Año: 2020 Volumen: 1
Número 6, ISSN: 2448-623X

Robot solucionador de laberintos usando procesamiento digital de imágenes

J. Sifuentes-Mijares[*1], F. H. Hiram Brahim[1], J. E. Márquez Acosta[1], J. L. Santillán Ávila[1].

***Resumen*— Un problema fundamental en los robots móviles es la determinación de trayectorias, es decir, establecer el camino por dónde el robot debe moverse. También lo es, el buscar el camino más óptimo para ir de un punto a otro, evitando obstáculos. El objetivo de este artículo es la realización práctica de un sistema de control por visión de un robot móvil que ponga a prueba dos aspectos: el uso del procesamiento digital de imágenes como una alternativa para la definición de trayectorias en robots móviles y el uso de uno de los algoritmos de la teoría de grafos para la determinación del camino más corto al objetivo. El sistema presentado es un robot móvil con ruedas capaz de seguir una trayectoria calculada; la cual es la solución de un laberinto. Esta solución se calcula a partir de una imagen tomada por una cámara, la cual está colocada en la parte superior de un laberinto. La imagen es procesada en Matlab para encontrar la solución del laberinto. En la determinación de la trayectoria se usa un algoritmo llamado "Floodfill", que siempre busca encontrar el camino más corto desde la salida hasta la meta. Finalmente, esta solución es enviada al robot mediante Bluetooth.**

***Palabras claves*— Control, Robot móvil, Procesamiento Digital de Imágenes PDI.**

***Abstract*— A fundamental problem in mobile robots is the determination of trajectories, in other words, establishing the path where the robot must move. It is also, looking for the most optimal way to go from one point to another, avoiding obstacles. The objective of this article is the practical realization of a vision control system of a mobile robot that tests two aspects: the use of digital image processing as an alternative for the definition of trajectories in mobile robots and the use of one of graph theory algorithms for determining the shortest path to the objective. The presented system is a mobile robot with wheels capable of following a calculated trajectory; which is the solution of a maze. This solution is calculated from an image taken by a camera, which is placed on top of a maze. The image is processed in Matlab to find the solution of the maze. In the determination of the trajectory an algorithm called "Floodfill" is used, which always seeks to find the shortest path from the start to the finish. Finally, this solution is sent to the robot via Bluetooth.**

***Keywords*— Control, Mobile robot, Digital Image Processing DIP.**

[1]*Tecnológico Nacional de México/Instituto Tecnológico de la Laguna, DIE & DEPI, Boulevard Revolución y Av. Tecnológico de la Laguna, Centro SN, C.P. 27000, Torreón, Coahuila, México.*

* jsifuentesm@correo.itlalaguna.edu.mx.

I. Introducción

Los robots han ido sustituyendo al hombre en muchas actividades físicas e inclusive intelectuales [1]. Un campo de innovación y desarrollo es el de los automóviles autónomos. Esto se logra usando sensores en el automóvil que detectan obstáculos. Apoyado de la fotografía satelital es posible monitorear el tráfico en tiempo real, decirle al automóvil como evitar algún obstáculo y más importante, indicarle al vehículo cual es el camino más óptimo para llegar a su destino.

El desarrollo de robots móviles, responde a la necesidad de extender el campo de aplicación de la robótica, algunos trabajos relacionados son los siguientes: En el 2019 Kovacs et al. presentaron un trabajo en el "International Conference on Engineering of Modern Electric Systems (EMES)", sobre un robot móvil autónomo que resuelve un laberinto usando inteligencia artificial, empleando el método de la mano izquierda y la de callejón sin salida [2]. En el 2019 T. Ribeiro et al. presentaron un trabajo sobre la simulación de un sistema de robot móvil "ROLL ONE" en un laberinto utilizando aprendizaje Q por refuerzo para lograr resolver el laberinto [3]. I. Iturrate et al. en el 2013 presentan una plataforma pedagógica basada en lo que ellos llaman juegos serios, en donde les dan a los estudiantes la posibilidad de conectarse remotamente con un robot, que está dentro de un laberinto, para la búsqueda de respuestas y pregunta. La plataforma les permite manejar o construir programas basados en bloques gráficos evitando así la difícil tarea de aprender la sintaxis de programación [4]. B. Rahnama et al. en el 2013 presentan un trabajo de robots multi agentes trabajando en forma cooperativa sobre un laberinto desconocido para investigar su estructura. Utiliza también el algoritmo Floodfill y Floodfill modificado y un algoritmo de semántica lógica ALCK_EF para resolver el laberinto [5]. En el 2012 el trabajo presentado por B. Rahnama et al. es un algoritmo que evita obstáculos y resuelve laberintos, utilizando para esto procesamiento digital de imágenes, antes de dar comienzo con los movimientos se tiene ya la ruta trazada de todo el mapa [6]. V. Aggarwalet al. en el 2013 presentan dos algoritmos un primero Floodfill y un segundo Floodfill con vista adelante, aplicados a un robot móvil tipo mouse, en donde se hacen las comparaciones entre los dos algoritmos [7].

Año: 2020 Volumen: 1
Número 6, ISSN: 2448-623X

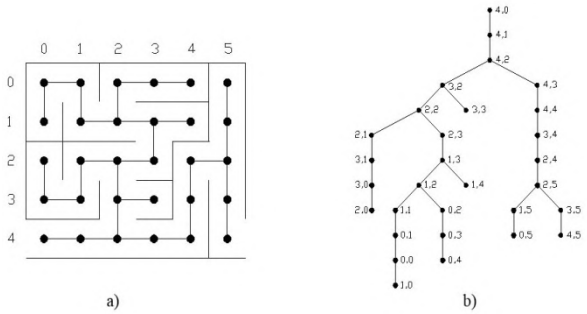


Figura. 2.1. Teoría de grafos para la solución de un laberinto.

El poder determinar cuál es el camino más corto, para un automóvil o para un robot móvil, tiene mucha importancia en la mayoría de las aplicaciones; Ya que implica llegar en menos tiempo al objetivo y no necesariamente, pero si en una gran parte de los casos, con un gasto menor de la energía consumida por el sistema.

La solución de laberintos es un problema común en el campo de los sistemas computacionales. Debido a que existen en la literatura el diseño de muy diversos algoritmos de solución [2]-[7]. Sin embargo, en la mayoría de los casos solo se encuentra la solución del laberinto, no se llega a la implementación. En este sistema presentado, la solución tiene una aplicación; se envían los resultados traducidos en instrucciones a un robot móvil con ruedas para que este las ejecute y así pueda salir del laberinto.

En la segunda parte se hablará sobre el algoritmo usado para la solución del laberinto. En la tercera se muestran las especificaciones del laberinto y del robot, construidos para este proyecto. En la cuarta sección se mencionan los detalles de la comunicación implementada entre Matlab y el robot y en la quinta se muestran los resultados. En la sección seis se dan las conclusiones.

II. Algoritmo Solucionador de Laberintos

La forma tradicional de resolver un laberinto y que se usa desde que éramos niños es mediante prueba y error. Simplemente se avanza hasta donde se puede y si no hay más camino que seguir, se retorna hasta un camino que no haya sido explorado.

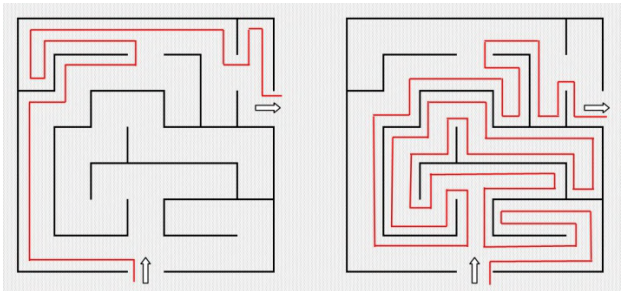


Figura. 2.2. Regla de la mano izquierda y derecha para un laberinto.

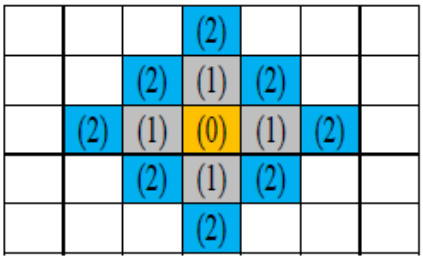


Figura. 2.3. Algoritmo Floodfill.

Actualmente existen muchos algoritmos para encontrar la solución de un laberinto. Algunos de estos un poco complicados, basados en la teoría de grafos, que consiste en tratar al laberinto como una serie de nodos conectados entre sí, ver Figura 2.1. Otros algoritmos son muy simples, como el de la regla de la mano derecha o izquierda, Figura 2.2, donde simplemente se avanza siguiendo una pared del laberinto (izquierda o derecha) hasta salir del laberinto.

En la teoría de grafos existen varios algoritmos que pueden resolver un laberinto tales como [8]:

- Breadth First Search (BFS)
- Depth Fisrt Search (DFS)
- Dijkstra
- A*

El algoritmo usado en este artículo es el llamado “Floodfill”, o algoritmo de relleno por difusión. Es un algoritmo que determina el área formada por elementos contiguos en una matriz multidimensional. Se usa en la herramienta “Bote de pintura” de programas de dibujo y en juegos como el Buscaminas.

Aplicado al laberinto este algoritmo ayuda a asignarle un costo a cada espacio del laberinto. Este costo es definido por “cuantos pasos se tienen que dar de la casilla actual para llegar al final del laberinto”. La Figura. 2.3. muestra el funcionamiento del algoritmo. Donde se puede ver como a los elementos que rodean al centro se les asigna un 1 porque cuesta un paso llegar de ellos al centro, a los alrededores de 1 se les asigna un 2 y así sucesivamente.

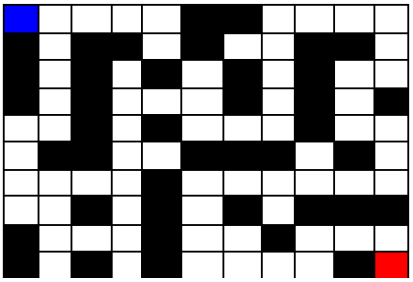


Figura. 2.4. Ejemplo del tipo de laberintos resueltos en este artículo.

Año: 2020 Volumen: 1
Número 6, ISSN: 2448-623X

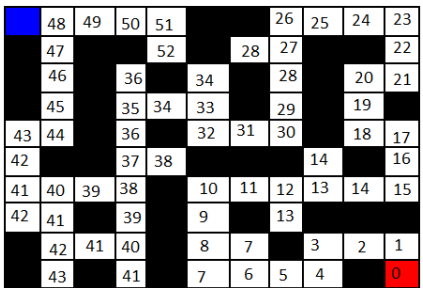
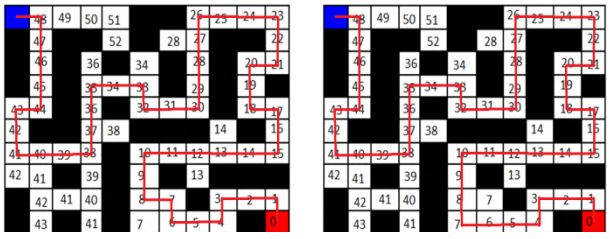


Figura. 2.5. Algoritmo Floodfill aplicado a la Figura 2.4.

El tipo de laberintos que se va a resolver son como los de la Figura 2.4. Donde la casilla azul indica el inicio y la roja el fin. El robot tiene permitido trasladarse por las casillas blancas y tiene prohibido las casillas negras. Y aplicando el algoritmo Floodfill a la Figura 2.4 se tiene como resultado la Figura 2.5. Una vez que se ha aplicado y se han "etiquetado" a todas las casillas del laberinto resta encontrar el camino más corto para llegar a la salida. Para esto se posiciona en la casilla de inicio (azul) y se pregunta: ¿Cuál de los vecinos tiene un menor costo para llegar a la salida?, de lo observado, se mueve a la casilla de menor costo y se hace otra vez la misma pregunta, recordando que solo es posible el movimiento sobre las casillas blancas. Así sucesivamente hasta que se encuentra la salida.

Algo importante es destacar que habrá situaciones en las que el costo de dos o más vecinos sea el mismo, es decir que costaría lo mismo moverse a una casilla o a otra, por lo que se podría decir que se tiene más de una solución para el camino más corto. Una forma de solucionar este problema sería escoger la casilla que provocaría en el robot un movimiento recto, ya que el hacer girar el robot cuesta más tiempo que simplemente moverse hacia adelante, ver Figura. 2.6a. En el caso de este proyecto no se realizó esta convención. El algoritmo aplicado elegiría cualquiera de las casillas con el mismo costo siguiendo la siguiente prioridad: arriba, derecha, abajo, izquierda, ver Figura 2.6b.

La siguiente Figura 2.6 muestra las dos posibles soluciones discutidas anteriormente.



a) b)

Figura. 2.6. a) Solución óptima, b) Solución de nuestro algoritmo.

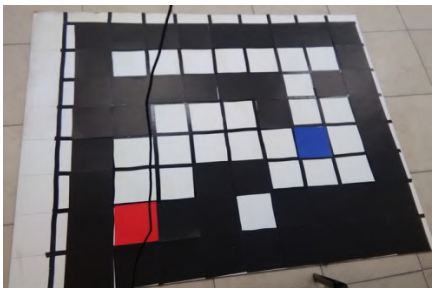


Figura. 3.1. Laberinto real usado para la realización de pruebas.

III. DISEÑO DEL LABERINTO Y DEL ROBOT

A. El laberinto

El laberinto real construido para las pruebas del algoritmo es muy parecido al de la Figura 2.4, salvo algunas diferencias, observar en la Figura. 3.1, el real.

A continuación, se muestran algunas especificaciones

- El laberinto mide 135cmx120cm.
- Es de dimensión 9x8 cuadros.
- Cada cuadro tiene como medidas 15cmx15cm.
- El cuadro azul es el cuadro donde inicia el robot.
- El robot puede trasladarse sobre los cuadros blancos.
- Los cuadros negros son inaccesibles para el robot.
- El cuadro rojo indica la salida del laberinto.

El suelo está construido sobre lámina de madera, donde se ha puesto una rejilla de cuadros vacíos. Con el fin de que el laberinto pueda modificarse se cuenta con cuadros negros removibles, usados para rellenar y darle la forma deseada.

B. El robot

El robot usado para la solución del laberinto es un robot móvil con dos ruedas actuadas y dos llantas de giro libre, llamadas ruedas locas, ver Figura 3.2. Algunas de las especificaciones del robot se mencionan a continuación:

- Cuenta con un microcontrolador Arduino Nano.
- Dos motoreductores de CD de 5V.
- Módulo de puente H L298.
- Modulo Bluetooth HC-05.
- Giroscopio MPU 6050.
- Sensor infrarrojo.
- Batería de 5V.

Este robot fue diseñado para tener un tamaño adecuado como para caber dentro de los cuadros del laberinto y poder girar sobre ellos. El puente H ayuda para los cambios de dirección del robot, ya que este logra que las llantas puedan girar en ambos sentidos. El modulo Bluetooth es el encargado de entablar comunicación entre Matlab y el robot.

Año: 2020 Volumen: 1
Número 6, ISSN: 2448-623X

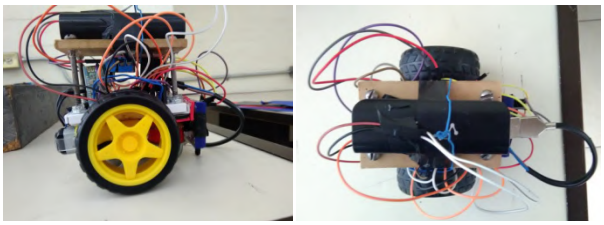


Figura 3.2. El robot móvil implementado.

El giroscopio tuvo que ser agregado, ya que en las primeras pruebas que se realizaron, el robot no avanzaba en línea recta, sino que se desviaba poco a poco. Con el giroscopio se resolvió este problema usando un control tipo proporcional, y que también es requerido en el momento de girar el robot un ángulo de 90 grados, Figura 3.3.

El sensor infrarrojo tiene la función de indicarle al robot cuando ya paso de un cuadro a otro, ver Figura. 3.3. Por último, la batería, que se encuentra en la parte superior del robot, se encarga de alimentar todos los circuitos anteriores.

C. Cámara

La cámara que se usó fue la Logitech Webcam HD Pro C920 que cuenta con una resolución máxima de 1920x1080px, ver Figura. 3.4. Para esta aplicación se usó una resolución mucho más baja: 320x240px. Esto con el fin de que no se tengan que procesar tantos datos y se mejore la velocidad con la que funciona el programa. Además de que no es necesaria una gran resolución de la imagen para aplicar el algoritmo de solución. La cámara se encuentra ubicada, aproximadamente a 1.8 metros, por encima del laberinto.

IV. Comunicación entre Matlab y Robot

Algo importante por mencionar es el cómo se le manda la solución desde Matlab hacia el robot: El algoritmo de solución devuelve como resultado los movimientos que se tienen que dar desde la casilla de inicio para llegar a la solución. Estos movimientos son dados de la forma arriba, abajo, izquierda, derecha. Hay que aclarar que los movimientos son considerados respecto al marco de referencia fijo de la cámara.

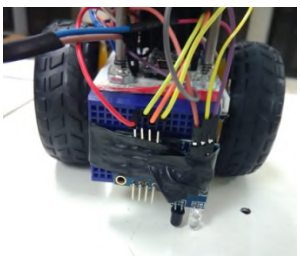


Figura. 3.3. Giroscopio y sensor infrarrojo en el robot.

Figura 3.4. Logitech Webcam HD Pro C920

Por ejemplo, la solución al laberinto de la Figura 3.1, es dada por la siguiente serie de movimientos: izquierda, izquierda, izquierda, izquierda, abajo, izquierda, abajo. Estos movimientos, son codificados y son transmitidos al arduino, vía Bluetooth, como caracteres ASCII. Siendo 'U' para el movimiento hacia arriba, 'D' hacia abajo, 'L' hacia izquierda y 'R' hacia derecha.

Es importante mencionar que el marco de referencia del robot no se encuentra fijo como el de la cámara. Así que, en ocasiones un comando izquierda puede significar cosas diferentes para el robot, como por ejemplo: puede suceder que el carro se mueva de frente ante una instrucción de izquierda, esto ocurre si es que este ya se encuentra orientado a la izquierda, o puede significar girar si se encuentra en la dirección arriba. Entonces para solucionar esto; lo que se hace es preguntar por el movimiento presente y por el movimiento siguiente. Por ejemplo, si el movimiento presente es derecha y el movimiento siguiente es también derecha, el robot resuelve que esto es moverse en línea recta. Otro ejemplo sería que el movimiento presente es derecha y el movimiento siguiente es arriba. Lo que el robot lo interpreta como girar hacia la izquierda.

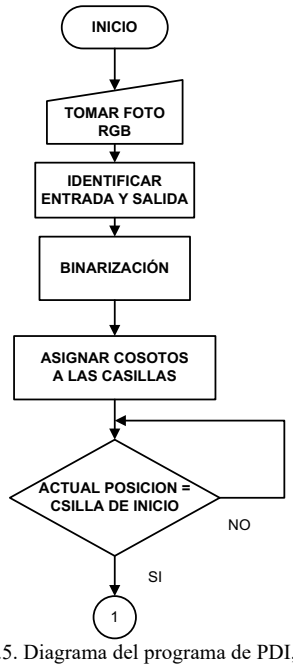


Figura. 3.5. Diagrama del programa de PDI, parte 1.

Año: 2020 Volumen: 1
Número 6, ISSN: 2448-623X

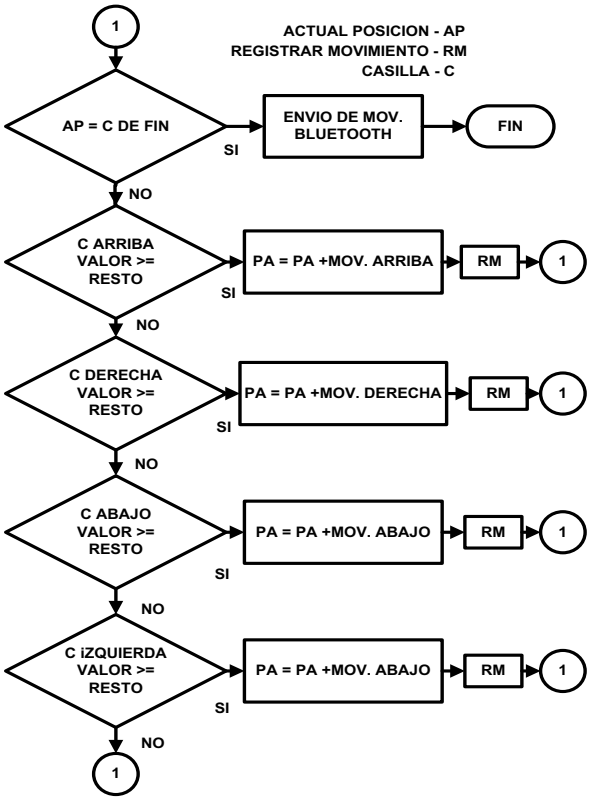


Figura. 3.6. Diagrama del programa de PDI, parte 2.

Un diagrama de flujo del programa implementado en el Matlab es mostrado en la Figura 5 y Figura 6. Este programa primero realiza un PDI para determinar la ubicación de casillas blancas y negras en el Laberinto, siendo las blancas por donde puede moverse el robot.
Después se aplica el algoritmo Floodfill asignando costos a estas casillas. En la segunda sección del programa, el algoritmo comienza a generar los movimientos para el robot haciendo la toma de decisiones en base a los costos asignado, la posición y orientación del robot móvil. El robot móvil se detiene al llegar a la casilla roja, que es el objetivo.

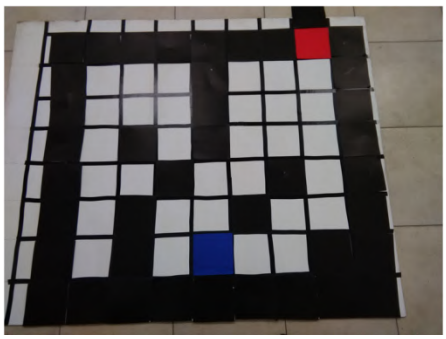


Figura. 5.1. Laberinto a resolver del ejemplo 1.

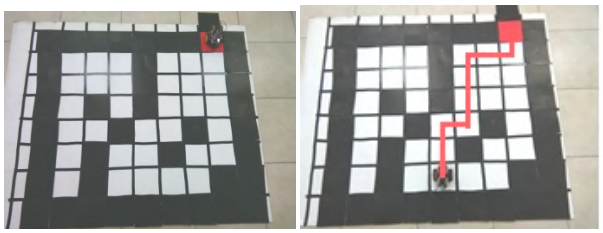


Figura. 5.2. Solución al laberinto del ejemplo1 ejecutada por el robot.

V. Resultados

Como parte de los resultados se muestran dos ejemplos de laberintos resueltos por el sistema implementado, con el fin de ejemplificar como es que el que el algoritmo y los movimientos del robot funcionan.

Ejemplo 1

Se resolverá el laberinto mostrado en la Figura 5.1.

El algoritmo implementado en Matlab proporciona como resultado, los siguientes movimientos: arriba, arriba, derecha, arriba, arriba, arriba, derecha, derecha, arriba. Como solución el robot se movió en el sentido y dirección que se muestra en la Figura 5.2. Algo que hay que destacar es que en el ejemplo 1 existen más de una solución y el algoritmo implantado resolvió la del camino más corto.

Ejemplo 2

Se resolverá el laberinto mostrado en la Figura 5.3

El algoritmo implementado en Matlab proporciona como resultado, los siguientes movimientos: arriba, arriba, arriba, derecha, derecha, abajo, abajo.Como solución el robot sigue la trayectoria de la Figura 5.4. En el ejemplo 2 solo existe una única soluciónal camino más corto y el algoritmo utilizado resolvió utilizar esta. Los movimientos del robot resultan ser algo lentos debido a la forma en que fue diseñado. Además de estos dos ejemplos se realizaron más pruebas donde los resultados también fueron correctos comportándose el sistema de la manera deseada.

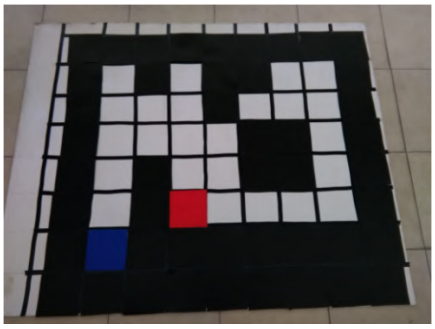


Figura. 5.3. Laberinto a resolver del ejemplo 2.

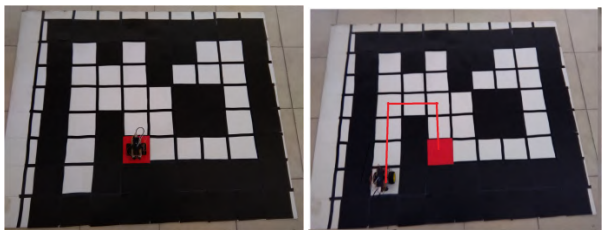


Figura. 5.4. Solución al laberinto del ejemplo 2 ejecutada por el robot

El análisis que llevó a la implementación de teoría de grafos en robots móviles fue satisfactorio, dado que los resultados fueron óptimos y a su vez dando apertura a nuevas variables que no se tenían previstas desde el inicio. A partir del conjunto de nodos, vértices o puntos, además del conjunto de aristas y líneas se logró desarrollar el modelado de un laberinto controlado para la obtención de la ruta más corta con propósito de dirigir el robot de un punto a otro. Este análisis se vuelve de suma importancia, por la oportunidad de analizar este sistema aplicando técnicas de procesamiento digital de imágenes y reconocimiento de patrones, proyectando una idea para el desarrollo de automóviles autómatas e incluso el mapeo de cualquier superficie, para llevar al punto más óptimo de la orientación de un objeto siendo autónomo.

VI. Conclusiones

En el presente trabajo se llevó a cabo la aplicación de un sistema que tenía como objetivo resaltar dos aspectos: por un lado la utilidad de sensores externos de visión para la obtención de información útil para robots móviles, en este caso encontrar la solución de un laberinto, que puede traducirse en identificar el camino más óptimo hacia un objetivo evadiendo obstáculos, mientras que por otro lado el uso de algoritmos de teoría de grafos como una alternativa para la definición de trayectorias en robots móviles.

Los resultados obtenidos fueron satisfactorios en los que siempre se encontraba el camino más corto a la salida del laberinto en cuanto a número de movimientos y el robot llegaba a su objetivo, con excepción en casos muy particulares, como lo son situaciones con poca luminosidad, donde la información obtenida por la cámara presenta errores que afectan el funcionamiento correcto del sistema dando soluciones erróneas al laberinto.

Otro detalle a resaltar es que en este artículo solo se trabajó con uno de los algoritmos de la teoría de grafos, lo que deja lugar a otras investigaciones donde se utilicen más algoritmos y se pueda hacer una comparativas entre estos e inclusive añadiendo más variables como lo es el tiempo en el que el robot sales del laberinto.

En cuanto al diseño del robot, no estuvo exento de problemas, en varias ocasiones hubo que rediseñarlo para que su tamaño fuera el adecuado al laberinto o para que sus movimientos fueran más precisos, con la adición del giroscopio.

VII. Referencias

[1] A. Ollero (2001), *Robótica: Manipuladores y robots móviles*, Barcelona España, Ed. MARCOMBO, p. 1

[2] Kovacs et al (2019), "Autonomous Line Maze Solver Using Artificial Intelligence", *International Conference on Engineering of Modern Electric Systems (EMES),* Oradea, Romania, pp 133-136.

[3] T. Ribeiro et al. (2019), "Q-Learning for Autonomous Mobile Robot Obstacle Avoidance", *International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp 1-7.

[4] I. Iturrate et al. (2013), "A Mobile Robot Platform for Open Learning based on Serious Games and Remote Laboratories", *1st International Conference of the Portuguese Society for Engineering Education (CISPEE),* pp. 1-7.

[5] B. Rahnama et al. (2013), "Design and Implementation of Cooperative Labyrinth Discovery Algorithms in Multi-Agent Environment", *The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE),* pp. 573-578.

[6] B. Rahnama et al. (2012), "An Image Processing Approach to Solve Labyrinth Discovery Robotics Problem", *6th International Conference on Computer Software and Applications Workshops,* pp. 631- 636.

[7] V. Aggarwal et al. (2013), "OPTIMIZATION OF FLOOD FILL ALGORITHM USING ITERATIVE LOOK-AHEAD AND DIRECTIONAL TECHNIQUE", *International Journal of Computer Science and Engineering (IJCSE),* IASET, Vol. 2, Issue 5, Nov 2013, 89-94.

[8] A. Mohammad (2017), "Intelligent maze solving robot based on image processing and graph theory algorithms*", International Conference on Promising Electronic Technologies*, pp. 48-53, DOI: 10.1109.

VIII. Biografía

Santillán Ávila Juan Luis: Nacido en Gómez Palacio Durango el 27 de Noviembre de 1997, Preparatoria (Técnico en Electrónica) CBTIS 4 C.D Lerdo Durango del 2012 al 2015. Licenciatura (Ingeniería Electrónica) Instituto Tecnológico de la Laguna Torreón Coahuila del 2016 al 2019. El actualmente estudia una maestría en ingeniería eléctrica en el área de mecatrónica y control en el instituto tecnológico de la laguna que se encuentra en el estado de Coahuila en la ciudad de Torreón. Realizó sus prácticas profesionales en la empresa Industria SIGRAMA como Ingeniero en ensamble.

Márquez Acosta Jesús Emanuel: Nacido en Torreón, Coahuila, el 15 de Mayo de 1998. Obtuvo el grado licenciatura en Ingeniería electrónica con especialidad en mecatrónica y control en el Instituto Tecnológico de la Laguna en Torreón, Coahuila, México en el año 2019.Actualmente estudia una maestría en ciencias en ingeniería eléctrica en el área de mecatrónica y control en el Instituto Tecnológico de la Laguna en Torreón, Coahuila, México.

Fematt Hernandez Hiram Brahim: Nacido en Anaheim California Orange Country, USA el 18 de Julio de 1997. Preparatoria (Técnico en

Año: 2020 Volumen: 1
Número 6, ISSN: 2448-623X

Electrónica) CBTIS 4, Cd. Lerdo, Durango del 2012 al 2015. Licenciatura (Ingeniería Electrónica) Instituto Tecnológico de la Laguna Torreón Coahuila del 2015 al 2019. Realizó sus prácticas profesionales en la empresa Automatización Programación y Control como Ingeniero en diseño y programación de sistemas de control para empresas 4.0.

Sifuentes Mijares, Juan: Nació en Gómez Palacio Durango, México el 5 de Agosto de 1968. Es Ingeniero Industrial en Electrónica por el Instituto Tecnológico de la Laguna. Es Maestro en Ciencias de la Ingeniería Eléctrica con especialidad en Control Automático por el Instituto Tecnológico de la Laguna. También es Doctor en Ciencias de la Ingeniería Eléctrica con especialidad en Control Automático por el Instituto Tecnológico de la Laguna. Estuvo en Valencia España realizando estudios de doctorado en Diseño de Sistemas Digitales.

Ha trabajado como profesor, en diferentes escuelas, tales como: La Universidad Iberoamericana IBERO, El Tecnológico Superior de Lerdo, la Universidad Autónoma del Noreste UANE y actualmente se desempeña como profesor investigador del Instituto Tecnológico de la Laguna.

El Dr. Sifuentes-Mijares recibió el premio al mejor artículo de su área, durante el congreso Internacional World Automation Congress, celebrado en la Gran Isla de Hawaii en el 2014.