PROCESAMIENTO DE IMÁGENES PARA LA PLANEACIÓN DE RUTAS PARA ROBOTS MÓVILES BASADOS EN LEGO MINDSTORMS NXT

IMAGE POCESSING FOR MOBILE ROBOTS PATH PLANNING BASED ON LEGO MINDSTORMS NXT

RESUMEN

El presente artículo contiene el procedimiento para implementar diferentes técnicas de procesamiento de imágenes y principalmente el algoritmo de esqueletización, con el objetivo de planificar rutas libres de obstáculos para robots móviles mediante una fotografía tomada sobre un área determinada con los obstáculos y el robot (construido con el kit de robótica Lego Mindstorms NXT). Se realiza el debido procesamiento a dicha imagen, para luego, de forma remota, enviar una serie de órdenes vía bluetooth desde un computador usando MATLABm las cuales son generadas por el algoritmo aplicado sobre la imagen ya procesada, con el fin de obtener el desplazamiento de la plataforma móvil a un punto de llegada determinado por el usuario.

Palabras clave: binarización, enrutamiento, esqueletización, imagen, odometría, robot.

ABSTRACT

This article contains the procedure to implement different image processing technics, mainly skeletonization algorithm in order to plan obstacle-free routes for mobile robots, using a photograph taken over an area with obstacles and the robot (built with LEGO Mindstorms NXT robotics kit). Processing is performed to that image, to send remotely a series of commands via bluetooth, from a computer using MATLAB; the orders are generated by the algorithm applied to the image already processed, in order to obtain the displacement of the mobile platform to an end point specified by the user.

Keywords: binarization, image, odometry, robot, routing, skeletonization.

Jesús Alberto Torres Velásquez
Estudiante de la Universidad Distrital Francisco José de Caldas
jesatorresv@correo.udistrital.edu.co
Bogotá, Colombia

Rafael Ricardo Viáfara Ávila
Estudiante de la Universidad Distrital Francisco José de Caldas
rrviafaraa@correo.udistrital.edu.co
Bogotá, Colombia

Fernando Martínez Santa
Magister en Ingeniería Electrónica y de Computadores
Docente de la Universidad Distrital Francisco José de Caldas
fmartinezs@udistrital.edu.co
Bogotá, Colombia

Tipo: Artículo de investigación

Fecha de Recepción: Noviembre 29 de 2013
Fecha de Aceptación: Noviembre 7 de 2014

1. INTRODUCCIÓN

Uno de los problemas actuales de la robótica es el desplazamiento autónomo de robots móviles en ambientes desconocidos, en específico la planeación de rutas ante la presencia de diversos obstáculos fijos. Lo anterior ha sido solucionado de diferentes maneras, como por ejemplo: el método de campos de potencial artificial [1], [2], diagramas de Voronoi [3], [4] y evasión de obstáculos por ultrasonido [5], entre otros. Existen básicamente dos formas de solucionar el problema, la primera consiste en que el robot móvil posea inteligencia (control) y sensórica abordo [5], [6]; la segunda consiste en que el control y/o la sensórica sean externos al robot [7] [8]. El segundo enfoque de solución usualmente utiliza una cámara en conjunto con una computadora, en donde la cámara cumple la función del sensor; por lo tanto, este enfoque requiere algún grado de procesamiento de imágenes. En lo que concierne a la planeación de rutas libres de obstáculos para robots móviles con inteligencia y/o sensórica externa, se han realizado muchas investigaciones, pero gran parte de ellas se han quedado en diseños y/o simulaciones y no se han implementado físicamente, puesto que es necesario tener en cuenta factores físicos, externos e internos del robot, los cuales implicarían rediseñar los algoritmos existentes [9], [10].

En nuestro caso se escoge la opción de recurrir a los diagramas de Voronoi por medio del método de la esqueletización, debido a que este no ha sido implementado en la planeación de rutas para robots móviles construidos con el kit de robótica Lego Mindstorms NXT [11], [12].

Este kit se compone principalmente de las siguientes características:

- Ladrillo Inteligente NXT con un microprocesador de 32-bit.
- 3 servomotores interactivos
- 1 x sensor visual ultrasónico
- 2 x sensor de presión mejorado
- 6 pilas 1,5 V AAA
- Fichas de construcción en general

2. ENTORNO

El área se determina en función de la altura de la cámara (2,4 m), ésta es: 1,7 x 1,2 m; la superficie donde se desplaza el robot es de color blanco, cuyo material es vinilo calibre 13. Los obstáculos poseen diversas formas y colores con el fin de exponer la eficacia del algoritmo de implementación ante diversos tipos de obstáculos. En este caso, los obstáculos tienen una altura de 4 mm, pero ésta no interfiere en su adecuada detección.

3. CONFIGURACIÓN DEL ROBOT

El robot está configurado como una plataforma diferencial (figura 1), con las siguientes características:

Ruedas de giro libre: 1.
Radio de las ruedas motrices: 2 cm.
Distancia entre ruedas: 13 cm.
Distancia entre ejes: 14 cm.
Dimensiones: altura 15 cm, largo 16 cm, ancho 15 cm.
Peso: aprox. 0,8 kg.

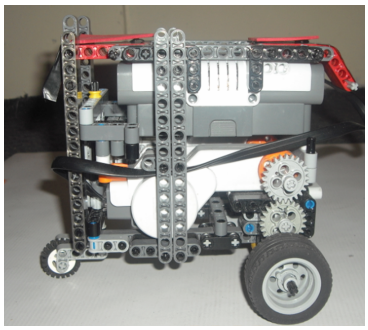

Figura 1. Configuración del robot

La configuración de las ruedas por facilidad en el desplazamiento es de tipo diferencial. Consiste en dos ruedas colocadas en el eje perpendicular a la dirección del robot. La figura 1 muestra este tipo de configuración. Cada rueda es controlada por un motor, de tal manera que el giro del robot depende de la diferencia de velocidad de las ruedas. Uno de los problemas a considerar es el equilibrio del robot, por lo que es conveniente agregar ruedas de libre giro. El problema que se puede dar en el caso

de presentarse más de 3 apoyos es la pérdida de tracción y errores en los cálculos de odometría (ante superficies irregulares), este tipo de configuración es apta para desplazarse en superficies planas *únicamente*.

Se elige esta configuración debido a que es posible con los motores que manejan tracción controlar dirección y requiere menos espacio para realizar un giro sobre su eje.

4. METODOLOGÍA

4.1. Descripción general del proyecto

La figura 2 representa los pasos que se siguieron para la implementación del algoritmo.

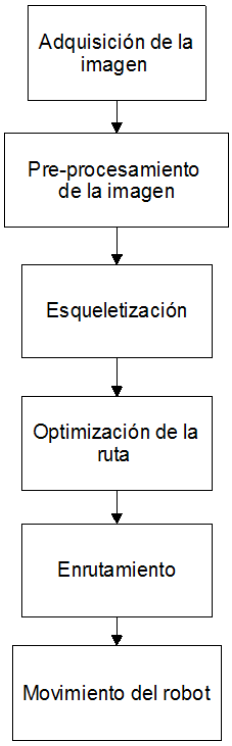

Figura 2. Diagrama general del proyecto

4.1. Algoritmo de preprocesamiento

Antes de realizar el procesamiento de la imagen, es necesario tomar una fotografía del área a tratar. Este proceso se ejecuta para especificar el formato y tamaño de la imagen (figura 3).

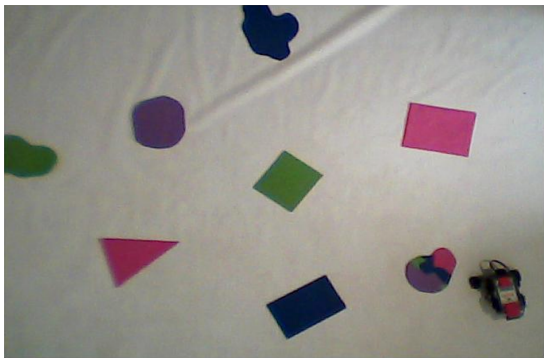

Figura 3. Adquisición de la imagen

Debido a que los objetos deben quedar claramente definidos para la posterior generación de la ruta de desplazamiento, es necesario realizar un tratamiento previo de la imagen. A continuación se describen los pasos (figura 4).

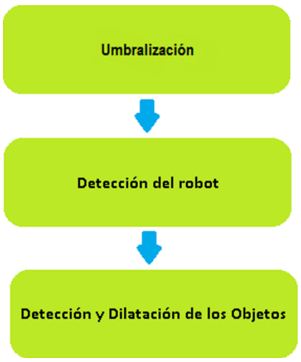

Figura 4. Tratamiento de la imagen

4.1.1. Umbralización

El proceso de binarización consiste en transformar una imagen RGB (en nuestro caso) a blanco y negro. El formato original de la imagen es RGB, esto es, tres dimensiones (red, green, blue), luego es necesario primero convertirla en escala de grises (una sola matriz) y finalmente pasarla a blanco y negro.

Una imagen binaria es una imagen en la cual cada píxel puede tener sólo uno de dos valores posibles, 1 o 0. Como es lógico suponer, una imagen en esas condiciones es mucho más fácil encontrar y distinguir características estructurales.

En visión computacional, el trabajo con imágenes binarias es muy importante ya sea para realizar segmentación por intensidad de la imagen, para generar algoritmos de reconstrucción o para reconocer estructuras [13].

La forma más común de generar imágenes binarias es mediante la utilización del valor umbral (ecuación (1)) de una imagen a escala de grises; es decir, se elige un valor límite (o bien un intervalo) a partir del cual todos los valores de intensidades mayores serán codificados como unos mientras que los que estén por debajo serán codificados a ceros (figura 5).

$$q(x,y) = \begin{cases} 255 & f(x,y) > k \\ 0 & para\ otro\ caso \end{cases} \quad (1)$$

q(x,y)=Imagen Salida en el punto (x,y)
p(x,y)=Imagen Entrada en el punto (x,y)

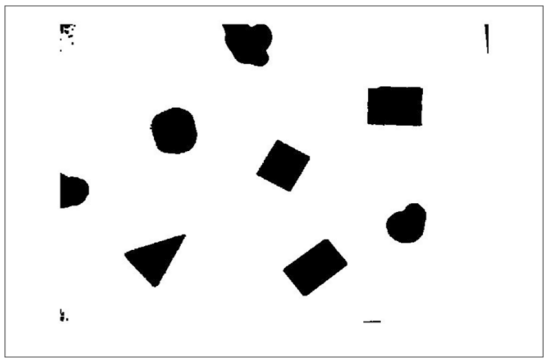

Figura 5. Imagen binarizada

A continuación se describen los pasos para lograr una correcta binarización de la imagen en la que se aprecie la distinción de objetos y fondo (figura 6).

- La imagen debe convertirse a escala de grises.
- Se convierte a blanco y negro, graduando el umbral de la binarización, de esta manera se eliminan sombras que se pueden presentar debido a la iluminación del ambiente.
- Se erosiona la imagen con el objetivo de eliminar ruido, éste utiliza un objeto estructurante el cual se convoluciona en toda la imagen.

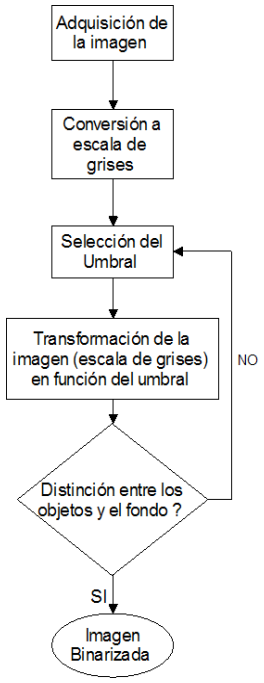

Figura 6. Proceso para binarizar la imagen

4.1.2. Detección del robot

Se realiza con el objetivo de eliminar el robot de la imagen y encontrar sus coordenadas, lo cual se lleva a cabo mediante los siguientes pasos (figura 7).

- Se ubican en la parte delantera y trasera del robot dos conjuntos de fichas rojas de diferentes áreas; esto se realiza para distinguir la parte delantera, debido a que el conjunto de fichas allí tiene un área mayor.
- Se utiliza una función de MATLAB que permite extraer los pixeles de color similares al rojo de la imagen que contienen el robot y los obstáculos en una nueva imagen binaria.
- Se etiquetan los objetos de la nueva imagen, posteriormente se calculan los centroides y se guardan (array B); y las dimensiones de los cuadrados que encierran los objetos (array A).
- A continuación se calculan las distancias entre los centroides, luego se determina la menor, la cual corresponde a la distancia entre la parte delantera y trasera del robot. Con la menor distancia se buscan en

el array B los centroides correspondientes entre los dos objetos más cercanos, y a partir de ellos se busca en el array A las áreas correspondientes a la parte trasera y delantera del robot.

- Posteriormente se calcula el centro del robot, luego se crea una imagen de color negro con un punto blanco, el cual corresponde al centro del robot, este punto será dilatado en función del tamaño del robot. Luego se convoluciona un objeto estructurante sobre la imagen, dilatando los pixeles que se encuentren dentro del objeto; el resultado resultado es una imagen de color negro con un polígono blanco (figura 8).
- Finalmente se efectúa la operación lógica OR entre la imagen binarizada y la imagen obtenida anteriormente, con el fin de eliminar el robot de la imagen.

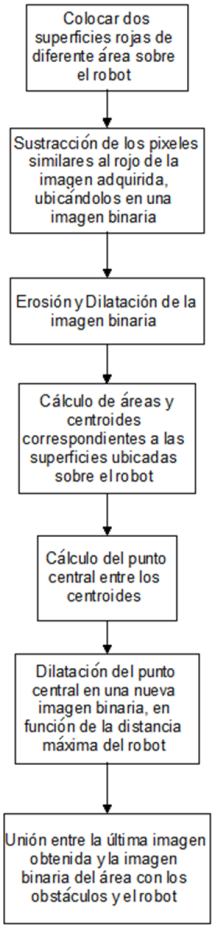

Figura 7. Proceso para detectar la ubicación del robot

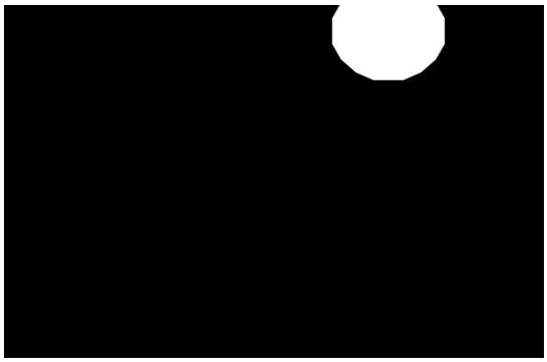

Figura 8. Detección del robot

4.1.3. Dilatación de los obstáculos

Se realiza con el objetivo de tener en cuenta el tamaño del robot al desplazarse a través de los objetos, para ello se realizaron los siguientes pasos (figura 9).

Dados dos conjuntos de píxeles A y B en Z^3 y ∅ definido como vacío, se dice que la dilatación de A con respecto a B es (ecuación (2)).

$$A \oplus B = \{x \,|\, (B)_x \cap A \neq \phi\} \qquad (2)$$

Donde A es la imagen original y B se denomina máscara de convolución, que para el caso de una imagen volumétrica se trata de un cubo [14].

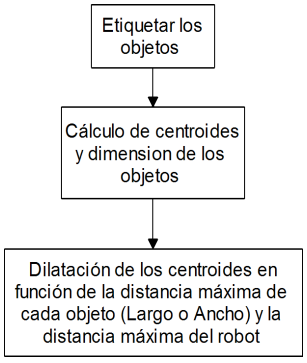

Figura 9. Dilatación de los obstáculos.

- Se etiquetan los objetos presentes en la imagen y se calculan los centroides.
- Se encasillan los objetos dentro de rectángulos para calcular su área, y en función de esta dilatarlos, teniendo en cuenta las dimensiones del robot (figura 10).

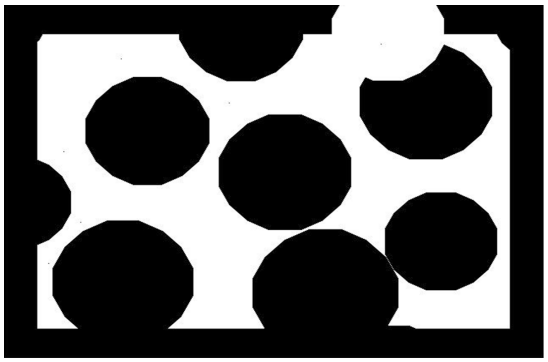

Figura 10. Dilatación de los obstáculos en función de sus dimensiones

5. ESQUELETIZACIÓN

El algoritmo de esqueletización de imágenes obtiene el esqueleto de los objetos en una imagen donde dichos objetos deben ser blancos en un fondo negro. Se puede definir como esqueleto a una línea con puntos medios desde los bordes de los objetos. La figura 11, corresponde a la imagen original, y la figura 12, el esqueleto de la imagen.

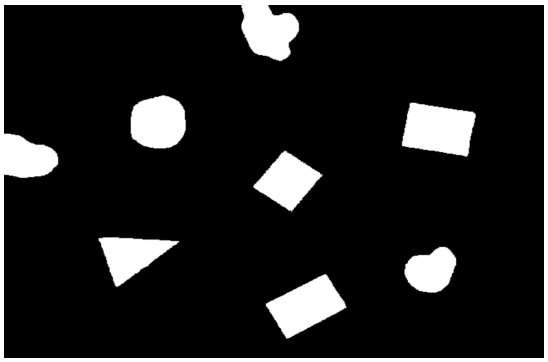

Figura 11. Imagen binarizada del escenario.

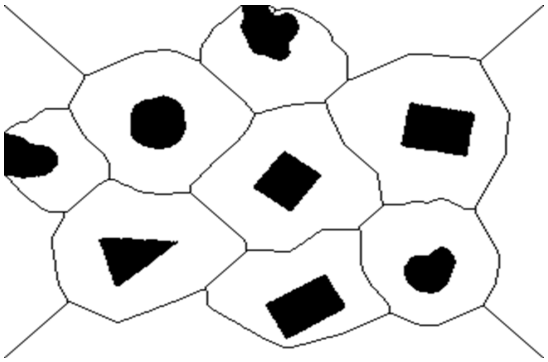

Figura 12. Imagen del esqueleto del escenario y los obstáculos

En este proceso es necesario tener en cuenta varios factores para el procesamiento del algoritmo: complejidad del algoritmo, tamaño de la imagen, procesador [15].

El procesador debe cumplir con mínimo 3 propiedades para ser útil:

- Lo más delgado posible
- Conectado
- Centrado

De esta manera se crea una única ruta comprendida entre el robot y un punto de llegada arbitrario.

Para llevar a cabo este proceso, la imagen debe estar totalmente binarizada, sin ruido y con objetos definidos, se calcula el esqueleto de la imagen, los puntos de cruce y los puntos finales. La esqueletización se puede lograr a través de una operación morfológica de erosión iterativa, con una matriz kernel de 3x3 pixeles. Dicha erosión se realiza hasta que los bordes resultantes no se pueden erosionar más sin perder los pixeles principales. A partir de la imagen generada con el algoritmo, se eliminan los caminos sin salida y los caminos desde los puntos de cruce hasta los puntos finales innecesarios (todos excepto inicial y final), cuyo resultado es una ruta única desde el punto inicial hasta el punto final, por último se realiza la diezmación a la ruta para convertirla a solo puntos.

6. PLANEACIÓN DE LA RUTA

6.1. Optimización de la ruta

Este proceso consiste en eliminar puntos de la ruta con el objetivo de disminuir la cantidad de órdenes enviadas al robot. Se ejecuta mediante los siguientes pasos (figura 13).

Se calcula una primera pendiente entre el punto inicial y el punto final, luego se contabiliza la cantidad de pixeles de color negro dentro de esa pendiente. Si dicha cantidad es superior a 10, se determina que existe un objeto entre estos dos puntos.

Posteriormente se vuelve a calcular la pendiente con un punto anterior al punto final y se repite el procedimiento anterior de contabilizar la cantidad de pixeles negros, así hasta que esta sea inferior a 10; cuando esto ocurra, significa que no hay obstáculos entre dichos puntos, luego se procede a eliminar los puntos existentes dentro la pendiente anteriormente calculada, y se vuelve a ejecutar el algoritmo con el punto de inicio, en la ubicación del anterior punto final, y el punto final volverá a ser el original.

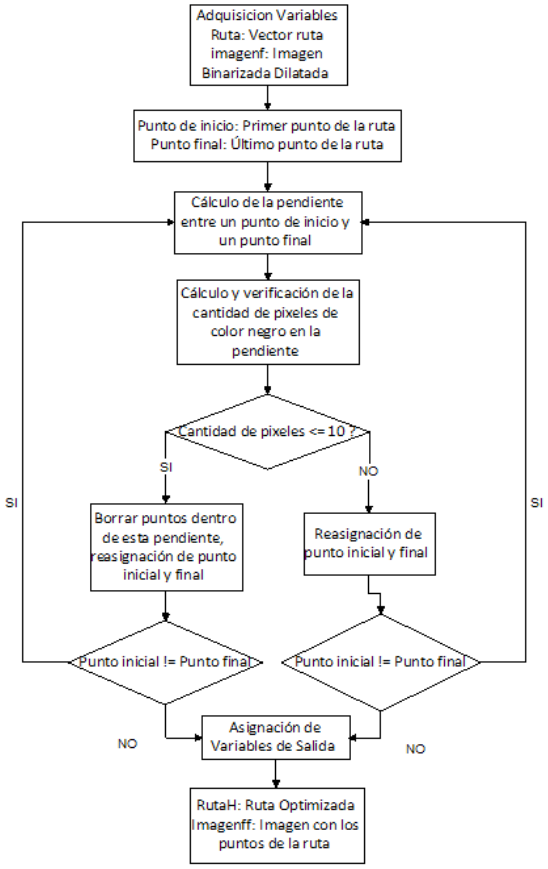

Figura 13. Optimización de la ruta

6.2. Algoritmo de enrutamiento

Se realiza un cálculo de distancias entre los puntos de la ruta de manera consecutiva, cálculo de ángulos, orientación y sentido de giro, mediante los siguientes pasos (figura 14).

- Se calcula la distancia existente entre los puntos de la ruta utilizando (Pitágoras).
- Se determina una ubicación de referencia, para este caso serán dos puntos sobre el robot, la cabeza y la cola, luego se calculan los grados que debe girar el robot respecto al segundo punto de la ruta.
- Teniendo en cuenta que el robot mantendrá el ángulo anterior, cuando éste se desplace a un punto volverá a calcular los grados que debe girar para quedar en línea recta con el próximo destino.
- El proceso anterior se repetirá hasta que el robot llegue al destino final.
- Se guardan las distancias, ángulos y sentidos de giro en vectores para procesarlos posteriormente.

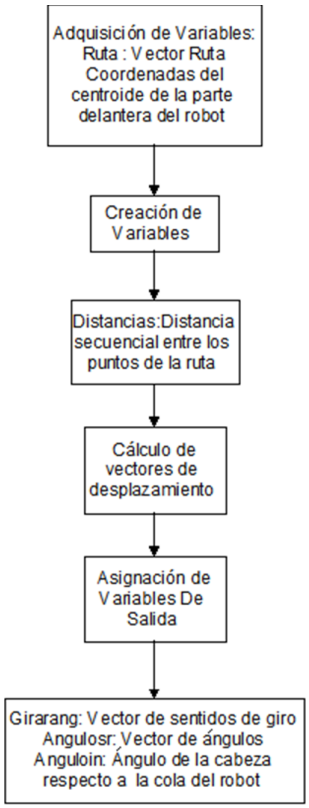

Figura 14. Cálculo de ángulos y distancia

7. MOVIMIENTO DEL ROBOT

Para ejecutar el desplazamiento del robot es necesario tener en cuenta su odometría, que consiste en calcular el desplazamiento del robot con respecto a un punto de referencia, realizando continuamente el cálculo del número de

vueltas por cada llanta. Gracias a la odometría se calculan constantemente vectores de destino y llegada, los cuales nos permiten reducir enormemente el error asociado a la fricción de las llantas con respecto a la superficie.

Para llevar a cabo este proceso es necesario utilizar el *toolbox* llamado RWTHMindstorms-NXT, el cual permite enviar ordenes al robot vía *bluetooth*, en conjunto con MATLAB 2011b. Este procedimiento se realiza mediante los pasos descritos en la figura 15.

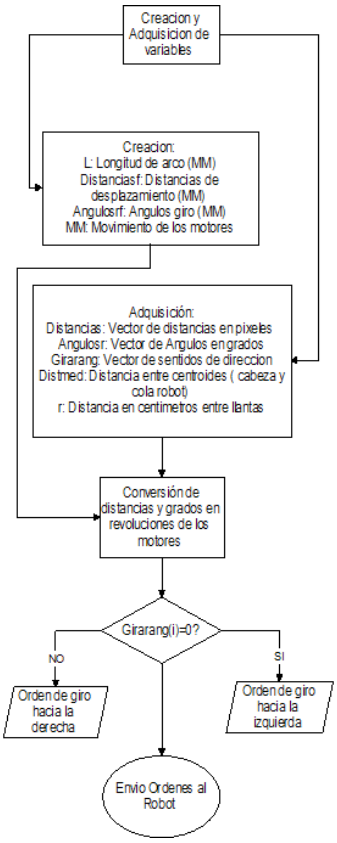

Figura 15. Proceso para ejecutar el movimiento del robot

- Se adquiere el vector de distancias del algoritmo de enrutamiento, se procesa en función del movimiento de los motores del robot, teniendo en cuenta el perímetro de las llantas y la relación existente entre centímetros y pixeles. En la ecuación (3) se relaciona la distancia en pixeles y la distancia en centímetros.

$$Distancia\,(cm) = \frac{9.5 \cdot Distancia\,(px)}{Distmed} \quad (3)$$

Donde:

Distancias (px): vector de distancias entre puntos de la ruta medida en pixeles. *Distmed:* distancia en pixeles entre el centro de la cabeza y la cola del robot. La constante 9,5 es la distancia en centímetros entre el centro de la cabeza y la cola del robot.

Para poder enviar la información a los motores es necesario realizar la conversión de centímetros a grados, mediante la ecuación (4).

$$Ángulo\ motores = \frac{Distancia\,(cm) \cdot 360}{Perimetros\ llantas} \quad (4)$$

- Posteriormente se adquiere el vector de ángulos del algoritmo de enrutamiento, con el objetivo de calcular los grados que el eje de los motores debe girar para realizar el giro de la estructura (robot), para ello se utiliza la herramienta matemática longitud de arco (ecuación (5)).

$$L = \frac{2\pi \cdot r \cdot a}{360} \quad (5)$$

Donde:

L: longitud de arco
r: distancias entre llantas
a: ángulo de desplazamiento

En nuestro caso, la longitud de arco debe ser dividida en 2, debido a que el movimiento se ejecuta con los dos motores, los cuales giran a la misma velocidad y en sentidos opuestos. Luego la longitud de arco es convertida a grados, como se explicó anteriormente para el giro del robot.

Se adquiere el vector de sentidos de giro para determinar hacia dónde debe girar el robot cuando le sea dado un ángulo.

8. OPTIMIZACIÓN DEL MOVIMIENTO

Con el objetivo de reducir el error acumulado y el error final del desplazamiento del robot, se plantearon finalmente cuatro algoritmos de optimización del movimiento.

En las siguientes imágenes, la línea de color negro representa la ruta generada por el algoritmo de optimización de la ruta y la línea de color rojo corresponde a la ruta recorrida por el robot.

8.1. Algoritmo 1: sin corrección

En este algoritmo simplemente el robot se desplaza según los cálculos predeterminados, enviando la ruta de desplazamiento total, los movimientos que debe realizar a medida que se desplaza hasta llegar al punto final. Este algoritmo presenta un error acumulado alto en rutas largas y por consiguiente un mayor desfase entre el punto final de llegada y el punto originalmente planteado (tabla 1 y figura 16).

Tabla 1. Algoritmo sin corrección

Algoritmo 1		
Prueba	Error cuadrático medio [cm]	Error final [cm]
1	5.81	3.65
2	9.80	12.77
3	13.11	8.10
4	6.77	6.92
5	28.45	18.51

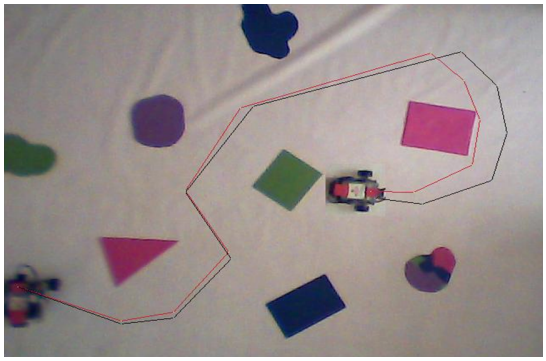

Figura 16. Ruta algoritmo sin corrección

8.2. Algoritmo 2: corrección en el último punto

Se realiza el desplazamiento del algoritmo anterior, pero se toma una nueva fotografía cuando ha culminado su desplazamiento, con la cual se obtiene la ubicación actual del robot; con ésta se calcula el ángulo y distancia necesarios para llegar al punto de destino planteado.

Este algoritmo presenta el mismo error acumulado del algoritmo 1, pero a diferencia de éste, el desfase final es menor (tabla 2 y figura 17).

Tabla 2. Corrección en el último punto

Algoritmo 2		
Prueba	Error cuadrático medio [cm]	Error final [cm]
1	4.15	2.00
2	4.98	1.60
3	15.21	1.41
4	5.51	0.00
5	3.99	0.26

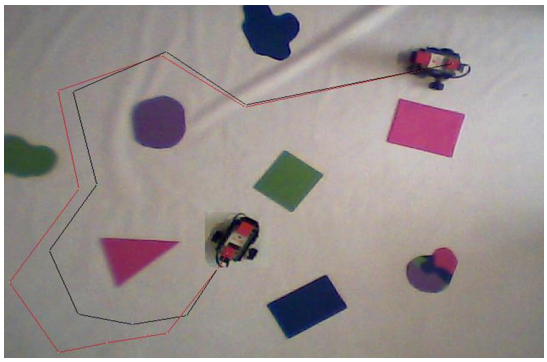

Figura 17. Ruta algoritmo corrección último punto

8.3. Algoritmo 3: corrección en el penúltimo punto

Se realiza el desplazamiento hasta el penúltimo punto, en éste se toma una nueva fotografía, que determina la ubicación actual del robot, el ángulo y distancia necesarios para que se dirija al punto final planteado.

Este algoritmo tiene como ventaja un menor error acumulado, dado que la ruta enviada tie-

ne una orden menos; además, a diferencia de los anteriores algoritmos, permite que el robot no se salga de la imagen (tabla 3 y figura 18).

Tabla 3. Corrección en el penúltimo punto

Algoritmo 3		
Prueba	Error cuadrático medio [cm]	Error final [cm]
1	13.61	4.46
2	22.08	6.12
3	15.23	1.85
4	16.43	2.84
5	4.60	1.53

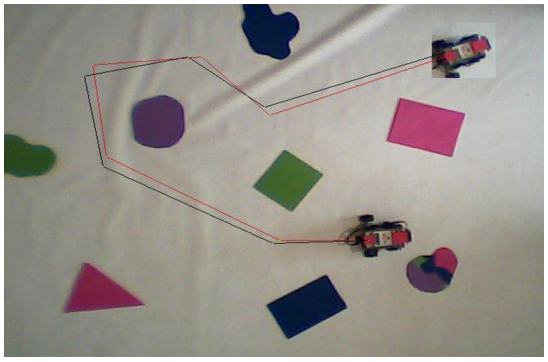

Figura 18. Ruta algoritmo corrección penúltimo punto

8.4. Algoritmo 4: corrección en tiempo real

Tabla 4. Corrección en tiempo real

Algoritmo 4		
Prueba	Error cuadrático medio [cm]	Error final [cm]
1	3.06	1.31
2	1.53	0.00
3	2.60	0.37
4	4.51	1.31
5	1.17	1.89

El desplazamiento se realiza tomando fotografías a medida que el robot avanza, calculando a su vez la ubicación actual del robot, la distancia actual y ángulo, para llegar al siguiente punto de la ruta planteada; esto se ejecuta sucesivamente hasta llegar al punto final de la ruta.

Éste presenta un error acumulado pequeño comparado con los algoritmos anteriores, independiente de la longitud de la ruta y, por lo tanto, resulta un error absoluto sobre el punto final de llegada relativamente pequeño (tabla 4 y figura 19).

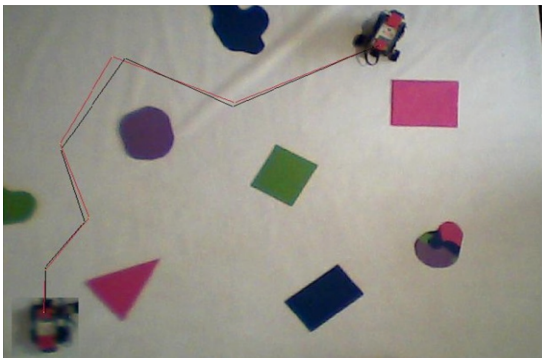

Figura 19. Ruta algoritmo corrección en tiempo real

9. RESULTADOS

En el proceso de la implementación del algoritmo de esqueletización se obtuvieron los siguientes resultados.

Se determinó la importancia de crear una interfaz gráfica que agrupara las funciones que hacen posible la visualización del desplazamiento del robot a través de obstáculos, de una manera gráfica y accesible al usuario, en la cual se observa la ruta planteada y el error (distancia en centímetros entre el punto calculado y el punto real de llegada). La iluminación sobre el área en lo posible no debe generar sombras para evitar que el algoritmo lo interprete como un obstáculo, pero tampoco debe llegar a un extremo de oscuridad.

Debido al brillo que ocasiona la iluminación sobre las piezas plásticas del robot (en especial las de color rojo), es necesario ubicar sobre ellas una superficie que genere la menor cantidad de reflejo hacia el lente de la cámara para detectar correctamente el robot independiente de su ubicación.

La comunicación *bluetooth* debe mantenerse durante el desplazamiento total del robot, debido a que si se ve interrumpida, será necesario

reiniciar la interfaz gráfica y volver a ejecutar el algoritmo de implementación.

Las pruebas de desplazamiento realizadas con la ruta original del algoritmo de esqueletización (es decir, sin quitar puntos de la ruta) proporcionaron como resultado gran consumo de energía y una velocidad de desplazamiento lenta, debido a la cantidad de órdenes enviadas. En las tablas 5 y 6 se aprecia la diferencia de tensión medida sobre la batería del robot una vez realizada la prueba, así como también el tiempo empleado.

Tabla 5. Pruebas ruta optimizada

Ruta Optimizada		
Prueba	ΔV [V]	Tiempo [s]
1	0.07	36.72
2	0.01	36.53
3	0.01	35.4
4	0.66	35.6
5	0.02	36.5

Tabla 6. Pruebas ruta original

Ruta Optimizada		
Prueba	ΔV [V]	Tiempo [s]
1	0.12	100
2	0.1	97
3	0.16	99
4	0.13	98
5	0.17	97

Si algún objeto es de color rojo, dicho objeto debe serlo en su totalidad, para evitar que el robot sea detectado erróneamente.

La altura de la cámara debe ser fija, debido a que ella interviene directamente en los cálculos de odometría del robot.

La resolución mínima sugerida es 640x480 dado que permite una definición aceptable de los objetos y, por otra parte, se logra una velocidad de procesamiento mayor en comparación con resoluciones superiores.

El tiempo de cálculo de la ruta, luego de obtener la imagen correctamente binarizada, es aproximadamente 0.2 s, tiempo superior al obtenido en implementación del algoritmo de campos de potenciales artificiales [2], pero lo suficientemente alto para realizar varias tomas de imágenes y cálculos de trayectorias mientras el robot se desplaza por el entorno.

El tiempo total de ejecución del cuarto algoritmo planteado es similar al tiempo de ejecución de los demás algoritmos, es decir, tomar la fotografía de nuevo y volver a procesarla requiere un tiempo aceptable para el movimiento del robot (3 s).

10. CONCLUSIONES

Se concluye a partir de las diferentes pruebas realizadas, en cuanto a los algoritmos de movimiento, que el cuarto algoritmo planteado posee principalmente la ventaja de no acumular el error adquirido durante el desplazamiento. Y, por otra parte, tiene en cuenta la ubicación actual del robot y hacia dónde debe dirigirse en cada punto de la ruta. El resultado: optimización del movimiento.

Analizando las ventajas y desventajas de las configuraciones típicas para robots móviles con ruedas se concluye que todas sirven para efectuar el desplazamiento, pero la más adecuada para el proyecto es la configuración diferencial, la cual permite realizar prácticamente cualquier ángulo de giro (en este caso los motores giran en sentido opuesto e igual velocidad), parámetro que no es posible obtener con las demás configuraciones en su totalidad.

Para disminuir el error de desplazamiento es recomendable que exista fricción entre la superficie del área determinada y las llantas del robot, puesto que la existencia de derrape entre la superficie de las llantas y el suelo provocaría fallos en la odometría.

La implementación física del algoritmo de esqueletización de imágenes en conjunto con otros algoritmos para el preprocesamiento de

la imagen es perfectamente viable para el planeamiento de rutas libres de obstáculos para robots móviles, teniendo en cuenta las restricciones dadas por el tamaño de los objetos y del robot y el espacio entre estos.

11. TRABAJO FUTURO

Es posible implementar un algoritmo de seguimiento de la ruta más complejo, teniendo en cuenta el modelo cinemático del robot y así obtener un movimiento fluido y sin pausas del robot, ya que hasta el momento el algoritmo consta de secuencias de giro y avance para cada punto de la ruta optimizada.

Se puede experimentar el algoritmo con diferentes tipos de robots, tales como plataformas triciclo, Ackerman, omnidireccionales, etc., para evaluar su efectividad de movimiento con el algoritmo de esqueletización.

Referencias

[1] V. Gonzáles y R. Parkin, "Evadiendo obstáculos con robots móviles", *Revista Digital Universitaria*, vol. 6, no. 1, pág. 2 – 9. Ene. 2005.

[2] L. Xu y T. W. S. Shing, "Self-organizing potential field network: A new optimization algorithm". *IEEE Transactions on Neural Networks*, vol. 21, no.9, pp. 1482-1495, 2010.

[3] E. De los Santos de la Rosa, *Heurística para la generación de configuraciones en pasajes estrechos aplicada al problema de los clavos*, Cholula, Puebla, cap. 1, mayo 2004.

[4] P. Bhattacharya y M. L. Gavrilova, "Roadmap-based path planning - using the voronoi diagram for a clearance-based shortest path", *IEEE Robotics & Automation Magazine*, vol. 15, no. 2, 58-66, 2008.

[5] M. Ibarra, F. Quiñones, I. García y J. Ramírez, *Desplazamiento de un robot con localización y evasión de obstáculos por visión y ultrasonido*, Puebla, 2009.

[6] J. Guzmán y A. Gutiérrez, "Planificación abstracta en los sistemas de robots autónomos", *Revista Colombiana de Tecnologías de Avanzada*, vol. 2, no. 20, 2012.

[7] M. Mora, L. Armesto y J. Tornero, *Sistema de navegación de robots móviles en entornos industriales*, Valencia, sep. 2004.

[8] G. E. Fainekos, "Revising temporal logic specifications for motion planning", Proceedings *IEEE International Conference on Robotics & Automation*,pp. 40-45, 2011.

[9] M. Egerstedt y P. Jensfelt, "A control theoretic formulation of the generalized slam problem in robotics", *American Control Conference*, pp. 2409-2414, 2008.

[10] S. M. LaValle, "Motion planning", *IEEE Robotics & Automation Magazine*, vol.18, no.1, pp. 79-89, 2011.

[11] M. Charles, *Geometric Path Planning For a Lego AUV*, Master of Arts in Mathematics University of Hawaii at Manoa, 2012.

[12] A. Murillo, A. Mosteo, J. Castellanos y L. Montano, *A Practical Mobile Robotics Engineering Course, using LEGO Mindstorms Department of Informatics and Systems Engineering*. University of Zaragoza: Spain, 2011.

[13] D. Zaldívar y E. Valdemar, *Visión por Computador utilizando MATLAB Y el Toolbox de Procesamiento Digital de Imágenes*, 2006.

[14] R. González and R. Woods, *Digital Image processing*, 3rd ed., Prentice Hall, 2007.

[15] F. Carranza, *Esqueletización: tópicos especiales en procesamiento gráfico*, Universidad Nacional de Trujillo: Perú, 2006.