ISSN 0121–1129

Evaluación de las técnicas de planificación de movimientos, Descomposición exacta trapezoidal y Descomposición adaptativa de celdas a través de mallas

Techniques' evaluation for motion planning, using exact and adaptive cells decomposition

Fecha de recepción: 2 de mayo de 2012
Fecha de aprobación: 29 de junio de 2012

Carlos Andrés Vélez Carvajal*,
Jaime Alberto Guzmán Luna**,
Ingrid Durley Torres Pardo***

Resumen

Compara el desempeño de dos técnicas de planificación de movimientos: la *descomposición exacta trapezoidal* y la *descomposición adaptativa de celdas a través de mallas*. Ambas técnicas se implementaron en robots móviles Lego Mindstorms, bajo el ambiente de desarrollo Java-LeJOS. Su evaluación se realizó en tres ambientes de prueba, asumiendo la geometría del robot de dos maneras: como un punto y como un círculo en el espacio. Esta evaluación permitió ver que la *descomposición exacta* presenta trayectorias más cortas que la *adaptativa;* que su desempeño es similar en cuanto al error generado en la localización real final del robot; y que ambas técnicas permiten hallar caminos con

Abstract

A comparison of two motion planning techniques: the trapezoidal exact cell decomposition and the adaptive cell decomposition through grids. Both techniques were implemented in two Lego Mindstorms mobile robots, by using the Java-LeJOS development environment. Their evaluation was conducted in three test environments, assuming the robot's geometry in two ways: As a point and a circle in space. The evaluation results showed that the exact decomposition gives a shorter path than the adaptive one, its performance in terms of error generated in the robot's final real location, is similar in both, and these techniques allow finding the shortest paths, when considering the robot's geometry as a circle,

* Estudiante Ingeniería de Sistemas, Universidad Nacional de Colombia, sede Medellín. Correo electrónico: caavelezca@unal.edu.co
** Ingeniero Civil, Universidad Nacional de Colombia, sede Medellín. Especialista en Comunicación Educativa, Universidad de Pamplona. Magíster en Ingeniería de Sistemas, Universidad Nacional de Colombia, sede Medellín. Doctor en Ingeniería: Énfasis en Sistemas,Universidad Nacional de Colombia, sede Medellín. Profesor, Escuela de Ingeniería de Sistemas, Universidad Nacional de Colombia, sede Medellín. Correo electrónico: jaguzman@unal.edu.co
*** Ingeniera de Sistemas, Universidad Distrital Francisco José de Caldas. Magíster en Ingeniería de Sistemas,Universidad Nacional de Colombia, sede Medellín. Docente investigador, Institución Universitaria Salazar y Herrera.
Correo electrónico: itorrespa@iush.edu.co

menor distancia cuando se considera la geometría del robot como un círculo, que cuando se considera como un punto, ya que se delimitan opciones de recorridos más cercanos a la realidad.

Palabras clave: planificación de movimientos, navegación de robots, robótica móvil, robot Lego Mindstorms, programación en Java-LeJOS.

instead of a dot, because the routes' option, are designed closer to reality.

Key Words: Motion Planning, Robot Navigation, Mobile Robotics, Lego Mindstorms Robots, Java-LeJOS Programming.

I. Introducción

El presente artículo pretende comparar el desempeño de dos técnicas usadas en la navegación de robots móviles a través de la planificación de movimientos; tales técnicas son: la descomposición exacta y la descomposición adaptativa de celdas, que fueron implementadas para la navegación de un robot tipo diferencial Lego Mindstorms haciendo uso del ambiente Java-LeJOS.

Para ello, en la sección II se define qué es la planificación en general y la planificación de movimientos; en la sección III se explica la arquitectura del modelo seguido para la implementación de dichas técnicas; en la sección IV se especifican las características del robot construido para la realización de las pruebas de las dos técnicas de planificación utilizadas, detallando su configuración geométrica, la cual se simplificó asumiéndolo como un robot circular; en la sección V se presentan los conceptos fundamentales de las técnicas que se emplearon, y se realiza una explicación general de cómo se desarrolla su implementación bajo el ambiente de desarrollo Java-LeJOS [1] para robots Lego Mindstorms [2]; en la sección VI se describe cómo se utilizó un compás magnético para corregir la trayectoria seguida por el robot en el mundo real, luego de aplicar los resultados obtenidos con las técnicas de planificación de movimientos que se usaron; en la sección VII se presentan los resultados obtenidos en tres escenarios distintos, midiendo el desempeño de ambas técnicas en cuanto a la distancia, su error en la localización final y el tipo de geometría utilizada, y, finalmente, la sección VIII presenta las conclusiones, una mirada al trabajo futuro y un análisis comparativo, basado en los resultados, del desempeño de las implementaciones realizadas.

II. La planificación para la navegación de robots

La planificación es un proceso abstracto y deliberativo que escoge y organiza acciones anticipando sus resultados esperados [3]. Al deliberar se busca alcanzar, tan bien como sea posible, los objetivos preestablecidos. Este proceso es tratado bajo la óptica de la inteligencia artificial a través de su área conocida como planificación automática. Uno de los campos de aplicación de esta área es la planificación de movimientos [4, 5, 6], la cual propone algoritmos orientados a la solución de problemas en los que, dada una posición inicial, un sistema móvil logre alcanzar una posición y estado final establecidos.

Dentro de las muchas técnicas de planificación de movimientos, se distinguen las técnicas por descomposición de celdas. Estas buscan descomponer el espacio en celdas identificando las celdas libres de obstáculos, con el fin de obtener un grafo de adyacencia en el cual buscar rutas existentes desde una posición inicial hasta una posición final [5].

Dos de las principales técnicas usadas son: la descomposición exacta de celdas y la descomposición adaptativa de celdas, como se explican más adelante en la sección 4. Estas técnicas son tratadasen diferentes trabajos [4, 5, 6], donde se explica su algoritmo básico y se exponen algunos ejemplos de su aplicación, pero en estos no se muestran resultados de su implementación ni mucho menos una evaluación práctica de su desempeño. En otros estudios realizados [14] se usan estas técnicas considerando la geometría del robot como un punto, sin considerar mecanismos que corrijan los errores de localización producidos durante el movimiento del robot. A pesar de su ilustración práctica, en este trabajo no se realizan pruebas que permitan caracterizar y comparar de manera concreta el desempeño de la una frente a la otra.

Dada la ausencia en la literatura sobre trabajos que permitan evaluar el desempeño de ambas técnicas de planificación de movimientos, este estudio se propone realizar un conjunto de experimentos que permitan su comparación desde un enfoque práctico.

III. Arquitectura del modelo propuesto

La arquitectura propuesta para la implementación de los algoritmos de planificación de movimientos consta de 5 componentes: (1) La Interfaz Gráfica de Usuario

(IGU) que permite la configuración del problema de planificación de movimientos por parte del usuario, (2) el módulo que permite incluir la geometría del robot en el cálculo de la solución del problema de planificación, (3) el planificador que permite realizar el cálculo del plan que debe seguir el robot para solucionar el problema, (4) el traductor que permite expresar el plan generado por el planificador en comandos propios del robot para la ejecución del plan, y por último, (5) el ejecutor del plan en el robot, que hace uso del plan traducido a comandos de operación del robot desde el módulo traductor, con el fin de llevar a cabo su ejecución y monitoreo para realizar las respectivas correcciones, si es necesario. En la Figura 1 se muestra la disposición por capas de esta arquitectura. A continuación, se tratan los principales detalles de cada módulo.

En el módulo relacionado con la interfaz gráfica, el usuario ingresa: (1)los puntos inicial y final, (2) los puntos que determinan los vértices de los obstáculos, (3) los puntos que definen el alto y ancho del escenario y del robot (de este último, estos puntos definen inicialmente un paralelogramo en el que queda totalmente inscrita la geometría real del robot). Los puntos se obtienen indicando las coordenadas *(x, y)* que arrojan una representación en un plano cartesiano que el programa entiende en pixeles, pero en la ejecución se establece una relación con las unidades de medida (por defecto en centímetros).

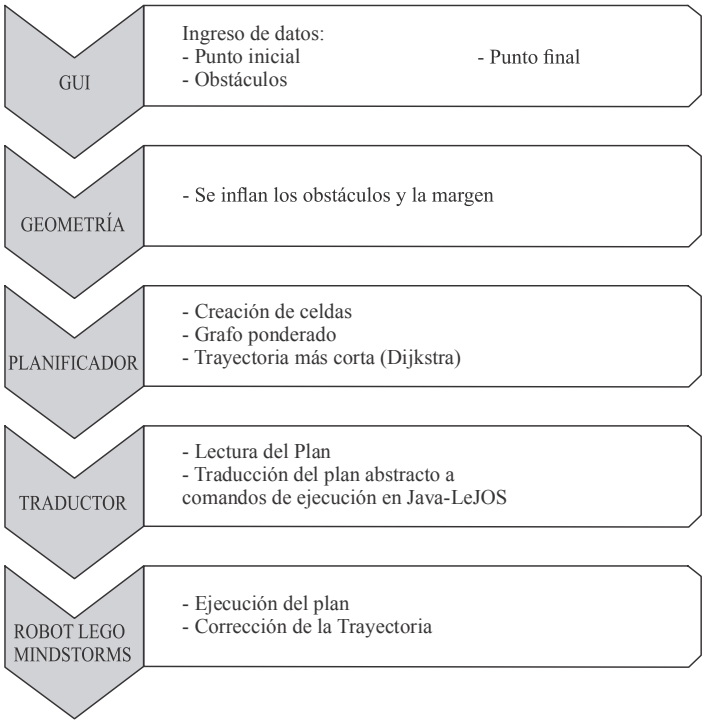

Figura 1. Arquitectura propuesta para la implementación de las técnicas de planificación de movimientos

En el módulo asociado a la caracterización de la geometría del robot en el marco del problema de planificación de movimientos, se extienden los obstáculos y el margen de acción del robot de manera proporcional al radio del robot, buscando con esto no representar el robot como un punto, sino como un objeto espacial. Los detalles de este módulo se tratarán en la sección V.

El módulo que contempla el planificador permite descomponer el espacio por donde se moverá el robot, siguiendo los criterios estipulados en cada técnica, con el fin de formar un grafo donde un nodo es una celda, ya sea trapezoidal o en malla, y las aristas representan la conexión de celdas adyacentes. Una vez se consigue el grafo ponderado, se busca hallar, en caso de existir, la trayectoria más corta haciendo uso del algoritmo de Dijkstra [7, 8]. Este módulo permite generara sí un plan abstracto de movimientos, el cual se compone de los puntos que debe visitar el robot.

El módulo traductor toma la información del plan abstracto, obtenido mediante el planificador, y lo convierte en comandos que el robot pueda interpretar, tales como ángulos para girar y distancias por recorrer. Para ello, se hace uso del lenguaje Java-LeJOS y de los métodos que la librería *lejos.robotics.navigation* de este lenguaje trae consigo [9]. Este traductor genera un plan ejecutable que puede ser usado en el robot que se diseña para la prueba. Este plan ejecutable como un programa en el lenguaje Java-LeJOS está asociado con la implementación propia del robot.

Finalmente, al módulo ejecutor, que está en el propio robot, se le pasa desde el módulo traductor el plan ejecutable con el fin de realizar cada uno de los comandos que conforman dicho plan; adicionalmente, este módulo implementa un monitor de la ejecución del plan, que emplea un método para corregir los errores originados al realizar los giros y desplazamientos del robot haciendo uso para estas correcciones de un compás magnético.

IV. El robot y su modelado en el sistema

El robot construido para realizar las pruebas consta de dos ruedas delanteras motorizadas estándar, que comparten el mismo eje de rotación, con una distancia entre ellas de 14 centímetros; además, posee una rueda no motorizada de tipo castor en la parte de atrás. Para representar el robot cuando se realizan los cálculos de su movimiento en un plano 2D, inicialmente se tomó desde la IGU el rectángulo que lo representa y se remodeló como un robot circular. En el caso del robot de prueba utilizado, este círculo tiene un diámetro de 15 centímetros y circunscribe todo el espacio que ocupa el robot en su proyección sobre el suelo en un plano de dos dimensiones. Los detalles de este se muestran en la Figura 2.

Finalmente, la representación como círculo del robot se transforma haciendo uso de un enfoque en el que el robot se representa como un punto, pero la representación de las fronteras de los obstáculos del ambiente se extienden en una proporción igual a la medida del radio del robot inicialmente modelado (en este caso 7.5 cm) [10,11]. Esta suposición también se aplica a las fronteras que circunscribe el ambiente de prueba por donde se moverá el robot. En la Figura 3 se observa una ilustración como ejemplo, donde se detallan las fronteras reales de los obstáculos y del propio ambiente en una dimensión *r* correspondiente al radio del robot contemplado inicialmente como un círculo.

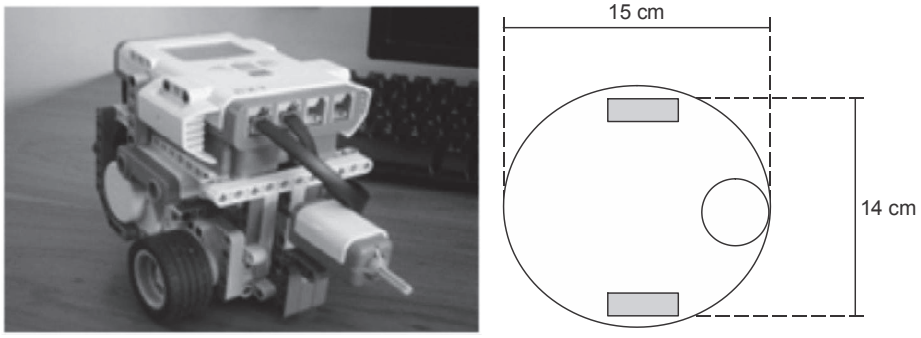

Figura 2. Robot físico usado en las pruebas y su representación geométrica inicial

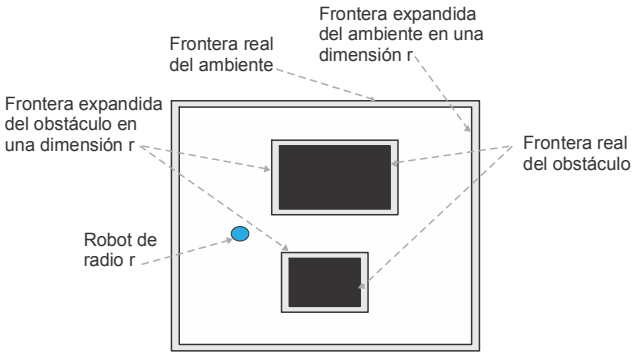

Figura 3. Ejemplo de área con fronteras extendidas para representar la geometría del robot

V. Características básicas del planificador

La descomposición del espacio de trabajo en celdas permite al robot conocer el espacio libre, el cual corresponde al conjunto de celdas que no contienen partes de algún obstáculo [4]. A continuación, se detallan las dos técnicas de descomposición de celdas, la técnica de descomposición exacta y la de descomposición adaptativa, implementadas en el planificador desarrollado. Estas se utilizaron para determinar no solo el espacio por donde se puede desplazar el robot en una configuración específica del ambiente, sino también para determinar la ruta más corta a seguir por este robot entre los dos puntos proporcionados por el problema de planificación de movimientos definido.

A. Descomposición exacta de celdas

Con la descomposición exacta de celdas, el planificador de rutas del robot tiene como objetivo encontrar un grafo de adyacencia que indique cuáles celdas libres de obstáculos comparten una frontera común, con el objetivo de identificar el espacio libre por donde se puede desplazar el robot. Los nodos del grafo corresponden a las celdas y las aristas conectan nodos de celdas adyacentes.

La descomposición de celdas en el planificador se realiza siguiendo los siguientes pasos: (1) Se construye el espacio de configuración asociado al espacio de trabajo del robot sobre el cual se identificarán las celdas, (2) se construyen las celdas de acuerdo con las indicaciones del algoritmo, (3) se construye el grafo de adyacencias, (4) se determinan las celdas que contienen el comienzo y el final, respectivamente, y, por último, (5) se busca una ruta dentro del grafo de adyacencia. Nótese que este grafo puede servir como un rutero (o mapa de ruta) del espacio libre.

Las celdas tienen una estructura simple, lo cual permite que sean cubiertas con movimientos sencillos del robot, tales como giros y avances en línea recta, ya que una vez el robot visita cada celda, el cubrimiento se logra. En otras palabras, el cubrimiento puede ser reducido a encontrar un camino a través del grafo de adyacencia.

Dentro de los algoritmos para la descomposición por celdas, uno de los más conocidos es la descomposición trapezoidal [4, 5, 6], que está relacionada con la representación poligonal del espacio, y aunque requiere mayor complejidad computacional que otras técnicas, presenta mayor exactitud y completitud. Esta técnica hace referencia, en particular, a las celdas de dos dimensiones que tienen forma de trapezoides. Un ejemplo de este tipo de descomposición se puede ver en la Figura 4, la cual detalla dos obstáculos y las posiciones de inicio y objetivo del robot.

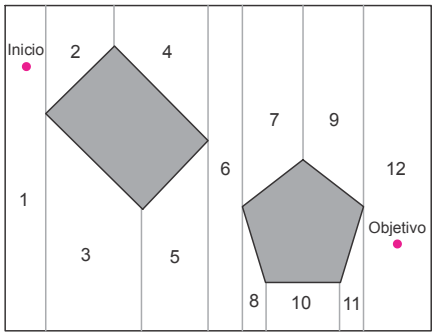

Figura 4. Ejemplo de un espacio de configuración descompuesto con trapezoides

En esta técnica se establece un sistema de coordenadas *(x, y)* para modelar el escenario en el cual se va a desplazar el robot. Así, el espacio libre estará delimitado por un polígono trapezoidal, y se asume que todos los obstáculos son polígonos.

Para realizar la descomposición en cada vértice *v* se dibujan dos segmentos, uno llamado extensión vertical superior, y el otro, extensión vertical inferior. El primero se refiere a incrementar la coordenada *y*, y el segundo, a decrementarla. Las extensiones verticales superior e inferior comienzan en cada vértice y terminan cuando interceptan el primer borde del polígono que se localice inmediatamente arriba o debajo de *v*, respectivamente; o bien, cuando llegan a la frontera del escenario. De esta manera los vértices tendrán una extensión vertical superior, inferior, ambas, o ninguna (en el caso de esquinas cóncavas).

Una vez que las celdas que contienen el punto inicial y el final estén definidas, se calcula el grafo de adyacencia para determinar la trayectoria. Sin embargo, el resultado de la búsqueda del grafo es una secuencia de nodos, y no una secuencia de puntos embebidos en el espacio libre. Es importante tener esto en cuenta, pues cada celda tendrá una posición y un área determinadas. Luego, se construye el camino conectando los puntos medios de las extensiones verticales a los centroides de cada trapezoide. Así se obtiene un camino libre de colisiones, a través del espacio libre que se deriva del grafo de adyacencia.

B.Descomposición adaptativa de celdas

Esta técnica, también conocida como descomposición aproximada, en comparación con la técnica basada en una descomposición exacta de celdas, presenta ciertas ventajas y desventajas. Como ventaja, señalamos que la implementación es más sencilla, pues consiste en la división en figuras conocidas y de fácil cómputo. Sin embargo, como desventaja está el hecho de ser una técnica incompleta, donde es posible no encontrar una solución aunque exista. En esta técnica,una celda puede ser etiquetada como:

- Vacía: si y solo si su interior no intercepta la región del obstáculo.
- Llena: si y solo si toda la celda está contenida en la región del obstáculo.
- Mixta: de otro modo diferente a los dos anteriores.

Se parte de tener todo el escenario como una celda llena. El objetivo es descomponer el espacio hasta que no queden más celdas mixtas. La eficiencia de esta técnica radica en que solo basta con descomponer las celdas mixtas, es decir, una vez que se tiene una celda llena, ya no se procede a dividirla.

Este tipo de descomposición presenta el problema de que su cálculo puede tender al infinito. Un ejemplo de esto ocurre cuando un obstáculo atraviesa la diagonal de la celda. Para su solución es necesario establecer una cota límite, que en la implementación corresponde al tamaño mínimo que debe tener una celda para poderse descomponer. Cuando se llegue

a una celda mixta, pero que sus dimensiones no cumplan la cota para dividirse, se considera una celda llena. Esta decisión genera la incompletitud del algoritmo, debido a que el punto final puede encontrarse en la parte vacía de esta celda mixta que se clasificó como llena.

La implementación desarrollada maneja una clase llamada *cell* (celda), que hereda de la clase *rectangle* (rectángulo) [12] existente en las librerías propias de Java. Por esta razón, una *cell* también es de tipo *rectangle*. Esto hace que *cell* herede cuatro enteros, provenientes de *rectangle*, que hacen referencia los dos primeros a las coordenadas *(x, y)* donde comienza el rectángulo; el tercero a la anchura; y el cuarto a la altura del rectángulo. Adicionalmente, *cell* tiene unos métodos que permiten conocer, dada una lista de polígonos que representan los obstáculos, si un objeto *cell* está lleno, vacío o mixto.

La eficiencia de esta técnica radica en dividir solamente las celdas mixtas, lo cual implica que es fundamental etiquetar adecuadamente las celdas. En la implementación se pregunta primero si la celda es llena. Para que cumpla con esa característica, se recorre cada uno de los obstáculos que son manejados como objetos de la clase *Polygon* (polígono) propia de Java. Esto permite preguntar si una celda manejada como un objeto rectángulo está contenida enteramente en el polígono del obstáculo. De manera que, si la celda está dentro del obstáculo, será una celda llena y no es necesario dividirla. De lo contrario, se continúa con la operación de buscar si es mixta. Sabiendo ya que no es llena, lo que se hace es recorrer cada punto, y si se encuentra un punto que esté dentro de la celda y dentro del obstáculo, se dice que es mixta. Para esto es muy importante ver que ya se ha descartado la posibilidad de que fuese una celda llena. Finalmente, en el caso de que sea vacía, retorna *false* pues no se encontraron puntos dentro de la celda que a la vez estuviesen dentro del obstáculo.

Una vez que todas las celdas hayan sido etiquetadas, lo que sigue es almacenar las celdas en una lista de objetos de la clase *cell*. Al dividir cada celda en otras cuatro se obtienen las nuevas celdas que describirán el espacio por el cual el robot se podrá desplazar. Esto se hace dividiendo en la mitad la anchura y la altura de cada celda, que, a su vez, se debe dividir, y estableciendo que cada celda comience en el punto respectivo de la división.

Para definir la cota límite, mencionada anteriormente, se sigue la sugerencia hecha en [10], donde el tamaño mínimo puede ser proporcional al tamaño del robot, y de acuerdo con la capacidad en cuanto a memoria del planificador, y las características deseadas. En este caso, se implementó de forma tal que la celda mínima fuera de 3cm de lado (9 cm^2 de área).

Al igual que con la descomposición exacta, se debe encontrar un grafo de conexión, el cual está asociado con la descomposición *P* del espacio, y se define de la siguiente manera:

- Los nodos del grafo son las celdas vacías de *P*.
- Dos nodos del grafo están conectados si y solo si las celdas correspondientes son adyacentes.

Ahora bien, lo que resta es establecer la trayectoria, lo cual se logra, primero, preguntando si el punto inicial y el final están en celdas vacías, ya que de lo contrario no habría trayectoria posible. Luego, se encuentra un camino, que en la implementación realizada fue el de la ruta más corta, haciendo uso de una variación del algoritmo de Dijkstra [7, 8].

VI. Corrección de la trayectoria

El modelo de error usado para los robots móviles tipo diferencial se relaciona con su odometría, que a su vez se da en función de la distancia recorrida por cada llanta. En [10], el estimado de la matriz de covarianza para el error en la posición se da con la expresión (1).

$$\Sigma_{\Delta} = covar\,(\,\Delta s_r \quad \Delta s_l\,) = \begin{bmatrix} k_r\left|\Delta s_r\right| & 0 \\ 0 & k_l\left|\Delta s_l\right| \end{bmatrix} \qquad (1)$$

Donde Δs_r, Δs_l son las distancias recorridas por cada llanta (para derecha, *r*; para izquierda, *l*), y k_r y k_l son constantes de error que representan parámetros

no determinísticos de la interacción del robot con el suelo. Se puede observar que la varianza de los errores de las llantas es proporcional al valor absoluto de las distancias recorridas.

Para reducir la incertidumbre, se hizo uso del compás fabricado por HiTechnic para el robot Lego Mindstorms; de manera que, antes de que viaje a cada punto del plan óptimo, el robot gira en dirección al siguiente punto por alcanzar, según los cálculos realizados por el planificador. En este punto, el ejecutor lee la dirección actual del robot haciendo uso del compás y lo compara con aquella a la que debería ir. En caso de que existan diferencias, se corrige el giro hasta llegar a la dirección deseada, disminuyendo así la incertidumbre en su trayectoria.

VII. Pruebas de experimentación

Con el fin de evaluar las dos técnicas de descomposición de celdas implementadas en el planificador se montó un ambiente de prueba básico, consistente en el uso de dos o tres obstáculos cuadrados y un área de trabajo cuadrada, dentro de la cual se puede desplazar el robot que se usa en este estudio (ver Figura 5).

En estas pruebas se utilizaron tres escenarios diferentes, como se puede observar en la Figura 6. En el primer escenario (Figura 6-a) se consideran dos obstáculos separados entre sí por una distancia igual a 2r, donde r=7.5 cm representa el radio de la circunferencia que circunscribe al robot utilizado. En el segundo escenario (Figura 6-b) se consideran dos obstáculos separados entre sí por una distancia menor a 2r. En el tercer escenario (Figura 6-c) se consideran tres obstáculos, los obstáculos A y B separados entre sí por una distancia menor a 2r, y la distancia entre A y C al igual que entre B y C es mayor a 2r.

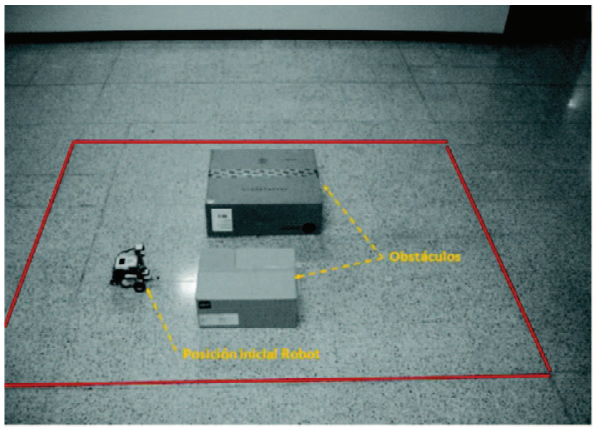

Figura 5. Un escenario real de las pruebas realizadas

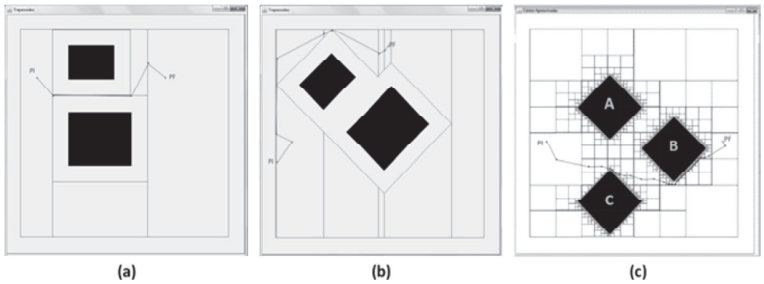

Figura 6. Escenarios de pruebas

Adicionalmente, para cada una de las dos técnicas que se debían evaluar en los tres espacios de configuración diseñados, se realizaron dos pruebas; la primera, sin contemplar la geometría del robot, asumiendo que el robot simplemente es un punto, y la segunda, contemplando la geometría del robot, asumiendo que el robot es un punto junto con la expansión de las fronteras de los obstáculos y las fronteras que encierran el espacio de trabajo en el que se mueve el robot, como se explicó en la sección V.

La idea de tener pruebas con geometría y sin geometría, busca evaluar el desempeño que podría haber tenido el robot considerando una geometría más aproximada a su verdadera configuración. En la Figura 6-a se observa la configuración realizada para la descomposición exacta sin geometría, mientras que en la Figura 6-b se observa la configuración para la descomposición exacta con geometría. Para el caso particular de las pruebas con la descomposición adaptativa, se consideró que la cota mínima de celda sería una celda de 3cm de lado (9 cm^2 de área). En la Figura 6-c se puede observar la ruta y la identificación de las celdas determinadas haciendo uso de la descomposición adaptativa con geometría, para el escenario 3.

Los resultados obtenidos en estas pruebas se detallan en las tablas 1, 2 y 3; cada una de las tablas contiene los resultados para cada escenario considerado según la Figura 6. Cada tabla maneja los siguientes datos: la distancia total recorrida por el robot en centímetros; la distancia al punto final en centímetros, que representa la distancia entre el robot una vez terminado su recorrido calculado y ejecutado hasta al punto final, y por último, el error en porcentaje (%), el cual se obtiene de dividir la distancia al punto final por la distancia total recorrida por el robot.

Tabla 1. Pruebas para el Escenario 1

	Distancia total recorrida (cm)	**Distancia al punto final (cm)**	**Error (%)**
D. Exacta sin geometría	176,4	8,1	4,6
D. Exacta con geometría	152,1	5,6	3,7
D. Adaptativa sin geometría	387,6	14,9	3,8
D. Adaptativa con geometría	142,6	7,6	5,3

Tabla 2. Pruebas para el Escenario 2

	Distancia total recorrida (cm)	**Distancia al punto final (cm)**	**Error (%)**
D. Exacta sin geometría	238,8	9,1	3,8
D. Exacta con geometría	135,2	14,0	10,4
D. Adaptativa sin geometría	307,3	45,6	14,8
D. Adaptativa con geometría	180,4	1,4	0,8

Tabla 3. Pruebas para el Escenario 3

	Distancia total recorrida (cm)	**Distancia al punto final (cm)**	**Error (%)**
D. Exacta sin geometría	311,1	16,5	5,3
D. Exacta con geometría	244,1	9,6	3,9
D. Adaptativa sin geometría	292,6	30,3	10,4
D. Adaptativa con geometría	327,0	9,7	3,0

En cuanto a la distancia total recorrida por el robot, para el caso de la técnica de descomposición exacta, se encontró que en los tres escenarios de prueba es menor la distancia total recorrida cuando se tiene en cuenta la geometría del robot que cuando no se tiene en cuenta; esto se justifica porque el robot puede generar errores al calcular zonas donde puede desplazarse como punto, pero no como un objeto en el espacio, lo cual debe ser corregido en tiempo de ejecución, causando que el robot realice recorridos adicionales que lleven a corregir su solución inicial. Para el caso de la descomposición adaptativa, para los escenarios 1 y 2, la distancia total recorrida por el robot es menor cuando se tiene en cuenta su geometría que cuando no se tiene en cuenta; en el caso del escenario 3 ocurre lo contrario, lo cual puede ser explicado por el hecho de que en el escenario 3 la distancia entre los obstáculos B y C es mucho más grande que el diámetro considerado para representar el robot; así, es viable que al realizar la descomposición sin geometría se hubiese encontrado una ruta más corta sin inconvenientes (es decir, que el robot se choque con alguno de los objetos) que la ruta encontrada al realizar la descomposición con geometría. Este último caso restringe más el área libre por donde se puede desplazar el robot, ya que extiende los obstáculos haciendo que la ruta obtenida por este método sea más larga.

De lo anterior se desprende que, en general, es más corto el recorrido realizado por el robot cuando se tiene en cuenta su geometría, que al representarlo como un punto en el espacio; esto es justificable, ya que la geometría, como un círculo, está más cercana a la geometría real del robot, lo cual permite calcular opciones de recorridos también más cercanos a la realidad.

En cuanto a la comparación de las dos técnicas y a la evaluación de los resultados, en términos de la distancia total recorrida, de la Tabla 4 obtenida de las Tablas 1, 2 y 3, se desprende que la técnica de descomposición exacta de celdas arroja mejores resultados que la técnica de descomposición adaptativa. En cuatro de los seis casos comparativos obtenidos, teniendo y no teniendo en cuenta la geometría, con la descomposición exacta se obtiene una distancia total menor que la obtenida con la técnica de descomposición adaptativa.

En cuanto al error generado al comparar las dos técnicas, haciendo uso de los datos de la Tabla 5, obtenida de las Tablas 1, 2 y 3, se observa que ambas técnicas presentan igual desempeño, ya que cada una presenta igual número de casos comparativos (tres), donde se obtiene un menor porcentaje (%) de error frente a la otra.

Tabla 4. Comparativo de la distancia total recorrida (cm)

	Escenario 1	**Escenario 2**	**Escenario 3**
D. Exacta sin geometría	176,4	238,8	311,1
D. Adaptativa sin geometría	387,6	307,3	292,6
D. Exacta con geometría	152,1	135,2	244,1
D. Adaptativa con geometría	142,6	180,4	327,0

Tabla 5. Comparativo del error (%)

	Escenario 1	**Escenario 2**	**Escenario 3**
D. Exacta sin geometría	4,6	3,8	5,3
D. Adaptativa sin geometría	3,8	14,8	10,4
D. Exacta con geometría	3,7	10,4	3,9
D. Adaptativa con geometría	5,3	0,8	3

VIII. Conclusiones y trabajo futuro

En este trabajo se definió una arquitectura para implementar los algoritmos de planificación de movimiento,la cual permitió evaluar dos técnicas para la descomposición del espacio de trabajo en celdas utilizadas para determinar el espacio libre por donde puede desplazarse un robot, con el fin de encontrar el plan de movimientos que debe seguir el robot para alcanzar un punto del espacio tomado como objetivo. Esta arquitectura consta de 5 componentes: la interfaz de usuario (IGU) del sistema, el módulo que permite incluir la geometría del robot, el planificador de movimientos, el traductor que permite trasladar el plan del planificador en comandos propios del robot, y el ejecutor del plan en el robot, dentro del cual se contempla un mecanismo de corrección de la trayectoria para enfrentar la incertidumbre en el desplazamiento por factores externos no identificados.

En el marco de este trabajo se desarrolló un robot y se implementó la arquitectura de planificación anteriormente enunciada. Para considerar la geometría del robot se optó por extender las fronteras reales de los obstáculos y del propio ambiente en una dimensión correspondiente al radio del robot, contemplado inicialmente como un círculo de radio 7.5 cm. Este robot fue empleado en las diferentes pruebas que se realizaron para evaluar las técnicas de descomposición de celdas.

Las técnicas de descomposición de celdas evaluadas son la descomposición exacta de celdas y la descomposición adaptativa de celdas. En el marco de este trabajo se realizó una descripción teórica de ellas y de su construcción y puesta a prueba según la arquitectura propuesta.

En la experimentación realizada sobre el uso de estas técnicas se operó con tres escenarios de prueba, a partir de los cuales fue posible obtener que: (1) para ambas técnicas en general es más corto el recorrido realizado por el robot cuando se tiene en cuenta la geometría del robot que al representarlo como un punto en el espacio, (2) la descomposición exacta de celdas presenta un mejor resultado en términos de la distancia total recorrida que la descomposición de celdas adaptativa, y (3) ambas técnicas de descomposición de celdas presentan igual desempeño en cuanto al error generado en la localización real frente a la esperada del robot, una vez se ha llegado al punto final del plan obtenido y ejecutado.

Ahora bien, es importante tener en cuenta que los escenarios escogidos para la experimentación no son todos los posibles, así que, como un trabajo futuro, se puede experimentar en diversos escenarios; por ejemplo, los dos algoritmos presentan dos desventajas que en el presente artículo no se manifestaron: por un lado, en el caso de la descomposición exacta, si las márgenes están muy lejos, aunque haya una ruta evidentemente más corta, él buscará el punto medio de las extensiones verticales, por tanto, su desplazamiento no es el esperado, y por otro, la descomposición adaptativa resulta inconveniente cuando es poco el espacio ocupado por los obstáculos, pues se podrían obtener en la descomposición celdas libres no divididas o muy grandes, haciendo que el robot tenga que desplazarse hasta los centroides de esas celdas grandes.

El compás permitió optimizar la ruta de navegación del robot móvil. Aunque todavía hay errores no determinísticos, sistemáticamente el compás magnético resulta ser una buena herramienta para la corrección de trayectorias. Como trabajo futuro se desea experimentar con otros sensores y técnicas que permitan disminuir el error en la localización real final del robot.

Como otra línea de trabajo futuro, al considerar la geometría del robot, es bueno pensar en aproximarlo al máximo posible a la configuración geométrica real del robot, siendo esto un trabajo para realizar con el fin de tener una configuración más acorde con la realidad.

Por último, se propone diseñar un ambiente de experimentación que trabaje por una configuración del robot lo más cercana posible a la real, teniendo en cuenta que el robot es tipo diferencial, con el fin de generar nuevos resultados en torno a la exactitud y la eficiencia de las trayectorias.

Agradecimientos

El presente trabajo ha sido apoyado por el proyecto «Programa de Fortalecimiento del Grupo de Investigación Sistemas Inteligentes Web –Sintelweb– código quipu 20201009532», de la Universidad Nacional de Colombia, sede Medellín.

De manera especial se agradece a Jeferson D. Ossa, estudiante del programa Ingeniería de Sistemas e Informática de la Universidad Nacional de Colombia, sede Medellín, por su apoyo en el proceso de implementación, llevado a cabo durante la experimentación.

Referencias

[1] LeJOS, Java for Lego Mindstorms [Online]. Disponible en: http://lejos.sourceforge.net/

[2] LEGO.com MINDSTORMS: Products - NXT 2.0 - 8547 [Online]. Disponible en: http://mindstorms.lego.com/en-us/products/default.aspx

[3] M. Ghallab, D. Nau, P. Traverso.Automated Planning, theory and practice.San Francisco: Morgan Kaufman, 2004.

[4] Jean-Claude Latombe. Robot Motion Planning. Kluwer Academic Publishers, 1991.

[5] H. Choset et al. Principles of Robot Motion: Theory, Algorithms, and Implementations. Boston: MIT Press, 2005.

[6] S. LaValle. Planning Algorithms. Cambridge: Cambridge University Press, 2006.

[7] K.H.Rosen. Discrete Mathematics and its Applications.Nueva York: McGraw-Hill, 2007.

[8] Dijkstra, implementación en Java. Disponible en: http://algowiki.net/wiki/index.php?title=Dijkstra's_algorithm

[9] lejos.robotics.navigation (leJOSNXJ API documentation). Disponible en: http://lejos.sourceforge.net/nxt/nxj/api/lejos/robotics/navigation/package-summary.html

[10] R. Siegwart, I. RezaNourbakhsh, D. Scaramuzza. Introduction to Autonomous Mobile Robots. The MIT Press, 2004.

[11] R. Robin. Murphy, Introduction to AI Robotics. Massachussetts: The MIT Press 2000.

[12] Rectangle (Java 2 Platform SE 6). Disponible en: http://download.oracle.com/javase/6/docs/api/java/awt/Rectangle.html

[13] Pilot (leJOSNXJ API documentation). Disponible en: http://lejos.sourceforge.net/nxt/nxj/api/lejos/robotics/navigation/Pilot.html

[14] J. A. Guzmán, I.D. Torres, C. A. Vélez, «Experiencias en la implementación de las Técnicas de Descomposición Aproximada y Trapezoidal para la Navegación en Robots Móviles Lego», presentado en la XI Latin American RoboticsCompetition, Bogotá, Colombia, 2011.