CIMAT Centro de Investigación en Matemáticas, A.C.

Redes de inferencia de controles para la planificación de movimientos en sistemas dinámicos

T E S I S

Que para obtener el grado de

Maestro en Ciencias con especialidad en Computación y Matemáticas Industriales

Presenta

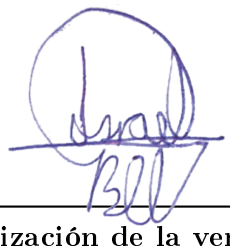
José Gonzalo Palomares Gutiérrez

Director de Tesis:

Dr. Israel Becerra Durán

Co-director de Tesis:

Dr. Rafael Eric Murrieta Cid



Autorización de la versión final

Guanajuato, Gto., Noviembre de 2022

Resumen

El campo de estudio que compete a esta tesis es referente a la conjunción que existe entre el área de la planificación de movimientos basada en muestreo y redes neuronales profundas. Particularmente se hace uso de la versión asintóticamente óptima del planificador *Stable Sparse Rapidly-Exploring Random Tree* (*SST**) para generar trayectorias asintóticamente-mínimas en tiempo, considerando sistemas con restricciones de movimiento (sistemas no holonómicos) y distintos órdenes en la dinámica de los modelos. Los estados intermedios que componen a las trayectorias obtenidas por el planificador *SST** en conjunto con los controles que las generaron, conforman las entradas y salidas, respectivamente, del conjunto de entrenamiento para una nueva variante de la red *MPNet*. Esto contrasta con otros trabajos de planificación para sistemas con restricciones diferenciales en los que se proponen redes neuronales que infieren estados que sirven como guía para que otro método complementario genere los controles que respeten la dinámica del sistema. El método propuesto utiliza una red neuronal para inferir directamente los controles del sistema (más un tiempo de aplicación de los mismos), acelerando el tiempo de generación de trayectorias hasta en tres ordenes de magnitud con respecto a otros métodos basados en aprendizaje, esto sin mencionar que también se obtiene un tiempo de computo significativamente menor al método *SST**. Finalmente, se estudia el desempeño que tiene esta variante de la red *MPNet* al ser entrenada con trayectorias cuyos estados iniciales se encuentran dentro de una vecindad, y como se generaliza el proceso de planificación al considerar estados iniciales de otros vecindarios.

Agradecimientos

Agradezco especialmente a mi asesor Dr. Israel Becerra Durán, así como a mi co-asesor Dr. Rafael Eric Murieta por las enseñanzas, asesoramiento y apoyo continuo que hicieron posible este trabajo de investigación.

Agradezco al Centro de Investigación en Matemáticas (CIMAT) por darme la oportunidad de continuar mis estudios de posgrado dentro de esta institución, así como al cuerpo académico por sus enseñanzas en diferentes disciplinas.

Le doy agradecimientos al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico necesario para la culminación de mis estudios de maestría.

Finalmente, agradezco con mucho cariño a mi familia cercana, a Chalo y Vero, así como a mis hermanos por el apoyo en mi vida diaria.

Índice general

Índice de figuras

Capítulo 1

Introducción

Esta tesis toma relevancia en el área de robótica, específicamente en el área de planificación de movimientos con el apoyo de técnicas de aprendizaje automático, tal como el uso de redes neuronales profundas.

La planificación de movimientos actualmente tiene un rol importante para la investigación referente a robótica e inteligencia artificial. Su objetivo fundamental es encontrar una trayectoria libre de colisiones y de bajo costo que logre conectar una configuración inicial y final [4, 5].

El área de planificación ha mostrado su relevancia mediante un gran número de tareas como lo son la manipulación usando brazos robóticos [6] , tareas de cobertura [7], el manejo de vehículos autónomos [8], etc. La planificación geométrica es bastante útil para resolver problemas que únicamente presentan variables cinemáticas, por ejemplo, la posición u orientación de cierto robot, y cuyas restricciones únicamente se limitan a la evasión de obstáculos. No obstante, el problema que incluye cinemática y dinámica toma en consideración la posición, velocidad y aceleración (y en algunos casos variables de ordenes superiores) de un sistema robótico, es decir, se consideran restricciones diferenciales. Las maniobras que debe realizar un robot con restricciones diferenciales para obtener una trayectoria entre dos estados corresponden por lo general a la solución de un problema de planificación computacionalmente demandante [5].

La planificación de movimientos ha tenido grandes avances desde que se presenta-

ron los primeros algoritmos cuya generación de soluciones exactas era el acercamiento preferido [9], pasando a través de descomposiciones en celdas, búsqueda de grafos, campos de potencial y métodos basados en muestreo. Particularmente, los métodos basados en muestreo han sido exitosos para la búsqueda de trayectorias libres de colisión en espacios de altas dimensiones y han evolucionado constantemente desde el método de mapas de caminos *PRM* [10] hasta el método *RRT* [11], el cual introduce el concepto de muestreo en el espacio de controles para tratar el problema en sistemas dinámicos. Relacionado a esto, se han hecho un gran número de trabajos referentes al problema de planificación con restricciones diferenciales [12, 13, 14, 15, 11]. A pesar de que los métodos basados en muestreo pueden encontrar con cierta facilidad una trayectoria factible en espacios de trabajo con pocos obstáculos, muchas veces pueden tardar un tiempo considerable en encontrar una solución al tratar con entornos complicados, por ejemplo, pasajes estrechos.

Algunos métodos de planificación basada en muestreo tienen la propiedad de optimalidad asintótica, no obstante, estos requieren de la solución de un problema con valores en la frontera (BVP por sus siglas en inglés) en el espacio de estados. Por otra parte, los métodos *SST* y *SST** [3] no requieren solucionar un BVP para obtener trayectorias de alta calidad y mantienen una estructura de datos de poco tamaño.

Recientemente se han implementado métodos basados en aprendizaje por refuerzo que se apoyan con modelos de redes neuronales para resolver problemas de planificación que involucran sistemas no holonómicos. En ocasiones estos métodos demandan una cantidad significativa recursos computacionales para encontrar los parámetros de ajuste adecuados. Una alternativa de planificación por medio de redes neuronales utiliza un acercamiento por medio de la imitación de trayectorias obtenidas por un método de planificación experto. La red MPNet, expuesta en [16], así como su versión de planificación en sistemas dinámicos [2] utiliza este acercamiento por medio de imitación para la generación de trayectorias asintóticamente óptimas, superando por varios ordenes de magnitud al desempeño en el tiempo de cómputo de algunos algoritmos en el estado del arte moderno.

Este trabajo se centra en la implementación de un algoritmo para resolver el pro-

blema de planificación para sistemas dinámicos mediante el uso de redes de inferencia que generan entradas de control. La idea principal del método propuesto consiste en el entrenamiento de los modelos neuronales, bajo un esquema de aprendizaje híbrido, utilizando los estados y *controles* obtenidos a través de un método de planificación dinámica. Específicamente, se hace uso del método *SST** para la generación de trayectorias asintóticamente óptimas, tomando como costo el tiempo de aplicación de las entradas de control Δt. El desempeño de este método se expone comparando tiempos de ejecución y costo de trayectorias con respecto al planificador *SST**.

Se proponen seis experimentos en donde se registra el desempeño del algoritmo propuesto para cuatro sistemas dinámicos diferentes. El alcance de esta tesis se limita a la aplicación del método considerando entornos de prueba en dos dimensiones, pero es posible generalizar esta metodología a espacios tridimensionales.

Este trabajo de investigación se encuentra organizado de la siguiente manera: en el capítulo 2 se presentan brevemente los elementos que conforman a los métodos de planificación basada en muestreo, así como la descripción de los algoritmos *PRM* y *RRT* junto a sus versiones óptimas *PRM** y *RRT**. Además, se introducen algunos acercamientos que utilizan redes neuronales para resolver el problema de planificación de movimientos.

En el capítulo 3 se mencionan los trabajos relevantes a esta investigación. Se presenta la formulación del problema de planificación en sistemas dinámicos, así como los métodos que fungen como herramienta para la construcción de la metodología utilizada. Se da una descripción de los planificadores *SST* y *SST**, los detalles de implementación de la red *MPNet* y su versión de planificación para sistemas dinámicos *MPC-MPNet*.

El capítulo 4 contiene la metodología propuesta para la solución del problema de planificación en sistemas dinámicos. Adicionalmente se mencionan los detalles de la implementación específica utilizada para los experimentos en este trabajo.

En el capítulo 5 se exponen los sistemas de prueba utilizados en los experimentos. Se muestran las ecuaciones de transición de estados para un modelo de *robot omnidireccional*, así como para el *robot de manejo diferencial* (*DDR* de primer y segundo

orden) y *robot tipo automóvil*.

En el capítulo 6 se presentan los resultados de los experimentos planteados. Los resultados exponen el desempeño de la metodología propuesta capturando valores estadísticos tales como el porcentaje de éxito y el costo de generación de trayectorias.

Finalmente, en el capítulo 7 se presentan las conclusiones generales de esta tesis así como el posible trabajo futuro.

Capítulo 2

Trabajo Previo

Durante las últimas décadas se han presentado numerosos métodos y clasificaciones relevantes a las técnicas de planificación de movimientos. En [5] se identifican métodos que funcionan en el espacio continuo, planificadores en dominios discretos, métodos de análisis combinatorio y métodos basados en muestreo, conocidos comúnmente como planificadores probabilísticos.

En la sección 2.1 de este capítulo se introduce de manera general el contexto necesario para comprender el funcionamiento de los métodos probabilísticos. En las secciones restantes se describen algunas de las técnicas de planificación que son relevantes a este trabajo y que en su mayoría han marcado un hito en el estado del arte moderno.

2.1. Métodos de planificación basada en muestreo

Un acercamiento popular para la solución de problemas en planificación de movimientos es mediante los *métodos basados en muestreo*. La idea principal es evitar la construcción explícita de un espacio de configuraciones $\mathcal{C}$ y en lugar de ello se explora este espacio utilizando un esquema de muestreo. Los planificadores basados en muestreo por lo general tienen problemas al lidiar con pasajes estrechos en el espacio de configuraciones, ya que hay baja probabilidad de generar una muestra sobre estos.

Los primeros planificadores probabilísticos utilizaban técnicas de optimización lo-

cal, comúnmente se usaban métodos de descenso de gradiente construidos a través del espacio de configuraciones para determinar una trayectoria en dirección a la región objetivo [4]. En ocasiones el resultado conducía a los sistemas de prueba hacia un estancamiento en mínimos locales. Tiempo después se abandonó la idea de usar técnicas de optimización local para dar un enfoque a la búsqueda de conectividad en espacios de configuraciones de altas dimensiones mediante el uso de muestreo [4].

Se define como *espacio de trabajo* al espacio cartesiano donde opera el robot. Este cuenta con ejes ortonormales ($X,\ Y\ y\ Z$) y es el conjunto de todos las posiciones alcanzables.

La configuración de un robot se refiere a un vector cuyas coordenadas se definen con respecto a cierto marco de referencia en donde cada uno de sus elementos representa un grado de libertad del robot. El conjunto de todas las configuraciones del robot es conocido como el espacio de configuraciones $\mathcal{C}$. El espacio de configuraciones es un espacio *n-dimensional* donde n es el número de grados de libertad del robot. El robot en el espacio de configuraciones es representado como un objeto puntual. La figura 2.1 muestra como se encuentra dividido el espacio de configuraciones $\mathcal{C}$: se denota como $\mathcal{C}_{free}$ al conjunto de todas las configuraciones posibles que se encuentren libres de colisión, y $\mathcal{C}_{obs}$ al conjunto de todas las configuraciones en colisión con obstáculos.

Sea c_{init} la configuración inicial de un sistema robótico y c_{goal} una configuración objetivo. A la ruta que va desde c_{init} a c_{goal} se le conoce como *camino*. Un camino π es un mapeo continuo o una función en $\mathcal{C}$ tal que $\pi : [0,1] \rightarrow \mathcal{C}$, con $\pi(0) = c_{init}$ y $\pi(1) = c_{goal}$, la cual únicamente toma en cuenta consideraciones geométricas. Una *trayectoria* τ es un camino con parametrización explícita en el tiempo, por ejemplo, acompañada por una descripción de las leyes de movimiento (incluyendo dinámica), esto es, $t \in [T_0, T_f] \rightsquigarrow \tau \in [0,1] : c(t) = \pi(\tau) \in \mathcal{C}_{free}$.

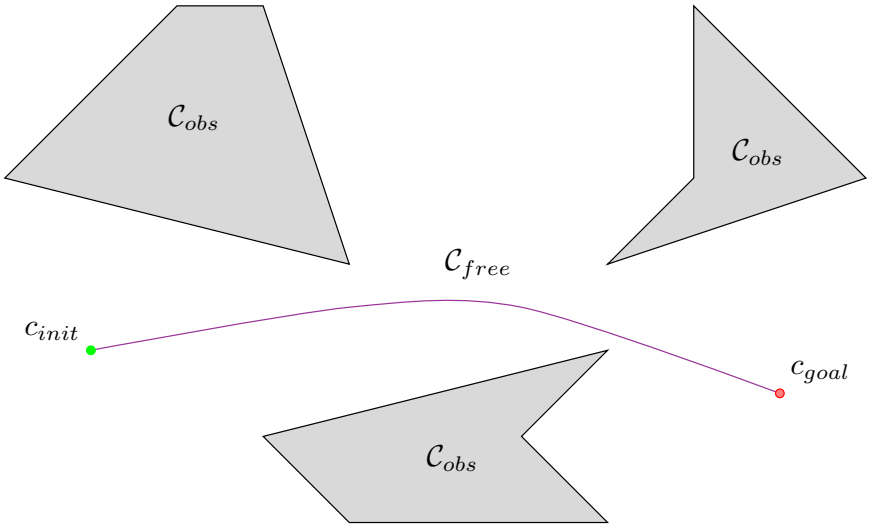


Figura 2.1: Espacio de configuraciones. El espacio de configuraciones en colisión se muestra de color gris, mientras que el espacio libre de colisiones abarca la región de color blanco. Se muestra una configuración inicial c_{init}, una configuración objetivo c_{goal} y una trayectoria en el espacio libre de colisiones.

Un estado $x \in \mathcal{X}$ puede ser simplemente una configuración c, o bien, una configuración junto a velocidades, aceleraciones o derivadas de órdenes superiores. De manera análoga al espacio de configuraciones, se define como $\mathcal{X}_{free}$ al espacio de estados libre de colisiones y $\mathcal{X}_{obs}$ al espacio de estados en colisión.

Sea $x \in \mathcal{X}_{free}$ un estado válido para cierto robot y $\delta > 0$ un número real, se dice que el estado x es $\delta - interior$ de $\mathcal{X}_{free}$ si una bola de radio δ centrada en x se encuentra totalmente dentro de $\mathcal{X}_{free}$. Al conjunto de todos los estados que son $\delta - interior$ de $\mathcal{X}_{free}$ se denota como $int_\delta(\mathcal{X}_{free})$, esto es, $int_\delta(\mathcal{X}_{free}) = \{x \in \mathcal{X}_{free} | B_{x,\delta} \subseteq \mathcal{X}_{free}\}$. En otras palabras, el conjunto $\delta - interior$ de $\mathcal{X}_{free}$ es el conjunto de todos los estados que se encuentran al menos a una distancia δ de cualquier obstáculo dentro del espacio de estados.

Sea $\sigma \in \Sigma_{free}$ un camino libre de colisión donde Σ_{free} es el conjunto de todos los caminos libres de colisión. Se dice que un camino homotópico a σ puede ser transformado continuamente a σ a través de $\mathcal{X}_{free}$. Una trayectoria libre de colisión $\sigma : [0, 1] \to \mathcal{X}_{free}$ tiene una *holgura δ fuerte* si σ se encuentra totalmente en $\delta - interior$ de $\mathcal{X}_{free}$. Esto es, $\sigma(\tau) \in int_\sigma(\mathcal{X}_{free})$ para todo $\tau \in [0, 1]$. Dado un problema de planificación de tra-

yectorias $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$, con estado inicial x_{init} y un conjunto de estados objetivo $\mathcal{X}_{goal}$, se dice que es *robustamente alcanzable* o *viable* si existe una trayectoria con holgura δ fuerte para $\delta > 0$. Por otro lado, un camino $\sigma : [0,1] \to \mathcal{X}_{free}$ se dice que tiene una *holgura δ débil*, si existe un camino σ' que tiene holgura δ fuerte y existe una homotopía ψ, con $\psi(0) = \sigma$, $\psi(1) = \sigma'$ y para todo $\alpha \in (0,1]$ existe $\delta_\alpha > 0$ tal que $\psi(\alpha)$ tiene una holgura δ_a fuerte [17]. En la figura 2.2 se explica gráficamente el concepto de holgura δ fuerte y holgura δ débil.

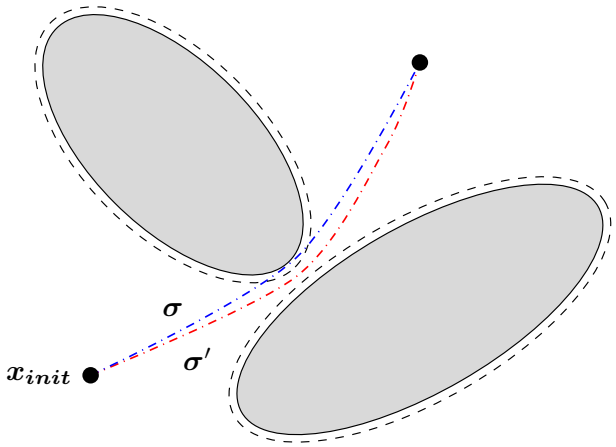


Figura 2.2: El camino σ tiene una holgura δ débil, el camino σ' tiene una holgura δ fuerte: este último pertenece totalmente al interior de $\mathcal{X}_{free}$ y tiene la misma clase homotópica que σ.

Se dice que un algoritmo de planificación de movimientos es completo cuando este es capaz de determinar si existe o no una solución (trayectoria libre de colisión) en un tiempo finito. En un contexto probabilístico, los algoritmos no son capaces de reportar fracasos en tiempo finito cuando la solución no existe.

Un concepto similar a la completitud es la *completitud probabilística*. Un algoritmo **ALG** se dice que es probabilísticamente completo si, para cualquier problema robustamente alcanzable, se cumple la siguiente condición:

$$\lim_{n\to\infty} P(\{\exists\, x_{goal} \in V_n^{ALG} \cap \mathcal{X}_{goal} : x_{init} \text{ conectado a } x_{goal} \text{ en } G_n^{ALG}\}) = 1, \qquad (2.1)$$

donde V_n^{ALG} es el conjunto de vértices en el grafo G_n^{ALG} generado por **ALG** despues de n muestras [17].

La ecuación (2.1) indica que un algoritmo es probabilísticamente completo cuando el límite inferior de la probabilidad de conectar x_{init} a x_{goal} en el grafo generado por **ALG** es igual a 1.

Los algoritmos *PRM* y *RRT*, así como algunos otros métodos relacionados, son probabilísticamente completos, por lo que la probabilidad de encontrar una solución, si existe, tiende a ser 1 cuando el número de muestras tiende al infinito.

Como se menciona al inicio de esta sección, los pasajes estrechos sobre $\mathcal{X}_{free}$ representan una dificultad para la obtención de una solución al problema de planificación de trayectorias dado un planificador basado en muestreo. La figura 2.3 muestra un ejemplo de pasaje estrecho: asumiendo que se efectúa el muestreo mediante una distribución uniforme, la probabilidad de generar muestras que se encuentran en el espacio libre de colisiones es baja. Se puede notar que al reducir el grosor del pasaje estrecho la probabilidad de generar muestras sobre este disminuye.

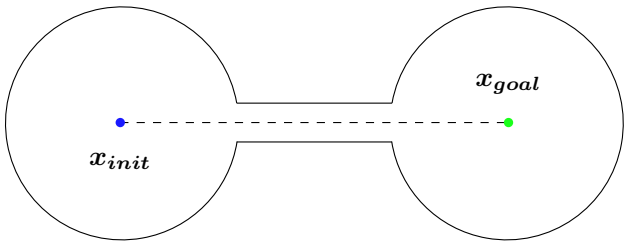


Figura 2.3: La conectividad entre estados resulta difícil al utilizar un método de planificación basada en muestreo pues existe la presencia de un pasaje estrecho.

2.2. PRM

El método probabilístico de mapas de caminos (*probabilistic roadmaps, PRM*) corresponde al primer algoritmo popular para la solución de problemas de planificación basada en muestreo. Este planificador se enfoca en el muestreo aleatorio del espacio de configuraciones así como en generar un mapa de caminos en forma de grafo no dirigido

cuyas aristas conectan dos configuraciones libres de colisión [10]. Este algoritmo es probabilísticamente completo.

Una característica interesante de este método es su capacidad de encontrar una solución en un tiempo de ejecución razonable cuando en el robot se tienen muchos grados de libertad. Al momento de ser publicado [10], habían pocos métodos eficaces para resolver problemas de dimensionalidad alta en el espacio de configuraciones, por lo que representa un trabajo destacable en el área de planificación de movimientos.

Algorithm 1 PRM

1: $V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
2: $G = (V, E)$;
3: **for** $i = 1, \cdots, n$ **do**
4: $\quad x_{rand} \leftarrow \text{Sample}(\mathcal{X}_{free})$;
5: $\quad W \leftarrow \text{Near}(V, x_{rand}, r)$;
6: $\quad V \leftarrow V \cup \{x_{rand}\}$;
7: $\quad$ **for** $x' \in W,$ en orden creciente de $\|x' - x_{rand}\|$ **do**
8: $\quad\quad$ **if** x_{rand} y x' no están en la misma componente conexa de G **then**
9: $\quad\quad\quad$ **if** Collision_Free(x', x'_{rand}) **then**
10: $\quad\quad\quad\quad E \leftarrow E \cup \{(x', x_{rand})\}$;
11: $\quad\quad\quad$ **end if**
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: **end for**
15: **return** $G = (V, E)$;

El algoritmo 1 muestra el esquema general correspondiente al proceso de planificación utilizando el método *PRM*. Los métodos probabilísticos que se mencionan en este trabajo tienen en su estructura algunas funciones en común; en el Apéndice C se definen las funciones utilizadas en el algoritmo 1 así como en los algoritmos subsecuentes.

El método *PRM* cuenta con una fase de aprendizaje y una fase de consulta: en la fase de aprendizaje se muestrea un estado aleatorio $x_{rand} \in \mathcal{X}_{free}$ y se agrega

al conjunto de nodos V. Luego, se verifican las posibles conexiones utilizando un planificador local (usualmente se usa una línea recta en el espacio de configuraciones para sistemas sin restricciones diferenciales) entre x_{rand} y los nodos en V que se encuentran dentro de una bola de radio r centrada en x_{rand}, considerando primero el nodo con menor distancia a x_{rand}, y continuando en orden creciente de distancia. Al encontrar una conexión libre de colisión, esta se agrega al conjunto de aristas E.

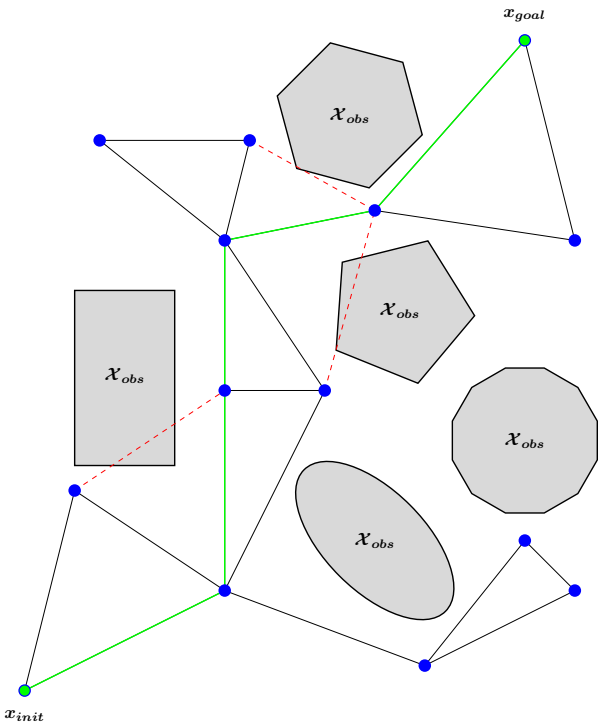


Figura 2.4: Método probabilístico de mapa de caminos (PRM). Primeramente se generan muestras aleatorias en el espacio libre de colisiones $\mathcal{X}_{free}$. Luego, estas muestras son conectadas unas a otras por un planificador local (usualmente se usan trayectorias en línea recta) considerando todas las posibles conexiones en un radio r. Las lineas de color verde representan las conexiones para alcanzar x_{goal} desde x_{init}.

Durante la fase de consulta, se solicita una trayectoria entre dos estados libres que se encuentren en el grafo. Los nodos inicial y final son conectados mediante una secuencia de aristas E que logre resolver el problema de planificación. En la figura 2.4 se muestran los nodos y aristas de un mapa de caminos, así como la trayectoria que conecta un estado inicial y final.

2.3. PRM*

En algunas ocasiones puede presentarse el caso donde la calidad de las trayectorias es de suma importancia, es posible que se desee obtener la trayectoria más corta ya sea en distancia o en cualquier otra variante del costo. Los algoritmos *PRM** y *RRT** descritos en [17], corresponden a las versiones óptimas del *PRM* y *RRT* respectivamente.

El algoritmo *PRM** (al igual que el algoritmo *RRT**) es probabilísticamente completo y asintóticamente óptimo, es decir, la solución tiende a ser óptima cuando el número de nodos en el grafo incrementa.

El algoritmo *PRM** es similar a su versión estándar, la diferencia principal es que para conectar los nodos dentro de una vecindad se utiliza un radio en específico. Un planificador local se usa para encontrar trayectorias libres de colisión entre un nodo y el subconjunto de nodos dentro de una vecindad. Esta vecindad cuenta con propiedades de optimalidad y eficiencia, pues el radio a utilizar varía dependiendo del número de nodos que existen en el grafo para asegurar la optimalidad asintótica, y a su vez reducir el número de cálculos. El radio se expresa en función de la cantidad actual de nodos en el árbol n y se define bajo la siguiente expresión:

$$r(n) = \gamma_{PRM}(log(n)/n)^{1/d}, \tag{2.2}$$

donde $\gamma_{RPM} > 2(1+d)^{1/d}(\mu(\mathcal{C}_{free})/\zeta_d)^{1/d}$, d es la dimensión del espacio de configuraciones $\mathcal{C}$, $\mu(\mathcal{C}_{free})$ es la medida de Lebesgue de $\mathcal{C}_{free}$ y ζ_d es el volumen de la esfera unitaria de dimension d. Se puede observar que el radio de conexión $r(n)$ disminuye cuando el número de muestras n incrementa. El número de conexiones promedio para cierto nodo es del orden de $log(n)$.

2.4. RRT

El árbol aleatorio de exploración rápida (*rapidly exploring random tree, RRT*) corresponde a una estructura de datos y a un algoritmo que está diseñado para la

exploración eficiente en espacios no convexos de alta dimensión, usando un árbol que se genera de manera aleatoria. El árbol se construye incrementalmente generando muestras desde una función de densidad uniforme, y añadiendo una muestra por iteración si esta se encuentra libre de colisiones. El *RRT* ha sido uno de los métodos más utilizados en el área de planificación de movimientos para robots autónomos [11] ya que es capaz de tratar al mismo tiempo con obstáculos y restricciones de movimiento.

La descripción del método *RRT* se muestra en el algoritmo 2, el cual se presentó por primera vez en [18].

Algorithm 2 RRT

$V \leftarrow \{x_0\}$; $E \leftarrow \emptyset$;
$G = (V, E)$;
for $i = 1, \cdots, n$ **do**
    $x_{rand} \leftarrow \text{Sample}(\mathcal{X}_{free})$;
    $x_{neareast} \leftarrow \text{Nearest}(G, x_{rand})$;
    $u \leftarrow \text{Select\_Input}(x_{rand}, x_{nearest})$;
    $x_{new} \leftarrow \text{New\_State}(x_{nearest,u})$;
    if $\text{Collision\_Free}(x_{nearest}, x_{new})$ **then**
        $V \leftarrow V \cup \{x_{new}\}$;
        $E \leftarrow E \cup \{(x_{near}, x_{new}\})$;
    end if
end for
return $G = (V, E)$;

Primeramente se inicializa un grafo con un solo nodo sin aristas para denotar el estado inicial. En cada iteración se muestrea un nuevo estado $x_{rand} \in \mathcal{X}_{free}$ y dentro del árbol se hace una búsqueda del nodo más cercano a x_{rand} (distancia euclidiana), el cual se denota como $x_{nearest}$. Posteriormente se tratará de expandir la estructura de árbol desde $x_{nearest}$ hacia x_{rand}. Dado el caso en donde la trayectoria esté completamente libre de colisión, se agrega x_{new} a la estructura de árbol y ($x_{nearest}$, x_{new}) al conjunto de aristas. Una versión del *RRT* que trata con restricciones diferenciales

selecciona un control válido u de manera aleatoria. Al aplicar esta entrada de control u durante cierto tiempo finito t, se genera un nuevo estado x_{new} por medio de la integración directa de la ecuación de transición de estados, donde finalmente se verifica que la trayectoria de $x_{nearest}$ a x_{new} se encuentre en el espacio libre de colisiones. En la figura 2.5 se muestra el proceso de expansión de la estructura de árbol dada una iteración.

Una vez construido el árbol se resuelve el problema de planificación al verificar si algún nodo corresponde al estado objetivo, o se encuentra dentro de la región objetivo.

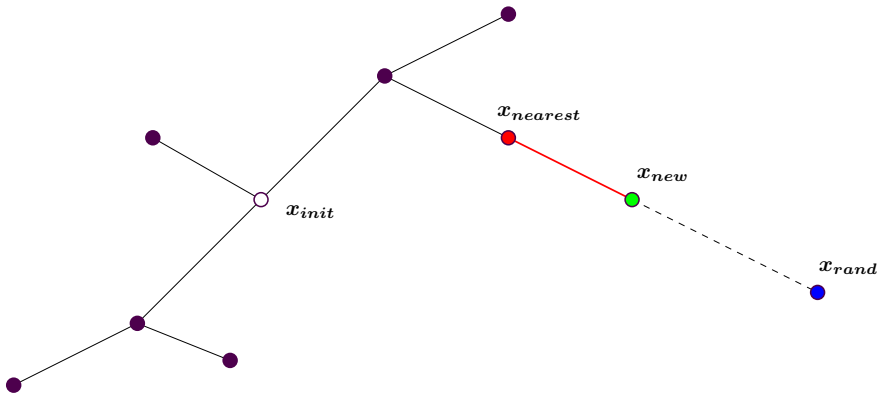


Figura 2.5: Método *RRT*. Primeramente se genera una muestra aleatoria x_{rand}, seguido de esto se encuentra el nodo más cercano en el árbol $x_{nearest}$, luego se calcula un nuevo nodo x_{new} mediante la aplicación de un control $u \in \mathcal{U}$ por cierto tiempo t. Finalmente se establece una conexión entre $x_{nearest}$ y x_{new} si la trayectoria entre estos está libre de colisiones.

2.5. RRT*

Al igual que en su versión estándar, el algoritmo *RRT** construye una estructura de árbol de manera incremental, pero en este caso, la trayectoria resultante converge a la solución óptima. El proceso de construcción es similar al del *RRT*, con la diferencia que en el *RRT** se añade un proceso de reconexión para asegurar que cierto nodo se alcanza a través del camino más corto. Otra diferencia con respecto al *RRT* es que el *RRT** considera el costo acumulado desde la raíz del árbol hasta cualquier nodo para alcanzar la optimalidad [17].

El método *RRT** utiliza la función de direccionamiento *Steer* (descrita en el Apén-

dice C), la cual formaliza la idea de prolongar el árbol hacia x_{rand}, pues permite el crecimiento de la estructura de datos y logra la optimalidad asintótica. La función de direccionamiento para sistemas dinámicos corresponde a la solución de un problema con valores en la frontera entre dos puntos (BVP). Este problema requiere de la solución de ecuaciones diferenciales con restricciones de contorno, lo cual representa una tarea difícil para una gran variedad de sistemas dinámicos.

La estructura del algoritmo RRT^* se muestra en 3; se agregan nodos a V de manera similar al método RRT, con la diferencia de utilizar la función de direccionamiento en lugar de muestrear controles. Para realizar una conexión del nuevo nodo x_{new} con el árbol se consideran los nodos que se encuentran dentro de una distancia $r(n)$ de x_{new}. Similar al algoritmo PRM^*, el radio $r(n)$ depende del número actual de nodos en el árbol n, y está definido por

$$r(n) = \text{mín}(\gamma_{RRT*}(\log(n)/n)^{1/d}, \eta), \tag{2.3}$$

donde $\gamma_{RRT*} > 2(1+1/d)^{1/d}(\mu(\mathcal{C}_{free})/\zeta_d)^{1/d}$, d es la dimensión del espacio $\mathcal{C}$, $\mu(C_{free})$ es la medida de Lebesgue de $\mathcal{C}_{free}$, ζ_d es el volumen de la esfera unitaria de dimensión d y η corresponde a un valor constante que acota la medida del radio.

Se puede dar el caso donde las conexiones no dan como resultado una nueva arista en el árbol. Particularmente, se genera una nueva arista dentro de una vecindad X_{near}, si esta arista se encuentra libre de colisión y tiene una trayectoria de costo mínimo hacia x_{new} desde un vértice en X_{near}. En el proceso de reconexión se generan aristas desde x_{new} a nodos en X_{near}, por ejemplo, a x', si la trayectoria para ir de x_{new} a x' tiene un costo menor que la trayectoria actual para ir a x'. Dado el caso, la conexión entre x' y su nodo padre se elimina para mantener la estructura de árbol.

Algorithm 3 RRT*

$V \leftarrow \{x_0\}; E \leftarrow \emptyset;$
$G = (V, E);$
for $i = 0, \ldots, n$ **do**
    $x_{rand} \leftarrow \text{Sample}(\mathcal{X}_f);$
    $x_{nearest} \leftarrow \text{Nearest}(G, x_{rand});$
    $x_{new} \leftarrow \text{Steer}(x_{nearest}, x_{rand});$
    if Collision_Free$(x_{nearest}, x_{new})$ **then**
        $X_{near} \leftarrow \text{Near}(G, x_{new}, r(i));$
        $V \leftarrow V \cup \{x_{new}\};$
        $x_{min} \leftarrow x_{nearest}$; $c_{min} \leftarrow \text{Cost}(x_{nearest}) + \text{Cost}((x_{nearest}, x_{new}));$
        for $x_{near} \in X_{near}$ **do**
            if Collision_Free(x_{near}, x_{new}) **and**
            $\text{Cost}(x_{near}) + \text{Cost}((x_{near}, x_{new})) < c_{min}$ **then**
                $x_{min} \leftarrow x_{near}; c_{min} \leftarrow \text{Cost}(x_{near}) + \text{Cost}((x_{near}, x_{new}));$
            end if
        end for
        $E \leftarrow E \cup \{(x_{min}, x_{new})\};$
        for $x_{near} \in X_{near}$ **do**
            if Collision_Free$(x_{new}, x_{near}$ **and**
            $\text{Cost}(x_{near}) + \text{Cost}((x_{near}, x_{new})) < \text{Cost}(x_{near})$ **then**
                $x_{parent} \leftarrow \text{Parent}(x_{near});$
                $E \leftarrow (E \setminus \{(x_{parent}, x_{near})\} \cup \{(x_{new}, x_{near})\})$
            end if
        end for
    end if
end for

El método RRT* cuenta con las siguientes limitaciones. 1) El algoritmo no incluye una subrutina apropiada que mencione, por ejemplo, la probabilidad de que una solución no exista, por otro lado, el algoritmo típicamente termina dependiendo del número de nodos actuales en el árbol. 2) No es del todo claro que tipo de funciones de

costo puede utilizar este algoritmo. 3) La incertidumbre en el movimiento del robot no se trata directamente, ni se minimiza.

2.6. Modelos neuronales

En robótica, el problema de planificación de caminos en un entorno con obstáculos en movimiento sigue siendo un problema a resolver. Una estrategia para lidiar con este tipo de entornos es mediante la predicción del movimiento de los obstáculos. En [19] se propone una acercamiento de predicción visual como estrategia de planificación. Recientemente, el problema de generación de fotogramas usando redes predictivas se ha manifestado con bastante popularidad y se ha abarcado en numerosas ocasiones [20, 21, 22]. Mientras que las redes neuronales convolucionales (*CNN*) han sido bastante exitosas para el aprendizaje de características por medio de imágenes estáticas [23, 24], la idea de las redes convolucionales con memoria de corto-largo plazo (*Convolutional Long Short Term Memory, LSTM*), se ha desarrollado para capturar las dependencias espaciales y temporales de datos de video [25].

En [19] se presenta la red de predicción de movimientos de robot (*Predicting Robot Motion Network, PROM-Net*), la cual es capaz de predecir la trayectoria de un objeto en movimiento dentro del espacio de trabajo de un sistema robótico dado. Esta toma como entrada los fotogramas generados por un modelo de predicción de controles, con el objetivo de realizar una predicción del estado del robot en los siguientes fotogramas. La *PROM-Net* además genera una representación de los estados en el espacio de trabajo.

Los autores de [26] presentan un algoritmo de planificación a través del entrenamiento de redes neuronales convolucionales (*CNN*) conocido como *Neural RRT**. Estas redes aprenden de un gran número de caminos óptimos obtenidos por el algoritmo *A** [27]. Dado un problema de planificación, este modelo es capaz de predecir rápidamente una densidad de probabilidad para el camino óptimo, la cual es utilizada para guiar al proceso de generación de muestras del *RRT**.

Capítulo 3

Marco Teórico

En este capítulo se presenta la investigación previa que sirve como base para la construcción de la metodología de planificación presente en este trabajo. La sección 3.1 da una breve explicación al problema de planificación en sistemas dinámicos. En las secciones 3.2 y 3.3 se explica el funcionamiento e implementación de los planificadores *SST* y *SST**, respectivamente, siendo este último fundamental para el desarrollo del algoritmo propuesto. La sección 3.4 exhibe una red de planificación neuronal, cuya arquitectura se asemeja a la utilizada en este trabajo. Finalmente, en la sección 3.5 se presenta una extensión de la misma red de planificación neuronal para la solución del problema con restricciones diferenciales.

3.1. El problema de planificación en sistemas dinámicos

El problema de planificación con restricciones cinemáticas y dinámicas consiste en determinar las entradas de control para conducir un sistema dinámico desde un estado inicial a un estado objetivo, evitando colisiones con obstáculos y respetando las restricciones diferenciales. Este problema fue propuesto por primera vez en [28].

Se dice que un sistema es *no holonómico* cuando cuenta restricciones diferenciales, esto es, no poder moverse de manera omnidireccional. La propiedad de no holonomía

dado un sistema dinámico puede determinarse mediante el teorema de Frobenius [14]. Dada una relación entre las configuraciones válidas para un robot y sus posibles derivadas, se presentan r restricciones asociadas a r ecuaciones que son lineales a las n derivadas de cada variable de configuración, estas ecuaciones determinan lo que se conoce como distribución $\triangle$ en el espacio de configuraciones. Recordando que la operación asociada a los corchetes de Lie entre dos campos vectoriales X y Y se define como $[X, Y] = \partial X.Y - \partial Y.X$, el teorema de Frobenius indica que las ecuaciones son integrables (holonómicas) si y solo si $\triangle$ es cerrada bajo la operación de los corchetes de Lie [13].

Desde la óptica de la teoría de control, una entrada de control corresponde a una variable o función que permite manipular el estado de un sistema dinámico. El *espacio de controles* es el conjunto de las entradas de control aplicables a dado sistema dinámico. En este trabajo el espacio de controles se denota con $\mathcal{U}$.

Dado un problema de planificación de movimientos, se consideran sistemas dinámicos que cumplen restricciones diferenciables invariantes en el tiempo:

$$\dot{x} = f(x(t), u(t)), \quad x(t) \in \mathcal{X}, u(t) \in \mathcal{U}. \tag{3.1}$$

La ecuación (3.1) se conoce como *ecuación de transición de estados*, y es la representación dinámica para un sistema dado. En el capítulo 5 se presentan las restricciones diferenciales para los sistemas utilizados en este trabajo.

En ocasiones, un sistema de la forma (3.1) puede de igual manera formularse como una combinación linear de campos vectoriales en una variedad $\mathcal{X}$ (el espacio continuo de estados):

$$\dot{x} = h_0(x) + \sum_{i=1}^{m} h_i(x) u_i, \tag{3.2}$$

donde cada h_i corresponde a un campo vectorial de $\mathcal{X}$. Cada entrada de control $u_i \in \mathcal{U}$ puede interpretarse como el coeficiente que determina que tanto influye el campo vectorial $h_i(x)$ en $\dot{x}$. El término h_0 en (3.2), se conoce como *término de deriva*,

y este se encuentra presente aún cuando no se aplica una entrada de control. Cuando $h_0(x) = 0$, se dice que el sistema no tiene deriva, por lo que (3.2) se expresa como

$$\dot{x} = \sum_{i=1}^{m} h_i(x)u_i, \tag{3.3}$$

o alternativamente

$$\dot{x} = H(x)u, \tag{3.4}$$

donde $H(x)$ es una matriz de dimensión $n \times m$ que contiene a los campos vectoriales $h_i(x)$.

La *controlabilidad* de un sistema dinámico indica si un estado $p \in \mathcal{X}$ es alcanzable desde otro a través de una secuencia de controles admisibles. Considerando un sistema dinámico de la forma (3.2), donde :

- El espacio de estados $\mathcal{X}$ es una variedad suave (continuamente diferenciable C^∞).
- $h = (h_0, \cdots, h_m)$ es un conjunto de campos vectoriales C^∞ en $\mathcal{X}$.
- $\mathcal{U}$ es un subconjunto de $\mathbb{R}^m$ tal que

$$\text{Aff}(\mathcal{U}) = \mathbb{R}^m,$$

  siendo $\text{Aff}(\mathcal{U})$ la envolvente afín de $\mathcal{U}$, esto es, el conjunto de todas las combinaciones lineales $\sum \alpha_i u_i$ con $u_i \in \mathcal{U}$, $\alpha_i \in \mathbb{R}$ y $\sum_i \alpha_i = 1$.

Dado un intervalo $[0, T]$. Si $q \in \mathcal{X}$ tiene la forma $x(T)$ para una trayectoria tal que $x(0) = p$, entonces se dice que q es alcanzable desde p en un tiempo T. El conjunto de todo q que es alcanzable desde p en tiempo un tiempo T para un sistema construido por $\Sigma = (\mathcal{X}, g, \mathcal{U})$ se denota como $\text{Reach}(\Sigma, \leq T, p)$.

Se dice que un sistema Σ es *localmente controlable en pequeño tiempo (STLC)* desde p si p es un punto interior de $\text{Reach}(\Sigma, \leq T, p)$ para todo $T > 0$ [29].

La *controlabilidad* de un sistema sin deriva puede ser caracterizada utilizando la condición del rango del álgebra de Lie ($LARC$). El teorema de Chow-Rashevskii [30, 31, 32] indica que un sistema afín es $STLC$ en $p \in \mathcal{X}$ si y solo si la dimensión del álgebra de Lie $\dim(\mathcal{L}_p(\triangle))$ es igual a n, la dimensión del espacio de estados $\mathcal{X}$. Si esta condición se mantiene para todo $p \in \mathcal{X}$, entonces se dice que todo el sistema es $STLC$.

Un concepto estrechamente relacionado a *controlabilidad* es el de *accesibilidad*. Si existe algún tiempo $T > 0$ por el cual la dimensión de $\text{Reach}(\Sigma, \leq T, p)$ es n (dimensión del espacio de estados $\mathcal{X}$), entonces se dice que el sistema compuesto por Σ es *accesible* desde p. Un sistema *localmente accesible en pequeño tiempo (STLA)* requiere que exista algún tiempo $T > 0$ tal que el conjunto interior de $\text{Reach}(\Sigma, \leq T, p)$ no sea vacío en $[0, T]$ [5].

3.2. SST

Las versiones óptimas de los métodos mencionados en el capítulo 2 tienen la facilidad de alcanzar la optimalidad asintótica conforme aumenta el número de iteraciones, sin embargo, obtener esta característica para cierto sistema dinámico requiere resolver un problema de valores en la frontera (BVP) en el espacio de estados subyacente. A su vez, resulta poco práctico idear un solucionador de BVP para una gran variedad de sistemas dinámicos. Debido a ello, se han implementado algunos acercamientos basados en muestreo que logran obtener garantías de optimalidad sin tener acceso a un solucionador de BVP.

Los métodos SST (*Stable Sparse RRT*) y SST^* [3, 33] contienen algoritmos que resultan ser eficaces sin necesidad de resolver un BVP, siendo el SST^* la versión asintóticamente óptima del SST. Ambos métodos convergen relativamente rápido a trayectorias de alta calidad con respecto a otros planificadores en el estado del arte, mientras mantienen una estructura de datos de poco tamaño que conducen a trayectorias localmente óptimas.

Gran parte de los planificadores basados en muestreo requieren dos etapas en su

construcción: selección y propagación. Adicionalmente, los métodos *SST* y *SST** incluyen una fase de poda para eliminar nodos no relevantes y mantener una estructura de datos de menor tamaño. Algunos de los aspectos que se introducen en el método *SST* son los siguientes:

- Se requiere un parámetro de entrada δ_{BN} que será utilizado en el proceso de selección. El valor de este parámetro influye en el número de nodos que se consideran para seleccionar el nodo que se desea propagar. Entre más grande sea este parámetro, será menos probable la propagación del árbol hacia zonas poco exploradas, lo cual tiene efecto en la calidad de las trayectorias generadas.

- Se requiere un parámetro de entrada δ_s, el cual será utilizado para evaluar si el nuevo nodo generado tiene localmente el mejor costo en una vecindad. Entre más grande sea este parámetro, mayor número de nodos se considerará para el proceso de poda.

- Se separan los nodos V de la estructura de árbol en dos subconjuntos: V_{active} y $V_{inactive}$. Los nodos en V_{active} corresponden a los vértices que tienen el mejor costo local de trayectoria desde la raíz dada una vecindad. Los nodos en $V_{inactive}$ corresponden a los nodos dominados en términos de costo de trayectoria pero cuya herencia conduce a un nodo con el mejor costo local en una vecindad, por ello, estos nodos se conservan para mantener conectividad.

- Para definir las vecindades locales, el método utiliza un conjunto auxiliar de estados conocidos como "testigos" denotado con S: para cada testigo $s \in S$, solo un nodo del árbol representará a dicho testigo, y este nodo tendrá el mejor costo de trayectoria desde la raíz dentro de un radio δ_s de distancia del testigo s. Dicho de otra forma, todos los nodos generados dentro de una distancia δ_s del testigo s tendrán un costo peor al del representante (denotado como $s.rep$) y serán removidos de V_{active}.

En el algoritmo 4 se presenta una breve descripción del *SST*. En los puntos siguientes de esta sección se describen los procesos de selección, propagación y poda

para este método.

Algorithm 4 SST(N, δ_{BN}, δ_s)

Ensure: G

$V_{active} = x_0; V_{inactive} = \emptyset;$
$G = (V = V_{active} \cup V_{inactive}, E);$
$s_0 \leftarrow x_0; s_{0,rep} = x_0; S = s_0;$
for $i, \ldots, N$ **do**
    $x_{select} \leftarrow$ Best_First_Selection($\mathcal{X}_{free}, V_{active}, \delta_{BN}$);
    $x_{new} \leftarrow$ Propagation(x_{select});
    if Is_Locally_the_Best(x_{new}, δ_s) **then**
        $V_{active} \leftarrow V_{active} \cup \{x_{new}\};$
        $E \leftarrow E \cup (x_{select}, x_{new});$
        Prune_Dominated_Nodes(x_{new});
    end if
end for

3.2.1. Selección

Al inicio de cada iteración, se utiliza el conjunto de nodos activos V_{active} para el proceso de selección mediante la aplicación del algoritmo 5. Al igual que en los enfoques clásicos de planificación probabilística, en este proceso se promueve la selección de nodos en secciones de $\mathcal{X}_{free}$ poco exploradas, pero localmente se seleccionarán los nodos que corresponden a una mejor ruta desde la raíz.

Inicialmente se obtiene una muestra aleatoria $x_{rand} \in \mathcal{X}_{free}$ en donde se centrará una bola de radio δ_{BN} (figura 3.1); todos los nodos activos del árbol que se encuentren dentro de este vecindario se almacenarán en un arreglo X_{near}. Para el proceso de propagación se seleccionará el nodo en X_{near} que tenga la mejor ruta desde la raíz, i.e. el nodo de menor costo acumulado en el vecindario. En caso de no existir un nodo activo dentro del vecindario local, se devuelve el nodo de V_{active} más cercano a x_{rand}.

Algorithm 5 Best_First_Selection($\mathcal{X}_{free}$, V_{active}, δ_{BN})

$x_{rand} \leftarrow$ Sample($\mathcal{X}_{free}$);
$X_{near} \leftarrow$ Near($V_{active}, x_{rand}, \delta_{BN}$);
if $X_{near} == \emptyset$ **then**
    return Nearest(V_{active}, x_{rand});
else
    return $\arg\min_{x \in X_{near}} cost(x)$
end if

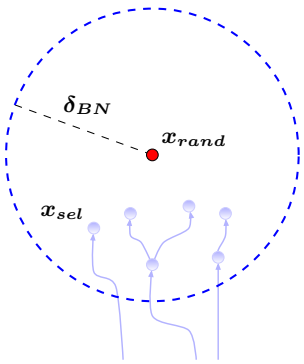


Figura 3.1: Parámetro de selección δ_{BN}.

3.2.2. Propagación

El proceso de propagación se realiza a través del muestreo aleatorio de controles $u \in \mathcal{U}$ así como del tiempo de aplicación de los mismos $t_{rand} \in [0, T_{prop}]$. Estos son aplicados mediante la integración directa de la ecuación de transición de estados (3.1), tomando como estado actual el nodo obtenido en el proceso de selección x_{select}.

Algorithm 6 Propagation(x_{select})

$t_{rand} \leftarrow$ Sample($[0, T_{prop}]$);
$u \leftarrow$ Sample($\mathcal{U}$);
$x_{new} \leftarrow \int_0^{t_{rand}} f(x(t), u)\, \mathrm{d}t + x_{select}$;
return x_{new};

En el algoritmo 7 se describen las condiciones para considerar la inclusión del nodo

recién propagado x_{new} a la estructura de árbol. Primeramente, se calcula el testigo más cercano $s_{new} \in S$ a x_{new}. Si s_{new} se encuentra alejado de x_{new} por una distancia mayor a δ_s, entonces la muestra x_{new} se convertirá en un nuevo testigo. El representante del testigo s se almacena en la variable x_{peer}. Se considera viable la adición del nuevo nodo x_{new} al árbol si al menos una de las siguientes dos condiciones se cumplen: (i) el costo de la nueva muestra $cost(x_{new})$ es inferior al costo del representante $cost(x_{peer})$ (ii) no existe un representante x_{peer}, es decir, la muestra x_{new} se ha añadido como testigo. El cumplimiento de las condiciones (i) y (ii) se ilustra en las figuras 3.2 y 3.3 respectivamente.

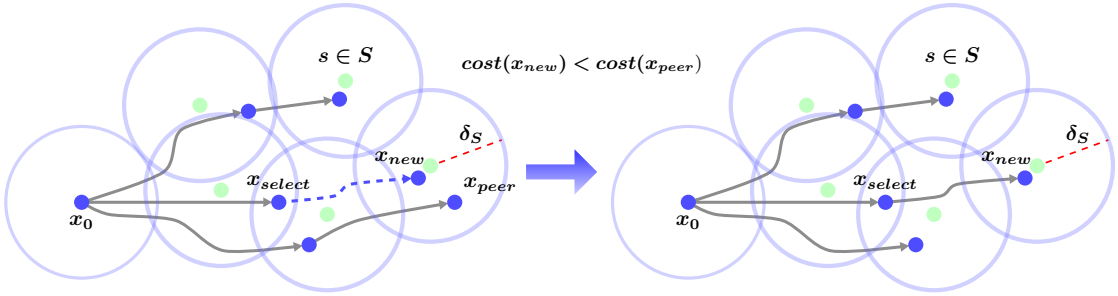


Figura 3.2: Los testigos $s \in S$ son representados de color verde, mientras que los nodos del árbol se identifican de color azul. La propagación se efectúa cuando $cost(x_{new}) < cost(x_{peer})$. El nodo x_{peer} se elimina dado el proceso de poda y el vértice propagado se agrega al árbol.

Algorithm 7 Is_Locally_the_Best (x_{new}, δ_s)

$s_{new} \leftarrow \text{Nearest}(S, x_{new})$;
if $\|x_{new} - s_{new}\| > \delta_s$ **then**
    $S \leftarrow S \cup \{x_{new}\}$;
    $s_{new} \leftarrow x_{new}$;
    $s_{new}.rep \leftarrow NULL$;
end if
$x_{peer} \leftarrow s_{new}.rep$;
if $x_{peer} == NULL$ or $cost(x_{new}) < cost(x_{peer})$ **then**
    return True;
end if
return False;

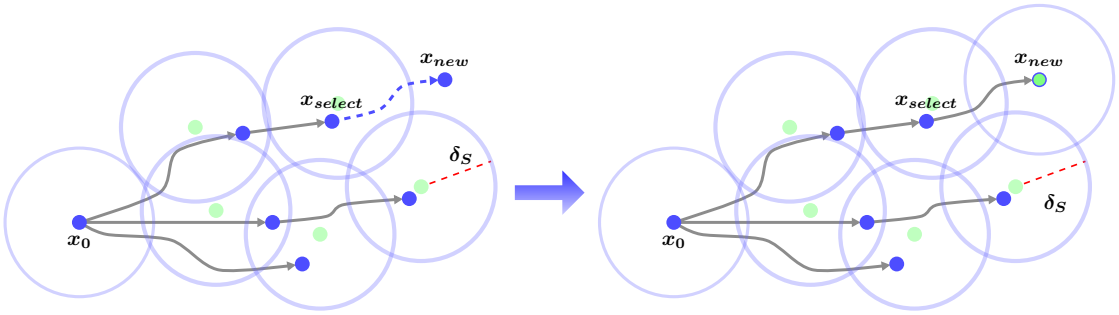


Figura 3.3: Propagación cuando x_{new} está lejos de un testigo (i.e. fuera de una vecindad). La arista (x_{select}, x_{new}) se añade al árbol y se agrega un testigo a S en la posición de x_{new}.

Una vez que se ha cumplido cualquier de las dos condiciones, se agrega el nodo x_{new} a la estructura de árbol y al conjunto de nodos activos V_{active}, caso contrario, se ignora el proceso de propagación sin que exista una modificación en el árbol.

3.2.3. Poda

Gracias al proceso de poda, el *SST* se convierte en una estructura de datos dispersa y acortada, ya que evita almacenar de manera indefinida cada nodo generado según crece el número de iteraciones.

El proceso de poda para nodos dominados se describe en el algoritmo 8. Este toma lugar una vez que se agrega un nuevo nodo x_{new} a la estructura de árbol. En un inicio se identifica el testigo del nuevo nodo s_{new} y su antiguo representante x_{peer}. El antiguo representante, el cual ha sido dominado por x_{new} en términos de costo, se remueve del conjunto de nodos activos V_{active} y se añade al conjunto de nodos inactivos $V_{inactive}$. Luego, x_{new} reemplaza a x_{peer} como el nuevo representante del testigo más cercano s. Si x_{peer} es un nodo hoja (no tiene descendencia), este será removido de la estructura de árbol.

La eliminación de x_{peer} puede desencadenar un efecto cascada para sus nodos ancestros; estos se eliminarán del árbol si se encuentran dentro del conjunto $V_{inactive}$, pues su función únicamente era la de conducir un nodo activo hacia x_{peer}. El efecto cascada termina una vez que se identifique un ancestro de x_{peer} que se encuentre en V_{active}.

Algorithm 8 Prune_Dominated_Nodes(x_{new})

$s_{new} \leftarrow \text{Nearest}(S, x_{new})$;
$x_{peer} \leftarrow s_{new}.rep$;
if $x_{peer} != NULL$ **then**
    $V_{active} \leftarrow V_{active} \setminus \{x_{peer}\}$;
    $V_{inactive} \leftarrow V_{inactive} \cup \{x_{peer}\}$;
end if
$s_{new}.rep \leftarrow x_{new}$;
while Is_Leaf(x_{peer}) and $x_{peer} \in V_{inactive}$ **do**
    $x_{parent} \leftarrow \text{Parent}(x_{peer})$;
    $E \leftarrow E \setminus (x_{parent}, x_{peer})$;
    $V_{inactive} \leftarrow V_{inactive} \setminus \{x_{peer}\}$;
    $x_{peer} \leftarrow x_{parent}$;
end while

3.3. SST*

La versión estándar del planificador *SST* no es asintóticamente óptima, principalmente debido al proceso de poda de tamaño fijo, el cual acota el tamaño del árbol. Una alternativa consiste en reducir lentamente los parámetros de selección δ_{BN} y poda δ_s: cuando δ_{BN} tiende a cero, se considera un número menor de nodos en el proceso de selección y se favorece la exploración del espacio de estados, haciendo que x_{select} se seleccione de manera aleatoria y uniforme entre los nodos activos que se encuentren en el árbol. Por otro lado, cuando δ_s tiende a cero se aumenta el tamaño del árbol, dando lugar a trayectorias de mejor calidad respecto al costo pero se hace más difícil la operación de poda, hasta el punto en donde se elimina este proceso y es posible generar todos los estados libres de colisión de $\mathcal{X}_{free}$.

El método *SST**, descrito en el algoritmo 9, sigue la idea anterior mediante la ejecución repetida del *SST* en una misma estructura de árbol, es decir, cada vez que se ejecuta el *SST* se continua el proceso de planificación con el árbol que se devolvió en la ejecución anterior. Adicionalmente, se introduce un proceso que reduce

los parámetros δ_{BN} y δ_s, mientras que se aumenta el número de iteraciones del *SST* en su siguiente ejecución. $\xi \in (0, 1)$ es una constante de reducción, en tanto que d y l son las dimensiones del espacio de estados y controles respectivamente. Reducir los parámetros de entrada en cada ejecución del *SST* permite que el método *SST** cumpla con la propiedad de optimalidad asintótica.

Algorithm 9 SST*$(n, N_0, \delta_{BN_0}, \delta_{s_0}, \xi)$

1: $N \leftarrow N_0$; $\delta_{BN} \leftarrow \delta_{BN_0}$; $\delta_s \leftarrow \delta_{s_0}$
2: **for** $j = 1, \ldots, n$ **do**
3: SST$(N, \delta_{BN}, \delta_s)$
4: $\delta_s \leftarrow \xi \cdot \delta_s$; $\delta_{BN} \leftarrow \xi \cdot \delta_{BN}$;
5: $N \leftarrow (1 + \log j) \cdot \xi^{-(d+l+1)j} \cdot N_0$
6: **end for**

3.4. MPNet

Recientemente se ha aprovechado el uso de las técnicas de planificación basada en muestreo para su uso en conjunto con métodos de aprendizaje profundo. Los autores de [16, 34] presentan un modelo de planificación conocido como "Motion Planning Networks" (MPNet) cuyo funcionamiento consiste en el entrenamiento de redes neuronales profundas [35].

3.4.1. Estructura

MPNet está conformado por dos componentes: una red de codificación y una red de planificación. La red de codificación (figura 3.4), denominada como "Enet", aprende a codificar el espacio de trabajo a partir de las nubes de puntos generadas por cada obstáculo del ambiente hasta comprimirlas en un espacio latente.

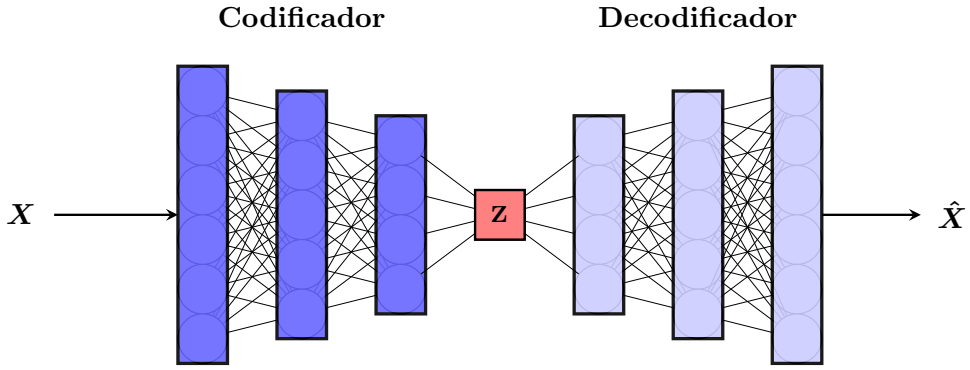


Figura 3.4: Autocodificador. X contiene la nube de puntos original mientras que $\hat{X}$ es una reconstrucción de los obstáculos del espacio de trabajo.

Enet toma la forma de un autocodificador contractivo (CAE) [36] donde la función de pérdida a optimizar se define como

$$l_{AE}(\theta^e, \theta^d) = \frac{1}{N_{obs}} \sum_{x \in D_{obs}} \|x - \hat{x}\|^2 + \lambda \sum_{ij} (\theta^e_{ij})^2, \tag{3.5}$$

donde θ^e y θ^d son los parámetros del codificador y decodificador respectivamente, λ es un coeficiente de penalización, D_{obs} es el conjunto de datos de la nube de puntos x de N_{obs} espacios de trabajo diferentes, y $\hat{x}$ es la reconstrucción obtenida por el decodificador.

La red de planificación, "Pnet", aprende a predecir la configuración del robot en un paso de tiempo $\hat{c}_{t+1}$ considerando la configuración actual c_t, la configuración objetivo c_T, y espacio latente Z obtenido mediante la codificación del espacio de trabajo. Pnet tiene la forma de una red neuronal profunda prealimentada (*feed-forward*) con parámetros θ^p. Para entrenar la red Pnet, se emplea un conjunto de datos de trayectorias $\sigma = \{c_1, c_2, \cdots, c_T\}$, las cuales pueden ser obtenidas utilizando cualquier método de planificación clásico, por ejemplo, un planificador basado en muestreo. La función objetivo de entrenamiento es la pérdida de error cuadrático medio entre la predicción $\hat{c}_{t+1}$ y la configuración actual c_t, esto es,

$$l_{Pnet}(\theta^p) = \frac{1}{N_p} \sum_{j}^{\hat{N}} \sum_{i=0}^{T_j - 1} \|\hat{c}_{j,i+1} - c_{j,i+1}\|^2, \tag{3.6}$$

donde N_p es un factor para promediar, T_j es la longitud de la trayectoria j, y $\hat{N}$ es el total de caminos para el entrenamiento.

MPNet es capaz de generar trayectorias libres de colisiones; las redes Enet y Pnet son acopladas para realizar la planificación tomando una heurística de generación de trayectorias de manera incremental y bidireccional (figura 3.5).

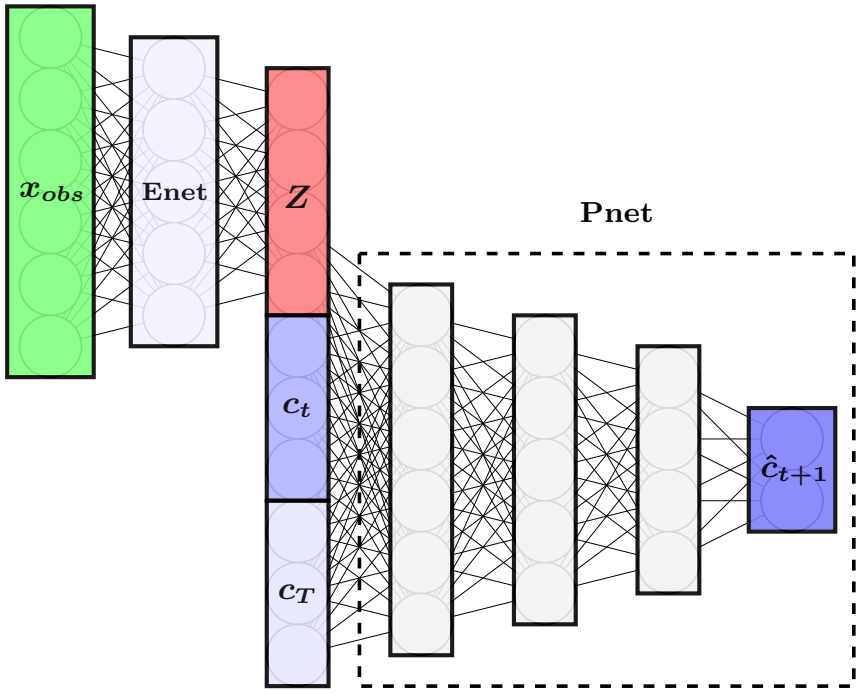


Figura 3.5: Arquitectura de MPNet.

Enet toma la información de cada obstáculo x_{obs} y la codifica en un elemento de espacio latente Z. Pnet es un modelo estocástico el cual durante su uso emplea *desconexión* (*dropout* en inglés) en la mayoría de sus capas. Las capas con desconexión [37] desactivan sus neuronas con una probabilidad $p \in [0, 1]$, por ello, cada vez que MPNet utilice la red de planificación, debido a la desconexión, se obtendrá un submodelo de

la red original dando lugar a un comportamiento estocástico. La entrada de la red Pnet es un vector concatenado del espacio latente Z, la configuración actual del robot $\hat{c}_t$, y la configuración objetivo c_T. La salida es la configuración $\hat{c}_{t+1}$ correspondiente al siguiente paso de tiempo, lo cual posiciona al robot más cerca del objetivo. Pnet se ejecuta de manera iterativa, es decir, el nuevo estado $\hat{c}_{t+1}$ se convierte el estado actual $\hat{c}_t$ durante el siguiente paso de tiempo, por lo que el camino se genera de manera incremental.

3.4.2. Estrategia de planificación

En [16] se mencionan las siguientes características de la estrategia de planificación propuesta.

- Planeación bidireccional: El algoritmo tiene un comportamiento bidireccional, es decir, se plantea la planificación hacia adelante, de configuración inicial a final, y a su vez inicia la planificación hacia atrás, de configuración final a inicial, hasta que ambos caminos se encuentren.

- Planificación recursiva: El algoritmo se ejecuta de manera recursiva y soluciona el problema de planificación utilizando la estrategia divide y vencerás. MPNet inicia con una planificación global, resultando en configuraciones libres de colisión que serán importantes para generar una trayectoria factible. Si alguna de estas configuraciones consecutivas no son conectables, MPNet las tomará como una nueva configuración inicial y final. Esta estrategia permite partir desde un problema global haciendo que la red de planificación se ejecute recursivamente, lo que genera problemas más sencillos de resolver hasta eventualmente encontrar una solución.

El algoritmo empleado en la estrategia de planificación está compuesto los siguientes elementos:

Planificador neuronal bidireccional (BNP): El BNP toma la representación del entorno Z, la configuración inicial del robot c_{init}, y la configuración objetivo c_{goal}

como entradas. El camino bidireccional es generado tal que dos caminos, de inicio a fin (σ^a) y de fin a inicio (σ^b), se construyen de manera incremental. Los caminos σ^a y σ^b se inicializan en la configuración inicial c_{init} y final c_{goal} respectivamente. Los caminos se expanden de manera alternada, es decir, para cierta iteración i, el camino σ^a es extendido, por lo que en la siguiente iteración, el camino σ^b será extendido. Esto se logra cambiando los roles de σ^a y σ^b al final de cada iteración. Al final de cada instancia de expansión, el planificador intentará conectar ambos caminos a través de una línea recta en caso de ser posible. El BNP regresa un camino concatenado σ conformado por las configuraciones de σ^a y σ^b.

Algorithm 10 BNP(c_{init}, c_{goal}, Z)

$\sigma^a \leftarrow \{c_{init}\}, \sigma^b \leftarrow \{c_{goal}\}$;
$\sigma \leftarrow \emptyset$;
for $i = 0, \ldots, n$ **do**
    $c_{new} \leftarrow \text{Pnet}(Z, \sigma^a_{end}, \sigma^b_{end})$;
    $\sigma^a \leftarrow \sigma^a \cup \{c_{new}\}$;
    Connect $\leftarrow$ steerTo($\sigma^a_{end}, \sigma^b_{end}$);
    if Connect **then**
        $\sigma \leftarrow$ Concatenate(σ^a, σ^b);
        return σ;
    end if
    SWAP(σ^a, σ^b);
end for
return $\emptyset$;

Replanificador: El BNP tendrá como salida un camino extenso de puntos críticos σ. Si todos los nodos consecutivos de σ son conectables, esto es, los caminos que pasan por estos se encuentran en el espacio libre de colisiones, entonces no será necesaria la etapa de replanificación. Sin embargo, si existe un nodo no conectable (nodo guía) en σ, se iniciará una etapa de replanificación (descrita en Algoritmo 11). Dada una trayectoria σ compuesta por las configuraciones $\sigma = \{c_0, c_1, \cdots, c_T\}$, el algoritmo utiliza una función de unión (steerTo) que itera sobre cada par σ_i y σ_{i+1} en σ para

verificar si existe una trayectoria en línea recta libre de colisiones entre estos. Si no se encuentra un camino libre de colisiones, una nueva trayectoria será determinada a través de un método de replanificación. Para replanificar, los nodos guía se utilizarán como un nuevo par de configuración inicial y final. Se proponen los siguientes dos métodos de replanificación:

- *Replanificador Neuronal (NP)*: El NP toma los nodos guía y realiza un número preestablecido de intentos (N_r) para encontrar una solución entre ellos usando el BNP.

- *Replanificador Híbrido (HP)*: HP combina el NP y un planificador clásico auxiliar (planificador oráculo). El HP utiliza los nodos guía e intenta encontrar una solución usando el NP. Si el NP falla después de un número N_r de intentos, un planificador auxiliar será utilizado para encontrar una solución, si existe, entre el par de configuraciones dadas.

Contracción Ingenua de Estados (LSC): Este proceso a veces es señalado como "shortcutting" [38]. Se implementa como una función recursiva que se ejecuta cuando en un camino $\sigma = \{c_0, c_1, \cdots, c_T\}$ es posible eliminar nodos redundantes mediante la conexión directa de dos nodos no consecutivos.

Conducción (steerTo): En esta función se navega a través de una línea recta que conecta un par de nodos para validar si la trayectoria se encuentra completamente en el espacio libre de colisiones. Para esto, la función discretiza la línea recta en puntos intermedios y se verifica que cada nodo discretizado esté libre de colisión.

Los resultados presentes en [16] han mostrado que la planificación de MPNet puede generalizarse a entornos diferentes a los utilizados en la etapa de entrenamiento. Además, MPNet presenta tiempos de cómputo que son consistentemente inferiores a los métodos de planificación de movimientos presentes en el estado del arte.

Algorithm 11 Replan(σ, Z)

Require: plan_oracle

$\sigma_{new} \leftarrow \emptyset$;
for $i = 0, \ldots, size(\sigma) - 1$ **do**
  if steerTo(σ_i, σ_{i+1}) **then**
    $\sigma_{new} \leftarrow \sigma_{new} \cup \{\sigma_i, \sigma_{i+1}\}$;
  else
    if plan_oracle **then**
      $\sigma' \leftarrow$ Oracle_Planner(σ_i, σ_{i+1})
    else
      $\sigma' \leftarrow$ BNP($\sigma_i, \sigma_{i+1}, Z$)
    end if
    if σ' **then**
      $\sigma_{new} \leftarrow \sigma_{new} \cup \sigma'$
    else
      return $\emptyset$;
    end if
  end if
end for

3.5. MPC-MPNet

En [2] se presenta un acercamiento para la solución del problema de planificación pen sistemas dinámicos utilizando como base la red de planificación neuronal *MPNet*. En este caso, se abandona la idea de expansión bidireccional y replanificación, ya que estos procesos pueden conducir a caminos no factibles.

La red *MPC-MPNet* genera iterativamente trayectorias mediante conducción local para construir caminos libres de colisión que conectan un estado inicial y un estado objetivo. Esta cuenta con un generador neuronal, un discriminador y dos algoritmos de planificación nombrados como *MPC-MPNetPath* y *MPC-MPNetTree*.

3.5.1. Construcción

En *MPC-MPNet* se incluye un codificador de observación que comprime la información del espacio de trabajo representado mediante un mapa de voxeles v, a una estructura de características latentes Z. El generador G con parámetros θ_g es un modelo neuronal estocástico que predice estados intermedio $\hat{x}_{t+1}$ dado una codificación Z, el estado actual del robot $x_t \in \mathcal{X}_{free}$ y un estado objetivo $x_{goal} \in \mathcal{X}_{goal}$, esto es,

$$\hat{x}_{t+1} \leftarrow G(Z, x_t, x_{goal}; \theta_g). \tag{3.7}$$

El generador neuronal utiliza *desconexión* en la mayoría de sus capas ocultas para generar muestras estocásticas. Tanto el generador como el codificador de observación son entrenados en conjunto mediante la optimización de la siguiente función de error cuadrático medio entre los estados predichos $\hat{x}$ y los estados de las trayectorias de demostración x^*:

$$L_{G_{\theta_g}} = \frac{1}{N_p} \sum_{i=0}^{N} \frac{1}{T_i} \sum_{j=0}^{T_1-1} \left\| \hat{x}_{i,j+1} - x^*_{i,j+1} \right\|^2, \tag{3.8}$$

donde N_p es el número total de trayectorias y T_i es la longitud de cada camino i en el conjunto de datos.

Debido al comportamiento estocástico del generador, las predicciones usualmente se dispersan en dirección al objetivo. Por ello, para seleccionar el mejor estado de un conjunto dado, se introduce una red de discriminación que predice el costo del tiempo de alcance (*time-to-reach*) de cierto estado hacia el objetivo. El discriminador D, tiene parámetros θ_d, y toma como entrada la codificación del entorno Z, así como los estados x_t y x_{goal}.

$$\hat{d}_t \leftarrow D(Z, x_t, x_{goal}; \theta_d), \tag{3.9}$$

El discriminador neuronal se entrena para disminuir el error cuadratico medio entre

el costo predicho y el costo real

$$L_{D_{\theta_d}} = \frac{1}{N_p} \sum_{i=0}^{N} \frac{1}{T_i} \sum_{j=0}^{T_i-1} \left\| \hat{d}_{i,j+1} - \left(\sum_{k=j+1}^{T_i} d^*_{i,k} \right) \right\|^2, \tag{3.10}$$

donde $d^*_{i,k}$ es el costo para llegar al objetivo desde el estado k en la trayectoria i.

Para satisfacer las restricciones dinámicas, se utiliza un modelo de predicción de controles (*Model Predictive Control, MPC*) como función de direccionamiento. El *MPC* toma el estado actual x_t y un estado objetivo $\hat{x}_{t+1}$ para generar una trayectoria optimizada $\sigma_t = [(x, y, \tau)_t]$ que minimiza el costo entre el estado propagado y $\hat{x}_{t+1}$. El algoritmo 12 describe el funcionamiento del *MPC*. Usando muestras de control y duración desde una distribución parametrizada $(u, \tau)_i \sim \mathcal{D}(u, \tau, \theta_{mpc})$, se genera una nueva trayectoria σ_i desde un estado inicial x_t. Estas propagaciones son clasificadas por medio de una función de costo definida por $d_s = d(\sigma_i, \hat{x}_{t+1}) + d_c(\sigma_i)$, donde $d(\cdot)$ es la métrica de distancia entre los estados dados y $d_c(\cdot)$ es una función de penalización por colisión. La función de costo selecciona las mejores muestras, es decir, los estados propagados con el menor costo, las cuales son utilizadas para actualizar los parámetros θ_{mpc} mediante la minimización de la pérdida de entropía cruzada (*CEM*) entre las distribuciones de los estados propagados y el estado objetivo $\hat{x}_{t+1}$.

Algorithm 12 MPC$(x_t, \hat{x}_{t+1})$

Iniciar parámetros θ_{mpc};
for $iter, \cdots, N_{iter}$ **do**
    for $u_i, \tau_i \sim \mathcal{D}(u, \tau; \theta_{mpc})$ **do**
        Propagar x_t con u_i, τ_i para generar σ_i;
        Evaluar $d_{si} = d(\sigma_i, \hat{x}_{t+1}) + d_c(\sigma_i)$;
        Seleccionar mejores muestras respecto a su puntaje;
        Actualizar θ_{mpc} con las mejores muestras;
        Actualizar la trayectoria óptima con el mejor σ^*_i;
    end for
end for
return σ^*_i

Se hace el uso de computo paralelo para las redes neuronal expuestas en este método, de esta manera, es posible generar un lote de $N_B \in \mathbb{N}$ muestras de manera simultanea. La siguiente notación denota las entradas vectorizadas en forma de lotes

$$B_t = \begin{bmatrix} x_t^1 \\ x_t^2 \\ \vdots \\ x_t^{N_B} \end{bmatrix}, \quad B_{goal} = \begin{bmatrix} x_{goal} \\ x_{goal} \\ \vdots \\ x_{goal} \end{bmatrix}, \quad B_Z = \begin{bmatrix} Z \\ Z \\ \vdots \\ Z \end{bmatrix}, \tag{3.11}$$

donde B_t, B_{goal} y B_Z corresponden a los lotes del estado actual, estado objetivo y codificación del entorno respectivamente.

3.5.2. Algoritmos

En [2] se proponen dos algoritmos de planificación tomando en consideración los elementos anteriormente descritos mediante la formación de una estructura de árbol que se expande de manera unidireccional.

Primer método: MPC-MPNetPath

En este método (algoritmo 13) se comienza a generar un lote de nuevas muestras utilizando el generador G. Con el uso del discriminador D, se selecciona la muestra x_{t+1} del lote B_t que tenga el mejor costo $\hat{d}$ para llegar al objetivo x_{goal}. El MPC toma el nodo actual x_t y selecciona el siguiente estado $\hat{x}_{t+1}$ para la ejecutar el proceso de propagación, obteniendo una trayectoria local σ_t. El estado terminal de σ_t da como resultado un estado válido x_{t+1} despues de la aplicación de u desde el estado x_t por una duración τ. La trayectoria local es añadida al árbol si es valida, caso contrario, se selecciona un nodo aleatorio en el árbol que será considerado como el nuevo estado x_t para la siguiente iteración. Una vez alcanzado el objetivo, se extrae el camino que conecta los estados x_{init} y x_{goal}.

Algorithm 13 MPC-MPNetPath(Z, x_{init}, x_{goal})

$T \leftarrow \{x_{init}\}$, $B_t \leftarrow x_{init}$;
$B_Z \leftarrow Z$, $B_{goal} \leftarrow x_{goal}$;
for $i, \cdots, n$ **do**
    $\hat{B}_{t+1} \leftarrow \mathrm{G}(B_Z, B_t, B_{goal}; \theta_g)$;
    $\hat{x}_{t+1} \leftarrow \arg\min_{\hat{x}_{t+1}} \mathrm{D}(B_Z, \hat{B}_{t+1}, B_{goal}; \theta_d)$;
    $\sigma_t \leftarrow \mathrm{MPC}(x_t, \hat{x}_{t+1})$;
    if Invalid(σ_t) **then**
        $B_t \leftarrow$ Random_Node(T);
    else
        Add_To_Tree(σ_t, T);
        Asignar B_t con estado términal de σ_t;
    end if
    if Reached(T, x_{goal}) **then**
        return ExtractPath(T);
    end if
end for
return $\emptyset$

Segundo método: MPC-MPNetTree

En el algoritmo 14 se presenta brevemente el segundo método de planificación. En cada iteración, se muestrea un lote de estados aleatorios B_{rand} y se encuentran sus vecinos más cercanos dentro del árbol. Estos nodos cercanos se consideran como el lote de estados actuales B_t, el cual será utilizado por el generador y posteriormente por el *MPC*. Las trayectorias locales que sean válidas son agregadas en la estructura de árbol hasta que exista una trayectoria que conduzca al objetivo.

Algorithm 14 MPC-MPNetTree(Z, x_{init}, x_{goal})

$T \leftarrow \{x_{init}\}$, $B_t \leftarrow x_{init}$;
$B_Z \leftarrow Z$, $B_{goal} \leftarrow x_{goal}$;
for $i, \cdots, n$ **do**
    $B_{rand} \leftarrow$ Random_Sample();
    $B_t \leftarrow$ Nearest_Neighbor(B_{rand}, T);
    $\hat{B}_{t+1} \leftarrow \mathrm{G}(B_Z, B_t, B_{goal}; \theta_g)$;
    $B_{\sigma_t} \leftarrow \mathrm{MPC}(B_t, \hat{B}_{t+1})$;
    addToTree(B_{σ_t}, T);
    if Reached(T, x_{goal}) **then**
        return Extract_Path(T);
    end if
end for

Capítulo 4

Propuesta de método

En este trabajo se presenta un método que emplea modelos neuronales para resolver el problema de planificación de movimientos con restricciones cinemáticas y dinámicas. Durante este capítulo se da una breve explicación del funcionamiento del método propuesto así como de las arquitecturas de las redes utilizadas. Adicionalmente, se detallan las condiciones y los parámetros de los modelos neuronales para la obtención de los resultados expuestos en el capítulo 6.

4.1. Metodología

Dado un sistema de movimiento de la forma (3.1), se genera una base de datos conformada por los estados y controles obtenidos mediante un algoritmo de planificación para sistemas dinámicos. Estos datos son utilizados para el entrenamiento de redes secuenciales.

Similar al planteamiento de *MPNet* [16, 34], se propone el uso de dos redes neuronales acopladas: la primera red es un autocodificador contractivo (CAE), la cual toma como entrada la nube de puntos P del espacio de trabajo y cuya función de pérdida a optimizar está definida en (3.5).

La segunda red, es una *red de inferencia de controles* que toma como entrada un vector concatenado por el espacio latente Z obtenido mediante el codificador, el estado actual del sistema x_i y el estado objetivo x_{goal}. La salida está conformada por

los controles y tiempos de aplicación de estos para conducir al sistema dinámico desde el estado x_i hacia un estado más cercano del objetivo x_{i+1}. La arquitectura de la red acoplada se muestra en la figura 4.1.

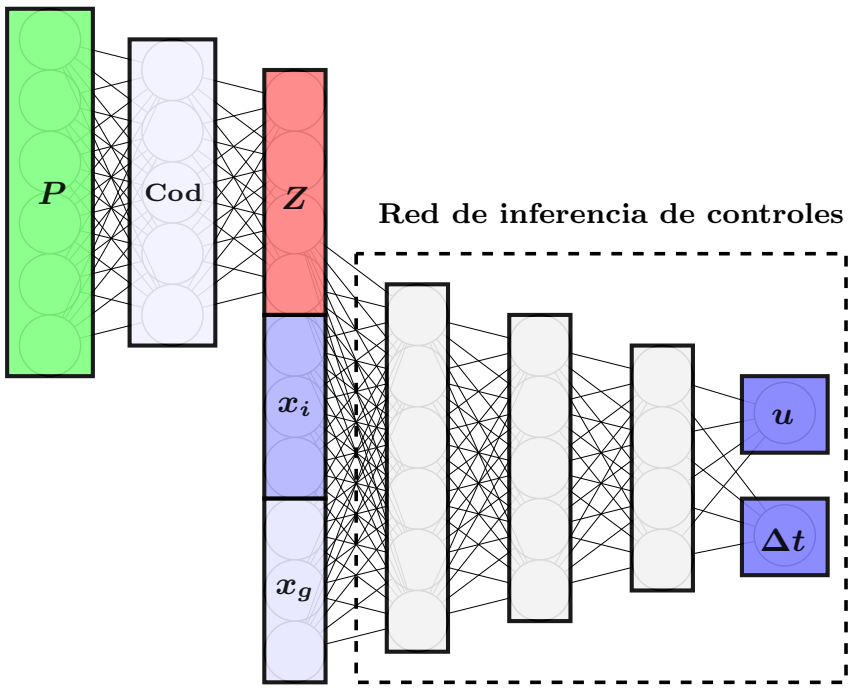


Figura 4.1: Red de inferencia propuesta. La arquitectura es similar a MPNet pero las *configuraciones* son sustituidas por *estados*. Se obtienen controles y tiempos de aplicación.

Para el entrenamiento del codificador se pueden utilizar diferentes espacios de trabajo. Los obstáculos de cada entorno se convierten nubes de puntos que serán reconstruidas mediante el decodificador. La red de autocodificación aprende de entornos de prueba para su reconstrucción, así como para lograr generar entornos únicos.

El entrenamiento de la *red de inferencia de controles* hace uso de los datos generados por un planificador de movimientos: para cada trayectoria obtenida mediante el algoritmo de planificación, se registran los estados x_i por los cuales pasa el robot, así como las entradas de control u_i y tiempos de aplicación Δt_i que se utilizaron en cada estado x_i para llegar al estado siguiente x_{i+1}. Los datos de entrenamiento están entonces conformados por el conjunto de estados y controles (junto con los tiempos de aplicación) que se obtienen del planificador para diferentes trayectorias. La función

objetivo de entrenamiento corresponde a la pérdida de error cuadrático medio entre los controles-tiempos inferidos por el modelo neuronal $(\hat{u}, \Delta\hat{t})_i$ y los controles-tiempos $(u, \Delta t)_i$ utilizados desde x_i para alcanzar x_{i+1} dada una trayectoria σ_j.

$$l(\theta) = \frac{1}{N_p} \sum_{j}^{\hat{N}} \sum_{i=0}^{T_j - 1} \left\| (\hat{u}, \Delta\hat{t})_{j,i} - (u, \Delta t)_{j,i} \right\|^2. \tag{4.1}$$

donde N_p es un factor para promediar, T_j es la longitud de la trayectoria j, y $\hat{N}$ es el total de controles-tiempos para el entrenamiento.

En la etapa de entrenamiento pueden utilizarse técnicas de regularización como las regularizaciones L1 y L2 , en las cuales se agrega una término adicional a la función de pérdida para imponer una penalización a los pesos w de la red y evitar el sobreentrenamiento:

$$\tilde{l}(\theta) = l(\theta) + \lambda \|w\|_1, \tag{4.2}$$

$$\tilde{l}(\theta) = l(\theta) + \frac{\lambda}{2} \|w\|_2^2.. \tag{4.3}$$

Las ecuaciones (4.2) y (4.3) corresponden a la regularización L1 y L2 respectivamente, donde $l(\theta)$ es la función de pérdida original con parámetros θ y λ es un valor que determina la influencia de la penalización. Adicionalmente, en el entrenamiento de la red de inferencia se puede incluir *desconexión* (*dropout*) [37] en las capas ocultas, esto es. desactivar las neuronas de la capa con una probabilidad $p \in [0, 1]$, haciendo que sus pesos no sean modificados. Al igual que la regularización L1 y L2, la *desconexión* se utiliza usualmente para evitar el sobreentrenamiento.

En la etapa de inferencia, se rechaza el resultado cuando i) los controles obtenidos se encuentran fuera del dominio del conjunto de controles permisibles $u \notin \mathcal{U}$, ii) el tiempo de aplicación de los controles es inválido, por ejemplo, tiempo negativo o iii) los controles conducen a una colisión. Al rechazar el resultado de una inferencia, es conveniente el uso de la *red de inferencia de controles* con *desconexión*, ya que esto

permite el uso parcial de la red, generando controles diferentes. Esta última red se manda a llamar un cierto número de veces hasta que se obtienen entradas de control y tiempos de aplicación válidos. Caso contrario, termina el proceso de planificación sin éxito.

El proceso de extensión toma lugar al aplicar los controles y tiempos inferidos en la *red de inferencia de controles*. Esto se logra mediante integración directa de la ecuación de transición de estados del sistema dinámico estudiado. El nuevo estado x_{i+1} se toma como estado actual x_i para una nueva inferencia hasta que la distancia entre este y el estado objetivo x_{goal} sea menor a cierto umbral ϵ. Dado este caso, termina el proceso de planificación satisfactoriamente.

El algoritmo presentado en 15 describe el proceso de planificación utilizando la red de *inferencia de controles*: primeramente se almacena el estado inicial en una estructura σ que contendrá a los estados válidos de la trayectoria generada. Durante n iteraciones, se seleccionará el estado actual x_i para obtener un par control-tiempo utilizando la red entrenada. Se verifica que el control y el tiempo de aplicación inferidos sean válidos para el sistema dinámico. Mediante la solución de la ecuación de transición de estados (linea 9) se obtiene un nuevo estado x_{i+1} donde se verifica si el camino hacia este se encuentra libre de colisión. Si el nuevo estado se encuentra libre de colisión, se agrega al conjunto σ y en la siguiente iteración pasa a ser el estado actual x_i. El proceso de planificación termina si x_i está lo suficientemente cerca de x_{goal} (línea 18). En caso de no encontrarse una trayectoria de x_{init} a x_{goal}, se devuelve la trayectoria parcial σ.

Algorithm 15 Kinodynamic_Planner(x_{init}, x_{goal}, Z)

$\sigma = \{x_{init}\}$;
Trials $= K$;
for $i, \cdots, n$ **do**
    $x_i = \sigma_i$;
    $k = 0$;
    while $k <$ Trials **do**
        $u_i, \Delta t_i \leftarrow$ Control_Inference(Z, x_i, x_{goal});
        if Control_Validity($u_i, \Delta t_i$) **then**
            $x_{i+1} = \int_0^{\Delta t_i} f(x(t), u_i)\,\mathrm{d}t + x_i$;
            if Is_Collision_Free(x_{i+1}, Z) **then**
                $\sigma = \sigma \cup \{x_{i+1}\}$;
                break;
            end if
        end if
        $k = k + 1$;
    end while
    if Is_Near($x_{i+1}, x_{goal}, \epsilon$) **then**
        return σ;
    end if
end for
return σ;

4.2. Detalles de implementación

En esta sección se describen las características de la implementación de la metodología anterior para los experimentos en este trabajo en particular.

El autocodificador tiene una arquitectura compuesta por tres capas ocultas para el codificador y tres capas ocultas para el decodificador. El codificador toma como entrada la nube de puntos de los entornos descritos en el capítulo 6 y da como salida un espacio latente Z de 24 elementos. En la tabla 4.1 se muestran las propiedades de

cada capa oculta del autocodificador. La función de pérdida es optimizada mediante el optimizador *ADAM* con parámetros $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-7$ y tasa de aprendizaje de $\alpha = 0.001$.

Capa	Oculta 1	Oculta 2	Oculta 3	Oculta 4	Oculta 5	Oculta 6
Red	Cod.	Cod.	Cod.	Dec.	Dec.	Dec.
Neuronas	512	256	128	128	256	512
Activación	PReLU	PReLU	PReLU	PReLU	PReLU	PReLU
Desconexión	no	no	no	no	no	no

Tabla 4.1: Propiedades de las capas ocultas del autocodificador.

La *red de inferencia de controles* toma como entrada el espacio Z obtenido por el codificador, el estado actual x_i y el estado parcial objetivo x'_{goal}, este último únicamente consta de la posición del estado objetivo x_{goal}. Esta red contiene 3 capas ocultas; la tabla 4.2 muestra el número de neuronas que tiene cada capa que conforma la red. Se utiliza el optimizador *ADAM* para minimizar la función de pérdida. Esta toma como parámetros $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-7$ y tasa de aprendizaje de $\alpha = 5e-4$.

Capa	Oculta 1	Oculta 2	Oculta 3	Salida
Neuronas	128	64	32	3
Activación	PReLU	PReLU	PReLU	Linear
Desconexión	no	50.0 %	50.0 %	no

Tabla 4.2: Propiedades de las capas de la red de inferencia de controles.

Para la generación de los pares control-tiempo del entrenamiento se utiliza el planificador *SST**. Los parámetros con los que opera el *SST** dependen de cada experimento y son descritos en las secciones del capítulo 6.

Cada modelo de inferencia es entrenado durante 1500 épocas, en lotes de 50 elementos. Se generan modelos para cada sistema dinámico descrito en el capítulo 5.

Para el proceso de propagación, tanto en el *SST** como en la aplicación de los controles inferidos, se utiliza el método de integración númerica *regla de Simpson de 3 subintervalos*, la cual está definida como

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{3}\sum_{i=0,i=i+1}^{n-2}[f(x_i) + 4f(x_{i+1}) + f(x_{i+2})], \tag{4.4}$$

siendo h la longitud de una partición uniforme del intervalo $[a, b]$ en n subintervalos [39].

Capítulo 5

Sistemas de prueba

En este capítulo se describen los sistemas de movimiento utilizados en los experimentos de este trabajo. Para cada sistema se menciona su respectiva ecuación de transición de estados, así como su condición de no holonomía mediante la aplicación del teorema de Frobenius. La sección 5.1 presenta un modelo simplificado de robot omnidireccional. En las secciones 5.2 y 5.3 se introduce un sistema robótico de manejo diferencial con dinámica de primer y segundo orden respectivamente. Finalmente, en la sección 5.4 se presenta un modelo simple de robot automóvil.

5.1. Robot omnidireccional

Los robot omnidireccionales usualmente utilizan una base con ruedas cuya instalación les permite modificar su dirección de movimiento sin necesidad de modificar su orientación. Algunos autores [40, 41] han planteado diferentes modelos para lograr el movimiento omnidireccional de un sistema robótico. En esta sección se presenta un modelo simplificado de un sistema dinámico capaz de ejecutar movimientos omnidireccionales en un espacio bidimensional. Sean u_x y u_y las entradas de control que determinan la velocidad horizontal y vertical del robot. El robot omnidireccional se puede modelar de la siguiente forma

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_y. \tag{5.1}$$

5.2. Robot de manejo diferencial (DDR): primer orden

El *robot de manejo diferencial (DDR)* se caracteriza por tener dos ruedas coaxiales con motores independientes. Para la construcción del modelo que describe el movimiento de un robot *DDR*, considerando sus restricciones, se requiere la distancia $L = 2b$ que existe entre las dos ruedas y el radio de estas r. Se considera la representación que se ilustra en la figura 5.1 cuyo marco de referencia tiene origen en el centro del eje que une a ambas ruedas. Las entradas de control u_l y u_r determinan la velocidad angular de la rueda izquierda y derecha respectivamente. Si $u_l = u_r > 0$, entonces el robot se moverá hacia adelante en la dirección que apunten las ruedas. La velocidad del robot es proporcional a r. En general, si ambas velocidades angulares son iguales $u_l = u_r$, la distancia recorrida en una duración t será rtu_l, i.e., el robot únicamente se mueve de manera traslacional. Si $u_l = -u_r \neq 0$, el robot rotará en sentido horario, en su posición, pues ambas ruedas se dirigen a direcciones opuestas.

Con estas observaciones, la ecuación de transición de estados para un robot *DDR* de primer orden puede escribirse como

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(u_l + u_r)\cos\theta \\ \frac{r}{2}(u_l + u_r)\sin\theta \\ \frac{r}{2b}(u_r - u_l) \end{bmatrix}. \tag{5.2}$$

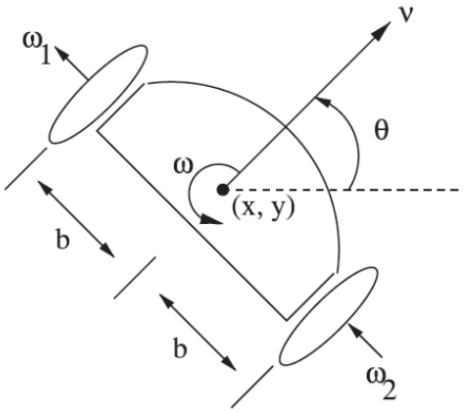


Figura 5.1: Representación de un robot de manejo diferencial, extraído de [1]. El ángulo θ denota la dirección a donde se dirige el robot, mientras que v es su velocidad lineal. ω_1 y ω_2 corresponden a la velocidad angular de la llanta izquierda y derecha respectivamente.

En ocasiones es conveniente realizar una transformación en el espacio de controles. Sea $u_\omega = (u_r + u_l)/2$ y $u_\psi = u_r - u_l$. En este caso, u_ω controla la traslación del robot, mientras que u_ψ controla su rotación. Bajo esta transformación, la ecuación de transición de estados toma la siguiente forma:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 0 \end{bmatrix} u_\omega + \begin{bmatrix} 0 \\ 0 \\ \frac{r}{2b} \end{bmatrix} u_\psi, \tag{5.3}$$

El DDR es capaz de moverse entre un par de configuraciones mediante: 1) la rotación en sitio para dirigir sus llantas en dirección a la posición objetivo, 2) trasladarse a la posición objetivo, y 3) rotar hacia la orientación deseada. La distancia total recorrida por el centro del eje del robot es la distancia Euclidiana en $\mathbb{R}^2$ entre ambas posiciones. Puesto que las rotaciones en sitio no influyen en la distancia recorrida, es conveniente establecer otra métrica de costo, tal como la cantidad o el tiempo de rotación de las ruedas [5].

El *DDR* de primer orden cumple con la propiedad de no holonomía: tomando $q = (x, y, \theta)$. Sea (5.3) un sistema afín sin deriva cuya matriz de campos vectoriales es

$$H(q) = \begin{bmatrix} r\cos\theta & 0 \\ r\sin\theta & 0 \\ 0 & \frac{r}{2b} \end{bmatrix}, \tag{5.4}$$

se genera la siguiente distribución no singular

$$\triangle = \text{span}\left\{ \begin{bmatrix} r\cos\theta & r\sin\theta & 0 \end{bmatrix}^T, \begin{bmatrix} 0 & 0 & \frac{r}{2b} \end{bmatrix}^T \right\}. \tag{5.5}$$

Sea $f = \begin{bmatrix} r\cos\theta & r\sin\theta & 0 \end{bmatrix}^T$ y $g = \begin{bmatrix} 0 & 0 & \frac{r}{2b} \end{bmatrix}^T$, y denotando los componentes de cada campo vectorial como subíndice. Al aplicar la operación de corchetes de Lie se obtiene

$$[f, g] = \begin{bmatrix} f_1\frac{\partial g_1}{\partial x} - g_1\frac{\partial f_1}{\partial x} + f_2\frac{\partial g_1}{\partial y} - g_2\frac{\partial f_1}{\partial y} + f_3\frac{\partial g_1}{\partial \theta} - g_3\frac{\partial f_1}{\partial \theta} \\ f_1\frac{\partial g_2}{\partial x} - g_1\frac{\partial f_2}{\partial x} + f_2\frac{\partial g_2}{\partial y} - g_2\frac{\partial f_2}{\partial y} + f_3\frac{\partial g_2}{\partial \theta} - g_3\frac{\partial f_2}{\partial \theta} \\ f_1\frac{\partial g_3}{\partial x} - g_1\frac{\partial f_3}{\partial x} + f_2\frac{\partial g_3}{\partial y} - g_2\frac{\partial f_3}{\partial y} + f_3\frac{\partial g_3}{\partial \theta} - g_3\frac{\partial f_3}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \frac{r^2}{2b}\sin\theta \\ -\frac{r^2}{2b}\cos\theta \\ 0 \end{bmatrix}. \tag{5.6}$$

La matriz aumentada de campos vectoriales es

$$H'(q) = \begin{bmatrix} r\cos\theta & 0 & \frac{r^2}{2b}\sin\theta \\ r\sin\theta & 0 & -\frac{r^2}{2b}\cos\theta \\ 0 & \frac{r}{2b} & 0 \end{bmatrix}. \tag{5.7}$$

Puesto que el rango de la matriz aumentada es 3 para todo $q \in \mathcal{C}$ (determinante $|H(q)| = \frac{r^4}{4b^2}$, $r, b > 0$), el sistema no es involutivo, i.e., existe un campo vectorial

obtenido mediante los corchetes de Lie que no pertenece a la distribución $\triangle$, por lo tanto, el sistema es no holonómico (teorema de Frobenius) [5].

El DDR de primer orden es localmente controlable en pequeño tiempo (STLC): sea n la dimensión del espacio de estados $\mathcal{X}$. El álgebra de Lie para un robot de manejo diferencial está dada por

$$\mathcal{L}(\triangle) = \text{span}\left\{ \begin{bmatrix} r\cos\theta & r\sin\theta & 0 \end{bmatrix}^T, \begin{bmatrix} 0 & 0 & \frac{r}{2b} \end{bmatrix}^T, \begin{bmatrix} \frac{r^2}{2b}\sin\theta & -\frac{r^2}{2b}\cos\theta & 0 \end{bmatrix}^T \right\}. \quad (5.8)$$

Puesto que $\dim \mathcal{L}(\triangle) = n = 3$, por el teorema de Chow-Rashevskii se dice que el sistema es *STLC* [5].

5.3. Robot de manejo diferencial (DDR): segundo orden

El *DDR* de segundo orden surge a partir del *DDR* de primer orden mediante la extensión de la ecuación de transición de estados. Las entradas ed control u_l y u_r corresponden a la acción de *acelerar* los motores de las ruedas, en lugar de manipular directamente sus velocidades. Sean ω_l y ω_r las velocidades angulares del motor izquierdo y derecho respectivamente. La ecuación de transición de estados se puede escribir como

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\omega_l} \\ \dot{\omega_r} \end{bmatrix} = \begin{bmatrix} \frac{\omega_r+\omega_r}{2} r\cos\theta \\ \frac{\omega_r+\omega_r}{2} r\sin\theta \\ \frac{\omega_r-\omega_l}{2b} r \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_l + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_r. \quad (5.9)$$

El DDR de segundo orden es localmente accesible en pequeño tiempo

(STLA): tomando $q = (x, y, \theta, \omega_l, \omega_r)$. De (5.9) se identifican los siguientes campos vectoriales:

$$h_0(q) = \begin{bmatrix} \frac{r}{2}(\omega_r + \omega_l)\cos\theta \\ \frac{r}{2}(\omega_r + \omega_l)\sin\theta \\ \frac{r}{2b}(\omega_r - \omega_l) \\ 0 \\ 0 \end{bmatrix}, \quad h_1(q) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad h_2(q) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \tag{5.10}$$

Para verificar si el sistema es *STLA*, el algebra de Lie $\mathcal{L}(\triangle)$, de la distribución $\triangle = \{h_1, h_2, h_3\}$ debe tener la misma dimensión que el espacio de estados $\mathcal{X}$, i.e, $n = 5$. A partir de la operación de los corchetes de Lie se definen los siguientes campos vectoriales:

$$h_3(q) = [h_2, h_0] \quad h_4(q) = [h_1, h_0], \quad h_5(z) = [h_1, [h_0, h_4]],$$

$$h_3(q) = \begin{bmatrix} \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta \\ -\frac{r}{2b} \\ 0 \\ 0 \end{bmatrix}, \quad h_4(q) = \begin{bmatrix} \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta \\ \frac{r}{2b} \\ 0 \\ 0 \end{bmatrix}, \quad h_5(q) = \begin{bmatrix} \frac{r^2}{2b}\sin\theta \\ -\frac{r^2}{2b}\cos\theta \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{5.11}$$

El resultado del determinante de la matriz aumentada $H'(q) = [h_1, h_2, h_3, h_4, h_5]$ es igual a $\frac{r^2}{4b^2}$ para $r, b > 0$. Por lo tanto, la dimensión del álgebra de Lie corresponde con la dimensión del espacio de estados $\mathcal{L}(\triangle) = n$, lo cual satisface la condición de Chow-Rashevskii para sistemas dinámicos con deriva [33].

5.4. Robot automóvil

El problema de planificación para el *robot automóvil* cuenta con restricciones diferenciales de primer orden. La figura 5.2 muestra el modelo de robot automóvil utilizado en [2]. Este mismo modelo se utiliza en los experimentos de la sección 6.6 de este trabajo.

El *robot automóvil* tiene un ángulo de direccionamiento limitado, por lo que no es capaz de moverse de manera instantánea hacia los lados. Se utiliza un modelo que solo es capaz de avanzar sin retroceder.

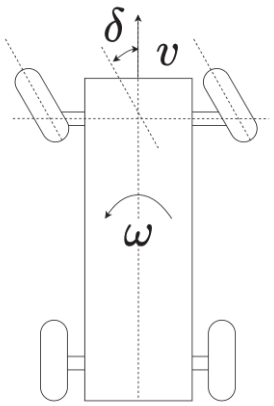


Figura 5.2: Representación de un robot tipo automóvil, extraído de [2].

Suponiendo que la magnitud de la velocidad v y el angulo de direccionamiento δ pueden ser manipulados mediante las entradas de control u_v y u_δ respectivamente. La ecuación de transición de estados para este modelo se puede escribir como

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} u_v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_\delta. \tag{5.12}$$

Se puede notar que la ecuación de transición de estados para el *robot de manejo diferencial* al aplicar la transformación en el espacio de controles 5.3 es idéntica a 5.12 (con $r = 1$ y $L = 1$). Por ello, se dice que el *robot DDR* puede simular fácilmente el movimiento de un *robot automóvil*. Adicionalmente, es fácil ver que el robot tipo

automóvil cuenta con la propiedad de **no holonomía** y es **STLC** siguiendo el mismo procedimiento que en la sección 5.2.

Capítulo 6

Experimentos

En este capítulo se presentan los resultados de algunos experimentos al utilizar la metodología propuesta en el capítulo 4.

En la sección 6.1 se reportan valores estadísticos de las trayectorias obtenidas con la red de inferencia, seleccionando estados iniciales aleatorios que se encuentran en la misma vecindad donde se ubican los estados iniciales de las trayectorias de entrenamiento. Asimismo, se hace el mismo análisis para estados iniciales aleatorios que están alejados de la vecindad de entrenamiento. Se hace uso de los primeros tres sistemas de movimiento descritos en el capítulo 5. La sección 6.2 presenta un reporte de los estadísticos correspondientes a las trayectorias obtenidas mediante modelos entrenados con un mayor número de muestras utilizando el DDR con dinámica de segundo orden. La sección 6.3 muestra el desempeño de los modelos al tratar con un pasaje estrecho. En 6.4 se reporta el efecto de variar algunos parámetros subyacentes al planificador SST* así como el intervalo del tiempo de aplicación de controles Δt. En la sección 6.5 se hace una comparación de los tiempos de planificación entre el algoritmo SST* y el método propuesto. Finalmente, en la sección 6.6 se presenta una comparación del tiempo de cómputo y costo de trayectoria entre los resultados del método MPC-MPNet y el método propuesto.

Para cada experimento, se muestra el valor promedio de cinco modelos de planificación entrenados bajo los mismos parámetros pero con diferente semilla. Se considera

como función de costo al tiempo acumulado en la aplicación de controles. Las tablas de resultados presentes en este capítulo tienen la siguiente nomenclatura:

- **Mínimo:** Menor costo en las trayectorias de ejecuciones exitosas (en segundos).
- **Máximo:** Mayor costo en las trayectorias de ejecuciones exitosas (en segundos).
- **Media:** Media aritmética del costo de trayectorias (en segundos).
- **SD:** Desviación estándar de los costos obtenidos (en segundos).
- **Éxitos:** Número de experimentos donde se resuelve el problema de planificación con un costo bajo-moderado.
- **Rechazos:** Número de experimentos donde se resuelve el problema de planificación con un costo alto.

En la tabla 6.1 se muestra el umbral en el tiempo de aplicación de controles (en segundos) para determinar si una trayectoria es de alto costo dado un entorno de trabajo: si se logra resolver el problema de planificación pero el costo de trayectoria es superior a este umbral, se considerará como rechazo, caso contrario se considera como éxito. Los valores del umbral se definen con respecto a la media aproximada en el costo de las trayectorias de los datos de entrenamiento. Los entornos de trabajo se definen en la sección 6.1.

	Primer Entorno	Segundo Entorno
Omnidireccional	35.0	40.0
DDR (1° orden)	35.0	70.0
DDR (2° orden)	35.0	40.0

Tabla 6.1: Umbral de costo (en segundos) para considerar una trayectoria como rechazo según el sistema dinámico y entorno.

En los experimentos de la secciones 6.1 a 6.5 se consideran robots tipo disco. El robot *omnidireccional* tiene las siguientes características

- **Magnitud máxima de velocidad:** $V_{max} = 3$.

- **Parte de un estado en reposo:** $V_0 = 0$.

El robot *DDR de primer orden* tiene los siguientes parámetros

- **Distancia entre el centro del robot y las ruedas:** $b = 0.71$.
- **Radio de las ruedas:** $r_{wheel} = 1$.
- **Magnitud máxima de velocidad:** $V_{max} = 3$.
- **Parte de un estado en reposo:** $V_0 = 0$, $\dot{\theta}_l = 0$, $\dot{\theta}_r = 0$.

El robot *DDR de segundo orden* cuenta con las siguientes características

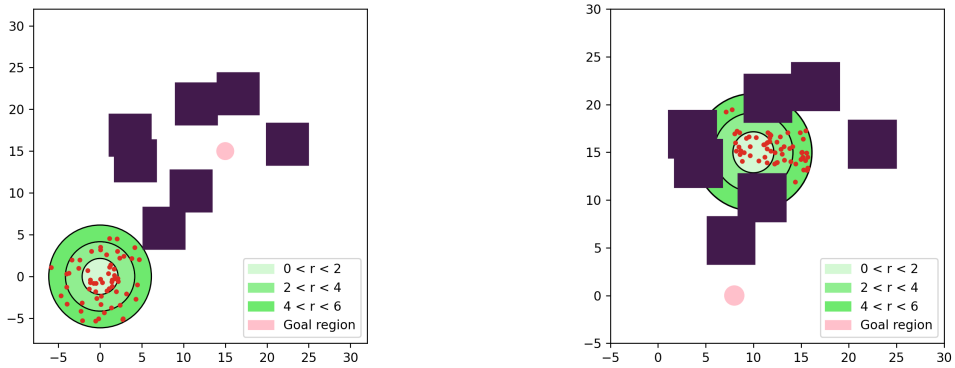
- **Distancia entre el centro del robot y las ruedas:** $b = 0.71$.
- **Radio de las ruedas:** $r_{wheel} = 1$.
- **Magnitud máxima de velocidad:** $V_{max} = 3$.
- **Parte de un estado en reposo:** $V_0 = 0$, $\dot{\theta}_l = 0$, $\dot{\theta}_r = 0$, $\dot{\omega}_l = 0$, $\dot{\omega}_r = 0$.

En la sección 6.6 se considera un robot automóvil con las siguientes características

- **Alto del robot:** $h = 5.7$
- **Ancho del robot:** $w = 7.2$
- **Magnitud máxima de velocidad:** $V_{max} = 3$.
- **Parte de un estado en reposo:** $V_0 = 0$.

6.1. Variación en los vecindarios de partida

En este experimento se utiliza la red de inferencia para obtener entradas de control que permiten conducir al sistema dinámico desde un estado inicial que se encuentra en cierta vecindad hacia una región objetivo. Para los modelos de robot omnidireccional y robot DDR de primer orden se entrenaron cinco modelos neuronales con 100, 200, 300 y 400 trayectorias, mientras que para el robot DDR de segundo orden se entrenaron cinco modelos con 200, 400, 600 y 800 trayectorias. En la figura 6.1 se muestra el primer entorno de trabajo. Las regiones en color verde (regiones "A" y "B") representan a los vecindarios en donde se obtuvieron los estados iniciales de prueba; las zonas en verde tenue ($0 < r < 2$) corresponden a las vecindades en donde se obtuvieron los estados iniciales para la obtención de trayectorias de entrenamiento. Las regiones de color verde ($2 < r < 4$) y verde fuerte ($4 < r < 6$) corresponden a las regiones en donde no se obtuvieron los estados iniciales para las trayectorias de entrenamiento de los modelos neuronales, esto con el objetivo de poner a prueba la capacidad de generalización de la red de inferencia al considerar estados iniciales en vencindarios diferentes al del entrenamiento. La zona de color rosa indica la región objetivo ($\epsilon_{goal} = 1$).



(a) Región "A". **(b)** Región "B".

Figura 6.1: Primer entorno de trabajo. Los estados iniciales de prueba se muestran como puntos de color rojo. Los obstáculos se presentan en color violeta, las vecindades de prueba en color verde y la zona objetivo en color rosa.

La figura 6.2 muestra un segundo entorno de trabajo. Para este ambiente únicamente se hicieron pruebas utilizando el robot DDR con dinámica de segundo orden.

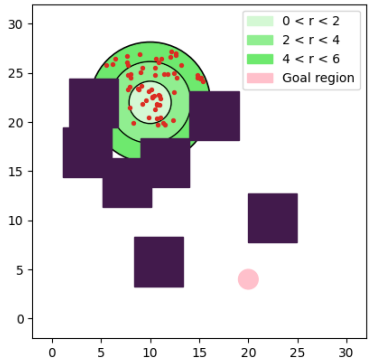


Figura 6.2: Segundo entorno de trabajo.

En cada prueba se utilizan 20 estados con posiciones aleatorias para cada vecindad. Las vecindades corresponden a la regiones en el espacio de estados libres de colisión $\mathcal{X}_{free}$ delimitadas por un conjunto de estados $R \subset \mathcal{X}_{free}$ centrados en $P_c = (x_0, y_0)$.

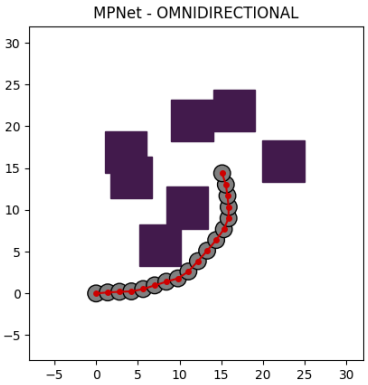
$$R = \left\{ x_{init} = (x, y, \theta, \cdots) \mid a < r < b, \quad \text{con } a, b \in \mathbb{R} \text{ y } r = \sqrt{(x - x_0)^2 + (y - y_0)^2} \right\}.$$

- Primer vecindad: Los estados en R toman valores entre $0 < r < 2$. En esta vecindad también se obtienen los estados iniciales correspondientes a las trayectorias de entrenamiento. Durante este capítulo se le nombrará ocasionalmente a esta zona como *vecindad inicial*.

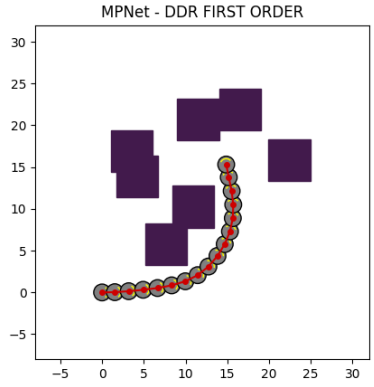
- Segunda vecindad: Los estados en R toman valores entre $2 < r < 4$. Las trayectorias de entrenamiento no incluyen estados iniciales de esta región. Ocasionalmente se le mencionará a esta región como *vecindad media*.

- Tercer vecindad: Los estados en R toman valores entre $4 < r < 6$. Al igual que en la segunda vecindad, las trayectorias de entrenamiento no incluyen estados iniciales de esta región. A esta región se le nombrará como *vecindad lejana*.

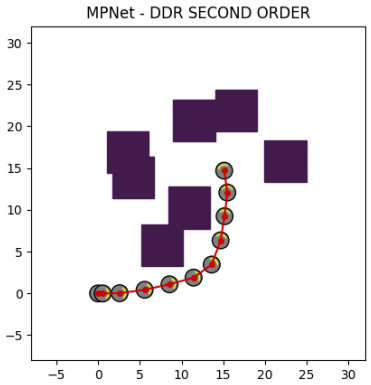
En la figura 6.3 se muestra un ejemplo de trayectoria para cada sistema dinámico de prueba partiendo desde un estado situado en la vecindad inicial de la región "A" del primer entorno de trabajo. En 6.3a se simula el comportamiento que tiene el robot omnidireccional al utilizar las entradas de control obtenidas con la red propuesta. Similarmente, en 6.3b y 6.3c se observa la trayectoria obtenida al utilizar un robot DDR con dinámica de primer y segundo orden respectivamente.



(a) Robot omnidireccional.



(b) Robot de manejo diferencial con dinámica de primer orden.



(c) Robot de manejo diferencial con dinámica de segundo orden.

Figura 6.3: Ejemplos de trayectorias generadas al resolver las ecuaciones de transición de estados para cada sistema estudiado, se utilizan los controles de entrada obtenidos con la red de planificación propuesta. Para cada caso, en color gris se muestra el robot tipo disco. En (a) y (b) se utilizan modelos entrenados con 400 trayectorias. En (c) se utiliza un modelo entrenado con 800 trayectorias. En (b) y (c) se representa la orientación del robot en color amarillo.

Inicialmente se utilizan los modelos de inferencia de manera completa, es decir, sin

aplicar *desconexión* en las capas ocultas a no ser que se generen resultados inválidos. Los estudios a algunos experimentos análogos de esta sección se pueden encontrar en el Apéndice A.1, donde se aplica *desconexión* en todo momento en cada una de las capas ocultas de la red de inferencia al momento de realizar la inferencia de nuevas entradas de control.

Para los experimentos en el primer entorno, se consideran las siguientes características

- Orientación inicial $\theta_0 = 0$
- Centro de la región A: $P_c = (0, 0)$.
- Centro de la región B: $P_c = (10, 15)$.
- Posición del estado objetivo, región A: $x'_{goal} = (15, 15)$.
- Posición del estado objetivo, región B: $x'_{goal} = (8, 0)$.

Similarmente, los experimentos en el segundo entorno toman en cuenta los siguientes valores

- Orientación inicial $\theta_0 = \frac{\pi}{2}$.
- Centro de la región inicial: $P_c = (10, 22)$.
- Posición del estado objetivo: $x'_{goal} = (20, 4)$.

6.1.1. Primer entorno, robot omnidireccional

Se consideraron los siguientes parámetros del método SST* para la generación de las trayectorias de entrenamiento:

- Intervalo de muestreo de controles: $u_i \in [-1, 1]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.6$.
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 7$.
- Parámetro de reducción $\xi = 0.93$.

Region A - Robot omnidireccional

En las tablas 6.2, 6.3 y 6.4 se muestran los resultados obtenidos al ejecutar la planificación sobre los estados iniciales propuestos de la región "A". La figura 6.4 muestra algunas trayectorias de ejemplo generadas con los controles de una red de inferencia. En 6.4a se presentan los caminos considerando estados iniciales del vecindadario inicial. 6.4b muestra caminos empezando en la vecindad media, y 6.4c presenta los caminos generados tomando estados de la vecindad lejana.

Se puede observar en la tabla 6.2 que la tasa de éxitos para todos los modelos de prueba es del 100 %, en otras palabras, en todos los casos se encontró un camino libre de colisión con un costo menor al umbral de rechazo 6.1. Los modelos entrenados con 400 muestras presentan el valor más bajo en la media, sin embargo, tanto el costo mínimo, costo máximo, media y desviación estándar varían por menos del 5 % tomando como referencia los valores obtenidos con el modelo de mayor número muestras con respecto a los demás modelos, es decir, los resultados obtenidos en todos los modelos son similares en términos de costo.

Analizando la tabla 6.3, cuando la selección de estados iniciales corresponde a la vecindad media, se puede observar que los modelos tienen una tasa de éxitos del 100 %. Al igual que en el caso anterior, los valores de costo mínimo, costo máximo, media y desviación estándar tienen una variación muy baja entre modelos.

En la tabla 6.4 se pueden observar los resultados cuando los estados iniciales se alejan aún más de la vecindad de entrenamiento. Los modelos de 400 muestras presentan una tasa de éxitos superior a los demás modelos, así como el menor valor en la media. Se pueden resaltar las siguientes características: las redes entrenadas con 100 trayectorias muestran la peor tasa de éxito, sin embargo, también presentan el valor más bajo en el mínimo. Las redes entrenadas con 200 muestras tienen una mayor tasa de éxito que las de 300 muestras, por lo que se puede decir que *un modelo entrenado con un mayor número de muestras no necesariamente presenta un mejor resultado.*

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	23.100	23.438	23.229	23.143
Máximo	27.516	27.214	26.718	26.636
Media	25.064	24.887	24.878	24.802
SD	1.275	1.316	1.267	1.255
Éxitos (%)	100.0	100.0	100.0	100.0
Rechazos (%)	0.0	0.0	0.0	0.0

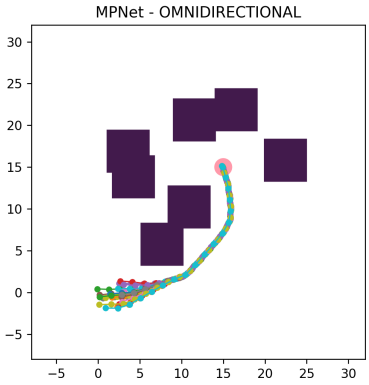
Tabla 6.2: Resultados estadísticos en la región "A" del primer entorno. Robot omnidireccional. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	21.587	21.936	21.736	21.625
Máximo	29.064	29.603	28.974	29.128
Media	25.094	25.291	25.122	25.053
SD	2.178	2.337	2.140	2.199
Éxitos (%)	100.0	100.0	100.0	100.0
Rechazos (%)	0.0	0.0	0.0	0.0

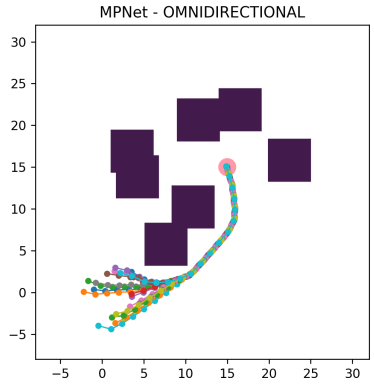
Tabla 6.3: Resultados estadísticos en la región "A" del primer entorno. Robot omnidireccional. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	20.101	20.156	20.229	20.119
Máximo	31.576	31.880	31.464	31.198
Media	25.639	25.694	25.563	25.299
SD	3.815	3.933	3.687	3.610
Éxitos (%)	85.0	89.0	87.0	91.0
Rechazos (%)	0.0	0.0	0.0	0.0

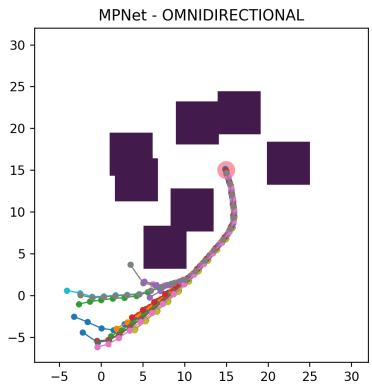
Tabla 6.4: Resultados estadísticos en la región "A" del primer entorno. Robot omnidireccional. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.4: Obtención de trayectorias en el espacio libre de colisión mediante un modelo entrenado con 400 trayectorias considerando los estados iniciales de la región de entrenamiento propuesta, robot omnidireccional.

Región B - Robot omnidireccional

Las tablas 6.5, 6.6 y 6.7 reportan resultados obtenidos al utilizar la red de planificación sobre los estados iniciales propuestos de la región "B". En la figura 6.5 se muestran algunos caminos obtenidos en la región "B" mediante la obtención de controles con una red de inferencia. En 6.4a se presentan los caminos considerando estados iniciales de la vecindad inicial. 6.4b muestra caminos empezando en la vecindad media, y 6.4c presenta los caminos generados tomando estados de la vecindad lejana.

Según la tabla 6.5 y tomando como referencia los resultados de los modelos entrenados con 400 muestras, los valores en el mínimo, máximo y media varían por menos del 5 % con respecto a los demás modelos. Los modelos con mayor tasa de éxito fueron los entrenados con 100 muestras, y las redes con 200 muestras presentaron un menor costo medio.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	18.048	18.139	18.187	18.143
Máximo	22.098	22.207	22.244	22.235
Media	20.135	19.946	20.043	20.106
SD	1.395	1.513	1.536	1.557
Éxitos (%)	100.0	99.0	95.0	98.0
Rechazos (%)	0.0	0.0	0.0	0.0

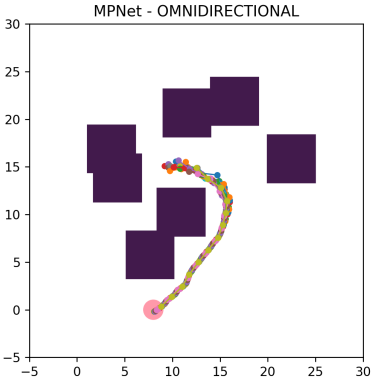
Tabla 6.5: Resultados estadísticos en la región "B" del primer entorno. Robot omnidireccional. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	16.565	16.430	16.418	16.428
Máximo	22.766	22.416	22.605	22.364
Media	18.848	18.652	18.604	18.766
SD	1.971	1.998	2.018	1.990
Éxitos (%)	99.0	100.0	98.0	99.0
Rechazos (%)	0.0	0.0	0.0	0.0

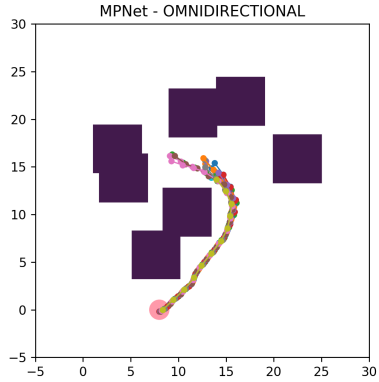
Tabla 6.6: Resultados estadísticos en la región "B" del primer entorno. Robot omnidireccional. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	13.689	13.546	13.793	13.787
Máximo	25.232	25.372	25.338	25.073
Media	17.049	16.931	17.291	17.046
SD	2.652	2.728	2.963	2.730
Éxitos (%)	97.0	98.0	100.0	98.0
Rechazos (%)	0.0	0.0	0.0	0.0

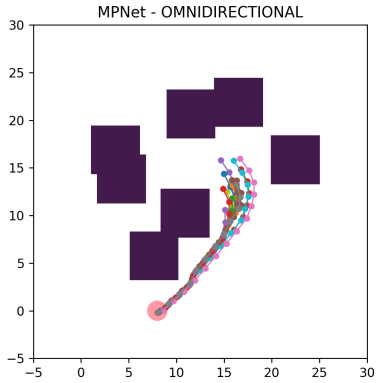
Tabla 6.7: Resultados estadísticos en la región "B" del primer entorno. Robot omnidireccional. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.5: Trayectorias obtenidos utilizado un modelo entrenado con 400 caminos considerando la segunda región de entrenamiento, robot omnidireccional.

Las tablas 6.6 y 6.7 indican que la tasa de éxitos para todos los modelos es casi del 100 % aún cuando se realiza la planificación fuera del vecindario de entrenamiento.

En ambos casos, el valor de costo medio, mínimo y máximo de todos los modelos son similares.

6.1.2. Primer entorno, DDR de primer orden

Se consideran los siguientes parámetros para el *SST**

- Intervalo de muestreo de controles: $u_i \in [-1.5, 1.5]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.6$.
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 7$.
- Parámetro de reducción $\xi = 0.95$.

Región A - DDR de primer orden

Las tablas 6.8, 6.9 y 6.10 muestran los resultados obtenidos de la planificación en la región "A" dentro de la vecindad de entrenamiento, vecindad media y vecindad lejana respectivamente. Las figuras en 6.6 muestran ejemplos de trayectorias en la región "A" al utilizar una red de inferencia entrenada con 400 muestras. 6.6a, 6.6b, 6.6c corresponden al vecindario inicial, medio y lejano respectivamente.

En todos los casos, se indica que los modelos entrenados con más muestras presentan una mayor tasa de éxitos. El valor medio más bajo se presenta en los modelos de 200 muestras, sin embargo, este valor varía por muy poco con respecto a los otros modelos. Una diferencia notable que se presenta al utilizar el DDR es que el desempeño de los modelos, principalmente los de menor número de muestras, disminuye ligeramente conforme se aleja el estado inicial de la región de entrenamiento. La tabla 6.10 exhibe que el proceso de planificación encontró trayectorias de alto costo, las cuales son clasificadas como rechazos.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	19.556	19.570	19.887	20.295
Máximo	22.917	22.564	22.906	22.890
Media	21.255	21.158	21.310	21.335
SD	0.935	0.965	0.972	0.866
Éxitos (%)	94.0	99.0	100.0	100.0
Rechazos (%)	0.0	0.0	0.0	0.0

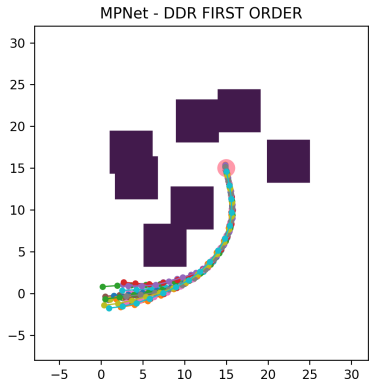
Tabla 6.8: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de primer orden. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	19.295	18.989	19.313	19.143
Máximo	30.122	26.229	26.787	25.970
Media	22.169	21.736	22.048	21.761
SD	2.918	2.131	2.120	1.914
Éxitos (%)	87.0	83.0	95.0	95.0
Rechazos (%)	0.0	0.0	0.0	0.0

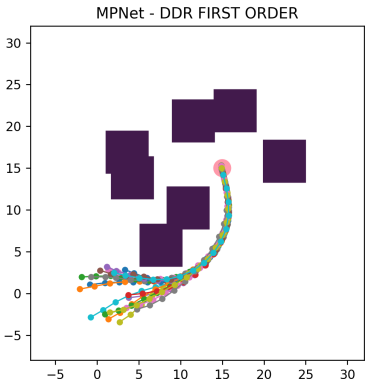
Tabla 6.9: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de primer orden. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	16.825	16.914	17.710	17.374
Máximo	31.074	31.956	30.589	32.321
Media	22.937	22.881	23.415	23.063
SD	4.032	4.894	3.755	3.961
Éxitos (%)	66.0	55.0	82.0	82.0
Rechazos (%)	3.0	13.0	2.0	8.0

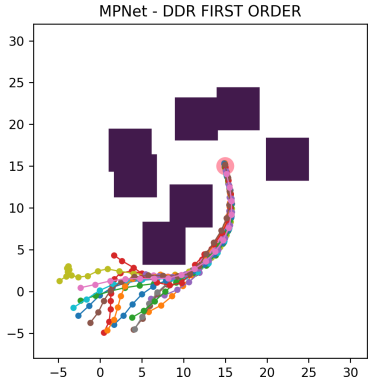
Tabla 6.10: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de primer orden. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.6: Obtención de trayectorias en el espacio libre de colisión mediante un modelo entrenado con 400 trayectorias considerando los estados iniciales de la región de entrenamiento propuesta, robot de manejo diferencial de primer orden.

Región B - DDR de primer orden

El desempeño del método de planificación para el DDR de primer orden en la región "B", es similar al desempeño del método en la región "A". En las tablas 6.11, 6.12 y 6.13 se pueden observar los valores estadísticos correspondientes al costo de trayectorias generadas en el vecindario inicial, medio y lejano respectivamente. En la figura 6.7 se ilustran ejemplos de trayectorias para el primer entorno, región "B", utilizando un modelo entrenado con 400 muestras para el robot DDR.

Al igual que en la región "A", el DDR disminuye su tasa de éxitos cuando se toman estados iniciales fuera de la vecindad de entrenamiento. En esta ocasión, el éxito se

disminuye considerablemente en la vecindad lejana y se presenta rechazo. En todos los casos, los modelos presentan un valor de media similar, variando por menos del 10 % entre modelos. En el vecindario inicial y medio, los modelos de 100 muestras presentan menos dispersión en el costo de trayectorias, mientras que en el vecindario lejano las redes de 400 muestras tienen menor varianza.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	17.658	17.506	17.560	17.733
Máximo	21.178	21.046	21.359	20.953
Media	19.558	19.313	19.142	19.282
SD	0.888	1.128	1.196	1.020
Éxitos (%)	95.0	98.0	96.0	95.0
Rechazos (%)	0.0	0.0	0.0	0.0

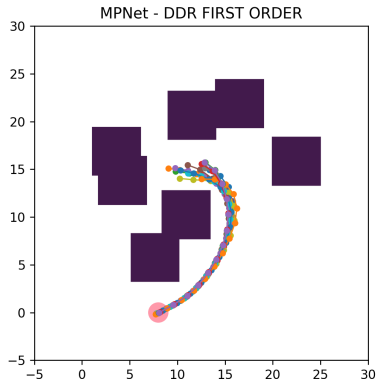
Tabla 6.11: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de primer orden. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	17.290	16.670	16.770	16.889
Máximo	20.872	20.985	22.125	21.093
Media	19.127	18.303	18.480	18.353
SD	1.114	1.393	1.750	1.595
Éxitos (%)	92.0	94.0	93.0	82.0
Rechazos (%)	0.0	0.0	0.0	0.0

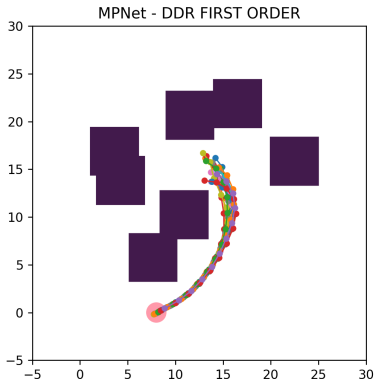
Tabla 6.12: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de primer orden. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	16.775	16.958	16.564	16.902
Máximo	28.269	32.400	28.500	28.595
Media	20.744	19.950	20.060	19.595
SD	5.285	5.107	4.629	3.773
Éxitos (%)	61.0	75.0	84.0	79.0
Rechazos (%)	4.0	10.0	2.0	5.0

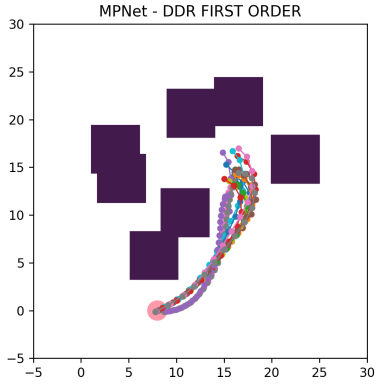
Tabla 6.13: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de primer orden. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.7: Trayectorias obtenidos utilizado un modelo entrenado con 400 caminos considerando la segunda región de entrenamiento, robot de manejo diferencial de primer orden.

6.1.3. Primer entorno, DDR de segundo orden

Para la generación de datos de entrenamiento con el *SST** se consideran los siguientes parámetros:

- Intervalo de muestreo de controles: $u_i \in [-1, 1]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.6$.

- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 6$.

- Parámetro de reducción $\xi = 0.95$.

Región A - DDR de segundo orden

Las tablas 6.14, 6.15 y 6.16 corresponden a los resultados estadísticos del costo de las trayectorias obtenidas para el DDR de segundo orden en la región "A". La figura 6.8 muestra algunos ejemplos de trayectorias obtenidas mediante la metodología propuesta utilizando redes entrenadas con 800 muestras en la región "A". 6.8a, 6.8b y 6.8c corresponden a la vecindad inicial, media y lejana respectivamente.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	11.276	11.232	11.042	10.683
Máximo	13.497	12.921	12.869	12.657
Media	12.136	11.838	11.810	11.505
SD	0.686	0.494	0.487	0.545
Éxitos (%)	63.0	79.0	92.0	98.0
Rechazos (%)	0.0	0.0	0.0	0.0

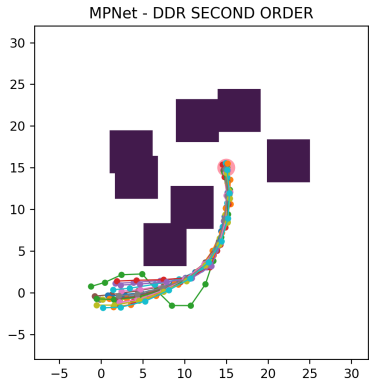
Tabla 6.14: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de segundo orden. $0 < r < 2$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	11.175	10.632	10.684	10.619
Máximo	15.684	15.617	14.008	14.683
Media	12.795	12.613	11.826	12.166
SD	2.140	2.419	1.258	1.572
Éxitos (%)	20.0	26.0	38.0	42.0
Rechazos (%)	0.0	1.0	0.0	0.0

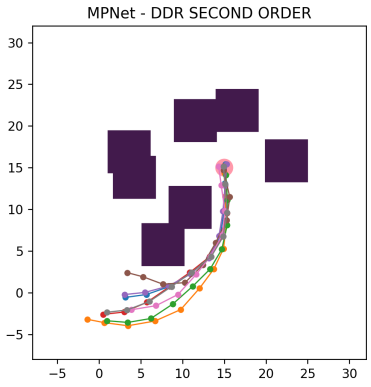
Tabla 6.15: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de segundo orden. $2 < r < 4$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	12.407	13.115	10.260	11.848
Máximo	13.461	21.136	18.442	15.557
Media	12.960	15.736	12.857	13.826
SD	0.713	4.482	3.621	1.594
Éxitos (%)	9.0	21.0	33.0	26.0
Rechazos (%)	0.0	0.0	0.0	1.0

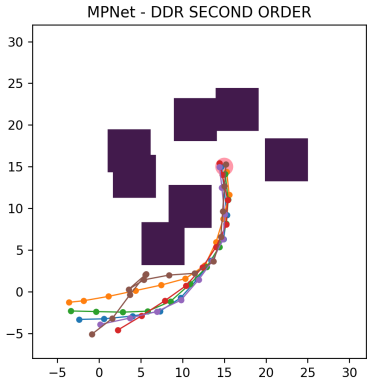
Tabla 6.16: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de segundo orden. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.8: Obtención de trayectorias en el espacio libre de colisión mediante un modelo entrenado con 800 trayectorias, robot de manejo diferencial de segundo orden.

En el vecindario inicial se muestra que las redes entrenadas con 600 y 800 trayectorias tienen las mejores tasas de éxito, sin embargo, el costo medio de trayectoria es

similar para todos los modelos (variación menor al 5 %). Al ejecutar la planificación fuera del vecindario inicial, el desempeño del método disminuye significativamente. Se puede observar tanto en 6.15 como en 6.16 que la tasa de éxito en ningun caso supera el 50 %. Esta observación sugiere que el método propuesto tiene dificultades para generalizar trayectorias fuera de la región inicial al utilizar sistemas dinámicos complejos. En la sección 6.2 se verifica si esta generalización es afectada por el número de caminos que se utilizan en el entrenamiento.

Región B - DDR de segundo orden

Los resultados estadísticos de los costos de trayectorias para este caso se presentan en las tablas 6.17, 6.18 y 6.19. Las imágenes en la figura 6.9 muestran ejemplos de trayectorias generadas con el apoyo de una red entrenada con 800 muestras en la región "B".

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	10.164	10.135	10.151	9.858
Máximo	11.266	11.524	11.408	11.474
Media	10.746	10.768	10.740	10.723
SD	0.354	0.448	0.412	0.462
Éxitos (%)	46.0	87.0	93.0	88.0
Rechazos (%)	0.0	0.0	0.0	0.0

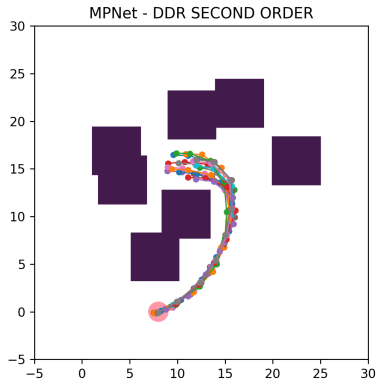
Tabla 6.17: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de segundo orden. $0 < r < 2$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	10.537	9.985	9.784	9.872
Máximo	11.677	11.818	11.756	11.680
Media	11.134	10.815	10.626	10.697
SD	0.494	0.601	0.697	0.623
Éxitos (%)	26.0	58.0	60.0	70.0
Rechazos (%)	0.0	0.0	0.0	0.0

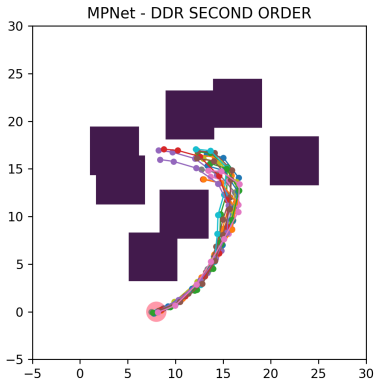
Tabla 6.18: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de segundo orden. $2 < r < 4$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	16.802	10.484	10.707	10.361
Máximo	21.501	13.379	15.448	18.630
Media	18.986	11.498	12.598	12.555
SD	2.804	1.277	2.674	4.273
Éxitos (%)	7.0	24.0	30.0	19.0
Rechazos (%)	0.0	0.0	0.0	1.0

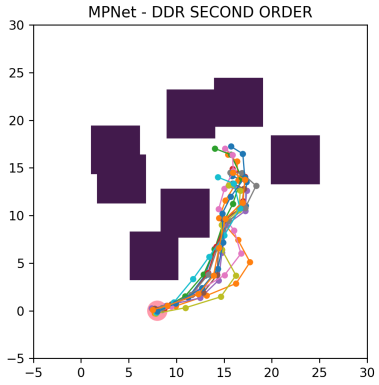
Tabla 6.19: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de segundo orden. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.9: Trayectorias obtenidos utilizado un modelo entrenado con 800 caminos en la región "B", robot de manejo diferencial de segundo orden.

Al igual que en la región "A", el desempeño del método disminuye significativamente al planificar fuera del vecindario inicial. Mientras que los modelos de 600 muestras

parecen tener la mejor tasa de éxitos en el primer vecindario, son superados en la vecindad media por las redes de 800 trayectorias. En la vecindad lejana ningún modelo supera el 30 % de éxitos. La media del costo en todos los vecindarios es similar para todos los modelos estudiados (variación menor al 10 %), exceptuando a las redes de 200 muestras en la vecindad lejana.

6.1.4. Segundo entorno, robot omnidireccional

El segundo entorno de trabajo se muestra en la figura 6.2. Para la generación de datos de entrenamiento, se consideraron los siguientes parámetros del planificador *SST**:

- Intervalo de muestreo de controles: $u_i \in [-1, 1]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.2$
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 6$.
- Parámetro de reducción $\xi = 0.95$.

Las tablas 6.20, 6.21 y 6.22 presentan los resultados estadísticos del costo de las trayectorias al ejecutar el método de planificación propuesto con diferentes redes de inferencia. En todos los casos, el porcentaje de éxitos supera el 90 %. Esto sugiere que los modelos de inferencia son capaces de generar satisfactoriamente entradas de control aún cuando el vecindario inicial se encuentra dentro de una concavidad en el espacio de trabajo. La variación que existe entre el costo medio de las trayectorias es inferior al 5 % considerando los modelos que otorgaron menor costo con respecto a los demás.

Las figuras 6.10a, 6.10b y 6.10c de 6.10 muestran ejemplos de trayectorias obtenidas al resolver el problema de planificación utilizando controles de redes entrenadas con 400 trayectorias.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	31.192	31.687	30.893	31.521
Máximo	34.192	34.747	34.369	34.237
Media	32.644	32.876	32.495	32.782
SD	0.854	0.976	1.012	0.905
Éxitos (%)	94.0	95.0	99.0	99.0
Rechazos (%)	0.0	0.0	0.0	0.0

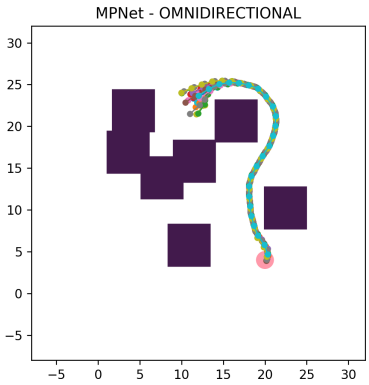
Tabla 6.20: Resultados estadísticos en el segundo entorno. Robot omnidireccional. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	29.477	29.769	29.190	29.598
Máximo	35.445	35.562	35.796	35.392
Media	33.050	33.331	32.985	33.380
SD	1.981	1.969	1.999	1.958
Éxitos (%)	99.0	95.0	99.0	100.0
Rechazos (%)	0.0	0.0	0.0	0.0

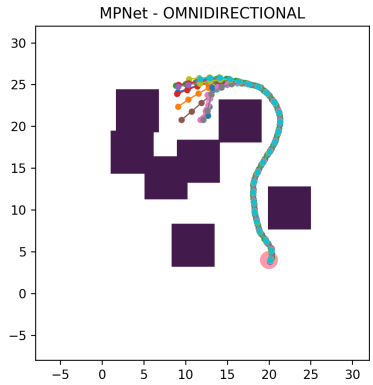
Tabla 6.21: Resultados estadísticos en el segundo entorno. Robot omnidireccional. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	26.070	26.634	25.991	26.653
Máximo	37.888	37.629	37.948	38.144
Media	31.413	31.679	31.379	31.695
SD	3.659	3.740	3.783	3.775
Éxitos (%)	99.0	97.0	100.0	98.0
Rechazos (%)	0.0	0.0	0.0	0.0

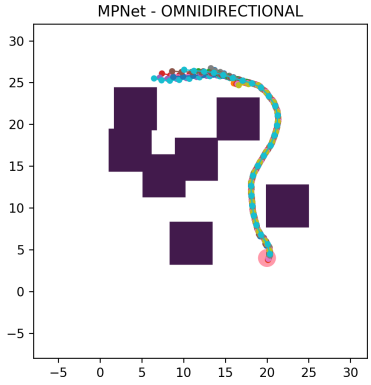
Tabla 6.22: Resultados estadísticos en el segundo entorno. Robot omnidireccional. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.10: Planificación para robot omnidireccional, las trayectorias son generadas obteniendo entradas de control de un modelo entrenado con 400 caminos en el segundo entorno de trabajo.

6.1.5. Segundo entorno, DDR de primer orden

- Intervalo de muestreo de controles: $u_i \in [-1, 1]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.2$
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 6$.

- Parámetro de reducción $\xi = 0.95$.

En las tablas 6.23, 6.24 y 6.25 se muestran los resultados estadísticos del costo de trayectorias obtenidas mediante el método propuesto para el robot DDR de primer orden. Algunas trayectorias generadas con el uso de una red entrenada con 400 muestras se muestran en la figura 6.11. Se puede observar que en todas los vecindarios, los modelos entrenados con 100 muestras tienen un desempeño pobre, pues su tasa de éxitos no pasa del 20 %, adicionalmente estos presentan una alta tasa de rechazos, por lo que la calidad de trayectorias generadas no es deseable. Por otra parte, las redes entrenadas con mayor número de muestras presentan un buen porcentaje de éxitos en el vecindario inicial, pero este disminuye conforme se consideran los estados iniciales de otros vecindarios. Los modelos de 300 y 400 muestras presentan una media de costo similar.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	52.418	50.646	47.248	47.994
Máximo	58.443	59.362	56.396	54.602
Media	54.547	53.773	49.784	50.205
SD	3.377	2.723	2.745	1.747
Éxitos (%)	17.0	70.0	91.0	94.0
Rechazos (%)	12.0	3.0	1.0	1.0

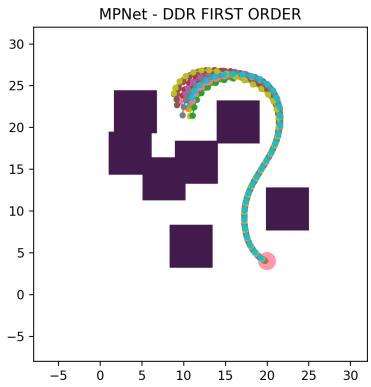
Tabla 6.23: Resultados estadísticos en el segundo entorno. DDR de primer orden. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	52.035	48.826	44.839	45.579
Máximo	61.411	61.612	58.564	57.613
Media	55.734	54.168	48.873	49.575
SD	4.032	4.466	4.309	4.018
Éxitos (%)	15.0	55.0	66.0	72.0
Rechazos (%)	17.0	2.0	6.0	12.0

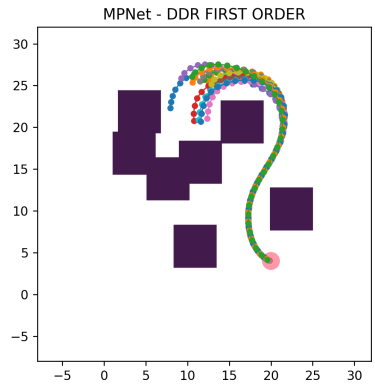
Tabla 6.24: Resultados estadísticos en el segundo entorno. DDR de primer orden. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	41.849	43.909	44.524	44.950
Máximo	64.076	56.298	53.170	54.814
Media	50.384	49.407	46.870	46.806
SD	7.665	4.098	2.792	3.269
Éxitos (%)	18.0	53.0	64.0	70.0
Rechazos (%)	13.0	8.0	12.0	22.0

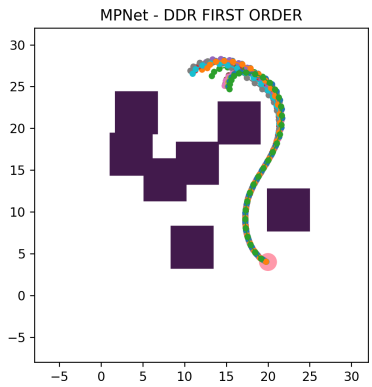
Tabla 6.25: Resultados estadísticos en el segundo entorno. DDR de primer orden. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.11: Se utiliza una red de inferencia para el robot de manejo diferencial de primer orden, las trayectorias son obtenidas usando un modelo entrenado con 400 caminos en el segundo entorno de trabajo.

6.1.6. Segundo entorno, DDR de segundo orden

Los valores del *SST** para generar los datos de entrenamiento son los siguientes

- Intervalo de muestreo de controles: $u_i \in [-1, 1]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.2$.
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 6$.
- Parámetro de reducción $\xi = 0.95$.

En las tablas 6.26, 6.27 y 6.28 se muestran los resultados estadísticos del costo para la vecindad inicial, media y lejana respectivamente. En la figura 6.12 se muestran algunas trayectorias generadas con el uso de una red entrenada con 800 muestras.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	15.292	14.658	14.624	15.002
Máximo	15.762	15.955	15.766	16.342
Media	15.537	15.286	15.176	15.563
SD	0.333	0.483	0.384	0.411
Éxitos (%)	5.0	55.0	67.0	60.0
Rechazos (%)	0.0	0.0	0.0	0.0

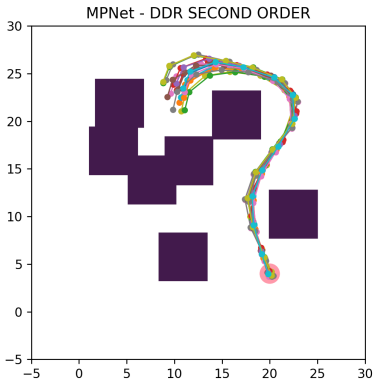
Tabla 6.26: Resultados estadísticos en el segundo entorno. DDR de segundo orden. $0 < r < 2$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	14.743	14.415	14.046	14.860
Máximo	15.393	15.481	16.935	16.787
Media	14.957	15.042	15.528	15.842
SD	0.361	0.505	1.098	0.719
Éxitos (%)	5.0	18.0	47.0	60.0
Rechazos (%)	0.0	0.0	0.0	0.0

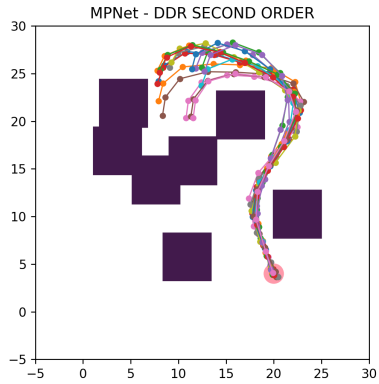
Tabla 6.27: Resultados estadísticos en el segundo entorno. DDR de segundo orden. $2 < r < 4$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	14.328	13.109	13.700	13.312
Máximo	14.329	15.557	13.909	14.974
Media	14.329	14.570	13.811	14.374
SD	0.000	1.124	0.111	0.934
Éxitos (%)	2.0	7.0	7.0	15.0
Rechazos (%)	0.0	0.0	0.0	0.0

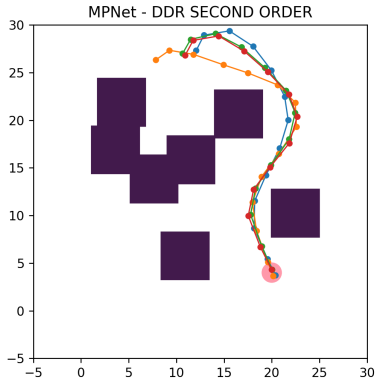
Tabla 6.28: Resultados estadísticos en el segundo entorno. DDR de segundo orden. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.12: Trayectorias obtenidas usando un modelo entrenado con 800 caminos en el segundo entorno de trabajo. Robot de manejo diferencial con dinámica de segundo orden.

Se observa que los modelos con 200 muestras tienen una tasa de éxito muy baja en todos los casos. Las redes entrenadas con 600 trayectorias tienen mayor número de

éxitos en la vecindad inicial. Por otro lado, los modelos de 800 muestras tienen mayor tasa de éxito en la vecindad media y lejana. El costo medio de trayectoria en todos los casos varía por menos del 10 % entre modelos. En este caso, la tasa de éxitos para todos los experimentos es muy baja, siendo como mejor resultado un 67 % de éxitos en el vecindario inicial utilizando las redes de 600 muestras.

Los resultados expuestos en esta sección indican que el proceso de planificación mediante el método propuesto para los tres sistemas dinámicos de prueba tiende a dificultarse cuando se toman estados iniciales de las vecindades medias y lejanas, pues el porcentaje de éxitos es por lo general inferior que al planificar tomando estados iniciales en la región de entrenamiento. Se observa que para el *robot omnidireccional* y el robot *DDR de primer orden*, la disminución en el porcentaje de éxitos al variar el vecindario de partida se presenta en menor medida con respecto a la disminución de éxitos para el robot *DDR de segundo orden*, por lo que la generación de controles y tiempos de aplicación fuera de la región de entrenamiento para este sistema se generaliza en un menor grado.

En la mayoría de los experimentos, los modelos neuronales entrenados con el mayor número de muestras se desempeñan mejor que los modelos entrenados con el menor número de muestras, ya que tanto el porcentaje de éxitos, y en ocasiones el costo promedio de trayectoria, es superior en estos.

6.2. Aumento de muestras: DDR de segundo orden

Los resultados expuestos en la sección 6.1.3 sugieren que el número de muestras utilizadas en el entrenamiento de los modelos puede influir en el porcentaje de éxitos de la planificación. En esta sección se presenta un estudio similar al de la sección 6.1 utilizando modelos neuronales entrenados con 1000, 2000, 3000 y 4000 trayectorias para el DDR de segundo orden. Los parámetros de entrada para la generación de datos son los mismos a los descritos en la sección 6.1.3, haciendo un estudio únicamente en la región "A" del primer entorno (figura 6.1a).

Las tablas 6.29, 6.30 y 6.31 corresponden a los resultados estadísticos que se obtuvieron al utilizar la metodología propuesta de planificación. Se puede resaltar que todos los modelos presentan una tasa de éxito mayor al 90 % cuando se toman estados iniciales de la vecindad de entrenamiento. Además, el costo promedio de trayectoria tanto en la vecindad inicial como en la vecindad media es similar para todos los modelos. En la vecindad media y lejana, el porcentaje de éxitos empieza a disminuir significativamente, siendo los modelos de 3000 y 4000 trayectorias los que presentan un mejor desempeño.

Contrastando el porcentaje de éxitos entre las redes de 1000 y 3000 muestras, se puede observar que al utilizar estas últimas se obtiene una mejora del 18 % al planificar en la vecindad media y del 31 % al planificar en la vecindad lejana. Las trayectorias generadas con los modelos entrenados con 4000 muestras presentan el menor costo medio en los últimos dos vecindarios, y una desviación estándar inferior a los demás modelos.

La figura 6.13 muestra ejemplos de caminos generados con una red de inferencia entrenada con 4000 muestras. Se puede resaltar la cantidad de trayectorias generadas en la vecindad media y lejana (6.13b y 6.13c respectivamente) es mayor a las generadas por la red de 800 muestras (6.8b y 6.8c).

	1000 muestras	2000 muestras	3000 muestras	4000 muestras
Mínimo	10.859	11.056	10.794	11.010
Máximo	12.862	13.215	12.931	12.814
Media	11.706	11.749	11.551	11.693
SD	0.525	0.590	0.517	0.486
Éxitos (%)	95.0	97.0	97.0	94.0
Rechazos (%)	0.0	0.0	0.0	0.0

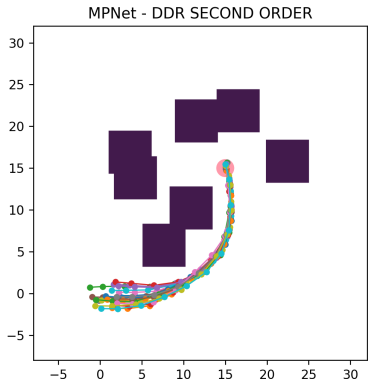
Tabla 6.29: Resultados estadísticos. Aumento de muestras. DDR de segundo orden. $0 < r < 2$.

	1000 muestras	2000 muestras	3000 muestras	4000 muestras
Mínimo	10.713	10.627	10.646	10.821
Máximo	14.524	18.846	15.628	13.404
Media	12.153	12.443	12.144	12.038
SD	1.385	2.729	1.497	0.826
Éxitos (%)	46.0	58.0	64.0	62.0
Rechazos (%)	0.0	0.0	0.0	0.0

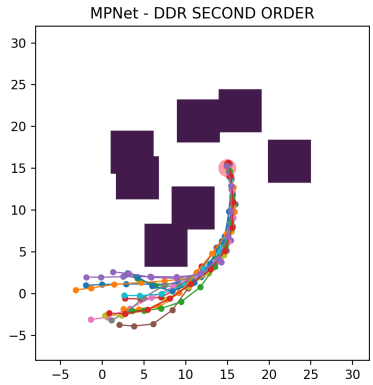
Tabla 6.30: Resultados estadísticos. Aumento de muestras. DDR de segundo orden. $2 < r < 4$.

	1000 muestras	2000 muestras	3000 muestras	4000 muestras
Mínimo	11.537	11.172	10.781	11.537
Máximo	19.046	17.383	17.763	16.668
Media	15.449	14.122	14.083	13.621
SD	2.936	2.063	2.223	1.532
Éxitos (%)	28.0	34.0	59.0	56.0
Rechazos (%)	0.0	1.0	0.0	0.0

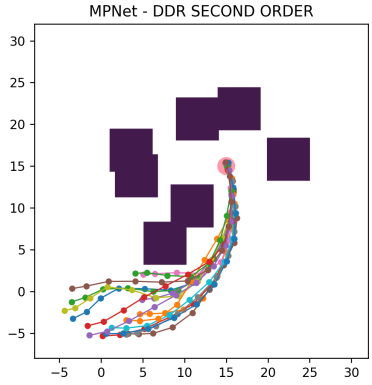
Tabla 6.31: Resultados estadísticos. Aumento de muestras. DDR de segundo orden. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.13: Trayectorias obtenidas mediante el uso de la red entrenada con 4000 caminos, se toman los estados iniciales propuestos en la región "A" del primer entorno para el entrenamiento. Robot de manejo diferencial con dinámica de segundo orden.

Comparando los resultados obtenidos en esta sección con los de la sección 6.1 para la planificación del *robot DDR de segundo orden*, se observa que el porcentaje de éxito para los experimentos en esta sección es superior, por lo que el número de muestras utilizadas en el entrenamiento de los modelos neuronales puede ser uno de los factores para una mejor generalización en la predicción de controles, tanto en la región inicial como en los vecindarios apartados.

6.3. Caminos sobre pasaje estrecho

En esta sección se presentan experimentos de planificación sobre un entorno con un pasaje estrecho. Se utiliza la región "B" del primer entorno descrito en la sección 6.1 para el entrenamiento de las redes. En este caso, la región objetivo se posiciona de tal manera que la trayectoria desde un estado inicial pueda recorrer el pasaje estrecho para completar el proceso de planificación.

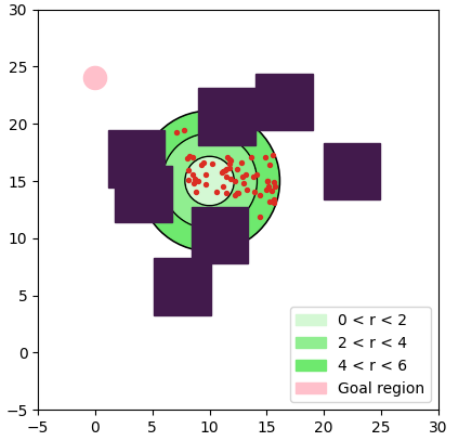


Figura 6.14: Pasaje estrecho: entorno de trabajo. Se toman los estados iniciales de la región "B" del primer entorno de trabajo propuesto en la sección 6.1. Se cambia la posición de la región objetivo.

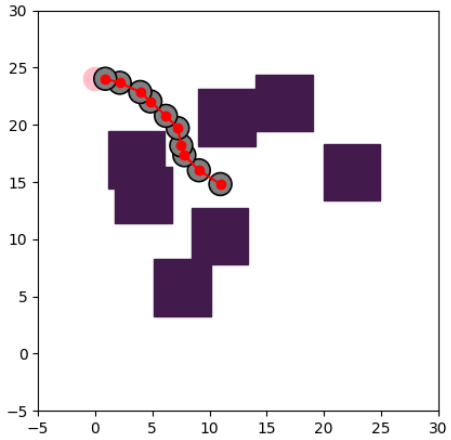


Figura 6.15: Trayectoria sobre pasaje estrecho obtenida con el planificador *SST**. *Robot omnidireccional.*

La figura 6.14 muestra la región objetivo y los estados iniciales utilizados para la evaluación de los experimentos. En la figura 6.15 se presenta un ejemplo de trayectoria obtenida mediante el planificador *SST**, la cual es utilizada en el entrenamiento de la *red de inferencia* para el *robot omnidireccional*; se puede observar que el área del robot cubre casi completamente el ancho del pasaje estrecho.

Para el robot omnidireccional y DDR de primer orden, se entrenaron cinco modelos de 100, 200, 300 y 400 trayectorias. Para el DDR de segundo orden se utilizan modelos de 200, 400, 600 y 800 trayectorias. Se consideran los siguientes valores

- Orientación inicial $\theta_0 = \pi$.
- Centro de la región inicial: $P_c = (10, 15)$.
- Posición del estado objetivo: $x'_{goal} = (0, 24)$.

Un estudio análogo a este experimento se presenta en el Apéndice B, donde se considera la misma región de entrenamiento y zona objetivo, pero las redes son entrenadas con trayectorias que no recorren el pasaje estrecho (costo alto) y que *no necesariamente* recorren el pasaje estrecho (costo variable).

6.3.1. Pasaje estrecho: robot omnidireccional

El método *SST** toma los siguientes parámetros para la generación de datos de entrenamiento

- Intervalo de muestreo de controles: $u_i \in [-1, 1]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.6$
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 7$.

- Parámetro de reducción $\xi = 0.93$.

Las tablas 6.32, 6.33 y 6.34 muestran los resultados obtenidos para la vecindad inicial, media y lejana respectivamente. En la figura 6.16 se presentan ejemplos de trayectorias generadas a través del pasaje estrecho, se usó una red de inferencia entrenada con 400 muestras.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	11.275	11.455	11.213	11.255
Máximo	14.222	14.144	14.338	14.056
Media	13.211	13.191	13.178	13.116
SD	0.899	0.859	0.898	0.880
Éxitos (%)	89.0	100.0	99.0	96.0
Rechazos (%)	0.0	0.0	0.0	0.0

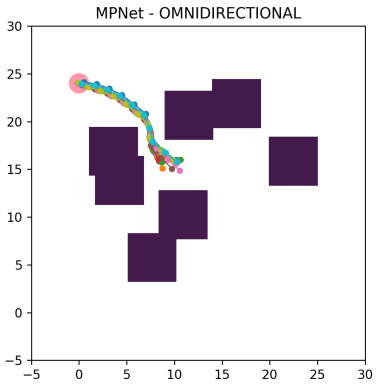
Tabla 6.32: Resultados estadísticos. Pasaje estrecho. Robot omnidireccional. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	10.169	9.865	9.879	9.864
Máximo	16.097	16.373	16.685	16.498
Media	13.922	14.063	13.941	14.051
SD	1.775	1.841	1.954	1.953
Éxitos (%)	78.0	100.0	97.0	96.0
Rechazos (%)	0.0	0.0	0.0	0.0

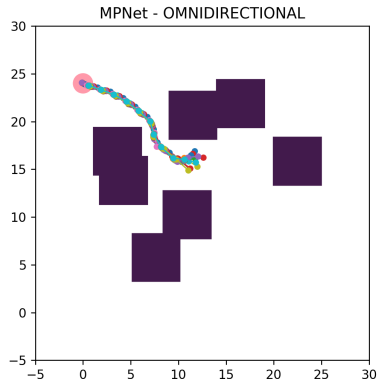
Tabla 6.33: Resultados estadísticos. Pasaje estrecho. Robot omnidireccional. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	8.352	8.270	8.222	8.255
Máximo	19.002	18.724	18.773	19.093
Media	16.706	16.812	16.733	16.748
SD	3.228	3.003	3.022	3.077
Éxitos (%)	69.0	94.0	93.0	91.0
Rechazos (%)	0.0	0.0	0.0	0.0

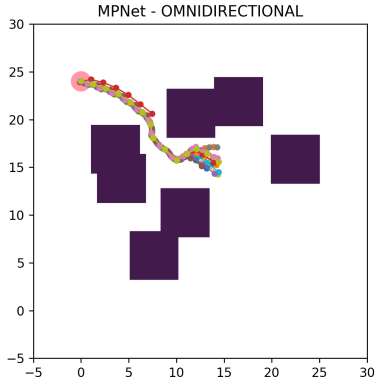
Tabla 6.34: Resultados estadísticos. Pasaje estrecho. Robot omnidireccional. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.16: Caminos obtenidos al utilizar la red entrenada con 400 muestras, los estados iniciales corresponden a la segunda región de entrenamiento. En todos los casos el robot cruza a través de dos obstáculos para llegar a la nueva región objetivo. Robot omnidireccional.

Las redes entrenadas con más de 200 muestras presentan una tasa de éxito mayor del 90 % para todos los casos. Por otra parte, las redes entrenadas con 100 muestras tienen una tasa de éxitos inferior a 90 %. La variación que existe en el valor de la media y desviación estándar para todos los modelos es inferior al 10 %, es decir, todos los modelos en promedio producen trayectorias de calidad similar.

6.3.2. Pasaje estrecho: DDR de primer orden

Los parámetros del *SST** en este caso son los siguientes

- Intervalo de muestreo de controles: $u_i \in [-1.5, 1.5]$.

- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.6$
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 7$.
- Parámetro de reducción $\xi = 0.95$.

Al planificar para un robot DDR el desempeño del método disminuye ligeramente en comparación con el robot omnidireccional. Las tablas 6.35, 6.36 y 6.37 muestran los estadísticos correspondientes al costo de trayectoria para cada vecindario. La figura 6.17 presenta ejemplos de caminos generados con el apoyo de una red de inferencia entrenada con 400 muestras. 6.17a corresponde a la vecindad inicial, 6.17b a la vecindad media y 6.17c a la vecindad lejana.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	14.906	13.209	13.538	13.182
Máximo	20.394	21.965	18.685	19.294
Media	16.507	16.318	15.146	15.299
SD	2.258	3.834	1.341	1.627
Éxitos (%)	36.0	81.0	76.0	76.0
Rechazos (%)	0.0	0.0	0.0	0.0

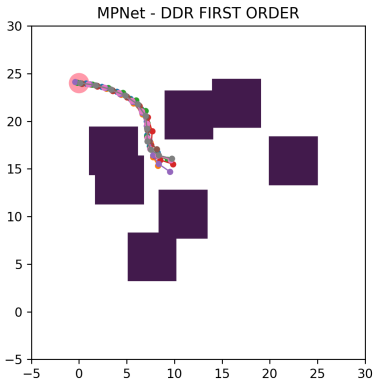
Tabla 6.35: Resultados estadísticos. Pasaje estrecho. DDR de primer orden. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	15.810	13.761	13.834	14.556
Máximo	19.947	20.375	18.579	20.512
Media	17.728	16.480	15.924	16.844
SD	1.607	2.439	1.350	1.748
Éxitos (%)	47.0	80.0	74.0	78.0
Rechazos (%)	0.0	0.0	0.0	0.0

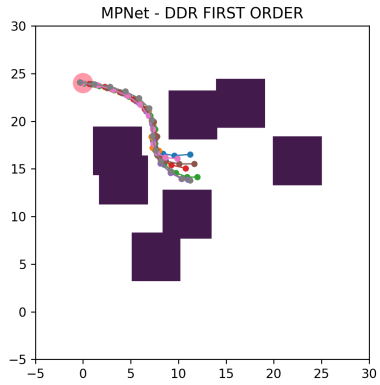
Tabla 6.36: Resultados estadísticos. Pasaje estrecho. DDR de primer orden. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	16.709	14.346	16.580	15.587
Máximo	22.242	24.247	22.223	26.922
Media	20.356	19.302	18.115	20.830
SD	2.637	4.015	2.216	4.447
Éxitos (%)	32.0	52.0	59.0	55.0
Rechazos (%)	0.0	0.0	0.0	0.0

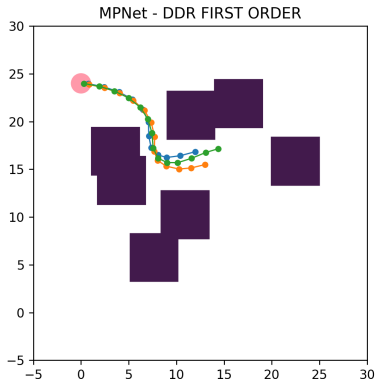
Tabla 6.37: Resultados estadísticos. Pasaje estrecho. DDR de primer orden. $4 < r < 6$.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.17: Ejemplos de caminos generados al utilizar la red entrenada con 400 muestras. El robot cruza a través de dos obstáculos para llegar a la región objetivo. Robot de manejo diferencial de primer orden.

Los modelos entrenados con 100 muestras tienen en general una menor tasa de éxito con respecto a los demás modelos. En todos los vecindarios, las redes con 300

muestras presentan la media del costo más baja. Para los modelos de 200, 300 y 400 muestras el porcentaje de éxito es similar en la vecindad inicial y media, y este disminuye ligeramente al tomar estados iniciales desde la vecindad lejana.

6.3.3. Pasaje estrecho: DDR de segundo orden

El *SST** para la generación de datos considera los siguientes valores

- Intervalo de muestreo de controles: $u_i \in [-1, 1]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.6$
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 6$.
- Parámetro de reducción $\xi = 0.95$.

Las tablas 6.38, 6.39 y 6.37 presentan los resultados estadísticos del costo de trayectorias generadas para la vecindad inicial, media y lejana respectivamente. La figura 6.18 muestra algunas trayectorias al utilizar una red de 800 muestras. 6.18a ilustra los caminos en la vecindad inicial, 6.18b en la vecindad media y 6.18c en la vecindad lejana.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	10.446	10.187	10.949	10.684
Máximo	12.833	12.983	12.712	12.735
Media	11.484	11.106	11.494	11.560
SD	0.755	0.856	0.551	0.653
Éxitos (%)	48.0	47.0	60.0	47.0
Rechazos (%)	0.0	0.0	0.0	0.0

Tabla 6.38: Resultados estadísticos. Pasaje estrecho. DDR de segundo orden. $0 < r < 2$.

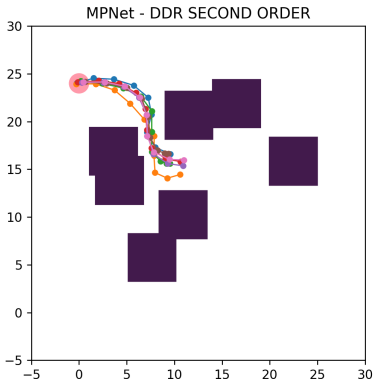
	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	11.624	11.642	11.902	11.226
Máximo	15.352	13.376	13.293	14.068
Media	12.912	12.551	12.639	12.515
SD	1.608	0.718	0.556	1.314
Éxitos (%)	27.0	28.0	31.0	33.0
Rechazos (%)	0.0	0.0	0.0	0.0

Tabla 6.39: Resultados estadísticos. Pasaje estrecho. DDR de segundo orden. $2 < r < 4$.

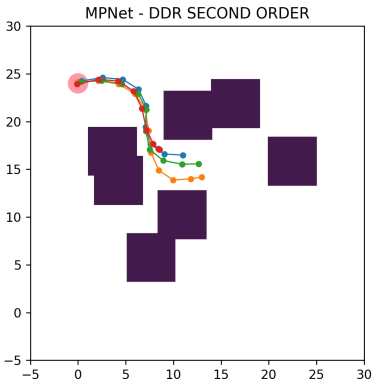
	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	13.231	12.851	13.696	11.357
Máximo	16.392	17.082	15.118	24.278
Media	14.661	14.668	14.423	17.041
SD	1.617	1.864	0.681	5.193
Éxitos (%)	14.0	19.0	13.0	13.0
Rechazos (%)	0.0	0.0	0.0	0.0

Tabla 6.40: Resultados estadísticos. Pasaje estrecho. DDR de segundo orden. $4 < r < 6$.

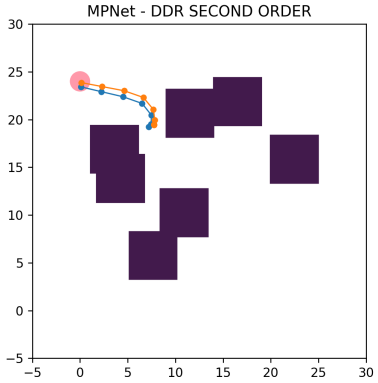
Se puede observar que el método de planificación tiene mayor dificultad para encontrar una trayectoria factible que en los casos anteriores. El porcentaje más alto de éxitos es del 60 %, correspondiente a las redes entrenadas con 600 muestras, por lo tanto, la tasa de éxitos en general es baja. El desempeño del método se empobrece significativamente al planificar fuera de la región de entrenamiento.



(a) $0 < r < 2$.



(b) $2 < r < 4$.



(c) $4 < r < 6$.

Figura 6.18: Ejemplos de trayectorias obtenidas al usar una red entrenada con 800 muestras. Se cruza un pasaje estrecho para llegar a la región objetivo. Robot de manejo diferencial con dinámica de segundo orden.

Los resultados de esta sección muestran que la obtención de los controles y tiempos de aplicación para la solución del problema de planificación mediante la metodología propuesta puede dificultarse al tratar de recorrer un pasaje estrecho. A su vez, se observa que con el *robot omnidireccional* es más sencillo resolver el problema de planificación que al utilizar el *DDR de primer y segundo orden*, pues el porcentaje de éxito se mantiene alto aún cuando se toman en consideración estados iniciales ajenos a la región de entrenamiento. Al utilizar método para los sistemas *DDR de primer y segundo orden* se presentan bajas tasas de éxito y se disminuye significativamente el desempeño del método al tomar estados iniciales que no corresponden a la región de entrenamiento.

6.4. Variación del intervalo del tiempo de muestreo

En esta sección se realiza un experimento similar al expuesto en la sección 6.3, el cual consiste en ejecutar el método de planificación propuesto sobre el mismo pasaje estrecho pero variando los parámetros del SST^* para la obtención de los datos de entrenamiento. Particularmente, se modifica el parámetro de entrada δ_{s0}, así como el intervalo de muestreo en el tiempo de aplicación de las entradas de control $\Delta t \in [0, \Delta t_{max}]$. Estas modificaciones se hacen con el objetivo de agregar maniobrabilidad en la navegación del pasaje estrecho. Se realizan experimentos con modelos neuronales entrenados con 800 muestras dado el entorno de la figura 6.14. Dicho esto, los parámetros utilizados en el SST^* son los siguientes:

- Intervalo de muestreo de controles: $u_i \in [-1, 1]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, \Delta t_{max}]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 0.4$
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 5$.
- Parámetro de reducción $\xi = 0.95$.

Las tablas 6.41, 6.42 y 6.43 exponen los resultados de los estadísticos obtenidos para la planificación en la vecindad inicial, media y lejana respectivamente. Se pueden remarcar algunas diferencias con los resultados obtenidos en el experimento previo: en la región inicial, al considerar $\Delta t_{max} = 1$, aumenta ligeramente el número de éxitos. Sin embargo, al tomar estados iniciales de otras vecindades, la tasa de éxitos en todos los casos resulta ser muy baja. Los nuevos experimentos presentan un valor inferior en el costo medio, mínimo y máximo de trayectoria en comparación con el experimento anterior.

	$\Delta t_{max} = 1$	$\Delta t_{max} = 2$	Previo
Mínimo	8.823	8.361	10.684
Máximo	12.265	9.041	12.735
Media	10.023	8.717	11.560
SD	1.584	0.281	0.653
Éxitos	61.0	42.0	47.0
Rechazos	0.0	0.0	0.0

Tabla 6.41: Resultados estadísticos. Variación de parámetros en el *SST**. DDR de segundo orden. $0 < r < 2$.

	$\Delta t_{max} = 1$	$\Delta t_{max} = 2$	Previo
Mínimo	10.083	8.882	11.226
Máximo	13.463	10.686	14.068
Media	11.252	9.557	12.515
SD	1.636	0.649	1.314
Éxitos	34.0	36.0	33.0
Rechazos	0.0	0.0	0.0

Tabla 6.42: esultados estadísticos. Variación de parámetros en el *SST**. DDR de segundo orden. $2 < r < 4$.

	$\Delta t_{max} = 1$	$\Delta t_{max} = 2$	Previo
Mínimo	10.922	9.697	11.357
Máximo	14.127	11.143	24.278
Media	12.262	10.371	17.041
SD	1.557	0.700	5.193
Éxitos	8.0	13.0	13.0
Rechazos	0.0	0.0	0.0

Tabla 6.43: esultados estadísticos. Variación de parámetros en el *SST**. DDR de segundo orden. $4 < r < 6$.

6.5. Tiempo de generación de trayectorias

En esta sección, el método propuesto se somete a pruebas que comparan el tiempo de cálculo necesario para generar una trayectoria, así como la calidad de estas, con respecto a los caminos generados por el planificador *SST**. Usando la región "A" del primer entorno de trabajo (figura 6.2), así como su respectiva zona objetivo, se seleccionan 20 estados iniciales de prueba en el vecindario de entrenamiento ($0 < r < 2$). Para los robots omnidireccional y DDR de primer orden se utilizan cinco modelos entrenados con 400 muestras, mientras que para el DDR de segundo orden se utilizan cinco redes entrenadas con 800 muestras, donde cada modelo neuronal tiene una semilla diferente. Los resultados obtenidos con el *SST** indican el promedio de 5 experimentos bajo las mismas condiciones iniciales, es decir, para cada estado inicial x_{init} se realiza la planificación cinco veces y se registra el promedio de costo y tiempo de ejecución. Los experimentos para el *SST** terminan al encontrar la primer trayectoria factible de x_{init} a $\mathcal{X}_{goal}$. Los parámetros del *SST**, tanto para las pruebas como para el entrenamiento son los mismos que se mencionan en la sección 6.1 para cada sistema dinámico.

6.5.1. Tiempo de cómputo: robot omnidireccional

La tabla 6.44 muestra los resultados obtenidos al planificar a partir de los estados iniciales de prueba hacia la región objetivo tomando en consideración el modelo de robot omnidireccional. En este caso se puede observar que el tiempo promedio de cómputo para el método propuesto supera por dos órdenes de magnitud al tiempo promedio del método *SST**. Por otro lado, la media del costo resulta ligeramente superior en comparación al costo obtenido con el *SST**. La desviación estándar en el tiempo de cómputo indica que la variación en el tiempo para generar una trayectoria al utilizar el método propuesto es muy baja, en contraste, el planificador *SST** supera por tres órdenes de magnitud a la desviación estándar del método propuesto. En ambos casos, se obtiene un porcentaje de éxitos del 100 %.

	Método propuesto	SST*
Mín. tiempo	0.049	1.341
Máx. tiempo	0.059	44.879
Med. tiempo	0.054	2.190
SD tiempo	0.003	4.423
Mín. costo	23.143	16.444
Máx. costo	26.636	28.985
Med. costo	24.802	20.001
SD costo	1.255	2.160
Éxitos (%)	100.0	100.0

Tabla 6.44: Comparación entre el tiempo de cómputo de trayectorias utilizando la metodología propuesta vs *SST**. Robot omnidireccional.

6.5.2. Tiempo de cómputo: DDR de primer orden

En la tabla 6.45 se presentan los resultados al considerar el sistema DDR con dinámica de primer orden. Al igual que con el robot omnidireccional, el tiempo de generación de trayectorias para el método propuesto es inferior por dos órdenes de magnitud en comparación con el tiempo promedio del *SST**. En esta ocasión, la media del costo para el método propuesto tiene un valor significativamente menor con respecto al *SST**. La desviación estándar para el tiempo de cómputo usando el método propuesto es inferior a la del *SST** por tres órdenes de magnitud. La tasa de éxitos para ambos métodos es del 100 %.

	Método propuesto	SST*
Mín. tiempo	0.047	1.812
Máx. tiempo	0.056	37.591
Med. tiempo	0.050	5.832
SD tiempo	0.002	5.817
Mín. costo	20.295	21.669
Máx. costo	22.890	70.084
Med. costo	21.335	39.164
SD costo	0.866	9.958
Éxitos (%)	100.0	100.0

Tabla 6.45: Comparación entre el tiempo de cómputo de trayectorias utilizando la metodología propuesta vs *SST**. DDR de primer orden.

6.5.3. Tiempo de cómputo: DDR de segundo orden

La tabla 6.46 muestra los resultados estadísticos obtenidos al utilizar el robot DDR con dinámica de segundo orden. En este caso, el tiempo promedio de generación de trayectorias es un orden de magnitud inferior para el método propuesto con respecto al método *SST**. La media del costo de trayectoria resulta inferior para el método propuesto en comparación con el *SST**. La desviación estándar para el tiempo cómputo en el método propuesto es inferior a la del *SST** por tres órdenes de magnitud. El método propuesto obtuvo un porcentaje de éxitos de 90 %, mientras que el *SST** obtuvo el 100 % de éxitos.

	Método propuesto	SST*
Mín. tiempo	0.090	1.967
Máx. tiempo	0.116	52.540
Med. tiempo	0.102	5.665
SD tiempo	0.007	7.362
Mín. costo	10.907	10.767
Máx. costo	12.674	32.783
Med. costo	11.580	15.576
SD costo	0.503	3.336
Éxitos (%)	90.0	100.0

Tabla 6.46: Comparación entre el tiempo de cómputo de trayectorias utilizando la metodología propuesta vs *SST**, DDR de segundo orden.

Los resultados de los experimentos de esta sección muestran que al utilizar los modelos neuronales para la generación de controles, el tiempo de cómputo para la solución del problema de planificación es drásticamente menor (hasta por dos ordenes de magnitud) al tiempo de cómputo utilizando el planificador *SST**. El costo promedio de trayectoria al utilizar el método propuesto con el *DDR de primer y segundo orden* es inferior al costo promedio de la primer trayectoria obtenida por el *SST**. Para el *robot omnidireccional* el costo promedio de trayectoria utilizando el método propuesto es ligeramente superior al obtenido mediante el *SST**.

6.6. Comparación con MPC-MPNet

Los estudios presentes en [2] utilizan la metodología descrita en la sección 3.5 de este trabajo. Dentro de estos estudios se incluye el resultado de la planificación de un robot automóvil en un entorno con múltiples pasajes estrechos. En esta sección se considera el mismo entorno y sistema dinámico descrito en [2], pero se utiliza la metodología propuesta en el capítulo 4. Se entrena una red de inferencia con 400 muestras para un robot automóvil. El espacio de trabajo para este experimento se muestra en la figura 6.19, donde los puntos rojos representan la distribución de los estados iniciales de las trayectorias de entrenamiento y la zona rosa denota la región objetivo.

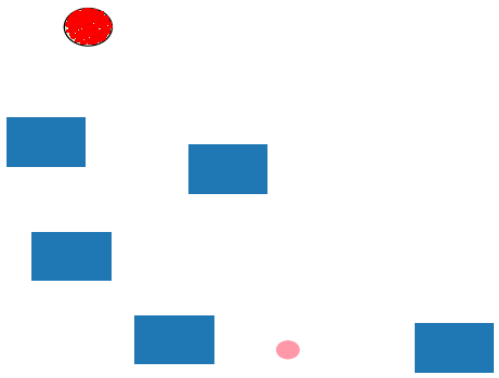


Figura 6.19: Entorno de trabajo y distribución de los estados iniciales en el entrenamiento.

Se consideran los siguientes valores

- Orientación inicial $\theta_0 = \pi$.
- Centro de la región inicial: $P_c = (-14, 25)$.
- Posición del estado objetivo: $x'_{goal} = (4, -12.2)$.

Se utilizan los intervalos de muestreo de controles y tiempos de propagación descritos en [2]. Los parámetros del SST^* utilizados para generar los datos de entrenamiento son los siguientes:

- Intervalo de muestreo del control de velocidad: $u_v \in [0, 2]$.
- Intervalo de muestreo del control de ángulo: $u_\delta \in [-0.5, 0.5]$.
- Intervalo de tiempo de propagación: $\Delta t \in [0, 2]$.
- Iteraciones iniciales: $N_0 = 30000$.
- Parámetro de selección: $\delta_{BN0} = 4.0$.
- Parámetro de poda: $\delta_{s0} = 1.2$
- Ejecuciones del SST como subrutina de SST*: $N_{SST} = 5$.
- Parámetro de reducción $\xi = 0.95$.

La tabla 6.47 muestra el tiempo de cómputo y el costo de trayectoria para el método propuesto y el uso de MPC-MPNet. Se puede observar que en este experimento el tiempo de computo utilizando el método propuesto es inferior por tres órdenes de magnitud al tiempo de computo usando MPC-MPNet. Adicionalmente el costo de trayectoria usando la red de inferencia propuesta es inferior al costo de la trayectoria del MPC-MPNet por casi 50 %.

	Tiempo (s)	Costo (s)
M. propuesto	7.8×10^{-2}	28.6
MPC-MPNet	12.3	57.0

Tabla 6.47: Comparación entre el tiempo de cómputo y costo de trayectoria entre el método propuesto y MPC-MPNet.

La figura 6.20 muestra la trayectoria obtenida al aplicar los controles generados por la red de inferencia propuesta. El elemento de color verde denota al robot en su estado inicial, mientras que el elemento de color rojo representa su estado final. En la figura 6.21 se observa la trayectoria expuesta en [2] para el mismo sistema dinámico bajo el mismo entorno al utilizar el *MPC-MPNet*.

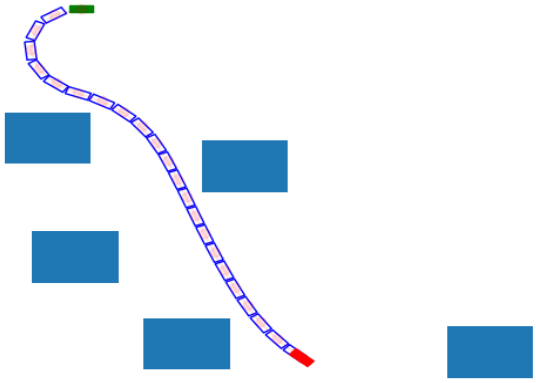


Figura 6.20: Trayectoria generada utilizando un modelo de 400 muestras. Robot tipo automóvil.

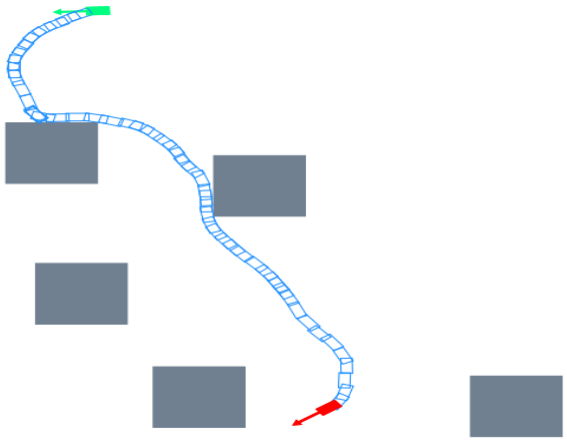


Figura 6.21: Trayectoria obtenida mediante MPC-MPNet para el robot tipo automóvil. Extraído de [3].

6.7. Discusión de los resultados

Los experimentos estudiados sugieren que existen varias situaciones en donde es conveniente el uso del método propuesto para la generación de trayectorias. En general, los experimentos en 6.1 presentan un porcentaje de éxito superior al 90 %, al utilizar cualquier sistema dinámico estudiado, si se inicia la planificación en el mismo vecindario de donde se obtuvieron las trayectorias de entrenamiento. Una observación interesante es el hecho que el robot omnidireccional *generaliza mejor* el proceso de planificación al tomar estados iniciales fuera de la región inicial, es decir, su porcentaje de éxito decae levemente sin alterar abruptamente el costo de los resultados; el DDR de primer orden también es capaz de generalizar trayectorias de este modo en un grado menor, por otra parte, el DDR de segundo orden, en su mayoría, se desempeña pobremente al tratar de generar trayectorias empezando por otros vecindarios. Debido a esto, surge la hipótesis que el grado de generalización de trayectorias podría estar relacionado con las propiedades de los sistemas robóticos tales como la dimensión del espacio de estados $\mathcal{X}$, deriva, holonomía, controlabilidad, etc.

Al utilizar el método propuesto surgen las siguientes preguntas: ¿cómo se puede aumentar la tasa de éxito de los modelos?, ¿qué influye en la calidad de las trayectorias generadas?. Los resultados de los experimentos en 6.2 sugieren que el porcentaje de éxito puede estar relacionado con el número de muestras que se utilicen en el entrenamiento. Por ejemplo, los modelos entrenados con 3000 y 4000 muestras presentan una mayor tasa de éxito en comparación con los modelos que fueron entrenados con un menor número de muestras (tales como los mostrados en la sección 6.1). Sin embargo, parámetros relacionados con el entrenamiento de las redes, por ejemplo, el número de épocas, optimizador, factor de aprendizaje, número de capas y neuronas, etc., juegan un papel muy importante en el comportamiento de la red de inferencia. El mismo experimento también expone que el costo promedio de las trayectorias generadas no difiere significativamente entre modelos. Por otra parte, los resultados en el experimento 6.4 sugieren que el costo de trayectoria está mayormente influenciado por los valores utilizados para la generación de los datos de entrenamiento. Particularmente,

la calidad de las muestras generadas por el *SST** se ve afectada por su parámetro de entrada δ_{s0}: entre más pequeño sea este valor, se generan trayectorias de mejor calidad a cambio de obtener una estructura de datos más densa, y por ende, se requiere un mayor tiempo de cómputo [3].

El experimento de planificación sobre pasajes estrechos para el DDR de segundo orden (6.3.3) exhibe que el porcentaje de éxito en este sistema podría etiquetarse como *bajo* con un 60 % en el mejor de los casos (vecindad inicial, 600 muestras). Sin embargo, este porcentaje supera al del método *SST** (bajo los parámetros del entrenamiento), el cual es del 34.9 % (349 éxitos de 1000 experimentos).

En la sección 6.5 se menciona que el tiempo de cómputo para la generación de trayectorias utilizando las *redes de inferencia de controles* es inferior hasta dos órdenes de magnitud en comparación con el planificador *SST**. Inclusive, para el DDR de primer y segundo orden las trayectorias presentan una media de costo menor al resultado del *SST**, por lo que se abre a la posibilidad que para sistemas robóticos más complejos se obtengan trayectorias de buena calidad en poco tiempo.

Los resultados de la sección 6.6 muestran que el método de planificación mediante la red de inferencia propuesta es capaz de competir con los modelos neuronales de planificación presentes en el estado del arte, pues en el caso estudiado se obtuvo un mejor desempeño contra la red de planificación MPC-MPNet al considerar un robot automóvil.

Capítulo 7

Conclusiones y trabajo futuro

En este trabajo se presentó una metodología que hace uso de redes neuronales capaces de inferir entradas de control y tiempos de aplicación de los mismos para solucionar el problema de planificación en sistemas robóticos con restricciones diferenciales. Esto contrasta con otros métodos de planificación que proponen redes neuronales que infieren estados que sirven como guía para que otro método complementario genere los controles que respeten la dinámica del sistema. Nuestro método resulta ser capaz de encontrar trayectorias factibles en un tiempo de computo de hasta tres órdenes de magnitud menor a los planificadores en el estado del arte, tal como *MPC-MPnet* y *SST**.

Las *redes de inferencia de controles*, al ser entrenadas con datos generados por el planificador *SST**, el cual entrega entradas de control que producen trayectorias asintóticamente óptimas bajo el costo del tiempo necesario para alcanzar una región objetivo. Los resultados de los experimentos indican que este método tiene un alto porcentaje de éxito al planificar sobre regiones consideradas en el entrenamiento de los modelos, y a su vez, es posible generalizar el proceso de planificación en regiones ajenas a este.

Como trabajo futuro, se espera utilizar esta metodología para diferentes sistemas dinámicos, por ejemplo, robots con movimiento en tres dimensiones, para comparar su desempeño con respecto a otros métodos que hacen uso de redes neuronales. Se espera hacer un análisis detallado para encontrar los factores que influyen directamente en

el porcentaje de éxito así como la estimación del número de muestras necesarias para lograr la planificación en todo el espacio de trabajo.

Apéndice A

Comportamiento estocástico por desconexión

En este anexo se presentan estudios análogos a los expuestos en la secciones 6.1 y 6.2. Se utilizan los modelos de inferencia de manera parcial, es decir, se aplica *desconexión* (*dropout*) en todas las capas ocultas para cada inferencia, esto da lugar a una generación no determinística de las entradas de control y tiempos de aplicación. Los resultados en A.1 presentan valores estadísticos para las regiones A y B del primer entorno que se muestran en las figuras 6.1a y 6.1b respectivamente. El anexo A.2 presenta resultados para el experimento de aumento de datos en el robot de manejo diferencial de segundo orden, utilizando la región A del primer entorno. Para cada caso, el umbral de rechazo se indica en la tabla 6.1.

A.1. Regiones A y B del primer entorno.

Primer entorno, región A, robot omnidireccional

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	23.027	23.053	23.043	23.064
Máximo	27.251	27.592	27.589	27.150
Media	25.044	25.029	25.039	24.975
SD	1.292	1.401	1.347	1.352
Éxitos (%)	98.0	100.0	100.0	98.0
Rechazos (%)	0.0	0.0	0.0	0.0

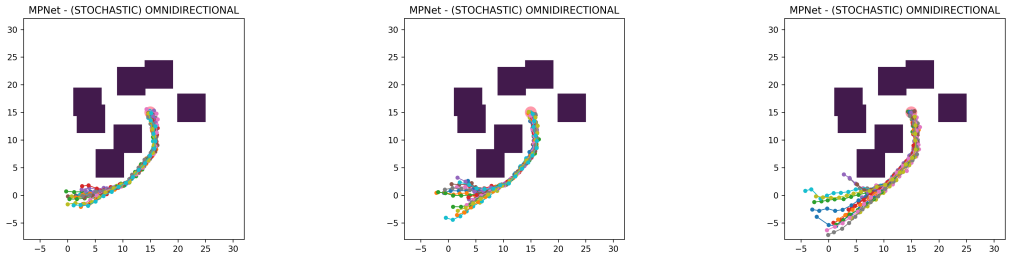
Tabla A.1: Resultados estadísticos en la región "A" del primer entorno. Robot omnidireccional. Comportamiento estocástico. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	21.585	21.620	21.924	21.497
Máximo	29.150	29.825	29.302	29.091
Media	25.175	25.360	25.156	24.965
SD	2.165	2.470	2.291	2.261
Éxitos (%)	98.0	99.0	99.0	99.0
Rechazos (%)	0.0	0.0	0.0	0.0

Tabla A.2: Resultados estadísticos en la región "A" del primer entorno. Robot omnidireccional. Comportamiento estocástico. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	20.034	20.275	20.138	20.259
Máximo	32.587	32.761	31.855	31.466
Media	25.890	25.836	25.520	25.390
SD	3.945	3.924	3.688	3.758
Éxitos (%)	87.0	88.0	92.0	90.0
Rechazos (%)	0.0	0.0	0.0	0.0

Tabla A.3: Resultados estadísticos en la región "A" del primer entorno. Robot omnidireccional. Comportamiento estocástico. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura A.1: Generación de caminos utilizando la red propuesta con *desconexión* durante la etapa de inferencia, entrenamiento de 400 trayectorias, estados iniciales en la primer región, robot omnidireccional.

Primer entorno, región A, DDR primer orden

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	19.493	19.474	19.792	19.948
Máximo	23.288	23.026	22.954	22.992
Media	21.301	21.227	21.373	21.396
SD	1.007	1.030	0.992	0.931
Éxitos (%)	94.0	100.0	100.0	100.0
Rechazos (%)	0.0	0.0	0.0	0.0

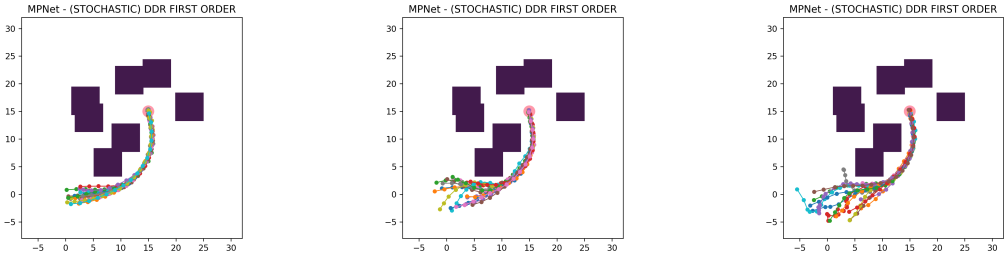
Tabla A.4: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de primer orden. Comportamiento estocástico. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	19.492	19.455	19.384	19.138
Máximo	30.795	32.354	26.195	25.676
Media	22.767	23.113	22.212	22.077
SD	3.183	3.657	1.999	2.045
Éxitos (%)	78.0	79.0	92.0	90.0
Rechazos (%)	1.0	1.0	0.0	0.0

Tabla A.5: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de primer orden. Comportamiento estocástico. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	17.425	17.364	17.849	17.638
Máximo	31.803	35.360	33.415	38.017
Media	23.655	24.477	23.845	24.291
SD	4.396	6.107	4.420	5.188
Éxitos (%)	68.0	55.0	74.0	75.0
Rechazos (%)	6.0	16.0	6.0	8.0

Tabla A.6: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de primer orden. Comportamiento estocástico. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura A.2: Generación de caminos utilizando la red propuesta con *desconexión* durante la etapa de inferencia, entrenamiento de 400 trayectorias, estados iniciales en la primer región, robot de manejo diferencial de primer orden.

Primer entorno, región A, DDR segundo orden

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	11.082	11.323	11.112	10.851
Máximo	14.936	14.969	13.380	13.935
Media	12.429	12.494	12.000	11.966
SD	1.205	1.520	0.659	0.920
Éxitos (%)	47.0	50.0	59.0	60.0
Rechazos (%)	1.0	0.0	0.0	0.0

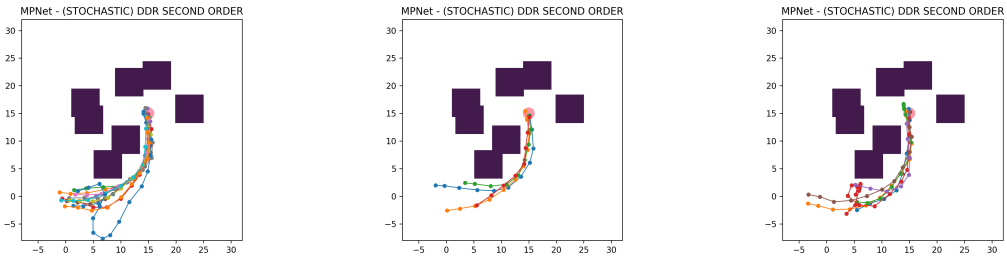
Tabla A.7: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de segundo orden. Comportamiento estocástico. $0 < r < 2$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	11.234	11.640	10.691	10.843
Máximo	16.819	19.007	13.884	13.623
Media	13.722	14.572	11.892	12.242
SD	2.871	4.283	1.152	1.068
Éxitos (%)	16.0	22.0	36.0	26.0
Rechazos (%)	0.0	1.0	0.0	0.0

Tabla A.8: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de segundo orden. Comportamiento estocástico. $2 < r < 4$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	15.227	13.186	11.830	11.494
Máximo	18.770	19.823	19.784	21.160
Media	16.643	15.712	13.975	14.863
SD	2.549	3.403	4.146	3.937
Éxitos (%)	10.0	14.0	31.0	27.0
Rechazos (%)	0.0	0.0	1.0	2.0

Tabla A.9: Resultados estadísticos en la región "A" del primer entorno. Robot DDR de segundo orden. Comportamiento estocástico. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura A.3: Generación de caminos utilizando la red propuesta con *desconexión* durante la etapa de inferencia, entrenamiento de 800 trayectorias, estados iniciales en la primer región, robot de manejo diferencial de segundo orden.

Primer entorno, región B, robot omnidireccional

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	18.070	18.033	17.831	18.211
Máximo	23.419	22.845	23.229	23.469
Media	20.392	20.125	20.324	20.389
SD	1.499	1.605	1.579	1.569
Éxitos (%)	99.0	97.0	98.0	99.0
Rechazos (%)	0.0	0.0	0.0	0.0

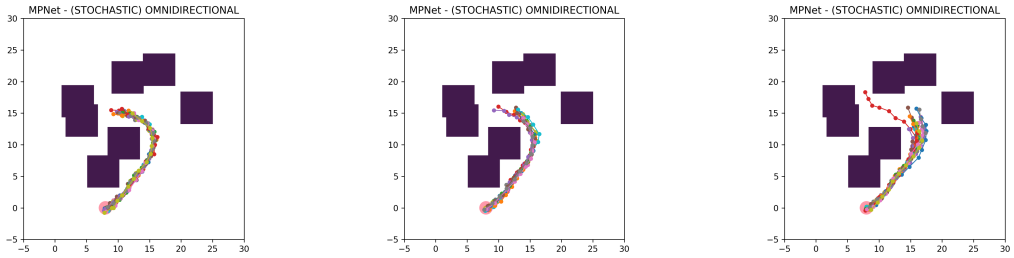
Tabla A.10: Resultados estadísticos en la región "B" del primer entorno. Robot omnidireccional. Comportamiento estocástico. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	16.858	16.237	16.633	16.422
Máximo	23.334	23.163	23.465	22.954
Media	18.966	18.849	18.936	18.784
SD	1.889	2.124	2.114	1.852
Éxitos (%)	98.0	99.0	99.0	99.0
Rechazos (%)	0.0	0.0	0.0	0.0

Tabla A.11: Resultados estadísticos en la región "B" del primer entorno. Robot omnidireccional. Comportamiento estocástico. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	13.719	13.248	13.624	13.366
Máximo	25.569	25.437	25.749	24.591
Media	17.246	17.159	17.385	17.071
SD	2.856	2.872	3.087	2.782
Éxitos (%)	97.0	97.0	100.0	97.0
Rechazos (%)	0.0	0.0	0.0	0.0

Tabla A.12: Resultados estadísticos en la región "B" del primer entorno. Robot omnidireccional. Comportamiento estocástico. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura A.4: Generación de caminos utilizando la red propuesta con *desconexión* durante la etapa de inferencia, entrenamiento de 400 trayectorias, estados iniciales en la segunda región, robot omnidireccional.

Primer entorno, región B, DDR primer orden

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	10.416	10.068	10.202	9.997
Máximo	11.741	12.375	17.128	14.400
Media	11.003	10.984	11.350	11.175
SD	0.549	0.712	2.448	1.396
Éxitos (%)	26.0	51.0	65.0	66.0
Rechazos (%)	0.0	0.0	2.0	0.0

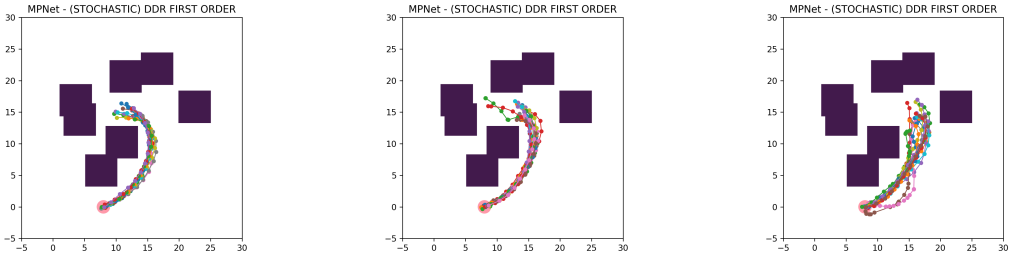
Tabla A.13: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de primer orden. Comportamiento estocástico. $0 < r < 2$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	9.805	9.632	9.545	9.674
Máximo	11.520	12.404	15.883	13.947
Media	10.690	10.894	11.166	10.998
SD	0.644	0.990	2.912	1.323
Éxitos (%)	26.0	40.0	53.0	47.0
Rechazos (%)	0.0	0.0	0.0	0.0

Tabla A.14: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de primer orden. Comportamiento estocástico. $2 < r < 4$.

	100 muestras	200 muestras	300 muestras	400 muestras
Mínimo	14.409	9.804	10.498	10.481
Máximo	21.939	15.871	17.483	15.475
Media	18.174	11.874	13.051	12.341
SD	4.620	3.686	2.798	2.066
Éxitos (%)	10.0	20.0	23.0	25.0
Rechazos (%)	1.0	1.0	2.0	1.0

Tabla A.15: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de primer orden. Comportamiento estocástico. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura A.5: Generación de caminos utilizando la red propuesta con *desconexión* durante la etapa de inferencia, entrenamiento de 400 trayectorias, estados iniciales en la segunda región, robot de manejo diferencial de primer orden.

Primer entorno, región B, DDR segundo orden

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	9.899	9.955	9.962	10.040
Máximo	12.996	13.244	14.532	11.711
Media	10.949	10.970	11.143	10.824
Varianza	1.532	1.014	2.025	0.272
Éxitos	35.0	54.0	61.0	55.0
Rechazos	0.0	0.0	0.0	1.0

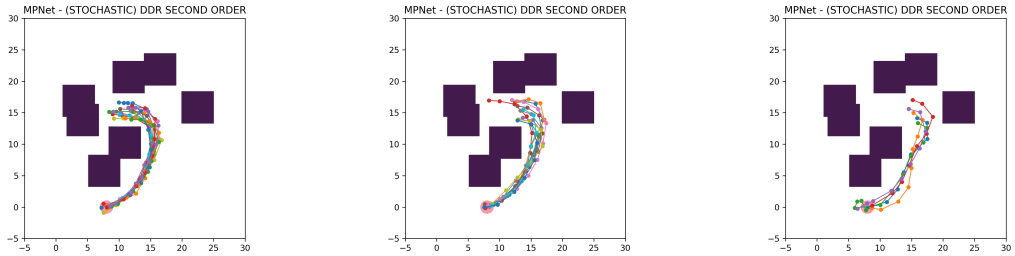
Tabla A.16: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de segundo orden. Comportamiento estocástico. $0 < r < 2$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	10.305	9.589	9.735	9.656
Máximo	11.989	12.317	13.552	15.314
Media	10.907	10.734	11.070	11.098
Varianza	0.563	0.631	1.702	4.587
Éxitos	18.0	53.0	51.0	54.0
Rechazos	0.0	0.0	1.0	0.0

Tabla A.17: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de segundo orden. Comportamiento estocástico. $2 < r < 4$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	13.152	8.559	11.314	8.149
Máximo	17.361	11.341	15.154	19.713
Media	14.189	9.480	12.814	11.618
Varianza	9.456	3.835	5.164	25.472
Éxitos	10.0	13.0	13.0	22.0
Rechazos	2.0	0.0	4.0	1.0

Tabla A.18: Resultados estadísticos en la región "B" del primer entorno. Robot DDR de segundo orden. Comportamiento estocástico. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura A.6: Generación de caminos utilizando la red propuesta con *desconexión* durante la etapa de inferencia, entrenamiento de 800 trayectorias, estados iniciales en la segunda región, robot de manejo diferencial de segundo orden.

A.2. Aumento de muestras: DDR de segundo orden

Primer entorno, región A, DDR segundo orden

	1000 muestras	2000 muestras	3000 muestras	4000 muestras
Mínimo	11.013	10.967	11.056	10.894
Máximo	14.524	13.401	13.425	13.166
Media	12.120	12.035	11.887	11.943
SD	1.229	0.746	0.699	0.650
Éxitos (%)	49.0	66.0	69.0	70.0
Rechazos (%)	0.0	0.0	0.0	0.0

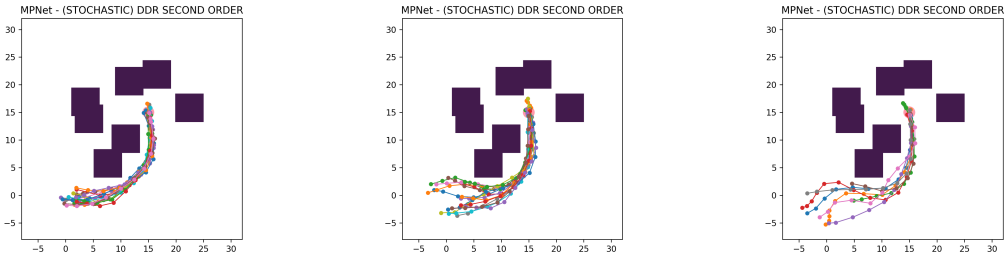
Tabla A.19: Resultados estadísticos. Aumento de muestras. DDR de segundo orden. Comportamiento estocástico. $0 < r < 2$.

	1000 muestras	2000 muestras	3000 muestras	4000 muestras
Mínimo	11.049	10.895	10.831	10.901
Máximo	15.684	14.846	15.822	15.615
Media	12.692	12.554	12.830	12.627
SD	1.803	1.240	1.503	1.474
Éxitos (%)	40.0	51.0	53.0	48.0
Rechazos (%)	1.0	0.0	0.0	0.0

Tabla A.20: Resultados estadísticos. Aumento de muestras. DDR de segundo orden. Comportamiento estocástico. $2 < r < 4$.

	1000 muestras	2000 muestras	3000 muestras	4000 muestras
Mínimo	11.076	11.689	10.857	11.300
Máximo	21.121	20.786	20.560	17.401
Media	14.711	15.077	14.763	13.858
SD	4.387	3.316	3.326	2.083
Éxitos (%)	26.0	32.0	45.0	48.0
Rechazos (%)	4.0	0.0	3.0	0.0

Tabla A.21: Resultados estadísticos. Aumento de muestras. DDR de segundo orden. Comportamiento estocástico. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura A.7: Trayectorias obtenidas mediante el uso de la red entrenada con 4000 caminos empleando *desconexión* en la etapa de inferencia, se toman los estados iniciales propuestos en la primer región de entrenamiento. Robot de manejo diferencial con dinámica de segundo orden.

Los resultados de este estudio indican que por lo general no es recomendable utilizar *desconexión* en todos los pasos de inferencia, ya que, al comparar los resultados de este anexo con los de las secciones 6.1 y 6.2, se puede observar que en la mayoría de las veces el costo medio de trayectoria es superior al aplicar *desconexión* y no existe una mejora significativa en el porcentaje de éxitos. Adicionalmente, la tasa de rechazos es ligeramente mayor al aplicar desconexión, pues usualmente se logra concretar el problema de planificación pues las trayectorias generadas tienen un alto costo.

Apéndice B

Trayectorias de costo alto y variable

En este anexo se presenta un estudio adicional para los experimentos de la sección 6.3. Al utilizar el método *SST** para generar la base de datos de entrenamiento tomando como entorno de trabajo el mostrado en la figura 2.3, se pueden generar trayectorias no deseadas de alto costo, i.e., caminos que no se encuentran sobre el pasaje estrecho. Los resultados presentes en este anexo corresponden a los valores estadísticos de la planificación utilizando redes entrenadas con estas trayectorias de alto costo (anexo B.1), así como utilizando redes entrenadas de manera híbrida, es decir, en la etapa de entrenamiento se utilizan trayectorias de alto costo, así como trayectorias que pasan sobre el pasaje estrecho (anexo B.2). Los experimentos se realizan utilizando el robot de manejo diferencial de segundo orden.

B.1. Trayectorias de alto costo

Primer entorno, DDR segundo orden

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	30.546	29.719	30.103	30.159
Máximo	33.286	32.567	32.830	32.871
Media	31.941	30.962	31.255	31.193
Varianza	0.668	0.889	0.772	0.750
Éxitos	94.0	94.0	97.0	93.0
Rechazos	0.0	1.0	0.0	0.0

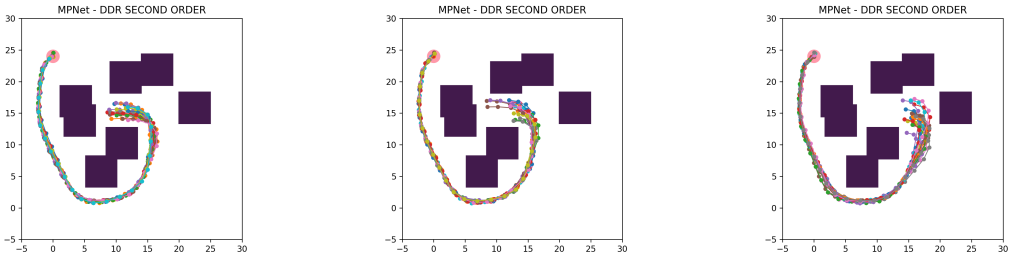
Tabla B.1: Resultados estadísticos. Trayectorias de alto costo. DDR de segundo orden. $0 < r < 2$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	30.151	29.339	30.086	29.571
Máximo	34.213	32.401	32.903	32.683
Media	31.494	30.435	30.865	30.648
Varianza	1.550	0.964	0.888	0.928
Éxitos	90.0	90.0	89.0	94.0
Rechazos	0.0	1.0	0.0	0.0

Tabla B.2: Resultados estadísticos. Trayectorias de alto costo. DDR de segundo orden. $2 < r < 4$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	29.785	28.601	29.907	29.317
Máximo	34.075	31.353	33.781	32.084
Media	31.949	30.019	31.171	30.443
Varianza	2.511	0.543	1.725	0.700
Éxitos	46.0	77.0	60.0	72.0
Rechazos	0.0	1.0	0.0	0.0

Tabla B.3: Resultados estadísticos. Trayectorias de alto costo. DDR de segundo orden. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura B.1: Caminos obtenidos al usar una red entrenada con 800 muestras. Se opta por tomar un camino extenso en lugar de pasar a través de dos obstáculos. Robot de manejo diferencial con dinámica de segundo orden.

B.2. Trayectorias de costo variable

Primer entorno, DDR segundo orden

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	29.785	28.601	29.907	29.317
Máximo	34.075	31.353	33.781	32.084
Media	31.949	30.019	31.171	30.443
Varianza	2.511	0.543	1.725	0.700
Éxitos	46.0	77.0	60.0	72.0
Rechazos	0.0	1.0	0.0	0.0

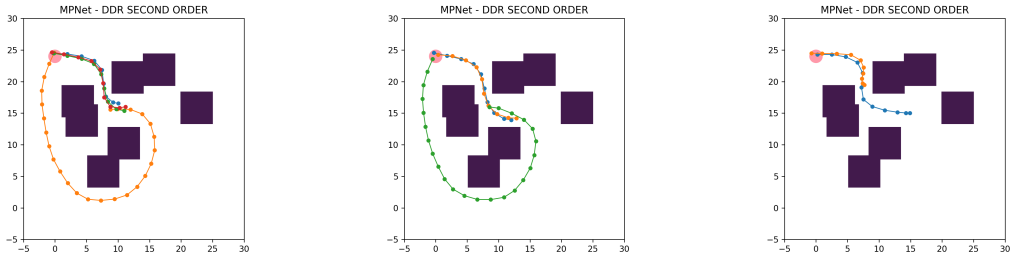
Tabla B.4: Resultados estadísticos. Trayectorias de costo variable. DDR de segundo orden. $0 < r < 2$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	29.982	27.002	21.834	20.271
Máximo	34.935	34.929	27.707	28.392
Media	32.788	31.546	24.976	22.748
Varianza	8.668	20.804	17.341	31.164
Éxitos	22.0	30.0	18.0	11.0
Rechazos	1.0	0.0	0.0	0.0

Tabla B.5: Resultados estadísticos. Trayectorias de costo variable. DDR de segundo orden. $2 < r < 4$.

	200 muestras	400 muestras	600 muestras	800 muestras
Mínimo	15.478	22.089	10.524	14.141
Máximo	20.995	27.667	11.004	22.586
Media	17.704	25.570	10.711	18.395
Varianza	14.553	13.612	0.158	30.420
Éxitos	15.0	25.0	6.0	10.0
Rechazos	0.0	0.0	0.0	2.0

Tabla B.6: Resultados estadísticos. Trayectorias de costo variable. DDR de segundo orden. $4 < r < 6$.



(a) $0 < r < 2$. **(b)** $2 < r < 4$. **(c)** $4 < r < 6$.

Figura B.2: Caminos obtenidos utilizando una red entrenada con 800 muestras. En algunas ocasiones el método de planificación es capaz de encontrar un camino de bajo costo. Robot de manejo diferencial con dinámica de segundo orden.

Los resultados de este estudio muestran que al planificar con el método propuesto utilizando modelos neuronales entrenados con trayectorias de alto costo la tasa de éxito para el robot *DDR de segundo orden* es alta en la región de entrenamiento y disminuye ligeramente al tomar estados iniciales de los otros vecindarios. Al no planificar sobre el pasaje estrecho, el costo promedio de trayectoria es alto comparado con el expuesto en la sección 6.3. Las trayectorias obtenidas con los modelos neuronales entrenados de manera híbrida presentan una alta varianza en el costo.

Apéndice C

Descripción de las funciones en los algoritmos

En este apartado se da una breve explicación del funcionamiento de cada función que se menciona de manera implícita en los algoritmos de este trabajo.

C.1. Planificadores basados en muestreo

- **Sample**(X): genera una muestra aleatoria de X; usualmente utilizada para muestrear en el espacio de configuraciones o estados.

- **Near**(V, x, r): genera el conjunto de nodos en V que se encuentran a una distancia de x menor o igual a r.

- **Collision_Free**(x, y): devuelve un valor booleano. Será 1 si la trayectoria que va de x a y se encuentra en el espacio libre de colisiones, 0 en otro caso.

- **Nearest**(V, x): se obtiene el elemento de V que se encuentra más cerca, bajo a cierta métrica, de x.

- **Select_Input**(x, y): se selecciona un control $u \in \mathcal{U}$, bajo cierto muestreo o heurística, que logre minimizar la distancia de x a y.

- **New State**(x, u): se obtiene el nuevo estado en el cual se encuentra el sistema dinámico después de la aplicación de u desde x, por algún intervalo de tiempo Δt.

- **Steer**(x, y): dados dos puntos $x, y \in X$, devuelve un punto z tal que minimiza $\|z - y\|$ al mismo tiempo que mantiene $\|z - x\| \leq \rho$, para cierto valor $\rho > 0$:

$$\text{Steer}(x, y) = \arg \min_{z \in \mathcal{B}_{x,\rho}} \|z - y\|.$$

- **Is_Leaf**(x): booleano, devuelve 1 si x es un nodo hoja en la estructura de árbol. 0 en caso contrario

- **Parent**(x): regresa al nodo padre de x, es decir, el nodo que se encuentra inmediatamente conectado con x que va en dirección hacia la raíz.

C.2. Modelos neuronales

- **Pnet**(Z, x, y): llama a la red de planificación para la generación de una nueva configuración $\hat{c} \in \mathcal{C}$.

- **Concatenate**(σ^a, σ^b): Une a dos caminos σ^a y σ^b en un solo arreglo σ.

- **SWAP**(σ^a, σ^b): el camino σ^a cambia de lugar con σ^b. Se utiliza para el proceso de planificación bidireccional. La configuración inicial pasa a ser configuración objetivo, y la configuración objetivo pasa a ser la configuración inicial.

- **Oracle_Planner**(x_{init}, x_{goal}): se manda a llamar a un método clásico para resolver el problema de planificación cuando la red no logra concretar un resultado.

- **Invalid**(σ): booleano, entrega 1 si la trayectoria σ no es factible. 0 en otro caso.

- **Random_Node**(T): se selecciona un nodo aleatorio en la estructura de árbol.

- **Add_To_Tree**(σ, T): se agrega la trayectoria σ a la estructura de árbol desde el nodo recién propagado.
- **Reached**(T, x_{goal}): booleano, regresa 1 si un camino en la estructura de árbol conduce hacia x_{goal}.
- **Extract_Path**(T): se obtiene el camino del árbol que conduce de x_{init} a x_{goal}.
- **Random_Sample**(): muestrea aleatoriamente el espacio de estados libre.
- **Nearest_Neighbor**(B, T): se obtienen los nodos más cercanos a las muestras que se encuentran en el lote B.
- **Control_Inference**(Z, x_{init}, x_{goal}): utiliza la red de inferencia de controles para desplazar al sistema robótico desde x_{init} en dirección a x_{goal}.
- **Control_Validity**(u, t): booleano, verifica si la entrada de control, así como el tiempo de aplicación son válidos.
- **Is_Collision_Free**(x, Z): booleano, se verifica si el estado propagado se encuentra en el espacio libre de estados.
- **Is_Near**(x, x_{goal}, ϵ): booleano, se verifica si el estado x se encuentra a una distancia menor a ϵ de x_{goal}.

Bibliografía

[1] U. Ruiz, R. Murrieta-Cid, and J. L. Marroquin, "Time-optimal motion strategies for capturing an omnidirectional evader using a differential drive robot," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1180–1196, 2013.

[2] L. Li, Y. Miao, A. H. Qureshi, and M. C. Yip, "Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4496–4503, 2021.

[3] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[4] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012.

[5] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[6] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation—a survey," *Robotics and Autonomous systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[7] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[8] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion

planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[9] J. Canny, *The complexity of robot motion planning*. MIT press, 1988.

[10] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[11] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[12] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, no. 2, pp. 121–155, 1993.

[13] J.-P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray, "A motion planner for nonholonomic mobile robots," *IEEE Transactions on robotics and automation*, vol. 10, no. 5, pp. 577–593, 1994.

[14] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," *Robot motion planning and control*, pp. 1–53, 1998.

[15] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[16] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[18] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Computer Science Dept. Oct.*, vol. 98, no. 11, 1998.

[19] M. Sarkar, P. Pradhan, and D. Ghose, "Planning robot motion using deep visual prediction," *arXiv preprint arXiv:1906.10182*, 2019.

[20] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[21] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Advances in neural information processing systems*, vol. 29, 2016.

[22] J. Xu, B. Ni, and X. Yang, "Video prediction via selective sampling," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[25] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[26] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.

[27] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *2013 IEEE International Conference on Robotics and Automation*, pp. 5539–5544, IEEE, 2013.

[28] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[29] H. J. Sussmann, "A general theorem on local controllability," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 158–194, 1987.

[30] A. Bloch and B. Brogliato, "Nonholonomic mechanics and control," *Appl. Mech. Rev.*, vol. 57, no. 1, pp. B3–B3, 2004.

[31] F. Bullo and A. D. Lewis, *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*, vol. 49. Springer, 2019.

[32] S. Sastry, *Nonlinear systems: analysis, stability, and control*, vol. 10. Springer Science & Business Media, 2013.

[33] R. Arteaga, E. Antonio, I. Becerra, and R. Murrieta-Cid, "On the efficiency of the sst planner to find time optimal trajectories among obstacles with a ddr under second order dynamics," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 674–681, 2021.

[34] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2118–2124, IEEE, 2019.

[35] E. F. Morales, R. Murrieta-Cid, I. Becerra, and M. A. Esquivel-Basaldua, "A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning," *Intelligent Service Robotics*, vol. 14, no. 5, pp. 773–805, 2021.

[36] R. Salah, P. Vincent, X. Muller, *et al.*, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. of the 28th International Conference on Machine Learning*, pp. 833–840, 2011.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[38] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *2010 IEEE international conference on robotics and automation*, pp. 2493–2498, IEEE, 2010.

[39] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical analysis*. Cengage learning, 2015.

[40] V. Kálmán, "Omnidirectional wheel simulation–a practical approach," *Acta Technica Jaurinensis*, vol. 6, no. 2, pp. 73–90, 2013.

[41] F. G. Pin and S. M. Killough, "A new family of omnidirectional and holonomic wheeled platforms for mobile robots," *IEEE transactions on robotics and automation*, vol. 10, no. 4, pp. 480–489, 1994.