia gama de planificadores basados en muestreo, incluyendo RRT, PRM, EST, FMT*, BIT* y muchos otros, permitiendo a los investigadores probar y comparar diferentes enfoques de manera eficiente. Esta biblioteca ha sido ampliamente adoptada en la comunidad de robótica, sirviendo como base para numerosos estudios y aplicaciones prácticas.

Con el nacimiento y estandarización de ROS (Quigley et al., 2009) y MoveIt (Coleman et al., 2014), OMPL se integró como un componente clave para la planificación de movimiento en robots. Esta integración ha permitido a los desarrolladores utilizar los planificadores de OMPL dentro de aplicaciones robóticas más amplias, facilitando la creación de sistemas robóticos complejos que requieren planificación de movimiento en tiempo real.

MoveIt, el principal *framework* de planificación de movimiento en ROS para manipuladores, ha reforzado aún más el papel de los planificadores basados en muestreo en robótica académica e industrial. Su modularidad permite adaptar

el *pipeline* de planificación a casos de uso que abarcan desde tareas sencillas de *pick-and-place* hasta manipulaciones coordinadas con dos brazos y reconfiguraciones dinámicas de la escena.

En el ámbito industrial, este tipo de planificadores se integran dentro de *frameworks* como ROS-Industrial (ROS-I) (Santos et al., 2020), que extiende ROS para su uso industrial, permitiendo soluciones flexibles e independientes del hardware en diversos sectores manufactureros. (Fresnillo et al., 2023) presentaron extensiones críticas a MoveIt para implementar cambiadores de herramientas automáticos, control de trayectoria del efector final con alta precisión y manipulaciones dual-arm coordinadas, validadas en un manipulador industrial de dos brazos. (Martinez et al., 2017) documentaron el despliegue de un robot Motoman de dos brazos en un entorno industrial dinámico, subrayando la practicidad de configurar manipuladores complejos con ROS-I.

3.3 Planificación basada en optimización local

A diferencia de los planificadores basados en muestreo, que exploran el espacio de configuración mediante una búsqueda aleatoria, los métodos de planificación por optimización abordan el problema como un proceso iterativo de refinamiento. Estos algoritmos comienzan con una trayectoria inicial (que puede ser generada por un planificador de muestreo o definida manualmente) y la ajustan para minimizar una función de coste que incorpora criterios como suavidad, longitud del camino y prevención de colisiones. Este enfoque es especialmente útil en entornos donde se requiere precisión y suavidad en las trayectorias, como en la manipulación robótica o el control de vehículos autónomos.

Los planificadores de muestreo óptimos que garantizan encontrar una solución global óptima, que puede requerir de un tiempo de exploración considerable, necesitan un postprocesado para: eliminar nodos redundantes, suavizar la trayectoria y calcular los perfiles de velocidad y aceleración.

Los planificadores centrados en optimización local se centran en mejorar una trayectoria inicial mediante un proceso iterativo de ajuste, minimizando una función de coste que incluye términos para suavidad, longitud y restricciones dinámicas. Este enfoque permite generar trayectorias más suaves y di-

námicamente viables de forma directa, aunque no garantiza encontrar la solución óptima global. La calidad de la trayectoria inicial es crucial, ya que un mal punto de partida puede llevar a mínimos locales no factibles. Los planificadores basados en optimización son especialmente útiles en tareas que requieren trayectorias suaves y precisas, como el montaje robótico, el pick-and-place de alta velocidad y el procesamiento de superficies.

CHOMP (Ratliff et al., 2009) introdujo el uso de gradientes funcionales para equilibrar suavidad y prevención de colisiones. CHOMP minimiza un coste que penaliza velocidades, aceleraciones y cambios bruscos, deformando iterativamente la trayectoria inicial para evitar colisiones. Este enfoque ha demostrado ser efectivo en entornos complejos, logrando trayectorias suaves y libres de colisión con alta eficiencia. Una de las características importantes de CHOMP es que incluso si la trayectoria inicial es infactible, el algoritmo puede corregir las colisiones mediante deformaciones iterativas, lo que lo hace robusto en entornos con obstáculos complicados. Sin embargo, CHOMP no garantiza encontrar una solución en todos los casos, ya que es un método puramente local y puede atascarse en mínimos locales.

STOMP (Kalakrishnan et al., 2011) es otro pionero en la optimización de trayectorias, que se diferencia de CHOMP al no depender del cálculo de gradientes analíticos. En lugar de seguir directamente la dirección de descenso más pronunciado, STOMP realiza una búsqueda estocástica alrededor de la trayectoria actual, generando múltiples trayectorias perturbadas y evaluando sus costes. Este enfoque permite manejar funciones de coste no diferenciables o con múltiples óptimos locales complicados, lo que lo hace adecuado para problemas con restricciones dinámicas complejas. STOMP ha demostrado ser efectivo en la planificación de trayectorias suaves y con prevención de colisiones, aunque puede requerir más evaluaciones de la función de coste para lograr convergencia.

TrajOpt (Schulman et al., 2014) es un enfoque que formula la planificación como un problema de optimización no lineal con restricciones, resolviendo subproblemas convexos aproximados. TrajOpt inicializa una trayectoria y luego itera resolviendo problemas de optimización cuadrática, modelando las colisiones con funciones convexas de penalización. Este enfoque permite realizar “saltos”

más informados en el espacio de trayectoria que un simple descenso de gradiente, logrando converger rápidamente a trayectorias libres de colisiones y de corta longitud. TrajOpt ha demostrado ser eficiente en la planificación de trayectorias para manipuladores industriales.

K-Order Markov Optimization (KOMO) (Toussaint, 2017) es un enfoque que permite incluir restricciones de orden superior en el problema de planificación, optimizando una trayectoria que cumple múltiples criterios simultáneamente. Aunque KOMO es poderoso en términos de expresividad, su complejidad computacional puede ser un obstáculo en entornos con muchos obstáculos o pasajes estrechos. Las aplicaciones de KOMO se limitan a problemas donde se conoce bastante sobre la tarea, ya que permite incluir ese conocimiento como término de coste o restricción rígida. Sin embargo, su uso en la práctica suele requerir combinaciones con otros métodos para mejorar su rendimiento.

Gaussian Process Motion Planner (GPMP) (Mukadam et al., 2018) introduce los procesos gaussianos en la planificación de trayectorias, modelando la trayectoria continua del robot como una realización de un proceso gaussiano. Este enfoque permite trabajar en "tiempo continuo" sin discretizar en exceso, y calcular gradientes analíticos de métricas como longitud de arco o cercanía a obstáculos. GPMP formula la planificación como un problema de inferencia en un factor graph probabilístico, logrando una alta tasa de éxito en la planificación de trayectorias suaves y libres de colisiones.

En conjunto, CHOMP, STOMP, TrajOpt, KOMO, GPMP y variantes representan una visión distinta a la de los planificadores de muestreo: en lugar de construir caminos explorando al azar el espacio, parten de una solución inicial tentativa y la pulen iterativamente mediante criterios de optimización. Su adopción en robótica industrial ha sido paulatina pero constante en la última década. *Frameworks* como MoveIt integran actualmente planificadores basados en optimización (CHOMP y STOMP desde sus primeras versiones, y más recientemente TrajOpt como plugin), lo que permite a los ingenieros aprovechar trayectorias más suaves en robots manipuladores de 6 GDL con un esfuerzo mínimo de parametrización. Aun así, en escenarios de extrema complejidad geométrica o topológica, los métodos de muestreo siguen siendo un respaldo valioso para encontrar un camino inicial que luego pueda refinarse. Tendencias recientes en

investigación tratan de unir ambas filosofías o de superarlas mediante aprendizaje: por un lado, combinar muestreo y optimización en algoritmos híbridos (Kamat et al., 2022); por otro, aprender de experiencias previas para inicializar mejor los optimizadores de trayectoria.

3.4 Métodos de planificación híbridos

Una tendencia reciente en planificación de movimiento es combinar algoritmos globales basados en muestreo con algoritmos locales de optimización de trayectorias, aprovechando las fortalezas de cada enfoque. En esencia, un planificador híbrido primero calcula rápidamente una trayectoria factible (aunque subóptima) mediante un método de muestreo probabilístico, y a continuación aplica un refinamiento mediante un optimizador local, típicamente CHOMP, STOMP o TrajOpt, para mejorar la suavidad, reducir la longitud o incrementar los márgenes de seguridad de esa trayectoria.

Este esquema en dos etapas de planificación permite sortear las limitaciones de cada técnica por separado: el planificador global asegura viabilidad aunque el espacio de configuración sea complicado, mientras que la optimización local pule la solución para acercarla a un óptimo de calidad y la parametriza para cumplir con restricciones dinámicas o de suavidad. En las últimas integraciones en el *framework* MoveIt es habitual encadenar un planificador de muestreo de OMPL (como RRT-Connect) con un post-procesador tipo CHOMP o STOMP; el planificador OMPL encuentra una ruta inicial libre de colisiones y luego el adaptador de trayectoria aplica CHOMP/STOMP para refinarla antes de ejecutarla (Liu and Liu, 2022).

La principal ventaja de este enfoque radica en mitigar mínimos locales y mejorar la calidad sin sacrificar demasiada eficiencia: algoritmos como CHOMP pueden atascarse si la trayectoria inicial no es buena, pero al partir de una ruta factible calculada por muestreo se reduce drásticamente ese riesgo. De hecho, estudios reportan que al optimizar trayectorias iniciales de OMPL con CHOMP o STOMP se obtienen caminos más cortos y suaves sin afectar la tasa de éxito de planificación (Liu and Liu, 2022).

En otras palabras, el *pipeline* híbrido logra trayectorias de mayor calidad a costa de un ligero aumento en el tiempo de cómputo, cumpliendo con las exigencias industriales de movimientos más óptimos y seguros sin dejar de hallar soluciones factibles en entornos complejos (Dai et al., 2018). Esta técnica, si bien reciente, se ha consolidado rápidamente en la práctica. En concreto, MoveIt incluye de serie adaptadores para enlazar planificadores OMPL con optimizadores (como *CHOMPOptimizerAdapter* y *STOMPOptimizerAdapter*), permitiendo refinar automáticamente cada trayectoria planificada (Liu and Liu, 2021).

Además de los *pipelines* secuenciales, se han desarrollado algoritmos híbridos integrados que combinan muestreo y optimización de forma más acoplada durante la planificación. Un ejemplo pionero es Regionally Accelerated BIT* (RABIT*) propuesto por (Choudhury et al., 2016), que integra el planificador óptimo por muestreo BIT* con pasos locales de CHOMP (Kim and Yoon, 2020). En RABIT*, cada vez que BIT* detecta una región difícil de explorar (por ejemplo, un paso estrecho en el espacio de configuración), invoca a CHOMP para optimizar localmente una trayectoria parcial, incorporando esa información de vuelta al árbol de búsqueda. Este enfoque demostró acelerar la convergencia hacia soluciones de menor coste manteniendo la optimización asintótica de BIT*, al combinar la exploración global con mejoras locales informadas. Como limitación, RABIT* asume tener precomputado un campo de distancia (mapa de potencial de obstáculos) para guiar la optimización, lo cual puede ser costoso o inviable en escenarios de alta complejidad geométrica.

Investigaciones posteriores han refinado esta idea eliminando dependencias de preprocesamiento y ampliando su aplicabilidad. Recientemente, (Kamat et al., 2022) presentaron BITKOMO, un planificador híbrido que integra BIT* con el método de optimización secuencial KOMO. BITKOMO actúa en modo *anytime*: alterna iterativamente entre expandir el árbol y refinar la mejor trayectoria actual mediante KOMO, de modo que cada iteración mejora el coste de la solución sin perder la garantía de convergencia óptima global heredada de BIT*.

En pruebas con manipuladores de hasta 14 GDL (dos brazos de 7 GDL) en entornos con pasajes estrechos, BITKOMO superó a los métodos individuales: halló caminos factibles aun cuando KOMO fracasaba, y convergió más rápida-

mente hacia la solución óptima que BIT* ejecutado por sí solo (Kamat et al., 2022).

Esto evidencia el potencial de sinergia entre ambas técnicas, logrando soluciones de alta calidad de forma más robusta y eficiente. Otra línea de trabajo híbrida combina planificadores globales multi-consulta con optimización continua durante la ejecución. Por ejemplo, (Dai et al., 2018) integran un *roadmap* disperso (una variante de PRM) con un optimizador de trayectoria en tiempo real: primero construyen un grafo global mínimo en el espacio libre, y luego, para cada consulta, extraen una ruta inicial del grafo y la refinan en línea mientras el robot se mueve. En sus experimentos con miles de escenarios, esta combinación superó tanto a planificadores de muestreo convencionales como a planificadores puramente locales en términos de tiempo total y calidad de trayectoria.

Este enfoque híbrido logra así planificaciones muy rápidas y adaptativas, reutilizando conocimiento previo (las estructuras de muestreo) pero sin quedar limitado por ellas, ya que el optimizador puede reajustar la trayectoria en respuesta a perturbaciones.

En resumen, los métodos de planificación híbridos representan el estado del arte para manipuladores: aprovechan la exploración aleatoria para garantizar factibilidad en espacios complejos, a la vez que incorporan refinamiento numérico para obtener trayectorias de calidad casi óptima. Su adopción en la industria ha sido rápida gracias a *frameworks* abiertos como MoveIt, que facilitan su uso sin necesidad de implementación desde cero.

3.5 Aprendizaje automático para planificación de trayectorias

En los últimos años, la planificación de trayectorias para manipuladores robóticos ha incorporado de manera creciente herramientas de aprendizaje automático, superando las limitaciones de los métodos puramente geométricos o probabilísticos. Dentro de este amplio campo, dos familias de técnicas han adquirido especial relevancia:

1. Aprendizaje supervisado, que aprovecha conjuntos de trayectorias ejemplo (obtenidas mediante demostraciones humanas o generadas *offline* por planificadores óptimos) para entrenar modelos capaces de predecir, ante un nuevo entorno y configuración inicial/meta, una trayectoria factible o las acciones más adecuadas. Estas redes aprenden a imitar comportamientos óptimos, ofreciendo tiempos de respuesta muy reducidos una vez entrenadas, aunque su calidad y robustez dependen de la cobertura y diversidad de los datos de entrenamiento.

2. Aprendizaje por refuerzo, en el que el manipulador se concibe como un agente que interactúa con su entorno recibiendo recompensas o penalizaciones (por alcanzar el objetivo, penalizaciones por colisión o por consumo energético, etc.) y, a través de ensayo y error, aprende una política capaz de generar trayectorias eficientes y seguras. Gracias a la capacidad de optimizar criterios complejos de manera directa y de adaptarse a cambios dinámicos, los métodos de refuerzo han demostrado un gran potencial para enfrentar espacios de alta dimensión y entornos inciertos, superando en muchos casos el desempeño de planificadores clásicos.

3. Enfoques híbridos, que combinan lo mejor de ambos mundos: por ejemplo, usar redes entrenadas de manera supervisada para guiar o acelerar a un planificador clásico (RRT*, PRM) hacia regiones prometedoras del espacio, o bien inicializar políticas de refuerzo con datos de demostración para mejorar la eficiencia de aprendizaje y la seguridad. Estas arquitecturas mixtas han demostrado mejorar tanto la calidad de la trayectoria como la velocidad de planificación, integrándose con facilidad en ecosistemas robóticos existentes.

En contraste, las técnicas de aprendizaje no supervisado apenas han penetrado en la práctica de la planificación de trayectorias industrial, al no proporcionar por sí mismas un mecanismo explícito para garantizar viabilidad o evitar colisiones. Por ello, este apartado se centrará en detallar primero los enfoques de aprendizaje supervisado y a continuación profundizará en las arquitecturas y algoritmos de aprendizaje por refuerzo que hoy constituyen la frontera de la investigación en planificación de trayectorias para robots manipuladores.

3.5.1 **Aprendizaje supervisado para planificación de trayectorias**

El aprendizaje supervisado emplea datos etiquetados (por ejemplo trayectorias ejemplo proporcionadas por un experto o generadas por un planificador óptimo) para entrenar modelos que asistan en la planificación. En este enfoque, un conjunto de situaciones de entrada se asocia a salidas deseadas y el modelo aprende esta correspondencia a partir de muchos ejemplos (Wang et al., 2021b). Típicamente, en robótica las entradas son representaciones del estado del manipulador y su entorno (nubes de puntos, imágenes, etc.), mientras que las salidas pueden ser trayectorias discretas o acciones continuas a tomar en cada paso.

Estos métodos han logrado entrenar manipuladores en tareas específicas usando grandes cantidades de datos etiquetados, demostrando capacidad de resolver problemas complejos de planificación (Wang et al., 2021b). Un ejemplo prominente es el aprendizaje por imitación (Song et al., 2020; Ross et al., 2011), donde el robot aprende a reproducir trayectorias demostradas por un humano o por otro planificador. Técnicas de imitación directa (clonado de comportamiento) entrenan redes neuronales para mapear estados a acciones replicando un conjunto de demostraciones expertas. Sin embargo, este enfoque puede enfrentar problemas de generalización fuera de las condiciones de entrenamiento. Para mitigar esto, existen esquemas iterativos como Dataset Aggregation (DAgger) (Gleave et al., 2022), que incorporan realimentación durante el entrenamiento. Otros enfoques basados en imitación incluyen Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016), que utiliza redes generativas para aprender políticas que imitan el comportamiento experto.

Otra aplicación del aprendizaje supervisado en planificación es el aprendizaje de funciones de coste o heurísticas. Aquí el modelo aprende a predecir, a partir de características del estado o del entorno, el *costo-to-go* o la viabilidad de ciertos movimientos, guiando así a un planificador tradicional. Por ejemplo, (Pfeiffer et al., 2017) entrenan una red neuronal convolucional para predecir un mapa de coste a partir de sensores, ayudando a planificar caminos libres de colisión para robots móviles. De manera análoga, en manipuladores se han aprendido heurísticas de búsqueda en el espacio de configuraciones, asignando costes elevados a configuraciones cercanas a colisiones para que algoritmos como A*

o RRT* planeen más eficientemente. Un caso de éxito representativo del aprendizaje supervisado es Motion Planning Networks (MPNet) (Qureshi et al., 2019). MPNet es una arquitectura de red neuronal entrenada con ejemplos de trayectorias libres de colisión generadas por planificadores tradicionales. Dada una representación del entorno y un estado inicial y meta, MPNet codifica el entorno y predice directamente una trayectoria factible para el manipulador. Este enfoque híbrido supervisado logró generalizar a entornos no vistos y generar caminos en tiempo inferior al segundo, superando significativamente en velocidad a planificadores convencionales. La principal fortaleza de métodos como MPNet es la eficiencia computacional una vez entrenados (responden muy rápido ante nuevas consultas) y su capacidad para aprender a evitar obstáculos de forma implícita. Además, al entrenar con una variedad de escenarios, pueden generalizar a configuraciones distintas de las vistas en el conjunto de entrenamiento.

Las principales limitaciones de los enfoques supervisados están ligadas al gran volumen de datos de calidad que necesitan. Obtener trayectorias de entrenamiento puede ser costoso, ya sea recabando demostraciones humanas o ejecutando planificadores intensivos fuera de línea para generar ejemplos. También enfrentan problemas de extrapolación: si durante la ejecución real el robot sale del espacio de estados cubierto por los datos de entrenamiento, la política aprendida puede fallar de manera catastrófica. Aunque técnicas de agregación de datos y aleatorización de dominio (*domain randomization*) (Tobin et al., 2017a) mitigan esto, sigue siendo un desafío lograr cobertura de todos los casos relevantes. Adicionalmente, los modelos supervisados aprenden a imitar un criterio fijo dado por los datos; por tanto, su desempeño está limitado por la calidad del experto o planificador que generó las trayectorias de referencia. No tienen, intrínsecamente, un mecanismo de mejora más allá de las demostraciones, a diferencia del aprendizaje por refuerzo que optimiza según una métrica de recompensa.

3.5.2 **Aprendizaje por refuerzo para planificación de trayectorias**

A diferencia del aprendizaje supervisado, donde un modelo imita trayectorias o costes predefinidos, el aprendizaje por refuerzo (Reinforcement Learning (RL))

aborda la planificación de movimiento como un problema de toma de decisiones secuenciales (Sutton and Barto, 2018). En la figura 3.7 se muestra la evolución de las publicaciones científicas relacionadas con el uso de aprendizaje por refuerzo en planificación de trayectorias, evidenciando el creciente interés en esta área desde 2018.

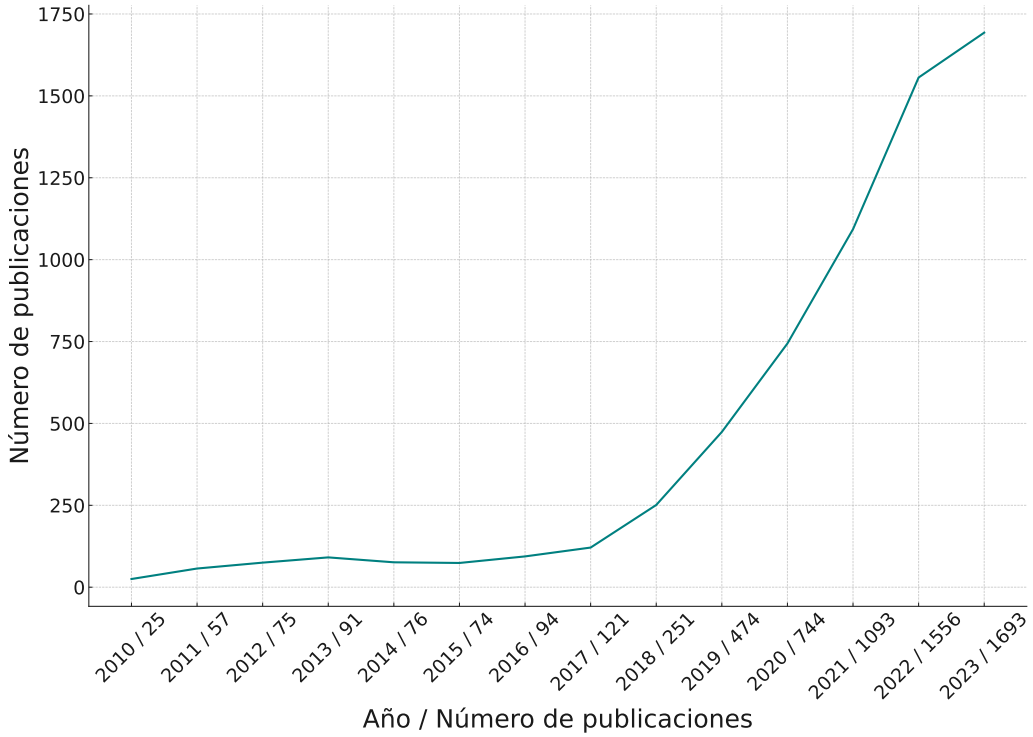

Figura 3.7: Evolución anual del número de publicaciones relacionadas con el uso de aprendizaje por refuerzo en planificación de movimientos. Los términos de búsqueda utilizados fueron: ((motion AND planning) OR (path AND planning) OR (trajectory AND planning) OR (collision AND avoidance)) AND (reinforcement AND learning). Se observa un crecimiento exponencial desde 2018, lo que refleja el creciente interés de la comunidad científica en la aplicación de técnicas de aprendizaje por refuerzo en tareas de planificación de trayectorias robots. (Elguea-Aguinaco et al., 2024)

En RL, el manipulador se modela como un agente que interactúa con un entorno, típicamente el robot y sus obstáculos, recibiendo recompensas por completar objetivos y penalizaciones por comportamientos indeseados. En este contexto, las recompensas pueden estar asociadas a alcanzar metas, evitar colisio-

nes, minimizar el tiempo de ejecución o reducir el consumo energético. El agente aprende a maximizar la recompensa acumulada a largo plazo mediante exploración y explotación de acciones en diferentes estados del entorno.

El objetivo del agente es aprender una política, es decir, la función que decide la acción a tomar en cada estado, que maximice la recompensa acumulada a largo plazo. Formulado típicamente como un Proceso de Decisión de Markov (MDP) (Puterman, 1994), el RL permite que el robot descubra por sí mismo trayectorias óptimas mediante ensayo y error, en lugar de depender de cálculos geométricos explícitos.

En este contexto, el aprendizaje profundo por refuerzo (Deep Reinforcement Learning (DRL)) ha emergido como una técnica poderosa para la planificación de trayectorias en robótica. Al combinar redes neuronales profundas con algoritmos de RL, los sistemas DRL pueden manejar espacios de estado y acción de alta dimensionalidad, logrando aprender representaciones complejas del entorno y optimizando políticas que generan trayectorias efectivas (Mnih et al., 2015b).

Existen dos enfoques principales en DRL: el basado en valores y el basado en políticas. En el primero, se aprende una función de valor que estima la recompensa esperada para cada acción en un estado dado, y se utiliza para guiar la selección de acciones. En el segundo, se entrena directamente una política que mapea estados a acciones sin necesidad de calcular valores intermedios. Los métodos basados en políticas suelen ser más eficientes en espacios de acción continuos, como los manipuladores robóticos (Lillicrap et al., 2015).

En la práctica, la mayoría de los avances recientes en DRL se basan en esquemas actor-crítico. Estos algoritmos combinan lo mejor de los enfoques basados en valores y en políticas: el actor representa la política que selecciona acciones, mientras que el crítico estima el valor esperado de esas acciones, proporcionando un aprendizaje más estable y eficiente. Esta arquitectura permite entrenar políticas en espacios de acción continuos y de alta dimensión, como los que presentan los manipuladores industriales, y ha demostrado ser especialmente robusta frente a la inestabilidad y la alta varianza que afectan a otros métodos.

En el marco de la robótica industrial, el DRL se ha consolidado como un enfoque eficaz para la planificación de trayectorias, especialmente en tareas que

demandan generalización, adaptabilidad y reactividad en tiempo real. A través de la optimización de políticas que mapean observaciones de alta dimensionalidad a acciones mediante interacción de prueba y error con el entorno, los métodos de DRL son capaces de aprender estrategias de control complejas que resultan difíciles de especificar de forma explícita o de resolver analíticamente (Sutton and Barto, 2018). Esta característica los hace idóneos para escenarios de manipulación robótica en entornos no estructurados o parcialmente observables, tales como el *picking* en almacenes, ensamblaje dinámico y colaboración humano-robot (Lillicrap et al., 2015; Kalashnikov et al., 2018).

Una vez entrenados, los planificadores basados en DRL generan trayectorias en tiempo de inferencia sin necesidad de resolver explícitamente un problema de planificación en línea, en contraste con los enfoques basados en muestreo u optimización, que normalmente requieren recalcular una solución para cada consulta de planificación. Algoritmos como Soft Actor-Critic (SAC) (Haarnoja et al., 2018), Proximal Policy Optimization (PPO) (Schulman et al., 2017) y Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018) han demostrado su eficacia en el control continuo de robots, abarcando tareas de alcance, agarre, inserción y seguimiento de trayectoria (Jurgenson and Tamar, 2019; Zhang et al., 2023; Ha et al., 2020; Tamizi et al., 2023; Lillicrap et al., 2015). Estos métodos aprovechan redes neuronales para la aproximación de funciones y escalan a espacios de acción de alta dimensionalidad, como el control en el espacio de configuración de manipuladores de 6 o 7 GDL.

Diversos estudios han validado la viabilidad de la planificación de trayectorias basada en DRL para robots industriales. Por ejemplo, se han entrenado políticas para resolver tareas que permiten alcanzar metas en el espacio de trabajo con estrategias de recompensas dispersas (Kalashnikov et al., 2018) o incluso ejecutar movimientos en el espacio de trabajo con restricciones de orientación específicas del efector final. Estas investigaciones suelen incorporar estrategias de aprendizaje basadas en *curriculum*, diseño de funciones de recompensa avanzadas para guiar el entrenamiento e incluso inyectan demostraciones de expertas generadas por planificadores clásicos para acelerar el aprendizaje en escenarios donde la recompensa se obtiene de manera dispersa (Schulman et al., 2017). Esta capacidad de integrar conocimiento previo posiciona al

DRL dentro de las arquitecturas híbridas de planificación, ya sea como planificador global, política de inicialización o módulo de control local (Johannink et al., 2019).

Una técnica particularmente efectiva en entornos con recompensas escasas es el Hindsight Experience Replay (HER) (Andrychowicz et al., 2018), que permite al agente reinterpretar experiencias fallidas como exitosas al redefinir el objetivo alcanzado como si hubiese sido el deseado. Esta estrategia aumenta significativamente la eficiencia del aprendizaje al densificar la señal de recompensa, y ha sido aplicada con éxito en tareas de manipulación como *pick-and-place*, empuje e inserción. HER resulta especialmente útil en planificación de trayectorias con objetivos en el espacio de trabajo difíciles de alcanzar inicialmente, donde una señal de éxito binaria es insuficiente para guiar el aprendizaje mediante métodos convencionales.

El DRL resulta especialmente indicado para problemas donde los planificadores tradicionales encuentran dificultades, como en dinámicas parcialmente observadas o restricciones no convexas. Además, las políticas de DRL producen salidas deterministas en tiempo de inferencia, lo que es ventajoso en aplicaciones donde la latencia y la repetibilidad son críticas (Peng et al., 2018). Recientes comparativas muestran que, una vez entrenados, los planificadores basados en DRL pueden igualar o superar a los clásicos tanto en tiempo de planificación como en suavidad de la trayectoria, especialmente en entornos dinámicos estructurados (Liu et al., 2024).

No obstante, la adopción del DRL en entornos industriales está limitada por varios retos. El más destacado es la eficiencia de muestreo: las políticas suelen requerir millones de interacciones con el entorno para converger. A esto se suma la brecha entre simulación y entorno físico, conocida como el problema *sim-to-real*, donde las discrepancias entre los entornos pueden provocar fallos en la transferencia de las políticas entrenadas. Para mitigar estos problemas, se han propuesto técnicas como *domain randomization* (Tobin et al., 2017b) y refinamiento con demostraciones reales, aunque su implantación en entornos de manufactura con restricciones temporales sigue siendo un desafío abierto (Liu et al., 2022).

A pesar de estas limitaciones, el DRL continúa ganando terreno como complemento (no sustituto) de los métodos clásicos. Su compatibilidad con arquitecturas híbridas y su capacidad de aprender de la experiencia lo convierten en una herramienta prometedora para la próxima generación de sistemas robóticos.

3.6 Marcos y herramientas de comparación y evaluación de planificación de trayectorias

La comunidad robótica ha desarrollado diversas infraestructuras experimentales y entornos estandarizados de benchmarking para evaluar planificadores de movimiento en robots manipuladores. La propia OMPL incluye una suite de algoritmos de planificación por muestreo y un módulo nativo para benchmarking.

Inicialmente, (Cohen et al., 2012) propusieron una infraestructura genérica de benchmarking de planificadores, pionera en el ecosistema ROS. Esta infraestructura sentó las bases del módulo de benchmarking en la conocida plataforma MoveIt, lanzada poco después. Desde sus inicios incorporó un paquete de benchmarking capaz de ejecutar múltiples planificadores sobre una misma escena y recopilar métricas en una base de datos

Después, (Moll et al., 2015) presentaron una infraestructura extensible integrada en OMPL que permite comparar hasta 29 planificadores distintos bajo un marco unificado. Esta plataforma soporta diferentes tipos de problemas de planificación, formatos estándar de registro de resultados y una herramienta interactiva de visualización de datos (OMPL Planner Arena) para analizar métricas de desempeño. En su artículo, (Moll et al., 2015) detallan consideraciones importantes para un benchmarking riguroso, como ejecutar múltiples repeticiones debido a la variabilidad intrínseca de los algoritmos aleatorios, y seleccionar un conjunto representativo de escenarios de prueba.

Estos avances culminan en la creación de marcos robustos, adoptados en herramientas consolidadas como OMPL y MoveIt. Por una parte, OMPL Planner Arena permite comparar planificadores de muestreo en un entorno unificado, facilitando la visualización y análisis de resultados. Por otra parte, MoveIt

Benchmark Suite ha evolucionado para evaluar tanto planificadores a nivel de trayectoria como de tarea, permitiendo pruebas comparativas *whole-stack* que abarcan desde el rendimiento de bajo nivel hasta el éxito en tareas complejas de manipulación.

Estos marcos han sido ampliamente adoptados y ampliados por la comunidad. Por ejemplo, la suite de benchmarking de OMPL ha sido utilizada en múltiples estudios comparativos, como el de (Liu and Liu, 2022), que comparó planificadores clásicos concatenados con planificadores basados en optimización local en un entorno de manipulación industrial. Estos estudios han demostrado la utilidad de los marcos para obtener conclusiones generalizables sobre el rendimiento de los algoritmos en diferentes escenarios.

Trabajos como el de (Chamzas et al., 2021) han introducido herramientas especializadas como MotionBenchMaker, que permite generar conjuntos de pruebas diversificados y reproducibles para evaluar planificadores en escenarios de manipulación. Esta herramienta facilita la incorporación de nuevos robots y entornos, generando problemas de planificación con dificultad controlada. A diferencia de los marcos anteriores, MotionBenchMaker se centra en la generación de datasets estandarizados, permitiendo comparativas más coherentes y generalizables. Un hallazgo importante de su trabajo es que ningún algoritmo de muestreo dominó consistentemente en todos los casos de prueba, subrayando la necesidad de disponer de múltiples escenarios para obtener conclusiones generalizables.

Por otro lado, herramientas como Robowflex (Kingston and Kavraki, 2022) han surgido para simplificar la configuración y ejecución de comparativas sobre MoveIt. Esta herramienta permite ejecutar comparativas sin necesidad de contar con un sistema basado en ROS, soportando variaciones automáticas de escenarios y definición de métricas personalizadas. Un aspecto destacado es que Robowflex almacena resultados en ficheros simples y puede ejecutarse en instancias paralelas, lo que agiliza la evaluación extensiva de planificadores incluyendo métodos de aprendizaje que requieren gran cantidad de muestras.

Recientemente han surgido algunos trabajos pioneros que comparan planificadores tradicionales con nuevos enfoques basados en aprendizaje automático, aunque dichos estudios son aún muy limitados en número. Por ejemplo, se

ha demostrado que integrar muestreadores, que codifican el entorno mediante redes neuronales profundas, en algoritmos clásicos (como MPNet acoplado a RRT*) permite generar trayectorias más cortas y suaves en comparación con el muestreo aleatorio puro (Qureshi and Yip, 2018). Estos enfoques basados en aprendizaje pueden reducir la longitud de las trayectorias y mejorar su suavidad manteniendo tiempos de cómputo competitivos. Sin embargo, incluso estos métodos de aprendizaje no garantizan un rendimiento superior en todos los casos. De hecho, se ha observado que algunos planificadores de aprendizaje pueden rendir peor que los métodos clásicos en determinados escenarios, requiriendo procesos adicionales como el reentrenamiento de la red para mejorar su precisión.

En general, las comparativas extensivas entre planificadores tradicionales y basados en aprendizaje son aún escasas, muy recientes y poco exhaustivas. La mayoría de trabajos previos tienden a enfocarse en unos pocos algoritmos o en escenarios limitados, ofreciendo una visión fragmentada del panorama

La literatura actual apenas comienza a explorar estas comparativas tradicionales con las basadas en aprendizaje, subrayando la necesidad de investigaciones más completas y rigurosas aplicadas a robots manipuladores industriales para determinar las fortalezas y debilidades relativas de cada enfoque.

3.7 **Resumen y conclusiones**

En conjunto, la evolución histórica y las metodologías analizadas ponen de manifiesto que no existe un planificador único que domine en todos los aspectos, sino que cada familia de algoritmos presenta fortalezas y limitaciones compensatorias (como resume la Tabla 3.1). Los métodos clásicos de planificación geométrica basados en muestreo, como RRT o PRM, han demostrado ser herramientas generales y eficientes para hallar trayectorias factibles en espacios de alta dimensión. Su carácter probabilístico completo asegura que encontrarán una solución si esta existe (dado suficiente tiempo), y en la práctica encuentran trayectorias geométricas válidas con rapidez incluso en robots de 6 GDL. Esto explica su adopción masiva en la industria a través de librerías estándar (OMPL/MoveIt).

Sin embargo, su principal debilidad es la calidad subóptima de las trayectorias obtenidas: algoritmos como RRT tienden a converger a soluciones viables pero lejos del óptimo, produciendo rutas más largas o con giros bruscos respecto a la trayectoria ideal. Esta deficiencia obliga a pasos de post-procesado, como atajos o suavizados, para pulir las trayectorias antes de ejecutarlas. Otra limitación inherente es su naturaleza estática: los planificadores geométricos generan una ruta fija sin capacidad de adaptarse a cambios repentinos; si el entorno varía o aparece un obstáculo inesperado, la trayectoria se debe replantear desde cero, lo que dificulta su uso directo en entornos altamente dinámicos o impredecibles.

Adicionalmente, aunque son relativamente robustos a la dimensionalidad, no garantizan resultados óptimos ni dinámicamente viables. Las variantes óptimas, como RRT* o PRM*, garantizan convergencia hacia la trayectoria de menor coste, pero a cambio de un cómputo mayor: requieren muchas más muestras y conexiones. En escenarios industriales donde prima la rapidez, un planificador estándar (RRT-Connect, PRM) suele preferirse para obtener una solución válida, sacrificando la optimización de la misma; mientras que en tareas repetitivas donde la calidad del recorrido impacta en ciclos de producción, puede justificarse invertir más cálculo en rutas más cortas y suaves. En cualquier caso, los planificadores basados en muestreo no explotan conocimiento previo del problema (más allá de modelos geométricos básicos): cada consulta se aborda desde cero (especialmente en los métodos mono-consulta), sin aprendizaje o adaptación entre ejecuciones.

Por otro lado, los métodos de planificación por optimización local (CHOMP, STOMP, TrajOpt, etc.) abordan la generación de trayectorias como un problema de refinamiento iterativo, lo que les confiere ventajas complementarias. Una vez disponen de una trayectoria inicial factible, son capaces de mejorarla sustancialmente según métricas de coste definidas: producen trayectorias más cortas, suaves y con mayores márgenes de seguridad que las obtenidas por planificadores de muestreo estándar. De hecho, penalizando aceleraciones y distancias a obstáculos en la función de coste, estos optimizadores generan movimientos que reducen vibraciones y tiempos muertos, alineándose con criterios industriales de eficiencia energética y precisión.

Además, incorporan de forma natural restricciones cinemáticas y dinámicas, devolviendo rutas ejecutables por el robot sin necesidad de ajustes posteriores en el controlador. No obstante, sus limitaciones son notorias: al ser técnicas puramente locales, necesitan una trayectoria inicial razonablemente buena para evitar quedar atrapados en mínimos locales. Si la trayectoria inicial proporcionada atraviesa un obstáculo, el algoritmo no tiene mecanismo para escapar de ese mínimo local. Además, son altamente sensibles a las condiciones iniciales: la calidad y viabilidad del resultado dependen críticamente de la ruta de partida. Por ejemplo, CHOMP puede atascarse en un óptimo local de coste si la trayectoria inicial discurre cerca de un valle estrecho entre obstáculos. Adicionalmente, ajustar los pesos y parámetros de coste (por ejemplo cuánta suavidad en relación a separación a obstáculos) requiere cierto conocimiento experto, para equilibrar objetivos sin provocar comportamientos indeseados (como desviarse excesivamente para evitar colisiones).

Pese a ello, cuando se aplican sobre buenas trayectorias base, estos métodos destacan por sus trayectorias de alta calidad, con poco o ningún post-procesado adicional. Dada la naturaleza complementaria de ambos enfoques, era natural que surgieran métodos híbridos que combinasen planificación global y optimización local. La idea fundamental es aprovechar un planificador basado en muestreo para evitar colisiones y obtener rápidamente alguna ruta factible, y acto seguido utilizar un optimizador para pulir esa ruta en términos de longitud, suavidad y viabilidad dinámica.

Este esquema de *pipelines* ha probado ser muy eficaz: estudios comparativos han mostrado que las trayectorias resultantes son significativamente más cortas y suaves que las originales, sin reducir apreciablemente la tasa de éxito ni aumentar en exceso el tiempo de cómputo (Liu and Liu, 2022). En la Tabla 3.1 se refleja esta fortaleza doble de la planificación híbrida, recibiendo valoraciones altas tanto en “resultado óptimo” como en “robustez”. En esencia, el planificador global garantiza no quedarse atascado (pues siempre que exista algún camino lo encontrará con probabilidad uno con suficientes muestras), y el refinador local garantiza que la solución final sea lo más cercana posible al óptimo y respete las restricciones de la tarea. La contrapartida es una mayor complejidad de implementación y una penalización temporal al añadir la fase de optimización.

También es cierto que, aunque estas arquitecturas mejoran la adaptación frente a cambios (pueden replanificar globalmente y ajustar localmente casi en tiempo real), siguen reaccionando de forma discreta: ante una modificación brusca del entorno, típicamente se aborta la trayectoria en curso y se lanza de nuevo el *pipeline* completo (o se conmutan ambos planificadores en modo online de manera recurrente (Chamzas et al., 2021)).

Esto implica que, en entornos altamente dinámicos, la planificación híbrida funciona mejor con replanificaciones frecuentes que aseguren una actualización continua de la trayectoria, pero aún no logra una adaptación continua totalmente fluida ante perturbaciones. La necesidad de una mayor adaptabilidad y autonomía ha empujado recientemente a la comunidad hacia enfoques basados en aprendizaje automático, con especial énfasis en el DRL.

Estas técnicas afrontan la planificación de movimientos desde una óptica distinta: en lugar de trazar caminos geométricos explícitamente, entrenan una política (típicamente una red neuronal) para que el esta sus acciones de movimiento en cada estado del robot, optimizando a largo plazo criterios de éxito. Como resume la Tabla 3.1, los planificadores basados en aprendizaje automática destacan en múltiples facetas operativas: una vez entrenados, generan trayectorias en tiempo prácticamente instantáneo (latencia de milisegundos determinista, solo una inferencia de red), pueden reaccionar a cambios en el entorno de forma inmediata al observarlos en cada paso o incluso replanificar completamente en tiempo real y tienden a producir movimientos eficientes si se les ha recompensado por ello durante la fase de entrenamiento (Tamizi et al., 2023).

Esto soluciona en principio varias limitaciones clásicas. Primero, la adaptabilidad: los enfoques de DRL han demostrado mayor robustez en entornos complejos y dinámicos, aprendiendo políticas que evitan obstáculos móviles y reaccionan a imprevistos en tiempo real. Segundo, un planificador basado en red neuronal no requiere un modelado geométrico exhaustivo: aprende implícitamente a navegar el espacio de configuración a partir de ejemplos o de interacción, reduciendo la dependencia de conocimiento experto previo sobre la estructura del problema. Esto se aprecia en resultados donde agentes de DRL descubren por sí solos estrategias de movimiento eficientes en espacios de configuración complejos. Tercero, las técnicas de aprendizaje permiten optimizar

directamente objetivos complejos de forma multi-criterio: por ejemplo, recompensando al agente por minimizar el tiempo de movimiento, la energía consumida y el riesgo de colisión simultáneamente, algo difícil de lograr con un planificador tradicional sin una cuidadosa sintonización de pesos en una función de coste. En resumen, los métodos basados en DRL ofrecen un comportamiento de planificación-reactiva unificado que potencialmente supera la necesidad de *pipelines* separados y ajustes manuales, alcanzando un nivel de autonomía acorde a las exigencias de la Industria 4.0 (Tamizi et al., 2023).

Pese a esto, las técnicas de DRL aún enfrentan importantes retos antes de su adopción industrial generalizada. En primer lugar, el coste de entrenamiento es elevado. A diferencia de un algoritmo de muestreo que no necesita entrenamiento previo, entrenar una política de DRL puede requerir millones de interacciones simuladas. Esto se traduce en horas o días de cómputo intensivo, para obtener un agente competente. Aunque este coste es offline (previo a la implantación) y se amortiza durante la fase operativa, supone una barrera práctica, más aún si el entorno o la tarea cambian y obligan a reentrenar. En segundo lugar, la ingeniería de recompensa y la seguridad durante el aprendizaje son desafíos no triviales. Un agente de DRL necesita una función de recompensa cuidadosamente diseñada para inducir el comportamiento deseado; de lo contrario, puede converger a políticas subóptimas o incluso peligrosas. Determinar la función de recompensa adecuada para resolver la tarea específica es complejo y suele requerir conocimiento experto. Incluso con una recompensa adecuada, durante la exploración inicial el agente puede incurrir en colisiones o acciones fuera de límites, por lo que entrenar con seguridad a un robot físico es prácticamente inviable; se depende casi siempre de una simulación. Esto conecta con un tercer obstáculo: la brecha *sim-to-real*. Las políticas aprendidas en simuladores suelen degradar su desempeño al transferirlas al mundo real debido a pequeñas discrepancias (modelos de fricción, ruidos no modelados, etc.). Estrategias como la aleatorización de dominio buscan mitigar este problema introduciendo variaciones en la simulación para forzar al agente a generalizar, pero garantizar una transferencia fiable sigue siendo un frente abierto de investigación.

Por último, existe el problema de la generalización: un agente entrenado para una tarea específica puede no adaptarse bien a variaciones de la misma (por

ejemplo, un cambio en la geometría del entorno o en los objetivos de la tarea). Esto limita su reutilización y requiere entrenamiento adicional para cada nueva variante de la tarea, lo que contrasta con los planificadores clásicos que pueden adaptarse más fácilmente a nuevas condiciones sin reentrenamiento.

En conclusión, el estado del arte en planificación de trayectorias para manipuladores de 6 GDL revela un amplio abanico de enfoques, cada uno con compromisos particulares. Los métodos geométricos probabilísticos aportan rapidez y garantías de factibilidad, pero requieren ajustes posteriores para alcanzar la calidad deseada. Los optimizadores locales brindan fineza y consideración dinámica, a expensas de depender de trayectorias iniciales y correr el riesgo de mínimos locales. Las combinaciones híbridas logran resultados sobresalientes integrando ambas fortalezas, aunque con mayor complejidad de implementación y tiempos de ejecución. Finalmente, las técnicas basadas en aprendizaje emergen para superar limitaciones persistentes, principalmente dotando al robot de capacidad de aprender de la experiencia y de adaptarse en entornos cambiantes, pero aún no reemplazan a los algoritmos tradicionales debido a sus elevados requisitos de datos, entrenamiento y garantías. El panorama actual deja patente que las técnicas de aprendizaje por refuerzo, convenientemente integradas, tienen el potencial de resolver muchas de las carencias históricas de la planificación de trayectorias, constituyendo así la frontera de desarrollo hacia robots industriales más inteligentes, autónomos y eficientemente operativos. Las limitaciones presentes en DRL son objeto de intensa investigación, y su superación marcará un hito en la siguiente generación de planificadores de movimiento.

Tabla 3.1: Comparativa de propiedades operativas y de implementación de distintas categorías de planificadores de trayectorias para brazos robóticos.

Tipo de planificación	Resultado óptimo	Probabilísticamente completo	Tiempo de planificación	Robustez	Adaptabilidad online	Post-procesado	Sensibilidad a cond. inic.	Integración en sistemas reales	Escalabilidad	Preparación previa	Conoc. experto previo	Reutilización	Naturaleza
Planificadores geométricos mono-consulta basados en muestreo	2	5	4	4	1	2	2	5	4	1	2	5	Estocástica
Planificadores geométricos multi-consulta basados en muestreo	2	5	4	0	2	2	2	2	3	1	4	1	Estocástica
Planificadores geométricos con objetivos de optimización global basados en muestreo	5	5	0	4	1	2	2	4	4	1	4	0	Estocástica
Planificadores basados en optimización local	4	1	4	3	1	5	1	2	2	1	5	4	Determinista
Esquemas de planificación híbrida (muestreo + optimización local)	5	5	4	5	1	5	2	2	3	1	5	0	Estocástica
Planificadores basados en aprendizaje supervisado	4	4	5	5	5	0	4	4	5	5	5	0	Determinista
Planificadores basados en aprendizaje por refuerzo	4	4	5	5	5	0	4	4	5	0	5	0	Determinista

Leyenda de símbolos: 5: Excelente 4: Bueno 3: Neutro 2: Deficiente 1: Muy deficiente
0: Dependiente del entorno o implementación

Los datos son el nuevo petróleo.

Clive Humby

CAPÍTULO

4

Diseño del proceso de planificación

Contenido del Capítulo

EN ESTE CAPÍTULO se describe en detalle el diseño técnico del proceso de planificación propuesto, el cual constituye el núcleo metodológico de la tesis. A partir de los objetivos y de la metodología definidos en el Capítulo 1, se desarrolla una arquitectura modular que integra herramientas de simulación, algoritmos de planificación clásicos y técnicas de aprendizaje por refuerzo profundo.

El diseño ha sido cuidadosamente estructurado para dar respuesta a las limitaciones identificadas en los enfoques tradicionales, especialmente en lo que respecta al carácter no determinista y la falta de reproducibilidad. Frente a ello, el enfoque adoptado se apoya en la construcción de un entorno de simulación cinemáticamente fiel, la generación de un conjunto de datos representativo del espacio de trabajo, y el entrenamiento de una política de planificación capaz de generalizar a nuevas tareas.

Un aspecto fundamental del diseño es su carácter genérico y extensible. Todas las fases del proceso, desde el muestreo del espacio de trabajo hasta la validación de la política entrenada, han sido concebidas de manera independiente del robot específico utilizado. Aunque el estudio se centra en el manipulador UR3e, el procedimiento propuesto puede adaptarse a otros brazos robóticos con mínima reconfiguración, lo que amplía su aplicabilidad en distintos entornos industriales o académicos.

El capítulo se estructura en varias secciones, cada una de las cuales corresponde a una de las fases clave del proceso:

- La selección del entorno de simulación más adecuado para un entrenamiento rápido, reproducible y representativo.
- El modelado detallado del manipulador y su espacio de trabajo en dicho entorno.
- La generación masiva de poses objetivo y el filtrado de configuraciones inválidas mediante cinemática inversa y detección de colisiones.
- La planificación sistemática de trayectorias con algoritmos clásicos, dando lugar a un conjunto de datos diverso y equilibrado.

- El entrenamiento de una política de planificación mediante aprendizaje por refuerzo profundo, empleando técnicas de curriculum learning y experiencia experta.
- La validación cruzada de los resultados, tanto en simulación como sobre el robot real.

Cada decisión de diseño se justifica en base a los requisitos operativos del entorno robótico, las propiedades deseables de la política objetivo y las posibilidades que ofrece el estado del arte en planificación y aprendizaje automático. Las secciones siguientes profundizan en los detalles técnicos de cada fase, abordando tanto los desafíos encontrados como las soluciones adoptadas para garantizar la validez, generalización y aplicabilidad del sistema propuesto.

4.1 Criterios de diseño generales

El proceso de planificación propuesto en esta tesis parte de la hipótesis planteada en el apartado 1.2. Para que este enfoque sea viable, el proceso se ha diseñado sustentándose en la metodología planteada en el apartado 1.3, que establece un ciclo iterativo de exploración, desarrollo y evaluación. Este ciclo permite ajustar el diseño en función de los resultados obtenidos, asegurando que cada iteración aporte valor y refinamiento al sistema.

En este contexto, los criterios generales de diseño se han definido a partir de tres factores clave:

1. Las limitaciones de los algoritmos de planificación clásicos.
2. Los requisitos operativos del entorno robótico industrial.
3. Las propiedades deseadas de la política objetivo.

En lo que respecta al primer punto, la revisión de la literatura ha puesto de manifiesto diversas carencias en los enfoques tradicionales (3.1). En particular, la mayoría de planificadores carecen de una de las características más críticas para su aplicación en entornos industriales: La reproducidibilidad y el determinismo. Muchos de los algoritmos existentes dependen de la aleatoriedad o de

heurísticas que no garantizan resultados consistentes, lo que dificulta su implementación en escenarios donde la precisión y la fiabilidad son esenciales.

Desde el punto de vista operativo, el diseño debe permitir generar un entorno de entrenamiento robusto, repetible y suficientemente representativo de los desafíos geométricos reales que afronta un manipulador. Para ello, es imprescindible asegurar que el entorno sea determinista (con el fin de reducir la varianza del proceso de aprendizaje), eficiente en términos de computación (para posibilitar múltiples episodios de entrenamiento por segundo) y fiel en la representación cinemática del sistema, obviando la complejidad dinámica innecesaria cuando el objetivo es puramente geométrico.

Por último, y en línea con la hipótesis planteada, se considera esencial que el proceso produzca una política que no esté especializada en una única tarea concreta, sino que sea capaz de generalizar a múltiples configuraciones del entorno y a metas arbitrarias dentro del espacio de trabajo. Esta política debe ser, además, interpretable como un módulo funcional reutilizable, de forma que pueda ser integrado como bloque básico dentro de estructuras de planificación más complejas o adaptado a nuevas tareas sin necesidad de rediseño completo.

El proceso propuesto se compone de las siguientes fases principales:

1. La selección de un simulador adecuado para modelar con fidelidad la cinemática de un manipulador industrial.

2. El desarrollo de un entorno de simulación controlado y flexible que sirva como modelo cinemático del manipulador.

3. La construcción de un conjunto de datos de trayectorias lo suficientemente amplio y representativo, que permita por una parte entrenar la política y por otra evaluar su rendimiento frente a métodos clásicos.

4. El entrenamiento de una política de DRL que sea capaz de generalizar a metas arbitrarias dentro del espacio de trabajo del manipulador.

5. La validación exhaustiva de la política en simulación y sobre el robot real y la comparación con planificadores clásicos para evaluar su rendimiento, generalización y aplicabilidad.

Cada una de estas fases responde a decisiones de diseño concretas que buscan garantizar tanto la viabilidad del enfoque como su valor añadido frente a alternativas existentes. En primer lugar, se prioriza el uso de un entorno simulado determinista y cinemáticamente fiel, ya que esto permite realizar miles de episodios de entrenamiento reproducibles sin la complejidad asociada a la simulación dinámica.

En segundo lugar, se opta por un modelado detallado del espacio de trabajo mediante el muestreo denso de poses y la planificación sistemática con algoritmos clásicos. Este procedimiento permite construir un conjunto de datos de referencia que no solo se utiliza para alimentar o guiar el proceso de entrenamiento (vía experiencia experta), sino también para realizar comparaciones justas y exhaustivas con los métodos existentes en la literatura.

En tercer lugar, el entorno de entrenamiento se formula cuidadosamente como un proceso de decisión de Markov, considerando múltiples representaciones del espacio de observación y acción, y evaluando distintas funciones de recompensa. Esta fase se ejecuta de forma iterativa, con ciclos de evaluación y rediseño que permiten refinar tanto el comportamiento de la política como su capacidad de generalización.

Por último, la validación del sistema se extiende más allá del entorno simulado, incluyendo pruebas reales sobre hardware físico. Este paso resulta esencial para comprobar la viabilidad de la transferencia sim-to-real, así como para evaluar la robustez, estabilidad y reproducibilidad del comportamiento aprendido.

En conjunto, el proceso diseñado responde a un enfoque metodológico estructurado que persigue no solo alcanzar una solución funcional, sino también generar conocimiento aplicable sobre cómo entrenar políticas generalizables en planificación geométrica. En las siguientes secciones se detallan cada una de las fases que componen este proceso, justificando las decisiones adoptadas y los mecanismos empleados para garantizar la validez y aplicabilidad de los resultados obtenidos.

Estos criterios han guiado todas las decisiones posteriores de diseño, desde la selección del simulador hasta la formulación del entorno de entrenamiento, la generación del conjunto de datos y la validación del sistema. En las siguientes

secciones se describen con detalle las decisiones técnicas adoptadas en cada fase, siempre alineadas con este conjunto de principios orientadores.

4.2 Selección del entorno de simulación

El proceso de entrenamiento propuesto en esta tesis requiere un entorno de simulación que satisfaga una serie de requisitos clave: alta fidelidad cinemática, reproducibilidad, ejecución determinista, capacidad de simulación a alta velocidad y compatibilidad con algoritmos de aprendizaje por refuerzo profundo. Estos requisitos responden directamente a los criterios metodológicos definidos previamente, y son fundamentales para garantizar la viabilidad del proceso iterativo de entrenamiento, validación y comparación.

Con base en estos criterios, se identifican cuatro simuladores ampliamente utilizados en el ámbito de la robótica y el aprendizaje por refuerzo que podrían ser candidatos viables: MuJoCo, Gazebo, Unity y CoppeliaSim. La elección de estos entornos no es arbitraria: todos han sido empleados de forma frecuente en la literatura reciente, ya sea en tareas de simulación física precisa, desarrollo de entornos personalizables, o entrenamiento de agentes de aprendizaje profundo (Kaup et al., 2024; Collins et al., 2021; Körber et al., 2021). Cada uno presenta ventajas y limitaciones que deben analizarse de forma comparativa antes de adoptar una decisión. La tabla 4.1 resume las características más relevantes de cada simulador en relación con los criterios definidos. Esta comparativa permite evaluar de forma objetiva cuál de ellos se adapta mejor a las necesidades del proceso de planificación propuesto.

Tabla 4.1: Comparativa de simuladores para planificación con aprendizaje por refuerzo

Criterio	PyBullet	MuJoCo	Gazebo	Unity	CoppeliaSim
Fidelidad cinemática	Alta (modo cinemático preciso)	Alta (basado en dinámica)	Alta (dependiente del modelo físico)	Media (centrado en gráficos)	Media
Velocidad de simulación	Muy alta (sin dinámica)	Alta (especialmente en GPU)	Baja	Media	Baja
Determinismo / reproducibilidad	Alta	Alta	Media-Baja	Media-Baja	Media
Compatibilidad con herramientas	Directa (URDF, Gym)	Limitada (requiere conversión)	Directa (ROS)	Limitada (requiere integración manual)	Alta (interfaz gráfica y scriptable)
Facilidad de desarrollo	Alta (API Python simple)	Media (requiere MJCF y configuración)	Media-Baja (complejo de integrar)	Alta (editor visual, pero no orientado a robótica)	Alta (GUI interactiva)
Uso frecuente en DRL con robots	Alta	Muy alta	Baja	Media	Media

A continuación se resumen las principales características de estos simuladores en relación con los criterios definidos:

- **Fidelidad cinemática y precisión posicional**: MuJoCo y Gazebo destacan por su fidelidad física, pero su precisión posicional depende del modelado dinámico, lo que puede introducir inestabilidad si no se configura adecuadamente. Unity y CoppeliaSim ofrecen buena visualización, pero menos control sobre la cinemática exacta de manipuladores. PyBullet, por su parte, permite operar en modo cinemático sin resolver dinámica, garantizando precisión posicional estable (Coumans and Bai, 2016; Lobbezoo and Kwon, 2023).

- **Velocidad de simulación**: MuJoCo presenta una ejecución muy eficiente, especialmente en GPU, pero su formato propio (MuJoCo XML Configuration Format (MJCF)) dificulta la portabilidad. Gazebo tiene tiempos de

simulación elevados debido al procesamiento físico y gráfico. Unity y CoppeliaSim penalizan el rendimiento al priorizar entornos visuales. PyBullet permite miles de pasos por segundo en modo sin dinámica, lo cual lo convierte en una opción destacada para entrenamiento masivo (Kaup et al., 2024).

- **Determinismo y reproducibilidad**: PyBullet garantiza entornos deterministas cuando se configuran correctamente las semillas y las condiciones iniciales. MuJoCo también ofrece buena reproducibilidad. Gazebo y Unity pueden introducir variabilidad aleatoria debido a componentes físicos y motores de renderizado (Collins et al., 2021).

- **Compatibilidad y extensibilidad**: Gazebo está profundamente integrado con el ecosistema ROS, pero su extensión requiere conocimientos avanzados. Unity requiere puentes externos para integrarse con robots reales. MuJoCo utiliza un formato propietario. PyBullet admite directamente modelos URDF y permite integrar fácilmente entornos compatibles con Gym, lo que simplifica su uso en aprendizaje por refuerzo (Panerati et al., 2021).

- **Curva de aprendizaje y facilidad de desarrollo**: CoppeliaSim y Gazebo ofrecen interfaces gráficas, pero requieren configuración detallada. PyBullet, en cambio, proporciona una Application Programming Interface (API) simple y directa en Python, muy útil para diseñar entornos personalizados con poco código y escasa sobrecarga de configuración.

A partir de este análisis, se opta por utilizar PyBullet como entorno de simulación principal para el desarrollo de esta tesis. Esta elección se justifica por su capacidad para ejecutar simulaciones rápidas y deterministas, su fidelidad en la representación cinemática de manipuladores articulados, su compatibilidad con el formato URDF y su integración directa con librerías de aprendizaje por refuerzo. Además, la posibilidad de operar en modo cinemático sin resolver fuerzas físicas reduce la carga computacional, permitiendo entrenar políticas con un número elevado de episodios por segundo sin sacrificar precisión posicional (Lobbezoo and Kwon, 2023).

PyBullet ha sido ampliamente empleado en trabajos recientes que combinan planificación con aprendizaje profundo, incluyendo entornos benchmark y simulaciones específicas de manipuladores industriales (Panerati et al., 2021; Kaup et al., 2024). Su soporte activo y la disponibilidad de ejemplos y documentación consolidan aún más su elección como herramienta robusta y fiable para el entrenamiento y la evaluación de políticas en el contexto de esta tesis.

4.3 Modelado del manipulador y entorno de simulación

Una vez definido el simulador base, se procede al diseño y modelado del entorno virtual en el que tendrá lugar el entrenamiento de la política. Este entorno debe representar de forma fidedigna la cinemática del manipulador, así como su espacio de trabajo, limitaciones articulares y posibles restricciones estructurales.

Como base del entorno se selecciona el manipulador UR3e con una pinza de agarre OnRobot RG2, perteneciente a la familia de robots colaborativos de Universal Robots. Esta elección responde a múltiples criterios estratégicos. En primer lugar, los robots de la serie Universal Robots (UR) son ampliamente utilizados en entornos industriales reales, en aplicaciones de ensamblado, manipulación ligera y automatización de procesos. Esto convierte su estudio en un objetivo relevante tanto desde el punto de vista académico como práctico.

En segundo lugar, toda la gama UR comparte una estructura cinemática común de seis grados de libertad, basada en una cadena de articulaciones rotacionales con el mismo orden y configuración de ejes. Esta uniformidad estructural convierte al UR3e en un candidato ideal para desarrollar políticas de planificación que puedan generalizarse, con mínimos ajustes, a otros modelos de la misma serie (como el UR5e o el UR10e), facilitando así el análisis de modularidad y escalabilidad propuesto en esta tesis.

Además, los robots UR cuentan con documentación técnica detallada, interfaces de bajo nivel accesibles (a través de URScript o sus drivers de ROS), y simuladores oficiales que permiten validar ciertos aspectos del comportamiento del sistema. Esta accesibilidad ha favorecido su adopción en numerosos trabajos

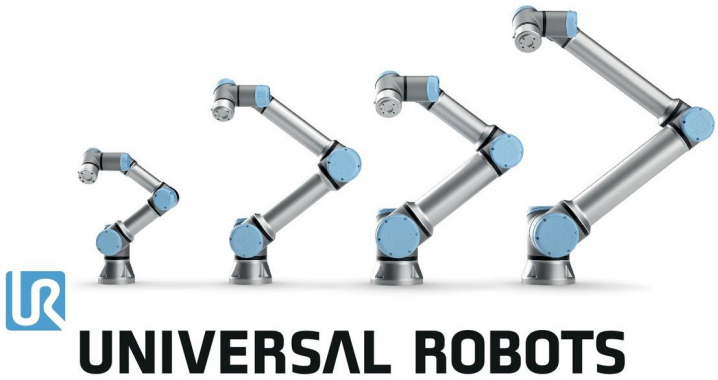

Figura 4.1: Gama de manipuladores colaborativos de Universal Robots. De izquierda a derecha: UR3e, UR5e, UR10e y UR16e. Como se observa, todos comparten una estructura cinemática común de seis grados de libertad, cambiando únicamente las dimensiones y capacidades de carga. (Universal Robots, 2025)

previos sobre aprendizaje por refuerzo profundo, donde se busca una transición eficiente del entorno simulado al entorno real (Kumar et al., 2016).

Para asegurar una representación precisa en simulación, se parte de un modelo URDF calibrado directamente sobre el robot real. En lugar de utilizar una versión genérica, se extrae el modelo mediante herramientas de calibración intrínseca proporcionadas por UR. Esta decisión permite reducir el desfase entre el modelo simulado y el comportamiento real del robot, mejorando la correspondencia cinemática y reduciendo el *sim-to-real gap* que habitualmente afecta a las políticas aprendidas en entornos puramente virtuales (James et al., 2019). La figura 4.2 muestra el modelo del manipulador UR3e con la pinza OnRobot RG2, representado en el entorno de simulación.

El entorno en sí representa un espacio de trabajo vacío, limitado por las restricciones propias del manipulador: límites articulares, posibles autocolisiones y colisión con el suelo. Esta simplificación intencionada permite aislar el problema de planificación geométrica sin interferencias externas, enfocando el entrenamiento en la resolución de tareas de alcance y orientación en todo el espacio operativo.

El modelo del entorno está construido sobre PyBullet, empleando los parámetros extraídos del URDF calibrado y una escena estática con plano de suelo. El entorno se codifica como una clase compatible con el estándar *gymnasium*, lo

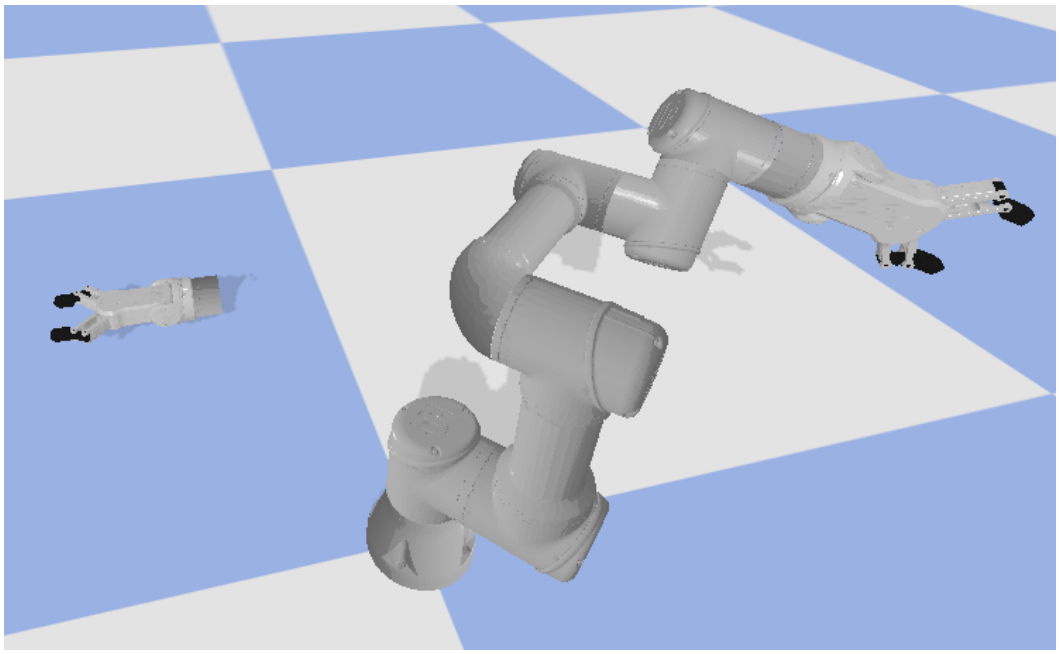

Figura 4.2: Entorno de simulación del manipulador UR3e con pinza OnRobot RG2. El entorno está diseñado para representar el espacio de trabajo del manipulador, con un plano de suelo y sin obstáculos adicionales. Adicionalmente, se incluye un eslabón final con otra pinza para representar la meta a alcanzar.

que permite su integración directa con algoritmos de DRL. La estructura modular del entorno facilita su extensión posterior a tareas más complejas, incorporación de restricciones adicionales o integración en planificadores jerárquicos.

4.4 Muestreo de poses y filtrado del espacio de configuración

El proceso de generación y filtrado de poses descrito en esta sección constituye una etapa fundamental en la construcción del conjunto de datos utilizado para el entrenamiento y evaluación de los planificadores. Este procedimiento ha sido diseñado de manera genérica y modular, de modo que puede aplicarse a cualquier tipo de manipulador robótico, independientemente de su configuración articular o especificaciones geométricas. A partir de una caracterización precisa del espacio de trabajo, se generan de forma masiva poses objetivo dis-

tribuidas uniformemente. Posteriormente, se evalúa su viabilidad mediante un análisis exhaustivo de las configuraciones articulares que permiten alcanzarlas, descartando aquellas que implican colisiones o están fuera del alcance físico del robot. Esta metodología garantiza que las trayectorias generadas posteriormente se basen exclusivamente en situaciones factibles, maximizando así la utilidad del conjunto de datos y asegurando su aplicabilidad en entornos reales o simulados. Además, su carácter extensible permite adaptar el proceso a diferentes plataformas robóticas sin requerir modificaciones estructurales significativas.

4.4.1 **Definición y muestreo del espacio de trabajo**

El proceso de generación del conjunto de tareas a resolver comienza con la caracterización del espacio de trabajo alcanzable del manipulador, en este caso para el UR3e.

Con el objetivo de cubrir de forma uniforme y representativa el espacio de trabajo del robot (apartado 2.1.4), se emplea una estrategia de muestreo esférico, que permite distribuir homogéneamente los puntos dentro de una envolvente hemisférica delimitada por restricciones físicas del robot.

Para generar una distribución balanceada, se realiza el muestreo en coordenadas esféricas:

$$r = U(r_{\min}, r_{\max}) \tag{4.1}$$

$$\theta = U(0, 2\pi) \tag{4.2}$$

$$\phi = \arccos(1 - U(0,1)) \tag{4.3}$$

donde $U(a,b)$ denota una distribución uniforme en el intervalo $[a,b]$. Estas muestras se convierten a coordenadas cartesianas mediante:

$$x = r\cos\theta\sin\phi \tag{4.4}$$

$$y = r\sin\theta\sin\phi \tag{4.5}$$

$$z = r\cos\phi \tag{4.6}$$

Los límites de muestreo se fijan para asegurar validez cinemática y evitar regiones no alcanzables o peligrosas: $r \in [0.1, 0.85]$ m, con muestreo completo en ángulo azimutal $\theta \in [0, 2\pi]$, y elevación hemisférica positiva. Para cada posición, se asignan orientaciones aleatorias válidas dentro del rango articular del UR3e, generando poses completas.

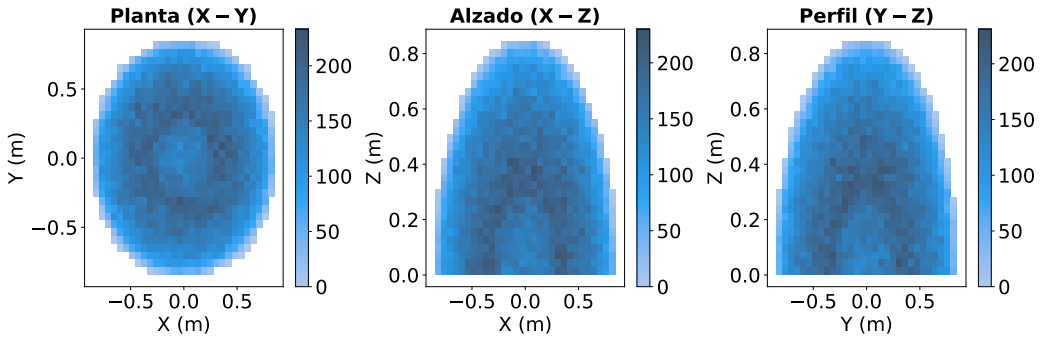

Figura 4.3: Distribución espacial de las poses objetivo generadas tras el muestreo. Se observa una cobertura uniforme del espacio de trabajo del UR3e, con exclusión de la región central próxima a la base.

Este procedimiento se repite hasta obtener una distribución de más de 130.000 poses candidatas. El número final de muestras se ajusta iterativamente para lograr una cobertura uniforme del espacio alcanzable, evitando tanto la concentración en regiones específicas como la aparición de huecos significativos. Esta densidad uniforme es esencial para asegurar que la política aprendida sea capaz de generalizar a lo largo de todo el espacio de trabajo del robot. El proceso de muestreo se ilustra en la figura 4.3, donde se observa una distribución homogénea de las poses generadas.

4.4.2 Filtrado de configuraciones no válidas

Una vez generadas las poses objetivo, es necesario garantizar que estas sean alcanzables desde un punto de vista cinemático y estén libres de restricciones físicas. Para ello, se utiliza el solucionador cinemático inverso IKFast (Diankov, 2010), capaz de calcular analíticamente todas las configuraciones articulares que permiten alcanzar una determinada pose del efector.

Para cada muestra (x, y, z, R), se obtienen todas las soluciones posibles:

$$\mathcal{Q}(x, y, z, R) = \{q_i \in \mathcal{C} \mid f(q_i) = (x, y, z, R)\} \tag{4.7}$$

La distribución en el espacio de configuración resultante se muestra en la figura 4.4, donde se observa una amplia variedad de configuraciones articulares asociadas a las poses muestreadas. Esta diversidad es crucial para el entrenamiento de políticas robustas, ya que permite explorar diferentes trayectorias y maniobras. Sin embargo, no todas las configuraciones generadas son viables. Por ello, es necesario aplicar filtros adicionales para descartar aquellas configuraciones que:

- Provocan autocolisiones internas del robot.
- Intersectan con el suelo u otros elementos del entorno.
- No son alcanzables debido a límites articulares o singularidades.

Estas condiciones se evalúan mediante herramientas de detección de colisiones implementadas en el motor de simulación y el entorno de planificación. Solo aquellas poses que cuentan con al menos una configuración articular válida, libre de colisiones y físicamente alcanzable, son consideradas para la siguiente fase. Para realizar este filtrado, se implementa el algoritmo 3, que itera sobre todas las poses muestreadas y aplica los criterios de viabilidad.

Después de aplicar este filtrado, se obtiene un conjunto de configuraciones válidas que cumplen con los criterios de viabilidad y colisión. La figura 4.5 muestra la distribución final del espacio de configuración tras aplicar los filtros. Se puede observar que las regiones inalcanzables y las zonas de colisión se han eliminado, dejando un conjunto de configuraciones válidas y diversas. Estas restricciones provocan una ligera reducción en el número de poses viables, pero garantizan que las configuraciones resultantes sean físicamente alcanzables y seguras para el robot. No obstante, la diversidad de configuraciones se mantiene alta y uniforme ya que cada pose muestreada puede tener múltiples configuraciones válidas asociadas, lo que permite una amplia exploración del espacio de configuración.

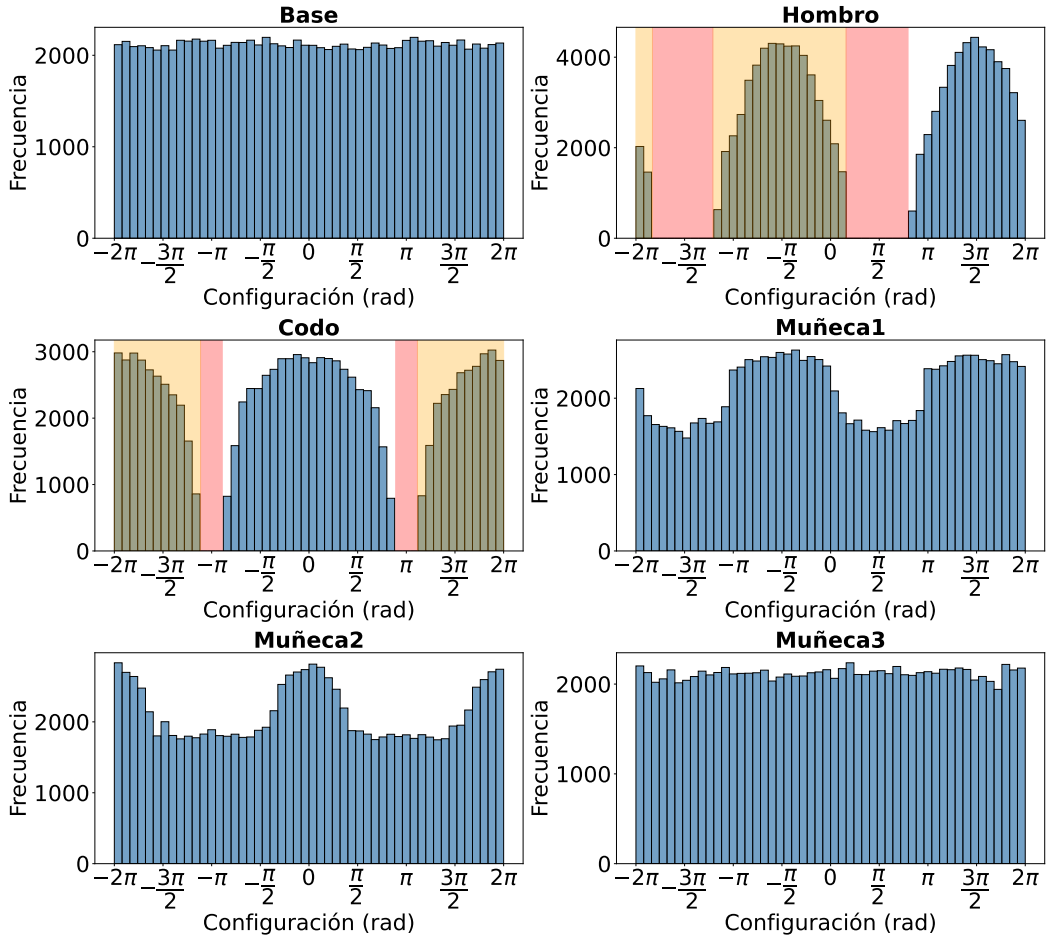

Figura 4.4: Distribución del espacio de configuración tras el muestreo inicial. En este muestreo inicial se observan regiones inalcanzables (naranja) y zonas donde se producen colisiones (rojo).

Como resultado, se obtiene un conjunto final de más de 100,000 poses viables, que forman la base para la generación de trayectorias. Este conjunto garantiza una amplia variedad de situaciones de planificación, abarcando desde trayectorias cortas hasta movimientos de gran alcance, y proporciona una base sólida para comparar el rendimiento de diferentes enfoques de planificación.

Algoritmo 3: Filtrado de poses válidas

Entrada: Modelo cinemático del robot, límites del espacio de trabajo $\mathcal{W}$
Salida : Conjunto de posiciones válidas y configuraciones asociadas $C = \{(x, y, z, R, q)\}$

```
Muestrear uniformemente las poses dentro del espacio de trabajo del robot;
Inicializar conjunto de configuraciones válidas: C ← ∅;
foreach pose muestreada (x, y, z, R) ∈ W do
    Calcular todas las soluciones de cinemática inversa Q(x, y, z, R) usando IKFast;
    foreach configuración articular q_i ∈ Q do
        if q_i está libre de colisiones y es alcanzable then
            Añadir (x, y, z, R, q_i) a C;
return C
```

4.5 Generación del conjunto de datos

Una vez obtenidas las configuraciones válidas del robot en el espacio de trabajo, se procede a planificar trayectorias utilizando el *pipeline* de MoveIt integrado con OMPL. Esta infraestructura permite convertir una petición de planificación a nivel cartesiano en una trayectoria completa en el espacio de configuración, respetando las restricciones cinemáticas y dinámicas del manipulador.

El proceso comienza con la conversión de los pares de poses inicial y final en configuraciones articulares válidas mediante IKFast, un solucionador analítico optimizado para eficiencia computacional (Diankov, 2010). Una vez definidas ambas configuraciones, el pipeline ejecuta una secuencia de adaptadores de preprocesado que validan los límites articulares, colisiones y ubicación dentro del espacio de trabajo. Posteriormente, OMPL ejecuta uno de sus planificadores probabilísticos para generar un camino libre de colisiones. Los planificadores considerados incluyen todas las implementaciones funcionales disponibles en el *pipeline* de planificación de OMPL integrado en MoveIt.

Una vez encontrada una trayectoria válida, se aplica un postprocesado que incluye la parametrización temporal mediante TOTG, asegurando que las velocidades y aceleraciones respeten las capacidades físicas del robot. Este enfoque

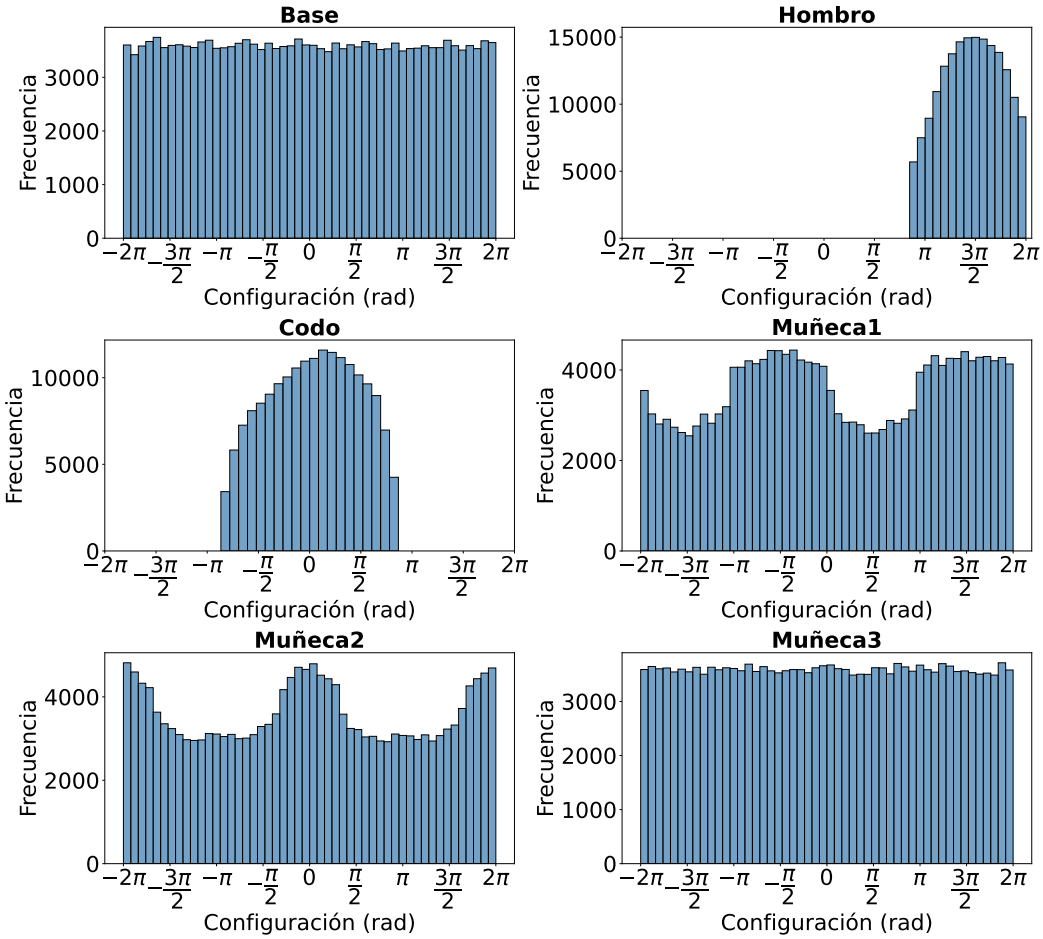

Figura 4.5: Distribución final del espacio de configuración tras aplicar los filtros de viabilidad y colisión.

permite obtener trayectorias realistas que pueden ejecutarse en un sistema real, además de ser útiles como banco de pruebas y demostraciones para acelerar el entrenamiento de políticas DRL.

Para garantizar una representación uniforme de la diversidad espacial, los pares de poses se seleccionan de acuerdo con su distancia euclídea en el espacio cartesiano, agrupándolos de manera uniforme. Esta estrategia asegura la inclusión equilibrada de trayectorias cortas, medias y largas, esto permite integrar un conjunto de datos diverso y representativo del espacio de trabajo completo del manipulador. La figura 4.6 muestra la distribución de distancias entre los pares de poses utilizadas en la generación del conjunto de datos, evidenciando una

representación uniforme de trayectorias cortas, medias y largas.

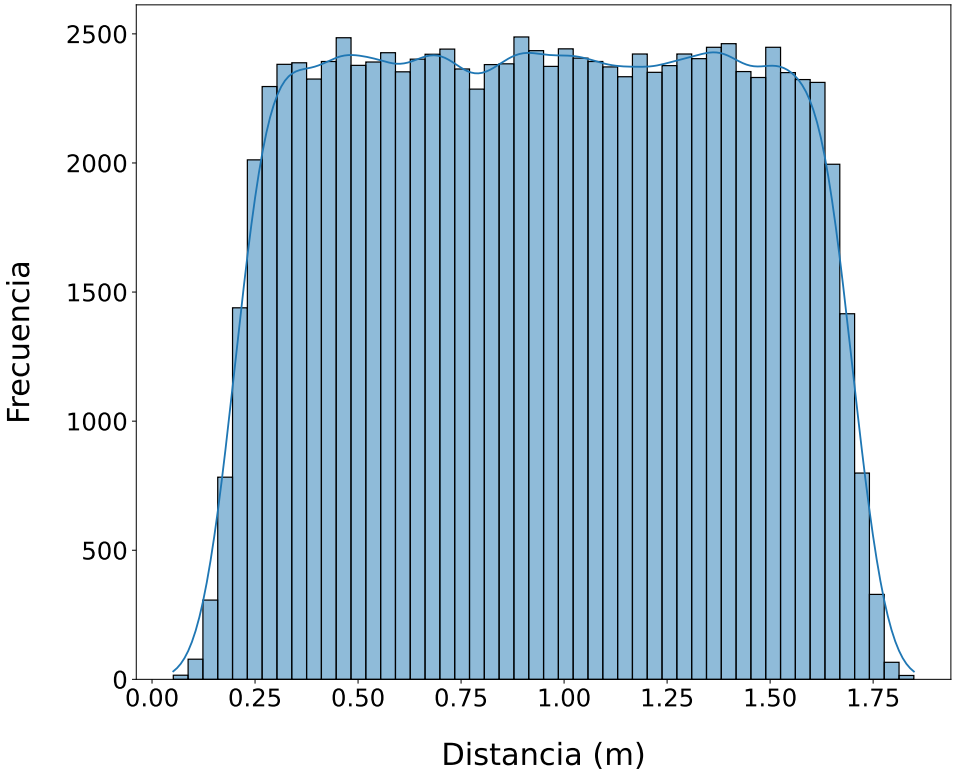

Figura 4.6: Distribución de distancias entre pares de poses utilizadas en la generación del dataset. Se observa una representación uniforme de trayectorias cortas, medias y largas.

Para cada par de poses:

- Se selecciona una configuración inicial válida.
- Se calcula una configuración final válida mediante IKFast.
- Se ejecuta el planificador, hasta un máximo de 10 intentos por par, con un límite de tiempo de 5 segundos por intento.
- Si la planificación es exitosa, se aplica TOTG con un factor de escalado del 10 % para velocidades y aceleraciones.

De cada trayectoria exitosa se almacenan los siguientes datos:

- Nombre del planificador utilizado.
- Poses inicial y final del efector final.
- Trayectoria geométrica en el espacio de configuración.
- Parametrización temporal de la trayectoria.
- Tiempo de planificación requerido.
- Longitud de la trayectoria en el espacio de configuración.

Este proceso se repite para las más de 100.000 pares de poses filtradas, resultando en un conjunto de datos diverso, equilibrado y representativo del espacio de trabajo completo del UR3e. Esta base de datos constituye tanto un *benchmark* para planificadores clásicos como un conjunto de demostraciones expertas para la fase de entrenamiento del agente de aprendizaje por refuerzo.

Todo el proceso de generación del conjunto de datos se implementa como un algoritmo modular, que permite su reutilización y adaptación a diferentes manipuladores o configuraciones. El algoritmo 4 describe los pasos detallados para construir el conjunto de trayectorias a partir de las poses válidas generadas en la fase anterior.

Algoritmo 4: Construcción del conjunto de trayectorias

Entrada: Conjunto de poses válidas y configuraciones asociadas C
Salida : Conjunto final de trayectorias D

```
1  Inicializar conjunto de trayectorias: D ← ∅;
2  Definir intervalos de distancia para muestreo de pares de poses;
3  foreach intervalo de distancia d do
4      Muestrear uniformemente pares de poses (p_inicio, p_meta) ∈ C a una
         distancia d;
5      foreach par (p_inicio, p_meta) do
6          Seleccionar aleatoriamente una configuración válida
             q_inicio ∈ Q(p_inicio);
7          Calcular configuración final q_meta resolviendo IK con
             restricciones usando IKFast para p_meta;
8          foreach planificador OMPL P en MoveIt do
9              for i = 1 hasta 10 do
10                 Intentar planificar la trayectoria τ = P(q_inicio, q_meta) con
                     un límite de 5 segundos;
11                 if τ es válida then
12                     Calcular parametrización temporal t = TOTG(τ);
13                     Calcular longitud de la trayectoria en el espacio de
                         configuración;
14                     Almacenar
                         (P, p_inicio, p_meta, τ, t, tiempo de planificación, longitud)
                         en D;
15 return D
```

El aprendizaje es experiencia, todo lo demás es información.

Albert Einstein

CAPÍTULO

5

Entrenamiento de nuestra solución basada en aprendizaje por refuerzo profundo

Contenido del Capítulo

ESTE CAPÍTULO constituye el núcleo experimental de la tesis, al describir el proceso completo de entrenamiento de una política determinista basada en DRL, diseñada para resolver la planificación geométrica de trayectorias en manipuladores industriales con metas definidas en *runtime*. Tras el marco conceptual presentado en los Capítulos 2 y 3, y el conjunto de datos y banco de pruebas introducidos en el Capítulo 4, aquí se aborda la formulación del problema como un MDP, la construcción de un entorno de entrenamiento reproducible y modular, la evaluación comparativa de algoritmos de DRL y la selección final de hiperparámetros, así como las técnicas de aceleración aplicadas para garantizar convergencia y estabilidad.

El objetivo principal es demostrar que es posible obtener una política capaz de reducir de forma consistente el error pose–objetivo en cualquier región del espacio de trabajo, explotando información articular y cartesiana para resolver la redundancia cinemática. Para ello se adopta un diseño del espacio de estados y de acciones que garantiza suficiencia de información sin introducir dependencias *ad hoc*, una función de recompensa híbrida combinada con un esquema progresivo de *curriculum learning*, y un entorno de simulación ligero pero fiel a las restricciones geométricas del robot. Asimismo, se analizan los algoritmos más representativos de control continuo (PPO, TD3, Deep Deterministic Policy Gradient (DDPG), SAC), optimizados mediante *Optuna*, de modo que la elección final del algoritmo y de sus parámetros esté respaldada empíricamente.

Para mitigar la elevada demanda de muestras propia del DRL, se integra inyección de experiencia experta a partir de trayectorias generadas con planificadores clásicos de OMPL, lo que permite acelerar el proceso de aprendizaje y mejorar la estabilidad de la política. Una vez entrenada, se presentan resultados que evidencian la capacidad de la política para resolver diferentes escenarios: trayectorias libres, trayectorias con restricciones intermedias y trayectorias hacia metas móviles. En todos los casos, las trayectorias geométricas producidas por la política se transforman en trayectorias ejecutables mediante TOPPRA, asegurando perfiles de velocidad y aceleración compatibles con los límites físicos del robot.

Finalmente, se aborda la transferencia *sim-to-real* a través de un módulo de integración Gymnasium–ROS, concebido en el marco del proyecto Europeo

ACROBA, que encapsula la política como un bloque funcional desplegable en robots reales. Este módulo permite ejecutar la política tanto en modos de tiempo real estricto como en ejecución por lotes, preservando determinismo, reproducibilidad y seguridad. De este modo, el capítulo no solo describe el entrenamiento en simulación, sino que completa la transición al robot físico, confirmando la aplicabilidad práctica de nuestra política basada en DRL.

La estructura es la siguiente: en la Sección 5.1 se detallan los componentes del MDP y las decisiones de diseño. La Sección 5.2 describe el entorno de entrenamiento y su integración con URDF y PyBullet. La Sección 5.3 presenta la evaluación comparativa de algoritmos y el ajuste de hiperparámetros. La Sección 5.4 analiza la inyección de experiencia experta y su impacto. En la Sección 5.5 se muestran, por una parte la influencia del *curriculum learning* y por otras los resultados de la política final y sus casos de uso. Por último, la Sección 5.6 recoge la transferencia al robot real mediante el módulo de integración Gymnasium–ROS.

5.1 Formulación del problema

Tal y como se introdujo en la Sección 2.2.3, la planificación geométrica consiste en encontrar una trayectoria continua que conecte dos configuraciones válidas del espacio de configuración del robot C_{free}, garantizando la ausencia de colisiones. Este planteamiento clásico puede extenderse y resolverse mediante técnicas de DRL, modelando el problema como un MDP.

En la Sección 2.3.2 se definieron los componentes del MDP para este problema. A continuación, se detallan las decisiones de diseño específicas para el caso de estudio. El objetivo es entrenar una política determinista capaz de resolver la planificación geométrica de trayectorias en un manipulador robótico de 6 GDL. Para ello, es necesario definir explícitamente:

- El espacio de estados, de forma que capture la información esencial para la planificación sin redundancias innecesarias.

- El espacio de acciones, que debe ser representativo, físicamente ejecutable y compatible con los algoritmos de DRL y las interfaces de control del manipulador.
- La función de recompensa, que guíe el aprendizaje evitando problemas de exploración ineficiente o convergencia prematura.
- La estructura de los episodios, que define el horizonte de decisión y las condiciones de terminación.

Las subsecciones siguientes detallan cada una de estas decisiones de diseño, justificando cómo contribuyen a que la política aprendida sea generalizable, modular y escalable, en línea con la hipótesis planteada en la Sección 1.2.

5.1.1 Espacio de estados

El estado observado por el agente combina información articular, cartesiana y de colisión en un vector híbrido de 14 dimensiones:

$$o = \left[q_1, \ldots, q_6,\ \Delta x, \Delta y, \Delta z, \Delta o_w, \Delta o_x, \Delta o_y, \Delta o_z, c\right] \in \mathbb{R}^{14} \tag{5.1}$$

Este diseño se justifica de la siguiente manera:

- Los valores articulares $q = [q_1, \ldots, q_6]$ permiten a la política explotar la estructura cinemática interna del robot. Incluir únicamente información cartesiana produciría ambigüedades debido a la redundancia en la cinemática inversa, lo que dificultaría la generalización.
- Las diferencias de posición $\Delta p = p_{\text{meta}} - p_{\text{actual}}$ y de orientación Δr proporcionan información directa del error con respecto a la meta. Esto acelera la convergencia y evita que el agente deba inferir la dirección del movimiento únicamente a partir de estados absolutos. El uso de la diferencia cartesiana respecto al objetivo favorece la generalización: el agente no memoriza trayectorias específicas, sino que aprende a reducir un error relativo en cualquier punto del espacio de trabajo.

- El indicador binario de colisión c resulta esencial en la exploración: sin esta señal, el agente podría aprender trayectorias que atraviesan regiones no válidas del espacio de configuración.

La diferencia de orientación se calcula como:

$$\Delta r = r_{\text{meta}} \otimes r_{\text{actual}}^{-1}, \quad r_{\text{actual}}^{-1} = [r_w, -r_x, -r_y, -r_z], \tag{5.2}$$

donde $\otimes$ denota el producto de cuaterniones.

En última instancia, la política debe ser capaz de representar o inferir el mapeo no unívoco entre poses y configuraciones articulares. Esta característica es especialmente relevante en manipuladores articulados, donde múltiples configuraciones pueden alcanzar una misma pose cartesiana. Por este motivo, resulta imprescindible que el vector de observaciones incluya información de configuración absoluta. Si se considerase únicamente la pose relativa al objetivo, el agente podría resolver tareas en regiones acotadas del espacio, pero carecería de la capacidad de generalizar a todo el dominio de trabajo y, además, se vería limitado para optimizar criterios asociados a la cinemática.

5.1.2 **Espacio de acciones**

El agente selecciona incrementos articulares continuos:

$$a = \Delta q = [\Delta q_1, \ldots, \Delta q_6] \in [-0.1, 0.1]^6 \tag{5.3}$$

Este diseño responde a tres criterios:

- El uso de incrementos articulares en lugar de posiciones absolutas evita problemas de discontinuidad en la acción y facilita la exploración local. Esto además garantiza que no se puedan producir discontinuidades en la trayectoria del robot.

- Frente a acciones en el espacio cartesiano, las acciones articulares garantizan que cualquier comando generado es ejecutable y respeta la topología real del espacio de configuración.

- El límite de ±0.1 radianes impone un control de grano fino sobre el movimiento, reduciendo el riesgo de oscilaciones bruscas y acelerando la convergencia hacia trayectorias suaves. Este intervalo permite avanzar mayores cantidades en regiones del espacio vacías mientras que en áreas más densamente pobladas de fuentes de colisión, puede restringir el incremento para evitar colisiones.

5.1.3 Dinámica de transición

El entorno evoluciona de manera determinista aplicando las acciones al modelo cinemático:

$$s_{t+1} = f(s_t, a_t) \tag{5.4}$$

La función f está implementada en PyBullet (Coumans and Bai, 2016), que permite simular de manera eficiente las transformaciones cinemáticas y la detección de colisiones. La dinámica completa del sistema (inercia, fricción, fuerzas) se omite deliberadamente, ya que el problema planteado se centra en la factibilidad geométrica. Esta simplificación reduce drásticamente el tiempo de simulación y, por tanto, el coste del entrenamiento.

5.1.4 Función de recompensa y *curriculum learning*

Durante el diseño de la recompensa se exploraron formulaciones *sparse*, en las que el agente únicamente recibe una señal positiva al alcanzar la meta, y formulaciones *dense*, que penalizan de forma proporcional el error cometido en cada paso. Ambas mostraron limitaciones: las recompensas escasas generaron dificultades de exploración y una asignación de crédito extremadamente pobre, mientras que las recompensas densas condujeron a convergencias prematuras hacia comportamientos subóptimos, como oscilaciones alrededor de la meta.

Como compromiso, se adoptó una recompensa híbrida. Además, con el objetivo de mejorar la exploración y el aprendizaje, se incorporaron elementos de *curriculum learning* en la función de recompensa. La recompensa está definida como:

$$r_t = -\alpha \cdot d_p(t) - \beta \cdot d_r(t) + \varphi_{\text{meta alcanzada}} \cdot r^{(i)}_{\text{éxito}}, \tag{5.5}$$

donde:

- $d_p(t) = \| p_t - p_{\text{goal}} \|_2$ es el error de posición,
- $d_r(t) = \arccos\left(2\langle r_t, r_{\text{goal}} \rangle^2 - 1\right)$ es el error de orientación,
- α, β son coeficientes de ponderación que permiten ajustar la importancia relativa de cada término,
- $\varphi_{\text{meta alcanzada}}$ es un indicador que se activa si $d_p(t) < \varepsilon_p^{(i)}$ y $d_r(t) < \varepsilon_r^{(i)}$,
- $r_{\text{éxito}}^{(i)}$ es la recompensa positiva de éxito, escalada por el nivel del *curriculum*.

La recompensa combina dos elementos principales:

- Una penalización densa proporcional al error en posición y orientación, que guía localmente al agente a reducir el número de pasos hacia la meta.
- Una señal discreta de éxito $r_{\text{éxito}}^{(i)}$, que refuerza los episodios completados con éxito dentro de las tolerancias establecidas por el *curriculum*.

El *curriculum learning* se integra directamente en esta formulación mediante la adaptación progresiva de las tolerancias de éxito $\varepsilon_p^{(i)}$ (tolerancia de posición) y $\varepsilon_r^{(i)}$ (tolerancia de orientación). Al inicio del entrenamiento se fijan valores laxos que permiten al agente lograr éxitos tempranos y acumular experiencias positivas. A medida que la política demuestra estabilidad en estas condiciones, las tolerancias se reducen de forma gradual hasta alcanzar valores finales de alta precisión.

Este enfoque introduce un proceso de aprendizaje progresivo: en las primeras fases, el agente aprende comportamientos poco exhaustivos, y en etapas posteriores desarrolla movimientos de mayor precisión tanto en traslación como en rotación. El refuerzo positivo asociado a cada nivel del *curriculum* se escala de manera que el agente mantiene un incentivo claro para progresar, evitando que se estanque en soluciones subóptimas de baja precisión. El *curriculum* avanza de manera progresiva con éxitos consecutivos del agente, para evitar que

un éxito temprano conduzca a una sobreoptimización prematura. La progresión se realiza de la siguiente manera:

$$\varepsilon_p^{(i+1)} = \text{máx}(\varepsilon_p^{(i)} \cdot \delta, \varepsilon_p^{objetivo}), \quad \varepsilon_r^{(i+1)} = \text{máx}(\varepsilon_r^{(i)} \cdot \delta, \varepsilon_r^{objetivo}) \tag{5.6}$$

Donde $\delta < 1$ es un factor de reducción que controla la velocidad de adaptación de las tolerancias y $r_{\text{éxito}}^{(i)}$ es la recompensa de éxito en el episodio i. La recompensa se escala en función del nivel de éxito alcanzado, de modo que:

$$r_{\text{éxito}}^{(i)} = \frac{r_0}{\delta^{\frac{i}{4}}} \tag{5.7}$$

En la práctica, el coeficiente de reducción δ se ajusta de manera empírica, lo que permite una progresión gradual en la dificultad de la tarea sin provocar estancamiento ni saltos abruptos en el aprendizaje. Permitiendo que las tolerancias para alcanzar la meta disminuyan de manera controlada y la recompensa aumente para premiar las metas cada vez más exigentes. Los valores finales establecidos para el *curriculum* son los siguientes:

- Tolerancias iniciales: $\varepsilon_p^{(0)} = 0.5$ m, $\varepsilon_r^{(0)} = 0.8$ rad
- Tolerancias finales: $\varepsilon_p^{(objetivo)} = 0.0005$ m, $\varepsilon_r^{(objetivo)} = 0.1$ rad
- Coeficiente de reducción: $\delta = 0.98$
- Número de niveles del *curriculum*: Para satisfacer las tolerancias objetivo marcadas se establecen 228 niveles de dificultad. En el caso de la orientación, la tolerancia objetivo marcada de 0.1 rad se alcanza en el nivel 103. Es decir, a partir del nivel 103, la tolerancia de orientación permanece constante en 0.1 rad, mientras que la tolerancia de posición continúa disminuyendo hasta alcanzar los 0.0005 m en el nivel 228.

El incremento de las tolerancias ε_p y ε_r ocurre de manera acoplada, es decir, se actualizan simultáneamente la tolerancia de posición y la tolerancia de orientación. Se han realizado experimentos para validar esta estrategia, mostrando que el acoplamiento en la adaptación de las tolerancias mejora la estabilidad

del entrenamiento y la convergencia hacia políticas más precisas. En los experimentos realizados de manera desacoplada, se observó que los agentes tendían a producir trayectorias geométricas con un mayor número de pasos, comúnmente intentando resolver primero la orientación y manteniéndola hasta alcanzar la posición deseada. Como resultado la trayectoria geométrica se volvía significativamente menos eficiente.

Por tanto, en el nivel i del *curriculum*, el criterio de éxito exige la satisfacción simultánea de las tolerancias de posición y orientación:

$$\mathscr{S}^{(i)} = \left\{ (d_p, d_r) \;\middle|\; d_p < \varepsilon_p^{(i)} \;\wedge\; d_r < \varepsilon_r^{(i)} \right\} \tag{5.8}$$

El avance del nivel i al i+1 se produce únicamente tras 100 episodios exitosos consecutivos bajo $\mathscr{S}^{(i)}$. El refuerzo discreto de éxito $r_{\text{éxito}}^{(i)}$ se concede exclusivamente cuando $(d_p, d_r) \in \mathscr{S}^{(i)}$; la satisfacción parcial (solo una métrica dentro de la tolerancia) no otorga recompensa y nunca desencadena el avance de nivel. Este acoplamiento impide que la política optimice un objetivo en detrimento del otro.

En resumen, la recompensa híbrida junto con el esquema de *curriculum learning* constituye un mecanismo dual: las señales densas guían la exploración local, mientras que la recompensa discreta condicionada al nivel de tolerancia garantiza que el aprendizaje avance de manera estructurada y progresiva hacia políticas de alta precisión y generalización en todo el espacio de trabajo.

5.1.5 Episodios y horizonte de decisión

El entrenamiento se organiza en episodios finitos, con una longitud máxima de 200 pasos. Esta longitud máxima se establece analizando las trayectorias disponibles en el conjunto de datos. En este, se observa que ninguna trayectoria supera los 50 pasos, lo que sugiere que la mayoría de las interacciones relevantes ocurren en un rango más corto. Se establece por tanto el límite de 200 pasos para permitir cierta flexibilidad y margen de exploración, sin que el agente pueda abusar de episodios excesivamente largos que podrían dificultar la convergencia. Cada episodio se inicializa seleccionando aleatoriamente una configuración inicial y una meta de entre el conjunto de poses válidas generado en

la sección 4.5. El episodio finaliza cuando se cumple alguna de las siguientes condiciones:

1. El efector final alcanza la meta dentro de las tolerancias de precisión establecidas por el *curriculum* de aprendizaje.

2. El robot entra en colisión consigo mismo o con el suelo.

3. Se alcanza el número máximo de pasos permitido.

Este diseño asegura que el agente se expone a una amplia variedad de situaciones representativas del espacio de trabajo, manteniendo un equilibrio entre diversidad de experiencias y coste computacional.

En conjunto, las decisiones adoptadas en la formulación del problema responden a los principios de generalización, modularidad y escalabilidad planteados en la hipótesis de esta tesis. El espacio de estados y acciones garantiza que la política aprenda un comportamiento que puede aplicarse en cualquier región del espacio de trabajo; la función de recompensa promueve soluciones eficientes; y la estructura de los episodios proporciona diversidad de experiencias que facilitan la robustez de la política aprendida.

5.2 Diseño del entorno de entrenamiento

El diseño del entorno de entrenamiento constituye un elemento clave en el proceso de obtención de políticas generalizables mediante DRL. El objetivo principal es garantizar que el agente interactúe con una representación del robot y del espacio de trabajo que sea lo suficientemente fiel para capturar las restricciones geométricas del problema, pero a la vez lo bastante ligera para permitir millones de interacciones en un tiempo computacional razonable.

En este contexto, se integra el modelo cinemático del manipulador UR3e dentro del motor de simulación *PyBullet*, ya empleado en la fase de generación de datos. Esta integración no se limita a un caso concreto, sino que sigue un procedimiento general basado en el uso de modelos en formato estandarizado

URDF. De esta manera, el entorno puede cargarse con cualquier robot que disponga de su descripción en URDF, lo que lo convierte en un marco adaptable y reutilizable para diferentes plataformas y escenarios de simulación.

La elección de este enfoque responde a varios criterios de diseño:

- **Consistencia con la hipótesis de trabajo**: La planificación se aborda desde un enfoque geométrico, por lo que la simulación se centra en la cinemática y en la detección de colisiones, omitiendo la dinámica. Esto permite reducir el coste de cómputo sin comprometer la validez de las trayectorias.

- **Determinismo y reproducibilidad**: PyBullet ofrece control total sobre la semilla de generación aleatoria y sobre las funciones de simulación, garantizando que las mismas configuraciones iniciales y acciones producen trayectorias idénticas. Esta propiedad es esencial para depurar, comparar y validar los experimentos de DRL.

- **Compatibilidad con Gymnasium**: Se desarrolla un entorno personalizado que cumple con la interfaz estándar de Gymnasium (Towers et al., 2024), lo que permite integrar de forma directa algoritmos de DRL que respeten la interfaz. Gracias a esta modularidad, el mismo entorno puede ser empleado para entrenar políticas en robots de distinta morfología sin necesidad de modificar la lógica del agente.

- **Fidelidad cinemática**: En el caso del UR3e, el modelo en URDF empleado fue extraído a partir de la calibración del robot real, lo que mejora la precisión de la cinemática directa e inversa y contribuye a reducir la brecha *sim-to-real*. Este mismo procedimiento puede aplicarse a otros robots siempre que se disponga de un URDF calibrado.

- **Modelado del entorno**: Además del manipulador, el entorno incluye elementos adicionales descritos en URDF, como un plano que representa el suelo y una pinza OnRobot RG2 integrada como efector final. Esto permite que el agente aprenda a evitar colisiones no solo con el propio robot, sino también con el entorno inmediato. De forma análoga, pueden integrarse otros obstáculos o escenarios más complejos mediante la misma interfaz.

La integración del modelo en PyBullet se implementa de forma que, en cada paso de simulación, el agente recibe un vector de observación conforme a la definición establecida en la Sección 5.1.1. A continuación, aplica una acción en el espacio articular según lo descrito en la Sección 5.1.2, y el entorno actualiza la configuración del robot mediante el solver cinemático interno de PyBullet. Este bucle define de forma natural la dinámica del MDP, con una transición determinista entre estados.

La figura 4.2 ilustra la configuración del entorno de simulación, destacando la disposición del manipulador UR3e y la pinza OnRobot RG2. La figura 5.1 presenta un esquema general y simplificado del diseño del entorno de entrenamiento, mostrando cómo se integran los diferentes componentes (modelo en URDF, motor de simulación PyBullet, interfaz de Gymnasium y agente DRL).

Finalmente, el entorno registra métricas adicionales (distancia a la meta, colisión, número de pasos) que se utilizan para calcular la recompensa descrita en la Sección 5.1.4 y para determinar las condiciones de finalización del episodio. De esta forma, la integración del modelo cinemático en PyBullet proporciona un marco coherente, eficiente y extensible para entrenar políticas de planificación geométrica basadas en DRL, manteniendo su aplicabilidad a diferentes robots y escenarios gracias al uso del formato estandarizado URDF.

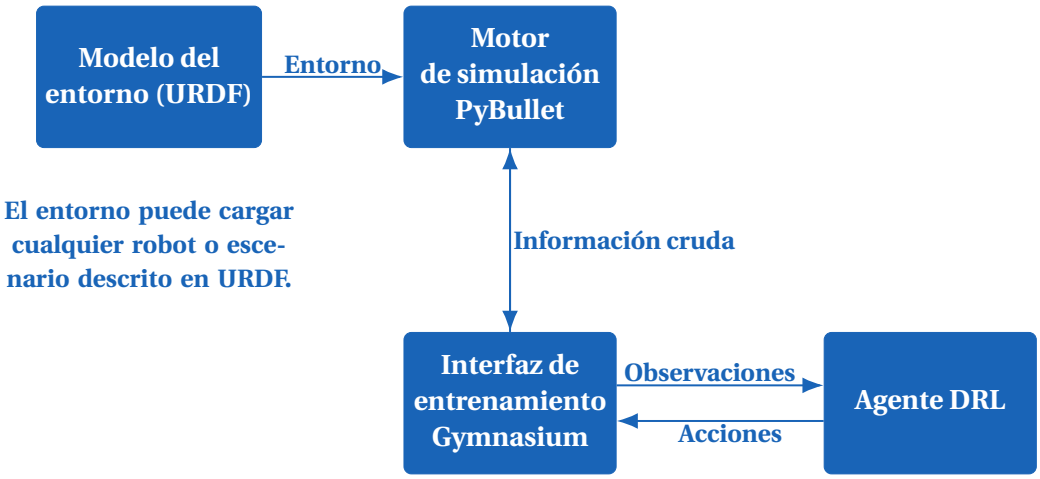

Figura 5.1: Esquema general del diseño del entorno de entrenamiento. El uso de modelos en formato URDF permite cargar diferentes robots y escenarios, integrándolos con PyBullet y la interfaz estándar de Gymnasium.

5.3 Evaluación de algoritmos y ajuste de hiperparámetros

El entrenamiento de políticas mediante DRL requiere la selección de un algoritmo de optimización adecuado y la correcta configuración de sus hiperparámetros. Una elección inadecuada puede derivar en problemas de inestabilidad, sobreajuste o falta de convergencia, especialmente en entornos con espacios de estado y acción de alta dimensionalidad, como ocurre en la planificación geométrica de manipuladores robóticos. Por ello, en esta tesis se ha llevado a cabo una evaluación comparativa de diferentes algoritmos, complementada con un proceso sistemático de ajuste de hiperparámetros mediante *Optuna* (Akiba et al., 2019; Fidalgo et al., 2023).

5.3.1 Evaluación comparativa de algoritmos

Se evaluaron cuatro de los algoritmos más representativos en tareas de control continuo:

- **Proximal Policy Optimization (PPO)**: Algoritmo *on-policy* ampliamente utilizado en robótica, caracterizado por su estabilidad en entornos de baja dimensionalidad, pero limitado en eficiencia muestral (Schulman et al., 2017).
- **Twin Delayed Deep Deterministic Policy Gradient (TD3)**: Algoritmo *off-policy* que extiende DDPG, diseñado para mejorar la estabilidad y evitar la sobreestimación de valores de acción (Fujimoto et al., 2018).
- **Soft Actor-Critic (SAC)**: Algoritmo *off-policy* basado en la maximización de la entropía, que promueve la exploración eficiente en espacios continuos y ha demostrado ser especialmente competitivo en tareas de control robótico (Haarnoja et al., 2018; Tang et al., 2024).
- **Deep Deterministic Policy Gradient (DDPG)**: Algoritmo *off-policy* que combina la política determinista con la optimización de la función de valor, siendo efectivo en entornos de acción continua (Lillicrap et al., 2015).

Los cuatro algoritmos se entrenaron bajo las mismas condiciones experimentales, incluyendo:

- Misma estructura de red neuronal (dos capas ocultas de 256 neuronas con activación por tangente hiperbólica).
- Mismo entorno de entrenamiento implementado en *PyBullet*.
- Conjunto de observaciones y acciones descrito en la Sección 5.1.
- Condiciones homogéneas de inicialización y *seed* aleatoria controlada para garantizar reproducibilidad.

La comparación se basó en métricas de convergencia (tasa de éxito en episodios de validación), estabilidad y eficiencia muestral (número de pasos hasta alcanzar un rendimiento dado).

5.3.2 Optimización de hiperparámetros

Para cada uno de los algoritmos evaluados, se realizó un ajuste sistemático de los hiperparámetros clave que influyen en el rendimiento del entrenamiento. Este proceso se centró en realizar entrenamientos reducidos con condiciones deterministas y variando sus hiperparámetros utilizando *Optuna*, una librería de optimización secuencial basada en modelos. Esta herramienta permite explorar automáticamente un espacio de búsqueda definido por el investigador, ajustando los parámetros en función de la métrica de rendimiento seleccionada. De esta manera se definieron rangos de búsqueda para los hiperparámetros más relevantes de cada algoritmo. Además de los hiperparámetros, se optimizaron los coeficientes de la función de recompensa, estableciendo como objetivo de optimización la tasa de éxito en el entorno de entrenamiento.

La optimización de los hiperparámetros se realizó utilizando los siguientes rangos de búsqueda para cada algoritmo:

- **Tasa de aprendizaje**: $[1 \times 10^{-5}, 1 \times 10^{-3}]$
- **Tamaño de lote**: $\{256, 512, 1024, 2048, 4096\}$

- **Factor de descuento** (γ): [0.90, 0.999]

La optimización de los coeficientes de la función de recompensa se realizó utilizando los siguientes rangos de búsqueda:

- **Coeficiente de posición** (α): [0.1, 1.0]
- **Coeficiente de orientación** (β): [0.1, 100.0]

La mayor tasa de éxito se obtuvo con $\alpha = 0.2$ y $\beta = 10.0$. Los resultados obtenidos para la elección de hiperparámetros se incluyen más adelante en la Tabla 5.1.

Después de realizar este primer ajuste de hiperparámetros, se realizaron varios entrenamientos completos de cada algoritmo con los hiperparámetros optimizados para comparar su rendimiento en condiciones equivalentes. Para cada algoritmo, se seleccionó la configuración que obtuvo la mayor recompensa acumulada durante el proceso de optimización y se entrenó durante varios millones de pasos con distintas semillas, evaluando periódicamente la tasa de éxito, recompensa, longitud de episodio y evolución del curriculum. En concreto, se realizaron 3 entrenamientos completos por cada algoritmo con diferentes semillas, y se promediaron los resultados para obtener métricas más robustas. Debido a la baja variabilidad observada en los resultados, se consideró que 3 repeticiones eran suficientes para evaluar el rendimiento de cada algoritmo. Cada una de las gráficas presentadas a continuación muestra la media y la desviación estándar de las 3 repeticiones para cada algoritmo.

La Figura 5.2 muestra la tasa de éxito de cada uno de los algoritmos evaluados con los hiperparámetros optimizados. Es decir, la cantidad de episodios en los que se ha alcanzado la meta. En esta se puede observar como SAC y PPO presentan una convergencia más rápida y estable en comparación con TD3 y DDPG, siendo SAC el algoritmo que alcanza los mejores resultados en términos de velocidad de aprendizaje y estabilidad. Tanto en el caso de TD3 como en el de DDPG, se interrumpió el entrenamiento debido a la falta de convergencia, incluso tras ajustar sus hiperparámetros.

La Figura 5.3 muestra la recompensa acumulada de cada uno de los algoritmos evaluados con los hiperparámetros optimizados. En esta se puede observar

como SAC es el algoritmo que alcanza la mayor recompensa acumulada, seguido de PPO. La significativa diferencia en la recompensa acumulada entre SAC y PPO se debe al progreso del *curriculum* de entrenamiento, que permitió a SAC beneficiarse de mejores recompensas al alcanzar metas cada vez más exigentes.

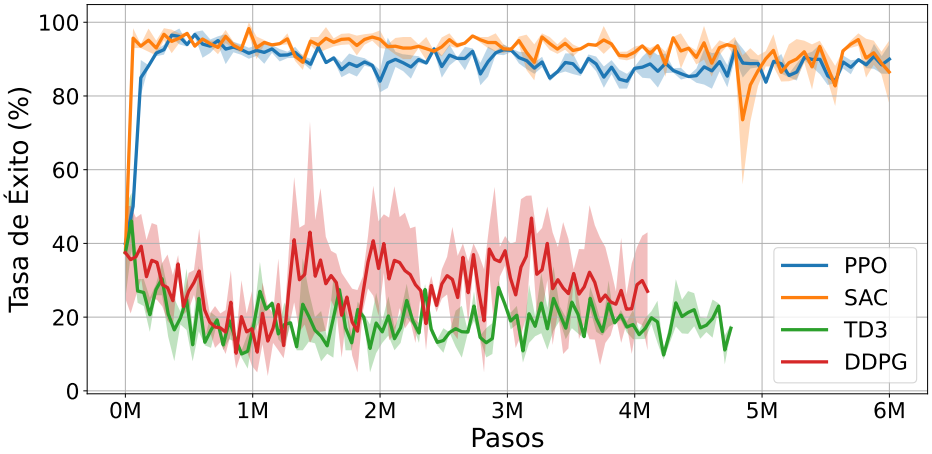

Figura 5.2: Tasa de éxito media (color sólido) junto con la desviación estándar (color claro) de cada uno de los algoritmos evaluados con los hiperparámetros optimizados.

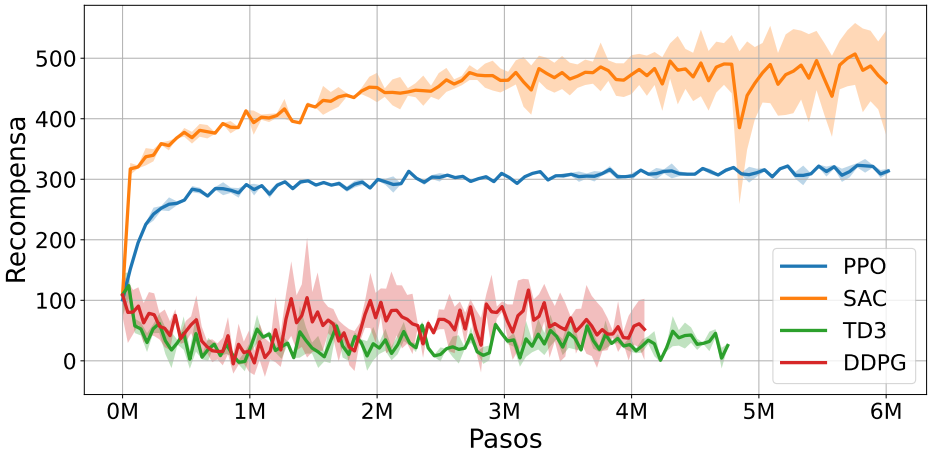

Figura 5.3: Recompensa acumulada media (color sólido) junto con la desviación estándar (color claro) de cada uno de los algoritmos evaluados con los hiperparámetros optimizados.

La Figura 5.4 muestra la longitud de episodio de cada uno de los algoritmos

evaluados con los hiperparámetros optimizados. En esta se puede observar como tanto SAC como PPO son los algoritmos que alcanzan la menor longitud de episodio. La longitud de episodio es una métrica importante en este contexto, ya que episodios más cortos indican que el agente está aprendiendo a alcanzar la meta de manera más eficiente.

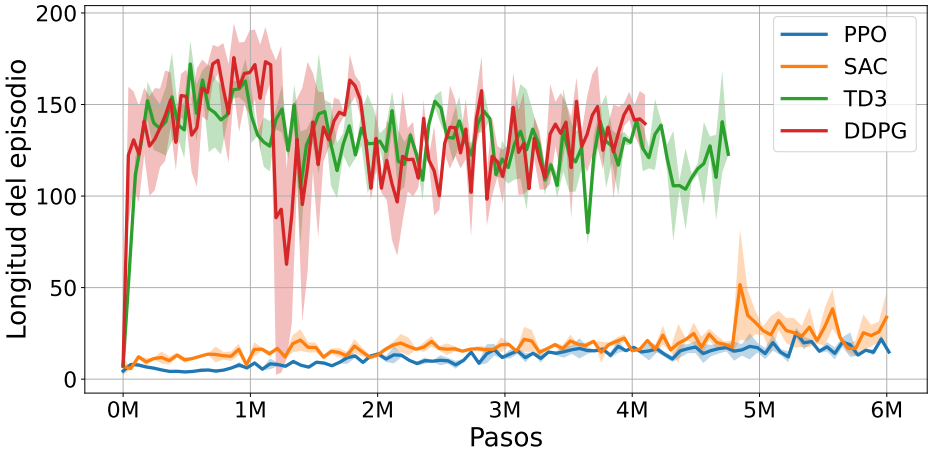

Figura 5.4: Longitud de episodio media (color sólido) junto con la desviación estándar (color claro) de cada uno de los algoritmos evaluados con los hiperparámetros optimizados.

Por último, en las Figuras 5.5 y 5.6 se muestra la evolución del *curriculum* de posición y orientación de entrenamiento para los algoritmos. Se observa como para el caso de SAC y PPO el *curriculum* progresa de manera constante a lo largo del entrenamiento. En el caso de SAC alcanzando las tolerancias finales de precisión en traslación y rotación, mientras que PPO no converge tan rápido. Para el caso de TD3 y DDPG no ocurre evolución alguna en el *curriculum*, ya que no son capaces de alcanzar las metas consecutivas necesarias, por lo que no se cumple la condición para avanzar en el *curriculum*.

Todos estos resultados indican que SAC es el algoritmo más adecuado para el problema de planificación geométrica planteado, obteniendo un rendimiento superior en términos de tasa de éxito, recompensa acumulada y eficiencia en la longitud de episodio. La tabla 5.1 resume los hiperparámetros finales utilizados en el entrenamiento de SAC.

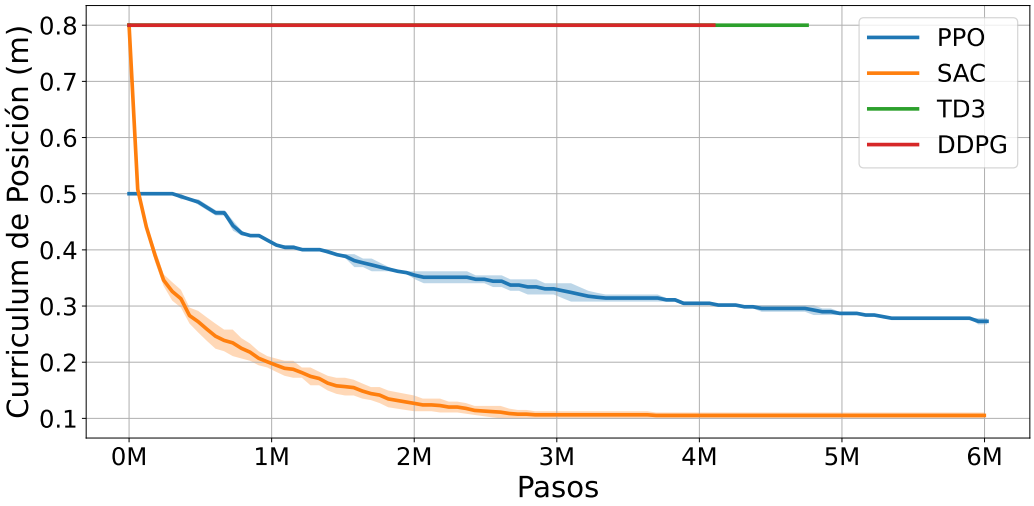

Figura 5.5: Progreso del curriculum de posición medio (color sólido) junto con la desviación estándar (color claro) de cada uno de los algoritmos evaluados con los hiperparámetros optimizados.

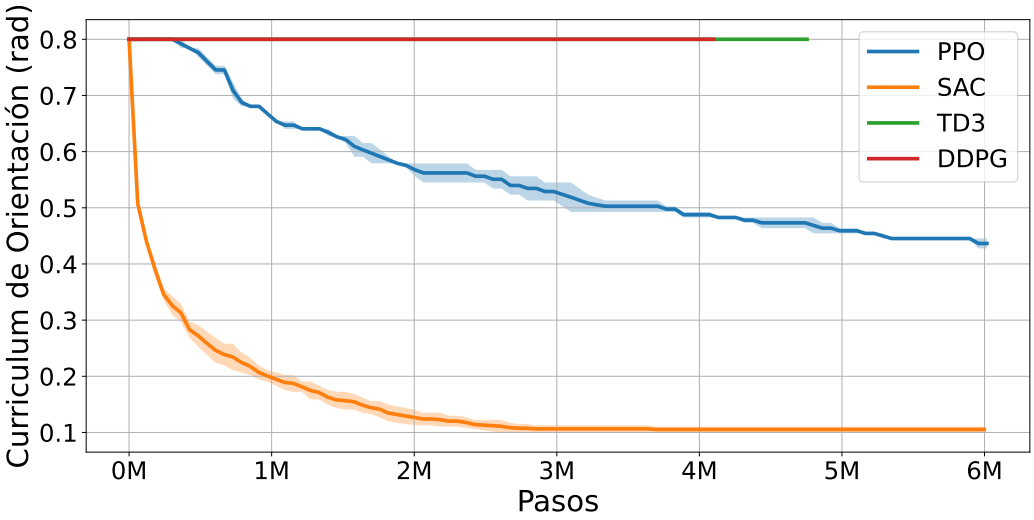

Figura 5.6: Progreso del curriculum de orientación medio (color sólido) junto con la desviación estándar (color claro) de cada uno de los algoritmos evaluados con los hiperparámetros optimizados.

Tabla 5.1: Hiperparámetros finales empleados en el entrenamiento con SAC.

Parámetro	**Valor**
Tasa de aprendizaje	3×10^{-4}
Tamaño de lote	4096
Factor de descuento (γ)	0.99
Tamaño del *replay buffer*	1×10^{6}

5.4 Inyección de experiencia experta

El aprendizaje por refuerzo profundo aplicado a planificación de trayectorias presenta un coste computacional elevado, debido a la gran dimensionalidad del espacio de estados. En este contexto, el agente requiere millones de interacciones con el entorno antes de alcanzar políticas útiles, lo que supone un obstáculo tanto en términos de tiempo de entrenamiento como de eficiencia muestral. Para mitigar este problema, se ha implementado inyección de experiencia experta con el fin de acelerar el proceso de aprendizaje y obtener políticas más precisas.

El procedimiento parte del conjunto de trayectorias generadas con los planificadores clásicos de OMPL, previamente descritos en el Capítulo 4.5. Dichas trayectorias cumplen dos funciones: por un lado, constituyen una base de comparación con los métodos de DRL; por otro, proporcionan una fuente de conocimiento experto que acelera la adquisición de comportamientos válidos.

Cada trayectoria experta se descompone en una secuencia de transiciones (s_t, a_t, r_t, s_{t+1}), donde:

- s_t corresponde al estado observado en el tiempo t,
- a_t se calcula como el incremento articular $\Delta q = q_{t+1} - q_t$,
- r_t se obtiene aplicando la función de recompensa híbrida definida en la Sección 5.1.4,
- s_{t+1} es el estado alcanzado tras aplicar a_t.

Estas transiciones se introducen directamente en el *replay buffer* del agente junto con las transiciones generadas mediante exploración autónoma. El esquema de inyección seguido es adaptativo: después de cada episodio fallido, se

añade al *buffer* una trayectoria experta completa, de modo que el agente observe secuencias exitosas de principio a fin. Conforme la política mejora y la tasa de éxito aumenta, la proporción de muestras expertas disminuye de manera natural, favoreciendo una transición suave desde la imitación inicial hasta el aprendizaje autónomo.

5.4.1 Impacto en la convergencia y estabilidad

El efecto principal de la inyección de experiencia experta se observa en dos aspectos:

1. **Velocidad de convergencia**: Los episodios exitosos inyectados permiten al agente evitar fases largas de exploración aleatoria infructuosa. Esto acelera la adquisición de las estrategias básicas de movimiento y reduce en varios órdenes de magnitud el número de pasos necesarios para alcanzar tasas de éxito significativas.

2. **Estabilidad del entrenamiento**: Al exponer al agente a trayectorias completas y libres de colisiones desde las primeras fases, se mitiga la aparición de comportamientos espurios (como oscilaciones o movimientos erráticos cerca de la meta) y se reduce la varianza en el retorno acumulado entre episodios consecutivos.

3. **Calidad de la política final**: La política entrenada con experiencia experta no solo converge más rápido, sino que también alcanza una mayor precisión en la reducción del error pose–objetivo, especialmente en regiones del espacio de trabajo menos exploradas durante la fase inicial de entrenamiento.

5.4.2 Comparación con entrenamiento sin experiencia experta

Para evaluar la influencia de esta técnica, se entrenaron agentes en dos condiciones: con y sin inyección de experiencia experta. En ambos casos se utilizaron

idénticos hiperparámetros, la misma función de recompensa y el mismo esquema de *curriculum learning*, de forma que las diferencias pudieran atribuirse exclusivamente a la técnica de aceleración.

La Figura 5.7 compara la tasa de éxito en el conjunto de validación entre ambos agentes a lo largo del entrenamiento. En ambos casos se observa una tasa de éxito elevada, en este caso, para los episodios iniciales y aproximadamente hasta los 4 millones de pasos, el agente entrenado con experiencia experta supera ligeramente al agente sin esta técnica. A partir de los 5 millones de pasos se observa una ligera reducción de la tasa de éxito para el agente con experiencia experta, llegando incluso a ser superado por el agente sin esta técnica. Esto se debe a que, mientras que el agente al que se le inyecta la experiencia experta continúa avanzando de *curriculum*, y por lo tanto, resolviendo metas más exigentes, el agente sin experiencia experta se estanca en un nivel intermedio del *curriculum*, alcanzando una tasa de éxito elevada para las metas menos exigentes. A pesar de esto, el agente entrenado con experiencia experta alcanza las metas más exigentes del *curriculum*, lo que se traduce en una mayor precisión y capacidad de generalización.

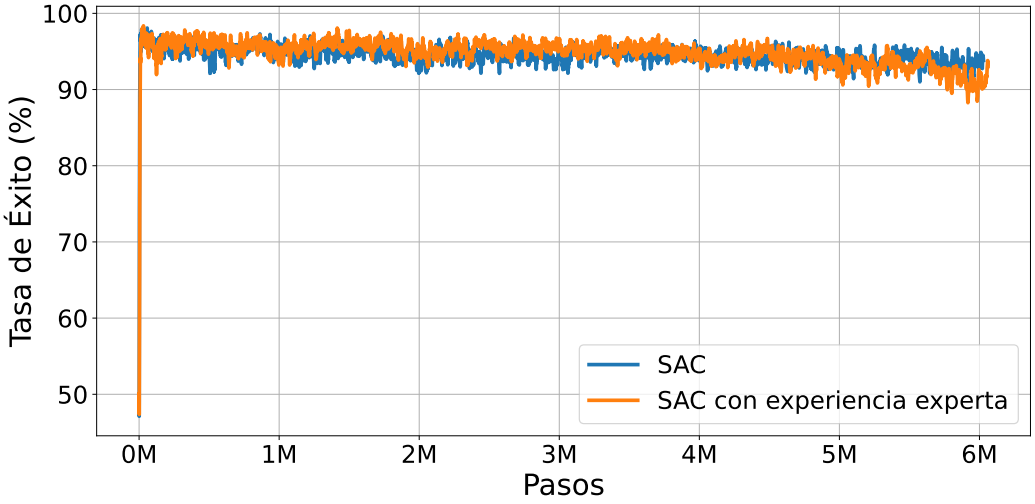

Figura 5.7: Comparación de la tasa de éxito en el conjunto de validación entre agentes entrenados con y sin inyección de experiencia experta.

Las Figuras 5.8 y 5.9 muestran el progreso del curriculum de entrenamiento para el algoritmo SAC, con y sin inyección de experiencia experta. En este caso,

el agente entrenado con inyección de experiencia experta muestra un avance más rápido en el curriculum, alcanzando metas más complejas en menos pasos. Además, en el lapso de los 6 millones de pasos, el agente con experiencia experta alcanza el nivel más alto del curriculum, mientras que el agente sin esta técnica no solo se queda atrás, sino que además muestra indicios de estancamiento en niveles previos.

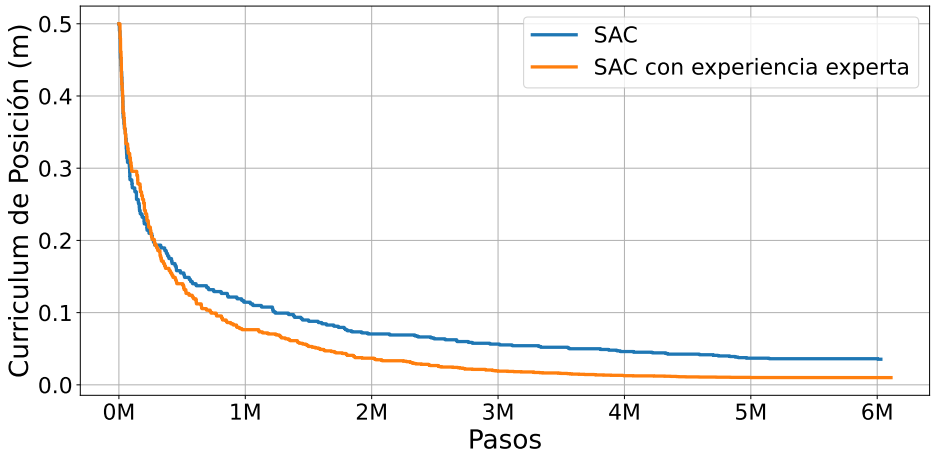

Figura 5.8: Progreso del curriculum de posición para el entrenamiento con SAC con y sin inyección de experiencia experta.

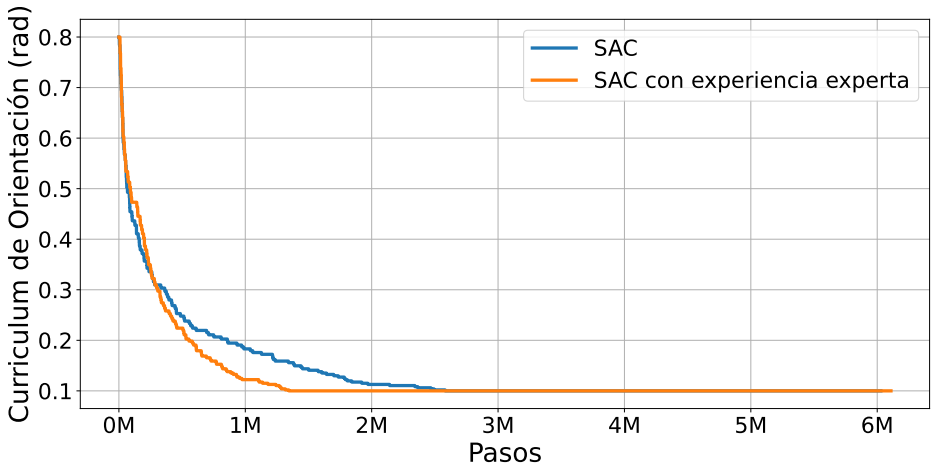

Figura 5.9: Progreso del curriculum de orientación para el entrenamiento con SAC con y sin inyección de experiencia experta.

La Figura 5.10 muestra la evolución de la función de recompensa acumulada en el conjunto de validación para ambos agentes entrenados con SAC. La recompensa obtenida por el agente con inyección de experiencia experta es consistentemente más alta, debido a la progresión más eficiente a través del curriculum, los valores obtenidos son más altos en cada etapa del entrenamiento.

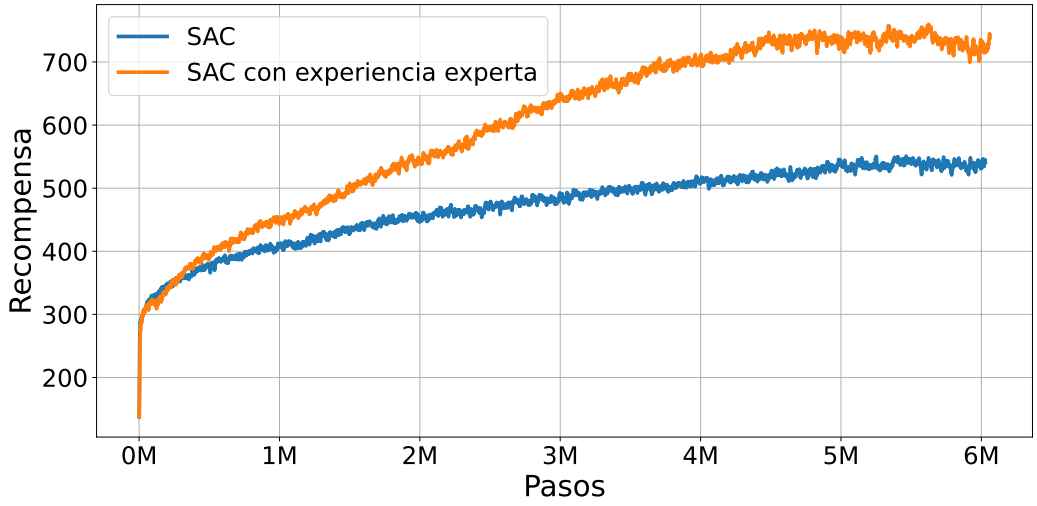

Figura 5.10: Comparación de la función de recompensa entre agentes entrenados con y sin inyección de experiencia experta.

Estos resultados muestran que la inyección de experiencia experta es una técnica efectiva para acelerar el entrenamiento de políticas en entornos de planificación de trayectorias. Los resultados muestran que, gracias a la inyección de experiencia experta, el agente es capaz de alcanzar los niveles objetivo de tolerancia en aproximadamente 5 millones de pasos, mientras que el agente sin esta técnica no sólo no logra alcanzar estos valores, si no que se estanca en niveles previos de posición.

En conclusión, la incorporación de experiencia experta no solo reduce drásticamente el tiempo de entrenamiento requerido, también mejora la estabilidad de la política aprendida.

5.5 Influencia del *curriculum learning*

El *curriculum learning* constituye uno de los elementos clave en el diseño de la política entrenada, al permitir una progresión gradual de la dificultad de la tarea mediante el ajuste adaptativo de las tolerancias de éxito en posición y orientación. Para analizar su influencia se llevaron a cabo dos experimentos controlados utilizando el algoritmo SAC como base, con idénticos hiperparámetros y condiciones de inicialización, variando únicamente la presencia o ausencia de *curriculum learning* y sin inyección de experiencia experta.

En el primer experimento se realizaron cinco entrenamientos independientes con SAC, inicializados con diferentes semillas, fijando desde el inicio las tolerancias de éxito en sus valores más exigentes: $\varepsilon_p = 0.0005$ m y $\varepsilon_r = 0.1$ rad. Bajo estas condiciones, el agente no dispone de recompensas tempranas y se enfrenta desde el inicio a la máxima dificultad de la tarea.

En el segundo experimento se aplicó el mismo procedimiento, pero incorporando el esquema de *curriculum learning* descrito en la Sección 5.1.4. De este modo, las tolerancias iniciales se establecieron en valores laxos ($\varepsilon_p^{(0)} = 0.5$ m, $\varepsilon_r^{(0)} = 0.8$ rad) y se fueron reduciendo progresivamente en función de los éxitos consecutivos del agente hasta alcanzar las tolerancias objetivo.

Las Figuras 5.11 y 5.12 muestran la evolución de la tasa de éxito y de la recompensa acumulada en el conjunto de validación para ambos casos. En el experimento sin *curriculum learning*, la tasa de éxito permanece a cero durante todo el entrenamiento, indicando que el agente es incapaz de alcanzar metas de alta precisión. La recompensa acumulada también se mantiene en niveles bajos, reflejando la ausencia de progresos significativos. En contraste, el agente entrenado con *curriculum learning* muestra una tasa de éxito creciente, alcanzando valores cercanos al 100 % tras aproximadamente 6 millones de pasos. La recompensa acumulada también experimenta un aumento sustancial, evidenciando una mejora continua en el desempeño del agente.

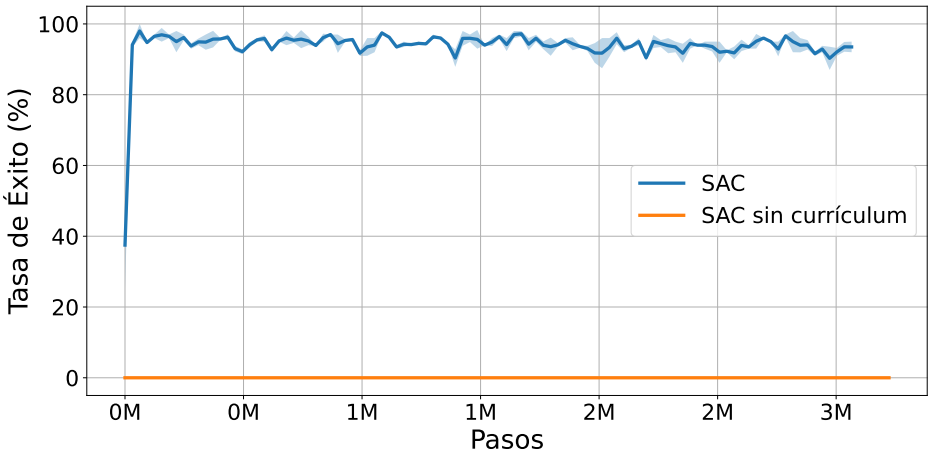

Figura 5.11: Comparación de la tasa de éxito entre agentes entrenados con y sin *curriculum learning*, se muestra la media (color sólido) y la desviación estándar (color claro) de cinco semillas independientes.

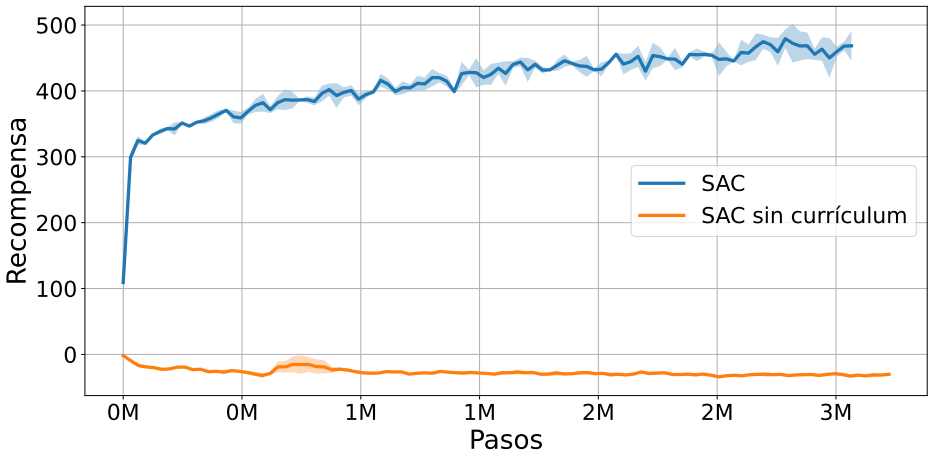

Figura 5.12: Comparación de la recompensa entre agentes entrenados con y sin *curriculum learning*, se muestra la media (color sólido) y la desviación estándar (color claro) de cinco semillas independientes.

5.5.1 **Política final**

El entrenamiento final de la política se llevó a cabo con la implementación de todas las técnicas de aceleración descritas en la Sección 5.4. Esto incluyó la inyección de experiencia experta en el *replay buffer* del agente, así como la adaptación del esquema de *curriculum learning* para facilitar el aprendizaje de tareas más complejas. El agente se entrenó durante un total de 6 millones de pasos, utilizando el entorno de entrenamiento basado en PyBullet y la configuración de hiperparámetros optimizada previamente (ver Tabla 5.1).

Para este entrenamiento, se utilizó el conjunto de datos generado en la fase inicial. En concreto, este conjunto de datos se dividió en conjuntos de entrenamiento, validación y prueba, con un 80 %, 10 % y 10 % de las trayectorias, respectivamente. El conjunto de entrenamiento se utilizó para entrenar al agente de DRL, mientras que los conjuntos de validación y prueba se emplearon para evaluar el rendimiento tanto de los métodos clásicos como de los basados en DRL para la planificación de movimientos. Esta división del conjunto de datos se realizó teniendo en cuenta los criterios expuestos en la Sección 4.5, es decir, cada uno de los subconjuntos contiene una representación equilibrada de trayectorias cortas, medias y largas, garantizando así una evaluación justa y representativa del rendimiento de los diferentes enfoques.

A continuación se muestran los resultados del entrenamiento realizado. En la Figura 5.13 se presenta la tasa de éxito junto con el progreso del curriculum de entrenamiento. Se observa que la tasa de éxito aumenta de manera consistente durante los primeros 3 millones de pasos, alcanzando valores por encima del 95 % de tasa de éxito. El nivel máximo de tolerancia en orientación se obtiene pasado el millón de pasos, mientras que el nivel máximo de tolerancia en posición se alcanza alrededor de los 5 millones de pasos. A medida que el curriculum de posición avanza y, por tanto, las metas se vuelven más exigentes, la tasa de éxito experimenta una ligera disminución, estabilizándose en torno al 94 %.

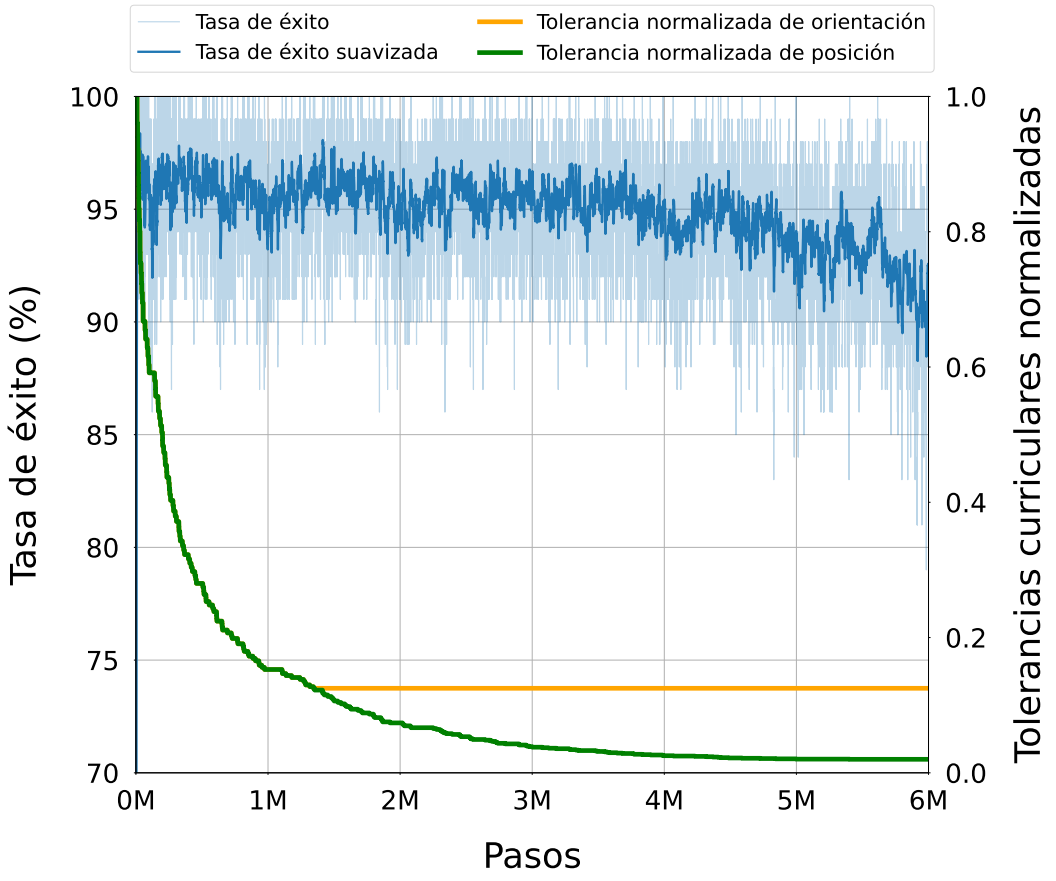

Figura 5.13: Tasa de éxito junto con el progreso del curriculum de entrenamiento.

La Figura 5.14 muestra la evolución de la recompensa acumulada durante el entrenamiento. Se observa una tendencia ascendente de manera consistente en la recompensa. Este incremento refleja la mejora continua del agente en la resolución de las tareas planteadas, beneficiándose tanto del *curriculum learning* como de la inyección de experiencia experta. A medida que el agente progresa en los diferentes niveles de dificultad, la recompensa continúa creciendo, premiando la capacidad del agente para alcanzar metas más exigentes con mayor precisión y eficiencia. Durante el último medio millón de pasos, la recompensa decae ligeramente después de haber estado estabilizada en el último nivel del *curriculum*.

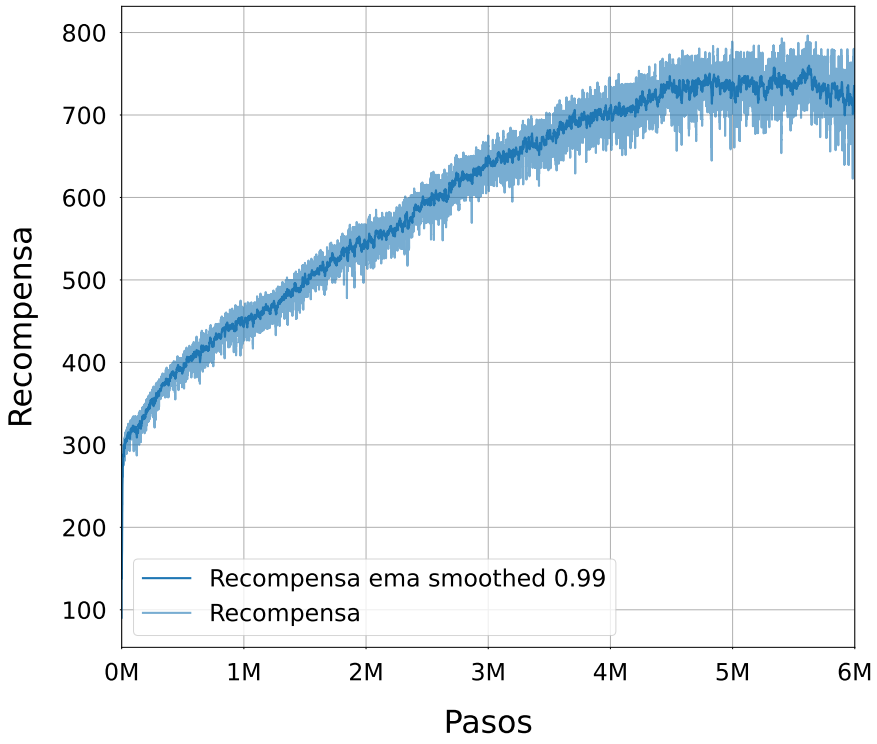

Figura 5.14: Recompensa promedio durante el entrenamiento.

La Figura 5.15 presenta la longitud de episodio a lo largo del entrenamiento. Se observa un aumento gradual en la longitud de los episodios, ya que, durante los primeros millones de pasos, el agente resuelve tareas relativamente sencillas que requieren menos pasos para alcanzar la meta. Conforme avanza el *curriculum* y las metas se vuelven más exigentes, la longitud de los episodios aumenta, reflejando la necesidad de ejecutar trayectorias más largas y complejas. Al alcanzar los niveles máximos de dificultad, la longitud de los episodios se estabiliza en torno a 20 configuraciones, indicando que el agente ha aprendido estrategias eficientes para resolver tareas de mayor complejidad sin incrementar innecesariamente el número de pasos. Las mayores longitudes registradas no superan las 50 configuraciones, lo que sugiere que el número máximo de pasos por episodio (200) es suficientemente alto para permitir la resolución de las tareas más complejas sin que el agente se vea limitado por este parámetro.

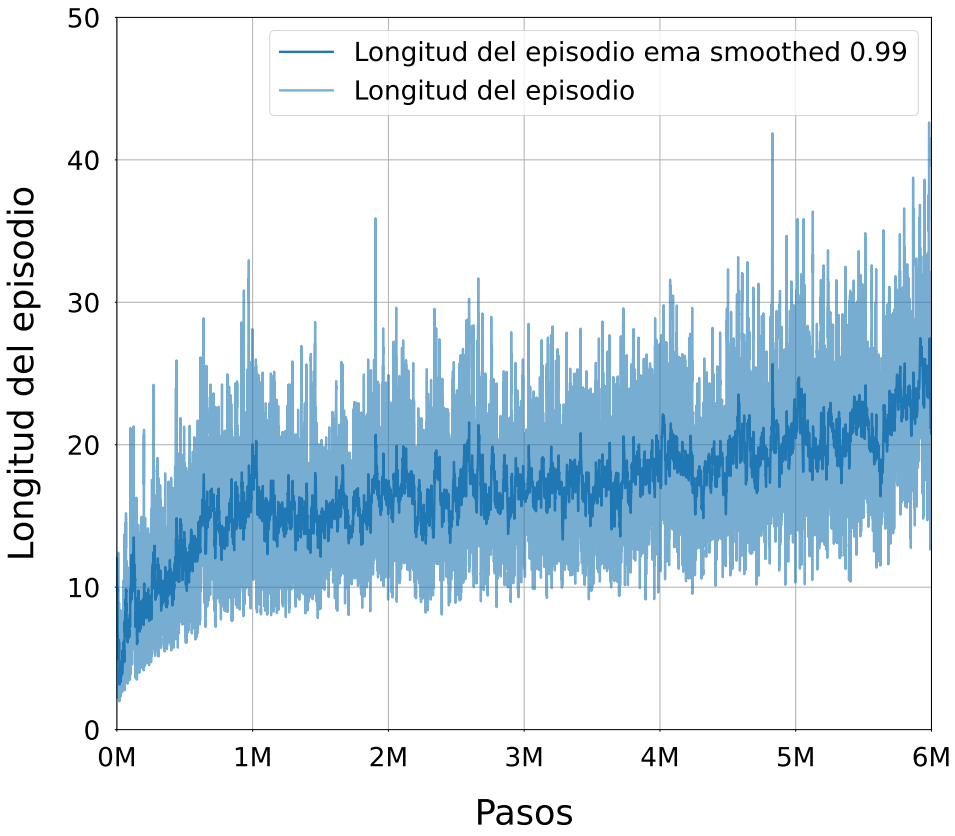

Figura 5.15: Longitud de episodio durante el entrenamiento.

5.5.2 **Resultado final**

Durante el entrenamiento se realizaron rutinas de evaluación periódicas para medir el rendimiento de la política y obtener la mejor política final. Los mejores resultados para el último nivel del *curriculum* se lograron alrededor de 5.5 millones de pasos.

Como resultado se obtiene una política que, con una pose objetivo y el estado actual del robot (configuración, pose y colisión), es capaz de sugerir el siguiente movimiento a realizar en el espacio de configuración del robot para alcanzar la meta. Gracias a este diseño, la política es capaz de adaptarse a diferentes casos de uso, logrando resolver gran parte de problemas cinemáticos de brazos articulados.

Para todos los casos de uso, se aplica un perfilado temporal, transformando la trayectoria geométrica total o parcial obtenida de la política en una trayectoria completa con perfiles de velocidad y aceleración, que garanticen un movimiento ejecutable por el actuador teniendo en cuenta sus restricciones físicas. Este perfilado temporal se realiza utilizando el algoritmo TOPPRA.

Un aspecto esencial en la aplicación práctica de la política aprendida es la transición desde trayectorias geométricas, producidas por la política, hacia trayectorias físicamente ejecutables en el robot real. La salida de la política corresponde a una secuencia de configuraciones articulares que conectan un estado inicial y una meta en el espacio de configuración. Estas trayectorias garantizan la factibilidad geométrica (sin colisiones y dentro de los límites articulares), pero no incorporan información temporal ni dinámica, por lo que no pueden ejecutarse directamente en un robot físico.

Para dotar a las trayectorias de un perfil temporal óptimo, se emplea el algoritmo TOPPRA (Pham and Pham, 2018). Este método toma como entrada una trayectoria geométrica $\tau(s)$ parametrizada por el espacio de configuración, con $s \in [0,1]$, y resuelve un problema de optimización para asignar una función de tiempo $t(s)$ que define las velocidades y aceleraciones articulares en cada punto, de modo que se minimiza el tiempo total de ejecución sujeto a las restricciones dinámicas del manipulador.

Las restricciones consideradas incluyen las cotas máximas de velocidad y aceleración de cada articulación, determinadas por las especificaciones físicas de los actuadores del UR3e. Formalmente, el problema se plantea como:

$$\dot{q}_i(t) \in [-\dot{q}_{i,\max}, \dot{q}_{i,\max}], \quad \ddot{q}_i(t) \in [-\ddot{q}_{i,\max}, \ddot{q}_{i,\max}], \quad \forall i \in \{1,\ldots,6\} \tag{5.9}$$

TOPPRA resuelve estas restricciones mediante un análisis de alcance que garantiza que, en cada punto de la trayectoria, existe un perfil de velocidad y aceleración que respeta los límites articulares. Como resultado, se obtiene una trayectoria $\tau(t)$ con parametrización temporal explícita, en la que cada configuración articular no solo es cinemáticamente válida, sino también ejecutable en tiempo real por el hardware.

Este proceso aporta dos beneficios clave:

- Asegura que la política aprendida pueda transferirse al robot real respetando sus limitaciones físicas.

- Produce trayectorias optimizadas en el tiempo, lo cual mejora la eficiencia de ejecución sin comprometer la seguridad.

En términos prácticos, la combinación de nuestra política basada en DRL para generar trayectorias geométricas y la post-procesación con TOPPRA constituye un pipeline robusto y escalable para la planificación y ejecución de movimientos en manipuladores industriales.

5.5.3 **Casos de uso de la política entrenada**

La política final obtenida puede aplicarse de forma versátil en diferentes contextos. Esta flexibilidad refleja su carácter modular y generalizable, acorde a los objetivos de la tesis. En particular, se identifican tres casos de uso fundamentales: planificación de trayectorias libres, planificación con restricciones en el espacio de trabajo y planificación hacia metas móviles. Cada uno de ellos responde a necesidades distintas en aplicaciones robóticas y permite demostrar la escalabilidad de la política.

5.5.3.1 **Planificación de trayectorias libres**

Este caso constituye el escenario más elemental, en el que se desea conectar una configuración inicial con una meta en el espacio de trabajo sin restricciones intermedias. La política se ejecuta iterativamente para generar incrementos articulares hasta que se alcanza una de las condiciones de terminación:

1. La meta es alcanzada dentro de las tolerancias de precisión definidas en posición y orientación.

2. El robot entra en colisión consigo mismo o con el suelo.

3. Se alcanzan los 200 pasos máximos definidos por episodio.

Este esquema resulta directamente aplicable a problemas típicos de manipulación donde se requiere alcanzar un objetivo puntual, como posicionar una herramienta o mover un objeto desde una localización inicial a otra. La figura 5.16 representa esquemáticamente el proceso.

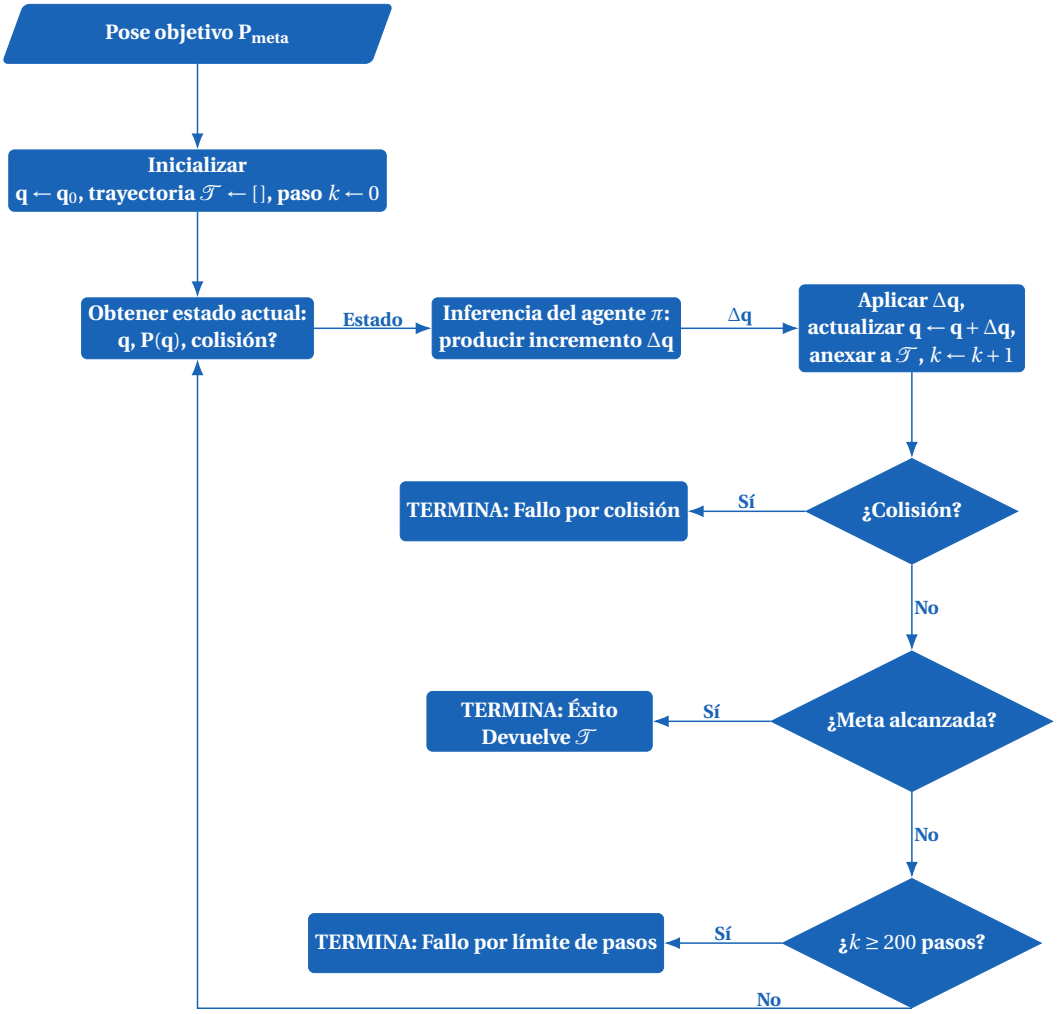

Figura 5.16: Diagrama de flujo del escenario elemental: conectar $\mathbf{q}_0$ con alguna de las configuraciones que logren $\mathbf{P}_{goal}$ mediante incrementos articulares generados por la política, con terminación por meta alcanzada, colisión, o por alcanzar los 200 pasos.

5.5.3.2 **Planificación de trayectorias con restricciones en el espacio de trabajo**

En muchos escenarios industriales y de servicio, la trayectoria deseada no se limita a un único par inicio–meta, sino que está compuesta por una secuencia de poses intermedias que deben alcanzarse en orden y que presentan ciertas restricciones cinemáticas. Este es el caso, por ejemplo, de procesos de ensamblaje, inspección, soldadura o pintura, donde el efector final debe recorrer un conjunto de posiciones predeterminadas en las que el efector final debe cumplir con ciertas orientaciones y velocidades.

La política se ejecuta de manera jerárquica: cada segmento entre dos poses consecutivas se planifica de forma independiente, siguiendo el mismo esquema de ejecución que en el caso anterior. Al completarse con éxito un segmento, se inicializa el siguiente hasta cubrir toda la secuencia de restricciones. Una vez obtenida la trayectoria geométrica completa que abarca todas las poses, se aplica el perfilado temporal mediante TOPPRA para garantizar la factibilidad dinámica de la trayectoria total.

Este procedimiento demuestra la modularidad de la política, que se integra como bloque base dentro de un planificador de alto nivel. La figura 5.17 muestra el proceso.

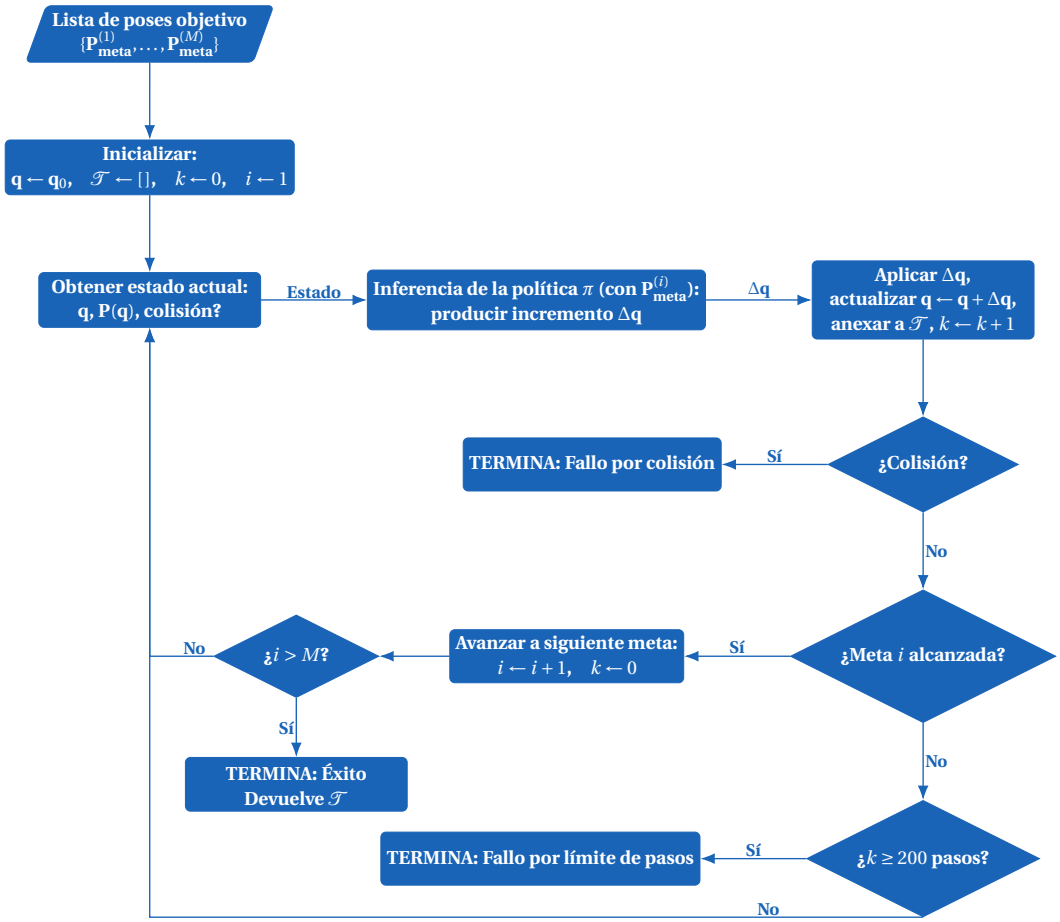

Figura 5.17: Diagrama de flujo para seguimiento de una trayectoria dada por una lista de poses en el espacio de trabajo que pueden incorporar restricciones geométricas: la política avanza meta a meta, reiniciando el contador de pasos por cada objetivo, y termina al completar la última meta o por fallo.

5.5.3.3 **Planificación hacia metas móviles**

Finalmente, se considera el caso de metas que cambian dinámicamente durante la ejecución. En este escenario, la meta puede desplazarse de un paso a otro, lo cual introduce una mayor incertidumbre y demanda adaptabilidad de la política. Ejemplos prácticos incluyen el seguimiento de objetos en movimiento en cintas transportadoras o la cooperación con otros robots o agentes humanos.

El procedimiento se mantiene análogo al de trayectorias libres, pero con dos diferencias principales:

- La meta se actualiza dinámicamente en cada paso, modificando la observación que recibe la política.

- Se incrementa el número máximo de pasos por episodio para compensar los cambios de meta, manteniendo los límites de colisión y las tolerancias de precisión.

La figura 5.18 representa esquemáticamente este proceso.

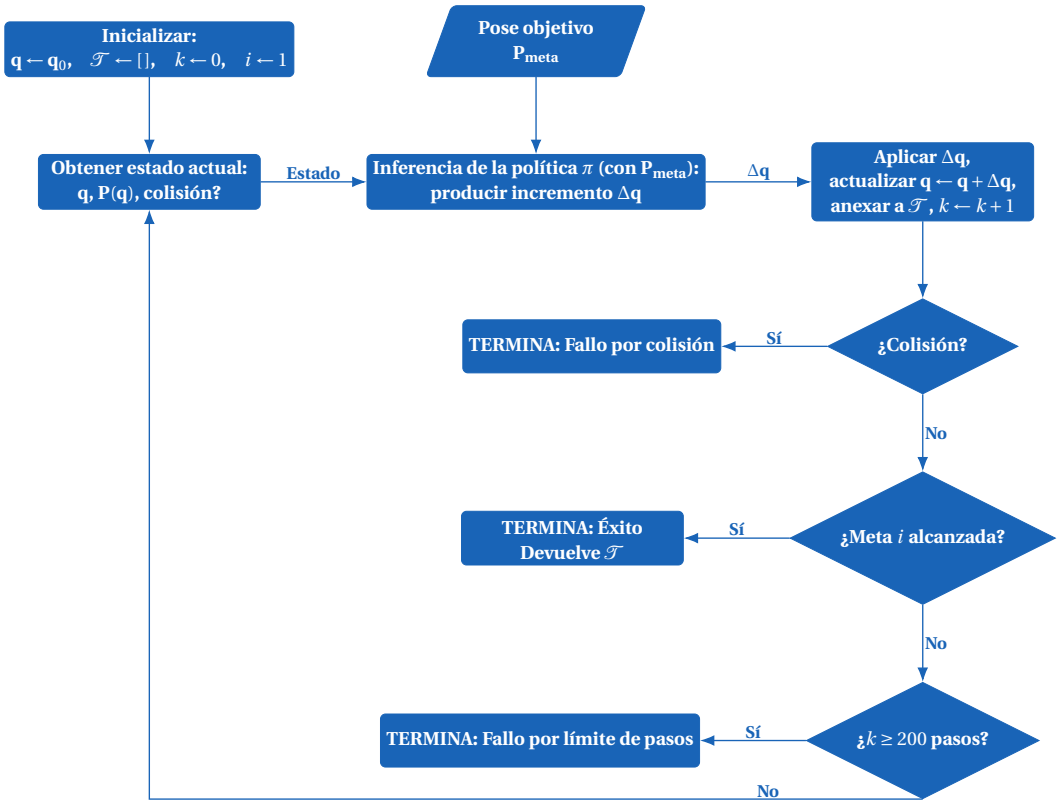

Figura 5.18: Diagrama de flujo para planificación hacia metas móviles: la política recibe una meta que puede cambiar en cada paso, y termina al alcanzar la meta, por colisión o por límite de pasos.

Estas aplicaciones evidencian que la política no es una solución *ad-hoc* restringida a un conjunto específico de trayectorias, sino un bloque funcional reutilizable que puede integrarse en arquitecturas de planificación más generales.

5.6 Transferencia al entorno real

La transferencia de la política aprendida al robot físico exige un *pipeline* de despliegue que preserve las garantías de sincronía y reproducibilidad establecidas en simulación. Para ello, se desarrolla un módulo en el marco del proyecto ACROBA (ACROBA Project, 2025), cuyo objetivo es ejecutar políticas entrenadas y mapear automáticamente observaciones y acciones a ROS, manteniendo compatibilidad con interfaces estándar de control. El módulo actúa como capa de integración entre la política y los *topics*/*services*/*actions* de ROS, y se diseña para operar tanto en configuraciones de tiempo real estricto como en esquemas donde el tiempo real no está garantizado. Esto permite adaptar la ejecución a las capacidades del controlador del robot y a los requisitos de seguridad, sin modificar la lógica de la política ni el agente de DRL.

Este módulo se concibe como una solución genérica y reutilizable, capaz de trabajar con distintos robots, controladores y políticas, simplemente ajustando un archivo de configuración. De este modo, se facilita la transferencia de políticas entrenadas en simulación a entornos reales, minimizando el esfuerzo de ingeniería adicional y preservando la modularidad del diseño. Aunque en este trabajo se valida con un robot Universal Robots UR3e y a la política desarrollada para la planificación de trayectorias, el enfoque es aplicable a una amplia gama de manipuladores y tareas.

La Figura 5.19 ilustra el *pipeline* de transferencia, donde la política entrenada se empaqueta en un *Action server* de ROS que expone una interfaz estándar para la ejecución de tareas.

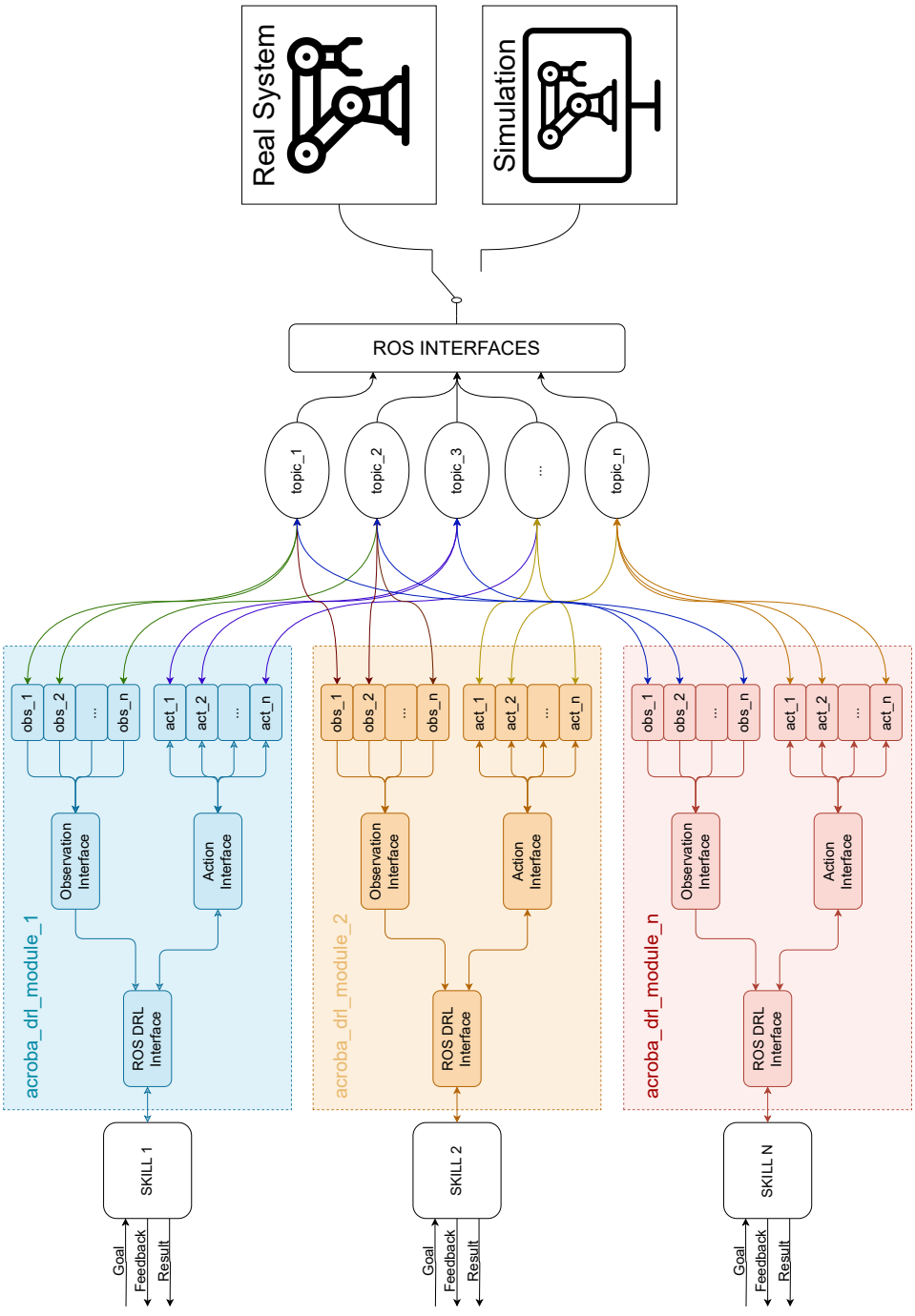

Figura 5.19: Esquema del *pipeline* de transferencia de la política aprendida al robot real mediante el módulo de integración Gymnasium–ROS. Cada política desplagada utilizando la interfaz de DRL se empaqueta en un *Action server* como una *skill*.

5.6.1 Arquitectura general y objetivos de diseño

El módulo proporciona:

- **Acoplamiento automático con ROS**: Una interfaz en Python que envuelve la política y la presenta como un entorno estilo Gymnasium (Towers et al., 2024), desacoplando la lógica de DRL de los detalles de ROS.
- **Mecanismo de mapeo declarativo**: Un archivo de configuración define el mapeo entre las claves de observación/acción de la política y los *topics*/*services*/*actions* de ROS (nombres de *topic*, tipos de mensaje, tasas de publicación, normalización/escala, límites articulares y marcos de referencia).
- **Ejecución segura**: Validaciones previas de límites, verificación de *frame* y filtros de colisión antes de publicar comandos, además de rutas de cancelación y pausa.
- **Portabilidad**: El uso de interfaces estándar y la configuración declarativa permiten reutilizar el módulo con distintos robots y controladores, simplemente ajustando el archivo de configuración.

5.6.2 Modos de ejecución: tiempo real estricto vs. no estricto

El módulo soporta dos modos operativos, seleccionables por configuración:

1. **Tiempo real estricto (*paso a paso*)**: En sistemas donde el controlador del robot admite ciclos de control en tiempo real, la política se ejecuta en un bucle cerrado, generando una acción por cada lote de observaciones recibidas. Este modo requiere que la política responda dentro de la ventana temporal disponible entre lecturas, garantizando que cada acción se base en la observación más reciente. Es adecuado para aplicaciones donde la latencia y la coherencia temporal son críticas. Como ejemplo, para el caso de la política entrenada en este trabajo:
   (a) La política recibe la observación actual del estado articular y la pose objetivo.

(b) Calcula el incremento articular Δq necesario para reducir el error pose–objetivo junto con la parametrización temporal.

(c) Publica Δq en el/los *topic/s* correspondiente del controlador del robot.

(d) Espera a la siguiente observación y repite el ciclo.

2. **No tiempo real (ejecución por lotes)**: La política se ejecuta en simulación local hasta completar la tarea o alcanzar un criterio de parada (por ejemplo, colisión, tiempo máximo, proximidad a la meta). En este modo, la política genera una secuencia completa de acciones basadas en observaciones simuladas, sin requerir respuesta en tiempo real. Para la política desarrollada, se envía la trayectoria completa al controlador del robot para su ejecución. Este modo es adecuado cuando el controlador no garantiza ciclos de control en tiempo real o cuando la tarea permite cierta latencia.

   Otra de las principales ventajas de este enfoque es la posibilidad de realizar una optimización offline de la trayectoria, utilizando técnicas de planificación y control más sofisticadas que las que podrían implementarse en tiempo real. Además, durante la generación de la trayectoria, la política no se ve influenciada por ruidos o latencias en las observaciones, lo que puede mejorar la calidad de la solución final.

5.6.3 Sincronización: esquemas síncronos y asíncronos

En configuraciones de tiempo real, se contemplan dos esquemas de sincronización:

- **Síncrono**: El ciclo `obs` → `act` → `obs` es estricto. La siguiente acción no se genera hasta recibir y validar la nueva observación. Este esquema maximiza la coherencia entre estado estimado y decisión de la política.

  Para esquemas en los que es necesario enviar consignas de configuración articular de manera individual, para evitar que los actuadores paren entre cada consigna, en este modo se ha diseñado e implementado un mecanismo de superposición prematura. Este mecanismo permite enviar la

siguiente consigna antes de que el robot haya alcanzado la anterior, asegurando que el controlador del robot siempre tenga una consigna válida y evitando paradas indeseadas entre consignas respetando las interfaces estándar de ROS.

La Figura 5.20 ilustra este mecanismo. En la gráfica de la izquierda, con una superposición del 90 %, se observa un perfil de velocidad articular continuo, sin paradas entre consignas, pero con un leve aumento de la velocidad. En contraste, la gráfica de la derecha muestra el comportamiento sin superposición, donde el robot se detiene entre cada consigna, lo que puede ser indeseable en aplicaciones que requieren movimientos fluidos y continuos.

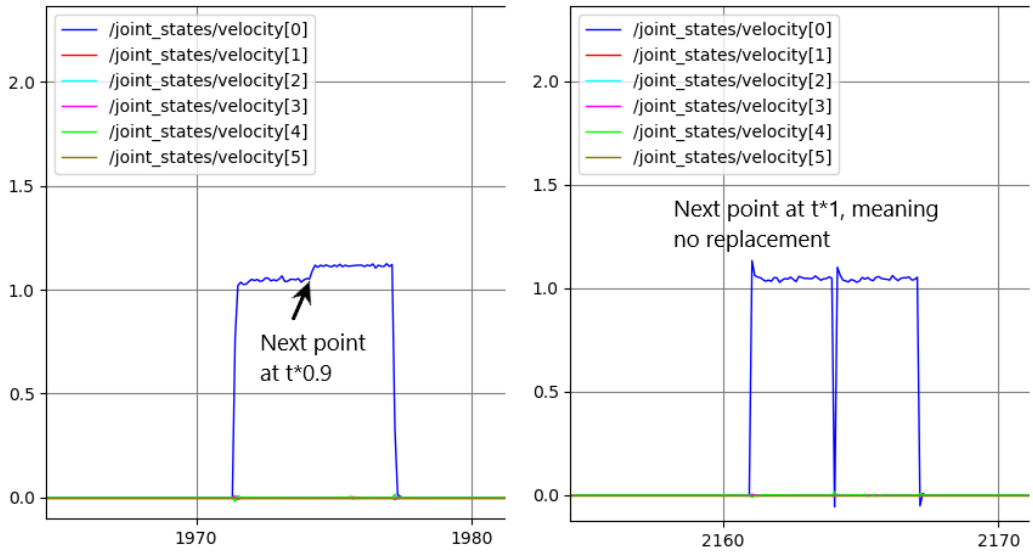

Figura 5.20: Mecanismo de superposición prematura en modo síncrono. La gráfica de la izquierda muestra el comportamiento con una superposición del 90 %, evitando paradas entre consignas. La gráfica de la derecha ilustra el comportamiento sin superposición, donde el robot se detiene entre cada consigna. Ambas gráficas representan la evolución de la velocidad articular en función del tiempo en rad/s.

- **Asíncrono**: La política opera con la última observación disponible, sin bloquear la generación de nuevas acciones si las lecturas llegan con latencia variable.

5.6.4 Configuración declarativa y *wrapper* Gymnasium–ROS

Todo el comportamiento del módulo se define en un archivo de configuración legible que especifica:

- **Observaciones**: Fuentes en ROS, frecuencia de muestreo, normalización y *clipping*.
- **Acciones**: Destinos en ROS (*topics* o *acciones*), saturación y límites.
- **Temporalidad**: Tasas de control, sincronización (`sync`/`async`), ventanas de espera y políticas de reintento.

La interfaz en Python expone los métodos presentes en *Gymnasium*, mapeando internamente las llamadas a publicación/suscripción de ROS, de forma que cualquier agente compatible con *Gymnasium* pueda operar sin cambios.

5.6.5 Ejecución como *Action Server* de ROS

El módulo ofrece una envoltura como *Action Server* de ROS para ejecuciones asíncronas de alto nivel:

- Una petición de *goal* lanza la política exactamente como la define el desarrollador (modo tiempo real o por lotes).
- El servidor publica *feedback* periódico (progreso, distancias a meta, estado de colisión) y finaliza con *result* y *status* estándar.
- Se soportan pausa, reanudar y cancelar según la API nativa de ROS, respetando las semánticas de seguridad del controlador.

Este patrón facilita la integración en arquitecturas jerárquicas (por ejemplo, planificadores de alto nivel que componen metas intermedias), manteniendo la política como bloque base reutilizable.

5.6.6 Trayectorias completas mediante perfilado temporal

Cuando la política opera en modo por lotes, la salida es una trayectoria geométrica en el espacio articular que se parametriza temporalmente mediante TOPPRA antes de su envío al controlador de seguimiento de trayectorias. Esto garantiza que los perfiles de velocidad y aceleración respeten las limitaciones físicas del manipulador y que el controlador disponga de toda la ruta temporalmente consistente.

La Figura 5.21 muestra una trayectoria generada por la política en el espacio articular, representada como una serie de puntos (configuraciones articulares). Estos puntos son el resultado de la política de nuestra solución basada en DRL, que ha aprendido a generar movimientos eficientes y seguros para alcanzar la pose objetivo. Esta trayectoria geométrica se somete a un proceso de parametrización temporal utilizando TOPPRA, que ajusta la velocidad y aceleración a lo largo de la trayectoria para cumplir con las restricciones del robot. Se observa cómo el agente ha decidido diferentes incrementos articulares en cada paso, permitiendo avanzar grandes distancias en situaciones favorables (zonas libres de obstáculos) y movimientos más pequeños en zonas críticas (cercanas a obstáculos o autocolisiones).

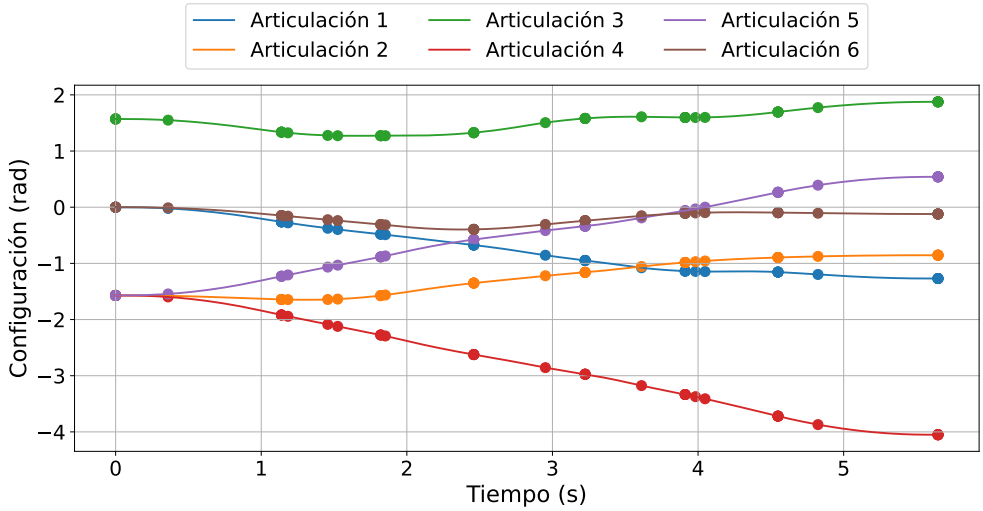

Figura 5.21: Parametrización temporal de la trayectoria generada por la política mediante TOPPRA.

La Figura 5.22 presenta el perfil de velocidad articular resultante tras aplicar TOPPRA a la trayectoria generada por la política. Este perfilado se ha calculado para un 10 % de la velocidad nominal del robot (2π rad/s), lo que implica que la velocidad máxima permitida en cualquier articulación es de 0.2π rad/s. Se observa que los perfiles cumplen estas restricciones de velocidad del robot, intentando maximizar la velocidad para reducir el tiempo de ejecución, pero sin sobrepasar los límites establecidos. Esto es crucial para garantizar una ejecución segura y eficiente de la trayectoria en el robot real.

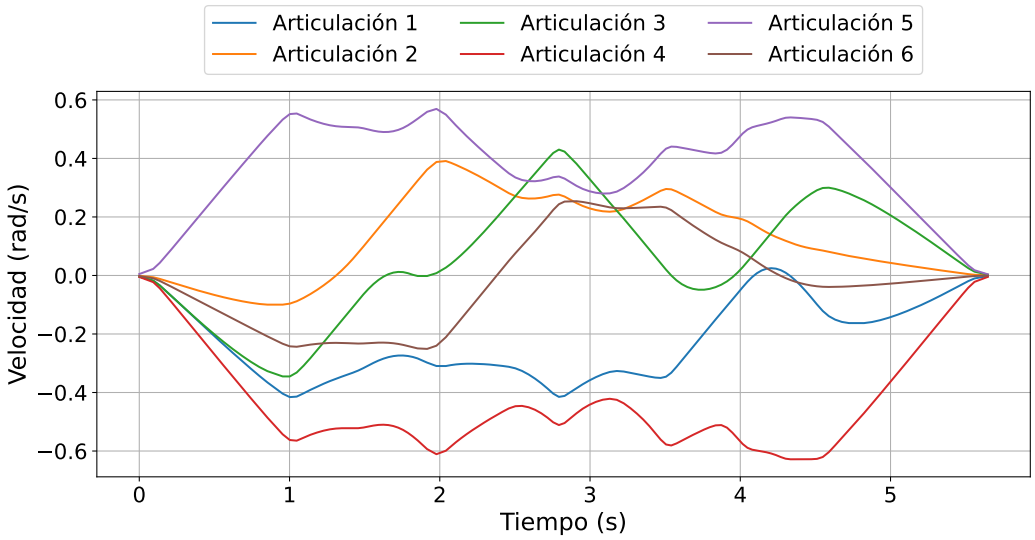

Figura 5.22: Perfil de velocidad articular resultante tras aplicar TOPPRA a la trayectoria generada por la política. Se observa que los perfiles cumplen con las restricciones de velocidad y aceleración del robot, asegurando una ejecución segura y eficiente.

La Figura 5.23 muestra el perfil de aceleración articular resultante tras aplicar TOPPRA a la misma trayectoria. Al igual que con la velocidad, los perfiles de aceleración cumplen con las restricciones del robot, lo que es crucial para evitar movimientos bruscos y garantizar la seguridad durante la ejecución. La aceleración máxima permitida en cualquier articulación es el 10 % de la aceleración nominal del robot (2π rad/s^2), de la misma manera que para el perfil de velocidad. Se observa que los perfiles generados por TOPPRA respetan este límite,

asegurando una transición suave entre las diferentes velocidades a lo largo de la trayectoria.

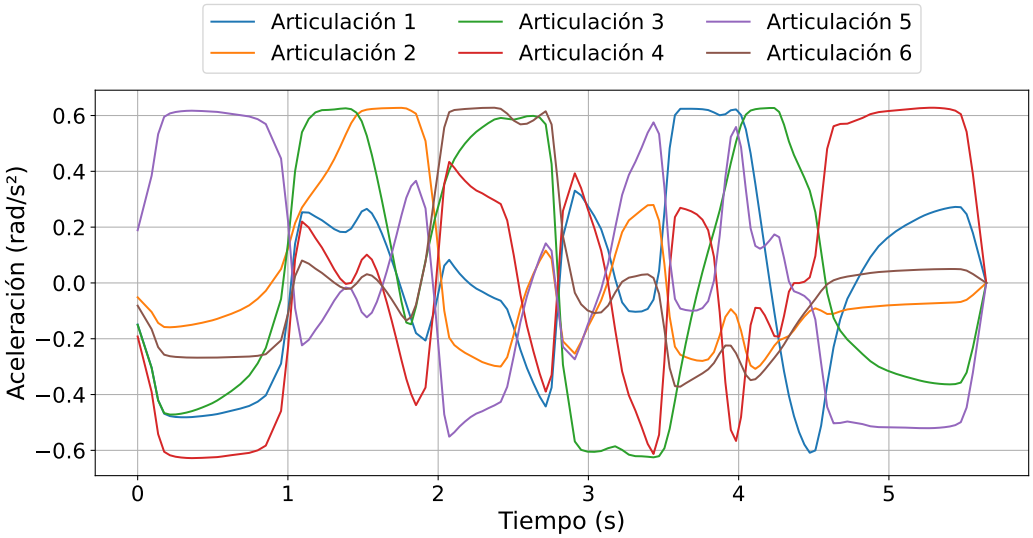

Figura 5.23: Perfil de aceleración articular resultante tras aplicar TOPPRA a la trayectoria generada por la política. Los perfiles de aceleración también cumplen con las restricciones del robot, lo que es crucial para evitar movimientos bruscos y garantizar la seguridad durante la ejecución.

5.6.7 Consideraciones prácticas para la transferencia

Para reducir la brecha *sim-to-real* se aplican las siguientes medidas:

- **Coherencia geométrica**: Uso de URDF calibrado del robot real, consistente con el empleado en simulación.
- **Determinismo y registro**: Uso de semillas, registro de *topics* críticos y repetibilidad de escenarios.
- **Validación progresiva**: Pruebas a baja velocidad, verificación de límites, incremento gradual de frecuencias de muestreo y tiempos de holgura.

En conjunto, este módulo permite desplegar la política aprendida en robots reales con un esfuerzo mínimo de ingeniería adicional, conservando la modu-

laridad del diseño: el mismo agente DRL que opera en simulación puede ejecutarse sobre ROS cambiando únicamente el archivo de configuración y el modo temporal (estricto/no estricto; síncrono/asíncrono), manteniendo las garantías de seguridad y compatibilidad con controladores industriales.

La simulación es a menudo más instructiva que la realidad.

Alexander Pope

CAPÍTULO

6

Evaluación y validación del enfoque propuesto

Contenido del Capítulo

ESTE CAPÍTULO presenta la evaluación comparativa y la validación final del enfoque propuesto frente a los planificadores clásicos de muestreo integrados en MoveIt/OMPL. Tras haber definido, entrenado y desplegado una política determinista basada en DRL en el Capítulo 5, aquí se contrasta su rendimiento con una batería amplia de planificadores estándar, bajo un protocolo experimental controlado y reproducible. El objetivo es doble: Cuantificar, con métricas homogéneas, el valor añadido del enfoque de DRL en términos de eficiencia temporal, tasa de éxito y calidad geométrica del movimiento, y analizar su comportamiento bajo condiciones reales de ejecución en un manipulador industrial, cerrando así el ciclo *sim-to-real*.

Para garantizar una comparación justa, todas las pruebas se realizan sobre el mismo conjunto de pares inicio–meta estratificadas por distancia, con idéntico modelo geométrico del robot y del entorno, mismas verificaciones de autocolisión y contacto con el suelo, y la misma etapa de perfilado temporal mediante TOPPRA. En consecuencia, tanto la política de DRL como los algoritmos de OMPL producen una salida comparable (trayectoria geométrica en C), que posteriormente se parametriza temporalmente con los mismos límites articulares. Esta estandarización permite evaluar, con una base común, métricas geométricas (longitud en C, número de puntos), cinemáticas (suavidad) y temporales (tiempo de planificación e incluso duración de ejecución tras la parametrización).

El capítulo enfatiza también propiedades no triviales para aplicaciones industriales, como el determinismo y la previsibilidad del tiempo de planificación. Mientras que los métodos por muestreo son probabilísticamente completos y, en algunos casos, ofrecen optimización asintótica, su uso práctico bajo límites temporales introduce incertidumbre en el resultado. Por el contrario, nuestra política basada en DRL, una vez entrenada, presenta inferencias deterministas y de latencia acotada, rasgo especialmente valioso en células robóticas donde la repetibilidad y la sincronización con otros subsistemas son requisitos clave.

Además de la comparación sistemática en simulación, se incluye una validación en el robot real mediante el módulo de integración desarrollado y descrito en el Capítulo 5. Se evalúan dos modos de ejecución complementarios: el modo por lotes, que envía de una sola vez la trayectoria perfilada, y el modo de tiempo

real estricto, en el que las acciones se envían paso a paso respetando las restricciones temporales del controlador. Las trayectorias ejecutadas físicamente se comparan con sus homólogas simuladas en términos de perfiles articulares y de velocidad, muestreados a 500 Hz, para cuantificar la fidelidad de la transferencia *sim-to-real*.

En conjunto, los resultados de este capítulo permiten responder con datos a las preguntas clave: ¿En qué condiciones el enfoque basado en DRL supera a los planificadores clásicos?, ¿Cuál es su margen de mejora en métricas de calidad de movimiento?, ¿Cómo se comporta en tiempo real en el robot?, y ¿Qué implicaciones prácticas tiene su determinismo para la integración en *pipelines* industriales?

La estructura del capítulo es la siguiente. En la Sección 6.1 se describe el protocolo experimental y las métricas de evaluación, y se detallan los modelos base y su configuración (con el conjunto de planificadores de OMPL y la política final entrenada). La Sección 6.2 presenta la validación en el robot físico, comparando perfiles ejecutados y simulados en los distintos modos de operación del módulo desarrollado. Finalmente, en la Sección 6.3 se discuten los resultados, destacando las ventajas del enfoque propuesto en eficiencia temporal, compacidad y suavidad de las trayectorias, tasa de éxito y determinismo, y se integran las conclusiones *sim-to-real* que sustentan la aplicabilidad industrial del método.

6.1 Comparación con los planificadores clásicos

6.1.1 Protocolo experimental

Para garantizar una comparación equitativa entre nuestra solución basada en DRL y los planificadores clásicos (OMPL/MoveIt), se sigue el siguiente protocolo:

- **Conjunto de datos:** Se utiliza el conjunto de prueba descrito en la Sección 4.5 (pares inicio–meta estratificados por distancia en el espacio de trabajo y validados cinemáticamente).
- **Condiciones de colisión:** Mismo modelo geométrico del robot y del entorno, mismas comprobaciones de autocolisión y contacto con el suelo.

- **Salida comparable:** Nuestra solución devuelve una trayectoria geométrica en C; los planificadores clásicos devuelven un camino en C. En ambos casos, se aplica TOPPRA para la parametrización temporal, evaluando métricas geométricas y temporizadas.

- **Límites de recursos:** Tiempo de planificación por consulta para OMPL (5 s y hasta 10 intentos); para nuestra política, tiempo de inferencia paso a paso con un máximo de 200 pasos por episodio.

- **Semillas y determinismo:** Se fijan semillas de simulación e inferencia para reproducibilidad. Los planificadores aleatorios usan un conjunto fijo de semillas por consulta.

6.1.2 Métricas de evaluación

Se reportan métricas geométricas, temporales y de calidad del movimiento (Moll et al., 2015):

- **Tasa de éxito** (%): Proporción de consultas con trayectoria libre de colisiones que conectan q_{start} con q_{goal} (criterios de precisión idénticos a los usados en entrenamiento).

- **Longitud en** C (rad): La longitud en el espacio de configuración se calcula como: $L_C = \sum_k \| q_{k+1} - q_k \|_2$.

- **Suavidad en el espacio de configuración**: La suavidad de las trayectorias en el espacio articular se calcula empleando la misma formulación cuadrática utilizada en TOPPRA. En este contexto, la trayectoria se representa como $q(t) \in \mathbb{R}^n$, donde n es el número de articulaciones. La métrica de suavidad S se define como la integral del cuadrado de la norma de la aceleración articular a lo largo del tiempo de ejecución,

$$S = \int_{t_0}^{t_f} \| \ddot{q}(t) \|_2^2 \, dt, \tag{6.1}$$

donde $\|\cdot\|_2$ denota la norma euclídea sobre todas las articulaciones. Esta métrica penaliza cambios rápidos en la velocidad, favoreciendo trayectorias con menores magnitudes de aceleración a lo largo del tiempo. En forma discreta, con aceleraciones muestreadas $\ddot{q}[k]$ en instantes t_k, la integral se aproxima mediante la regla del trapecio como

$$S \approx \sum_{k=1}^{N} \|\ddot{q}[k]\|_2^2 \Delta t_k, \tag{6.2}$$

donde $\Delta t_k = t_k - t_{k-1}$. Esta formulación está alineada directamente con las funciones de coste cuadráticas convexas soportadas por TOPPRA, permitiendo optimizar no solo el tiempo de recorrido sino también la suavidad del movimiento (Pham and Pham, 2018).

- **Tiempo de planificación** (s): Se incluye en el tiempo de planificación, por una parte, en el caso de nuestra solución basada en DRL, el tiempo que se tarde en realizar la inferencia completa de la trayectoria cinemática y el tiempo de parametrización temporal. Por otra parte, en el caso de los algoritmos tradicionales se incluye el tiempo de planificación del algoritmo, el tiempo de post-procesado y el tiempo de parametrización temporal. En ambos casos, se considera el tiempo total desde la consulta hasta la obtención de la trayectoria ejecutable.

- **Tiempo de ejecución** (s): Una vez realizada la parametrización temporal, el tiempo que se tarda en ejecutar la trayectoria en el robot.

Para cada métrica se incluye una tabla resumen que contiene la media junto con la desviación estándar para cada método. Además, se incluyen diagramas de cajas para los métodos más representativos, que permiten visualizar la distribución completa de los datos obtenidos.

6.1.3 Modelo base y configuración

Se incluyen todos los planificadores basados en muestreo incluidos en el *framework* MoveIt, a través de su *pipeline* de planificación con OMPL. Los planificadores considerados son: RRT, RRT*, RRT-Connect, KPIECE, PDST, Transition-

based Rapidly-exploring Random Tree (TRRT), PRM, EST, Bidirectional Transition-based Rapidly-exploring Random Tree (BiTRRT), Bidirectional Expansive Space Trees (BiEST), BIT*, Bidirectional KPIECE (BKPIECE), LAZY PRM*, LAZY BKPIECE, PRM*, Projection-based EST (PROJEST), SBL y STRIDE. Se emplean los hiperparámetros por defecto de MoveIt para cada planificador. El objetivo es evaluar el rendimiento de los planificadores en su configuración estándar, tal como se utilizarían en la práctica para aplicaciones generalizadas.

En el caso de la política, se utiliza el modelo final descrito en la Sección 5.5.1. Para todos los métodos se utiliza el subconjunto de test del conjunto de datos expuesto en la Sección 4.5. Se toma como partida la configuración inicial de la trayectoria y se ejecuta cada uno de los algoritmos, obteniendo como resultado una trayectoria geométrica en C. A continuación, se aplica TOPPRA para obtener la parametrización temporal y se evalúan las métricas descritas en la Sección 6.1.2.

6.1.4 **Resultados y discusión**

A continuación se presentan los resultados obtenidos en la comparación entre la política basada en DRL y los planificadores clásicos de OMPL.

La Tabla 6.1 resume las métricas clave obtenidas para cada método, incluyendo tiempo de planificación, número de puntos de la trayectoria geométrica, tasa de éxito y tiempo máximo de planificación. Los valores promedio se acompañan de sus desviaciones estándar para reflejar la variabilidad entre pruebas. Se puede observar como la política basada en DRL supera de manera consistente a los planificadores clásicos en todas las métricas evaluadas. La tasa de éxito del 94.12 % obtenida por DRL es notablemente superior al 92.97 % del mejor planificador clásico (PRM). El tiempo de planificación promedio de 0.01322 segundos es significativamente menor que los 0.03360 segundos del PRM, representando una mejora sustancial en eficiencia temporal. Además, la trayectoria generada por nuestra solución es más corta y suave, con un promedio de 7.20 puntos y una métrica de suavidad de 1.2286, en comparación con los 18.61 puntos y 1.2987 del PRM. El máximo tiempo de planificación registrado para la

solución propuesta fue de 0.023 segundos, muy inferior al tiempo máximo establecido para los algoritmos tradicionales de 5 s.

Tabla 6.1: Resumen de las métricas de rendimiento promedio para DRL y los planificadores de OMPL. La tabla destaca las métricas clave, incluyendo tiempo de planificación, número de puntos de la trayectoria geométrica, tasa de éxito y tiempo máximo de planificación. Para el tiempo de planificación y el número de puntos de la trayectoria geométrica, se muestran los valores promedio junto con sus desviaciones estándar para reflejar la variabilidad entre pruebas.

Método	Tiempo de planificación (s)	Número de puntos de la trayectoria	Tasa de éxito (%)	Máximo tiempo de planificación (s)	Suavidad
DRL	**0.01322 ± 0.003**	**7.20 ± 4.25**	**94.12**	**0.023**	**1.2286 ± 0.4285**
BITSTAR	0.03349 ± 0.190	18.46 ± 10.54	92.91	5.028	1.2869 ± 0.5108
PRM	0.03360 ± 0.182	18.61 ± 11.09	92.97	5.039	1.2987 ± 0.5149
RRTConnect	0.03542 ± 0.203	18.51 ± 10.61	92.86	5.048	1.2874 ± 0.4983
PROJEST	0.03665 ± 0.067	19.01 ± 12.07	92.60	5.016	1.2941 ± 0.5124
EST	0.03992 ± 0.065	18.87 ± 11.92	92.86	5.013	1.2961 ± 0.5141
BIEST	0.04051 ± 0.185	19.78 ± 12.45	91.36	5.042	1.3177 ± 0.5372
BITRRT	0.04057 ± 0.278	19.29 ± 13.34	91.68	5.043	1.3189 ± 0.5353
RRT	0.04578 ± 0.236	18.56 ± 10.83	92.92	5.052	1.2999 ± 0.5251
KPIECE	0.07106 ± 0.220	20.56 ± 13.03	90.59	5.233	1.3478 ± 0.5485
STRIDE	0.07476 ± 0.176	20.75 ± 13.75	90.58	5.054	1.3475 ± 0.5691
PDST	0.11133 ± 0.339	20.32 ± 12.33	90.49	5.067	1.3372 ± 0.5289
TRRT	0.28084 ± 0.963	20.33 ± 12.04	90.62	5.085	1.3572 ± 0.5793

Método	Tiempo de planificación (s)	Número de puntos de la trayectoria	Tasa de éxito (%)	Máximo tiempo de planificación (s)	Suavidad
BKPIECE	0.38242 ± 0.265	18.79 ± 11.65	92.74	5.046	1.2923 ± 0.5306
SBL	0.52241 ± 0.296	18.85 ± 12.04	92.95	5.055	1.3016 ± 0.5466
LAZY BKPIECE	1.44631 ± 0.534	20.87 ± 13.79	90.47	5.121	1.3627 ± 0.6179
LAZYPRM*	5.00021 ± 0.002	18.37 ± 11.65	92.46	5.074	1.2799 ± 0.4776
RRT*	5.003 ± 0.006	18.08 ± 9.38	92.99	5.008	1.2713 ± 0.4597
PRM*	5.00617 ± 0.005	18.14 ± 9.67	92.94	5.954	1.2880 ± 0.4979

La Figura 6.1 muestra un diagrama de cajas comparativo del tiempo de planificación para los métodos más representativos. En estos diagramas sólo se han incluido los métodos tradicionales con mejores resultados. Éstos métodos son: RRT, RRT-Connect, KPIECE, PDST, TRRT, PRM, EST y BiEST.

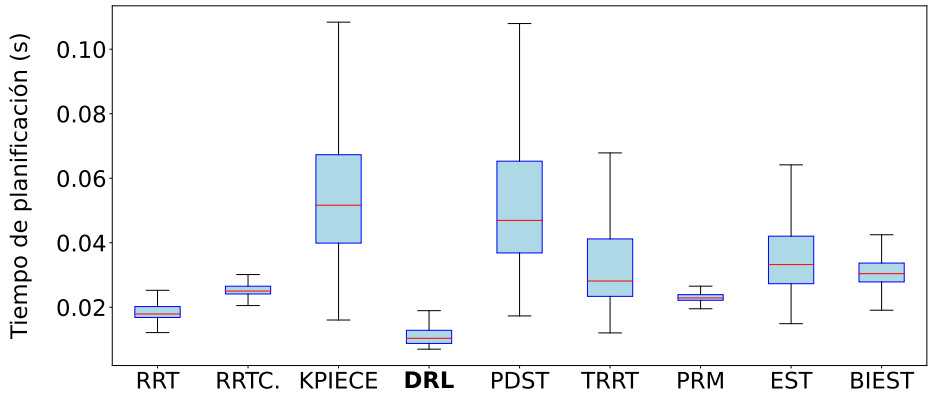

Figura 6.1: Diagrama de cajas comparativo del tiempo de planificación para los métodos más representativos.

Se observa que existen diferencias notables en la dispersión de los resultados: algunos planificadores, como KPIECE y PDST, muestran una mayor variabi-

lidad y valores máximos más elevados, mientras que otros, como RRT, RRT-Connect, PRM y el propio DRL, presentan distribuciones más compactas. En cuanto a la tendencia central, la mediana de nuestra solución basada en DRL se sitúa entre las más bajas del conjunto, alrededor de 0.01–0.02 s, lo que refleja una planificación más rápida en promedio. Por el contrario, KPIECE y PDST alcanzan medianas cercanas a 0.04–0.05 s, situándose entre los métodos más lentos. En conjunto, el gráfico evidencia que, tanto nuestra política como RRT destacan por tiempos de planificación reducidos y baja dispersión, mientras que otros planificadores exhiben mayor variabilidad en su desempeño.

La Figura 6.2 muestra el número de puntos en las trayectorias generadas por los distintos planificadores evaluados. Se observa que la mayoría de los métodos, como RRT, RRT-Connect, KPIECE, PDST, TRRT, PRM, EST y BiEST, producen trayectorias con medianas comprendidas entre 15 y 20 puntos, acompañadas de una amplia variabilidad en sus distribuciones. En contraste, el enfoque basado en DRL genera trayectorias considerablemente más cortas, con una mediana cercana a 7 puntos y una dispersión reducida en comparación con el resto de planificadores. Este resultado indica que las trayectorias obtenidas mediante la solución propuesta tienden a ser más compactas, mientras que los planificadores clásicos presentan una mayor heterogeneidad en el número de puntos que conforman sus soluciones.

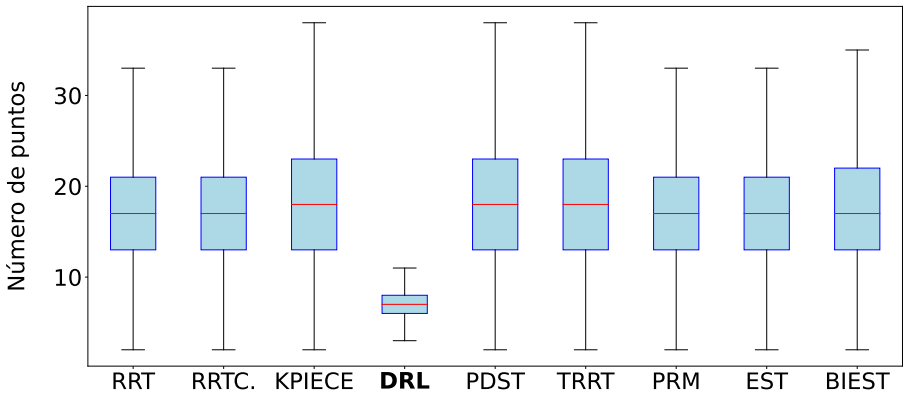

Figura 6.2: Diagrama de cajas comparativo del número de pasos del camino para los métodos más representativos.

La Figura 6.3 muestra la longitud de las trayectorias en el espacio de configuración, expresada en radianes, para los diferentes planificadores evaluados. En general, se observa que la mayoría de los métodos presentan medianas cercanas a 3 rad, con distribuciones relativamente similares entre sí. El enfoque basado en DRL exhibe una mediana ligeramente inferior, en torno a 2.5 rad, además de una dispersión más reducida respecto a los planificadores clásicos. Por el contrario, métodos como KPIECE, PDST y TRRT alcanzan valores máximos más elevados, lo que refleja una mayor variabilidad en la longitud de las trayectorias generadas. En conjunto, el gráfico indica que la política propuesta tiende a producir trayectorias más compactas y consistentes, mientras que algunos planificadores de muestreo generan soluciones más heterogéneas.

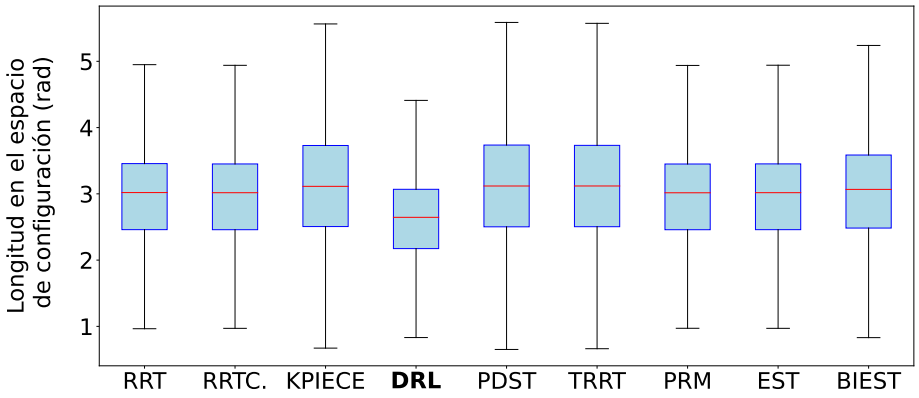

Figura 6.3: Diagrama de cajas comparativo de la longitud del espacio de configuración para los métodos más representativos.

La Figura 6.4 presenta el tiempo de ejecución de las trayectorias generadas por los distintos planificadores evaluados. Esta métrica se obtiene de la parametrización temporal aplicada utilizando TOPPRA. sEn términos generales, todos los métodos muestran medianas comprendidas entre 5 y 6 segundos, con distribuciones similares y una dispersión moderada. El enfoque basado en DRL se sitúa en valores comparables a los planificadores clásicos, sin diferencias significativas en la tendencia central. No obstante, algunos métodos como KPIECE, PDST y TRRT presentan valores máximos más elevados, lo que indica una mayor

variabilidad en ciertos casos. En conjunto, los resultados sugieren que, en lo relativo al tiempo de ejecución, el rendimiento de nuestra solución es consistente y se mantiene en el mismo rango que el de los planificadores de muestreo.

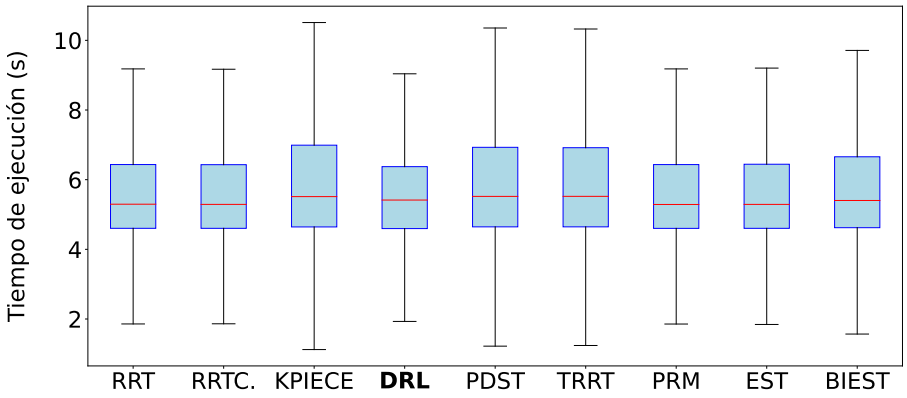

Figura 6.4: Diagrama de cajas comparativo del tiempo de ejecución de la trayectoria para los métodos más representativos.

Por último, la Figura 6.5 muestra el tiempo de ejecución de las trayectorias generadas por los distintos planificadores evaluados, incluyendo los valores atípicos. Es decir, se trata de la misma figura que la anterior, pero con un enfoque en los valores atípicos. Se observa que la mayoría de los métodos tradicionales presentan valores atípicos alrededor de 5 segundos, el tiempo máximo establecido. En contraste, el enfoque basado en DRL mantiene una distribución más consistente y sin valores atípicos extremos, lo que indica una mayor fiabilidad en el tiempo de ejecución de las trayectorias generadas. En conjunto, el gráfico sugiere que, el enfoque basado en DRL destaca por su estabilidad y ausencia de casos extremos.

6.2 Validación en el robot real

El módulo se valida con un robot Universal Robots UR3e equipado con una pinza OnRobot RG2. La política entrenada en simulación se despliega en el robot real utilizando el modo por lotes, enviando la trayectoria completa al controlador del robot con el módulo desarrollado. La validación se realiza en un entorno

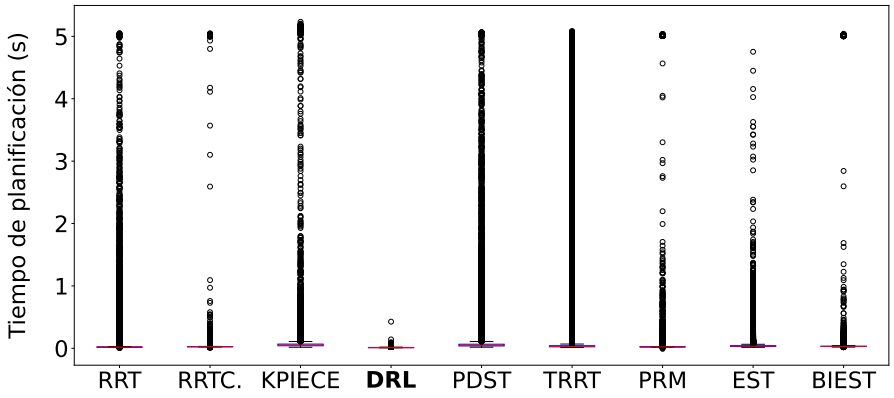

Figura 6.5: Diagrama de cajas comparativo del tiempo de planificación de la trayectoria para los métodos más representativos, incluyendo valores atípicos.

controlado, replicando las condiciones del escenario de simulación. Se ejecutan múltiples trayectorias desde diferentes configuraciones iniciales hacia una pose objetivo fija, comparando las trayectorias ejecutadas por el robot real con las generadas por la política en simulación. Se mide la precisión en el espacio articular y en el perfil de velocidad para cada trayectoria. Se omite la aceleración debido a la falta de sensórica directa en el robot. El muestreo se realiza a 500 Hz tanto en el robot real como en la simulación, asegurando una comparación coherente.

Las Figuras 6.6 y 6.7 muestran los perfiles de configuración y velocidad articular, respectivamente, para cada una de las articulaciones comparando la trayectoria ejecutada por el robot real (línea continua) con la trayectoria computada por la política (línea discontinua). Se observa una alta fidelidad entre ambas trayectorias, con errores imperceptibles a simple vista.

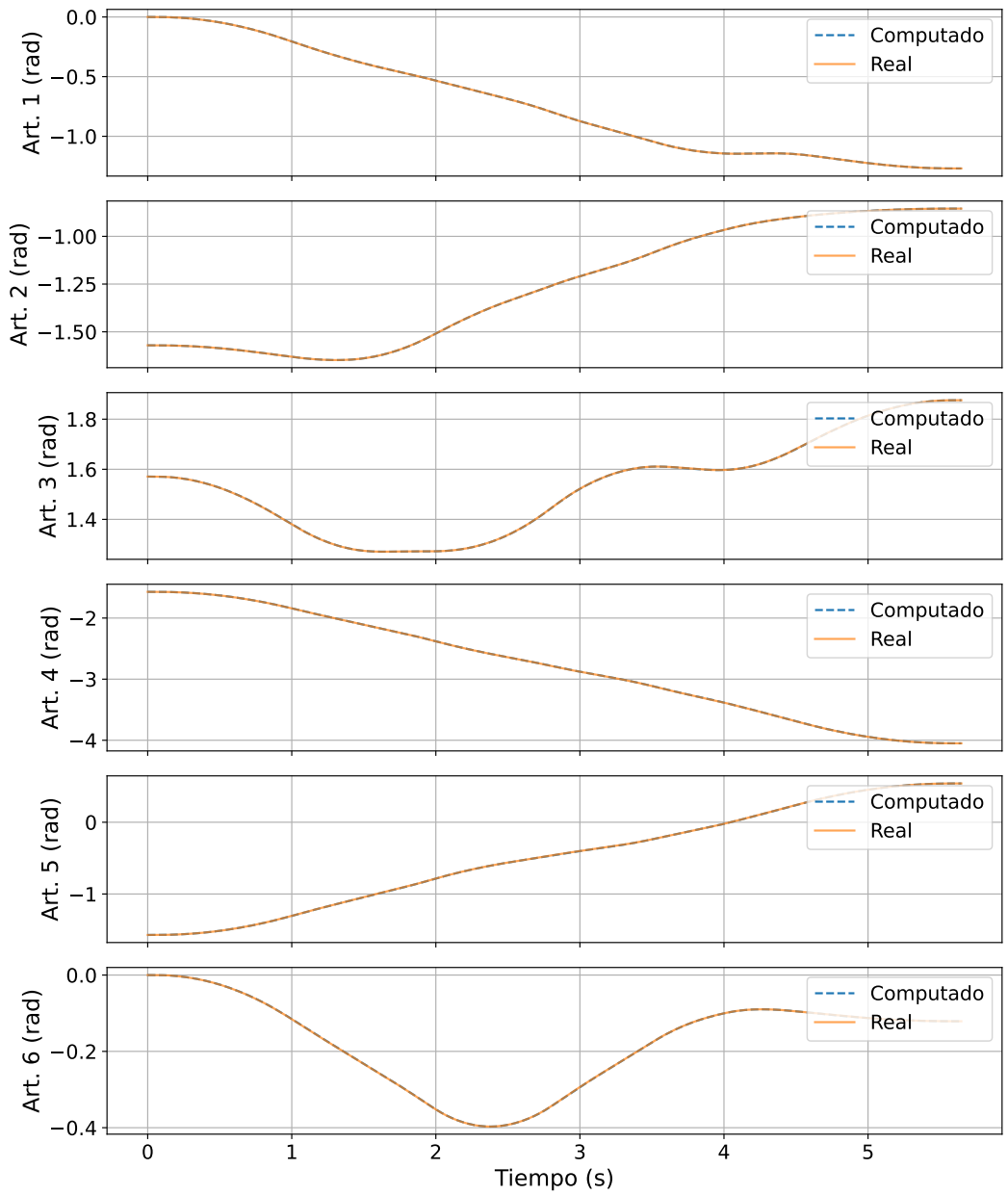

Figura 6.6: Comparativa entre la trayectoria ejecutada por el robot real (línea continua) y la trayectoria computada por la política (línea discontinua) en el espacio articular. Se observa una alta fidelidad entre ambas trayectorias.

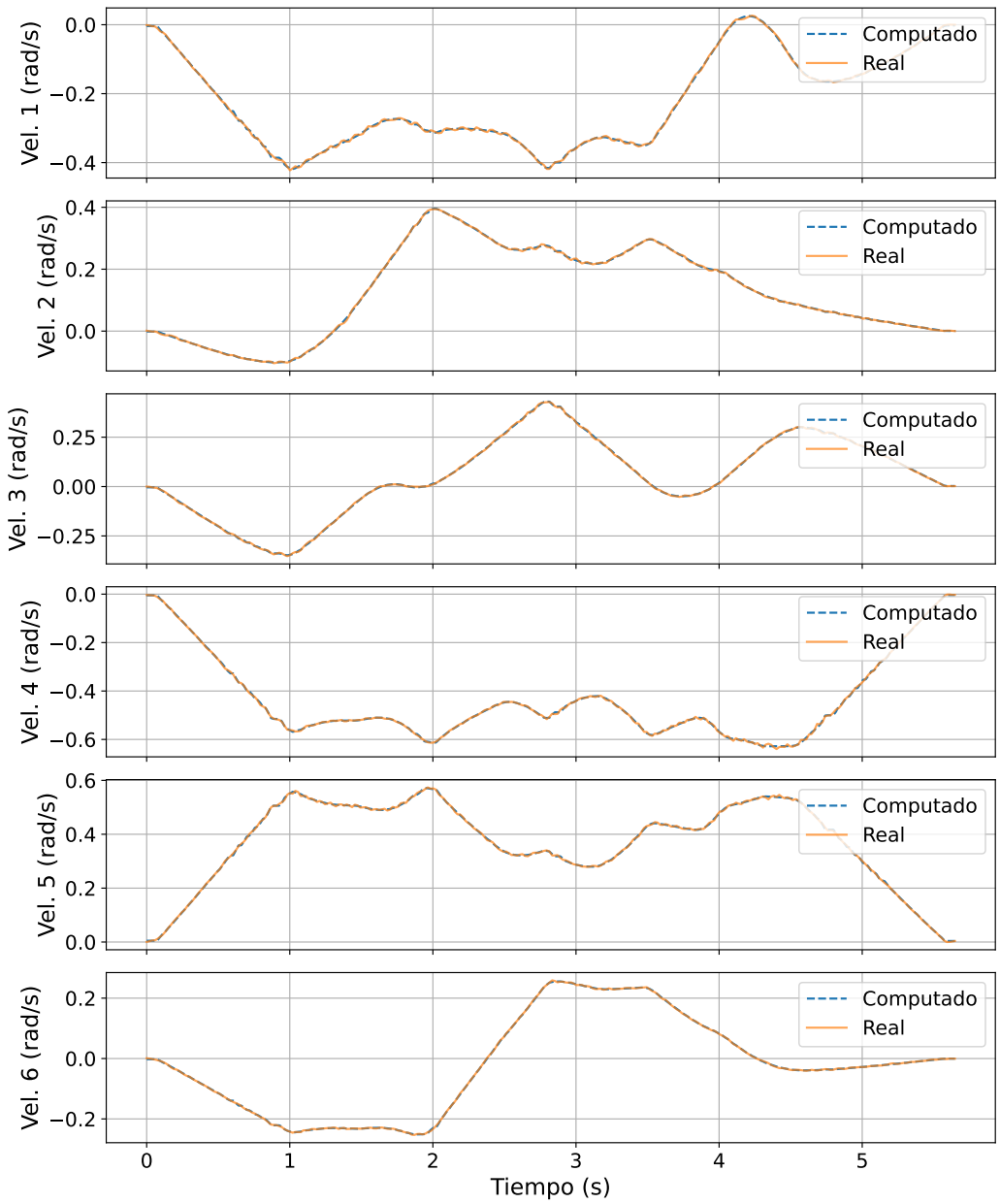

Figura 6.7: Comparativa entre el perfil de velocidad articular ejecutado por el robot real (línea continua) y el perfil computado por la política (línea discontinua). Se observa una alta fidelidad entre ambos perfiles.

El mismo procedimiento se repite para el modo de tiempo real estricto con un 95 % de superposición prematura, obteniendo resultados igualmente satisfactorios, aunque con un ligero aumento del error debido a la naturaleza del control en tiempo real.

Las Figuras 6.8 y 6.9 muestran los perfiles de configuración y velocidad articular, respectivamente, para cada una de las articulaciones en este modo de ejecución. Aunque se observa un ligero aumento del error en comparación con el modo por lotes, la fidelidad entre la trayectoria ejecutada por el robot real y la generada por la política sigue siendo alta para la configuración articular, con errores aún imperceptibles a simple vista.

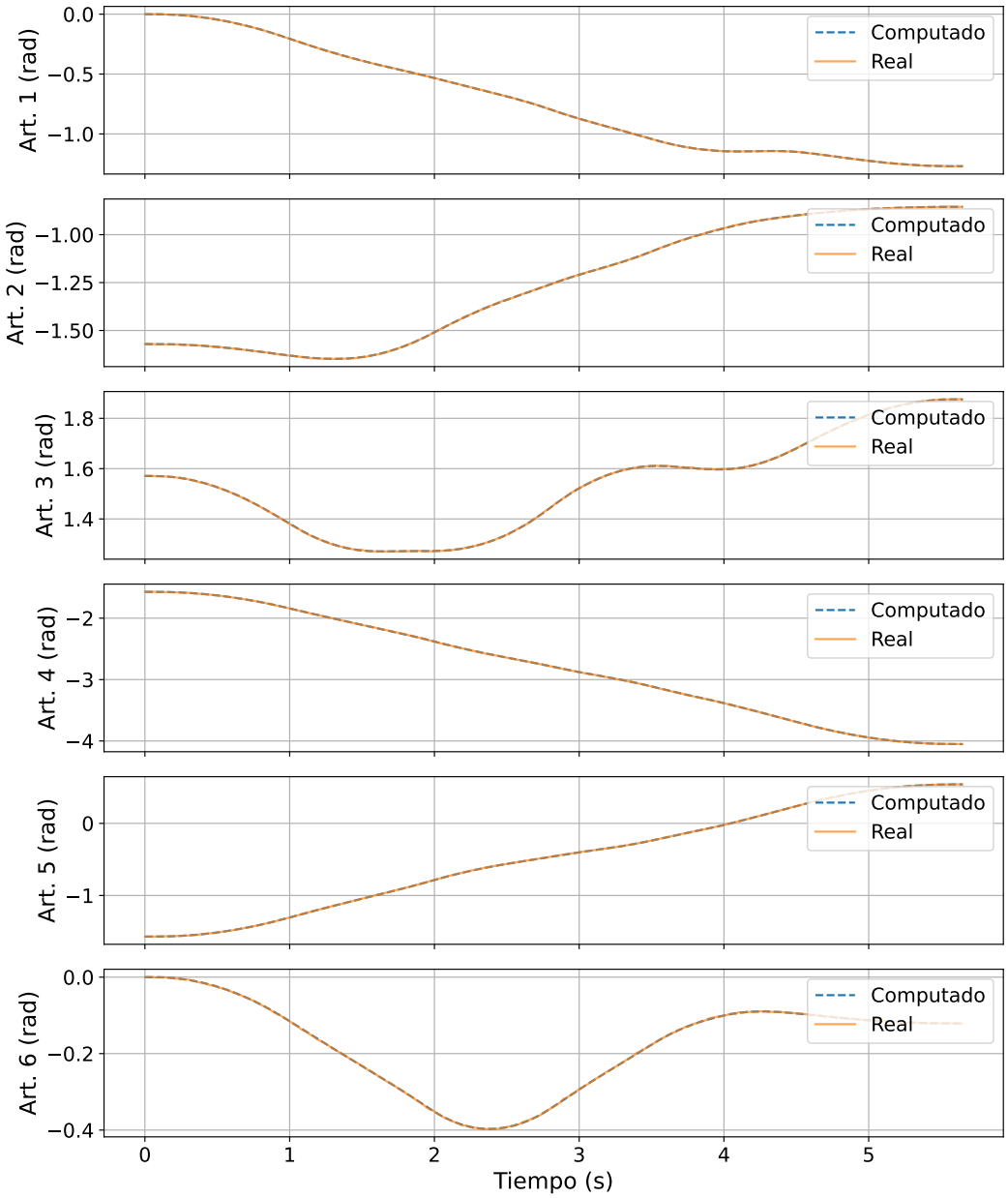

Figura 6.8: Comparativa entre la trayectoria ejecutada por el robot real (línea continua) y la trayectoria computada por la política (línea discontinua) en el espacio articular, en modo de tiempo real estricto con un 95 % de superposición prematura.

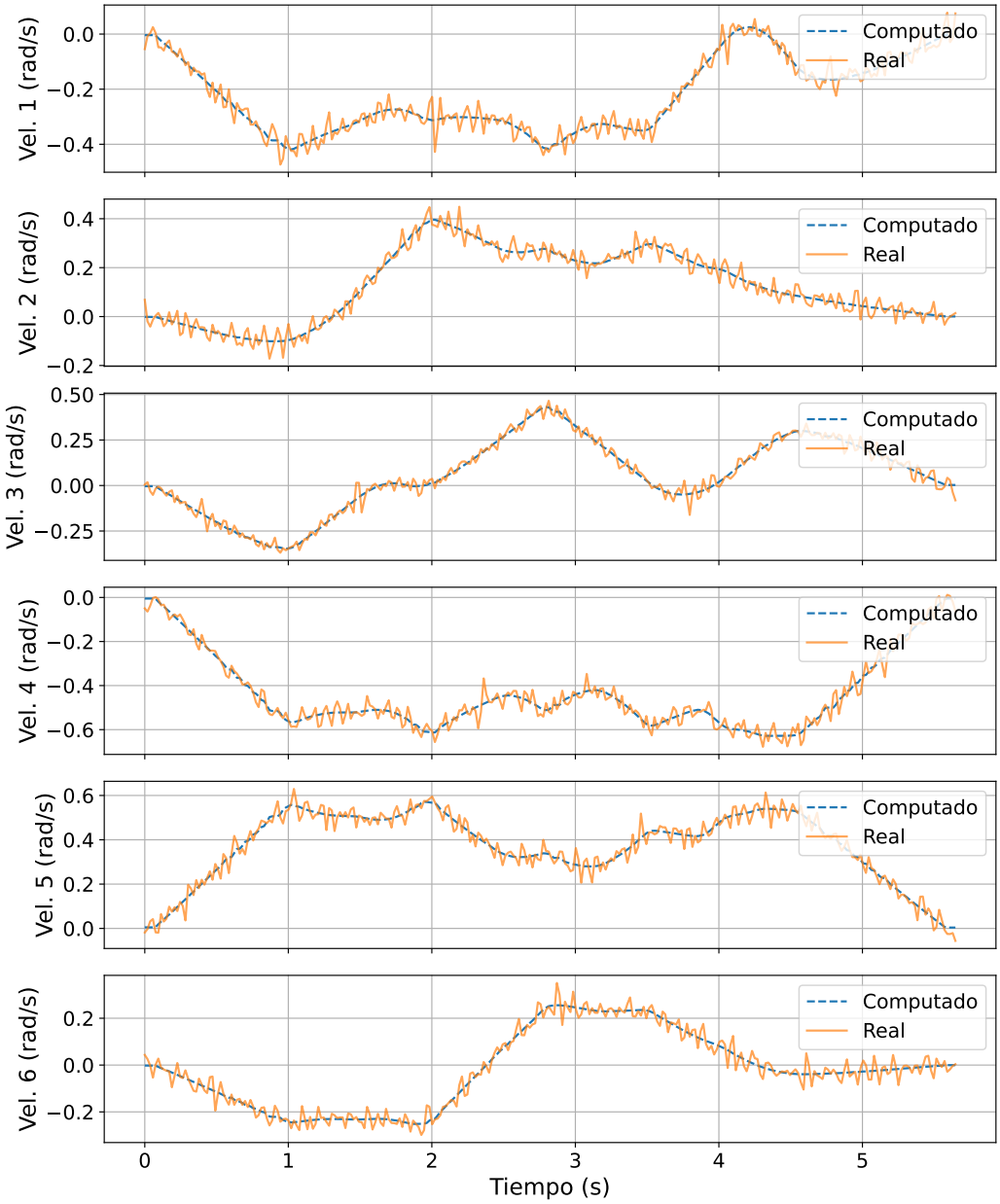

Figura 6.9: Comparativa entre el perfil de velocidad articular ejecutado por el robot real (línea continua) y el perfil computado por la política (línea discontinua), en modo de tiempo real estricto con un 95 % de superposición prematura.

Cada uno de los procesos se repite para 100 trayectorias diferentes, al 10 % de velocidad nominal de los actuadores, obteniendo resultados consistentes en todas las ejecuciones, con una desviación estándar baja en los errores medidos. Las Tablas 6.2 y 6.3 presentan el error absoluto medio y la desviación estándar para cada articulación del robot UR3e en ambos modos de ejecución del módulo desarrollado. En el modo por lotes, el error medio por articulación es extremadamente bajo, con valores en el rango de 0.00018 a 0.00062 rad (0.01°a 0.035°), y desviaciones estándar también muy pequeñas.

En el modo de tiempo real estricto con un 95 % de superposición prematura, el error medio por articulación aumenta ligeramente, situándose entre 0.00063 y 0.00083 rad (0.036°a 0.048°). Aunque este aumento es notable, los errores siguen siendo bajos y aceptables para la mayoría de aplicaciones industriales. La desviación estándar en este modo también es mayor, reflejando la variabilidad introducida por las condiciones de control en tiempo real.

Tabla 6.2: Error absoluto medio de configuración para cada articulación del robot UR3e para cada modo de ejecución del módulo desarrollado. Se incluye la desviación estándar para cada articulación y modo de ejecución.

Articulación	Por lotes (rad)	Tiempo real (95 % superp.) (rad)
J1	0.00033 ± 0.000024	0.00074 ± 0.00023
J2	0.00022 ± 0.000016	0.00069 ± 0.00024
J3	0.00024 ± 0.000022	0.00068 ± 0.00023
J4	0.00062 ± 0.000026	0.00083 ± 0.00025
J5	0.00053 ± 0.000025	0.00080 ± 0.00025
J6	0.00018 ± 0.000011	0.00063 ± 0.00024

En cuanto al perfil de velocidad, el error medio en el modo por lotes es igualmente bajo, con valores entre 0.00114 y 0.00260 rad/s, y desviaciones estándar pequeñas. Sin embargo, en el modo de tiempo real estricto, el error medio por articulación aumenta significativamente, situándose entre 0.02263 y 0.02543 rad/s. Este aumento considerable del error en el perfil de velocidad es comprensible, dado que la política debe operar bajo restricciones temporales estrictas y la superposición prematura puede afectar la precisión del seguimiento de la velocidad. La desviación estándar en este modo también es mayor, indicando una

mayor variabilidad en la ejecución de las velocidades articulares.

Tabla 6.3: Error absoluto medio de velocidad por articulación en el robot UR3e para cada modo de ejecución del módulo desarrollado. Se incluye la desviación estándar para cada articulación y modo de ejecución.

Articulación	Por lotes (rad/s)	Tiempo real (95 % superp.) (rad/s)
J1	0.00238 ± 0.00017	0.02290 ± 0.0071
J2	0.00160 ± 0.00012	0.02362 ± 0.0082
J3	0.00226 ± 0.00017	0.02263 ± 0.0071
J4	0.00260 ± 0.00019	0.02469 ± 0.0082
J5	0.00249 ± 0.00018	0.02543 ± 0.0083
J6	0.00114 ± 0.00008	0.02409 ± 0.0080

6.3 Discusión de resultados

Los resultados obtenidos permiten extraer varias conclusiones relevantes sobre el comportamiento de la política basada en DRL frente a los planificadores clásicos de muestreo.

En primer lugar, destaca la eficiencia temporal de nuestra solución basada en DRL. La política es capaz de generar trayectorias en el orden de milisegundos, con tiempos de planificación más de dos órdenes de magnitud inferiores a los límites temporales impuestos a los planificadores clásicos. Esto tiene un impacto directo en aplicaciones en línea, donde la necesidad de respuestas rápidas es crítica (por ejemplo, en planificación reactiva ante cambios en el entorno o metas móviles). En contraste, los métodos de muestreo dependen de la construcción de grafos o árboles exploratorios que, aunque sean probabilísticamente completos o incluso presenten características de optimización asintótica, requieren tiempos de cálculo mayores y muestran mayor variabilidad en su desempeño.

En segundo lugar, el enfoque basado en DRL genera trayectorias más compactas y suaves. El número de puntos intermedios es significativamente menor, lo que simplifica la parametrización temporal y ejecución. Además, la métrica de suavidad confirma que las trayectorias propuestas por la política tienden a presentar menores aceleraciones cuadráticas acumuladas, lo que se traduce en

movimientos más eficientes desde el punto de vista energético y menos exigentes para los actuadores. Esta característica es especialmente relevante en manipuladores industriales, donde la reducción de picos de esfuerzo mecánico se asocia a un menor desgaste y mayor vida útil del sistema.

Un aspecto fundamental es que la inyección de experiencia experta no ha limitado la capacidad de innovación de la política aprendida. Aunque este mecanismo se utilizó para acelerar las fases iniciales del entrenamiento, evitando periodos prolongados de exploración aleatoria, y mejorar la precisión del agente, los resultados finales muestran mejoras claras frente a los planificadores clásicos en métricas como número de puntos de la trayectoria, suavidad y tasa de éxito. Esto confirma que el agente no ha acabado codificando el comportamiento de los algoritmos tradicionales, sino que ha desarrollado soluciones novedosas que superan consistentemente a las obtenidas mediante muestreo. La experiencia experta, por tanto, ha actuado como un catalizador del aprendizaje sin comprometer la calidad o la originalidad de la política resultante.

Otra ventaja distintiva de nuestra solución basada en DRL es el determinismo total de sus soluciones. Una vez entrenada, la política siempre devuelve la misma trayectoria para un par inicio–meta dado, lo que elimina la variabilidad presente en los algoritmos tradicionales basados en muestreo. Estos últimos, aunque son probabilísticamente completos y asintóticamente óptimos en teoría, deben ejecutarse bajo límites de tiempo máximos en la práctica. Este hecho introduce una fuente importante de incertidumbre en entornos industriales, donde la repetibilidad y la previsibilidad son cualidades altamente valoradas. Además, los experimentos muestran que, si se estableciera como límite temporal el tiempo de planificación alcanzado por la política de DRL, la tasa de éxito de los planificadores clásicos caería por debajo del 50 %. Esto evidencia que la aparente superioridad teórica de los planificadores clásicos pierde relevancia en escenarios con restricciones de tiempo real. Cabe señalar, además, que el tiempo de planificación medido para la política se ha calculado asumiendo un máximo de 200 pasos por episodio. Esta cifra es conservadora y permite cierto margen de exploración, pero podría reducirse significativamente sin impacto apreciable en la tasa de éxito, lo que haría que los tiempos de planificación fueran aún menores y reforzara todavía más la ventaja frente a los planificadores clásicos.

Los resultados obtenidos en el robot real refuerzan estas conclusiones. Tal como se expone en la Sección 6.2, la comparación entre las trayectorias ejecutadas físicamente y las generadas en simulación muestra una fidelidad extremadamente alta en el espacio articular, con errores absolutos medios en el rango de 0.00018 a 0.00083 rad, imperceptibles en la práctica. Incluso en condiciones de control en tiempo real estricto, donde se introduce variabilidad adicional, los errores permanecen en niveles muy reducidos y dentro de los márgenes aceptables para aplicaciones industriales. En cuanto al perfil de velocidad, aunque el modo en tiempo real presenta un aumento del error medio (del orden de 0.02 rad/s), este comportamiento es coherente con las restricciones temporales y de superposición prematura propias de la ejecución física. En conjunto, los resultados confirman que la política entrenada no solo supera a los métodos clásicos en simulación, sino que además mantiene su validez y precisión tras la transferencia al robot real, mitigando de manera efectiva el *sim-to-real gap*.

En cuanto a la tasa de éxito, la política no solo se mantiene en niveles comparables a los mejores planificadores clásicos, sino que los supera de forma consistente. Esto valida la hipótesis de que, con un conjunto de entrenamiento suficientemente diverso y un proceso de aprendizaje bien estructurado (función de recompensa híbrida, *curriculum learning* e inyección de experiencia experta), la política puede generalizar a nuevas consultas sin necesidad de exploración explícita en tiempo de ejecución.

Por último, en lo relativo al tiempo de ejecución de las trayectorias, los resultados muestran que tanto nuestro enfoque basado en DRL como los planificadores clásicos producen trayectorias cuyo perfil temporal se encuentra en un rango similar tras aplicar la parametrización con TOPPRA. Esto confirma que la ventaja de nuestra propuesta reside principalmente en la rapidez y consistencia de la generación de trayectorias geométricas, mientras que el tiempo de ejecución está condicionado por las limitaciones físicas de los actuadores.

En síntesis, los resultados evidencian que la política de DRL no solo es competitiva, sino que supera de forma consistente a los planificadores clásicos en todos los indicadores clave: eficiencia temporal, compacidad y suavidad de las trayectorias, tasa de éxito y determinismo. Esta superioridad práctica refuerza la aplicabilidad del enfoque en contextos industriales, donde las limitaciones de

tiempo real y la necesidad de fiabilidad determinista son requisitos fundamentales. Además, la validación en el robot real confirma que estas ventajas no se restringen al ámbito simulado, sino que se trasladan de forma efectiva al mundo físico, reforzando el potencial de adopción del enfoque en aplicaciones industriales reales.

El futuro pertenece a quienes creen en la belleza de sus sueños.

Eleanor Roosevelt

CAPÍTULO 7

Conclusiones y líneas futuras

Contenido del Capítulo

ESTE CAPÍTULO constituye la culminación de la tesis doctoral, al sintetizar los resultados obtenidos y reflexionar sobre el alcance, la validez y la proyección futura del enfoque propuesto. Tras el recorrido iniciado con el análisis de los métodos clásicos de planificación de trayectorias en el Capítulo 3, continuado con el diseño del entorno experimental en el Capítulo 4, el entrenamiento y validación de políticas en el Capítulo 5, y su evaluación comparativa frente a planificadores tradicionales en el Capítulo 6, este capítulo ofrece una visión integradora que permite consolidar las aportaciones de la investigación y proyectarlas hacia nuevas direcciones.

El objetivo principal es doble: por un lado, sintetizar de manera crítica los hallazgos de la tesis, vinculándolos con la hipótesis y los objetivos planteados inicialmente; y por otro, identificar las contribuciones más relevantes en términos científicos, metodológicos y aplicados, así como las oportunidades que emergen para trabajos futuros. En este sentido, se destacan aspectos clave como la validez empírica de la hipótesis central, la capacidad de la política basada en DRL para superar limitaciones de los métodos clásicos, la aplicabilidad práctica del enfoque en entornos industriales reales y su potencial de extensión a escenarios más generales.

Asimismo, se presenta una revisión estructurada de las contribuciones académicas derivadas de la investigación, incluyendo publicaciones en revistas indexadas y congresos internacionales, que han servido como vehículos de validación parcial y de diseminación en la comunidad científica. Finalmente, se plantean las líneas futuras de investigación, centradas en la transferencia de conocimiento entre robots y entornos, la integración con algoritmos de optimización local y la exploración de sistemas multimodales que combinen distintas estrategias de planificación.

La estructura del capítulo es la siguiente: En la Sección 7.1 se recogen las conclusiones generales de la investigación; en la Sección 7.2 se revisa la validación de la hipótesis central y de cada uno de los objetivos específicos; en la Sección 7.3 se presentan las principales contribuciones y publicaciones asociadas; y finalmente, en la Sección 7.4 se discuten las líneas de investigación que se abren a partir de este trabajo.

7.1 Resumen de la investigación y conclusiones

El trabajo desarrollado en esta tesis doctoral se ha centrado en la planificación geométrica de trayectorias para manipuladores industriales mediante un enfoque basado en DRL, formulando el problema como un MDP y evaluando la viabilidad de políticas deterministas entrenadas en entornos de simulación ligeros y reproducibles. La hipótesis de partida planteaba que, al integrar un diseño riguroso del espacio de estados, un espacio de acciones articulado físicamente ejecutable, una función de recompensa híbrida con *curriculum learning* y episodios estructurados de manera eficiente, era posible obtener políticas que superasen en consistencia y eficiencia a los planificadores clásicos basados en muestreo.

La investigación se inició con la definición formal del problema, especificando la transición de la planificación geométrica clásica hacia un marco basado en DRL. En este contexto, se diseñó un espacio de estados de 14 dimensiones que combina información articular, errores cartesianos relativos y un indicador binario de colisión. Esta formulación permitió representar la cinemática redundante del robot y proporcionar al agente información suficiente para evitar trayectorias inválidas. El espacio de acciones se definió en términos de incrementos articulares continuos acotados, garantizando la factibilidad geométrica y la compatibilidad con las interfaces de control del robot. La dinámica de transición se implementó en PyBullet, centrada únicamente en cinemática y detección de colisiones, eliminando la simulación dinámica para maximizar la velocidad de entrenamiento.

La función de recompensa se concibió como híbrida, integrando señales densas de error en posición y orientación, junto con una recompensa discreta de éxito. Este esquema se enriqueció con un *curriculum learning* progresivo en el que las tolerancias de precisión se reducen de manera controlada hasta alcanzar valores exigentes. La recompensa de éxito se escala dinámicamente para incentivar la superación de cada nivel de dificultad. Este diseño evitó la convergencia prematura y permitió un aprendizaje gradual hacia políticas de alta precisión. Los episodios se estructuraron con un máximo de 200 pasos y condiciones de

terminación definidas por colisiones, éxito o agotamiento de pasos, manteniendo un equilibrio entre diversidad de experiencias y coste computacional.

El entorno de entrenamiento se diseñó sobre PyBullet, empleando descripciones URDF calibradas para el robot UR3e, lo que proporcionó precisión cinemática y redujo la brecha *sim-to-real*. El uso de URDF aseguró la generalidad del enfoque, permitiendo cargar cualquier robot o escenario sin alterar la lógica de entrenamiento. El entorno se implementó con compatibilidad total con Gymnasium, posibilitando la integración con algoritmos de DRL de forma directa y estandarizada. En cada paso, el agente recibe observaciones estructuradas, aplica acciones incrementales y el entorno actualiza el estado del robot de manera determinista, garantizando reproducibilidad y depuración sistemática.

Durante el entrenamiento, se incorporaron técnicas de aceleración que resultaron decisivas. La inyección de experiencia experta, basada en trayectorias generadas con planificadores de OMPL, permitió poblar el *replay buffer* con transiciones exitosas, reduciendo las fases iniciales de exploración aleatoria e incrementando la estabilidad de la política. Es relevante destacar que los resultados mostraron que esta técnica no indujo comportamientos de imitación directa, sino que facilitó la adquisición de soluciones originales con mejores métricas de longitud, suavidad y tasa de éxito que las trayectorias de referencia. Junto con ello, el uso de *curriculum learning* garantizó que el agente progresara hacia tareas más precisas, manteniendo la estabilidad del entrenamiento. Finalmente, la optimización de hiperparámetros mediante *Optuna* permitió afinar parámetros críticos como la tasa de aprendizaje, el tamaño de lote o los coeficientes de la recompensa, logrando un entrenamiento más robusto.

La evaluación comparativa de algoritmos mostró que SAC supera de manera consistente a alternativas como PPO, TD3 y DDPG. Con seis millones de pasos de entrenamiento, la política final alcanzó una tasa de éxito superior al 94 %, generando trayectorias compactas y suaves con una longitud media en el espacio de configuración inferior a la obtenida por los algoritmos de OMPL. La salida de la política, consistente en trayectorias geométricas, se perfiló temporalmente mediante TOPPRA, incorporando restricciones de velocidad y aceleración articulares para garantizar ejecutabilidad física en el robot real.

La comparación con los planificadores clásicos puso de relieve diferencias sustanciales. Si bien estos últimos son probabilísticamente completos y asintóticamente óptimos en teoría, en la práctica requieren tiempos máximos de planificación que introducen incertidumbre, un aspecto inaceptable en entornos industriales. La política entrenada ofrece, en cambio, un comportamiento determinista y reproducible, con tiempos de planificación del orden de milisegundos y trayectorias que, además de cumplir con los criterios de factibilidad geométrica, exhiben mejores propiedades de suavidad y menor número de puntos intermedios. Cabe señalar que si se limitara el tiempo máximo de planificación de los planificadores tradicionales al tiempo requerido por la política basada en DRL, su tasa de éxito no superaría el 50 %, mientras que la política mantiene resultados consistentes.

En síntesis, los resultados validan la hipótesis inicial: una política determinista entrenada mediante DRL, diseñada con observaciones y acciones adecuadas, enriquecida con experiencia experta y *curriculum learning*, y ejecutada sobre un entorno modular basado en URDF, constituye una alternativa superior a los planificadores clásicos de muestreo. El enfoque ofrece ventajas claras en eficiencia temporal, calidad de las trayectorias, estabilidad y reproducibilidad. Al integrar además un módulo de transferencia al dominio real, que adapta la política a interfaces de ROS con soporte tanto para tiempo real estricto como no garantizado, se ha demostrado que la solución no solo es teóricamente sólida, sino también aplicable en escenarios industriales.

Las conclusiones finales consolidan la validez del enfoque desarrollado: el aprendizaje por refuerzo profundo, cuando se diseña y entrena adecuadamente, no solo resuelve la planificación geométrica de manipuladores, sino que establece un nuevo estándar de eficiencia y fiabilidad frente a los algoritmos clásicos.

7.2 Hipótesis y validación de objetivos

La hipótesis central de esta tesis establecía que es posible diseñar un proceso general basado en DRL que permita obtener una política determinista capaz de resolver la planificación geométrica de trayectorias en manipuladores industriales, alcanzando cualquier meta definida en tiempo de ejecución y aplicable de

forma modular y escalable a diferentes tareas de planificación. Los resultados obtenidos a lo largo del trabajo permiten validar esta hipótesis, ya que se ha demostrado que el enfoque propuesto no solo es competitivo frente a los planificadores clásicos, sino que además los supera en aspectos clave como determinismo, eficiencia temporal, calidad de las trayectorias y consistencia de resultados (véase Capítulo 6.1).

En relación con los objetivos, se planteó un objetivo principal y siete objetivos específicos. A continuación se enumeran los objetivos formulados al inicio de la investigación junto con su validación y la referencia a los capítulos donde se abordan:

Objetivo principal. *Desarrollar un proceso de entrenamiento general, basado en técnicas de DRL, que permita obtener una política determinista capaz de resolver la planificación geométrica de manipuladores robóticos articulados. Esta política deberá ser capaz de alcanzar metas definidas en tiempo de ejecución, cubriendo de forma robusta todo el espacio de trabajo, y concebida para ser escalable, modular y reutilizable como bloque base dentro de planificadores de trayectorias más complejos.*

La validación de este objetivo constituye el eje vertebrador de toda la tesis. El proceso diseñado, descrito en el Capítulo 5, combina un diseño cuidadoso del entorno de entrenamiento con técnicas de aceleración del aprendizaje (*curriculum learning* e inyección de experiencia experta, Sección 5.4) y una evaluación exhaustiva de algoritmos (Sección 5.3). La política final, obtenida mediante SAC, alcanza una tasa de éxito del 94.12 % (Tabla 6.1), superando al 92.97 % del mejor planificador clásico (PRM). Las trayectorias generadas son más cortas (7.20 puntos frente a 18.61) y más suaves (1.2286 frente a 1.2987). Además, el tiempo de planificación promedio es de 0.01322 s, muy inferior al de los planificadores clásicos (Figura 6.1). Estos resultados confirman que el objetivo principal se cumple de forma robusta, y que el enfoque desarrollado constituye una alternativa práctica y eficiente para la planificación geométrica en manipuladores industriales.

OE1. *Analizar las limitaciones de los métodos clásicos de planificación geométrica, basados en muestreo y optimización local, en contextos donde las metas se definen en tiempo de ejecución, identificando sus principales cuellos de botella en términos de adaptabilidad, eficiencia y reutilización.*

Este análisis se desarrolló en el Capítulo 3, donde se identificaron los principales cuellos de botella de los planificadores de OMPL: tiempos de planificación elevados, alta variabilidad en los resultados y dependencia crítica de parámetros de configuración. El Capítulo 6.1 confirmó empíricamente estas limitaciones: Al imponer como tiempo máximo de planificación el empleado por la política de DRL, la tasa de éxito de los planificadores clásicos caería por debajo del 50 %. Esto revela que, en la práctica, los planificadores probabilísticamente completos no son adecuados para entornos industriales donde los tiempos deben estar garantizados. En consecuencia, este objetivo se valida mostrando la necesidad real de un enfoque alternativo.

OE2. *Formular el problema de planificación geométrica como un entorno de aprendizaje por refuerzo profundo, definiendo una representación abstracta y general del estado y la meta, un espacio de acción aplicable a distintos manipuladores, y una función de recompensa que incentive la convergencia rápida, la seguridad y la suavidad del movimiento.*

Este objetivo se aborda en detalle en el Capítulo 5. Se diseñó un MDP con un espacio de estados de 14 dimensiones, que combina información articular, cartesiana y de colisión, evitando redundancias y resolviendo ambigüedades cinemáticas. El espacio de acciones, definido como incrementos articulares acotados, garantiza continuidad y ejecutabilidad en cualquier manipulador. La función de recompensa híbrida, con soporte de *curriculum learning*, resolvió los problemas de convergencia observados con funciones puramente densas o escasas (Sección 5.1.4). Este diseño ha demostrado ser generalizable, pues puede aplicarse a cualquier robot descrito en URDF, validando plenamente el objetivo.

OE3. *Diseñar y entrenar una política determinista y generalizable mediante técnicas de aprendizaje por refuerzo profundo.*

El Capítulo 5 mostró cómo la combinación de estas técnicas permitió alcanzar tolerancias finales de 0.0005 m en posición y 0.1 rad en orientación tras seis millones de pasos de entrenamiento. La progresión del *curriculum* (Figuras 5.5 y 5.6) evidencia que la política avanzó de forma estable y sin estancamientos, mientras que la inyección de experiencia experta aceleró el aprendizaje sin introducir sesgos hacia los planificadores clásicos (Sección 5.4). En conjunto, los mecanismos introducidos cumplieron su propósito y validan este objetivo.

OE4. *Validar empíricamente la capacidad de generalización de la política en escenarios no vistos durante el entrenamiento, verificando su capacidad para alcanzar metas arbitrarias distribuidas por el espacio de trabajo y para operar bajo restricciones articulares y estructurales realistas.*

Este objetivo se evaluó en el conjunto de test (Capítulo 6), donde la política mantuvo una tasa de éxito superior al 94 % y produjo trayectorias consistentes en suavidad y longitud, incluso bajo restricciones articulares y geométricas realistas. Los resultados demuestran que la política no memoriza trayectorias, sino que aprende reglas generales de planificación, lo que valida su capacidad de generalización.

OE5. *Evaluar la modularidad y escalabilidad del enfoque propuesto, aplicando la política como bloque funcional dentro de planificadores jerárquicos y tareas multimodales.*

Este objetivo se abordó en el Capítulo 5, donde se analizó cómo esta política se puede aplicar a tres escenarios distintos: planificación de trayectorias libres, planificación con restricciones en el espacio de trabajo y planificación hacia metas móviles. En este mismo capítulo se presentó el módulo de integración desarrollado en el proyecto ACROBA, que permite adaptar la política a cualquier manipulador descrito en URDF y desplegarla para cualquiera de los escenarios mencionados.

Además, en el Capítulo 6 se validó el funcionamiento de la política para la planificación de trayectorias libres. En todos los casos se mantuvo el rendimiento, demostrando la modularidad y escalabilidad del enfoque. Estos resultados confirman que el objetivo se ha cumplido de forma sólida.

OE6. *Comparar el rendimiento del enfoque con técnicas clásicas de planificación tradicionales, utilizando métricas adecuadas.*

El Capítulo 6 recoge la validación de este objetivo. En todas las métricas, nuestra política basada DRL obtuvo mejores resultados: mayor tasa de éxito (94.12 % frente a 92.97 %), menor longitud en el espacio de configuración (2.5 rad frente a 3 rad de mediana), trayectorias más suaves (1.2286 frente a 1.2987) y tiempos de planificación drásticamente inferiores (0.01322 s frente a 0.03360 s en promedio). A diferencia de los planificadores clásicos, el enfoque propuesto ofrece determinismo total, eliminando la incertidumbre asociada a los límites de tiempo. Estos resultados validan el objetivo de manera contundente.

OE7. *Validar la aplicabilidad de la política en un entorno real mediante su ejecución en un robot físico, evaluando la capacidad de transferencia sim-to-real.*

La validación de este objetivo se expone en el Capítulo 6. El módulo de integración desarrollado en el proyecto ACROBA permitió ejecutar la política en un UR3e real a través de ROS, adaptando automáticamente la interfaz de Gymnasium a servidores de acción. Las trayectorias generadas por la política fueron perfiladas temporalmente mediante TOPPRA, garantizando el respeto de las limitaciones dinámicas del robot.

Se realizaron pruebas para validar los distintos tipos de integración planteados en el módulo de integración. En el modo por lotes, donde la trayectoria completa se genera en simulación y se envía de forma íntegra al controlador, se observaron errores absolutos medios en configuración extremadamente bajos, comprendidos entre 0.00018 y 0.00062

rad (0.01°–0.035°) con desviaciones estándar igualmente reducidas (Tabla 6.2). Estos valores reflejan una fidelidad prácticamente perfecta entre la trayectoria planeada y la ejecutada, lo que confirma que la política transferida es capaz de operar en el robot real con una precisión superior a la requerida habitualmente en entornos industriales.

En el modo de tiempo real estricto, con un 95 % de superposición prematura, el error medio aumentó ligeramente, situándose entre 0.00063 y 0.00083 rad (0.036°–0.048°). Aunque este incremento es atribuible a la propia naturaleza del control en tiempo real, los valores obtenidos siguen siendo notablemente bajos y adecuados para aplicaciones prácticas. Asimismo, el análisis del perfil de velocidades articulares mostró un error medio en el rango de 0.00114–0.00260 rad/s en el modo por lotes, que aumentó hasta 0.02263–0.02543 rad/s en el modo en tiempo real (Tabla 6.3). Este incremento se debe a las restricciones temporales de ejecución, pero aun así la variabilidad medida se mantiene dentro de márgenes razonables, con trayectorias estables y reproducibles.

En ambos modos de operación, los resultados fueron consistentes a lo largo de las 100 trayectorias evaluadas, con desviaciones estándar reducidas en todas las articulaciones, lo que demuestra la robustez del enfoque. La alta fidelidad entre la simulación y la ejecución física confirma que el *gap* sim-to-real se mitiga de manera efectiva gracias a la calibración precisa del modelo URDF, la parametrización temporal con TOPPRA y la integración transparente mediante ROS.

En conjunto, los resultados obtenidos confirman la validez de la hipótesis central y muestran que todos los objetivos planteados al inicio de la investigación han sido alcanzados. La política propuesta constituye un bloque funcional eficiente, determinista y reutilizable, superando a los métodos clásicos y demostrando un potencial de aplicación real en entornos industriales que requieren soluciones rápidas, fiables y reproducibles.

7.3 Contribuciones principales

Los resultados de esta tesis doctoral se han difundido en forma de publicaciones científicas, que reflejan tanto los avances metodológicos como las aplicaciones prácticas del trabajo. En total, se han producido cinco contribuciones: un artículo publicado en revista indexada JCR, un artículo actualmente en revisión y tres artículos en congresos internacionales. Estas publicaciones han servido como vehículos de validación parcial de los objetivos de la tesis, permitiendo contrastar los resultados con la comunidad científica y obtener retroalimentación a lo largo del proceso.

7.3.1 Resumen de publicaciones

La Tabla 7.1 recoge de forma resumida las publicaciones generadas durante el desarrollo de esta tesis, indicando su estado, el capítulo de la memoria en el que se relaciona su contenido y el objetivo específico al que da soporte.

ID	Tipo	Estado	Capítulos relacionados	Objetivos asociados
P1	Revista	Publicado	Cap. 4,Cap. 5,Cap. 6	OE1, OE2, OE3, OE4, OE6
P2	Revista	En revisión	Cap. 5	OE5, OE7
P3	Conferencia	Publicado	Cap. 4	OE5, OE7
P4	Conferencia	Publicado	Cap. 4,Cap. 5	OE3, OE4
P5	Conferencia	Publicado	Cap. 5,Cap. 6	OE5, OE7

Tabla 7.1: Resumen de publicaciones derivadas de la tesis y su relación con los objetivos específicos.

7.3.2 Artículos de revista

P1 - Astorquia et al. (2025). El artículo titulado *Comparative Benchmark of Sampling-Based and DRL Motion Planning Methods for Industrial Robotic Arms*, publicado en la revista *Sensors*, constituye la publicación central de la tesis. Presenta un estudio comparativo entre los planificadores de trayectorias clásicos basados

en muestreo (implementados en OMPL/MoveIt) y la política de DRL entrenada mediante el proceso descrito en los capítulos 4, 5 y 6. El trabajo demuestra que el enfoque propuesto no sólo es competitivo, sino que supera a los métodos clásicos en tasa de éxito, suavidad y compacidad de las trayectorias, además de garantizar tiempos de planificación deterministas. Esta contribución valida los objetivos OE1, OE2, OE3, OE4 y OE6, mostrando la eficacia del enfoque en comparación con las técnicas tradicionales.

I. F. Astorquia, G. Villate-Castillo, A. Tellaeche, & J.-I. Vazquez, "Comparative Benchmark of Sampling-Based and DRL Motion Planning Methods for Industrial Robotic Arms," *Sensors*, vol. 25, no. 17, 2025. doi: 10.3390/s25175282.

P2 - Gonzalez-Santocildes et al. (en revisión). El artículo titulado *Can Robots Adapt to Human Comfort? Experimental Validation of a Reinforcement Learning Approach* se encuentra actualmente en revisión. En este se extiende el uso del aprendizaje por refuerzo profundo hacia el ámbito de la interacción humano-robot. El trabajo incluye validaciones experimentales en entornos reales, donde las políticas entrenadas deben adaptarse a criterios de confort humano, lo que conecta con los mecanismos de transferencia *sim-to-real* descritos en el capítulo 5. Esta publicación está directamente vinculada a los objetivos OE5 y OE7, ya que pone de manifiesto la modularidad y capacidad de adaptación del enfoque propuesto más allá de la planificación geométrica pura.

A. Gonzalez-Santocildes, A. Eguiluz, J.-I. Vazquez & I. F. Astorquia, "Can Robots Adapt to Human Comfort? Experimental Validation of a Reinforcement Learning Approach," *Advanced Intelligent Systems* (en revisión).

7.3.3 **Artículos de congreso**

P3 - Astorquia et al. (2022). El artículo *On the creation of a robotics software architecture for AI-based advanced applications*, presentado en la conferencia ETFA 2022, se centra en la creación de una arquitectura de software para integrar aplicaciones de inteligencia artificial en robótica, proporcionando una base modular y extensible. Se relaciona con el capítulo 4, donde se aborda la construcción del entorno de entrenamiento y la integración de modelos URDF en

PyBullet. Su aportación conecta con los objetivos OE5 y OE7, al evidenciar la importancia de arquitecturas abiertas y escalables para la integración de políticas de DRL en sistemas complejos.

I. F. Astorquia, A. T. Iglesias, B. S. Urquijo, J.-I. Vazquez & I. P. López, "On the creation of a robotics software architecture for AI-based advanced applications," 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), Stuttgart, Germany, 2022, pp. 1–6. doi: 10.1109/ETFA52439.2022.9921495.

P4 - Astorquia et al. (2023). En el trabajo *On Selecting Optimal Hyperparameters for Reinforcement Learning Based Robotics Applications: A Practical Approach*, presentado en ICINCO 2023, se comparte un estudio sobre la selección y ajuste de hiperparámetros en aplicaciones de DRL en robótica. Los resultados experimentales, alineados con el trabajo realizado en los capítulos 4 y 5, muestran cómo el uso de *Optuna* permite explorar de forma sistemática el espacio de hiperparámetros y alcanzar configuraciones estables. Esta contribución está relacionada con los objetivos OE3 y OE4, al reforzar la importancia del diseño metodológico y la validación de la robustez de las políticas en entornos no vistos.

I. F. Astorquia, G. Villate, A. Tellaeche, & J.-I. Vázquez, "On Selecting Optimal Hyperparameters for Reinforcement Learning Based Robotics Applications: A Practical Approach," Proceedings of the International Conference on Informatics in Control, Automation and Robotics, 2023, pp. 123–130. doi: 10.5220/0012158200003543.

P5 - Astorquia et al. (2022). El artículo *A method for multi-robot arm system implementation using the ROS framework*, presentado en la conferencia Interdisciplinary Conference on Mechanics, Computers and Electrics, presenta un método para la implementación de sistemas multi-robot con ROS. Este se vincula estrechamente con los capítulos 5 y 6, donde se discute la transferencia *sim-to-real* y la integración de políticas en arquitecturas reales basadas en ROS. Su aportación está asociada a los objetivos OE5 y OE7, al mostrar la aplicabilidad del enfoque a entornos distribuidos y configuraciones multi-robot.

I. F. Astorquia, A. Tellaeche, & J.-I. Vázquez, “A method for multi-robot arm system implementation using the ROS framework,” 2022 Interdisciplinary Conference on Mechanics, Computers and Electrics, Barcelona, Spain, 2022.

7.3.4 **Síntesis**

En conjunto, estas cinco publicaciones reflejan la evolución de la investigación realizada en esta tesis. Las contribuciones abarcan desde los fundamentos metodológicos (formulación del problema y diseño del entorno) hasta la validación experimental en robots reales, pasando por la comparación sistemática con los métodos clásicos y el análisis del ajuste de hiperparámetros. Cada publicación se alinea de manera directa con uno o varios de los objetivos específicos planteados, confirmando la coherencia entre los resultados obtenidos y las metas iniciales de la tesis.

7.4 **Líneas futuras de investigación**

Aunque los resultados obtenidos en esta tesis doctoral validan la hipótesis planteada y demuestran la aplicabilidad práctica del enfoque propuesto, también abren nuevas oportunidades de investigación que podrían ampliar significativamente el alcance, la robustez y la escalabilidad del método. A continuación se presentan tres líneas principales de trabajo futuro, todas ellas motivadas tanto por los logros alcanzados como por las limitaciones identificadas a lo largo del desarrollo de esta investigación.

7.4.1 **Transferencia de conocimiento a otros robots y entornos**

La primera línea de investigación, y sin duda la más relevante, se refiere a la transferencia de conocimiento a otros robots y entornos. El enfoque desarrollado en esta tesis ha mostrado ser modular y escalable gracias al uso de descripciones URDF, lo que facilita la integración de la política en diferentes plataformas. Sin embargo, el proceso de entrenamiento descrito se ha centrado en un manipulador concreto (UR3e), y aún no se ha evaluado de forma sistemática hasta qué punto la política obtenida puede reutilizarse en otros sistemas.

Una cuestión fundamental es determinar si la política aprendida puede transferirse de manera directa a robots con morfologías distintas, por ejemplo, robots colaborativos de mayor carga, manipuladores con cadenas cinemáticas redundantes o robots cartesianos, o si sería necesario un proceso de adaptación parcial. Este problema conecta con dos campos de creciente interés: el *transfer learning* y el *meta-reinforcement learning*. En ambos casos se busca reducir la dependencia de grandes volúmenes de datos de entrenamiento reutilizando representaciones, políticas o funciones de valor previamente aprendidas en un dominio.

Aplicado al caso de la planificación geométrica en robótica industrial, se abren varias preguntas:

- ¿Cuánto conocimiento de la política base puede aprovecharse realmente al cambiar de robot?
- ¿Qué reducción en el número de pasos de entrenamiento puede lograrse?
- ¿Es viable implementar un esquema de inicialización universal que permita desplegar la misma política en diferentes plataformas con una mínima readaptación?

Responder a estas cuestiones tiene un enorme valor práctico. Si la transferencia resultara efectiva, se podría reducir drásticamente el coste computacional del entrenamiento, lo que haría posible desplegar políticas generalistas en múltiples fábricas y líneas de producción heterogéneas. Además, se avanzaría hacia el desarrollo de bibliotecas de políticas entrenadas previamente que, de manera análoga a los modelos preentrenados en visión por computador o procesamiento del lenguaje, podrían servir como bloques de construcción para aplicaciones industriales específicas.

Este planteamiento conecta con dos campos de creciente interés: el *transfer learning* y el *meta-reinforcement learning* (Zhu et al., 2023). Recientes estudios en manipulación robótica muestran, por ejemplo, que es posible acelerar el aprendizaje en nuevos entornos mediante esquemas de transferencia con pocas muestras (He et al., 2025).

Por último, cabe mencionar que esta línea de trabajo tiene implicaciones directas en el ámbito del *sim-to-real transfer*. La transferencia de conocimiento no solo entre robots distintos, sino también entre entornos con dinámicas o restricciones variables, permitiría validar si el enfoque mantiene su capacidad de generalización cuando las condiciones de operación se apartan de las idealizadas en simulación.

7.4.2 Integración con algoritmos de optimización local

Una segunda línea de investigación se centra en la integración del enfoque propuesto con algoritmos de optimización local, con el objetivo de diseñar planificadores híbridos que combinen rapidez y determinismo con refinamiento de precisión. En esta tesis se ha demostrado que la política propuesta basada en DRL genera trayectorias geométricas más compactas y con menos puntos intermedios que los métodos clásicos de muestreo implementados en OMPL. Esta característica tiene un gran potencial cuando se combina con algoritmos de refinamiento, tales como CHOMP o STOMP, que mejoran la suavidad de trayectorias iniciales mediante optimización iterativa.

La hipótesis de trabajo sería que, al disponer de trayectorias iniciales con menos puntos y mayor regularidad, el proceso de optimización convergería más rápidamente y con mejores resultados que cuando se aplica a trayectorias largas e irregulares generadas por algoritmos de muestreo. Este planteamiento abre varias líneas de experimentación:

- Comparar el refinamiento de trayectorias generadas por la solución propuesta frente a trayectorias de OMPL utilizando métricas de suavidad, tiempo de ejecución y consumo energético.
- Analizar el impacto en el tiempo total de planificación (planificación + optimización) en entornos con restricciones de tiempo real.
- Evaluar si los algoritmos de optimización local, al aplicarse sobre el resultado del enfoque propuesto, pueden compensar pequeñas desviaciones introducidas por la transferencia al robot físico, reduciendo el *sim-to-real gap*.

Este tipo de integración híbrida responde a una tendencia creciente en la robótica industrial: aprovechar los beneficios de la planificación de alto nivel (rápida y determinista) con refinamientos de bajo nivel que aseguren calidad y seguridad en la ejecución. En un escenario ideal, el sistema podría generar en milisegundos una trayectoria válida con DRL y optimizarla localmente en un margen de tiempo reducido, alcanzando un equilibrio entre eficiencia temporal y precisión geométrica.

7.4.3 **Exploración de enfoques multimodales**

Finalmente, una tercera línea de investigación apunta hacia el diseño de sistemas multimodales que integren y coordinen múltiples algoritmos de planificación. La literatura reciente ha mostrado que no existe un planificador universalmente óptimo: cada técnica presenta ventajas en escenarios concretos. Bajo esta perspectiva, un sistema que ejecute en paralelo varios algoritmos (DRL, métodos de muestreo de OMPL, optimización local) y seleccione en tiempo de ejecución la mejor trayectoria según criterios definidos (tiempo, suavidad, distancia, seguridad) podría superar a los enfoques monolíticos.

Este tipo de aproximaciones híbridas ha comenzado a explorarse en navegación móvil mediante la combinación de planificadores clásicos y basados en aprendizaje (Sharma et al., 2024), lo que abre la puerta a investigar esquemas similares en planificación geométrica de manipuladores.

Este paradigma introduce retos interesantes:

- Definir criterios objetivos y eficientes para seleccionar, en tiempo real, la mejor trayectoria entre múltiples candidatas.
- Diseñar arquitecturas que permitan la ejecución concurrente de algoritmos heterogéneos sin penalizar el tiempo de respuesta.
- Explorar esquemas de cooperación donde DRL genere trayectorias iniciales y OMPL actúe como verificador o refinador de seguridad.

Un enfoque multimodal también podría integrarse en el marco de la planificación jerárquica. Por ejemplo, para trayectorias cortas y sin restricciones

complejas podría priorizarse el uso de la política de DRL, mientras que para trayectorias de larga distancia en entornos muy congestionados podría asignarse un planificador clásico. De este modo, se aprovecharían las fortalezas de cada técnica sin que sus limitaciones penalicen el rendimiento global.

En síntesis, esta línea de investigación plantea la posibilidad de diseñar sistemas de planificación verdaderamente adaptativos, donde el criterio de selección entre algoritmos no esté predefinido, sino que emerja dinámicamente en función del problema a resolver. Esto abriría la puerta a sistemas de planificación más robustos y versátiles, capaces de responder con garantías a la diversidad de escenarios presentes en la robótica industrial contemporánea.

Bibliografía

ACROBA Project (2025). Sitio web oficial del proyecto acroba. `https://acrobaproject.eu/`. Accedido: 04-Feb-2025. 158

Aine, S. and Likhachev, M. (2013). Anytime truncated d* : Anytime replanning with truncation. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search (SoCS 2013)*, volume 4, pages 2–10. AAAI Press. 78

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2623–2631, New York, NY, USA. Association for Computing Machinery. 135

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2018). Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 31, pages 5048–5058. 92

Åström, K. and Hägglund, T. (1995). *PID Controllers: Theory, Design, and Tuning*. ISA - The Instrumentation, Systems and Automation Society, 2 edition. 41

Bhatt, P., Malhan, R., Shembekar, A., Yoon, Y. J., and Gupta, S. (2019). Expanding capabilities of additive manufacturing through use of robotics technologies: A survey. *Additive Manufacturing*, 31:100933. v, 26

Bobrow, J. E., Dubowsky, S., and Gibson, J. S. (1985). Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17. 39

Bohlin, R. and Kavraki, L. E. (2000). Path planning using lazy prm. In *Proceedings-IEEE International Conference on Robotics and Automation*, volume 1, pages 521–528. IEEE. 75

Chamzas, C., Quintero-Peña, C., Kingston, Z., Orthey, A., Rakita, D., Gleicher, M., Toussaint, M., and Kavraki, L. E. (2021). Motionbenchmaker: A tool to generate and benchmark motion planning datasets. *IEEE Robotics and Automation Letters*, 7:882–889. 94, 98

Choudhury, S., Gammell, J. D., Barfoot, T. D., Srinivasa, S., and Scherer, S. (2016). Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning. In IEEE, editor, *Proceedings of (ICRA) International Conference on Robotics and Automation*, pages 4207–4214. 84

Cohen, B., Şucan, I. A., and Chitta, S. (2012). A generic infrastructure for benchmarking motion planners. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 589–595. IEEE. 93

Coleman, D. T., Sucan, I. A., Chitta, S., and Correll, N. (2014). Reducing the barrier to entry of complex robotic software: a moveit! case study. *Journal of Software Engineering in Robotics*, 5(1):3–16. 79

Collins, J., Chand, S., Vanderkop, A., and Howard, D. (2021). A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431. 108, 110

Coumans, E. and Bai, Y. (2016). *Pybullet, a python module for physics simulation for games, robotics and machine learning*. Manual de software. 109, 128

Craig, J. J. (2004). *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, 3 edition. 21, 27

Dai, S., Orton, M., Schaffert, S., Hofmann, A., and Williams, B. (2018). Improving trajectory optimization using a roadmap framework. In *Proceedings of the IEEE Aerospace Conference (AERO 2018)*, pages 1–9. IEEE. 84, 85

Diankov, R. (2010). *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute. Tesis doctoral. 115, 118

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271. 35

Dobson, A. and Bekris, K. E. (2014). Sparse roadmap spanners for asymptotically near-optimal motion planning. *The International Journal of Robotics Research*, 33(1):18–47. 65, 76

Elguea-Aguinaco, I., Inziarte-Hidalgo, I., Bøgh, S., and Arana-Arexolaleiba, N. (2024). A review on reinforcement learning for motion planning of robotic manipulators. *International Journal of Intelligent Systems*, 2024:1636497. vi, 89

Fidalgo, I., Villate, G., Tellaeche, A., and Vázquez, J. (2023). On selecting optimal hyperparameters for reinforcement learning based robotics applications: A practical approach. In *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO*, pages 123–130. INSTICC, INSTICC, SciTePress. 135

Fresnillo, P. M., Vasudevan, S., Mohammed, W. M., Lastra, J. L. M., and García, J. A. P. (2023). Extending the motion planning framework-moveit with advanced manipulation functions for industrial applications. *Robotics and Computer-Integrated Manufacturing*, 85:102503. 80

Fujimoto, S., Hoof, H. v., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80, pages 2587–2601. PMLR. 91, 135

Gammell, J. D., Barfoot, T. D., and Srinivasa, S. S. (2018). Informed sampling for asymptotically optimal path planning. *IEEE Transactions on Robotics*, 34(4):966–984. 77

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 2997–3004. Institute of Electrical and Electronics Engineers Inc. vi, 77, 78

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2015). Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2015)*, volume 2015-June, pages 3067–3074. Institute of Electrical and Electronics Engineers Inc. 77

Gasparetto, A. and Scalera, L. (2019). From the unimate to the delta robot: The early decades of industrial robotics: Proceedings of the 2018 hmm iftomm symposium on history of machines and mechanisms. In Ceccarelli, M., editor, *History of Machines and Mechanisms: Proceedings of the 2018 HMM IFToMM Symposium on History of Machines and Mechanisms*, pages 284–295. Springer. 53

Geraerts, R. and Overmars, M. H. (2007). Creating high-quality paths for motion planning. *The International Journal of Robotics Research*, 26(8):845–863. 78

Gipson, B., Moll, M., and Kavraki, L. E. (2013). Resolution independent density estimation for motion planning in high-dimensional spaces. In *2013 IEEE International Conference on Robotics and Automation*, pages 2437–2443. IEEE. 67

Gleave, A., Taufeeque, M., Rocamonde, J., Jenner, E., Wang, S. H., Toyer, S., Ernestus, M., Belrose, N., Emmons, S., and Russell, S. (2022). Imitation: Clean imitation learning implementations. arXiv:2211.11972v1 [cs.LG]. 87

Ha, H., Xu, J., and Song, S. (2020). Learning a decentralized multi-arm motion planner. *Proceedings of Machine Learning Research*, 155:103–114. 91

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. volume 5, pages 2976–2989. International Machine Learning Society (IMLS). 91, 135

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107. 35

Hauser, K. and Ng-Thow-Hing, V. (2010). Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *2010 IEEE International Conference on Robotics and Automation*, pages 2493–2498. IEEE. 78

Haviland, J. and Corke, P. (2024). Robotics software: Past, present, and future. *Annual Review of Control, Robotics, and Autonomous Systems*, 7(Volume 7, 2024):253–283. 54

He, Y., Wallbridge, C. D., Hernández, J. D., and Colombo, G. B. (2025). Few-shot transfer learning for deep reinforcement learning on robotic manipulation tasks. In Huda, M. N., Wang, M., and Kalganova, T., editors, *Towards Autonomous Robotic Systems*, pages 85–92, Cham. Springer Nature Switzerland. 205

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4572–4580, Red Hook, NY, USA. Curran Associates Inc. 87

Hsu, D., Latombe, J.-c., and Motwani, R. (1997). Path planning in expansive configuration spaces. *International Journal of Computational Geometry & Applications*, 9:495–512. 66

Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In Morán, F., Moreno, A., Merelo,

J. J., and Chacón, P., editors, *Advances in Artificial Life*, pages 704–720, Berlin, Heidelberg. Springer Berlin Heidelberg. 48

James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., and Bousmalis, K. (2019). Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks . In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12619–12629, Los Alamitos, CA, USA. IEEE Computer Society. 112

Janson, L. and Pavone, M. (2016). *Fast Marching Trees: A Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions*, pages 667–684. Springer International Publishing, Cham. 67, 68, 69

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E., and Levine, S. (2019). Residual reinforcement learning for robot control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2019)*, volume 2019-May, pages 6023–6029. Institute of Electrical and Electronics Engineers Inc. 92

Jurgenson, T. and Tamar, A. (2019). Harnessing reinforcement learning for neural motion planning. In *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, pages 16–29. PMLR. 91

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). Stomp: Stochastic trajectory optimization for motion planning. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4569–4574. 58, 81

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. (2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of Machine Learning Research*, volume 87, pages 651–673. ML Research Press. 91

Kamat, J., Ortiz-Haro, J., Toussaint, M., Pokorny, F. T., and Orthey, A. (2022). Bitkomo: Combining sampling and optimization for fast convergence in optimal motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3553–3560. IEEE. 83, 84, 85

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30:846–894. 59, 68, 69

Kaup, M., Wolff, C., Hwang, H., Mayer, J., and Bruni, E. (2024). A review of nine physics engines for reinforcement learning research. *arXiv preprint arXiv:2407.08590*. July 2024. 108, 110, 111

Kavraki, L. E., Švestka, P., Latombe, J. C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580. 5, 35, 56, 62, 64

Khatib, O. (1990). *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, pages 396–404. Springer New York, New York, NY. 55

Kim, D., Kwon, Y., and Yoon, S.-e. (2018). Adaptive lazy collision checking for optimal sampling-based motion planning. In *2018 15th International Conference on Ubiquitous Robots (UR)*, pages 320–327. 75

Kim, D. and Yoon, S. E. (2020). Simultaneous planning of sampling and optimization: study on lazy evaluation and configuration free space approximation for optimal motion planning algorithm. *Autonomous Robots*, 44:165–181. 84

Kindel, R., Hsu, D., Latombe, J.-C., and Rock, S. (2000). Kinodynamic motion planning amidst moving obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2000)*, volume 1, pages 537–543. IEEE. 70

Kingston, Z. and Kavraki, L. E. (2022). Robowflex: Robot motion planning with moveit made easy. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3108–3114. IEEE. 94

Kuffner, J. J. and LaValle, S. M. (2000). Rrt-connect: an efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2000)*, volume 2, pages 995–1001. 73, 74

Kumar, V., Gupta, A., Todorov, E., and Levine, S. (2016). Learning dexterous manipulation policies from experience and imitation. *arXiv preprint arXiv:1611.05095*. 112

Kunz, T. and Stilman, M. (2013). Time-optimal trajectory generation for path following with bounded acceleration and velocity. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, pages 208–215. IEEE. 39

Körber, M., Lange, J., Rediske, S., Steinmann, S., and Glück, R. (2021). Comparing popular simulation environments in the scope of robotics and reinforcement learning. In *Proceedings of the IEEE International Conference on Industrial Technology (ICIT 2021)*, pages 1192–1199. IEEE. 108

Ladd, A. M. and Kavraki, L. E. (2005). Fast tree-based exploration of state space for robots with dynamics. In Erdmann, M., Overmars, M., Hsu, D., and van der Stappen, F., editors, *Algorithmic Foundations of Robotics VI*, pages 297–312. Springer Berlin Heidelberg, Berlin, Heidelberg. 67

LaValle, S. and Kuffner, J. (1999). Randomized kinodynamic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1999)*, volume 1, pages 473–479. IEEE. 70

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign. Informe técnico. 35, 56, 62, 66

Lavalle, S. M. (2006). *Planning Algorithms*. Cambridge University Press. v, 5, 22, 66

Li, Y., Littlefield, Z., and Bekris, K. (2014). Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5). 71

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*. 43, 90, 91, 135

Liu, H., Ying, F., Jiang, R., Shan, Y., and Shen, B. (2024). Obstacle-avoidable robotic motion planning framework based on deep reinforcement learning. *IEEE/ASME Transactions on Mechatronics*. 92

Liu, S. and Liu, P. (2021). Robot motion planning benchmarking and optimization through motion planning pipeline. *IEEE International Conference on Automation Science and Engineering*, 2021-August:633–638. 84

Liu, S. and Liu, P. (2022). Benchmarking and optimization of robot motion planning with motion planning pipeline. *International Journal of Advanced Manufacturing Technology*, 118(3–4):949–961. 60, 83, 94, 97

Liu, Y., Xu, H., Liu, D., and Wang, L. (2022). A digital twin-based sim-to-real transfer for deep reinforcement learning-enabled industrial robot grasping. *Robotics and Computer-Integrated Manufacturing*, 78:102365. 92

Lobbezoo, A. and Kwon, H.-J. (2023). Simulated and real robotic reach, grasp, and pick-and-place using combined reinforcement learning and traditional controls. *Robotics*, 12(1). 109, 110

Lozano-Perez, T. (1983). Robot programming. *Proceedings of the IEEE*, 71(7):821–841. 53, 54, 55

Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120. 55

Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10. 4

Luo, S. and Li, Q. (2024). A review of the trajectory planning of industrial robots. *Transactions on Engineering and Technology Research*, 2:203–207. 62

Martinez, C., Barrero, N., Hernandez, W., Montaño, C., and Mondragon, I. (2017). Setup of the yaskawa sda10f robot for industrial applications, using ros-industrial. *Lecture Notes in Networks and Systems*, 13:186–203. 80

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2015a). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533. 43

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015b). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533. 90

Moll, M., Sucan, I. A., and Kavraki, L. E. (2015). Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. *IEEE Robotics & Automation Magazine*, 22(3):96–102. 93, 172

Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., and Ueda, K. (2016). Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2):621–641. 4, 5

Muhayyuddin, Akbari, A., and Rosell, J. (2017). Physics-based motion planning with temporal logic specifications. *IFAC-PapersOnLine*, 50(1):8993–8999. 20th IFAC World Congress. 72

Mukadam, M., Dong, J., Yan, X., Dellaert, F., and Boots, B. (2018). Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340. 82

Panerati, J., Zheng, H., Zhou, S., Xu, J., Prorok, A., and Schoellig, A. P. (2021). Learning to fly – a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. 110, 111

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3803–3810. 92

Pfeiffer, B., May-Benson, T., and Bodison, S. (2017). State of the science of sensory integration research with children and youth. *American Journal of Occupational Therapy*, 72:7201170010p1. 87

Pham, H. and Pham, Q. C. (2018). A new approach to time-optimal path parameterization based on reachability analysis. *IEEE Transactions on Robotics*, 34:645–659. 39, 152, 173

Pieper, D. L. (1968). *The Kinematics of Manipulators Under Computer Control*. PhD thesis, Stanford University. 29

Plaku, E. (2012). Path planning with probabilistic roadmaps and co-safe linear temporal logic. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2269–2275. 71

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience. 90

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*. 79

Qureshi, A. and Yip, M. (2018). Deeply informed neural sampling for robot motion planning. pages 6582–6588. 95

Qureshi, A. H., Simeonov, A., Bency, M. J., and Yip, M. C. (2019). Motion planning networks. 88

Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 489–494. 58, 81

Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. 87

Sánchez, G. and Latombe, J.-C. (2003). A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In Jarvis, R. A. and Zelinsky, A., editors, *Robotics Research*, pages 403–417, Berlin, Heidelberg. Springer Berlin Heidelberg. 76

Santos, C. H., Robinson, M. M., and Unemyr, E. (2020). Ros-industrial: An open-source approach to revolutionizing industrial automation - webinar — ros-industrial. 80

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33:1251–1270. 58, 81

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. 91, 135

Sharma, V. D., Lee, J., Andrews, M., and Hadžić, I. (2024). Hybrid classical/rl local planner for ground robot navigation. 207

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Springer. 22, 25, 38

Song, J., Lanka, R., Zhao, A., Bhatnagar, A., Yue, Y., and Ono, M. (2020). Learning to search via retrospective imitation. In *Proceedings of Machine Learning Research*. PMLR. 87

Spong, M. W., Hutchinson, S., and Vidyasagar, M. (2006). *Robot Modeling and Control*. Wiley. 19

Srihari, K. and Deisenroth, M. P. (1988). Robot programming languages—a state of the art survey. In Radharamanan, R., editor, *Robotics and Factories of the Future '87*, pages 625–635, Berlin, Heidelberg. Springer Berlin Heidelberg. 54

Strub, M. P. and Gammell, J. D. (2020). Advanced bit* (abit*): Sampling-based planning with advanced graph-search techniques. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, page 130–136. IEEE. 78

Şucan, I. A. and Kavraki, L. E. (2010). *Kinodynamic Motion Planning by Interior-Exterior Cell Exploration*, pages 449–464. Springer Berlin Heidelberg, Berlin, Heidelberg. 71

Sultanov, R., Sulaiman, S., Li, H., Meshcheryakov, R., and Magid, E. (2022). A review on collaborative robots in industrial and service sectors. In *2022 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–7. 4, 6

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2 edition. v, 6, 42, 43, 89, 91

Tai, L., Paolo, G., and Liu, M. (2017). Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 31–36. 6

Tamizi, M. G., Yaghoubi, M., and Najjaran, H. (2023). A review of recent trend in motion planning of industrial robots. *International Journal of Intelligent Robotics and Applications*, 7:253–274. 91, 98, 99

Tang, C., Abbatematteo, B., Hu, J., Chandra, R., Martín-Martín, R., and Stone, P. (2024). Deep reinforcement learning for robotics: A survey of real-world successes. 135

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017a). Domain randomization for transferring deep neural networks from simulation to the real world. 49, 88

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017b). Domain randomization for transferring deep neural networks from simulation to the real world. *IEEE International Conference on Intelligent Robots and Systems*, 2017-September:23–30. 92

Toussaint, M. (2017). *A Tutorial on Newton Methods for Constrained Trajectory Optimization and Relations to SLAM, Gaussian Process Smoothing, Optimal*

Control, and Probabilistic Inference, pages 361–392. Springer International Publishing, Cham. 82

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, H., and Younis, O. G. (2024). Gymnasium: A standard interface for reinforcement learning environments. 133, 160

Universal Robots (2025). Sitio web oficial de universal robots. `https://www.universal-robots.com/es/`. Accedido el 1 de enero de 2025. vi, 112

Wang, C., Zhang, Q., Tian, Q., Li, S., Wang, X., Lane, D., Petillot, Y., and Wang, S. (2020). Learning mobile manipulation through deep reinforcement learning. *Sensors*, 20(3). 6

Wang, J., Zhang, T., Ma, N., Li, Z., Ma, H., Meng, F., and Meng, M. (2021a). A survey of learning-based robot motion planning. *IET Cyber-Systems and Robotics*, 3. 60, 61

Wang, J., Zhang, T., Ma, N., Li, Z., Ma, H., Meng, F., and Meng, M. (2021b). A survey of learning-based robot motion planning. *IET Cyber-Systems and Robotics*, 3. 87

Zhang, L., Cai, K., Sun, Z., Bing, Z., Wang, C., Figueredo, L., Haddadin, S., and Knoll, A. (2024). Motion planning for robotics: A review for sampling-based planners. 58

Zhang, S., Xia, Q., Chen, M., and Cheng, S. (2023). Multi-objective optimal trajectory planning for robotic arms using deep reinforcement learning. *Sensors*, 23(13). 91

Zhao, W., Queralta, J. P., and Westerlund, T. (2020). Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. 48

Zhu, Z., Lin, K., Jain, A. K., and Zhou, J. (2023). Transfer learning in deep reinforcement learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45:13344. 205

Şucan, I. A., Moll, M., and Kavraki, L. (2012). The open motion planning library. *IEEE Robotics and Automation Magazine*, 19:72–82. 57

Declaración de Autoría

Yo, Ignacio Fidalgo Astorquia, declaro que esta tesis doctoral es un trabajo original realizado por mí como estudiante de doctorado en la Universidad de Deusto. Toda la ayuda recibida y las ideas procedentes de otras fuentes han sido debidamente identificadas, reconociendo sus correspondientes contribuciones y citándolas adecuadamente.

Este trabajo no contiene material que haya sido presentado, en forma idéntica o similar, ante ningún tribunal académico, salvo en aquellos casos en los que se indique expresamente en la tesis.

Esta tesis fue finalizada el día 18 de Septiembre de 2025.