Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 22 (2025) 57-68

www.revista-riai.org

Una estrategia híbrida de aprendizaje por refuerzo informada por RRT* para la planificación de caminos de robots móviles en minería a cielo abierto

Sebastian Zapata[a], Ricardo Urvina[a], Katherine Aro[a], Eduardo Aguilar[a], Fernando Auat Cheein[b,c], Alvaro Prado,[a,*]

[a]*Departamento de Ingeniería de Sistemas y Computación, Universidad Católica del Norte, Casa Central, Av. Angamos 0610, Antofagasta 1249004, Chile*
[b]*Departmento de Ingeniería Electrónica, Universidad Técnica Federico Santa María, Valparaiso 2340000, Chile*
[c]*School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, EH14 4AS, UK*

To cite this article: Zapata, S., Urvina, R., Aro, K., Aguilar, E., Cheein, F., Prado A. 2025. A hybrid reinforcement learning strategy informed by RRT* for path planning of mobile robots in open-pit mines. Revista Iberoamericana de Automática e Informática Industrial 22, 57-68. https://doi.org/10.4995/riai.2024.21581

Resumen

Este trabajo introduce una estrategia híbrida de planificación de caminos para vehículos robóticos tipo diferencial, combinando métodos de aprendizaje por refuerzo con técnicas de muestreo aleatorio. Específicamente, se utiliza aprendizaje por refuerzo basado en Q-Learning (QL) para encontrar un camino global mediante la exploración y explotación de la información del entorno, donde un agente aprende a tomar acciones maximizando recompensas. El agente utiliza un método de muestreo RRT* (Rapidly-exploring Random Trees) para obtener puntos factibles de camino y acelerar la búsqueda, combinando las ventajas de QL con RRT* (MQL) para mejorar el muestreo y generar caminos suaves y factibles en espacios de alta dimensionalidad (Smooth Q-Learning - SMQL). Se realizó un análisis de rendimiento del método híbrido propuesto en condiciones de minería a cielo abierto, considerando criterios de maniobrabilidad, completitud, alcanzabilidad y robustez en entornos rectos, estrechos, intrincados y tipo helicoidal con restricciones de terreno. Mediante simulaciones se demostró que SMQL supera las limitaciones de QL y RRT*, al lograr mejor exploración y alcanzar rápida convergencia de recompensas. Por completitud, caminos previamente planificados con SMQL y MQL se prueban en un controlador de movimiento y un robot Husky A200, alcanzando una reducción del costo de error del 81.9 % y 76.4 % y esfuerzo de control del 79.8 % y 83.5 % en comparación con QL, respectivamente. Se espera que estos resultados tengan un impacto en el ahorro de recursos energéticos del robot al seguir rutas planificadas en ambientes mineros.

Palabras clave: Planificación de caminos, Q-Learning, RRT*, robot móvil autónomo, minería a cielo abierto

A hybrid reinforcement learning strategy informed by RRT* for path planning of mobile robots in open-pit mining

Abstract

This work introduces a hybrid path planning strategy for differential-drive robotic vehicles, combining reinforcement learning methods with sampling techniques. Specifically, Q-Learning (QL) is used to find a global path by exploring and exploiting environmental information, where an agent learns to take actions while maximizing rewards. The agent uses a random sampling method based on Rapidly-exploring Random Trees (RRT*) to speed up the search of feasible navigation points, combining the advantages of QL with RRT* (MQL) to improve sampling and generate smooth and feasible paths in high-dimensional spaces (Smooth Q-Learning - SMQL). The effectiveness of the proposed hybrid method was validated under open-pit mining conditions through a performance analysis based on maneuverability, completeness, reachability, and robustness in environments such as straight roads, narrow spaces, intricate areas, and helicoidal configurations with terrain constraints. Simulations demonstrated that SMQL overcomes the limitations of QL and RRT*, achieving suitable exploration of the search space and rapid convergence of rewards. Paths previously planned with SMQL and MQL are tested on a motion controller and a Husky A200 robot, achieving a reduction in error cost of 81.9 % and 76.4 % and control effort of 79.8 % and 83.5 % compared to QL, respectively. It is expected that these results will impact energy resource savings for the robot when following planned routes in mining environments.

Kewords: Path planning, Q-Learning, RRT*, autonomous mobile robot, open-pite mining

*Autor para correspondencia: alvaro.prado@ucn.cl

1. Introducción

La industria minera es vital en el sector primario global, destacándose por sus ingresos y su impacto social significativo en el desarrollo humano (Gao et al., 2020). La eficiencia en la extracción mineral ha evolucionado hacia la maximización de recursos y la minimización de costos, empleando técnicas de geoquímica, geofísica y cartografía digital (Zhang et al., 2021; Vásquez et al., 2023). En términos de seguridad laboral, se implementan tecnologías para mitigar riesgos y proteger a los trabajadores contra accidentes (Peng et al., 2019). Estas incluyen maquinaria autónoma, sistemas de monitoreo y robots específicos (ver Figura 1) (Betz et al., 2022; Zhao and Bi, 2020a). Por ejemplo, en (Zhao and Bi, 2020b) se propone la planificación de rutas para camiones en minería a cielo abierto, utilizando mapas digitales de alta resolución para optimizar el transporte y mejorar la eficiencia operativa en la mina. A pesar de estas medidas, los accidentes en la industria minera persisten, en su mayoría debido a errores humanos o fallos de comunicación (Zhao et al., 2009; Lazányi, 2023), subrayando la necesidad continua de sistemas autónomos de planificación de movimiento en este sector.

La inclusión de vehículos autónomos en tareas de transporte de minerales puede minimizar los riesgos asociados a la conducción manual, aunque persisten desafíos como la navegación, tracción, estabilidad y consumo de energía (Prado et al., 2018c). La planificación de caminos es crucial en un ambiente minero, impactando costos al optimizar recursos y reducir tiempos de inactividad. Existen soluciones que consideran el riesgo de colisión para mejorar la confiabilidad (Ko et al., 2020), buscan rutas óptimas para reducir el tiempo de llegada evitando obstáculos lo más pronto posible (Goutham et al., 2023), y minimizan distancias de recorrido considerando perfiles de velocidad y puntos estratégicos (Tan et al., 2021). A pesar de esto, la condición del terreno, crucial para la navegabilidad en minería por la tracción crítica de la carga y desgaste de la rueda (Aguilera-Marinovic et al., 2017; Prado et al., 2021), es a menudo omitida. Este trabajo se enfoca en una solución de guiado autónomo adaptable a estas condiciones.

Los vehículos autónomos en la minería siguen rutas predefinidas mediante puntos de referencia, especialmente en operaciones no manuales (Zhang et al., 2021). Este proceso involucra establecer inicialmente un camino mediante un operario auxiliar utilizando un vehículo referente, registrando balizas de posición georreferenciadas. Estos puntos conforman un camino preestablecido que guía al vehículo autónomo hacia áreas de faena o descarga, típicamente inalterable una vez establecido. Sin embargo, estos vehículos enfrentan limitaciones al no poder adaptarse a imprevistos en la ruta (Prado et al., 2018b). Por ejemplo, en (Zhang et al., 2024), se investigó la planificación de rutas en túneles mineros, donde las señales de posicionamiento son obstruidas y la estructura compleja dificulta la navegación hacia puntos de extracción. Esto puede llevar a problemas como la detención de la flota de carga y retrasos significativos en el transporte. Por tanto, se requiere una planificación flexible que permita ajustes en tiempo real ante obstáculos y cambios, abordando así los desafíos operativos en entornos mineros.

Frente a los inconvenientes del guiado autónomo, este trabajo de investigación propone un planificador de caminos que

Figura 1: Robot prototipo para planificadores de caminos.

sea capaz de proveer rutas de referencia para un sistema de control realimentado que busca el movimiento autónomo de vehículos robóticos mineros. La estrategia híbrida del planificador se basa en una ruta global libre de obstáculos definida por un modelo de Q-Learning (QL) y una planificación local basada en RRT*, aprovechando la exploración de espacios continuos y la explotación del ambiente. Aunque los obstáculos de terreno no son estructurales ni móviles, representan un obstáculo dinámico debido a la naturaleza variable del terreno, resaltando la importancia de la planificación local. Para el planificador propuesto, el espacio de trabajo es caracterizado por un mapa topológico del ambiente mediante una cámara RGB que cubre la zona navegable y la posición de obstáculos, mientras que los obstáculos pueden seguir una dinámica desconocida. A diferencia de otros trabajos, se considera la condición de la superficie del terreno con tracción disminuida dentro del esquema de planificación, implicando la inclusión de regiones resbaladizas como obstáculos para la maniobrabilidad, siendo así parcialmente transitables ciertos sectores. Dado el uso práctico de robots autónomos tipo diferencial en ambientes mineros y sus capacidades inerciales y de estabilidad, el método híbrido busca generar un camino compatible con las dinámicas de un robot diferencial. Para validar esta propuesta, se compara el desempeño del planificador híbrido con el del planificador original (Cheein et al., 2017). Para evaluar el desempeño del planificador propuesto se utilizan métricas como el error acumulado de seguimiento de camino, esfuerzo de control acumulado, tiempo de respuesta y convergencia en términos de recompensa del algoritmo de aprendizaje (Prado et al., 2018a). Las pruebas se realizan en simulación y experimentación de campo bajo distintas configuraciones del espacio de trabajo y terreno.

Lo que continua del escrito está organizado de la siguiente manera. La segunda sección presenta los trabajos relacionados en el estado del arte, mientras que la tercera sección presenta la metodología propuesta para el diseño del planificador. En lo

que continúa se presenta la implementación del planificador, las pruebas, y los resultados con su respectivo análisis. Finalmente, en la última sección se exponen las conclusiones de este trabajo.

2. Trabajos relacionados

Los métodos como búsqueda aleatoria, campos potenciales artificiales y enfoques bioinspirados (Jayaweera and Hanoun, 2020; Zhang et al., 2019; Luo et al., 2022) pueden presentar alta carga computacional, limitaciones en tiempo real, y dependencia de las dinámicas del robot y del entorno. En contraste, algoritmos de optimización como Rapidly-exploring Random Trees (RRT*) se centran en la exploración del espacio de búsqueda mediante la construcción gradual de un árbol de decisión hasta alcanzar el objetivo (Chen et al., 2021). Sin embargo, estudios como en (Cheein et al., 2017) señalan que RRT* no siempre garantiza un camino completo con un número limitado de muestras aleatorias, lo que subraya la necesidad de métodos complementarios que optimicen la exploración con información previamente aprendida del camino o del mapa de navegabilidad.

Los planificadores de rutas basados en datos, como el aprendizaje supervisado o por refuerzo, emplean técnicas de aprendizaje automático para determinar un camino óptimo utilizando la experiencia (Aradi, 2022; Prado et al., 2018a). A diferencia de los planificadores basados en aprendizaje supervisado, los planificadores por refuerzo pueden aprovechar la información en curso generada por la interacción entre el agente y su entorno, lo que los hace favorables en tareas repetitivas de la minería donde se priorizan rutas navegables (Agrawal et al., 2022). Por ejemplo, el método de Q-Learning (QL) (Yin et al., 2022) prescinde parcialmente de un modelo y utiliza datos del estado del robot en tiempo real para optimizar una política que maximice las recompensas a lo largo del aprendizaje.

Trabajos recientes (Fakhrudin et al., 2022; Wang et al., 2022) han desarrollado planificadores de caminos basados en QL en entornos complejos, mejorando la precisión y la tasa de convergencia de las decisiones del agente. Los algoritmos QL combinados con redes neuronales profundas, como Deep QL, también buscan encontrar soluciones en ambientes complejos como laberintos intricados, aunque la exploración efectiva y el rendimiento computacional dependen de la estructura del ambiente y la complejidad de los datos de entrenamiento (Nippun and Kochuvila, 2023; Orozco-Rosas et al., 2022; Zheng and Liu, 2019). Entonces, este trabajo se centra en desarrollar una solución computacionalmente simplificada y adaptativa que aprenda del entorno para su implementación en vehículos robóticos en ambientes de operaciones mineras.

3. Metodología

En esta sección se detalla el diseño del planificador de caminos propuesto para la guía de vehículos autónomos terrestres bajo condiciones de minería en cielo abierto. El esquema de la Figura 2 muestra la arquitectura diseñada para el planificador propuesto, el cual considera una estrategia híbrida donde se obtiene un camino preliminar mediante la exploración y explotación de información del ambiente para evitar posibles colisiones, y luego se utiliza un camino sub-muestreado base como nueva información para aprender de la experiencia pasada y de la inspección del ambiente. Este proceso se lleva a cabo a través de dos métodos de planificación, siendo el primero Q-Learning (QL) basado en la maximización permanente de recompensas y el segundo RRT* para generar muestras pseudo-aleatorias de posición. Una vez generada la solución del método propuesto

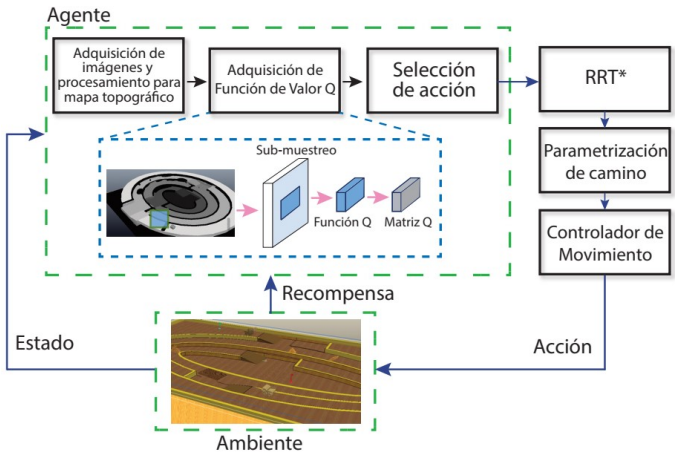


Figura 2: Arquitectura general del planificador de caminos propuesto, utilizando un esquema híbrido entre QL y RRT*.

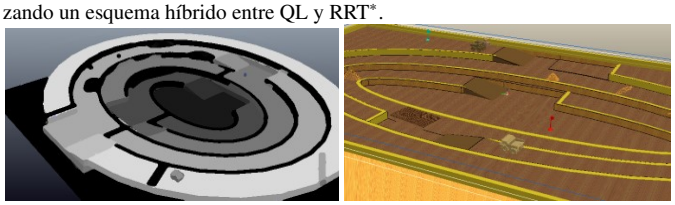


Figura 3: Ambiente de simulación y mapa, utilizados como entrada para el algoritmo de planificación propuesto.

$[x_{MQL}, y_{MQL}]^T$, se considera la parametrización del camino en función de un perfil prescrito de velocidad máxima con el fin de obtener una trayectoria cinemáticamente compatible con las dinámicas del robot $[x_{SMQL}, y_{SMQL}]^T$. Como requisito, los algoritmos requieren conocimiento de la posición inicial $[x_0, y_0]^T$, la posición objetivo $[x_f, y_f]^T$, y el mapa del ambiente M. Para realizar una validación funcional del camino y del desempeño que resulte de su uso en un robot autónomo, se aplica un controlador de movimiento para seguimiento de camino en función de acciones de tracción y rotación.

3.1. *Renderización del mapa topográfico*

Dado que para un planificador de caminos es esencial el conocimiento a priori del entorno, en esta sección se indica el proceso de como se obtiene, acondiciona y renderiza el mapa para el diseño. Entonces, un mapa topográfico del espacio de trabajo es extraído de un entorno de navegación, el cual es ensamblado con formas convexas para formas estructuradas del entorno y no-convexas para el terreno, permitiendo adicionalmente incorporar características cinemáticas y dinámicas de un robot. La Figura 3 muestra un mapa del ambiente de prueba y una vista 3D capturada con una cámara RGB. Desde una elevación que permite cubrir el área del espacio de trabajo, la ubicación del sistema de visión asume una perspectiva vertical a la superficie del terreno. El procesamiento de las imágenes del ambiente comprende una serie de transformaciones hasta llegar a una imagen utilizada en el algoritmo de planificación propuesto. Brevemente:

1. Se captura el entorno minero con una cámara, obteniendo una imagen RGB que se transforma en escala de grises.

2. La imagen se redimensiona y se aplican transformaciones homogéneas para alinear el mapa con los ejes del sistema de trabajo.
3. Se calibran los parámetros intrínsecos de la cámara y los coeficientes de enfoque para reducir distorsión y ruido.
4. Se aplica un filtro Gaussiano para resaltar los píxeles relevantes y eliminar datos irrelevantes (Geusebroek et al., 2003).
5. Se ajustan los puntos de posiciones iniciales y finales al nuevo marco de referencia, generando una imagen en escala de grises acorde a la complejidad del ambiente.

El ambiente permite interactuar con las dinámicas del robot en terrenos con coeficientes de rozamiento dinámico entre [0.1, -1]. La información de rozamiento se asocia a cada punto del entorno dentro del mapa. Esta condición permite que ciertas zonas sean navegables y otras parcialmente transitables. En simulación se construyeron niveles de mina con alturas promedio de 5m, dividiendo el espacio de trabajo e identificando restricciones de borde del mapa que evitan alcanzar el punto objetivo en desniveles, permitiendo la exploración solo en zonas navegables. Marcas que delimitan cada nivel se incluyen en el mapa como restricciones límite para estructurar el ambiente y evitar que el camino solución se dirija al punto fin sin haber atravesado el espacio navegable. Se incorporaron rampas y parterres equidistantes que dividen el espacio de trabajo, distinguiendo la dirección de vía y desniveles. Además, se inserta un punto de no retorno para evitar que el algoritmo decida una sola dirección de exploración, generando una solución que evita zonas de descenso súbito.

3.2. Estrategia de Q-learning para planificación de camino

El algoritmo de planificación de caminos propuesto se basa en Q-learning (QL), el cual utiliza principios del aprendizaje por refuerzo (Fakhrudin et al., 2022). QL modela el sistema dinámico del robot como un proceso de decisión de Markov (MDP), caracterizado por la relación $s' = f(s, a)$, donde s es el estado actual del sistema, representado por la posición del robot en el mapa, y $a = l(s)$ es una ley de acción determinada recursivamente en base al estado s y su evolución s'. Para diseñar el agente de aprendizaje, basado en la evolución de las dinámicas del MDP, se define una recompensa acumulada $r' = \rho(s, a)$ descrita de la siguiente forma:

$$\sum \gamma^k r' = \sum \gamma^k \rho(s, l(s)) \tag{1}$$

donde $0 < \gamma < 1$ es un factor de descuento que busca penalizar las recompensas futuras basandose en las transiciones de posiciones del robot entre espacios ocupados o libres dentro del mapa. La relación (1) representa las recompensas acumuladas y descontadas en el futuro a partir del estado actual s y después de la aplicación de una política $l(s)$. A partir de (1) y el principio de Bellman (Yin et al., 2022), es posible definir una función de valor-acción para la matriz Q de la siguiente forma:

$$Q^{\pi}(s_0, a_0) = \sum_{k=0} \gamma^k \rho(s, l(s)) \tag{2}$$

donde operando se tiene que:

$$Q^{\pi}(s_{k+1}, a_k) = \rho(s_0, a_0) + \gamma \sum_{k=1} \gamma^k \rho(s_{k+1}, l(s_{k+1})) \tag{3}$$

La función de valor (3) representa una matriz de posibilidades en que un estado del sistema evoluciona de una posición inicial s_0 a una siguiente de acuerdo a una acción a_0, siguiendo la política de acción de control l. Para asegurar la búsqueda de un camino a partir de cualquier estado inicial dentro del espacio de trabajo, cada fase de aprendizaje considera $\forall s_0 \in \mathcal{S}$. La acción del estado que maximiza la función de recompensa futura en (3) está dada por $l^*(s_{k+1}) = \gamma \max Q^{\pi,*}(s_{k+1}, a_k)$, resultando en una nueva función acción-valor definida iterativamente por $Q^{\pi,*}(s_{k+1}, a_k)$. Entonces, recursivamente, la función optimizada de acción-valor (3) para la recompensa futura es:

$$Q^{\pi,*}(s_{k+1}, a_k) = \rho(s_k, a_k) + \gamma \max Q^{\pi,*}(f(s_k, a_k), v_k) \tag{4}$$

La relación en (4) ilustra el principio de optimalidad del algoritmo QL, indicando que las acciones óptimas futuras dependen solo del estado actual s_k (por ejemplo, la posición actual del robot). Esta relación se resuelve iterativamente hacia adelante (5) para mejorar la eficiencia computacional y la convergencia de la función de acción-valor $Q^{\pi,*}$. La acción a_k se calcula como la suma de una recompensa inmediata y el valor óptimo ponderado por el factor de descuento γ, basado en la mejor acción tomada en el estado previo. La función de optimización de Q, que maximiza las recompensas, se actualiza iterativamente en el sistema robótico utilizando la tasa de aprendizaje $\alpha \in [0, 1]$, sin necesidad de cálculos hacia atrás ni de un modelo matemático del sistema. Entonces Q es dada por:

$$\begin{aligned} &Q_{i+1}(s_k, a_k) = \\ &Q_i(s_k, a_k) + \alpha[\rho(s_{k+1}, a_k) + \gamma \max Q_i(s_{k+1}, v_k) - Q_i(s_k, a_k)] \end{aligned} \tag{5}$$

donde α hace referencia a la tasa de aprendizaje que permite ajustar la rapidez de convergencia con la que se optimiza Q^{π}. El factor de descuento γ pondera las recompensas pasadas a lo largo del recorrido. Los parámetros α y γ se seleccionan de manera que en conjunto generan el balance entre maximizar la recompensa acumulada y alcanzar el menor tiempo de convergencia en el algoritmo. Haciendo distinción con enfoques tradicionales en que se usan funciones univariadas (Fakhrudin et al., 2022; Wang et al., 2022), en este trabajo se utiliza una función multivariable de par valor-acción Q para el planificador de caminos. La función Q corresponde a una matriz de posibles posiciones libres u ocupadas por obstáculos, siendo un punto dentro del mapa la posición del robot dentro de la matriz Q y sus elementos circundantes las posiciones que posibilitan alcanzar el punto objetivo. Los valores de acciones de control son generados pseudoaleatoriamente para permitir la exploración y explotación de la información adquirida en la navegación. En particular, se ha establecido que los valores de la función objetivo sean representados por el máximo valor de Q, mientras que los demás valores de esta función disminuyen gradualmente respecto del máximo a medida que se alejan de la posición objetivo, garantizando de esta forma la direccionalidad hacia el punto fin.

3.2.1. Política de control

La política que define las acciones de control permite planificar el camino mediante ocho posibles estados, los cuales se identifican como las posibles direcciones en las cuales el robot puede navegar a partir de un punto pivote dentro del mapa. Los ocho posibles estados son i) posición superior, ii) posición

inferior, iii) posición lateral izquierda, iv) posición lateral derecha, y v) las cuatro posibles diagonales. En cada iteración, el algoritmo toma la decisión sobre la acción más apropiada de acuerdo a estos posibles estados mediante la utilización de los valores Q generados previamente. Para mantener el balance entre la exploración del entorno y la explotación de la información, se propone la siguiente política de control π:

$$\pi(s_k) = \begin{cases} \arg\max_{a \in \mathcal{A}} Q(s_k, v_k), & \text{if } 0 \leq \mu \leq \epsilon \\ \text{rand}(a_k), & \text{if } \epsilon \leq \mu \leq 1 \end{cases} \quad (6)$$

donde μ es un valor aleatorio generado en el rango de $[0, 1]$ por episodio de entrenamiento; ϵ es la tasa de exploración, y $rand(a)$ es un valor aleatorio seleccionado de entre todas las posibles acciones $\mathcal{A}$. Dependiendo del estado actual y si el valor de ϵ es mayor que el número aleatorio, la próxima acción es aquella que optimiza los valores en una tabla Q construida por pares estado-acción.

Para evitar la sobre-exploración del mapa, la sobre-explotación de la información o una transición abrupta de la exploración a la explotación desde un valor inicial prescrito ϵ_0, se considera que ϵ varía de manera decreciente de la siguiente manera:

$$\epsilon_{k+1} = \epsilon_k + (\epsilon_0 - \epsilon_f)e^{-k\beta} \quad (7)$$

donde β es una constante que controla la velocidad de decaimiento de la exploración, permitiendo avanzar a la explotación.

3.2.2. *Recompensa de aprendizaje*

Para lograr el aprendizaje, se plantean reglas de recompensas asociadas al alcance del punto objetivo y penalizaciones basadas en riesgos de navegación (Fakhrudin et al., 2022). Se considera la máxima recompensa al lograr el objetivo cuando el robot llega a una vecindad B_{fin} alrededor del punto fin P_{fin} sin retornos inesperados. También se recompensa el avance con velocidad lineal v sin encontrar obstáculos y la exploración en dirección al punto objetivo manteniendo la orientación θ en B_{e_θ}. Se penalizan los giros inesperados que exceden un rango de rapidez angular y cuando la posición del robot está cerca de un obstáculo. Las acciones se penalizan con distintos niveles según la zona de derrape B_{slip} y la prohibición de navegabilidad B_{obs}. La función de recompensas y penalizaciones se plantea de la siguiente forma:

$$\rho = \begin{cases} -\kappa_d \times \left[d_{actual} + \frac{\kappa_b}{\kappa_d + d_{obs}} \right]^2, & \text{Si } P_{actual} \notin B_{fin} \\ -\kappa_\theta \times e_\theta, & \text{Si } e_\theta \notin B_{e_\theta} \\ -\kappa_\omega \times \omega^2,\ \kappa_v \times v^2 & \text{Si } \omega \neq 0,\ \ v \neq 0 \\ -k_g, & \text{Si } P_{actual} \in B_{slip} \\ 0, & \text{en otro caso} \end{cases} \quad (8)$$

Los espacios de vecindad y zonas donde ocurren las penalizaciones en Eq. 8 se detallan a continuación. El espacio de la posición actual del robot, P_{actual}, está dado por $P_{actual} \in \mathbb{R}^2$, representando las coordenadas (x, y) en el plano del mapa. El espacio del área final, B_{fin}, se define como $B_{fin} = \{(x, y) \in \mathbb{R}^2 \mid \sqrt{(x - x_f)^2 + (y - y_f)^2} \leq \epsilon_f\}$, donde (x_f, y_f) es el punto fin del camino planificado y ϵ_f es el radio de la región objetivo. El espacio del error de orientación aceptable, B_{e_θ}, está dado por $B_{e_\theta} = \{e_\theta \in \mathbb{R} \mid \theta_{\min} \leq e_\theta \leq \theta_{\max}\}$, donde e_θ es la desviación angular permitida entre la orientación actual del robot y la dirección del camino, con límites $[\theta_{\min}, \theta_{\max}]$. Finalmente, el espacio de las zonas de deslizamiento, B_{slip}, se define como $B_{slip} = \{(x, y) \in \mathbb{R}^2 \mid |v - \omega r| > \epsilon\}$ está en una región propensa a deslizamiento cuando la velocidad lineal de la rueda ωr difiere de la velocidad lineal del chasis v, identificando así regiones del mapa con alta probabilidad de deslizamiento que son comunes en minas debido a condiciones de interacción con el terreno.

Las constantes k se priorizan de acuerdo a:

- Distancia d_{actual} entre el punto actual y objetivo es recompensada para que tenga alta prioridad con $k_d = 10000$.
- Distancia d_{obs} entre los puntos con posibilidad de colisión respecto del robot en el mapa, es penalizada con $k_b \in [0, 1]$.
- Error de orientación del móvil e_θ respecto a su ángulo θ y el segmento comprendido entre punto actual y fin, es penalizado con $k_\theta \in [0, 1]$. De manera similar, las constantes k_ω y k_v están en el rango $[0, 1]$ y $[0, 10]$, respectivamente.
- Posiciones actuales del robot P_{actual} en sitios resbaladizos por el terreno B_{slip}, es penalizado con $k_g \in [10, 1000]$.
- Acercamiento a la meta: la penalización es adaptable a la cercanía o lejanía del objetivo. En caso de acercarse, se aplica una penalización $k_b \in [0, 0{,}01]$, mientras que en caso de que el robot se aleja se considera una penalización en un rango de $[0, 10]$.

Los parámetros de la función de recompensa ρ y los del modelo de predicción se seleccionaron utilizando métodos heurísticos (Nippun and Kochuvila, 2023), abordando la evaluación de la calidad de respuesta del planificador y el aprendizaje estable (i.e., mediante la recompensa acumulada y promedio móvil) a lo largo del proceso de exploración y explotación. En particular, para los parámetros de la función de recompensa, dado que el proceso de planificación puede inicialmente aislarse de la solución que integra la evasión de obstáculos, se inicia el ajuste de los parámetros k_d, k_ω, k_θ, y k_v desde un punto de inicio hasta un punto fin sin obstáculos. A partir de esta primera estimación, los mismos parámetros se ajustan seleccionando otros puntos de inicio y fin de tal forma que todo el espacio de búsqueda sea explorado. Luego, se consideran los obstáculos estructurales y no estructurales en la escena para ajustar k_b y k_g, asegurándose de mantener el modelo de aprendizaje para el entrenamiento.

La primera línea del Algoritmo 1 inicializa las variables de tasa de aprendizaje, descuento, exploración y constantes de recompensas. El mapa topográfico del ambiente es preparado como parte de los estados en la línea 2. Las posiciones iniciales, finales y vecindades en las zonas de llegada y de slip son también fijadas antes del entrenamiento conforme a la línea 3. En particular, los puntos de prueba durante el entrenamiento son aleatorios para garantizar la exploración fuera de una zona fija y por completitud de búsqueda en todo el espacio de trabajo. La búsqueda permite aquí incorporar elementos imprevistos dentro de la planificación que no han sido considerados durante el pre-procesamiento del mapa topográfico. La matriz Q es inicializada en la línea 5 con estados conformados por el mapa M y la posición del robot. En líneas 6-7, se obtienen los estados inicial s_0 y final s_f acorde a la ubicación en M. En líneas 8-21,

se ejecuta un ciclo repetitivo de aprendizaje para un máximo de episodios ζ. Dentro del ciclo de aprendizaje, entre líneas 10-19,

Algorithm 1 Q-Learning (QL) para la planificación de caminos

1: Inicializar $\alpha, \gamma, \epsilon_0$, y constantes para ρ en (8)
2: Renderizar mapa M del ambiente
3: Inicializar posiciones (x_0, y_0), (x_f, y_f), y vecindades B_{fin}, $B_{slip} \in M$
4: Inicializar pasos de entrenamiento p
5: Inicializar matriz $Q \leftarrow$ valores_Q(M, x, y)
6: Inicializar estado $s_0 \leftarrow$ mapeo_pos(M, x_0, y_0)
7: Inicializar estado $s_f \leftarrow$ mapeo_pos(M, x_f, y_f)
8: **for** $\xi \leftarrow 1$ **to** $\xi_{\max}$ **do**
9: $s \leftarrow s_0$
10: **for** $p \leftarrow 1$ **to** $p_{\max}$ **do**
11: $a \leftarrow$ seleccion_accion(s, Q, ϵ)
12: $[x_{QL}(p),\ y_{QL}(p),\ \rho] \leftarrow$ aplica_accion(a, s, M, Q, B_{slip})
13: $s' \leftarrow$ extraer_estado(M, x_{QL}, y_{QL})
14: $Q \leftarrow$ actualizar_Q($a, s, \rho, s', Q, \epsilon$) basándose en (5)
15: $s \leftarrow s'$
16: **if** $s \in B_{fin}$ **then**
17: **break**
18: **end if**
19: **end for**
20: $\epsilon \leftarrow \epsilon$ from (7)
21: **end for**
22: **if** $s \notin B_{fin}$ **then**
23: $[Q, x_{QL}, y_{QL}] \leftarrow$ valor_accion($a, \gamma, \xi_{\max}, \epsilon, M, x_0, y_0, x_f, y_f$)
24: Regresa a línea 8
25: **end if**
26: **return** $Q,\ L_{QL} = [x_{QL}, y_{QL}]^T$

Finalmente, se ajusta el parámetro k_g para equilibrar la alcanzabilidad del punto fin y la evasión de zonas resbaladizas. Las recompensas y penalizaciones se seleccionan para lograr caminos suaves y cortos, aunque otras reglas podrían cambiar el desempeño del planificador. Las reglas y valores indicados permiten un camino alcanzable, admisible y completo para un robot uniciclo en un ambiente tipo mina con condiciones terra-mecánicas hostiles. En línea 11, se elige la acción a que maximiza Q y se actualiza para continuar el aprendizaje. En líneas 12-14, se actualizan los valores de Q basados en la recompensa según la función (5). Si el punto evaluado no está en la vecindad del punto fin B_{fin}, la búsqueda continúa actualizando Q. El criterio de parada se establece en línea 16, y línea 22 establece una condición de reintento al no alcanzar la vecindad del punto fin. Finalmente, en la línea 26, se obtiene la matriz Q y el camino resultante $L_{QL} = [x_{QL}, y_{QL}]^T$.

*3.3. Técnica de Q-learning informada por RRT**

Dada la complejidad para estimar un camino dependiente de la configuración del entorno, surge la necesidad de generar muestras factibles de posiciones de robot con información espacial que permitan encontrar la solución. Se propone obtener puntos potenciales que guíen la búsqueda de QL, adaptando el algoritmo RRT* (Rapidly-exploring Random Trees) de (Chen et al., 2021).

A pesar de la compatibilidad de RRT* en el muestreo y exploración del entorno, sus principales fortalezas son su capacidad para explorar conexiones entre nodos en un mapa de alta dimensionalidad y adaptarse a obstáculos inciertos del terreno. RRT* permite, en cada iteración, conectar puntos con alta probabilidad de alcanzar el destino en un espacio libre de colisión del robot y expandirse desde el nodo más cercano al objetivo. Para explorar nuevos puntos, se utiliza una función de selección que direcciona hacia el punto fin en una región libre. El desplazamiento considera una ruta directa hacia el nodo más cercano sin colisiones. En caso de riesgo de colisión, se generan nuevos

Algorithm 2 Q-Learning Muestreado por RRT* - MQL

1: Inicializar $\alpha,\ \gamma,\ \epsilon_0,\ Q$, nodos, y constantes para ρ en (8)
2: Inicializar distancia de conexión d_{RRT} en nodos RRT*
3: Renderizar mapa M del ambiente
4: Inicializar posiciones (x_0, y_0), (x_f, y_f), y vecindades B_{fin}, $B_{slip} \in M$
5: Inicializar número de muestras filtradas N y pasos de entrenamiento p
6: Inicializar nodos en punto origen y ponderaciones η_x y η_y
7: **for** $\xi \leftarrow 1$ **to** $\xi_{\max}$ **do**
8: $s \leftarrow s_0$
9: **for** $p \leftarrow 1$ **to** $p_{\max}$ **do**
10: $a \leftarrow$ seleccion_accion(s, Q, ϵ)
11: $[x_{QL}(p),\ y_{QL}(p),\ \rho] \leftarrow$ aplica_accion($a, s, M, Q, B_{slip}, nodos$)
12: $s' \leftarrow$ extraer_estado($M, x_{\mathrm{QL}}, y_{\mathrm{QL}}$)
13: $Q \leftarrow$ actualizar_Q($a, s, \rho, s', Q, \epsilon$) basándose en (5)
14: $s \leftarrow s'$
15: **if** $s \in B_{fin}$ **then**
16: **break**
17: **end if**
18: $\epsilon \leftarrow \epsilon$ from (7)
19: **if** $s \notin B_{fin}$ **then** (Procedimiento paralelizable)
20: $[\lambda, nodos] \leftarrow$ RRT*(x_0, y_0, x_f, y_f, M) (Chen et al., 2021);
21: (λ es dirección del nodo)
22: $[Q, x_{QL}, y_{QL}] \leftarrow$ valor_accion($\lambda, \gamma, \xi, \epsilon, M, x_0, y_0, x_f, y_f$)
23: $nodos\_vecinos \leftarrow$ busqueda_vecindad(nodos, x_{QL}, y_{QL}, s_f)
24: $d_{RRT} \leftarrow$ minima_distancia($nodos\_vecinos, s_f$)
25: **end if**
26: $[x_{\mathrm{MQL}}, y_{\mathrm{MQL}}] \leftarrow$ cercania($nodos, d_{RRT}, s_0, x_{\mathrm{QL}}, y_{\mathrm{QL}}$)
27: $[x_{\mathrm{SMQL}}, y_{\mathrm{SMQL}}] \leftarrow$ filtro_Gauss(x_{MQL}, y_{MQL}, N) usando (9)
28: $nodos = nodos\_vecinos$
29: Regresa a línea 6
30: **end for**
31: **end for**
32: **Return** $L_{MQL} = [x_{\mathrm{MQL}}, y_{\mathrm{MQL}}]^T,\ L_{SMQL}[x_{\mathrm{SMQL}}, y_{\mathrm{SMQL}}]^T$

puntos para encontrar la mejor ruta al destino. Si no hay solución física o la búsqueda se queda entrampada, la muestra no se considera válida para QL. Una vez alcanzado el destino, se conectan los puntos en el árbol para evaluar la ruta más corta y rápida desde el inicio. Estas muestras se usan en lugar de puntos aleatorios durante el entrenamiento de QL.

Para acoplar la ruta generada con muestras de RRT*, el espacio de búsqueda de QL es integrado con los nodos de exploración dentro del espacio de trabajo del robot. Además, dado que QL realiza iteraciones basadas en episodios y actualizaciones de la función de valor Q, el algoritmo RRT* realiza iteraciones por cada época de entrenamiento para expandir el árbol y optimizar el camino. En particular, la sincronización entre las iteraciones de RRT* y el entrenamiento en QL consiste en ejecutar alternadamente los dos algoritmos, donde cada iteración de RRT* se utiliza solo para mejorar las acciones de QL, y no viceversa. En la práctica, dado que información del mapa es compartido entre RRT* y QL, el algoritmo RRT* puede ser ejecutado paralelamente al algoritmo QL, evitando carga computacional significativa. Por otro lado, dado que se requiere equilibrar la exploración del entorno con RRT*, se priorizó la búsqueda de RRT* por su principio basado en árboles de decisión y se priorizó la explotación de QL por su capacidad de aprovechar la información del aprendizaje previo.

3.3.1. Filtro Gaussiano para suavizado de camino

Debido a la naturaleza exploratoria de los algoritmos QL y RRT*, la ruta resultante L_{MQL} puede ser irregular y no cumplir

con restricciones mínimas de navegabilidad, como la geometría del camino, perfiles de velocidad y radios de giro adecuados para la rotación del robot uniciclo. Por lo tanto, es crucial aplicar una técnica de suavizado antes de la parametrización para asegurar que la cinemática del robot sea compatible con la ruta planificada. El método de suavizado emplea un filtro Gaussiano que pondera las últimas N posiciones del camino, penalizando puntos con alta respuesta transitoria y preservando aquellos con baja respuesta durante el estado permanente. El tamaño N de las muestras a filtrar se selecciona basado en pruebas de rendimiento del planificador, donde se garantiza la evasión de obstáculos y la adecuada cinemática del robot. Además, el filtro controla la curvatura al ponderar los desplazamientos lateral y longitudinal al generar muestras evaluando al menos N puntos consecutivos, de acuerdo con:

$$x_{SMQL,j} = \frac{\sum_{i=1}^{N} w_i(y_j) x_{MQL,i}}{\sum_{i=1}^{N} w_i(y_j)}, \text{ donde } w_i(y_j) = e^{-\eta_y (y_{MQL} - y_{MQL,i})^2}$$
$$y_{SMQL,j} = \frac{\sum_{i=1}^{N} w_i(x_j) y_{MQL,i}}{\sum_{i=1}^{N} w_i(x_j)}, \text{ donde } w_i(x_j) = e^{-\eta_x (x_{MQL} - x_{MQL,i})^2} \quad (9)$$

donde w_i pondera la distancia entre los puntos del camino aprendido y sus nodos vecinos determinados por la mínima distancia d_{RRT}. Los puntos suavizados $L_{SMQL} = [x_{SMQL}, y_{SMQL}]^T$ en (9) indican que la coordenada vertical $y_{SMQL,j}$ se calcula como un promedio ponderado de las coordenadas verticales originales, utilizando la función de peso Gaussiana $w_i(x_{MQL,j})$ evaluada en el punto j. Este enfoque suaviza el camino considerando la variabilidad de los puntos en ambos ejes.

El Algoritmo 2 combina *RRT** y QL para obtener el camino muestreado L_{MQL}, mientras que el cálculo del camino suavizado se obtiene de (9). El Algoritmo 2 aprovecha la información provista por el método RRT* para proveer información a QL, integrando los datos sub-muestreados del camino pre-planificado, i.e., nodos de conexión. Brevemente, similar al Algoritmo 1, en líneas de código 1-6 se inicializan los parámetros para RRT* y QL. En particular, la inicialización comprende la renderización del mapa M con sus estados iniciales y finales, se fijan nodos iniciales de RRT*, se ajusta la cantidad de muestras a filtrar y se configuran las ponderaciones del filtro Gaussiano. Luego, líneas 7-31 implementan episódicamente el método híbrido de QL informado por RRT*, donde la acción calculada de acuerdo al estado actual y la matriz de valor Q es aplicada al sistema para definir el siguiente estado (ver líneas 10-13). En este punto es de resaltar que QL se alimenta de nodos extraídos previamente de RRT* previo al análisis de cercanía con puntos de QL.

En línea 13 se actualiza la matriz de valor Q con la nueva posición probable del robot en el ambiente, y se actualiza la posición en línea 14. Si se alcanza el punto fin, se repite el entrenamiento para un nuevo episodio ξ (ver línea 9). Sin embargo, si el punto fin no ha sido alcanzado dentro de la vecindad B_{fin}, líneas 19-25 de RRT* se ejecutan en paralelo al algoritmo QL. Línea 20 indica el cálculo de nodos del algoritmo RRT* que garantiza la mejor conexión posible de un conjunto de soluciones de caminos dentro de un árbol de decisión. Además, se entrega la dirección λ de los nodos que apuntan al punto objetivo. El algoritmo seguidamente identifica en cada episodio la cercanía entre nodos que describen los puntos del camino y los puntos de QL $[x_{MQ}, y_{MQ}]$ para ser seleccionados en conjunto con su dirección (ver líneas 24-26). Finalmente, el algoritmo filtra los puntos del camino solución de QL muestreado (L_{MQL}), derivando en un camino suave compuesto por $[L_{SMQL} = x_{SMQ}, y_{SMQ}]$ y sigue entrenando episódicamente (ver líneas 29-32).

Tabla 1: Parámetros para entrenamiento de los planificadores

Parámetro / Técnica	QL	MQL	SMQL
Tasa de aprendizaje: α	0.7	0.7	0.7
Factor de descuento: γ	0.95	0.95	0.95
Tasa de exploración inicial: ϵ_0	1	1	1
Tasa de exploración final: ϵ_f	0.001	0.001	0.001
Constante de decaimiento: β	0.005	0.005	0.005
Total de episodios	500	500	500
Ditancia conexión: d_{RRT}	-	0.001	0.001
Ponderaciones de filtro: $\eta_x = \eta_y$	-	-	0.5

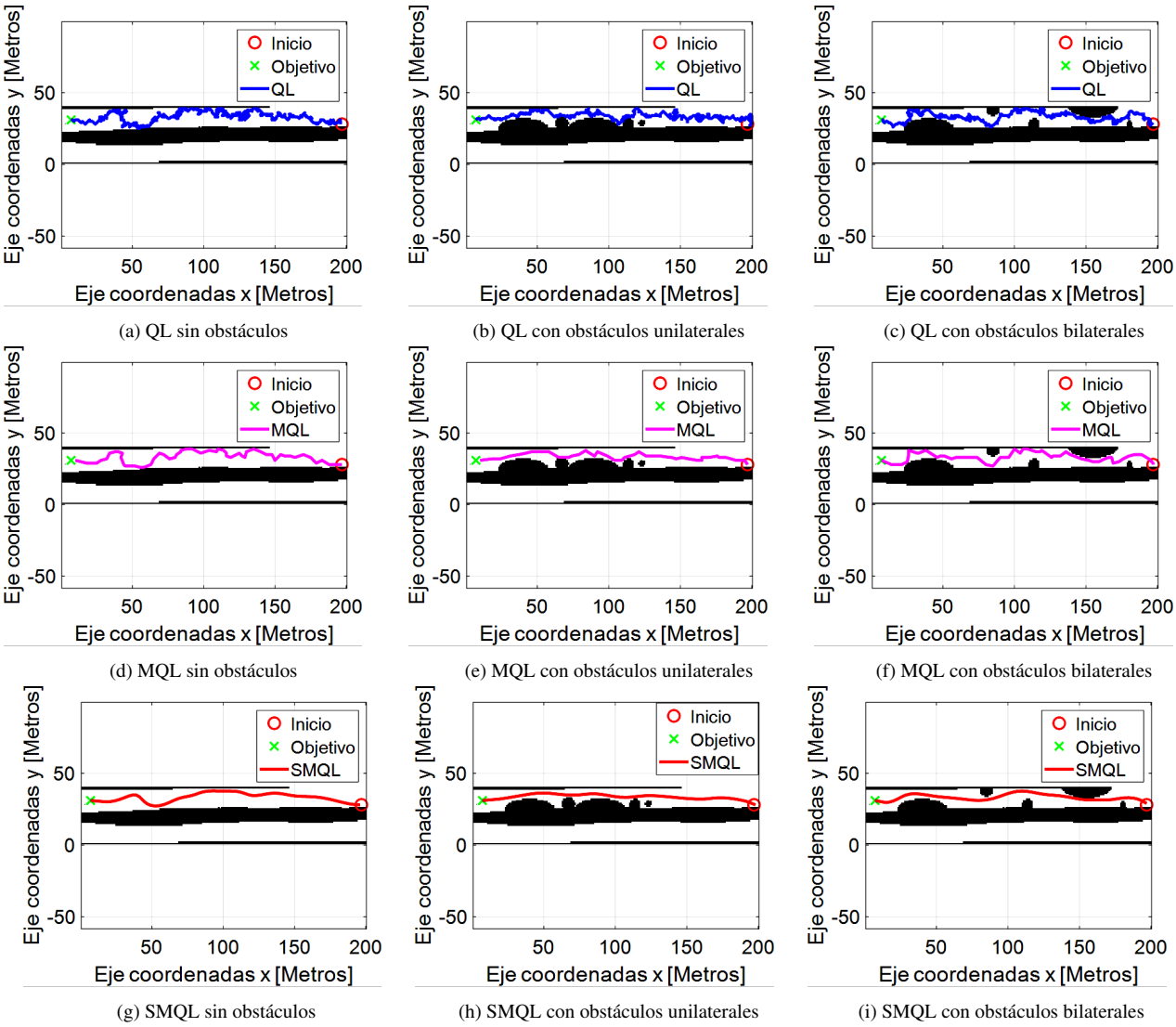
4. Resultados de pruebas experimentales y análisis

En este estudio, se llevaron a cabo dos pruebas utilizando los planificadores de caminos diseñados, donde se evaluaron tres variantes del algoritmo: a) Q-Learning (QL), b) QL con información a priori de muestras RRT* (MQL), y c) suavización del camino planificado MQL (SMQL). La primera prueba consistió en analizar el desempeño de los planificadores, variando la configuración en el espacio de trabajo del robot. Mientras tanto, la segunda prueba consistió en evaluar un planificador de caminos diseñado en un seguidor de trayectorias luego de haber sido parametrizada la ruta pre-planificada. En lo que respecta a la primera prueba, se consideraron cambios estructurales del entorno y diferentes condiciones de exploración. Para lograr una representación cercana a los entornos de minas a cielo abierto, se construyó un ambiente de entrenamiento con restricciones de terreno resbaladizas (ver Figura 3). De acuerdo a la configuración de los ambientes de prueba, la caminos resultantes son: i) caminos rectos con obstáculos para prueba de maniobrabilidad, ii) caminos intrincados y orientados al objetivo en un espacio de trabajo tipo laberinto para prueba de alcanzabilidad y completitud, y iii) caminos helicoidales en ambientes a desnivel y restricciones de terreno, conforme a un mapa similar al de faenas mineras.

Para evitar el sobreajuste en el aprendizaje de navegación, se implementaron varias estrategias:

- Variación de puntos inicio-fin: durante el entrenamiento, se modificaron los puntos inicio y fin entre diversas ubicaciones dentro del ambiente, incluyendo zonas de deslizamiento. Esto aseguró una exploración completa, previniendo el sobreajuste para un par de puntos fijos.
- Entrenamiento progresivo: se llevó a cabo un entrenamiento gradual comenzando desde la tarea de seguimiento de caminos libres de obstáculos. Se incrementó progresivamente la complejidad del entorno al introducir obstáculos estructurales y áreas resbaladizas. Esta metodología evitó el sobreajuste inicial y mejoró la capacidad del modelo.

Los parámetros del modelo de planificación se determinaron por otro lado mediante evaluación heurística del desempeño del planificador y su estabilidad (Nippun and Kochuvila, 2023), considerando además que la rapidez de convergencia de la política sea en un tiempo razonable. En general, los valores



(a) QL sin obstáculos (b) QL con obstáculos unilaterales (c) QL con obstáculos bilaterales

(d) MQL sin obstáculos (e) MQL con obstáculos unilaterales (f) MQL con obstáculos bilaterales

(g) SMQL sin obstáculos (h) SMQL con obstáculos unilaterales (i) SMQL con obstáculos bilaterales

Figura 4: Resultados de planificación de caminos usando QL, MQL y SMQL en un espacio estrecho con diferentes configuraciones de navegabilidad y obstáculos. Las Figuras (a)-(c) muestran los caminos generados por QL, (d)-(f) por MQL, y (g)-(i) por SMQL. SMQL produce caminos suaves y maniobrables tanto en áreas con obstáculos como sin ellos, demostrando eficiencia en navegación comparado con QL y MQL. Sin embargo, QL y MQL también logran caminos alcanzables dentro de las restricciones del espacio navegable.

de los parámetros se establecieron de manera que fueran suficientemente altos para evitar una recompensa acumulada oscilante, pero no tan bajos como para ralentizar el aprendizaje. Los parámetros para los tres algoritmos de aprendizaje se indican en Tabla 1. Adicionalmente, en esta prueba se presentan resultados de convergencia de recompensas de los algoritmos de aprendizaje propuestos.

La tarea de ensamblar modelos fue descartada en este trabajo debido a la diversidad de las estructuras de los ambientes. La combinación de modelos de diferentes entornos no mejora la generalización; de hecho, resulta en sobreajuste al cambiar puntos de inicio y fin, o al incluir otras zonas de deslizamiento en la topografía del ambiente. La segunda prueba consiste en validar el camino previamente planificado en un controlador de movimiento para seguimiento de trayectorias en el robot de la Figura 1, tomando como referencia el controlador y su ajuste de (Cheein et al., 2017; Prado et al., 2018a). El objetivo de la prueba es evaluar cualitativa y numéricamente el desempeño del planificador de caminos en un marco de control de seguimiento de trayectorias. Su validez se cuantifica mediante métricas que determinan el rendimiento del controlador en función de la referencia de la trayectoria parametrizada con un perfil de velocidad uniforme para todos los casos de prueba.

Las simulaciones y pruebas del planificador de caminos se realizaron en CoppeliaSim, usando MATLAB™ (versión R2023b) de MathWorks, Inc©. En la práctica, las experimentaciones de campo se llevaron a cabo en el Centro de Investigación Avanzada en Ingeniería Eléctrica y Electrónica AC3E, UTFSM, Chile, con un robot monociclo Husky A200 equipado con un computador portátil Intel Core™ i7-10750H @ 2.6 GHz y 16GB de RAM. Una red de comunicación local permite adquirir datos sincronizados de sensores y transmitir acciones al robot. Además de los sensores de visión, el robot se instrumentó con DGPS en modo RTK-GNSS (SF-3040), IMU (VN-200) y encoders internos para posicionamiento y orientación. Para simular condiciones reales de un vehículo autónomo, se utilizó una rapidez constante de $v = 0{,}7$ m/s, acorde a las velocidades en la industria minera. A continuación, se presentan los resultados obtenidos durante las pruebas.

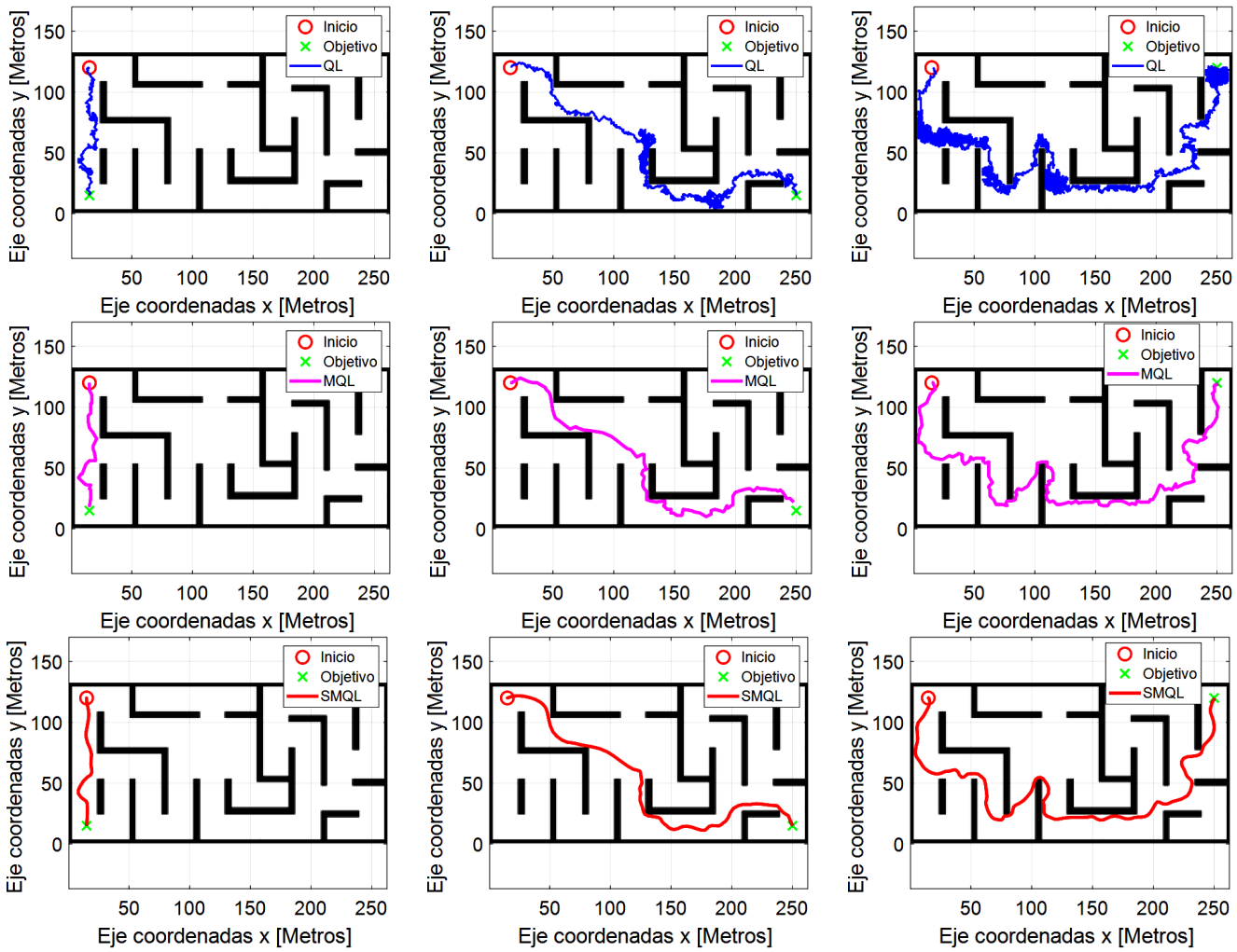


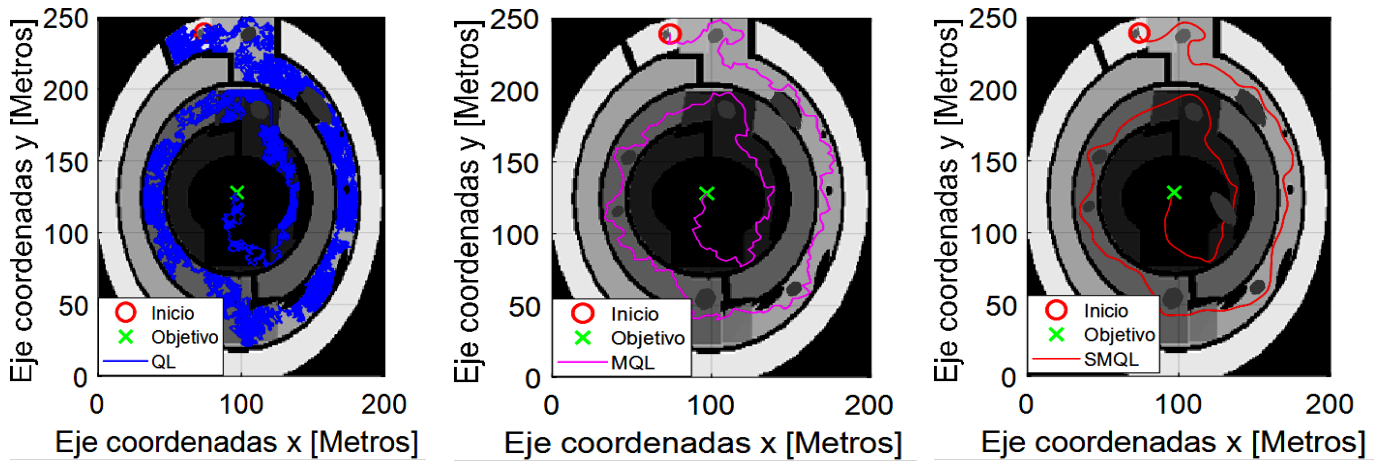
Figura 5: Resultados obtenidos por el planificador de caminos en un entorno intrincado tipo laberinto. Cada fila muestra la implementación de las tres versiones de los algoritmos en estudio (QL, MQL, SMQL) para distintas posiciones inicio y fin en el entorno. Cada columna presenta los caminos planificados con los diferentes cambios de posición del punto inicio y fin. Se observa que los caminos resultantes de SMQL son completos, alcanzables y particularmente más suaves en comparación con QL y MQL. Además, es relevante indicar que SMQL, con el filtro Gaussiano, no genera condiciones propensas a colisiones.

4.1. Prueba de maniobrabilidad: camino recto

La prueba evalúa la maniobrabilidad de caminos planificados bajo diversas configuraciones de obstáculos en un espacio de trabajo. Se analizan tres variantes: escenario sin obstáculos, obstáculos en un lado del camino, y obstáculos en ambos lados del robot. La Figura 4 muestra los resultados de los planificadores QL, MQL y SMQL, indicando que todos los caminos son navegables en un área de prueba de aproximadamente 200 m de largo por 40 m de ancho. Sin embargo, el planificador QL presenta mayor variabilidad en los caminos, potencialmente conduciendo a condiciones de colisión en espacios estrechos. Tanto QL como MQL generan caminos que pueden aún no ser cinemáticacamente no compatibles con el robot diferencial debido a la exploración, pero son factibles. MQL muestra una reducción parcial en el comportamiento oscilatorio comparado con QL. Por otro lado, SMQL exhibe una mejor capacidad para maniobrar alrededor de obstáculos en comparación con QL y MQL, a pesar de que el suavizado del camino puede afectar secciones de navegabilidad.

4.2. Prueba de alcanzabilidad: entorno intrincado

Esta prueba tiene como objetivo verificar la alcanzabilidad y cobertura del camino ante cambios en el punto de inicio y final, así como la irregularidad estructural de espacios intrincados. En este caso, el espacio de búsqueda se complica debido a las diversas soluciones que pueden encontrarse y a la exigencia de flexibilidad geométrica que debe cumplirse. La Figura 5 ilustra los resultados obtenidos, donde cada fila representa los resultados de las tres versiones de los algoritmos propuestos, y en cada columna se varían los puntos de exploración. Para los tres casos y métodos, los caminos son completos. Al analizar los resultados de QL, se observa un camino ruidoso y sinuoso que no se orienta estrictamente hacia el punto fin. Este problema puede atribuirse a la prioridad dada a la exploración y a la misma cantidad de muestras predefinidas en todos los algoritmos propuestos. Por ejemplo, en el tercer caso de QL (primera fila y tercera columna de la Figura 5) se observa que, al llegar al punto fin, el algoritmo sigue realizando la búsqueda y genera un comportamiento ruidoso en su vecindad, produciendo un camino inestable y cinemáticamente incompatible con el robot diferencial. A diferencia de QL, MQL llega y permanece en el punto fin, explorando la solución sobre el camino más cercano. Por otro lado, el algoritmo SMQL muestra la misma tendencia y solución que los otros dos planificadores, pero representa un camino visualmente más suave en comparación con QL y MQL.



(a) Planificador QL en camino helicoidal (b) Planificador MQL en camino helicoidal (c) Planificador SMQL en camino helicoidal

Figura 6: Resultados de los planificadores de caminos utilizando los algoritmos QL, MQL y SMQL en un mapa helicoidal, similar al que se encuentra en entornos mineros de faenas a cielo abierto. Por inspección, se observa que el planificador SMQL evade obstáculos estructurales y de terreno para alcanzar el punto objetivo, bordeando el entorno de trabajo

.

Tabla 2: Índices de desempeño del controlador de movimiento obtenidos durante el seguimiento de trayectorias, tomando como referencia los caminos pre planificados dentro de un esquema de control de lazo cerrado.

Entorno	Método (recorrido)	Tiempo (s)	C_{xy}	C_{vw}	C_{Tot}
Entorno recto	QL (608.8 m)	870	756.2	529.2	1285.4
	MQL (231 m)	330	280.8 (62.8 %)	66.9 (87.4 %)	347.7 (72.9 %)
	SMQL (178.5 m)	256	197.7 (73.8 %)	39.7 (94.5 %)	237.4 (81.5 %)
Intrincado	QL (1623 m)	2320	2014.2	1568.0	3582.2
	MQL (352.3 m)	504	413.3 (79.5 %)	150.8 (90.3 %)	564.1 (84.2 %)
	SMQL (308.5 m)	441	375.5 (81.4 %)	168.6 (89.2 %)	544.2 (84.8 %)
Entorno helicoidal	QL (20112 m)	4881.5	1846.1	19446.1	6727.6
	MQL (899 m)	1285	1151.3 (76.4 %)	305.1 (83.5 %)	1526.0 (77.3 %)
	SMQL (708.3 m)	1012	885.9 (81.9 %)	372.3 (79.8 %)	1258.0 (81.3 %)

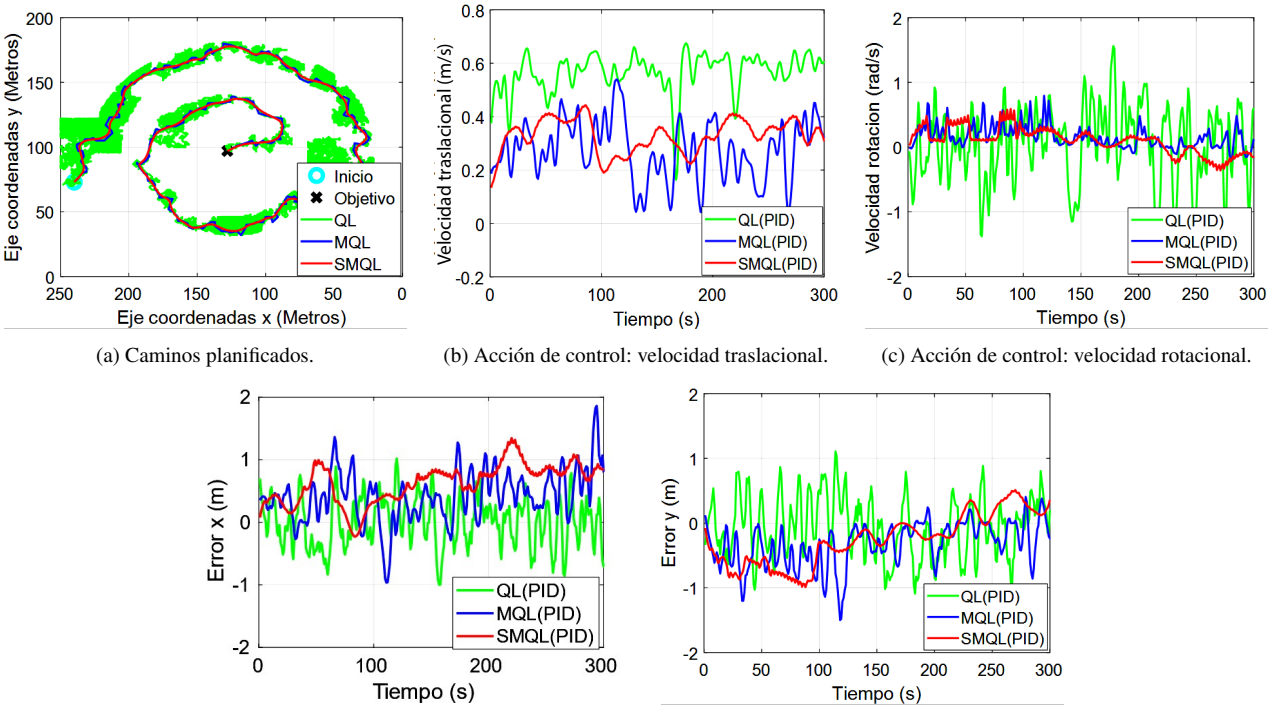
4.3. Prueba de robustez en entorno minero

El objetivo de esta prueba es evaluar la robustez del planificador frente a perturbaciones de terreno en un entorno helicoidal, similar a los encontrados en la minería de cielo abierto. El mayor desafío en esta prueba radica en que el planificador de caminos debe bordear el espacio de trabajo helicoidal a pesar de una mayor longitud de trayecto que el de una línea recta al punto fin, contrario a su principio de diseño de mínima distancia. Sin embargo, el planificador debe aún cumplir que al llegar al punto fin, la distancia total del camino debe ser la menor. La Figura 6 muestra los resultados alcanzados por los algoritmos QL, MQL y SMQL. En los tres casos se observa que la solución está dentro del espacio de trabajo y evita colisionar con las restricciones del mapa y con las condiciones de terreno resbaladizas implantadas en el entorno. Similarmente, los tres planificadores evitan llegar al objetivo en línea recta y cumplen con las restricciones de borde del entorno minero. Sin embargo, se observa que el algoritmo QL genera una solución infactible para un entorno helicoidal dada la variabilidad del camino encontrado. Aquí se visualiza la necesidad de haber complementado el algoritmo de QL con el algoritmo RRT*, mejorando la rapidez de exploración y alcance al punto fin. El problema de infactibilidad en QL se debe a la búsqueda desinformada en ciertos puntos del mapa que podrían formar parte de la solución, pero que no son necesariamente considerados como puntos de mínima distancia. Por otro lado, el camino entregado por el algoritmo SMQL cumple con restricciones de movilidad de tal forma que es cinemáticamente compatible con las dinámicas del robot y evade los obstáculos estructurales del entorno y del terreno resbaladizo, a diferencia de los métodos QL y MQL.

4.4. Seguimiento de trayectoria en campo

Esta prueba valida en campo un camino planificado en un controlador de seguimiento de trayectorias utilizando un PID modificado basado en álgebra lineal según (Cheein et al., 2017), el cual requiere que la trayectoria esté parametrizada. El camino se parametriza aquí con una velocidad constante de $v = 0{,}7$ m/s y un tiempo de muestreo de 0.1 s. Los resultados del controlador para trayectorias helicoidales se presentan en la Figura 7, mientras que la Tabla 2 resume su desempeño en todos los entornos de prueba.

Las trayectorias seguidas por el controlador PID sobre los caminos dados por MQL y SMQL son similares (ver Figura 7), mientras que el obtenido para QL difiere por el grado de oscilaciones que presenta. Los errores en el eje lateral y longitudinal presentan más variación respecto de la referencia con QL que con aquellos caminos con MQL y SMQL. Este mismo resultado se traduce a un impacto en el esfuerzo de control. La velocidad traslacional en la Figura 7 ilustra una tendencia relativamente suave para MQL y SMQL con respecto a QL, reduciendo así el impacto sobre el actuador del robot (ver Figura 7). Adicionalmente, la velocidad de traslacional con QL es mayor y ruidosa con respecto a los otros dos métodos, lo que tiene un impacto en el tiempo de llegada al punto fin. La respuesta de velocidad de



(a) Caminos planificados.

(b) Acción de control: velocidad traslacional.

(c) Acción de control: velocidad rotacional.

(d) Error de seguimiento en eje x.

(e) Error de seguimiento en eje y.

Figura 7: Resultados de campo del controlador de seguimiento de trayectorias de acuerdo a los caminos planificados. La Figura (a) muestra los caminos planificados con QL, MQL y SMQL, parametrizados para su aplicación en el controlador PID y el seguimiento por parte del robot. Las Figuras (b)-(c) presentan las acciones de control generadas por el controlador, mientras que las Figuras (d)-(e) muestran los errores de seguimiento de trayectoria en los ejes longitudinal y lateral al seguir los caminos planificados, respectivamente. Se observa que tanto el esfuerzo de control como los errores de seguimiento de trayectoria son mas bajos cuando se usa un camino planificado con SQML en comparación con QL y MQL.

rotación del robot es atenuada con SMQL, siendo un resultado esperable al filtrar la curva a partir de una solución MQL.

Para una evaluación cuantitativa, se utilizan métricas de desempeño de controladores de seguimiento de trayectorias que incluyen el costo de error de seguimiento de trayectoria C_{xy}, costo de esfuerzo de control C_{vw} y costo total acumulado C_{Tot} (Prado et al., 2018a). Además, se incluye en la Tabla 2 una comparativa entre el recorrido y el tiempo total transcurrido en que el robot cubre el camino planificado. Aquí se observa que el método SMQL logró obtener la longitud de recorrido y tiempo de ruta más bajos en comparación con los otros dos métodos para todos los casos. Por ejemplo, para el entorno minero, SMQL alcanza el punto fin en 708.3 m, mientras que QL y MQL lo hacen en 20112 m y 899 m, respectivamente. Por otro lado, el método MQL exhibe un menor C_{xy} respecto de QL, reduciéndose así en 62.8 %, 79.5 %, 76.4 % por cada prueba relacionada con el entorno recto, intricado y helicoidal, respectivamente. De forma similar, el controlador de movimiento mejora su desempeño utilizando un camino planificado con SMQL respecto de QL, mostrando una reducción de C_{xy} en: 73.8 %, 81.4 %, 81.9 %, con respecto a las pruebas anteriormente mencionadas. El controlador PID con el método SMQL presentó un mejor desempeño en términos de los tres índices en comparación con QL, alcanzando valores de hasta 81.9 % en C_{xy}, 79.8 % para C_{vw}, y 81.3 % del costo total C_{Tot} en el recorrido más largo (entorno minero). Por otro lado, el seguidor de trayectorias con el método SMQL con respecto a MQL presentó un rendimiento similar en términos de C_{xy}, mas no así con C_{vw}, donde MQL presentó mayor esfuerzo de control en la mayoría de los casos, siendo el máximo incremento de 7.1 %. Los resultados resaltan la mejora del desempeño del controlador utilizando el método SMQL en comparación con QL y MQL, dado que existe una mejor exploración del espacio de trabajo y una respuesta rápida que converge a la ruta aprendida.

5. Conclusiones

En este paper se desarrolló una estrategia de planificación de caminos para robots diferenciales en ambientes con terreno deslizante y obstáculos, empleando una técnica híbrida que integra Q-learning (QL) con muestreo aleatorio basado en RRT*. Esta metodología generó una ruta muestreada, MQL, la cual fue posteriormente suavizada para cumplir con las restricciones cinemáticas del robot, resultando en SMQL. La validación del planificador se realizó en tres escenarios diversos para evaluar maniobrabilidad, completitud, alcanzabilidad y robustez del camino planificado. SMQL demostró una reducción significativa en el comportamiento oscilatorio y mejor capacidad de evasión de obstáculos en comparación con QL y MQL. En términos de completitud, todos los métodos alcanzaron el punto objetivo, aunque con ciertas limitaciones de navegabilidad debido a la incertidumbre del entorno. Para la alcanzabilidad, SMQL superó las respuestas inestables observadas en QL y MQL. En un escenario helicoidal representativo de minas a cielo abierto, SMQL mostró una compatibilidad cinemática superior con el robot diferencial, validando su potencial implementación en un controlador de movimiento. En pruebas con un seguidor de trayectorias, SMQL mejoró significativamente el rendimiento del controlador en comparación con QL y MQL, reduciendo el costo de error en un 81.9 % y el esfuerzo de control en un

79.8 %. Se espera que estos resultados contribuyan a la reducción del consumo energético en vehículos autónomos en la industria minera. Como futuras investigaciones se espera integrar las incertidumbres del robot y las condiciones del entorno para fortalecer aún más el desempeño del planificador.

Agradecimientos

Los autores agradecen el apoyo de ANID mediante Proyecto Fondecyt de Iniciación en Investigación 2023, N° 11230962. Se agradece al apoyo al Centro Avanzado de Ingeniería Eléctrica y Electrónica - AC3E, Chile (ANID FB 0008). También se reconoce el apoyo de la Universidad Católica del Norte bajo el proyecto 202203010029 -VRIDT-UCN, y al proyecto Anillo de Investigación en Ciencia y Tecnología -ACT210052.

Referencias

Agrawal, R., Singh, B., Kumar, R., 11 2022. Classical approaches for mobile robot path planning: A review. IEEE, pp. 400–405.
DOI: 10.1109/ICCCIS56430.2022.10037620

Aguilera-Marinovic, S., Torres-Torriti, M., Auat-Cheein, F., 2017. General dynamic model for skid-steer mobile manipulators with wheel–ground interactions. IEEE/ASME Transactions on Mechatronics 22 (1), 433–444.
DOI: 10.1109/TMECH.2016.2601308

Aradi, S., 2 2022. Survey of deep reinforcement learning for motion planning of autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems 23, 740–759.
DOI: 10.1109/TITS.2020.3024655

Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V., Mangharam, R., 2022. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. IEEE Open Journal of Intelligent Transportation Systems 3, 458–488.
DOI: 10.1109/OJITS.2022.3181510

Cheein, F. A., Torres-Torriti, M., Hopfenblatt, N. B., Álvaro Javier Prado, Calabi, D., 12 2017. Agricultural service unit motion planning under harvesting scheduling and terrain constraints. Journal of Field Robotics 34, 1531–1542.
DOI: 10.1002/rob.21738

Chen, J., Zhao, Y., Xu, X., 2021. Improved rrt-connect based path planning algorithm for mobile robots. IEEE Access 9, 145988–145999.
DOI: 10.1109/ACCESS.2021.3123622

Fakhrudin, M., Sutisna, N., Syafalni, I., Adiono, T., 2022. Algorithm and rtl architecture design of q-learning algorithm for path planning. In: 2022 International Symposium on Electronics and Smart Devices (ISESD). pp. 1–6.
DOI: 10.1109/ISESD56103.2022.9980743

Gao, S., D., Y., Chen, B. M., July 2020. A frontier-based coverage path planning algorithm for robot exploration in unknown environment. IEEE, pp. 3920–3925.
DOI: 10.23919/CCC50068.2020.9188784

Geusebroek, J.-M., Smeulders, A., van de Weijer, J., 2003. Fast anisotropic gauss filtering. IEEE Transactions on Image Processing 12 (8), 938–943.
DOI: 10.1109/TIP.2003.812429

Goutham, M., Boyle, S., Menon, M., Mohan, S., Garrow, S., Stockar, S. S., 3 2023. Optimal path planning through a sequence of waypoints. IEEE Robotics and Automation Letters 8, 1509–1514.
DOI: 10.1109/LRA.2023.3240662

Jayaweera, H. M., Hanoun, S., 2020. A dynamic artificial potential field (d-apf) uav path planning technique for following ground moving targets. IEEE Access 8, 192760–192776.
DOI: 10.1109/ACCESS.2020.3032929

Ko, C., Han, S., Choi, M., Kim, K.-S., 10 2020. Integrated path planning and tracking control of autonomous vehicle for collision avoidance based on model predictive control and potential field. IEEE, pp. 956–961.
DOI: 10.23919/ICCAS50221.2020.9268369

Lazányi, K., 1 2023. Perceived risks of autonomous vehicles. Risks 11, 26.
DOI: 10.3390/risks11020026

Luo, J., Wang, Z.-X., Pan, K.-L., 2022. Reliable path planning algorithm based on improved artificial potential field method. IEEE Access 10, 108276–108284.
DOI: 10.1109/ACCESS.2022.3212741

Nippun, K., Kochuvila, S., 2023. Mobile service robot path planning using deep reinforcement learning. IEEE Access 11, 100083–100096.
DOI: 10.1109/ACCESS.2023.3311519

Orozco-Rosas, U., Picos, K., Pantrigo, J. J., Montemayor, A. S., Cuesta-Infante, A., 2022. Mobile robot path planning using a qapf learning algorithm for known and unknown environments. IEEE Access 10, 84648–84663.
DOI: 10.1109/ACCESS.2022.3197628

Peng, S. S., Du, F., Cheng, J., Li, Y., 3 2019. Automation in u.s. longwall coal mining: A state-of-the-art review. International Journal of Mining Science and Technology 29, 151–159.
DOI: 10.1016/j.ijmst.2019.01.005

Prado, A., Michałek, M., Cheein, F. A., 2018a. Machine-learning based approaches for self-tuning trajectory tracking controllers under terrain changes in repetitive tasks. Engineering Applications of Artificial Intelligence 67, 63–80.
DOI: https://doi.org/10.1016/j.engappai.2017.09.013

Prado, A. J., Auat Cheein, F. A., Blazic, S., Torres-Torriti, M., 2018b. Probabilistic self-tuning approaches for enhancing performance of autonomous vehicles in changing terrains. Journal of Terramechanics 78, 39–51.
DOI: https://doi.org/10.1016/j.jterra.2018.04.001

Prado, A. J., Torres-Torriti, M., Cheein, F. A., 10 2021. Distributed tube-based nonlinear mpc for motion control of skid-steer robots with terra-mechanical constraints. IEEE Robotics and Automation Letters 6, 8045–8052.
DOI: 10.1109/LRA.2021.3102328

Prado, J., Yandun, F., Torriti, M. T., Cheein, F. A., 2018c. Overcoming the loss of performance in unmanned ground vehicles due to the terrain variability. IEEE Access 6, 17391–17406.
DOI: 10.1109/ACCESS.2018.2808538

Tan, C. S., Mohd-Mokhtar, R., Arshad, M. R., 2021. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. IEEE Access 9, 119310–119342.
DOI: 10.1109/ACCESS.2021.3108177

Vásquez, M., Yanascual, J., Herrera, M., Prado, A., Camacho, O., 2023. A hybrid sliding mode control based on a nonlinear pid surface for nonlinear chemical processes. Engineering Science and Technology, an International Journal 40, 101361.

Wang, C., Yang, X., Li, H., 2022. Improved q-learning applied to dynamic obstacle avoidance and path planning. IEEE Access 10, 92879–92888.
DOI: 10.1109/ACCESS.2022.3203072

Yin, Z., Cao, W., Song, T., Yang, X., Zhang, T., 6 2022. Reinforcement learning path planning based on step batch q-learning algorithm. IEEE, pp. 630–633.
DOI: 10.1109/ICAICA54878.2022.9844553

Zhang, C., Yang, X., Zhou, R., Guo, Z., 2024. A path planning method based on improved a* and fuzzy control dwa of underground mine vehicles. Applied Sciences 14 (7).
DOI: 10.3390/app14073103

Zhang, C., Zhou, L., Liu, H., 2019. Lalo-check: A path optimization framework for sampling-based motion planning with tree structure. IEEE Access 7, 100733–100746.
DOI: 10.1109/ACCESS.2019.2930634

Zhang, Y., Xia, Q., Xie, P., 6 2021. Research and implementation of path planning for mobile robot in unknown dynamic environment. IEEE, pp. 622–626.
DOI: 10.1109/ICAICA52286.2021.9498260

Zhao, J., Zhu, L., Liu, G., Liu, G., Han, Z., 8 2009. A modified genetic algorithm for global path planning of searching robot in mine disasters. IEEE, pp. 4936–4940.
DOI: 10.1109/ICMA.2009.5246026

Zhao, Z., Bi, L., 9 2020a. A new challenge: Path planning for autonomous truck of open-pit mines in the last transport section. Applied Sciences 10, 6622.
DOI: 10.3390/app10186622

Zhao, Z., Bi, L., 2020b. A new challenge: Path planning for autonomous truck of open-pit mines in the last transport section. Applied Sciences 10 (18).
DOI: 10.3390/app10186622

Zheng, S., Liu, H., 2019. Improved multi-agent deep deterministic policy gradient for path planning-based crowd simulation. IEEE Access 7, 147755–147770.
DOI: 10.1109/ACCESS.2019.2946659