Análisis de los algoritmos de planificación de trayectorias RRT, PRM y Voronoi en la solución de un laberinto modular controlado por una plataforma de dos GDL

Analysis of RRT, PRM and Voronoi path planning algorithms to solve a modular maze using a two-DOF platform

Iván Palacios Serrano[1]* Christyan Cruz Ulloa[2] Manuel Barraza Rodríguez[2, 3]

Recibido 15 de abril de 2021, aceptado 4 de febrero de 2022
Received: April 15, 2021 Accepted: February 4, 2022

RESUMEN

En este artículo se presenta el análisis de tres técnicas de planificación de trayectorias para resolver un sistema bola-laberinto con una plataforma de dos grados de libertad. El objetivo del sistema es que la bola recorra una trayectoria desde un punto inicial hasta un punto de destino (definidos por el usuario) en el laberinto. Para lo cual se implementaron los algoritmos RRT (Rapidly Exploring Random Trees), PRM (Probabilistic Roadmap) y los diagramas de Voronoi junto con el algoritmo de búsqueda A*. La arquitectura del sistema consta de cuatro subsistemas denominados mecánico, de visión, de planificación y de control. La principal contribución del trabajo es la evaluación de los algoritmos en un sistema físico, así como un análisis de resultados completo (gráfico y analítico). La evaluación experimental se basa en el análisis de cuatro configuraciones diferentes del laberinto y las métricas de tiempo de ejecución y longitud de la trayectoria. En este contexto, se realizaron 20 ejecuciones de cada algoritmo para cada configuración, luego se determinó el tiempo medio y la distancia media junto con sus intervalos de confianza del 95%. Los principales resultados muestran que el algoritmo RRT presenta mayor variación en sus resultados, la trayectoria con mayor longitud y el mejor rendimiento en cuanto a tiempo de ejecución. Por otra parte, el algoritmo PRM genera la trayectoria con menor longitud, sin embargo, tiene el peor rendimiento con respecto al tiempo de ejecución. Finalmente, la técnica con diagramas de Voronoi tiene menor variación en sus datos y presenta la trayectoria más suave y equidistante entre las paredes del laberinto.

Palabras clave: Planificación de trayectorias, RRT, PRM, diagramas de Voronoi, visión por computador, control automático.

ABSTRACT

This paper analyzes three path planning techniques to solve a ball-maze system with a two-degree-of-freedom platform. The system's objective focuses on the ball traveling a path from an initial point to a final point (defined by the user) in the maze. The RRT algorithms (Rapidly Exploring Random Trees), PRM (Probabilistic Roadmap), and Voronoi diagrams were implemented using the A search algorithm.*

[1] Universidad de Cuenca, Red Sísmica del Austro, Facultad de Ingeniería. Cuenca, Ecuador. E-mail: ivan.palacios@ucuenca.edu.ec
[2] Centro de Automática y Robótica-UPM-CSIC, Universidad Politécnica de Madrid - Consejo Superior de Investigaciones Científicas. España. E-mail: ccruz@etsii.upm.es, m.barraza@alumnos.upm.es
[3] Universidad de Tarapacá, Departamento de Ingeniería Eléctrica-Electrónica. Arica, Chile. E-mail: mabarrazar@academicos.uta.cl
* Autor de correspondencia: ivan.palacios@ucuenca.edu.ec

The system architecture consists of four subsystems called mechanical, vision, planning, and control. The main contribution of this work is the evaluation of the algorithms on a physical system and a complete results analysis (graphical and analytical). The experimental tests were performed based on analyzing four different maze configurations, the run time, and path length metrics. In this context, 20 algorithm executions were developed for each configuration, then the meantime and mean length and their 95% confidence intervals were determined. The main results show that the RRT algorithm presents a more significant variation in its data, the longest path length, and the best performance in terms of run time. Moreover, the PRM algorithm generates the path with the shortest length but has the worst performance concerning run time. Finally, the Voronoi diagrams' technique takes less time to execute, has less variation in its data, and presents the smoothest and equidistant path between the maze walls.

Keywords: Path planning, RRT, PRM, Voronoi diagram, computer vision, automatic control.

INTRODUCCIÓN

Las técnicas de planificación de trayectorias son de gran utilidad en la solución y optimización de problemas en múltiples áreas. Por ejemplo, en el campo de la robótica y automatización [1], vehículos autónomos [2], fabricación y ensamblaje de piezas [3], aplicaciones aeroespaciales [4], diseño de medicamentos [5] e incluso en juegos [6]. El objetivo principal es obtener una trayectoria desde un punto inicial hasta un punto de destino en un ambiente dado. Este entorno puede ser muy complejo y con una gran variedad de obstáculos, lo cual implica problemas con alto grado de complejidad para un operador y resulta conveniente el uso de sistemas autónomos [5].

Para la planificación de trayectorias es necesario conocer la información del entorno (v.g. dimensiones, número de obstáculos, posición inicial, punto de destino) previo a la ejecución del algoritmo de planificación. Las métricas principales para evaluar un planificador consisten en la longitud de la trayectoria y el tiempo de ejecución [7] . Por otra parte, entre los algoritmos de planificación más comunes se tienen los basados en muestreo como RRT (Rapidly Exploring Random Trees) [8] o PRM (Probabilistic Roadmap) [9]. Además, existen planificadores discretos que utilizan algoritmos de búsqueda como el A* o el método Dijkstra. Adicionalmente, se dispone de los diagramas de Voronoi como una técnica especial para mantener equidistante la ruta entre los obstáculos y suavizar la trayectoria [10].

La robótica es una de las principales aplicaciones que emplea planificación de trayectorias, en la literatura se presentan diversos trabajos sobre este tema. Por ejemplo, en [11] se utiliza el algoritmo RRT para la planificación de trayectorias dinámicas de un robot manipulador. Por otro lado, para el movimiento de robots móviles omnidireccionales, en [12] se emplea el algoritmo PRM y el diagrama de Voronoi. Además, la planificación de trayectorias es una técnica muy utilizada para vehículos aéreos no tripulados [13], vehículos submarinos autónomos [14] y vehículos de superficie no tripulados [15].

En cuanto a la aplicación de planificadores en laberintos, estos tienen como principal objetivo determinar trayectorias libres de colisiones y con longitud mínima entre dos sitios del laberinto. En este sentido, se tienen dos escenarios principales. El primero considera un robot móvil dentro de un laberinto como se presenta en [16], donde se emplean dos enfoques para determinar una trayectoria libre de colisiones, específicamente se utilizan los algoritmos PRM y genético. Una solución similar, que desarrolla el hardware y el software para navegación y planificación de un robot móvil, se indica en [17]. Por último, basado en la teoría de grafos y comparando tres tipos de algoritmos, en [18] se describe un robot que resuelve laberintos con la ruta más corta.

Por otra parte, el segundo escenario consiste en resolver el sistema bola-laberinto. En tal contexto, en [19] se plantea la resolución de laberintos con canicas mediante planificación reforzada. Por otro lado, en [20] se presenta un sistema de una bola en un plato con un laberinto y realiza la planificación de la trayectoria junto con control difuso.

En este artículo se presenta el análisis de tres algoritmos de planificación de trayectorias para resolver un sistema bola-laberinto en una plataforma de dos grados de libertad. El objetivo del trabajo es evaluar los algoritmos con las métricas de longitud de la trayectoria, tiempo de ejecución y correcta resolución del laberinto. Para el desarrollo se implementó un sistema integral que consta de una plataforma móvil de dos grados de libertad, un laberinto modular y una bola que debe desplazarse desde un punto inicial hasta un punto de destino elegidos por el usuario. Para cumplir tales objetivos, el sistema consta de un subsistema mecánico, un subsistema de adquisición y procesamiento de imágenes, un subsistema de planificación de trayectorias con tres métodos y un subsistema de control.

Las tres técnicas implementadas para la planificación de trayectorias consisten en los algoritmos RRT, PRM y los diagramas de Voronoi junto con el algoritmo A*. La principal contribución de este trabajo radica en la comparación integral (gráfica y analítica) de los tres algoritmos de planificación sobre un sistema físico.

A continuación, primero se presenta la metodología del sistema. Luego se desarrollan los experimentos y un análisis de los resultados generados por los algoritmos. Finalmente, se describen las conclusiones.

METODOLOGÍA

Se presenta una descripción general del sistema y se detalla cada uno de los subsistemas, los cuales son: mecánico, de adquisición y procesamiento de imágenes, de planificación de trayectorias y subsistema de control.

Descripción general del sistema

En la Figura 1 se presenta un diagrama de bloques con la descripción general del sistema. Como se puede observar, el sistema se dividió en 4 subsistemas que son mecánico, de visión, de planificación de trayectorias y de control. Los cuales permiten cumplir el objetivo planteado de resolver el problema del recorrido del laberinto con una bola. Los subsistemas de visión, planificación y control son ejecutados de manera secuencial y con asistencia supervisada, es decir tras la culminación de cada uno un operador indica que el siguiente debe continuar. Los subsistemas de visión, subsistema de planificación de trayectorias y subsistema de control han sido desarrollados mediante Matlab. A continuación, los siguientes apartados describen a detalle la funcionalidad de cada subsistema.

Subsistema mecánico

El desarrollo de pruebas y validación de algoritmos de planificación implementados se llevó a cabo mediante una plataforma modular. La plataforma se muestra en la Figura 2, la misma consta de 2 grados de libertad (GDL), dimensiones generales de 30 x 30 x 50 [*cm*], su estructura (elementos de color azul en la Figura 2) se ensambló mediante piezas de aluminio y elementos de sujeción mecánica. Sobre la estructura ensamblada, se colocaron los diferentes elementos de control y electrónica.

Cabe destacar que el laberinto implementado es de tipo modular, se efectuó el montaje sobre una base metálica con piezas de color negro realizadas en impresión 3D. Las piezas tienen en su base un sistema de anclaje magnético, lo cual permite lograr modularidad y plantear diferentes laberintos.

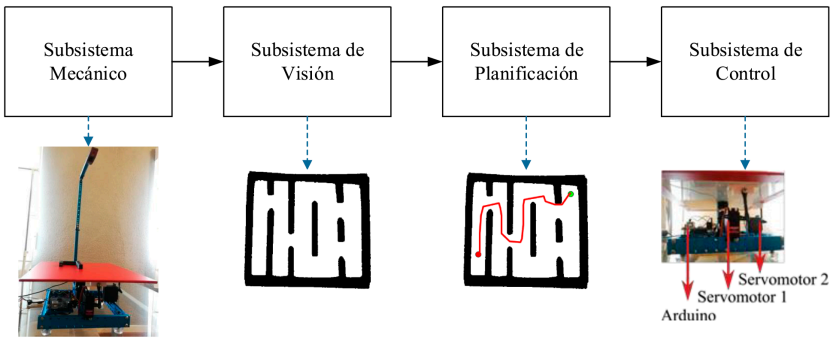

Figura 1. Diagrama de bloques con la descripción general del sistema.

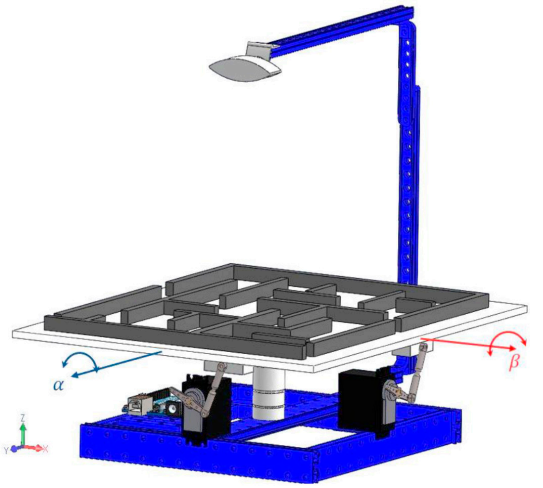

Figura 2. Parámetros y componentes del subsistema mecánico.

Esto tiene como objetivo evaluar la eficacia de los algoritmos ante diferentes escenarios.

El sistema de visión consta de una cámara fija de la marca HP, dispuesta a una distancia de 30 [*cm*] respecto a la base del laberinto, de tal manera que el ángulo de cobertura de la imagen abarque toda la placa.

El movimiento de la placa se ha desarrollado gracias al diseño mecánico implementado, el cual consta de una rótula y dos servomotores. La placa del laberinto está anclada en la parte central a la rótula mientras cada servomotor se colocó de manera perpendicular, a noventa grados como se muestra en la Figura 2. El movimiento de los dos GDL que permitirá inclinar la plataforma para el deslizamiento de la bola, se realiza gracias a los servomotores, de este modo los giros en los ejes (en rojo), (en azul), permiten controlar los ángulos de inclinación β y α respectivamente.

Subsistema de visión

El subsistema de visión es el encargado de generar una imagen binaria del laberinto y las coordenadas iniciales de la bola. Para tal objetivo se instaló una cámara web en una posición fija del sistema mecánico, dicha cámara tiene una resolución de 640x480 píxeles.

El subsistema de visión realiza las etapas que se indican en la Figura 3. En primera instancia, se adquiere la imagen del laberinto a color. Luego, se realiza el procesamiento de la imagen, específicamente se utiliza la técnica de segmentación para pasar la imagen a color a una imagen binaria, la misma tiene color negro en las paredes del laberinto y color blanco en los carriles por donde debe circular la bola. De forma similar, mediante la técnica de segmentación, se obtienen las coordenadas iniciales de la bola, en píxeles. Finalmente, se guarda una matriz de unos y ceros correspondientes al mapa binario del laberinto y las coordenadas de la posición inicial de la bola, las cuales se pasan al subsistema de planificación.

Subsistema de planificación

El objetivo del subsistema de planificación es obtener el mejor camino para resolver el laberinto. Este camino consiste en una ruta libre de colisiones, con la menor longitud y el menor tiempo de ejecución. Para tal objetivo se implementaron tres algoritmos de planificación de trayectorias, los cuales son el algoritmo RRT, PRM y los diagramas de Voronoi junto con el algoritmo A*.

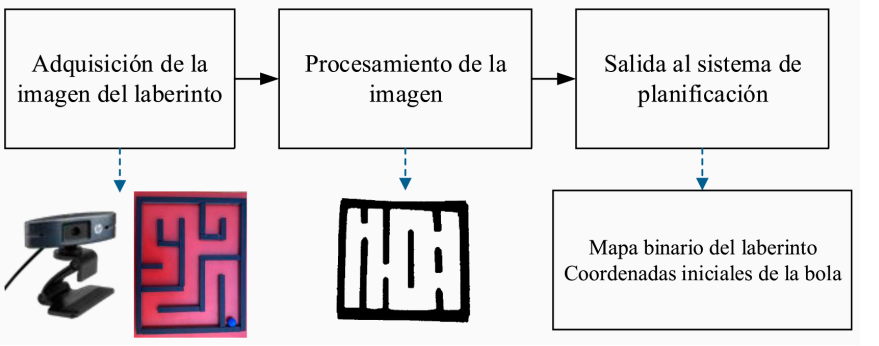

Figura 3. Etapas del subsistema de visión.

Las etapas del programa implementado para este subsistema se indican en la Figura 4. Como se muestra, en primer lugar, el subsistema de visión proporciona la imagen binaria del laberinto y la posición inicial de la bola. A continuación, el usuario debe indicar el punto de destino y el método que desea ejecutar. Cabe mencionar que el punto de destino se selecciona gráficamente en el mapa del laberinto. Finalmente, el programa ejecuta el algoritmo seleccionado y devuelve la trayectoria para la solución del laberinto. Dicha solución consiste en una matriz de coordenadas en píxeles con todos los puntos de la trayectoria, la misma se guarda en un fichero y se pasa al subsistema de control para resolver el laberinto.

En cuanto a los métodos implementados, el primer algoritmo de planificación consiste en el RRT y su diagrama de flujo se detalla en la Figura 5. Como se puede observar, primero se realiza la lectura de parámetros, los cuales son la distancia máxima prefijada entre nodos y el máximo número de iteraciones permitido. Luego, se crea un árbol de exploración rápida de dos componentes. La primera componente comienza en el punto de inicio y la segunda en el punto de destino, de esta forma se obtiene la solución con mayor rapidez.

La siguiente etapa consiste en la generación de un punto aleatorio dentro del mapa y su validación.

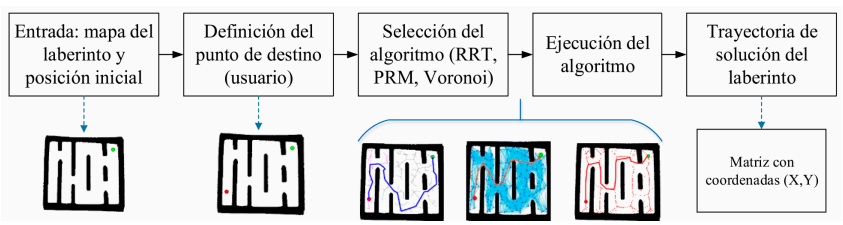

Figura 4. Diagrama de bloques con las etapas del programa de planificación.

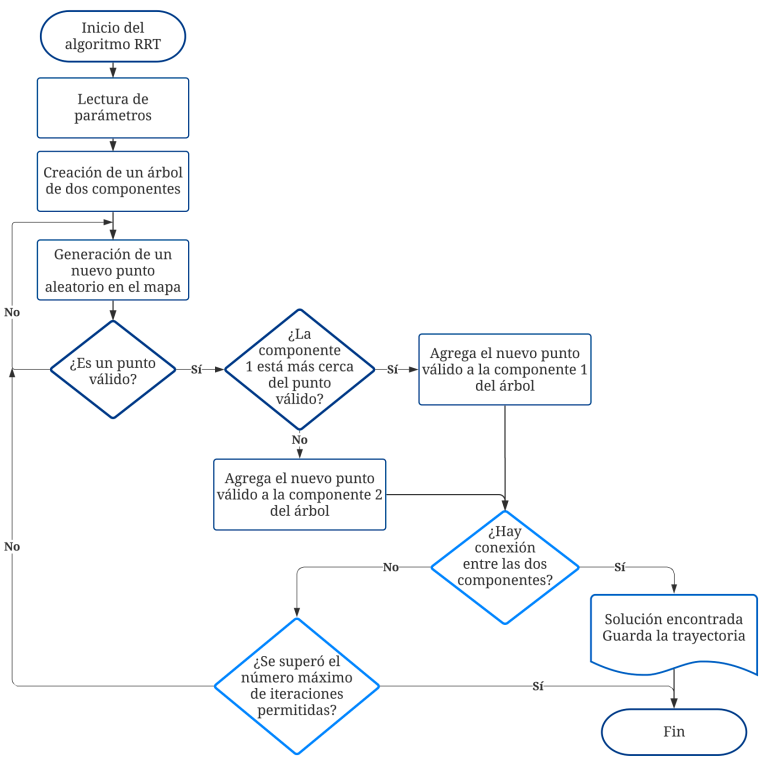

Figura 5. Diagrama de flujo del algoritmo RRT implementado.

Para validar dicho punto se comprueba que sus coordenadas correspondan a un espacio del mapa sin obstáculos y que la distancia con los nodos del árbol sea menor que la distancia máxima permitida. A continuación, solo si el punto es válido se agrega a la componente más cercana del árbol. Finalmente, si existe conexión entre las dos componentes, significa que se encontró la solución y se guarda la trayectoria con los puntos de las dos componentes. Caso contrario, se genera un nuevo punto aleatorio y se repite el procedimiento. No obstante, si se supera el número máximo de iteraciones termina el programa sin solución.

Con respecto a la segunda técnica, se empleó el algoritmo PRM. En este sentido, se utilizó la librería de Matlab [21] que recibe como parámetros el mapa binario del laberinto, los puntos de inicio y destino, el número máximo de nodos prefijado y la máxima distancia entre nodos permitida. Como resultado, la librería devuelve el camino entre los dos puntos objetivo.

Finalmente, se implementó una tercera técnica de planificación de trayectorias. Con el objetivo de obtener un camino equidistante entre las paredes del laberinto, se emplearon los diagramas de Voronoi y para determinar la ruta más corta se utilizó el algoritmo de búsqueda A*. El diagrama de flujo de esta técnica se presenta en la Figura 6. Como se puede observar, primero se leen los parámetros que indican la distancia mínima y máxima permitida entre nodos. A continuación, se calculan los vértices del diagrama de Voronoi mediante la función disponible en Matlab. Luego, se validan dichos vértices solo si se encuentran dentro del mapa y si sus coordenadas están fuera de las paredes del laberinto.

Una vez obtenidos los vértices, se ejecuta el algoritmo de búsqueda A* para obtener el camino más corto entre los puntos de inicio y de destino. Posteriormente, se realiza un filtrado de los nodos en exceso de la trayectoria en función de la distancia mínima y máxima permitidas entre nodos. Cabe mencionar que, por experimentos empíricos, se determinó que el controlador tiene un mejor rendimiento cuando la trayectoria dispone de la menor cantidad de nodos y la distancia entre dos nodos consecutivos no es demasiado pequeña (v.g. mínimo de 20 píxeles). En este sentido, los diagramas de Voronoi generan un número determinado de nodos y usualmente

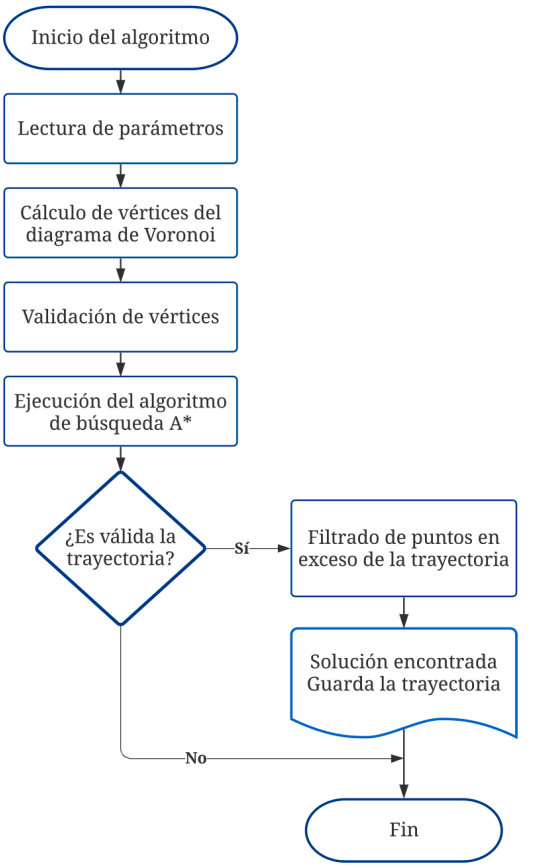

Figura 6. Diagrama de flujo de la técnica implementada con diagramas de Voronoi y el algoritmo A*.

con una separación inferior a la distancia mínima deseada. Por tales motivos se realizó este filtrado, específicamente, si la longitud entre dos nodos es menor que la distancia mínima prefijada se elimina uno de ellos, siempre y cuando la longitud con el siguiente nodo no sea mayor que la distancia máxima permitida. Finalmente, el camino generado consiste en la solución del laberinto y se almacena para enviar al subsistema de control.

Subsistema de control

Para el último subsistema que resuelve físicamente el laberinto, se implementaron dos controladores PID (Proporcional, Integral y Derivativo) uno para la posición en x, y otro para la posición en y. En la Figura 7 se presenta el diagrama de bloques del controlador implementado. Como se muestra, en primer lugar, se recibe el camino del subsistema

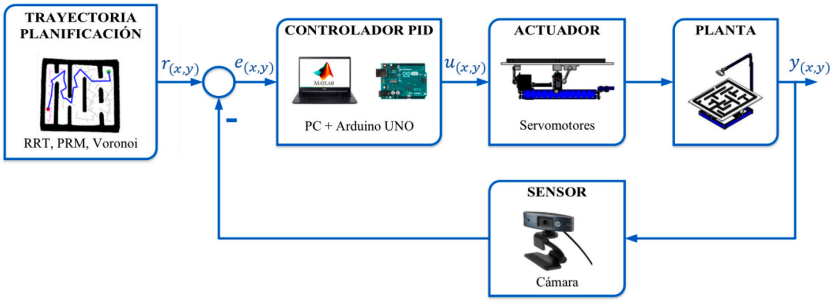

Figura 7. Diagrama de bloques del controlador del sistema.

de planificación, este es un fichero que contiene las posiciones de paso $r_{(x,\ y)}$. Luego, utilizando la cámara se obtiene la posición $y_{(x,\ y)}$ de la bola en el laberinto en tiempo real. Posteriormente se calcula el error de posición $e_{(x,\ y)}$ y se realiza el control PID del sistema. La salida del controlador $u_{(x,\ y)}$ va a una tarjeta Arduino que se encarga de mover los servomotores, específicamente con una señal PWM a cada servomotor. Una vez alcanzada la posición deseada, avanza a la siguiente posición que tiene el fichero con los puntos de la trayectoria y se repite el procedimiento hasta finalizar el camino.

EVALUACIÓN EXPERIMENTAL Y RESULTADOS

Para evaluar los algoritmos de planificación implementados se consideraron las métricas de longitud de la trayectoria y tiempo de ejecución. Adicionalmente, se emplearon cuatro laberintos diferentes, los cuales se indican en la Figura 8. Como se puede observar, en el caso del laberinto 1 se tiene un solo camino para llegar desde el punto de inicio al de destino, mientras que en los demás laberintos se tienen al menos dos caminos. Esto permite obtener diferentes soluciones para un mismo laberinto. Por último, con el objetivo de efectuar un análisis estadístico de los resultados, se realizaron 20 ejecuciones de cada algoritmo para cada laberinto y con distintos valores de los parámetros.

Los parámetros de configuración para los tres métodos se detallan en la Tabla 1, inicialmente se efectuaron múltiples experimentos con diferentes valores y distintos algoritmos, de esta manera se definieron los parámetros de forma empírica. A partir de estas pruebas, se eligieron diversos valores de la distancia máxima permitida entre los nodos

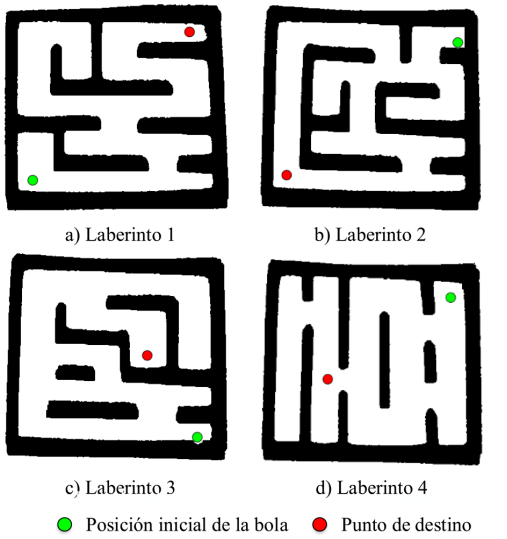

Figura 8. Laberintos empleados en los experimentos.

(también denominada δ) para obtener los resultados. En particular, se escogieron los valores desde 50 hasta 200 píxeles en intervalos de 25 píxeles. En tal sentido, si los algoritmos emplean valores de δ superiores a 200 píxeles, se comprobó que la trayectoria es demasiado brusca y se complica el control del sistema. Por otra parte, con una distancia δ inferior a 50 píxeles, usualmente los métodos no encontraban la solución.

Continuando con la configuración de parámetros, el algoritmo RRT requiere otro parámetro que es el número máximo de iteraciones. En tal contexto, se definió un valor de 2000 ya que con más iteraciones aumenta el tiempo de ejecución y con un valor menor disminuye la efectividad para encontrar soluciones. Por otro lado, el algoritmo PRM requiere un parámetro adicional que es el número máximo

Tabla 1. Parámetros configurados en los algoritmos de planificación.

Algoritmo RRT	
Distancia máxima permitida entre nodos o δ (px)	50 a 200
Número máximo de iteraciones	2000
Algoritmo PRM	
Distancia máxima permitida entre nodos o δ (px)	50 a 200
Número máximo de nodos	500
Diagrama de Voronoi y algoritmo A*	
Distancia máxima permitida entre nodos o δ (px)	50 a 200
Distancia mínima permitida entre nodos (px)	20

de nodos, en este sentido se eligió el valor de 500. Después de varios experimentos, se determinó que en algunas ejecuciones no se encuentra solución si el número máximo de nodos es menor.

Por último, en el diagrama de Voronoi junto con el algoritmo A* se realizó la configuración de dos parámetros, los cuales son la distancia mínima y máxima permitidas entre nodos. Cabe recalcar que el diagrama de Voronoi por si solo no requiere parámetros de distancia y el número de nodos es fijo según la topología del laberinto. Sin embargo, en el método implementado se tiene una etapa de filtrado de nodos después del algoritmo A*, la misma necesita estos dos parámetros, tal como se indicó en subsistema de planificación. Dicho filtrado tiene como objetivo disminuir el número de nodos, lo cual mejora la etapa de control.

Evaluación gráfica

Con respecto al análisis gráfico de los resultados obtenidos con los algoritmos de planificación, por un lado, se presentan las soluciones de los tres métodos para los dos primeros laberintos con una configuración de la distancia máxima permitida entre nodos de 100 píxeles. Dichas soluciones se muestran en la Figura 9. A partir de las gráficas se infiere que el laberinto 1 siempre tiene el mismo camino como solución, mientras que para el laberinto 2 los algoritmos RRT y PRM obtienen dos soluciones distintas. Cabe mencionar que el método del diagrama de Voronoi con el algoritmo A* siempre obtiene el mismo camino, esto es por el modelo determinístico que emplean los diagramas de Voronoi.

Además, las gráficas muestran que el algoritmo RRT genera trayectorias con cambios bruscos. Por otra parte, el algoritmo PRM presenta trayectorias muy cercanas a las paredes del laberinto. En consecuencia, con estos dos algoritmos se complica el control del sistema. Por último, la técnica que emplea los diagramas de Voronoi y el algoritmo A* proporciona trayectorias más suaves y usualmente distantes de las paredes de los laberintos, lo cual facilita el control.

Por otro lado, en la Figura 10 se muestran capturas de varias soluciones de los tres algoritmos de planificación para el laberinto 4 y con diferentes

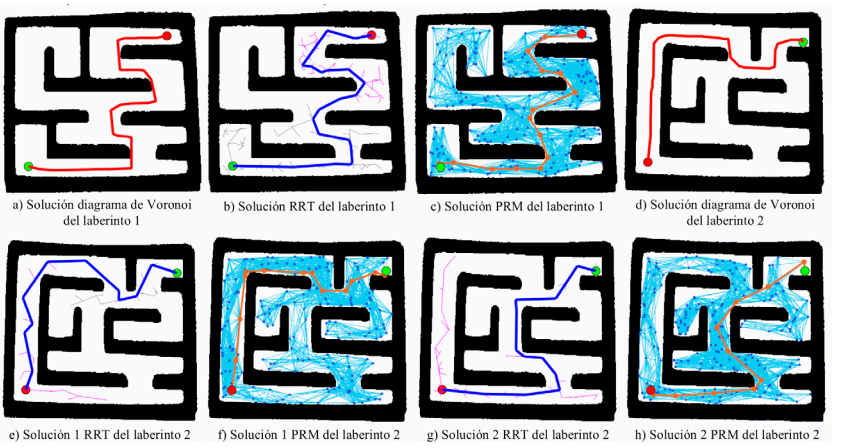

Figura 9. Soluciones de los métodos de planificación de trayectorias para los laberintos 1 y 2.

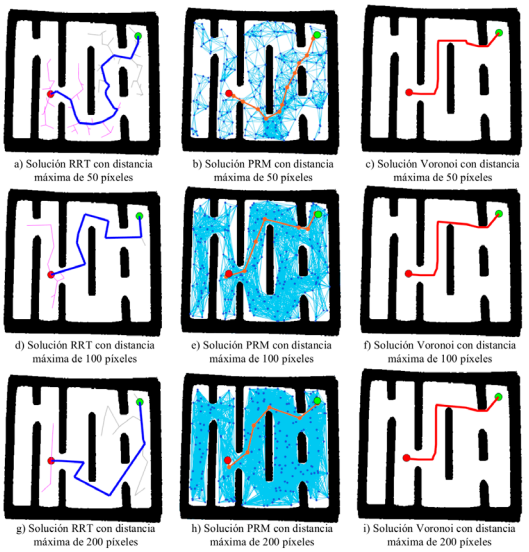

Figura 10. Soluciones de los métodos de planificación de trayectorias para el laberinto 4 con distintas configuraciones de la distancia máxima permitida entre nodos.

configuraciones de la distancia máxima permitida entre nodos. Específicamente, se presentan los resultados con los valores de δ de 50, 100 y 200 píxeles. Como se puede observar, en el algoritmo RRT a mayor δ disminuye el número de nodos y por lo tanto disminuye el tiempo de ejecución. En cuanto al algoritmo PRM, se concluye que a mayor δ aumenta el número de enlaces entre los nodos, lo cual implica un mayor tiempo de ejecución.

Finalmente, en el caso del método con el diagrama de Voronoi y el algoritmo A*, la solución cuando el parámetro δ es de 50 píxeles difiere ligeramente de las soluciones cuando δ tiene un valor de 100 y 200 píxeles (en estos dos casos se tiene la misma solución). Esto es consecuencia de la definición del diagrama de Voronoi que siempre genera el mismo resultado. Cabe mencionar que está pequeña variación en los resultados ocurre por la etapa de filtrado que se realiza al final del método, la misma logra que disminuya el número de nodos de la trayectoria. En particular para estos resultados, sin efectuar el filtrado se tendría un total de 43 nodos, mientras que, cuando δ es de 50 píxeles se obtuvo un total de 25 nodos y para una distancia máxima permitida de 100 y 200 píxeles el número de nodos en la trayectoria resultante es de 24.

Evaluación analítica

Continuando con los experimentos, se efectuó una evaluación analítica para cada algoritmo y cada laberinto con distintas configuraciones de la distancia máxima permitida entre nodos. Específicamente, se realizaron 20 ejecuciones de cada método para cada configuración y luego se desarrolló un análisis estadístico de los resultados. Con respecto a las métricas de evaluación, en cada instancia se obtuvo la distancia de la trayectoria en píxeles y el tiempo de ejecución en segundos. A continuación, para los resultados finales se calculó el tiempo de ejecución medio, la longitud media y los intervalos de confianza del 95% tanto del tiempo como de la distancia.

En la Tabla 2 se detallan los resultados obtenidos de la longitud media de la trayectoria y en la Tabla 3 se muestran los resultados de tiempo de ejecución medio para los tres algoritmos en todos los laberintos y con distintas configuraciones del parámetro δ. En cuanto al tiempo de ejecución de la tercera técnica, el mismo se calculó como la suma del tiempo de generación de los diagramas de Voronoi más la ejecución del algoritmo A*.

Siguiendo con el análisis de resultados, se graficaron los valores obtenidos de la longitud media de las trayectorias y del tiempo medio de ejecución con los intervalos de confianza del 95%. Estos resultados corresponden a los tres algoritmos y para cada laberinto. Para todas las gráficas el eje *x* corresponde a las configuraciones de distancia máxima permitida entre nodos que va desde 50 a 200 píxeles en intervalos de 25 píxeles. En la Figura 11 se presentan las gráficas de la longitud media de la trayectoria.

A partir de estos resultados se concluye que el algoritmo RRT siempre generó la trayectoria de mayor longitud y el algoritmo PRM la trayectoria de menor distancia para todos los laberintos. En particular, los caminos generados por el algoritmo PRM usualmente pasan muy cerca de las aristas del laberinto, lo cual contribuye para que genere la menor longitud de la trayectoria. Además, si se analizan los intervalos de confianza del 95%, se aprecia que los algoritmos RRT y PRM presentan mayor variación de sus resultados, esto se debe a que son dos métodos que generan sus nodos de forma aleatoria. Por otra parte, la técnica con diagramas de Voronoi muestra dispersión cero en sus resultados de distancia. Es decir, siempre se obtiene la misma

Tabla 2. Resultados de distancia media de la trayectoria.

Algoritmo	Distancia máxima permitida entre nodos (píxeles)						
	50	75	100	125	150	175	200
Distancia media de las trayectorias para el laberinto 1 (píxeles)							
RRT	–	922,80	951,78	936,22	927,55	955,27	1004,03
PRM	785,88	756,52	747,47	758,29	753,71	749,31	751,62
Voronoi - A*	790,02	780,82	780,82	780,82	780,82	780,82	780,82
Distancia media de las trayectorias para el laberinto 2 (píxeles)							
RRT	750,25	742,99	763,41	748,73	767,32	768,98	790,63
PRM	685,53	643,54	627,33	636,60	636,55	630,04	635,96
Voronoi - A*	679,10	678,92	678,92	678,92	678,92	678,92	678,92
Distancia media de las trayectorias para el laberinto 3 (píxeles)							
RRT	797,52	890,52	898,73	901,62	899,79	898,60	927,06
PRM	730,92	690,54	685,40	676,13	677,01	684,16	681,46
Voronoi - A*	800,24	800,24	800,24	800,24	800,24	800,24	800,24
Distancia media de las trayectorias para el laberinto 4 (píxeles)							
RRT	513,18	514,04	509,36	518,24	514,09	550,40	521,10
PRM	407,95	372,26	375,84	376,71	373,16	369,06	376,57
Voronoi - A*	387,80	387,80	382,84	382,84	382,84	382,84	382,84

Tabla 3. Resultados de tiempo de ejecución medio.

Algoritmo	Distancia máxima permitida entre nodos (píxeles)						
	50	75	100	125	150	175	200
Tiempo medio de ejecución para el laberinto 1 (segundos)							
RRT	–	2,14	1,92	1,85	2,45	2,47	1,64
PRM	1,20	1,13	1,78	2,56	3,15	4,34	3,78
Voronoi - A*	1,83	1,19	1,28	1,49	1,59	1,72	1,44
Tiempo medio de ejecución para el laberinto 2 (segundos)							
RRT	2,19	1,61	1,15	1,08	1,03	1,07	1,09
PRM	0,63	1,19	1,64	2,26	2,85	3,85	5,43
Voronoi - A*	1,11	1,38	1,49	1,45	1,49	1,51	1,61
Tiempo medio de ejecución para el laberinto 3 (segundos)							
RRT	2,80	1,81	1,47	1,12	0,92	0,99	0,72
PRM	0,80	1,17	1,71	2,32	2,93	4,61	4,53
Voronoi - A*	1,57	1,46	1,63	1,57	1,70	1,80	1,71
Tiempo medio de ejecución laberinto 4 (segundos)							
RRT	2,30	1,04	0,50	0,42	0,42	0,45	0,41
PRM	0,93	1,29	1,56	2,11	2,72	3,32	3,98
Voronoi - A*	1,52	1,41	1,26	1,25	1,22	1,33	1,27

longitud de la trayectoria si se ejecuta está técnica en un mismo laberinto y con parámetros iguales. Lo cual es consecuencia de que estos diagramas corresponden a un modelo determinístico.

En cuanto a los resultados de tiempo de ejecución, los mismos se grafican en la Figura 12. Como se puede observar, el algoritmo RRT presenta un menor tiempo de ejecución en la mayoría de experimentos mientras

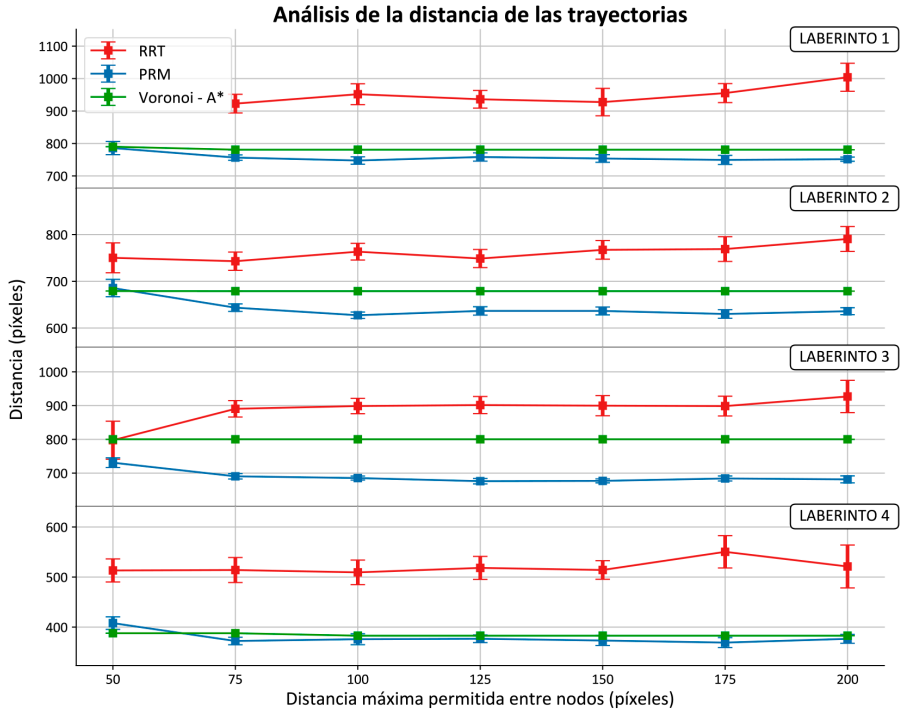

Figura 11. Análisis de la distancia de las trayectorias generadas por los algoritmos de planificación.

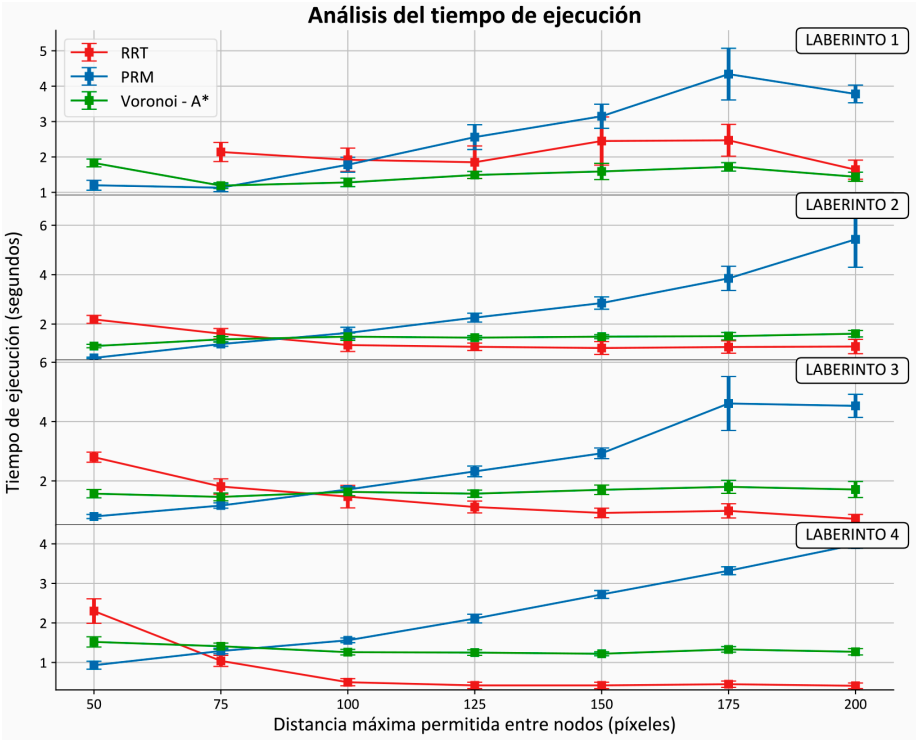

Figura 12. Análisis del tiempo de ejecución de los algoritmos de planificación.

que el algoritmo PRM es el de peor rendimiento con respecto a esta métrica. Además, la técnica que emplea los diagramas de Voronoi presenta un comportamiento casi uniforme en cuanto a los tiempos de ejecución.

Adicionalmente, se observa que el algoritmo RRT presenta un mal rendimiento cuando el parámetro δ es de 50 píxeles. Lo cual sucede porque con esta configuración dicho método requiere demasiados nodos para encontrar una solución, en consecuencia, aumenta el número de iteraciones y el tiempo de ejecución. En el caso particular del laberinto 1, este algoritmo no encontró una solución dado que la distancia entre el punto de inicio y el de destino es elevada junto con la separación entre las paredes que es reducida.

A partir de estos resultados se infiere que al modificar el parámetro δ se afecta principalmente el tiempo de ejecución de cada algoritmo. Por ejemplo, en el algoritmo RRT el tiempo de ejecución disminuye a medida que aumenta el parámetro δ. En este sentido, al aumentar la distancia permitida entre nodos se requieren menos nodos para encontrar la solución, por lo tanto, menos iteraciones y disminuye el tiempo de ejecución. Por otra parte, en el algoritmo PRM sucede lo contrario y el tiempo de ejecución aumenta a medida que aumenta δ. En tal contexto, al aumentar la distancia permitida entre nodos también aumenta el número de conexiones entre todos los nodos, lo cual incrementa el tiempo de ejecución. Por último, si se modifica el parámetro δ no influye en los resultados de la técnica que emplea los diagramas de Voronoi y el algoritmo A*. Esto se debe al modelo determinístico de los diagramas de Voronoi.

Finalmente, en base a los resultados obtenidos, se puede concluir que para este tipo de sistemas el algoritmo PRM obtiene la trayectoria con menor longitud, el algoritmo RRT usualmente tiene el menor tiempo de ejecución y la técnica implementada con diagramas de Voronoi y el algoritmo de búsqueda A* genera la trayectoria más suave y equidistante a las paredes del laberinto. Por tales motivos, en función de los requerimientos del sistema se debe elegir uno de los tres métodos.

CONCLUSIONES

En este artículo se desarrolló un análisis completo de los algoritmos de planificación de trayectorias RRT, PRM y diagramas de Voronoi con el algoritmo A*. Para este análisis se empleó un sistema bola-laberinto con una plataforma de dos grados de libertad. En este sentido, el sistema consiste de cuatro subsistemas denominados mecánico, de visión, de planificación y de control. En particular, el diseño del sistema se basa en un laberinto modular que permite al usuario crear su propio laberinto.

En cuanto a los experimentos, se desarrolló un análisis integral gráfico y analítico de las soluciones de los tres algoritmos para cuatro laberintos diferentes. Además, se eligieron las métricas de longitud media de la trayectoria y tiempo medio de ejecución para evaluar el rendimiento de los métodos. Adicionalmente, se modificó el parámetro de la distancia máxima permitida entre nodos y se realizaron 20 repeticiones de cada experimento para efectuar un análisis estadístico.

Con respecto a los resultados obtenidos en los experimentos, se concluye que el algoritmo RRT presenta la mayor variación en sus datos, la trayectoria de mayor longitud para todas las configuraciones del laberinto y el mejor rendimiento en cuanto al tiempo de ejecución. Por otra parte, el algoritmo PRM genera la trayectoria de menor longitud, no obstante, tiene un mal rendimiento con respecto al tiempo medio de ejecución. Finalmente, la técnica con diagramas de Voronoi presenta la menor variación en sus datos, la trayectoria más suave y equidistante entre las paredes del laberinto.

TRABAJOS FUTUROS

Como trabajo futuro se plantea implementar la opción de replanificación de trayectorias. Lo cual tiene como objetivo agregar obstáculos en el laberinto, durante la ejecución en tiempo real de la trayectoria, y que el sistema posea la capacidad de realizar automáticamente otra planificación y encontrar una nueva solución.

AGRADECIMIENTOS

Los autores agradecen a la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) del Ecuador por el apoyo brindado para llevar a cabo esta investigación.

REFERENCIAS

[1] B.K. Patle, G. Babu, L.A. Pandey, D.R.K. Parhi and A. Jagadeesh. "A review: On path planning strategies for navigation of mobile robot". Def. Technol. Vol. 15 Nº 4, pp. 582-606. 2019. ISSN: 2214-9147. DOI: 10.1016/j.dt.2019.04.011.

[2] M. Candeloro, A.M. Lekkas and A.J. Sørensen. "A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels". Control Eng. Pract. Vol. 61 Nº January, pp. 41-54. 2017. ISSN: 0967-0661. DOI: 10.1016/j.conengprac.2017.01.007.

[3] I. Aguinaga, D. Borro and L. Matey. "Parallel RRT-based path planning for selective disassembly planning". Int. J. Adv. Manuf. Technol. Vol. 36 Nº 11-12, pp. 1221-123. 2008. ISSN: 1433-3015. DOI: 10.1007/s00170-007-0930-2.

[4] B. Li, P. Feng, L. Zeng, C. Xu and J. Zhang. "Path planning method for on-machine inspection of aerospace structures based on adjacent feature graph". Robot. Comput. Integr. Manuf. Vol. 54 Nº April, pp. 17-34. 2018. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2018.05.006.

[5] S.M. LaValle. "Planning algorithms". Cambridge University Press. New York, USA. ISBN: 9780511241338. 2006.

[6] I. Cruz, P. Lara, M. Gutiérrez, S. De los Cobos, E. Rincón and R. Mora. "A Stochastic Algorithm for Solving Mazes". Rev. Matemática Teoría y Apl. Vol. 26 Nº 2, pp. 319-337. 2019. ISSN: 1409-2433. DOI: 10.15517/rmta.v26i2.38322.

[7] A.K. Sadhu, S. Shukla, T. Bera and R. Dasgupta. "Safe and fast path planning in cluttered environment using contiguous free-space partitioning". International Conference on Robotics and Automation (ICRA). Montreal, Canada. May 20-24, 2019.

[8] D. Connell and H.M. La. "Dynamic path planning and replanning for mobile robots using RRT". IEEE International Conference on Systems, Man and Cybernetics (SMC). Banff, Canada. October 5-8, 2017.

[9] K. Cao, Q. Cheng, S. Gao, Y. Chen and C. Chen. "Improved PRM for Path Planning in Narrow Passages". IEEE International Conference on Mechatronics and Automation (ICMA). Tianjin, China. August 4-7, 2019.

[10] P. Bhattacharya and M.L. Gavrilova. "Roadmap-based path planning - Using the voronoi diagram for a clearance-based shortest path". IEEE Robot. Autom. Mag. Vol. 15 Nº 2, pp. 58-66. 2008. ISSN: 1558-223X. DOI: 10.1109/MRA.2008.921540.

[11] K. Wei and B. Ren. "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm". Sensors (Switzerland). Vol. 18 Nº 2. 2018. ISSN: 1424-8220. DOI: 10.3390/s18020571.

[12] M.R.H. Al-dahhan and K.W. Schmidt. "Path Planning Based on Voronoi Diagram and PRM for Omnidirectional Mobile Robots". DTSS2019 international conference. Ankara, Turkey. October 23-25, 2019.

[13] Y. Lin and S. Saripalli. "Sampling-Based Path Planning for UAV Collision Avoidance". IEEE Trans. Intell. Transp. Syst. Vol. 18 Nº 11, pp. 3179-3192. 2017. ISSN: 1558-0016. DOI: 10.1109/TITS.2017. 2673778.

[14] Z. Zeng, K. Sammut, A. Lammas, F. He and Y. Tang. "Efficient Path Re-planning for AUVs Operating in Spatiotemporal Currents". J. Intell. Robot. Syst. Theory Appl. Vol. 79 Nº 1, pp. 135-153. 2015. ISSN: 1573-0409. DOI: 10.1007/s10846-014-0104-z.

[15] R. Song, Y. Liu and R. Bucknall. "Smoothed A* algorithm for practical unmanned surface vehicle path planning". Appl. Ocean Res. Vol. 83, pp. 9-20. 2019. ISSN: 0141-1187. DOI: 10.1016/j.apor.2018.12.001.

[16] R.M.C. Santiago, A.L. De Ocampo, A.T. Ubando, A.A. Bandala and E.P. Dadios. "Path planning for mobile robots using genetic algorithm and probabilistic roadmap". HNICEM 2017 - 9th Int. Conf. Humanoid, Nanotechnology, Inf. Technol. Commun. Control. Environ. Manila, Philippines. December 1-3, 2017.

[17] S. Jose and A. Antony. "Mobile robot remote path planning and motion control in a maze environment". IEEE Int. Conf. Eng. Technol. ICETECH. Coimbatore, India. March 17-18, 2016.

[18] M.O.A. Aqel, A. Issa, M. Khdair, M. Elhabbash, M. Abubaker and M. Massoud.

"Intelligent maze solving robot based on image processing and graph theory algorithms". Int. Conf. Promis. Electron. Technol. ICPET. Deir El-Balah, Palestine. October 16-17, 2017.

[19] M. Zucker and J.A. Bagnell. "Reinforcement planning: RL for optimal planners". IEEE Int. Conf. Robot. Autom. Saint Paul, USA. May 14-18, 2012.

[20] C.C. Cheng and C.C. Chou. "Fuzzy-based visual servo with path planning for a ball-plate system". International Symposium on Intelligent Computing Systems. Mérida, Mexico. March 16-18, 2016.

[21] I. The MathWorks. "Probabilistic Roadmaps (PRM) - MATLAB & Simulink". Date of visit: March 4, 2021. URL: https://es.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html