Desarrollo de un Algoritmo Híbrido Dijkstra - Montecarlo para la Planeación y Optimización de Rutas en un Robot Móvil

Aceves-Fernandez Marco Antonio, Salinas-González Eric Francisco, Gorrostieta-Hurtado Efrén, Pedraza-Ortega Jesús Carlos, Ramos Arreguín Juan Manuel y Tovar Arriaga Saúl

Universidad Autónoma de Querétaro, Facultad de Informática
Campus Juriquilla, Av. de las Ciencias S/N, Juriquilla,
Delegación Santa Rosa Jauregui, CP 76230, Querétaro, México
marco.aceves@uaq.mx

Resumen

Se presenta un algoritmo híbrido para la planeación y optimización de rutas en un robot móvil. Este algoritmo se desarrolló utilizando una técnica basada en el algoritmo de Dijkstra para la planeación de rutas y el algoritmo de localización de Montecarlo (Montecarlo Localization ó MCL) para optimizar la ruta. Se realizaron pruebas bajo diversos escenarios de laberintos y se mostró que las pruebas fueron consistentes en todos los escenarios. Estas pruebas, demostraron la robustez del método planteado como una alternativa a los métodos existentes de planeación de rutas en robots móviles.

Palabras clave: Monte Carlo, Algoritmo Dijkstra, Robot Móvil, planeación de rutas.

1. Introducción

El principal objetivo de la robótica es la construcción de máquinas capaces de realizar tareas con la flexibilidad, la robustez y la eficiencia que exhiben los seres humanos. Los robots son potencialmente útiles en escenarios peligrosos para el ser humano, sucios o difíciles[1].

El robot móvil se caracteriza por realizar una serie de desplazamientos (navegación) y por llevar a cabo una interacción con distintos elementos de su entorno de trabajo (operación), que implican el cumplimiento de una serie de objetivos impuestos según cierta especificación.

Las tareas involucradas en la navegación de un robot móvil son: la percepción del entorno a través de sus sensores, de modo que le permita crear una abstracción del mundo; la planificación de una trayectoria libre de obstáculos y el guiado del vehículo a través de la referencia construida. De forma simultánea, el vehículo puede interaccionar con ciertos elementos del entorno. Así, se define el concepto de operación como la programación de las herramientas de a bordo que le permiten realizar la tarea especificada.

Muchos algoritmos que construyen mapas en tiempo de ejecución, necesitan robots con suficiente memoria, poder de procesamiento o técnicas de sensores avanzados[2]. Sin embargo, en muchas ocasiones es necesario utilizar robots limitados por razones de costo y espacio.

Existen diversas técnicas como lo son: Dead-Reckoning, Cadenas de Markov, redes de Bayes o Filtro de Kalman, Node Combination, entre otras [3-8]

El algoritmo de localización de Markov es más exacto que el filtro de Kalman y puede ser utilizado para una localización global. Sin embargo la localización de Markov es difícil de implementar. [5]

2. Algoritmo Dijkstra

El algoritmo Dijkstra se utilizará en esta contribución para obtener la ruta más corta del lugar origen al lugar destino [9].

Se comienza con un grafo dirigido conexo sin ciclos G. A cada arista e = (a, b) de este se le asigna un número real positivo llamado el peso de e, que se denota con w(e) o w(a, b). Si $x,y \in V$ pero $(x,y) \notin E$, se define que $w(x,y) = \infty$.

Cuando se da un grafo G con las asignaciones de peso aquí descritas, se dice que el grafo es un grafo ponderado.

Dado G, para cada $e = (a,b) \in E$ se interpreta p(e) como la longitud de una ruta directa de a a b. Para cualesquiera dos vértices $a,b \in V$ se escribe d(a,b) como la distancia (más corta) de a a b. Si no existe

tal camino (en G) de aab entonces se define d(a,b) = ∞, Para cualquier a$\in$ V, d(a,a) = 0.
Se Fija ahora v0$\in$ V. Entonces para cualquier v$\in$ V, se determina d(v0,v) un camino simple dirigido de v0 a v si d(v0,v) es finito.
Sea | V | = n. Para determinar la distancia más corta de un vértice fijo v0 a los demás vértices de G, así como un camino simple dirigido más corto para cada uno de estos vértices, se aplica el siguiente algoritmo.
Paso 1: Se inicializa i = 0 y S0 = {v0}. Se etiqueta v0 con (0,—) y cada v ≠ v0 con (∞, —)

Paso 2: Para cada v$\in$ Si, se reemplaza, la etiqueta de v por la nueva etiqueta final (L(v),y), donde

$$L(v) = \text{mín}\ \{L(v), L(u) + p(u,v)\},\ U \in Si \qquad (1)$$

Y y es un vértice en Si que produce el L(v) mínimo. Si se hace un reemplazo, esto se debe al hecho de que se puede ir de v0 a v y recorrer una distancia más corta si se recorre un camino que incluye una arista (y,v).
Paso 3: Si cada vértice de Si (para algún $0 \leq i \leq n-2$) tiene la etiqueta (∞, —) , entonces el grafo etiquetado contiene la información del camino más corto.
Si no, existe al menos un vértice v $\in$ Si que no está etiquetado como (∞, —) y se realizan las siguientes tareas:
Se selecciona un vértice vi+1, tal que L(vi+1) sea mínimo. Puede haber varios de estos vértices cuyo caso se elige alguno de los posibles candidatos. El vértice vi+1 es un elemento de Si que es el más cercano a v0.
Se actualiza i= i+1.
Si i = n – 1, el grafo etiquetado contiene la información del camino más corto. Se regresa al paso 2.

3. Localización Montecarlo (MCL)

Localización Monte Carlo (MCL) es una técnica estadística para la localización del robot basada en la localización de Markov. Como en la localización de Markov, se requiere un mapa del entorno del Robot. Esto es un filtro de Bayes [10].
Localización Monte Carlo (MCL) fue desarrollado por Dellaert y Fox [11] Ellos tomaron el método de filtrado de partículas utilizado en otras áreas como visión por computadora. Localización Monte Carlo es uno de los métodos más populares en la localización, porque ofrece soluciones a una gran variedad de problemas.

La idea principal del filtro de Bayes es estimar una densidad de probabilidad sobre el estado espaciado condicionado sobre los datos. Este posterior que se suele llamar la creencia (Bel) y se denota

$$Bel(xt) = p(xt|\ d0...t) \qquad (2)$$

En la ecuación 2, x denota el estado, xt es el estado en el tiempo t y d0, t denota el tiempo inicial desde 0 a t. Denotando o (observación) y a (acción).
El modelo de movimiento, p(x‘|x,a), es una generalización probabilística de la cinemática del robot. Para un robot que operan en el plano las posiciones de x y x’ son variables en tres dimensiones.

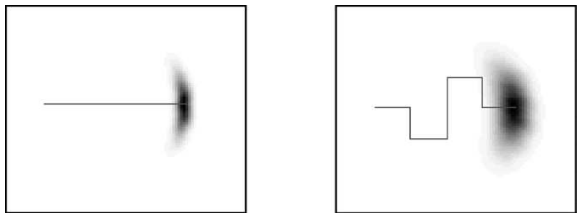

Fig. 1. Densidad p(x‘|x,a) después de mover 40 metros (diagrama izquierdo) y 80 metros (diagrama de la derecha).

En el ejemplo de la figura 1, la postura inicial de x se muestra a la izquierda, y la línea continua representa los datos de odometría, medido por el robot. El área sombreada de la derecha muestra la densidad p(x‘|x,a): la posición más oscura, es la que tiene mayor probabilidad. Una comparación de los dos diagramas revela que el margen de incertidumbre depende del movimiento global: A pesar de que la postura de un robot libre de ruido es la misma para ambos segmentos de movilidad, la incertidumbre en el diagrama de la derecha es más grande debido a la mayor distancia recorrida por el robot[11].
Un modelo de toma de muestras es una rutina que acepta x y a como entrada y genera al azar posiciones x’ distribuidas de acuerdo a p(x‘|x,a). La secuencia de conjuntos de partículas se aproxima a la densidad de un robot que mide sólo odometría.

4. Resultados Experimentales

Para poder modelar el comportamiento de este algoritmo: Dijkstra - Montecarlo, se construye el ambiente por el cual Robot tendrá que navegar. En éste laberinto de n * n (figura 2).el robot navegará desde el inicio(nodo 1) hasta alcanzar la meta (en este caso el nodo 25).

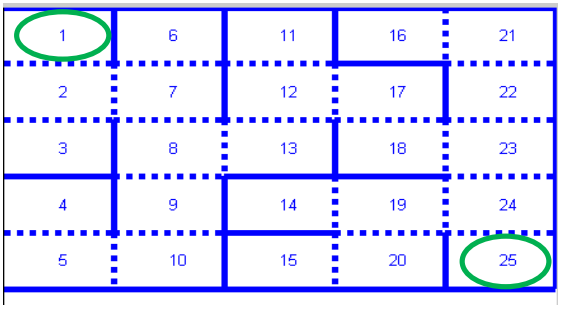

Fig. 2. Generación del Laberinto

En la Figura 2 se muestran las paredes (líneas solidas), mientras que las puertas por donde el Robot puede navegar hasta encontrar su objetivo se muestra mediante las líneas punteadas. Para encontrar el camino más corto dentro del laberinto, se utilizara el algoritmo de Dijkstra y la teoría de grafos. En la figura 3 se muestra el camino que siguió el robot navegando en el laberinto de la figura 2.

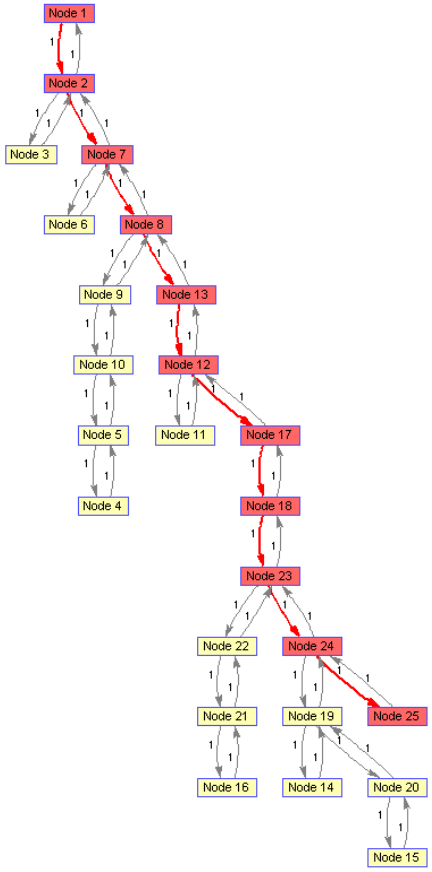

Fig. 3. Cálculo de la ruta mas corta mediante el algoritmo Dijkstra.

En el caso de la figura 3, la ruta más corta es; 1,2,7,8,13,12,17,18,23,24,25 . Una vez obtenida se traza la trayectoria por la cual navegara el Robot. La simulación se realiza por medio del algoritmo de Montecarlo. Esto se muestra en la figura 4.

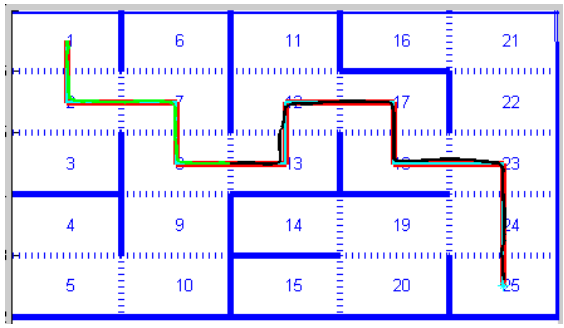

Fig. 4. Simulación de la navegación del Robot en una ruta establecida mediante Localización Monte Carlo.

Para un laberinto de 5 x 5 como el que se muestra en la figura 4, se realizaron diversos experimentos para demostrar que sin importar la posición de inicio o final, el algoritmo Dijkstra – MCL obtiene la ruta mas corto en un tiempo relativamente corto. Los tiempos para obtener la ruta mas corta y para recorrer la trayectoria se muestran en las tablas 1 y 2, respectivamente.

No. de Exp.	Tiempo
1	8.14E-04
2	9.62E-04
3	3.96E-04
4	4.68E-04
5	7.73E-04
6	1.16E-01
7	2.51E-04
8	2.47E-04
9	2.50E-04
10	2.52E-04

Tabla 1. Resultados experimentales de un laberinto de 5x5 para obtener la ruta mas corta.

Como puede verse en la tabla 1, los tiempos para poder calcular la ruta mas corta son consistentes, aunque depende el tiempo del número de nodos a recorrer. Además, se realizaron experimentos con laberintos de 6x6, 7x7 y 10x10, utilizando la metodología citada. La tabla 2 muestra los resultados de un laberinto 10x10.

No. de Exp.	Tiempo
1	2.94E+01
2	3.07E+01
3	9.04E+01
4	4.20E+01

5	1.22E+01
6	8.84E+01
7	1.52E+02
8	2.47E-04
9	3.56E+01
10	4.88E+01

Tabla 2. Resultados experimentales de un laberinto de 10x10 para obtener la ruta mas corta.

5. Conclusiones y trabajo futuro

Se ha mostrado que el algoritmo Dijkstra – MCL es útil para la planeación y optimización de rutas para robots móviles. Este algoritmo ha demostrado consistencia en diferentes escenarios de laberintos, por lo que puede ser usado en aplicaciones donde se necesita robustez y consistencia.
Como trabajo futuro se sugiere la implementación de este algoritmo en software embebido para implementarse en un robot móvil.

Referencias

[1] Tingle, D; Ball E; Augat, M., **Maze Solving by Learning State topologies**, Comp. Sci., Swarthmore College, 2009.

[2] Bruggemann, Bernd; Kamphans, Tom; Langetepe, Elmar, **Escaping from a Labyrinth with One-way Roads for Limited Robots**, Comp. Sci., University of Bonn, Bonn, Germany,2009.

[3] Rohrmuller, Florian; Althoff, Matthias; Wollherr, Dirk; Buss, Martin, **Probabilistic Mapping of Dynamic Obstacles Using Markov Chains for Replanning in Dynamic Environments**, IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, Sept, 22-26, 2008.

[4] Widodo Budiharto, Djoko Purwanto and Achmad Jazidie, **A robust obstacle avoidance for service robot using Bayesian approach**, International Journal of Advanced Robotic Systems, Vol. 8, No. 1, ISSN 1729-8806, pp 37-44, 2011.

[5] Ghahraman Zoubin, **An introduction to Hidden Markov Models and Bayesian**, International Journal of Pattern Recognition and Artificial Intelligence, no. 15, vol. 1, pp. 942, 2001.

[6] Rong-Jong Wai • Chia-Ming Liu • You-Wei Lin, **Robust path tracking control of mobile robot via dynamic petri recurrent fuzzy neural network**, Soft Comput., no 15, pp. 743–767, 2011.

[7] Yanrui Geng Æ Jinling Wang, **Adaptive estimation of multiple fading factors in Kalman filter for navigation applications**, GPS Solut., vol. 12, pp. 273–279, 2008.

[8] Xin Lu, Martin Camitz, **Finding the shortest paths by node combination**, Applied Mathematics and Computation, vol. 217, pp.6401–6408, 2011.

[9] Youming Li, Ardian Greca, and James Harris, **On Dijkstra's Algorithm for Deadlock Detection**, Advanced Techniques in Computing Sciences and Software Engineering, pp. 385-387, 2010.

[10] Stamenova, Stiliyana, **Solving the Maze: Robot Localization Using the Monte Carlo Localization Algorithm and Shape Context**, Macalester College, Ph.D. Thesis 2009.

[11] Thrun,Sebastian;Fox,Dieter;Burgard,Wolfram;Dellaert,Frank,**Robust Monte Carlo Localization for Mobile Robots**, Comp. Sci., Carnegie Mellon University ,Pittsburgh, PA,2001.