Planificación de movimiento de un robot lego basado en la aplicación del algoritmo *Dist-Bug*

Motion´s planning of a lego robot based on the implementation of the algorithm *Dist-Bug*

Diego Fernando Márquez, Est. & Ingrid Durley Torres M.Sc. (c)
Jaime Alberto Guzmán Luna, Ph.D.
Universidad Nacional de Colombia, sede Medellín
{dfmarque, jaguzman, idtorresp}@unal.edu.co

Recibido para revisión 05 de mayo de 2011, aceptado 28 de junio de 2011, versión final 25 de julio de 2011

***Resumen*— En este artículo se aplica una técnica de planificación de movimiento basada en el algoritmo DistBug para la construcción de rutas de navegación de un robot móvil LEGO Mindstorm NXT 2.0. El propósito de esta planificación, consiste en que el robot construya una ruta de navegación en tiempo real y libre de colisiones, que lo lleve desde un punto inicial a un punto final, mientras percibe la información del mundo, el cual asume como incierto, excepto su posición inicial y su posición de final. Para esta tarea, el robot utiliza sensores de sonar y de contacto como elementos de detección a fin de identificar la distribución de los obstáculos y la posición actual del robot.**

***Palabras Clave*— Planificación de movimiento, Algoritmo *DistBug*, LEGO Mindstorm, Construcción de rutas.**

***Abstract*— In this article applies a motion's planning technique based on the algorithm for constructing DistBug navigation through a LEGO Mindstorm NXT robot mobile 2.0. The purpose of this planning, is that the robot has to build a shipping route in real time and free of collisions, which take you from a starting point to an endpoint, while receiving the world's information, which assumes as uncertain, except its initial position and end position. For this task, the robot uses sonar sensors and contact detection elements to identify the distribution of obstacles and the robot's current position.**

***Keywords*— Motion's planning, Algorithm DistBug, LEGO Mindstorm, Routes bulding.**

I. INTRODUCCION

La construcción de rutas que han de guiar a un robot en su trayectoria de movimiento evitando obstáculos, desde un estado inicial a un estado final, ha sido tratada ampliamente en la literatura [1]; Aunque varias técnicas han sido propuestas, una de las más eficientes resulta encontrarse en la aplicación de las técnicas algorítmicas. La mayor parte de estas propuestas, construyen su solución basados en un diseño espacial donde reciben posiciones de un plano cartesiano y algunos ángulos que direccionan el sentido de orientación del movimiento. Bajo este enfoque el robot busca desplazarse a partir de una secuencia de configuraciones espaciales, evitando obstáculos hasta alcanzar la posición definida en su punto final.

En esta dirección ha surgido la familia de algoritmos bugs. Los cuales deben construir un camino para mover el robot al punto meta en una línea recta, esto si el camino es claro. Cuando se encuentra un obstáculo, el robot, sigue el límite del obstáculo es decir, empieza a circunnavegarlo hasta que el camino sea claro nuevamente [2]. Esta propuesta ha sido extendida con las técnicas de planificación [3] buscando construir de manera abstracta un plan (secuencia) que defina el conjunto de posiciones espaciales de un plano cartesiano bidimensional. Una ruta que permita Estos algoritmos presenta algunas ventajas en comparación con los otros algoritmos de planificación. El robot solo está interesado en llegar a su ubicación objetivo si el objetivo es alcanzable, sin embargo si es inalcanzable, el robot es capaz de terminar la tarea encomendada.

Este trabajo presenta una experiencia práctica en la implementación del algoritmo DistBug sobre el robot móvil Lego NXT 2.0 [7] construido como mecanismo de experimentación y comprobación de la correcta funcionalidad del algoritmo en mención.

Con el objetivo de brindar una visión más detallada del diseño de implementación de esta experiencia de aplicación en el dominio de la robótica móvil, este documento se organiza como sigue: en la sección 2, se realiza una revisión del marco teórico que resume los principales conceptos dando un énfasis especial al algoritmo DistBug. La sección 3 detalla el proceso de implementación del algoritmo DistBug con el robot de experimentación. En la sección 4 se define un caso de experimentación y se recopilan los resultados del mismo. Finalmente en la sección 5 presenta las conclusiones y el trabajo futuro sobre proyecto.

II. LA PLANIFICACIÓN PARA LA NAVEGACIÓN DE ROBOTS

Planificar es un proceso abstracto y deliberativo que escoge y organiza acciones anticipando sus resultados esperados [$]. La deliberación busca lograr, tan bien como sea posible, algunos objetivos preestablecidos. La planificación es un área de la Inteligencia Artificial que estudia este proceso de deliberación computacionalmente. Así mismo, la planificación de movimientos se preocupa por la ejecución de una trayectoria geométrica desde una posición inicial en el espacio hasta un objetivo, a través de un sistema móvil, como lo puede ser un carro, un camión, un brazo mecánico o, en el caso aquí presentado, un robot móvil Lego Mindstorms.

El algoritmo descrito en este artículo se centra en la implementación del algoritmo DistBug [4,6], el cual hace parte de la mencionada familia de los Bugs. El DistBug utiliza los sensores de manera eficiente para definir una nueva ruta y encontrar la meta [5].

III. LA FAMILIA DE LOS BUG'S

Los Bugs son un conjunto de algoritmos que pueden ser implementados como planificadores simples, para la construcción de rutas de navegación, en ambientes reales con presencia de obstáculos. Cuando, uno de estos algoritmos, se enfrenta a un obstáculo desconocido, es capaz de generar fácilmente el contorno del objeto obstáculo con el que se encuentra, este contorno es representado como un conjunto de coordenadas de una superficie 2D, si solo si, un camino a la meta existe. El objetivo, entonces es generar un camino o ruta libre de colisiones, mediante la circunnavegación y las restricciones dadas por el obstáculo u objeto. Los algoritmos Bugs han sido implementados en el dominio de la robótica móvil con el fin de planificar los movimientos a través de la construcción de una ruta solución libre de colisiones, que parte de un punto inicial hasta llegar a un final, Los algoritmos actúan bajo tres hipótesis acerca del robot móvil: i) El robot es un punto, ii) Maneja localización perfecta (no incertidumbre), iii) Sensores Precisos [4]. El conjunto de algoritmos Bugs se compone de los siguientes: Bug1 [5-8-3], Bug2 [5-8-3] y DistBug [4, 6], entre otros. A continuación se detalla brevemente el algoritmo DistBug, con el fin de dar una visión de la experiencia de aplicación desarrollada [9].

El algoritmo Distbug trata de asegurar que el destino es alcanzado, si es posible, o indicar si el objetivo es inalcanzable. El algoritmo es reactivo en el sentido de que depende de los datos tomados de su entorno para tomar decisiones locales, y no utiliza ninguna forma de representación interna del medio ambiente.

El algoritmo se compone de dos modos de locomoción: (i) se mueve directamente hacia la meta entre los obstáculos siguiendo los bordes de los obstáculos. De esta manera, los bordes de los obstáculos son circunnavegados hasta que cierta condición indica que se debe volver hacia el punto final. La dirección que indica en qué modo se debe seguir los bordes, depende de la posición del punto final e influirá en la longitud de la ruta que será cubierta. El algoritmo decide en qué dirección se moverá sobre la base de la información adquirida.

El algoritmo DistBug, fue inventado por Kamon y Rivlin en 1997 [6]. Es muy similar al algoritmo bug2 porque tiene una condición idéntica acerca de irse, con apenas una sutil diferencia. La única diferencia real es que DistBug no mantiene una lista de los puntos anteriores, mientras que Bug2 sí lo hace.

El siguiente pseudocódigo nos muestra cómo funciona el algoritmo DistBug en forma global:

- Trazar ruta directa desde el punto inicial al punto final.
- Si el objetivo es alcanzado hacer la parada.
- Si se hace un contacto con un obstáculo seguir las fronteras (hacia la izquierda) hasta que llegar a la meta sea posible.
- Repetir

Esta técnica es simple, pero en un ambiente controlado es capaz de evitar obstáculos y evitar una trampa como los mínimos locales. Sin embargo, se limita a un medio ambiente estático, sin obstáculos en movimiento.

En este algoritmo, el robot tiene un máximo de detección sensorial representado en un rango *R* (*distancia detectada*) y dos comportamientos básicos que son límite de seguimiento límite de movimiento a las acciones de meta.

Como se observa en Figura1, el modelo de planificación de movimientos es representado en plano cartesiano de dos dimensiones. Y opera de la siguiente manera: en primer lugar, el robot se mueve hacia la *G* (*punto final*) hasta que se enfrenta a un obstáculo. Entonces su límite de acción de seguimiento se activa en sentido horario (por defecto). Durante los limites siguientes el robot realiza el registro de las distancias mínimas, denominadas *dmin*(*G*), que significa la distancia obtenida de la trayectoria a *G* (*punto final*) lograda desde el punto de impacto pasado. El robot también detecta la distancia en el espacio libre *F* (*distancia libre*), que es una distancia de un obstáculo del robot de localización, *X* (punto de localización), en la dirección de G (*punto final*).

Si ningún obstáculo se detecta, *F* (*distancia libre*) se encuentra en *R* (*distancia detectada*). El robot deja obstáculo límite como se indica en la Figura1 solo cuando la ruta de acceso a *G* (*punto final*) es clara, es decir cuándo:

$$d(X, G) - dmin \leq F(G) - Paso \quad (1)$$

donde, $d(X, G)$ es la distancia localizada en un punto X al punto G (punto final) - *dmin* que corresponde a las distancias obtenidas de la trayectoria, son menore o iguales a la distancia

libre a G – *Paso,* que es la constante de desplazamiento predefinida[6].

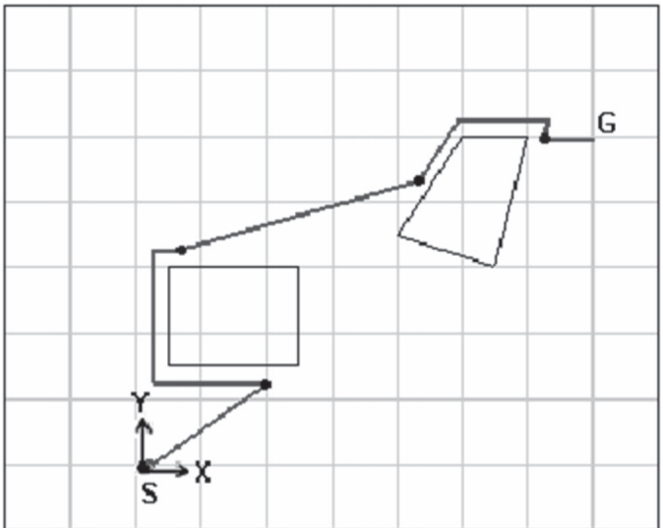


Figura1. Ruta o camino generado por DistBug.

IV. MODELO

La arquitectura básica del modelo propuesto, la cual se detalla en la Figura2, consiste en 3 módulos básicos:

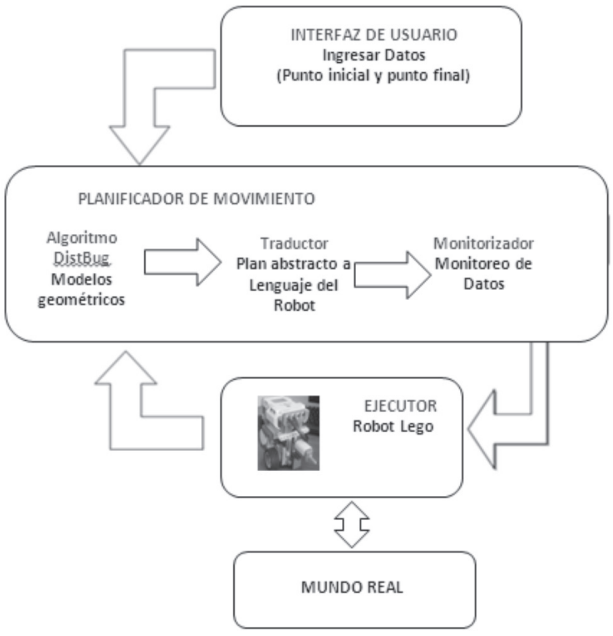


Figura2. Arquitectura propuesta para el algoritmo

A. Interfaz GUI (Interfaz Gráfica de Usuario)

La interfaz de usuario, mostrada en la Fig.3, actúa como un puente entre el sistema de planificación de movimiento del robot y el usuario final; en ella, un usuario, puede ingresar los elementos o datos que configuran un problema de navegación en un robot. Tal solicitud se realiza de forma clara para el usuario y representa las coordenadas cartesianas en dos puntos, del área de trabajo que se irá a definir para el robot.

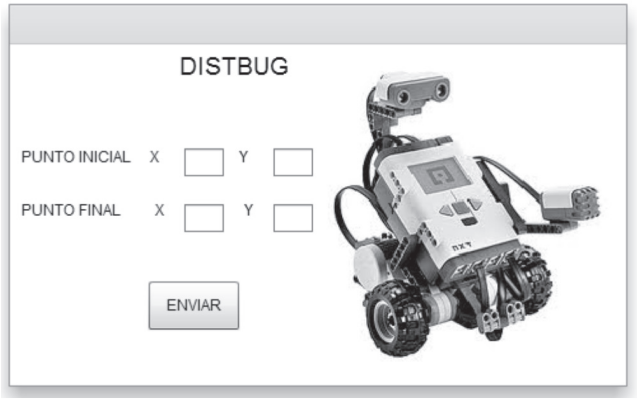


Figura3. Interfaz de usuario para ingreso de información del robot móvil.

B. Modelo de planificación

En este módulo se implementa el algoritmo *DistBug* como elemento matemático básico que actúa como planificador en este caso, mediante modelos geométricos que permiten seguir el comportamiento de este algoritmo (definido por los pasos citados en la sección II). Además, este módulo, se encuentra acompañado de un software traductor, que lee las descripciones del plan en un nivel abstracto y las traduce en órdenes propias del ambiente de programación Java-Lejos, que implementa los comandos de control de cada uno de los componentes físicos del robot permitiendo el monitoreo de sus datos.

Este módulo también cuenta con un monitorizador de datos que permite indicar al algoritmo que debe re-planificar, cuando se requieran generar nuevas trayectorias a partir de datos arrojados por el robot móvil constituyendo así un modelo dinámico que se acopla a los cambios que brinda el mundo abierto.

C. Ejecutor del robot móvil

Este módulo es quizás el más llamativo del modelo, debido a que representa la parte tangible del movimiento del robot; en él se ejecutan los comandos bajo el ambiente Java-Lejos que coordina los movimientos del robot móvil, permitiendo que se realice las distancias y los ángulos de giro que permitan cumplir con la tarea que tenía el algoritmo DistBug.

En este módulo además se encuentra el robot Lego diseñado e implementado, para el proceso de experimentación y con el cual se obtendrá toda la información del mundo real, además de afectarlo con la propia ejecución del robot. La arquitectura del robot construido en este trabajo, se reduce a tener dos servomotores a modo de llanta (en un mismo eje). Se le debe indicar al programa la distancia entre las mitades de cada llanta, y el diámetro de las mismas, medida que para ambas deben ser iguales. Las pruebas que se realizaron se hicieron con un robot cuyo diámetro de llantas era de 4.33cm y la distancia entre ambas era de 10.5cm este diseño es mostrado en la Figura3.

Revista Avances en Sistemas e Informática, Vol.8 No.2, julio de 2011 - Medellín. ISSN 1657-7663

Otra característica importante que se destaca dentro de este diseño del robot está representada en el uso de dos tipos de sensores con los cuales va a recibir e intercambiar información con su entorno. Para este caso en especial corresponden a los sensores de sonar y táctil como los de la Fig.4. El sensor de sonar permite mapear el medio ambiente a través de la identificación de obstáculos los cuales identifica la distancia de los obstáculos. Mientras, que el sensor táctil, permite circunnavegar alrededor de los obstáculos.

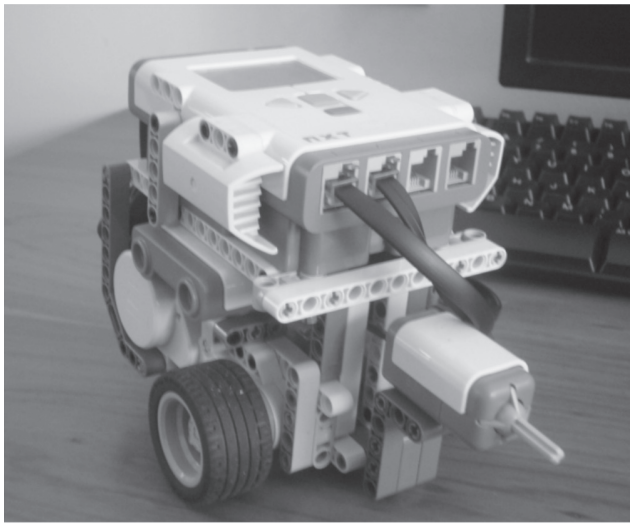


Figura3. Arquitectura Física del robot Lego.

Figura4. Sensor sonar – Sensor Táctil

V. IMPLEMENTACIÓN

En general, la implementación del modelo anterior se realiza a traves del algoritmo DistBug el cual se publica como pseudocódigo [4]. Pero el software programado está basado en dos lenguejes, JAVA y java Lejos [10], este último corresponde al lenguaje con el cual se programa la serie de robots lego MINDSTORM NXT 2.0. [7].

El funcionamiento de este modelo ya implementado corresponde a los siguientes pasos:

- Inicialmente se ingresan los datos de usuario como son los puntos iniciales y finales que corresponden a la ruta que debe seguir el robot móvil esto a través de una GUI (interfaz gráfica de usuario).
- Estos datos, son retomados por el módulo de planificación quien mapea en su modelo lógico ambos puntos según sus coordenadas. Posteriormente siguiendo el algoritmo en pseudocódigo, citado en la sección II y utilizando diferentes métodos geométricos que permitan cumplir con la lógica del algoritmo DistBug, se traza la línea recta desde el punto inicial al final y en caso de encontrarse obstáculos en su ruta, el robot conservando la dirección al punto final, se dirigirá a cada uno de esos obstáculos que interfieren con su objetivo e intentará circunnavegarlo, retomando información constantemente del ambiente. En este modelo se deben calcular estos datos adquiridos por el robot móvil dados por la percepción del mundo para que el algoritmo pueda tomar una decisión de cuál es la distancia más corta y generar la nueva trayectoria hacia la meta.
- Una vez tomada una decisión esta debe ser ejecutada para ello, se realiza una traducción de la nueva ruta, convirtiendo esos datos finales en términos de ángulos y distancias que son parámetros que interpreta el ejecutor sobre el robot móvil, haciendo uso para ello de la *librería lejos.robotics. navigation* [10] que le permite al robot móvil girar los ángulos indicados o la distancia a recorrer (en centímetros), sin necesidad de preocuparse por los grados que debe girar cada uno de los dos motores (llantas).

En este funcionamiento se utiliza geometría para que el robot se pueda localizar como un punto dentro de su modelo lógico, así como para actualizar su posición y datos de orientación. Es importante resaltar, que el robot móvil debe actualizar su localización con frecuencia para no pasar los puntos de contacto con los obstáculos o la pendiente tomada como dirección en la ubicación del objetivo final

VI. EXPERIMENTOS Y RESULTADOS

Con el fin de evaluar el correcto funcionamiento del modelo implementado, se define inicialmente un área de trabajo real, determinada por las siguientes medidas: 2,5 m de longitud, anchura 2,5m y un obstáculo continuo (línea de madera) creada a propósito, como parte del ambiente natural del mundo abierto. Fig.5.

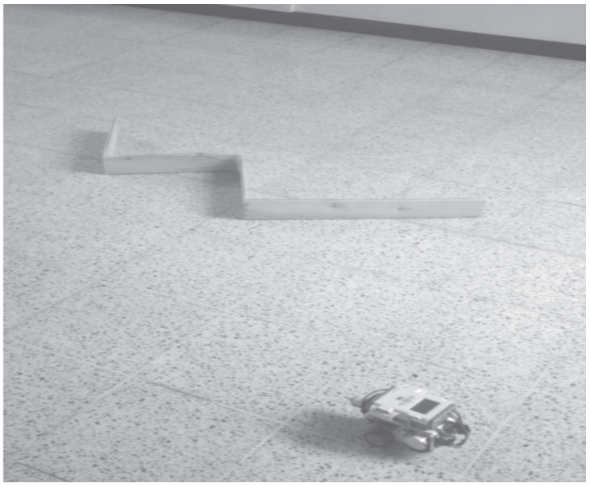


Figura5. Ambiente real de prueba para el robot.

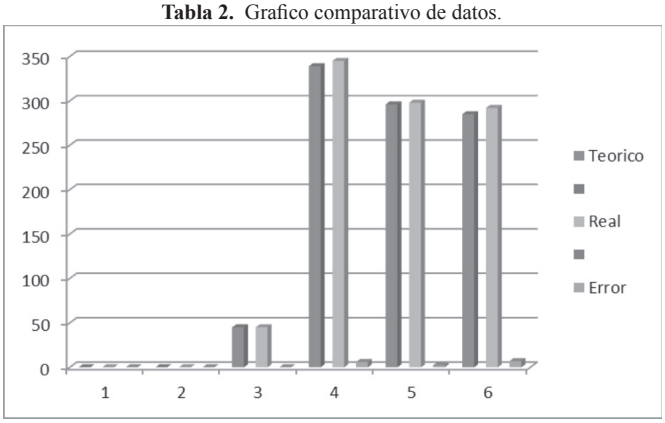
Aunque los modelos reales, se implementa en coordenadas 3D, para facilidad de de este experimento, el robot móvil será graficado tomando en consideración una superficie real 2D que tiene un punto inicial S en la coordenada (0,0), y un punto final G en la coordenada (240, 240). Los datos experimentales son detallados y registrados en las tablas, I y II mostradas a continuación.

El experimento entonces consistió en comparar que datos debían dar en el modelo teórico, según las medidas reales calculadas sobre el escenario implementado, frente a los obtenidos por el modelo de planificación acá desarrollado. Los resultados pueden verse en la Tablas a continuación.

Tabla 1. Comparación de datos teóricos y experimentales.

Tabla de pruebas DistBug			
Variables	Teórico	Real	Error
S(x,y) [cm]	S(0,0)	S(0,0)	0
G(x,y) [cm]	G(240,240)	G(240,240)	0
$\theta_{pendiente}$	45	45	0
D_{total} [cm]	339	345	6
D_{min} [cm]	296	298	2
$D_{new\ path}$ [cm]	285	292	7

Tabla 2. Grafico comparativo de datos.



Para este tipo de resultados es claro que existen errores en la aplicación; errores que pueden ser atribuidos por ejemplo al propio robot como por ejemplo la distancias de sus ruedas, mal alineamiento de las mismas, errores en los ángulos de giro, entre otros. Además se encuentran otros tipos de errores que no son propios del robot como suelos desnivelados, derrapes entre otros. Por estos motivos es claro que debe existir una gran incertidumbre en el momento de la aplicación o creación de la nueva ruta por parte del algoritmo DistBug sobre el robot móvil.

En varias pruebas realizadas se intercambia las ruedas del robot móvil (ver Figura6) y se encuentra que algunas presentaban de acuerdo a su diseño mucha más fricción, lo cual hacia que el error en su desplazamiento fuera mayor. En otros casos aunque las ruedas fueran de diámetro mayor presentaba un error mucho menor ya que tenían un diseño más sencillo en el momento del desplazamiento.

En particular la teoría de los Bugs de movimiento se puede aplicar a los robots móviles, pero para este caso se debe tener también considerar que para un desarrollo óptimo, las teorías de error que permita minimizar la incertidumbre en los desplazamientos, son un aspecto prioritario.

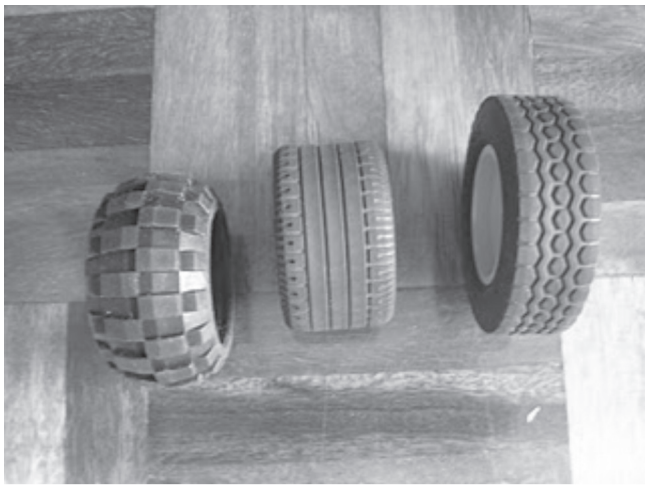


Figura6. Tipos de ruedas para desplazamiento del robot móvil.

Haciendo un análisis más enfocado al desplazamiento del robot móvil se tiene como una posible solución a la incertidumbre un método de localización que permita verificar la estimación de la posición de este. La Fig.7, muestra un ejemplo de ello. Pero aún se debe tener en cuenta las diferentes variables en la medición que se generan mediante el desplazamiento en mundo abierto.

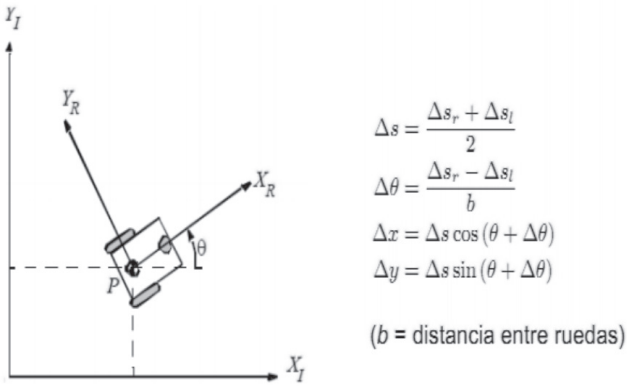


Figura7. Método de localización para estimación de posición

VII. CONCLUSIONES Y TRABAJOS FUTUROS

En la construcción de la ruta por parte del robot se sigue el modelo del algoritmo DistBug, el cual se basa en que no se conoce ningún parámetro o variable del mundo y en el caso de estudio solo hace uso de un sonar como sensor para la

percepción del mundo y su posterior toma de decisiones con dicha información.

Se realizó una implementación del algoritmo y su posterior experimentación haciendo uso del lenguaje JAVA-Lejos en un robot LEGO de tres ruedas.

Los resultados obtenidos presentan que existen factores que originan errores en el desarrollo del algoritmo identificándose factores como la ficción de las llantas y la propia geometría del robot.

Como trabajo futuro se ha planteado la elaboración de un estudio más detallado de las variables que afectan la precisión del robot en su desplazamiento al aplicar el algoritmo DistBug estudiado, para brindar una solución mas precisa al problema de la planificación de movimientos haciendo uso de este algoritmo.

AGRADECIMIENTOS

Este trabajo es apoyado por el proyecto "Programa de Fortalecimiento a Grupos de Investigación y de Creación Artística de la Universidad Nacional de Colombia 2010", mediante el cual se apoyan semilleros de investigación de la sede.

REFERENCIAS

[1] J-C. Latombe, Robot Motion Planning. Kluwer Academic Publishers, 1991.
[2] M. I. Ribeiro, Obstacle Avoidance, 2005.
[3] V. Lumelsky and Stepanov, Path-planning strategies for a point mobile automaton amidst unknown obstacles of arbitrary shape, in Autonomous Robots Vehicles, I.J. Cox, G.T. Wilfong (Eds), New York, Springer, pp. 1058 – 1068, 1990.
[4] Ng. James and T. Bräunl, Performance Comparison of Bug Navigation Algorithms, J Intell Robot Syst, Springer Science 50:73–84 DOI 10.1007/s10846-007-9157-6, 2007.
[5] J. Lumelsky, V.J., Stepanov, Dynamic path planning for a mobile automaton with limited information on the environment, IEEE Trans. Automat. Contr. 31, 1058–1063, 1986.
[6] [6] Kamon, I., Rivlin, E., Sensory-based motion planning with global proofs, IEEE Trans. Robot. Autom.13, 814–822, 1997.
[7] LEGO.com MINDSTORMS: Products - NXT 2.0 - 8547 [Online]. Disponible: http://mindstorms.lego.com/en-us/products/default.aspx
[8] S. P. Bingulac, "On the compatibility of adaptive controllers (Published Conference Proceedings style)," in *Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory*, New York, 1994, pp. 8–16.
[9] Seth , KANTOR, George A. ,BURGARD, Wolfram , KAVRAKI, Lydia E. ,and THRUN, Sebestian , Principles of Robot Motion : Theory, Algorithms, and Implementations, The MIT Press, 5 Cambridge Center, 1st Edition, pp. 17 - 38, 2005.
[10] Alpaslan YUFKA and Osman PARLAKTUNA, Performance comparison of bug algorithms for mobile robots.
[11] LeJOS, Java for Lego Mindstorms [Online]. Disponible: http://lejos.sourceforge.net/.
[12] M. Ghallab, D. Nau, P. Traverso, Automated Planning, theory and practice. San Francisco: Morgan Kaufman 2004, pp. 1–3.