Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

**Unidad Zacatenco**

**Departamento de Computación**

# Rutas de navegación y evasión de obstáculos de robots móviles en terrenos exteriores.

**Tesis que presenta**

**José Alejandro González Sarabia**

**para obtener el Grado de**

**Maestro en Ciencias**

**en Computación**

**Dr. José Matías Alvarado Mentado**

**México, Distrito Federal** **Noviembre, 2012**

# Resumen

En el estado del arte se explican los antecedentes sobre cómo los robots móviles realizan la navegación y evasión de obstáculos, en particular los robots con ruedas focalizando en el tipo de ruedas, la cinemática y la odometría. El objetivo de la tesis concierne a creación de rutas y la evasión de los obstáculos, inmóviles y casos sencillo de obstáculos móviles sobre terrenos de tierra compacta, pasto y pavimento; en particular haciendo ajuste de velocidad en función de las características de cada terreno. Desarrollamos la construcción del mapa con los algoritmos para obtener la información del terreno y construir las rutas de navegación. Para la evasión de un obstáculo, el robot lo detecta mediante sensores ultrasónicos y recalcula la ruta de navegación, tanto para obstáculos fijos como para algunos en movimiento; para los obstáculos en movimiento el robot calcula si lo evade adelantándose al obstáculo o esperando a que pase, dependiendo si lo detecta o no con los sensores. Asimismo, se realiza el ajuste de velocidad para la navegación del robot según las características del terreno y se combina la evasión de obstáculos fijos con el ajuste de velocidad.

Los resultados a nivel simulación muestran la creación de rutas de navegación del robot a partir de la información obtenida del mapa y los obstáculos. Con el robot físico se prueban sus capacidades de navegación y evasión en terrenos exteriores, midiendo con odometría la precisión/error entre la posición final del robot y la meta, y el tiempo utilizado. Se prueba la evasión de obstáculos, móviles e inmóviles, algunas con obstáculos fijos tal que el robot hace ajuste de velocidad. Los resultados de los experimentos muestran que los porcentajes de error dependen de la cantidad de obstáculos a evadir y de la dificultad del terreno de desplazamiento, siendo menos relevante la longitud de la ruta de navegación del robot. Considerando las características del terreno para decidir la velocidad de navegación del robot, más alta posible pero sin riesgo de derrapes, se disminuye el margen de error a la meta, medido con odometría. Se concluye que la construcción de rutas de navegación considerando las características del terreno y la evasión de obstáculos, fijos o móviles, haciendo los ajustes de velocidad adecuados para evitar derrapes o colisiones, hace segura y eficiente la navegación del robot en terrenos exteriores de tierra, pasto y pavimento.

# Abstract

We begin with the state of art, describing how to make mobile robots navigation and obstacle avoidance, in particular wheeled robots focusing on the type of wheel, kinematics and odometry. The objective of the thesis is concerned the creation of routes and avoidance of obstacles, immobile and simple cases of moving obstacles, both on terrain of compacted soil, grass and pavement; in particular by making speed adjusting according to characteristics of each terrain. We deploy algorithms to construction of the map with the terrain's information to build navigation routes. For the avoidance of an obstacle, the robot detects the obstacle by ultrasonic sensors and recalculates the route navigation, both for immobile obstacles and dome moving. For moving obstacles evades, the robot calculates if can advance the obstacle or should waiting it pass, depending on whether or not obstacle is detected by the sensors. Robot also makes speed updating for robot navigation by regarding the terrain features and it combines with immobile obstacle avoidance.

We tested mobile and immobile obstacle avoidance, some with immobile obstacles such that the robot makes speed updating. The results of the experiments show that the error rates depend on the number of obstacles to avoid and the difficulty of the terrain to navigate, being the route's length less relevant. The terrain features are regarded to determine the robot's speed hence use the highest possible for navigation at the time it minimizing the risk of skidding, and as a consequence decrease the margin of error to the goal, it measured by using odometry; in addition the time spent from initial to goal position is measured to establish the benefits from speed updating. We conclude on the advantages to use the construction of navigation routes considering the terrain features and the immobile or mobile obstacle avoidance, by making appropriate speed updates to avoid skidding or collisions hence making safe and efficient navigation of robot over terrains of, respectively, compact soil, grass and pavement.

# Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACyT) de México, por la beca de maestría de A. González Sarabia, CVU 369351.

Agradezco al Dr. José Matías Alvarado Mentado por aceptar ser mi asesor y por su dirección durante este trabajo de tesis.

Agradezco a mis sinodales por las observaciones y opiniones dadas para mejorar este trabajo.

Agradezco a mi madre y mis tíos por el apoyo brindado durante todo este tiempo, para lograr esta meta más en mi vida.

Agradezco a mis primas por su apoyo brindado durante este tiempo.

Agradezco a todos mis amigos y compañeros que siempre estuvieron ahí para aportar con ideas y opiniones, así como de su apoyo.

Y en especial agradezco a Marina del Carmene Azueta Vargas (DEP), por todo su apoyo brindado, para lograr este objetivo más en mi vida.

# Índice general

# Índice de figuras

# Indice de Tablas

# Capítulo 1

# Introducción

*Este capítulo lo dividiremos en cinco secciones: en la primera describiremos los antecedentes del problema y en la segunda el problema motivo de la tesis; en la tercera sección planteamos los objetivos a alcanzar, generales y específicos; en la penúltima sección se describen los resultados esperados, y por último describimos el contenido y la estructura del resto de la presente tesis.*

## 1.1. Antecedentes

Desde que se crearon los robot móviles, ha sido evidente lo fácil que estos podrían llegar a colisionar, ya sea con su medio ambiente o entre ellos mismo. A partir de esto los investigadores han buscan realizar métodos de anti-colisión y de navegación segura. Esta tesis aborda para un robot móvil con ruedas, ambos temas, la navegación y la evasión de obstáculos en su ruta en terrenos exteriores de pavimento, tierra, pasto y tierra-pasto.

Actualmente, los robots con ruedas se mueven y, en algunos casos, evaden obstáculos. Lo hacen con rapidez y precisión, pero siempre y cuando sea la navegación sobre superficies lisas, sin irregularidades, cómo puede verse en las competencias ROBOCUP o FIRA [26] [6] [18] [45] [38]. Sin embargo, tal precisión y agilidad en superficies irregulares está aún lejana de lograrse [2] [35] [24].

El presente trabajo abona en mejorar el desempeño de un robot con ruedas al desplazarse en terrenos con superficies irregulares. En particular, un reto a resolver es evitar derrapes (y posibles extravíos) del robot al girar sobre sus ruedas para evadir un obstáculo: la potencia requerida del motor para una buena locomoción del robot depende, asimismo, de la fricción de las ruedas sobre la superficie del terreno, y por tanto de las características del terreno.

Mediante métodos de planeación de ruta se busca encontrar una ruta libre de

colisión entre la posición inicial del robot y el punto meta, valiéndose asimismo de métodos de evasión de obstáculos en la trayectoria de navegación. Un robot móvil puede encontrarse con obstáculos estáticos o en movimiento. Para que sean útiles en el mundo real, los robots móviles deben moverse con seguridad en entornos no estructurados y alcanzar sus metas pese a los cambios que ocurran. Los entornos reales rara vez son predecibles o perfectamente conocidos para el robot.

Las etapas a considerar para la navegación en terrenos exteriores del robot con ruedas que se utiliza para el desarrollo de esta tesis son:

1. La planificación del movimiento: una ruta completa desde el punto inicial hasta el punto meta, que el robot debe de seguir.

2. Evaluación de la ruta local. Las rutas de navegación posible son evaluadas, para de entre ellas, elegir la que sea libre de obstáculos y tal que ofrezca las mejores condiciones de navegación para el robot.

3. Capacidad de reacción. actualizar la dirección de movimiento basado en la información actual de la ruta, si hay obstáculos detectarlos para evadirlos en su ruta a la meta.

Los diferentes métodos de navegación combinan estas etapas del movimiento para la construir las rutas de navegación de un robot. Los métodos de navegación de un punto inicial a un punto meta como el del gradiente [28], los campos potenciales artificiales [27] [39], y algunos otros emplean herramientas como visión por computadora [2][12] o GPS [1]. Por otra parte los métodos de evasión de obstáculos en la ruta de navegación del robot pueden categorizarse dependiendo si el obstáculo tiene o no movimiento; para la evasión de obstáculos en movimiento se generalizan algunos métodos, parar conocer el desplazamiento y así re-calcular la ruta de navegación.

## 1.2. Descripción del problema

Nuestro reto consiste en lograr que un robot móvil con ruedas sea capaz de navegar por terrenos exteriores, con obstáculos fijos o en movimiento, con velocidad y aceleración dinámica. Este reto es descompuesto en dos sub-problemas para abordarlos de manera más sencilla. Los sub-problemas a considerar son:

1. Llegar a la meta: es un problema global, pero las rutas se construyen utilizando la información local, considerando las características del terreno y, en particular, los obstáculos a evadir.

2. Evasión de obstáculos. Se resuelve utilizando información local, considerando la distancia al obstáculo, y la velocidad tanto del robot como la del obstáculo.

El primer sub-problema de un robot navegando en un terreno exterior lo plantea la diversidad de terrenos cuyas características pueden poner en riesgo al robot; las características de los terrenos dependen si son de tierra, pasto, o gravilla, entre otros. En terrenos exteriores el robot debe evitar derrapes y colisiones, así como superar hoyos y colinas, entre otros. Estas dificultades pueden evitarse o minimizarse considerando las características del terreno para obtener una velocidad adecuada tal que el robot navegue de manera segura sobre ellos. El sub-problema de evasión del obstáculo mientras el robot navega hacia el punto meta, debe considerar en su solución, la velocidad, aceleración y tamaño del robot, así como la velocidad, aceleración y el tamaño del obstáculo, y las características del terreno para encontrar una ruta adecuada y de ser posible óptima.

En la Figura 1.1 se muestra un escenario donde el robot $R_1$ debe llegar a la meta y el robot $R_2$ es un obstáculo móvil interpuesto en la ruta de $R_1$. En el problema de evadir un obstáculo encontramos dos casos principales que son:

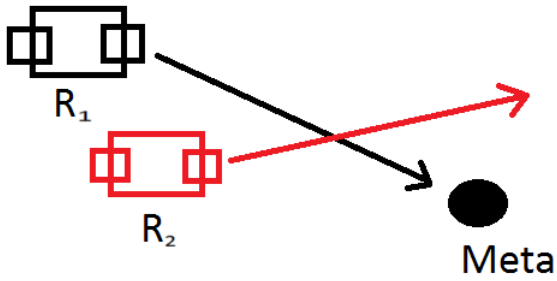


Figura 1.1: Navegación de un robot móvil con un obstáculo en su área de navegación.

1. Caso estático. Usualmente es simple identificar una ruta entre varias rutas a la meta .

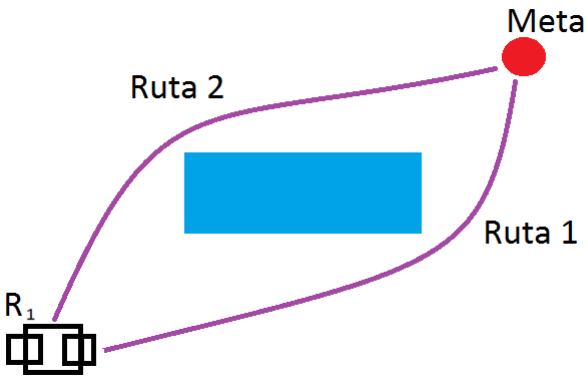


Figura 1.2: Navegación con un obstáculo estático.

2. Caso dinámico. Este caso es complicado y debe considerarse la velocidad del obstáculo para realizar la evasión. Como se muestra en la Figura 1.3 el $R_2$ se interpone en el camino del $R_1$ obstaculizando la ruta para llegar a la meta.

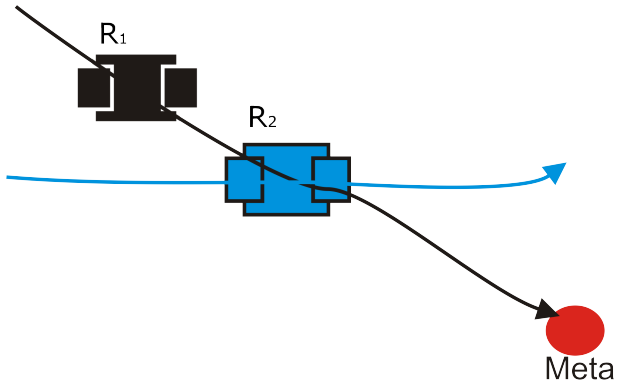


Figura 1.3: Navegación con un obstáculo dinámico.

En la Figura 1.2 se muestra un escenario donde el robot puede elegir entre diferentes rutas. El obstáculo estático (caja, piedra, etc.) no tiene movimiento, el robot puede llegar al punto meta con los métodos tradicionales de navegación y evasión de obstáculos. La problemática de evadir el obstáculo es la siguiente:

- Evadirlo por el frente, sin colisionar.
- Detenerse y esperar el paso del obstáculo y continuar a la meta.

La evasión se ilustrada en la Figura 1.4. La forma en que evadiremos el obstáculo depende las características del terreno y la capacidad de nuestro robot, para realizar la evasión.

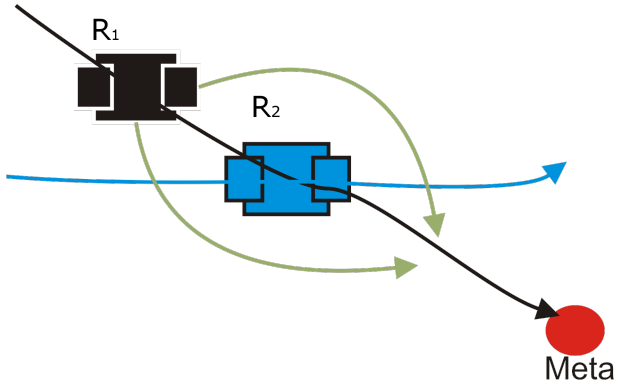


Figura 1.4: Evasión de un obstáculo en movimiento.

## 1.3. Objetivos

**General:** Construir rutas de navegación para un robot con ruedas, en terrenos de pavimento, tierra o pasto, y tal que la navegación sea segura, sin derrapes, extravíos o colisiones evadiendo obstáculos. Asimismo se pretende minimizar el error a la meta.

**Particulares**

1. Para el problema de navegación y evasión de obstáculos, las variables a considerar son:
   - Construir mapa de navegación.
   - Construir rutas
   - Analizar formas de evadir obstáculos.
2. Diseñar e implementar algoritmo para la construcción del mapa y ruta de navegación
   - Construir algoritmo para la generación del mapa de navegación mediante campos potenciales artificiales.
   - Construir un algoritmo para la elección de ruta mediante frente de onda.
   - Construir un algoritmo para la evasión de los obstáculos móviles e inmóviles.
   - Integrar los algoritmos en un simulador.
3. Implementar la navegación de ruta en un robot.
   - Programar capacidades de navegación del robot.
   - Realizar pruebas de precisión con que el robot llega a la meta.
4. Implementar los algoritmos para le evasión de obstáculos estático en un robot.
   - Programar capacidades de evasión del robot.
   - Realizar pruebas de precisión con que el robot llega a la meta.
5. Implementar los algoritmos para le evasión de obstáculos en movimiento sencillos.
   - Programar capacidades de evasión del robot.
   - Realizar pruebas de precisión con que el robot llega a la meta.

## 1.4. Resultados esperados

Para un robot móvil con ruedas que navega en terrenos exterior se pretende que la navegación y evasión de obstáculos sea tal que:

- A nivel de simulación generar la ruta óptima de navegación en presencia de diversos obstáculos.
- Con el robot físico la evasión de obstáculos estáticos, y de obstáculos dinámicos con movimiento sencillos.
- Minimizar los derrapes en tres distinto terrenos ajustando la velocidad de navegación del robot.
- Minimizar el error entre la posición final del robot y la meta.

## 1.5. Organización del documento

Este trabajo está compuesto por cinco capítulos: en el primer capítulo se ha descrito el problema a tratar, los objetivos de este trabajo, y los resultados esperados. En el segundo capítulo se muestra el estado del arte del problema a estudiar en la tesis, se resume como ha sido estudiado este problema desde sus inicios hasta la fecha, diversos trabajos sobre los problemas de navegación en exteriores y evasión de obstáculos. En el tercer capítulo describimos la solución y métodos planteados para resolver el problema del primer capítulo; se describe como se realiza la locomoción de nuestro robot, la creación de ruta y el mapa de navegación, como realizamos la evasión de los obstáculos y el ajuste de velocidad. En el cuarto capítulo se muestran los resultados obtenidos; a nivel simulación se describe los resultados de la obtención de la ruta óptima en presencia de diferente número de obstáculos, y a nivel físico se presentan tabla de resultados en distintos terrenos mostrando el error de llegar a la meta en una línea recta, el error de llegar a la meta en presencia de obstáculos y por último el error de llegar a la meta con un ajuste de velocidad. En el quinto capítulo tenemos la discusión y la conclusión del trabajo tesis, además de mención el trabajo a futuro para este problema. Por ultimo encontramos los Apéndices que muestran códigos desarrollados para el trabajo y la bibliografía consultada.

# Capítulo 2

# Estado del Arte

*Se explica cómo los robots móviles realizan la navegación y evasión de obstáculos, problemas cuyo estudio data hace más de 25 años. Nos concentramos en robots con ruedas focalizando en el tipo de ruedas, la cinemática y la odometría; se describen las limitaciones y ventajas para construir una plataforma física adecuada para la solución de nuestro problema. Como último punto abordaremos la evasión de los obstáculos, inmóviles y casos sencillo de obstáculos móviles, analizando diferentes técnicas que se han utilizado para realizar las evasiones de obstáculos.*

## 2.1. Navegación

Para que sean eficaces en el mundo real, los robots deben de moverse de manera segura en entornos no estructurados y alcanzar sus metas a pesar a los cambios que presente en su entorno, rara vez son del todo predecibles o conocido por el robot. Las etapas a considerar para el movimiento del robot móvil son:

1. Planificar la ruta de la posición actual al punto meta.
2. Evaluación local de la ruta: existen varias rutas posibles, y se elige la más navegable a partir de una evaluación local del terreno.
3. Capacidad de reacción: basado en la información actualizada de su ruta, el robot elije si continúa o modifica la ruta para navegar sin colisionar.

Los métodos de navegación combinan estas etapas del movimiento para construir la ruta de navegación [26] [17] para el robot. Actualmente existen diversas técnicas para encontrar rutas de navegación, como: métodos del gradiente [28], algoritmos de bug [32], campos potenciales artificiales [37], mencionando algunos de ellos. Los problemas de navegación de un robot, como llegar a la meta. Este un problema de tipo global, pero las rutas se construyen utilizando la información local, considerando la topología del terreno y los obstáculos que obstruyan la ruta de navegación.

En [5] no dice que un robot no debe de estar atado a una ruta de navegación pues podría quedar bloqueado por un obstáculo o en un callejón sin salida. Por esto el robot autónomo debe de ser capaz de explorar su entorno para evitar quedar atrapado. Otro problema que presenta la navegación es el terreno por el cual navega. Uno de los principales problema que encontramos al medir el desplazamiento del robot con odometría es el error que se genera con los derrapes, al girar de las ruedas del robot, durante su recorrido para evadir obstáculos y modificar su ruta. Entre más irregular sea el terreno y más obstáculos encuentren para evadir, mayor dificultad durante la navegación del robot. Para estimar la estimación de la posición del robot se realiza mediante odometría [34] [16].

### 2.1.1. Robots autónomos móviles

Usualmente, un robot, tanto autónomo como no autónomo, percibe su entorno y actúan sobre él, esto es, el robot no actúa sobre una abstracción o modelo, sino directamente sobre el mundo físico; su experiencia del mundo y sus acciones sobre el mismo se producen de forma directa haciendo uso de sus propias capacidades físicas. En el caso de un robot autónomo, la autonomía consiste en actuar sin la supervisión de un agente externo que controle sus movimientos o toma de decisión [25].

Los robots autónomos son entidades físicas con capacidad de percepción sobre un entorno y que actúan sobre el mismo en base a dicha percepciones, sin supervisión directa de otros agentes [25] [30]. Un robot autónomo suele ser móvil, entendiendo por móvil que no se encuentra fijado a una posición y puede desplazarse por su entorno, pero no hay nada que en el principio obligue a ello. A la inversa, un robot móvil no es necesariamente autónomo: existen multitud de robots móviles que son teleoperados en mayor o menor medida.

Navegación de robot autónomo en terrenos exteriores ha sido un área de interés, buscando obtener un alto grado de autonomía para la exploración de misiones en la tierra y otros planetas [3]. Las dificultades de moverse por terrenos exteriores son la topología del terreno y los obstáculos que encontramos en nuestra ruta. Para usar un robot es necesario tener alto grado de autonomía convirtiendo esto en una tarea difícil. Los sistemas de locomoción juegan un papel muy importante en la exploración de superficies ya que se puede mejorar el rendimiento utilizando vehículos con ruedas, gusanos o híbridos.

La navegación autónoma es compleja, la detección y evasión de obstáculo, así como la obtención de información de las características del terreno para no derrapar o volcarse, es requerida. Los datos del medio ambiente deben de ser precisos y procesados rápidamente por el sistema de navegación del robot.

Para que sea exitosa la navegación autónoma de un robot en superficies exteriores, requiere de una selección adecuada de la arquitectura del robot [30], seleccionar los métodos para el reconocimiento y evasión de obstáculos [39] y por ultimo un control de velocidad [2].

El robot utilizado es este trabajo de tesis tiene una cierta autonomía dado que es capaz de:

- tomar su propia decisión a la hora de elegir una ruta de navegación y/o,
- ajustar la velocidad dependiendo de las características del terreno.

Para realizar la elección de la ruta el robot es capaz de detectar un obstáculo, evadirlo y tomar la medida necesaria para continuar con su ruta a la meta. Para ajustar la velocidad se procesas una imagen de las características del terreno por el cual navega el robot.

### 2.1.2. Robots móviles con ruedas

Las ruedas de un robot son consideradas el mecanismo de locomoción más popular de un robot móvil, debido a que se puede lograr buena eficiencia y es un mecanismo relativamente fácil de implementar [41].

Generalmente un robot está diseñado con ruedas, esto debido a que todas las ruedas mantienen un contacto con el suelo todo el tiempo. De esta es necesario solo tres ruedas para garantizar un balance estable; aunque un robot con dos ruedas puedes ser estable [39] [11]. Cuando tenemos más de tres ruedas es necesario utilizar un sistema de suspensión para garantizar que todas las ruedas estén en contacto con la superficie en terrenos desiguales [43].

En lugar de preocuparse por el equilibrio del robot, la investigación se enfoca en el problema de la tracción y estabilidad; movilidad y control. Un robot con ruedas puede tener suficiente tracción y estabilidad para poder cubrir todos los terrenos deseados; además de que cualquier configuración de ruedas permite realizar un control de velocidad en el robot [23] [22].

Existen diferentes variedades de ruedas, en la Figura 2.1 mostramos los cuatro principales tipos. La elección del tipo de la rueda es importante ya que tiene efecto en la cinemática del robot. La rueda estándar y la rueda de castor tienen un eje primario de rotación y una alta direccionalidad. El movimiento en diferentes direcciones debe de ser dirigido a lo largo del eje vertical. La principal diferencia entre estas dos ruedas es que la rueda estándar puede lograr un movimiento en una dirección conciso sin correr efectos adversos, mientras que la rueda de castor gira alrededor de un eje

conjunto, causando una fuerza puede causar devociones al distribuirla al chasis del robot durante el movimiento [9] [7].

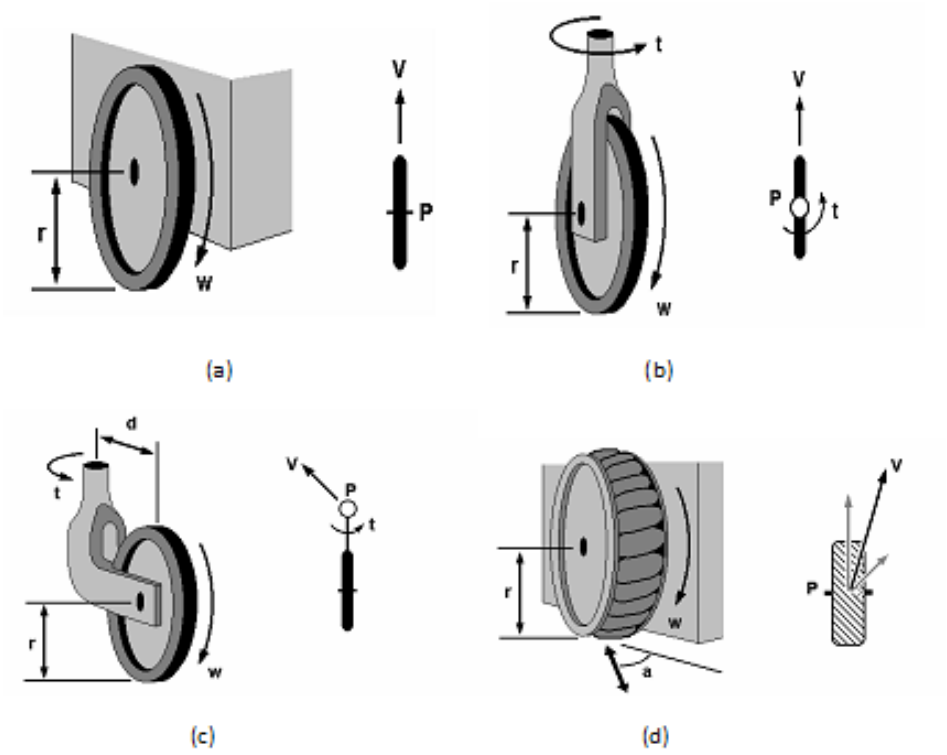


Figura 2.1: Tipos de Ruedas: (a) Rueda fija, (b) Rueda orientada centrada,(c) Rueda orientada descentrada (Rueda de castor), (d) Rueda sueca [10]

La rueda sueca y la rueda esférica son dos diseños que no están tan limitando por la direccionalidad de la rueda convencional estándar [14]. Las funciones de la rueda sueca son como las funciones de una rueda estándar, pero nos ofrece una baja resistencia en otra dirección. Los rodillos unidos alrededor de la circunferencia de la rueda son pasivos y el eje primario de la rueda sirve como la articulación solamente al movimiento [42]. La principal ventaja de este diseño es que, aunque la rotación de la rueda sueca se alimenta a lo largo del eje principal, la rueda cinemática mente puede moverse con muy poca fricción a lo largo de muchas trayectorias posibles, y no solo hacia adelante y hacia atrás [15] .

### 2.1.3. Cinematica

La cinemática estudia los movimientos aislados desde las fuerzas y torques asociados con el movimiento linear y angular respectivamente. Además de las derivadas del movimiento con respecto al tiempo, es decir, velocidad y aceleración. En pocas palabras la cinemática en robótica puede ser interpretada como los objetos en movimiento mecánico de todo tipo [30].

En robot con ruedas pretende estimar nuevas posiciones a partir del conocimiento de las acciones motrices y los sensores internos. Los movimientos del robot, puede ser:

- Directa: dada la posición inicial y los movimientos realizados, determina la posición final del robot.
- Inversa: dada la posición inicial y final deseadas, obtenemos la serie de movimientos que el robot debe de realizar.

Como lo describimos anteriormente existen diferentes tipos de ruedas (tracción y dirección) como diferentes tipos de configuraciones otorgándonos diferente propiedades cinemáticas. Un robot móvil normalmente tiene tres grados de libertad Figura 2.2 respecto a una referencia: posición en el plano $(X, Y)$ y orientación $(\Theta)$, Idealmente de inicie el robot debe de poder moverse a cualquier posición y orientación $(X, Y, \Theta)$ [17].

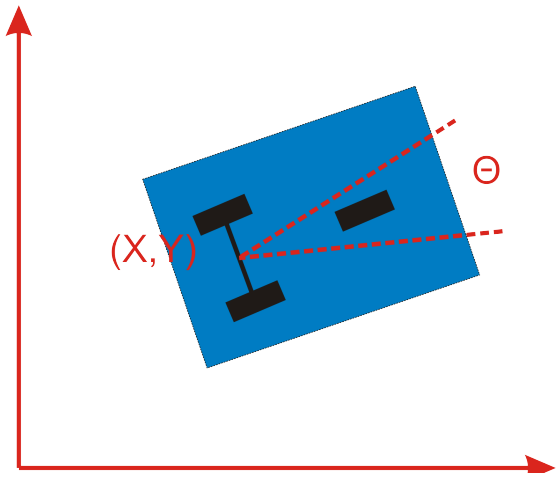


Figura 2.2: Grados de libertad de un robot móvil diferencial

Todo robot tiene restricción debido a la configuración de las ruedas, estando limitando en los movimiento que podemos realizar como se muestran en la Figura 2.3.

Se tienen dos principales tipos de restricciones que se encuentran íntimamente ligados con las configuraciones de las ruedas principalmente las motrices. Las restricciones son:

- Holonómicas: los grados de libertad del robot están desacoplados. Como los robots diferenciales y síncronos: se puede desacoplar la posición de orientación (rotando sobre su eje).
- No-Holonómicas: los grados de libertad están acoplados. En el caso de los triciclos y los carro: para dar vuelta debe moverse hacia enfrente o hacia atrás siendo más complejo llegar a la posición final deseada.

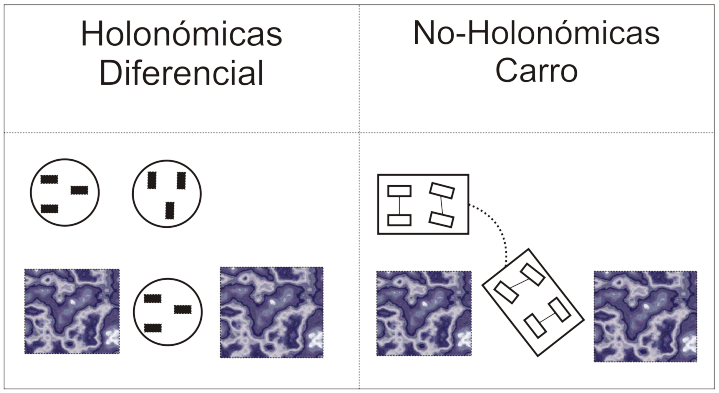


Figura 2.3: Restricciones cinemáticas para un robot móvil

La configuración de las ruedas en un robot móvil nos otorga diferentes grados de libertad para el cálculo de la cinemática, existiendo diferentes tipos de configuraciones [16]. Las cuales se muestran a continuación:

- Diferencial

  Este es uno de los esquemas más sencillos, básicamente consiste de dos ruedas en un eje en común, donde cada rueda se controla independientemente, donde sus movimientos pueden ser:

  - Línea recta.
  - En arco.
  - Vuelta sobre su propio eje.

  Este esquema además de las dos ruedan utiliza una o dos ruedas adicionales tipo castor para mantener el balance, esta forma tiene diferente nombre dependiendo de las ruedas: Con tres ruedas se denomina triangulo. Este esquema puede presentar problema de estabilidad, y con cuatro ruedas se denomina diamante. Aquí puede ocurrir la perdida de contacto de alguna rueda con la superficie teniendo así una predica de tracción, esto hace que requiera de un sistema de suspensión (Ver Figura 2.4) .

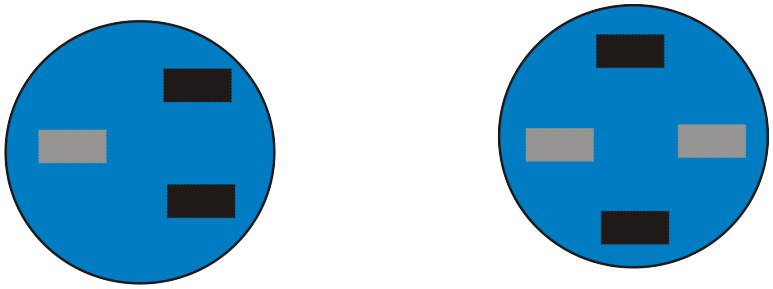

Figura 2.4: Arreglo de ruedas diferencial

- Síncrono

  Las ruedas se mueven de forma síncrona, es decir al mismo instante. El movimiento síncrono es un caso particular del diferencial, donde cada eje se mueve en forma dependiente para dar vuelta y avanzar. Las ruedas están ligadas de forma tal que siempre apunta en la misma dirección y para dar vuelta giran las ruedas sobre su eje vertical, por lo que la dirección de la estructura se mantiene por lo que se requiere de un mecanismo para mantener el frente estructural del robot en la dirección de las ruedas (Ver Figura 2.5).

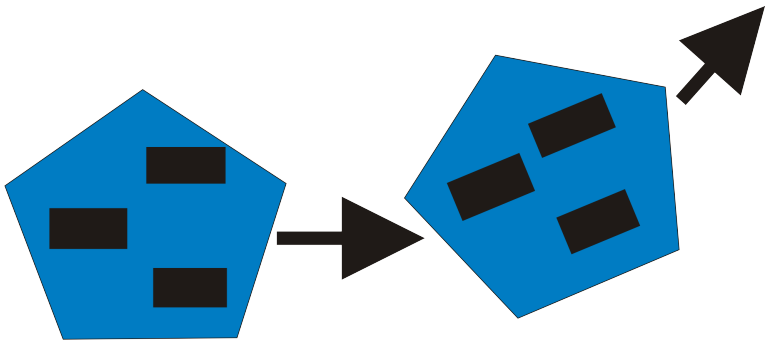

Figura 2.5: Arreglo de ruedas síncrono (giro a la izquierda)

- Triciclo

  Los triciclos tiene dos ruedas fijas que le dan la tracción, además cuenta de un rueda para la dirección normalmente no tiene tracción. Este sistema tiene buena estabilidad y simplicidad mecánica, tiene facilidad para ir recto y cinemática es más compleja (Ver Figura 2.6).

Figura 2.6: Arreglo de ruedas en forma de triciclo.

- Carro

  Los sistemas de carro son similares al triciclo solo que cuentan con dos ruedas de tracción y dos rueda para dirección. Tienen una mayor complejidad mecánica que el triciclo por el acoplamiento entra las dos ruedas de dirección. Sus principales ventajas son la buena estabilidad y facilidad de ir derecho. La desventaja es una complejidad cinemática (Ver Figura 2.7).

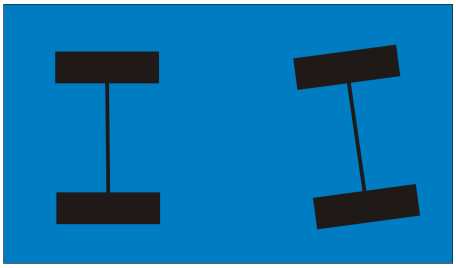

Figura 2.7: Arreglo de ruedas en forma de triciclo Carro.

- Omnidireccional

  Los sistemas omnidireccionales cuentan con tres ruedas colocadas a 120° tal como se muestra en la figura. Donde cada una de las ruedas tiene la capacidad de girar en ambos lados y se logra un control lineal más simplificado que en el caso del robot diferencial (Ver Figura 2.8).

Figura 2.8: Arreglo de ruedas omnidireccional

### 2.1.4. Odometría

Los robots móviles usan la odometría para estimar y no determinar su posición relativa a su posición inicial. Es bien sabido que la odometría proporciona una buena precisión a corto plazo. Sin embargo la idea fundamental de la odometría es la integración de información incremental del movimiento a lo largo del tiempo, lo cual conlleva una inevitable acumulación de errores, esta acumulación causa grandes errores en la estimación de la posición, los cuales van aumentado proporcionalmente con la distancia recorrida por el robot. A pesar de estas limitación, muchos investigadores están de acuerdo en que la odometría es una parte importante del sistema de navegación de un robot, y que debe usarse como medida del posicionamiento absolutas para proporciona una estimación de la posición más fiable.

La odometría se basa en ecuaciones simples que se pueden implementar fácilmente y que utilizan datos de enconders situación en las ruedas del robot. Sin embargo, la odometría también está basada en la suposición de que las revoluciones de las ruedas pueden ser traducidas en un desplazamiento lineal relativo al suelo.

La odometría es uno de los métodos más utilizados para obtener la distancia que recorre un robot desde su posición hasta un punto meta. La medición precisa es difícil pues los motores no siempre están sincronizados, y/o el terreno es irregular y el robot puede tomar una dirección errónea en su navegación del punto meta. Existen diversas técnicas que nos ayudan a mantener con la sincronización de los motores algunos de estos métodos son controladores PID [40] o controladores difusos [6]. Ejemplo, si el motor A gira más rápido que el motor B, se disminuye la velocidad del motor A.

Los errores de precisión en la medición odométrica pueden agruparse errores sistemáticos y errores no sistemáticos. Entre los errores sistemáticos destacan:

- Los diámetros de las ruedas no son iguales.
- La medida de los diámetros de las ruedas difieren del diámetro de fábrica de las ruedas.
- Mal alineación de las ruedas.
- Resolución discreta del encoder.

Entre los errores no sistemáticos se encuentran:

- Desplazamiento en suelos desnivelados.
- Desplazamiento sobre objetos inesperados que se encuentren en el suelo.
- Patinaje de las ruedas debido a:
    - Suelo resbaladizos.

- Sobre-aceleración.
- Derrapes (debido a una rotación excesivamente rápida).

Los métodos basados en insectos (bug) utilizan este tipo de navegación, por ejemplo, una hormiga desde su hormiguero, punto $S$ (inicial), va a un punto $P$ donde hay comida; la trayectoria que sigue la hormiga, comúnmente, no es en línea recta sino con desviaciones debido a la topología del terreno por el que navega, los obstáculos que se encuentra [32], como se muestra en la Figura 2.9.

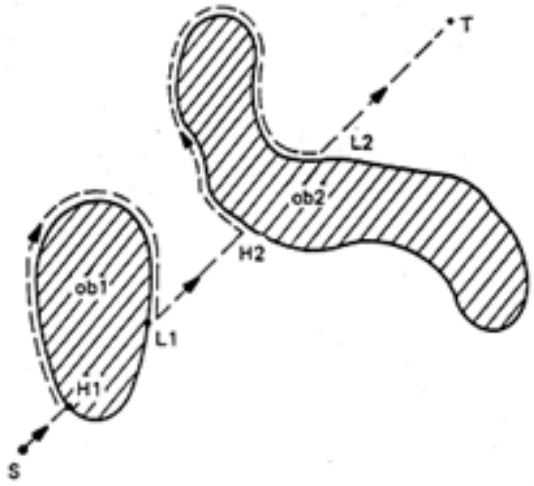


Figura 2.9: Navegación realizada por algoritmos tipo *Bug* [32]

Aunque estos métodos de navegación son simples y eficientes, no todos contemplan la existencia de obstáculos en la ruta. Por lo cual si el robot encuentra un obstáculo es necesario que cuente con un mecanismo capaz de evitarlo y continuar con su ruta hacia la meta, como lo hace el algoritmo de insectos.

### 2.1.5. En terrenos exteriores

Los robots autónomos, son empleados para exploración de terrenos desconocidos, ya sea para explotación terrestre o planetaria, donde podemos encontrar diversos terrenos: polvo, tierra suelta y rocas. Para estas misiones es necesario utilizar robots con un alto grado de autonomía, sin embargo existen dificultades al controlarlos remotamente. El "Spirit and Opportunity roves", necesitaban 26 minutos en recibir y enviar instrucciones hacia y desde la tierra por ejemplo [3].

La navegación autónoma a lo largo de terrenos exteriores es compleja debido a la necesidad de identificar las características de los terrenos para evitar derrapes y la detección de obstáculo para evitar colisionar [2]. Actualmente, al pasar a un ambiente exteriores, un robot autónomo debería ser capaz de mantener una velocidad

de control. Sin embargo, el control de velocidad de un robot con respecto a la características del terreno ha sido mucho menos estudiada, por su poca eficiencia y seguridad. El control de velocidad con respecto a las características del terreno es un requisito para poder mantener al robot alejado del riesgo de volcaduras o derrapes debido a las superficies irregulares que navega [24]: La velocidad del vehículo debe de ser actualizada conforme se obtiene la información de las características del terreno, garantizando una navegación segura.

Debido a que en ambientes externos el terreno puede ser desconocido, el robot debe de ser capaz de adaptarse a los nuevos terrenos encontrados en su ruta con la información que ha obtenido a priori. Debido a que si no es capaz de reconocer el terreno puede llegar a perder movilidad o puede ponerse en riesgo [4]. Una de las principales habilidades que el robot debe de poseer al navegar en terrenos exteriores es distinguir entre los distintos tipos de terrenos, esto se logra mediante un paradigma de clasificación bajos redes neuronales, donde al sistema es entrenado con un conjunto de terrenos a priori.

Actualmente existen diversos métodos para realizar la navegación en exteriores, si bien estos métodos son muy utilizados en interiores esto no impide que puedan ser utilizados en exteriores, algunos de ellos son los siguientes: un método clásico y aun utilizado es el campo potencial artificial [27], basados en la propagación de ondas para obtener información de los valores de navegación, estos valores pueden estar basados en consideraciones topológicas de los terrenos, distancias, o cualquier otro valor importante [8]; Los GPS son herramientas empleadas para obtener la posición en la que se encuentra el robot basado en la triangulación satelital [1], los GPS en la actualidad son una herramienta muy confiable para obtener la posición en cualquier parte además de que funcionan muy bien en terrenos abiertos [36]. Por ultimo tenemos los métodos basados en la visión por computadora [12], existen diversos métodos para realizar la navegación como: odometría visual [34], basados en característica [20], métodos basados en apariencias [31], entre otros.

Los métodos basados en odometría visual, son los métodos basados en puntos importantes en este las esquinas de una imagen o escena. Un método muy utilizado para obtener la posición del robot es *Harri's Corner* [34]; este método funciona de tal manera que obtiene las una característica importante de un objeto, en esta caso las esquinas, por ejemplo de una mesa, una puerta, etc. De esta manera puede obtener una posición aproximada de donde se encuentra el robot respecto a un objeto seleccionado.

Otras técnicas como métodos basados en apariencias [31] o características del terreno [20]; buscan obtener información directa del terreno por donde navega el robot, esto para poder determinar si es seguro navegar. Estos métodos se apoyan directamente de la información estructural del robot para determinar si los terrenos son navegables o no, esto se realiza empleando información previamente entrena en el

robot para de esta manera minimizar el daño que se pueda causar por volcaduras o caídas del robot. La forma en que se eligen los terrenos se desarrolla mediante métodos de aprendizaje como lo son las redes neuronales y/o lógicas difusas [2] [4] [20].

Para poder realizar la navegación con un robot móvil es necesario, primero elegir una buena configuración de ruedas, y luego elegir un método adecuado para realizar la navegación. Pero qué hay de los obstáculos que se encuentren en nuestro camino, aunque alguno de los métodos presentados en esta sección, son capaces de evadir obstáculos no son los únicos métodos que existen; además de que los obstáculos encontrados en la ruta pueden tener movimiento, por este motivo en la siguiente sección se habla de ellos.

## 2.2. Evasión de obstáculos

A través de los años evadir un obstáculo se ha vuelto un problema primordial a resolver; debido a lo entorno en que un robot navega existen obstáculos que lo bloque en su navegación a la meta. Únicamente construir rutas de navegación para llegar a la meta no es insuficiente, ya que el robot puede encontrar con obstáculos en su ruta de navegación; siendo primordial encontrar una manera de evadirlos [33]. La evasión de obstáculos en entornos exteriores es considerada más difícil que en entornos interiores. Clasificándolos en dos categorías:

- En la primera categoría, el movimiento del obstáculo es desconocida para el robot.
- La segunda categoría, el movimiento del obstáculo es conocido para el robot.

Algunos trabajos, agregar a estas categorías un factores de incertidumbre como la predicción de la trayectoria del obstáculo o el historial de recorrido del obstáculo para conocer su comportamiento [27]. Hay soluciones que asumen el conocimiento a priori de la trayectoria del obstáculo o de la selección de la trayectoria usando solamente información del obstáculo y determinar la velocidad a la que robot avanza en la trayectoria, o soluciones que requiere que el robot mantenga una cierta distancia de su trayectoria.

Un robot debe de navegar satisfactoriamente a través de los obstáculos, alcanzar la meta, y ser eficiente; evitando colisionar con objetos como una roca, un arbusto, etc., sino que además debe ser capaz de evitar caer en un hoyo o viajar sobre terrenos que puedan causar que se vuelque el robot.

### 2.2.1. Controlador PID (Proporcional, Integral, Derivativo)

El controlado bajo PID, Figura 2.10, es un mecanismo de control por re-alimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obte-

ner, para aplicar una acción correctora que ajusta el proceso. El cálculo del control PID se da en tres parámetros distinto:

- El valor proporcional que determina la reacción del error actual.
- El integral genera una corrección proporcional a la integral del error.
- El derivativo determinando la reacción del tiempo en el que el error se produce.

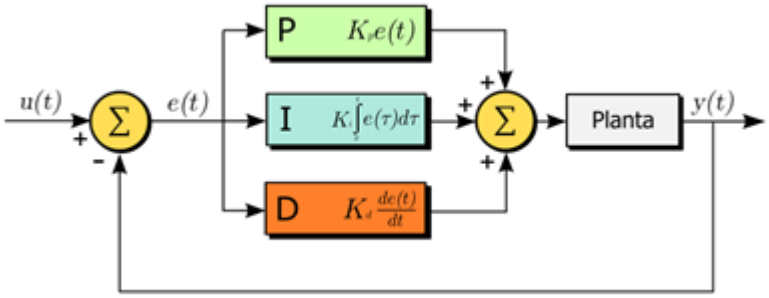


Figura 2.10: Controlador PID

Para el funcionamiento del controlador PID necesitamos:

- Un sensor, que determine el estado del sistema (sensor infrarrojo, acelerómetro, etc.).
- Un controlador, que genera la señal que gobierna al actuador.
- Un actuador, que modifique al sistema de manera controlada (motor, válvula, etc.).

El sensor es el encargado de proporcionar una señal lógica o digital al controlador la cual representa el punto actual en el que se encuentra nuestro sistema. El controlador lee una señal externa que representa el valor al que se desea llegar, con estas dos señal el controlador se encarga de obtener el error estimado, para posteriormente ser utilizada por los tres componentes del controlador PID. La señal resultante, antes de ser pasada al actuador debe de ser transformada para su compatibilidad [40].

### 2.2.2. Campos potenciales artificiales

Una técnica muy utilizada son los campos potenciales artificiales [27], este método se basa en simulan el comportamiento de los campos de fuerzas, donde estos campos pueden tener fuerzas atrayentes o repulsivas. La idea es otorgar al obstáculo una fuerza repulsiva para que el robot móvil sea repelido por el obstáculo actuando de manera como una colina, mientras que al punto meta le otorgamos una fuerza atractiva en este caso el robot es atraído hacia ese punto actuando como un valle [35] [37]

[44]. Para poder identificar una fuerza atrayente o una repulsiva se hace mediante la asignación de valores en el mapa de navegación si la fuerza es atrayente el valor es pequeño mientras que si es repelente el valor es mayor.

En la Figura 2.11 se observa la gráfica del campo potencial artificial; donde el punto meta se encuentra en la parte más baja de la gráfica cercana al cero, teniendo una fuerza atrayente, mientras el obstáculo en la ruta se encuentra en la parte central de la gráfica, representado por una colina otorgándole una fuerza repulsiva.

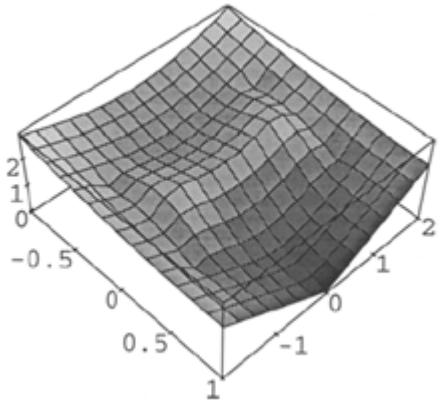


Figura 2.11: Gráfica ejemplo de un Campo potencial artificial

Konolige [28] en su trabajo realiza la construcción de su mapa de navegación por medio de campo potenciales artificiales basados en la propagación de onda. A pesar de contar con un mapa con la información de donde se encuentra la meta y los obstáculos, se debe de emplear algunos métodos para construir la ruta que nos llevara a la meta y la forma de evadir obstáculos. Para esto Konolige utiliza el métodos del gradiente, con esto se puede obtener un vector de dirección para el movimiento del robot construyendo una ruta, y debido a la naturaleza de los campos se evaden los obstáculos.

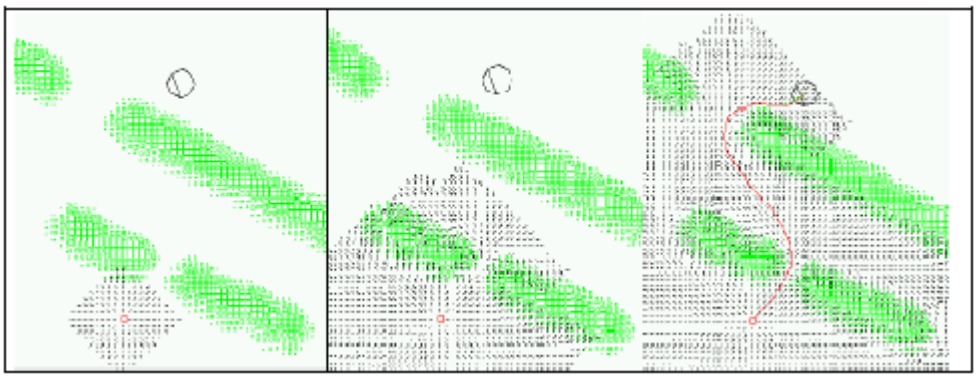

Figura 2.12: Método de navegación por gradiente, propuesto por K. Konolige [28].

En la Figura 2.12 se observa el funcionamiento del método propuesto por Konolige, divido en tres recuadros e iniciando de izquierda a derecha encontramos el punto meta marcado con un circulo y un punto inicial. En el primer recuadro tenemos en el primer tiempo la propagación del gradiente, esto se realizar para ir conociendo las direcciones y el entorno que rodea al robot, en el segundo recuadro observamos la propagación más avanzada y aquí podemos observar como el método evade los obstáculos, y en el último recuadro tenemos la propagación final del gradiente y la ruta que el robot debe de seguir para llegar a la meta.

### 2.2.3. Dependiente de las características del terreno

Otro método que podemos encontrar para realizar la evasión son los métodos basado en visión por computadora como es el caso de los basado en las características de los terreno [20]. El trabajo de Howard podemos observar la construcción de un mapa de navegación a partir de las características de los terrenos por la cuales un robot puede navegar de manera segura sin comprometer su integridad, delimitando las áreas riesgos como no navegables o poco navegables, y las áreas seguras como navegables, esto es determinado mediante el análisis de las características de los terrenos y las características del robot.

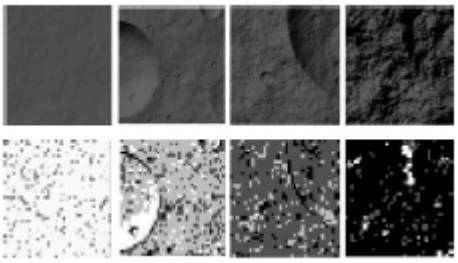

Figura 2.13: Métodos basados en características del terreno de navegación [32].

Como podemos observar en la Figura 2.13 la navegación del robot está limitada por sus capacidades para poder transitar algunos terrenos. En la Figura vemos en el primer par de recuadros encontramos un terrenos llano en el cual navegar es factible y seguro, por lo cual este terrenos es marcado por un color blanco de navegable. En los recuadros dos y tres vemos terrenos irregulares que hay que considerara para ver si es seguro para el robot navegar en ellos; cómo podemos ver por el análisis del método los recuadro blancos se van escurriendo indicando que para las capacidades del robot ya no son totalmente adecuada para navegar esos terrenos, siendo posible transitarlos pero con limitaciones de velocidad o movilidad. Mientras que en el último par de imágenes observamos un terreno irregular haciendo imposible la navegación del robot sin que este sufra daños estructurales, por los cual los recuadro se vuelven negros.

Los métodos mencionados anteriormente, se basan en una navegación a nivel local, debido a que la construcción de un mapa a nivel global es muy costos en términos de recursos para un robot móvil por sus limitados recursos de memoria y procesamiento. Además a nivel local podemos estar al tanto de los cambios que puedan ocurrir alrededor del robot.

Estos métodos para la evasión de obstáculos son confiables y utilizados, pero como se mencionó en las basadas en características del terreno, el robot está limitado por sus capacidades estructurales. De esta manera navegar sobre terrenos desconocido es una problemática debido a que el robot se puede topar en algún momento con un terreno irregular y peligros, por lo cual se debe de garantizar la seguridad del robot conforme navega, y no solamente evitar las colisiones. Para otorgar una autonomía completa al robot a la hora de navegar por terrenos exteriores y con obstáculos desconocidos.

# Capítulo 3

# Navegación y evasión de obstáculos

*Se desarrolla la metodología de la solución propuesta para la navegación del robot: se introducen los modelos de locomoción o cinemática utilizados, y asimismo, se desarrolla la construcción del mapa con los algoritmos para obtener la información del terreno y construir las rutas de navegación. En la penúltima sección explicamos cómo se realiza la evasión de los obstáculos tanto inmóviles como algunos casos sencillos de obstáculos móviles. Por último se aborda la construcción e incorporación del ajuste de velocidad en la navegación del robot.*

## 3.1. Modelo de locomoción

El robot con ruedas basa su movilidad en locomoción diferencial por lo que no tiene ruedas directrices, y el cambio de dirección se realiza modificando la velocidad relativa de las ruedas izquierda o derecha. En teoría esta es la mecánica más fácil de construir, únicamente se necesitan ruedas de tracción, ya que la direccionalidad se consigue con la diferencia de velocidades (y sentidos) de estas ruedas. Para darle mayor estabilidad se suelen usar una o ruedas, además de las motrices, para distribuir el peso del robot de manera adecuada impidiendo que este se incline. Las ventajas son, un sistema barato, fácil de implementar y un diseño simple. El modelo de locomoción utilizado se muestra en la Figura 3.1.

Sin embargo, esto puede dar problemas de pérdida de tracción de las ruedas en pistas irregulares; los problemas son un control difícil y la desviación en la trayectoria a la meta, y por tanto requerimos un control de precisión capas de minimizar este problema.

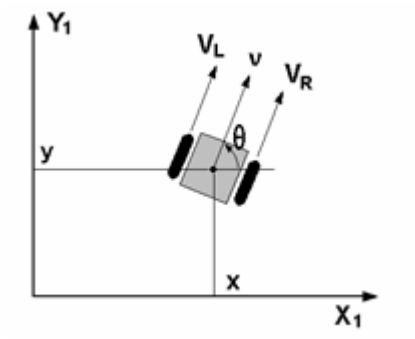


Figura 3.1: Modelo de locomoción diferencial para vehículos con ruedas

El modelo de locomoción diferencial nos permite conocer la distinta información de nuestro robot, como la velocidad de giro de cada una de las ruedas motrices, el ángulo de vuelta de nuestro robot, así como la estimación de la posición en la cual se encuentra. El modelo de la cinemática diferencial se muestra en la Figura 3.2.

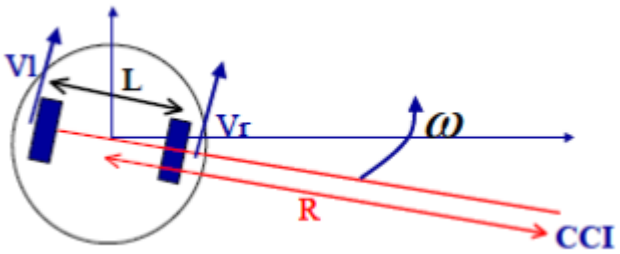


Figura 3.2: Ecuaciones para la Cinemática Diferencial

$$R = \frac{L}{2}\frac{V_r + V_l}{V_r + V_l}, \omega = \frac{V_r + V_l}{L}, \tag{3.1}$$

$$V_r = \omega \times R - \frac{L}{2}, V_l = \omega \times R + \frac{L}{2} \tag{3.2}$$

Dónde:

$\omega$: Velocidad angular
$R$: Radio de giro $L$: Distancia entre ruedas
$V_r$: Velocidad rueda derecha $V_l$: Velocidad rueda izquierda

## 3.2. Generación de mapa y rutas

La construcción del mapa de navegación se basa en asignar costos de las celdas en la ruta para llegar a la meta. La asignación de costos se realiza mediante la pro-

pagación de onda desde la posición actual del robot hacia la meta o posición final, construyendo una matriz de representación del tamaño NxM, representando los puntos desde el inicial al punto meta.

El mapa de navegación representa el área global, y servirá como bitácora de lo ocurrido durante toda la ruta de navegación. La representación de un obstáculo también está contemplada otorgando un valor para su representación. Una vez encontrado un cambio en el mapa este se actualizará para generar las nuevas rutas de navegación. El algoritmo 1, nos muestra cómo se realiza la construcción y expansión de la onda a nivel simulación donde únicamente obtenemos los valores de propagación, a nivel simulación nosotros podemos agregar diferentes obstáculos en diferentes posiciones.

**Algorithm 1** Creación del mapa de Navegación

**Entrada:** Onda = 2, posiciones Iniciales $i = P_{orix}, j = P_{oriy}$, mapa de navegación $Mapa[N][M]$, posición inicial Mapa $P_a = Mapa[i][j]$, posiciones destino $ii = P_{dex}, jj = P_{desy}$

**Salida:** $Mapa$

**while** $Mapa[P_{dex}][P_{desy}] \neq 0$ **do**
  **while** $vecindad P_a exista 0$ **do**
    $vecindad = P_a + 1$
  **end while**
  $P_a = Mapa[i][j+1]$
  **if** $j = ancho$ **then**
    $j = P_{oriy}$
    $i = i + 1$
  **end if**
**end while**
**return** *mapa.*

El mapa global de navegación con obstáculos construido a partir del algoritmo 1 se muestran en la Figura 3.3 contiene un obstáculo marcado por el número 45 (color amarillo), además podemos visualizar la ruta óptima de navegación obtenida marcada por un tono azul. También, en presencia de un obstáculo, se observa la propagación de la onda en la asignación de los valores y para el calcular la ruta óptima.

### 3.2.1. Rutas

La creación de las rutas de navegación se basa en la sumatoria de los costos de las celdas que recorremos en el mapa desde el punto inicial al punto meta.

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 42 | 40 | 38 | 38 | 36 | 35 | 35 | 33 | 34 | 31 | 31 | 29 | 28 | 29 | 28 | 24 | 23 | 25 | 21 | 21 |
| 40 | 41 | 38 | 36 | 35 | 36 | 36 | 34 | 32 | 31 | 30 | 30 | 27 | 27 | 28 | 26 | 25 | 25 | 22 | 23 | 23 |
| 41 | 38 | 39 | 37 | 35 | 34 | 33 | 31 | 31 | 31 | 31 | 28 | 29 | 27 | 26 | 23 | 25 | 22 | 20 | 19 | 21 |
| 37 | 38 | 37 | 36 | 35 | 35 | 31 | 31 | 32 | 31 | 30 | 29 | 26 | 26 | 25 | 22 | 22 | 22 | 21 | 21 | 21 |
| 38 | 36 | 37 | 36 | 35 | 31 | 30 | 32 | 28 | 29 | 27 | 26 | 26 | 22 | 23 | 24 | 22 | 21 | 19 | 20 | 19 |
| 38 | 34 | 33 | 33 | 34 | 30 | 29 | 31 | 27 | 26 | 26 | 24 | 23 | 25 | 23 | 21 | 19 | 20 | 20 | 18 | 17 |
| 35 | 33 | 32 | 34 | 32 | 31 | 29 | 27 | 28 | 26 | 24 | 24 | 24 | 22 | 23 | 20 | 18 | 17 | 19 | 15 | 18 |
| 34 | 34 | 34 | 30 | 32 | 31 | 28 | 26 | 25 | 27 | 24 | 25 | 22 | 20 | 21 | 19 | 20 | 16 | 15 | 15 | 15 |
| 34 | 33 | 30 | 30 | 29 | 30 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 19 | 17 | 17 | 14 | 14 |
| 31 | 31 | 31 | 31 | 30 | 27 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 17 | 15 | 13 | 15 | 12 |
| 32 | 32 | 29 | 27 | 26 | 26 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 16 | 16 | 13 | 13 | 13 |
| 30 | 28 | 30 | 27 | 26 | 25 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 15 | 15 | 13 | 12 | 11 |
| 29 | 27 | 26 | 25 | 25 | 24 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 13 | 13 | 12 | 11 | 12 |
| 28 | 27 | 25 | 27 | 26 | 23 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 13 | 10 | 9 | 11 | 11 |
| 29 | 27 | 26 | 25 | 25 | 24 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 11 | 12 | 11 | 10 | 10 |
| 26 | 27 | 25 | 25 | 22 | 21 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 10 | 8 | 7 | 9 | 7 |
| 27 | 25 | 22 | 21 | 21 | 19 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 9 | 10 | 8 | 8 | 5 |
| 25 | 25 | 23 | 21 | 21 | 18 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 45 | 8 | 7 | 8 | 4 | 7 |
| 22 | 23 | 23 | 22 | 18 | 17 | 19 | 18 | 16 | 14 | 14 | 12 | 13 | 11 | 11 | 8 | 6 | 5 | 4 | 6 | 6 |
| 22 | 22 | 22 | 20 | 19 | 16 | 18 | 14 | 14 | 14 | 11 | 10 | 12 | 8 | 7 | 7 | 6 | 7 | 3 | 3 | 2 |
| 24 | 20 | 20 | 19 | 19 | 16 | 15 | 17 | 16 | 13 | 13 | 11 | 9 | 10 | 9 | 7 | 5 | 4 | 5 | 3 | 1 |

Figura 3.3: Mapa de navegación construido con el algoritmo 1

$$P = P_1, P_2, P_3, P_4, \ldots \tag{3.3}$$

$P$ es el conjunto de todos los puntos que visitamos desde el punto inicial al punto meta.

$$N_k = \min_i C(P_i) \tag{3.4}$$

$N_k$ es la ruta o rutas con el mejor costo de navegación encontrado. Para obtener las rutas de navegación optimas, obtenemos primeramente todas las posibles rutas por la cuales podemos llegar a la meta. Una vez obtenidas las rutas, buscamos las de menor costo posible.

$$P_k = \sum_i C_{(x,y)} + \sum_i A(C_{(x,y)}, C_{(x+i,y+j)}) \tag{3.5}$$

$P_k$ son los diferentes trayectos que podemos encontrar con base en el costo de celda más la distancia que existe del punto en donde se encuentra el robot al siguiente punto.$C_(x,y))$ nos muestra los costos de las celdas que se han visitado en la trayectoria de navegación. Este costo está dado por la matriz del mapa global; aquí podemos ver que los valores de los obstáculos elevarán los valores del costo de navegación por tal motivo se descartarán para no ser transitados. $A(C_(x,y)), C_(x+i,y+j)))$ es el valor de distancia que existe entre los puntos adyacentes a la posición actual en la que se encuentra el robot. Esta distancia junto con el valor de navegación nos ayudará a

tomar la siguiente dirección y posición a la que se desea avanzar el robot. Para la obtención de la distancia planeamos utilizar las siguientes distancias:

Distancia Euclidiana

$$d(p_1, p_2) = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)} \tag{3.6}$$

Distancia Manhattan

$$d(p_1, p_2) = |(x_1 - x_2)| - |(y_1 - y_2)| \tag{3.7}$$

La utilización de una distancia como un parámetro nos ayudará a elegir la distancia más corta que existe entre el robot y un punto vecino, además el valor de distancia funcionara como un parámetro extra para facilitar la elección del camino más corto. La distancia Euclidiana equivale a la longitud del segmento de recta trazado entre los dos puntos, como se ilustra en la Figura 3.4, mientras que la distancia Manhattan es la representación en un plano de calles de una ciudad. Mostrado en la Figura 3.4.

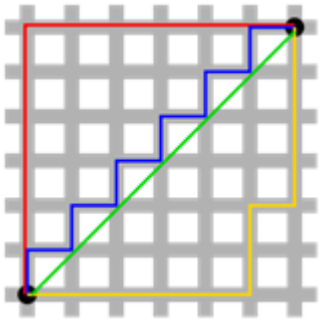

Figura 3.4: Rutas con distancias Euclidiana y Manhattan

Ahora refiriéndonos a los puntos vecinos a la posición actual del robot podemos definir la vecindad como el entorno que rodea al robot, respetado en el mapa como las celdas vecinas de la posición actual del robot, obteniendo dos principales vecindades que son las siguientes:

Vecindad de cuatro. Esta vecindad contempla solamente cuatro vecinos, dos horizontales y dos verticales, y deja huecos en las diagonales (Ver Figura 3.5).

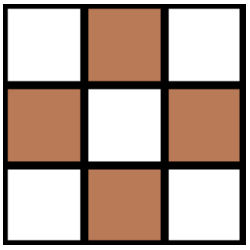

Figura 3.5: Vecindades de cuatro localidades

Vecindad de ocho. La vecindad de ocho permite conocer el entorno de manera más completa, ya que además de considerar las cuatro vecindades anteriores también considera las diagonales teniendo una vista más completa del entorno que al robot rodea (Ver Figura 3.6).

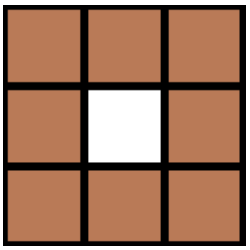

Figura 3.6: Vecindades de ocho localidades

Para nuestro problema utilizaremos la vecindad de ocho debido a que nuestro robot puede tomar la decisión de moverse ya sea de manera horizontal, vertical o diagonal dependiendo de los valores correspondientes al mapa, la distancia y las características del terreno.

### 3.2.2. Costo de celda

El costo de la celda de navegación son los pesos o valores de las celdas que recorre el robot hasta el punto meta. $C(x, y)$ y es obtenida como describiremos a continuación: el primer valor de costo será obtenido de la propagación del frente de onda en el mapa de navegación y será nuestro valor de referencia para realizar la navegación.

$$C_{(x,y)} = \text{mín}\,(ValorVecinoCPA) \tag{3.8}$$

La configuración de los campos potenciales y en especial la propagación del frente de onda (Ver Figura 3.7), ayuda a mantener un rumbo hacia la meta, avanzando siempre de manera frontal o lateral y nunca hacia atrás. Sin embargo, en este trabajo, el costo obtenido del campo potencial no es suficiente, ya que para elegir una

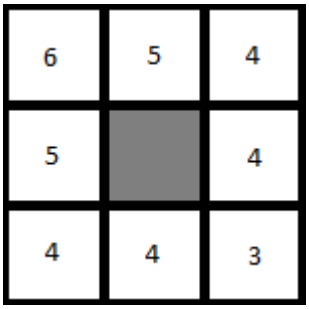


Figura 3.7: Costo de celda de navegación obtenida por campos potenciales artificiales

ruta óptima de navegación el robot debe considerar no solo el hecho de llegar a la meta, sino también la distancia que debe de recorrer para poder minimizar el costo del camino.

En la Figura 3.8 se observan las distintas rutas que un robot móvil puede tomar para llegar a la meta. En este trabajo se busca tomar la ruta más corta, agregando a la ecuación de costo la variable de distancia local, es decir la distancia entre las vecindades, y la ecuación queda:

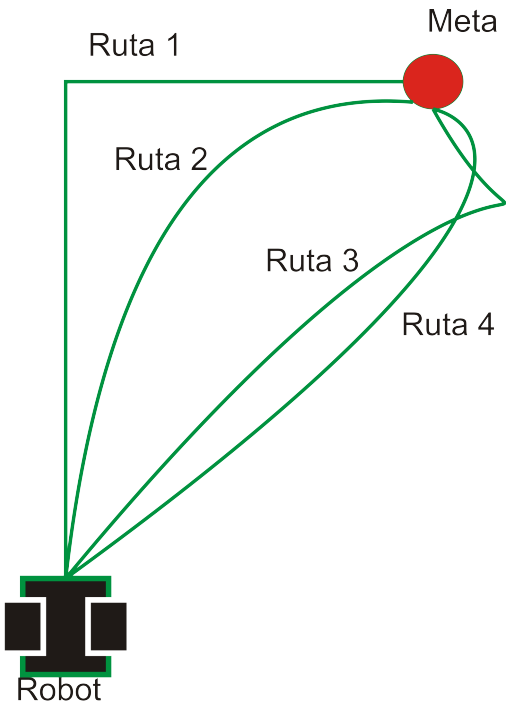


Figura 3.8: Rutas de navegación de un robot móvil

$$C_{(x,y)} = \text{mín}\,(ValorvecinoCPA - Distancia) \tag{3.9}$$

Con esto la ecuación del costo está más cercana a los parámetros de la realidad, ya que el robot puede navegar con una velocidad constante sobre algún tipo de terreno, ¿Pero cual sería una velocidad recomendad para que el robot navegue en terrenos irregulares? ¿Una velocidad alta, moderada, o baja? Lo más adecuado para el robot es navegar a una velocidad baja ya que disminuiría el problemas de derrape o volcaduras, aunque el tiempo de navegación del robot duraría más tiempo en hacer el recorrido al punto meta.

En este trabajo abordamos un control de velocidad con el propósito de elegir una velocidad adecuada para recorrer los terrenos de navegación del robot, teniendo la ecuación siguiente:

$$C_{(x,y)} = \text{mín}\,(ValorvecinoCPA - Distancias - Velocidad) \tag{3.10}$$

En la ecuación 3.10 se agrega la velocidad, esto para poder determinar si el siguiente movimiento del robot es el adecuado conforme a las características del terreno, añadiéndole un peso extra al costo de la celda. Obteniendo así, el mejor siguiente paso, aunque puede que no seas la ruta de menor distancia pero si el más rápido. Una vez que el robot empieza la navegación, este puede elegir entre varias rutas encontradas. El robot debe ser capaz de elegir la ruta más confiable para navegar. En la siguiente figura se muestran tres rutas que pueden ser confiables navegar (Ver Figura 3.9).

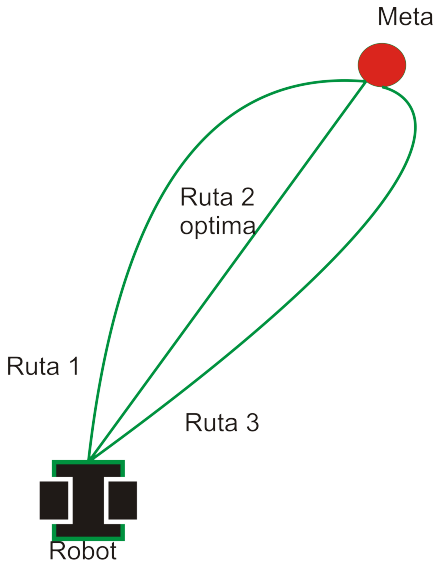


Figura 3.9: Rutas óptimas de navegación para un robot móvil

Con esto damos paso a nuestro siguiente problema el cual es la ruta de navegación que vamos a utilizar para llegar al punto meta. Además veremos como elegimos nuestra nueva ruta en caso de que un objeto obstaculice nuestra ruta de navegación

### 3.2.3. Algoritmos

La creación de las rutas de navegación puede ser construida de diferentes maneras. Una manera simple de generarla las rutas de navegación es con los campos potenciales artificiales [30] [23] [36] debido a que solamente se toma la información generada por el campo para elegir la siguiente posición de nuestra ruta. El punto ventajoso de utilizar campos potenciales es que la evasión del obstáculo está integrada al momento de realizar la ruta (Ver figura 3.10). El algoritmos 2 nos muestra cómo se realiza la construcción del mapa de navegación mediante la técnica de los campos potenciales artificiales.

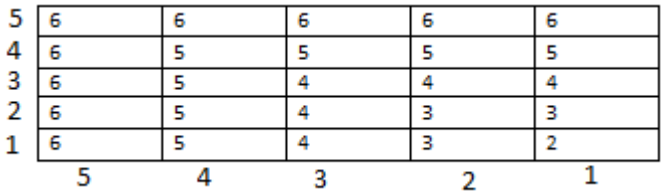


Figura 3.10: Mapa de navegación generado por campos potenciales artificiales.

La ruta de navegación obtenida para este ejemplo sería la ruta óptima representada por las diagonales, y sin ningún obstáculo en la ruta obtenida, ver Figura 3.11.

**Algorithm 2** Creación de ruta de navegación

**Entrada:** Posición destino $desx = x, desy = y$, Posición origen $orix = z, oriy = w, Minimo = N \cdot .M$

**Salida:** Ruta de navegación

$Ruta[x] = Mapa[desx][desy]$
**while** $Mapa[desx][desy] = Mapa[orix][oriy]$ **do**
  **for** $NumVecinos = 0; NumVecinos < 8; NumVecinos++$ **do**
    **if** $Vecinos < minimo$ **then**
      $desx = Vecindadx[i]$
      $desy = Vecindady[i]$
    **end if**
  **end for**
  $Ruta[x+1] = Mapa[desx][desy]$
**end while**
**return** *Ruta*.

Habiendo agregado el costo de celda y el costo total de toda la trayectoria, las coordenadas de toda la ruta son las siguientes,

$$\text{Ruta} = ((5,5),(4,4),(3,3),(2,2),(1,1)), \text{ con valores de celda} = (6,5,4,3,2)$$

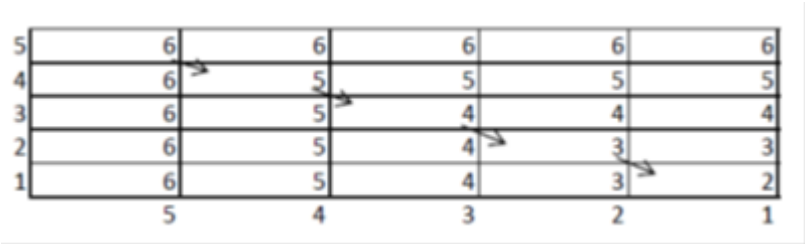


Figura 3.11: Mapa de navegación con ruta óptima.

Este ejemplo para la construcción de la ruta es muy sencillo al no encontrarse ningún obstáculo. Un ejemplo ahora con un obstáculo, mostrado en la Figura 3.12 por los valores 25, bloquea la ruta de navegación obtenida anteriormente, y se usa el algoritmo para buscar nuevas rutas y cumplir con el objetivo de llegar a la meta. Pero notemos lo que ocurre, a continuación:

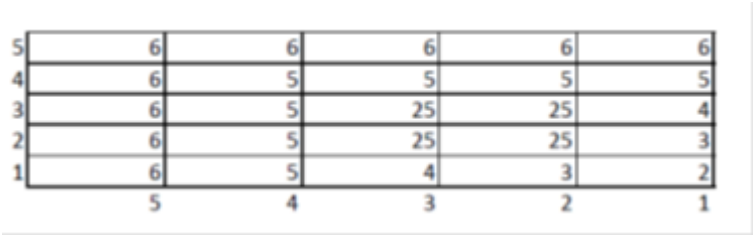


Figura 3.12: Mapa de navegación con un obstáculo presente.

Como observamos en la Figura 3.13, cuando se encuentra un obstáculo podemos observar que se crean dos rutas de navegación idénticas, con el mismo costo de navegación, pero con diferente ruta:

$$\text{Ruta1} = ((5,5),(4,4),(4,3),(4,2),(3,1),(2,1),(1,1)), \text{ con valor de celdas} = (6,5,5,5,4,3,2)$$
$$\text{Ruta2} = ((5,5),(4,4),(3,4),(2,4),(1,3),(1,2),(1,1)), \text{ con valor de celdas} = (6,5,5,5,4,3,2)$$

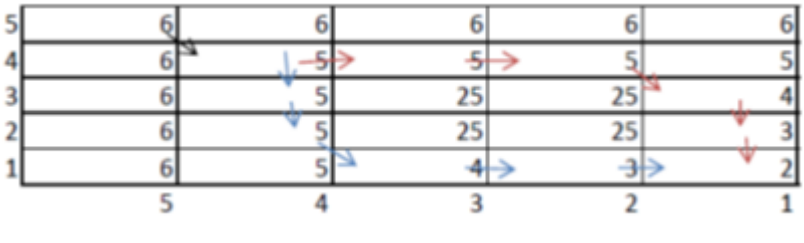


Figura 3.13: Mapa de navegación con un obstáculo y rutas de navegación.

La pregunta es ¿cuál de las dos rutas es mejor? Para un caso en interiores cualquiera de las dos rutas sería buena, ya que la velocidad sería constante, pero para

nuestro problema la información obtenida no sería suficiente para determinar la ruta óptima. Para la obtención de la ruta óptima, utilizamos un análisis de la rugosidad de los terrenos por los cuales transita el robot para conocer la velocidad adecuada para navegar dicho terreno, reducir el tiempo de navegación a través de la ruta y se evita derrapes que podríamos presentarse.

En la siguiente sección veremos cómo realizamos ese ajuste de velocidad para la navegación de los terrenos a una velocidad por medio de una red neuro-difusa.

## 3.3. Evasión de obstáculos

La propuesta planteada para evadir obstáculos es mediante métodos de campo potenciales artificiales que sirven no solamente sirve para construir el mapa de navegación por el cual el robot se desplaza; sino, también sirve para conocer la ubicación de los obstáculos que se encuentran en medio de la ruta a la meta(Ver Figura 3.14).

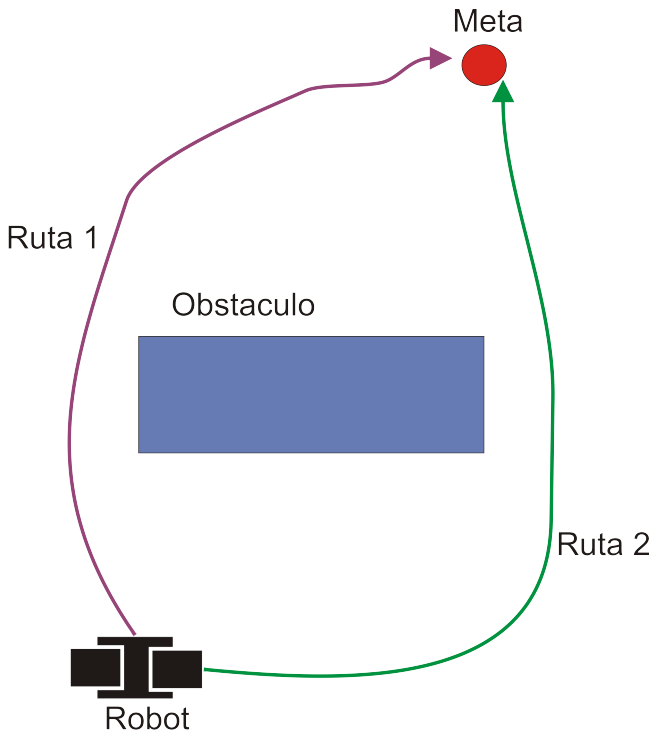


Figura 3.14: Navegación realizando la evasión de un obstáculo.

La propuesta es construir un mapa de navegación como lo planteamos anteriormente en este capítulo, representado por una matriz que contiene los datos obtenidos por el Algoritmo 1. El algoritmo de evasión de obstáculo se realiza mediante las mediciones obtenidas de los sensores ultrasónicos montados en el robot en movimiento.

Para la detección de un obstáculo la distancia propuesta para el sensor debe de marca una distancia mayo a:

$$45 > distancia > 30$$

La distancia para detectar a un obstáculo esta entre el rango mayor a treinta centímetros y menor a cuarenta y cinco centímetros. Esta distancia es considerada adecuada debido a que el robot necesita un margue de espacio entre el obstáculo y el, para poder girar y empezar a realizar la evasión de los obstáculos, para poder re-calcular la nueva ruta con el Algoritmo 2 propuesto en secciones pasadas.

La evasión de los obstáculos inmóviles, de velocidad y aceleración cero, como rocas, arbustos es simple. Los obstáculos móviles tienen velocidad y aceleración distinta de cero y pueden invadir la ruta de navegación. Además, los obstáculos en el mapa de navegación estarán representados con una forma cuadrada para facilitar el reconocimiento y elección de la ruta. Esta simplificación de la forma de obstáculo facilita el tratamiento de obstáculos con formas irregulares, y para los alcances de esta tesis resulta suficiente. En este trabajo se considera que los obstáculos móviles se mueven a velocidad constante. La detección de los obstáculos en movimientos se realiza de la misma forma como realizamos la detección de los obstáculos inmóviles solamente se agrega un retardo *delay* en la observación que hace el robot para conocer si el obstáculo se mueve o no.

Cuando el robot detecta un obstáculo lo agrega a su mapa de navegación y recalcula la ruta para navegar utilizando los tres algoritmos descritos en este Capítulo; el algoritmo 1 construye el mapa de navegación, posteriormente se agrega el obstáculo detectado con el algoritmo 3 y por último se recalcula la ruta con el algoritmo 2. Con esto ya tenemos nuestro nuevo mapa y ruta de navegación con un obstáculo (móvil o inmóvil) en el mapa de navegación. Si el obstáculo es móvil y, durante el retardo deja de estorbar la ruta de navegación del robot, este continua con su ruta de navegación anterior desechando la ruta recalculada.

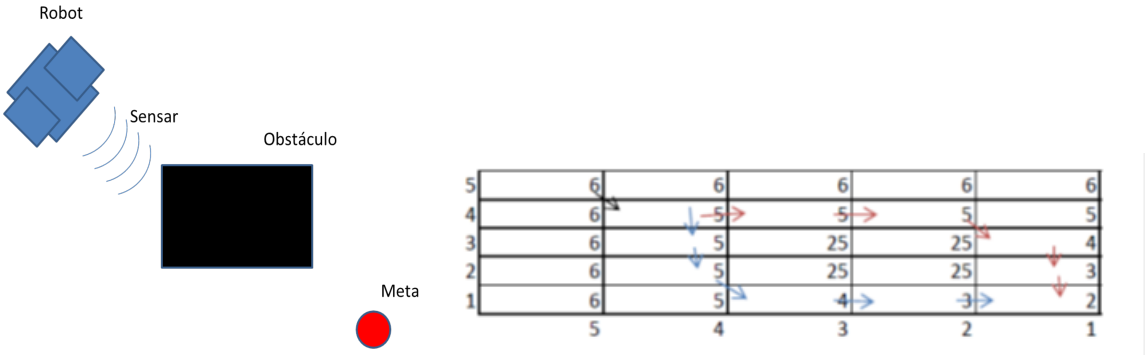


Figura 3.15: Censado del obstáculos y actualización de la ruta de navegación.

Como se muestra en la Figura 3.15 el robot obtiene la información de un obstáculo y este automáticamente se debe de actualizar en nuestro mapa de navegación y se actualiza la ruta de navegación óptima. Una vez realizado esto la ruta de navegación actualizada es enviada al robot para corregir y recorrer el camino hasta la meta como muestra el algoritmo 3 en la línea 7 y 12 La actualización del mapa se encarga de actualizar nuestro mapa de navegación y generar la nueva ruta de navegación.

**Algorithm 3** Sensado de obstáculo y actualización del mapa de navegación

**Entrada:** $SensorD = null, SensorI = null, P_actx, P_acty$
**Salida:** *Ruta*

```
    while 1 do
      SensorD = Adquisiciondedatodelsensorderechodelrobot
3:    SensorI = Adquisiciondedatodelsensorizuierdodelrobot
      if SensorD > 45andSensorD < 30 then
        Obsx = P_actx + SensorD
6:      Obsy = P_acty + SensorD
        ActualizaMapa
      end if
9:    if SensorI > 45andSensorI < 30 then
        Obsx = P_actx + SensorI
        Obsy = P_acty + SensorI
12:     ActualizaMapa
      end if
    end while
15: return Ruta
```

Aquí podemos como identificamos los obstáculos en el mapa de navegación. Primero obtenemos la información de lo que se encuentra frente del robot para conocer si existe un obstáculo en nuestra ruta en caso de que exista un obstáculo este tiene que ser registrado en nuestro mapa y actualizar el mapa con el obstáculo. El segundo

paso es generar nuestra nueva ruta de navegación, debido al obstáculo que agregado al mapa podría obstaculizar nuestra ruta de navegación o no.

En el siguiente capítulo describiremos las pruebas que se realizan a nivel simulación en la generación de los mapas de navegación y la elección de la ruta óptima con diferente número de obstáculos, mostrando los resultados obtenidos para este tipo de pruebas. En las pruebas físicas son más extensas debido a que se realizaran tres diferentes tipos de pruebas, de odometría, evasión de obstáculos y navegación con ajuste de velocidad. Para cada una de estas pruebas se han diseñado distintos escenarios desde los diferentes terrenos que navegara el robot y los distintos obstáculos que puede haber presente en su recorrido hasta la meta. El reporte de los resultados está conformado por un número significativo de pruebas representado en tablas en donde se muestra el error del robot al llegar a la meta.

## 3.4. Ajuste de velocidad

Para realizar una navegación de un robot con ruedas en un entorno exterior, es necesario que el robot sea capaz de reconocer las características del terreno por el cual navega, esto para ajusta su velocidad. En este trabajo nos basaremos en lo realizado en [2] [29].

Al imitar la percepción humana para el ajuste de la rugosidad de los terrenos y la estimación de la velocidad con el enfoque de lógica difusa, el robot será capaz de moverse a través de terrenos mediante la adaptación de su velocidad de acuerdo a las texturas suaves y las irregularidades que encuentre en su camino y decidir si es capaz de navegar el terreno.

El ajuste de velocidad busca imitar la percepción de los humanos para esto se emplean un enfoque neuro-difuso para la actualización de la velocidad. El método visión basada en apariencias (ABV) [31], tiene la eficacia requerida para el reconocimiento promedio de las texturas o aplicado al promedio de reconocimiento de las apariencias de los terrenos. ABV soporta eficientemente la actualización de la velocidad para la navegación de exteriores mostrado en esta sección.

Las redes neuronales artificiales (RN) son un sistema de conocimiento y automatización de procesos inspirados en sistema nervioso de los animales. Este sistema es conectado entre neuronas que trabajan en red, en nuestro propuesta utilizaremos las RN para el reconocimiento de las rugosidades. Mientras para la estimación de las rugosidades se utiliza lógica difusa. La lógica difusa es conocida por ser un método para tratar con conocimiento impreciso, utilizando reglas lingüísticas, los sistemas de lógica difusa simulan la toma de decisión humana para hacer frente a los conceptos expresados sin claridad, esto para hacer frente a los datos imprecisos o imperfectos

para la mejora de la representación del conocimiento y el razonamiento de la incertidumbre. La diferencia entre los métodos la lógica difusa y los métodos neuro-difusos, es que los métodos neuro-difusos necesitan de un menor número de reglas, mientras que la lógica difusa convención necesita de un gran número de reglas. Esto simplifica la estructura del modelo neuro-difuso, reduciendo el número de neuronas, capas ocultas y reduce el tiempo de cálculo

Por lo tanto, la metodología propuesta es la siguiente, un conjunto de texturas de tres terrenos representativos de la navegación serán modeladas con el método ABV, y una red neuronal entrenada con el conjunto de texturas representativa, son los componentes principales para la clasificación.

La red neuronal difusa (RNF) esta entrenada para determinar la velocidad adecuada con respecto a la clase de textura del terreno. El robot se mueve a la velocidad estimada y adquiere una nueva imagen del terreno; este ciclo se repite mientras el robot este en movimiento. (Ver Figura 3.16).

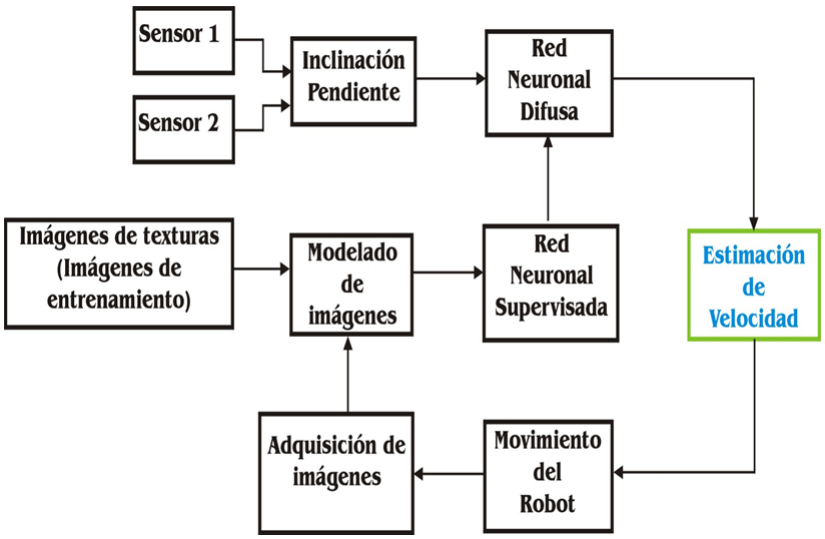


Figura 3.16: Bloque de diagrama del enfoque para la actualización de velocidad

El algoritmo está divido en dos etapas la primera etapa es la identificación de rugosidades cual será entrenada por medio de una red neuronal supervisada, mientras la segunda etapa la actualización de la velocidad y el robot en movimiento realizada por medio de la red neuronal difusa.

La descripción de la etapa uno es la siguiente:

- Seleccionar una imagen del terreno.
- Caracterizar la textura utilizando el método ABV.

- Entrenar la red neuronal supervisada con las clases de texturas establecida por un conductor humano experto.

- Entrenar la red neuronal difusa para determinar la velocidad obtenida de la clases de textura acordada por el conductor experto, construimos el conjunto difuso y la inferencia de las reglas del sistema IF-ELSE.

En la segunda etapa describiremos lo siguiente:

- Adquirir la imagen del terreno de la cámara del robot.

- La red neuronal supervisada clasifica el terreno.

- La red neuronal difusa tiene como entrada el terreno clasificado, de esta manera con la rugosidad de la textura, determina una velocidad para navegar. El robot ajusta mediante un sistema de control mecánico.

- Este ciclo se repite mientras el robot este en movimiento.

La red neurona propuesta contiene cinco capas. Dos capas de entrada $x$ (textura) e $y$ (pendiente), y una de salida $f$ (velocidad del robot). En la Figura 3.17 se ilustra la arquitectura de la red neuronal difusa. Las texturas son clasificadas por su rugosidad en el siguiente conjunto difuso: high (H), medium (M) y low (L), mostrado en la figura 3.18. La inclinación de la pendiente se divide en el siguiente conjunto difuso: plain (PI), slighly plain (SP), slighly sloped (SS), moderato sloped (MS), high slope (HS) y very high (VH), como se ilustra en la Figura 3.19. Por ultimo en la Figura 3.20 observamos los valores de salida high velocity (HV), moderate velocity (MV), low velocity (LV) y stop velocity (ST). Las reglas de inferencia de la lógica difusa están listadas en la Tabla 3.1. A continuación se explicara las capas de la red nuero-difusa.

**Capa 1**: la capa uno está compuesta por dos entras de las funciones de membresía donde todo nodo i en esta capa es un nodo adaptativo con otro nodo de la función:

$$\begin{array}{ccc} O_{1,1} = u_r(L) & O_{1,4} = u_r(PL) & O_{1,7} = u_r(MS) \\ O_{1,2} = u_r(M) & O_{1,5} = u_r(SP) & O_{1,8} = u_r(HS) \\ O_{1,3} = u_r(H) & O_{1,6} = u_r(SS) & O_{1,9} = u_r(VH) \end{array}$$

Donde $x(o, y)$ son los nodo de entrada con el conjunto de etiquetas lingüísticas [L, M, H] o [PL, SP, SS, MS, HS, VH]; $u_r$ y $u_s$ son respectivamente las funciones de membresía para la rugosidad y la pendiente.

**Capa 2**: Cada nodo de esta capa es una etiqueta de un nodo fijo ?, cuya salida es el producto de todas las señales de entradas, $O_{(2,i)} = \omega_i, i = 1, \ldots, 18$ , donde:

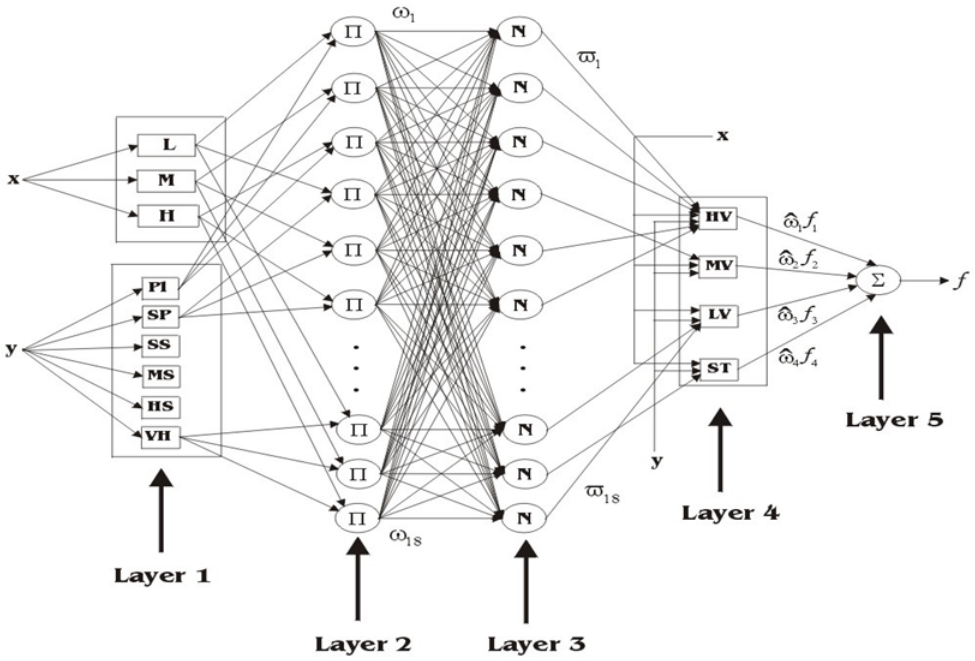


Figura 3.17: Arquitectura de la red neuronal difusa. Red neuronal de cinco capas, las dos primeras de entrada (textura y pendiente), la tercera capa el conjunto de las reglas base, la cuarta un conjunto de término de los valores de membresía de salida y la quinta capa la salida de la velocidad del robot para el terreno.

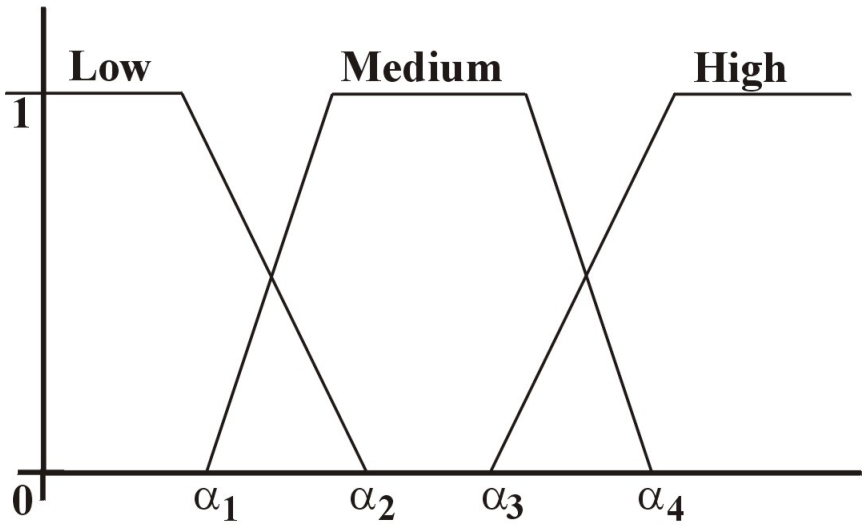


Figura 3.18: Función de membresía para la clasificación de las rugosidades del terreno.

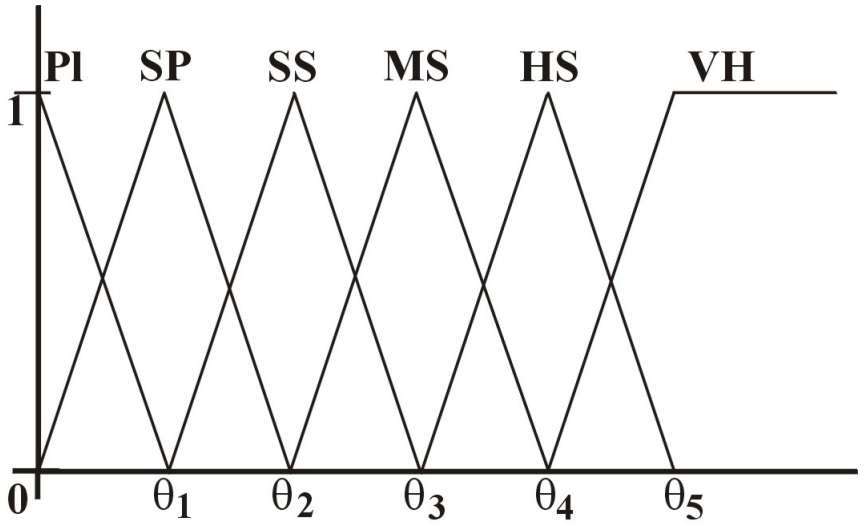


Figura 3.19: Función de membresía para determinar la inclinación de la pendiente.

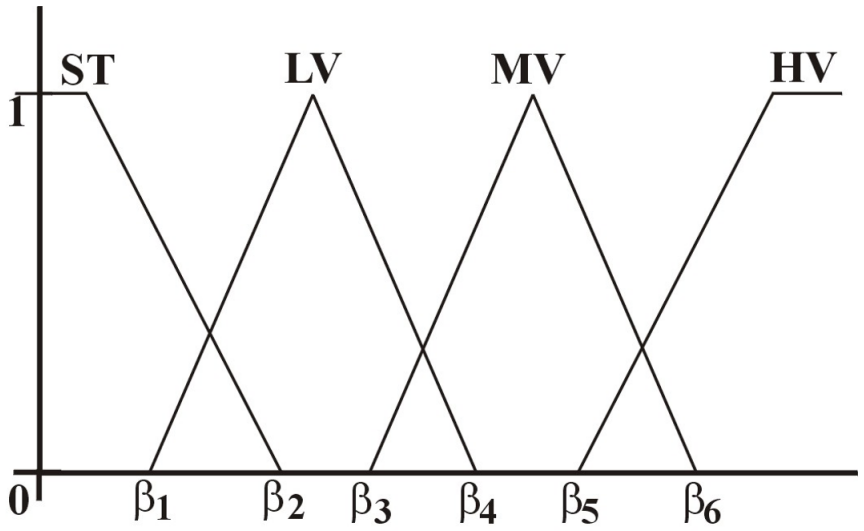


Figura 3.20: Función de membresía para determinar la velocidad del robot.

$$\begin{array}{lll}
\omega_1 = u_r(L)u_s(PL) & \omega_7 = u_r(L)u_s(SS) & \omega_{13} = u_r(L)u_s(HS) \\
\omega_2 = u_r(M)u_s(PL) & \omega_8 = u_r(M)u_s(SS) & \omega_{14} = u_r(M)u_s(HS) \\
\omega_3 = u_r(H)u_s(PL) & \omega_9 = u_r(H)u_s(SS) & \omega_{15} = u_r(H)u_s(HS) \\
\omega_4 = u_r(L)u_s(SP) & \omega_{10} = u_r(L)u_s(MS) & \omega_{16} = u_r(L)u_s(VH) \\
\omega_5 = u_r(M)u_s(SP) & \omega_{11} = u_r(M)u_s(MS) & \omega_{17} = u_r(M)u_s(VH) \\
\omega_6 = u_r(H)u_s(SP) & \omega_{12} = u_r(H)u_s(MS) & \omega_{18} = u_r(H)u_s(VH)
\end{array}$$

Cada nodo de salida representa la fuerza de las regalas. El operado T-norm es el operador empleado para el producto algebraico.

**Capa 3**: Cada nodo en esta capa es una etiqueta de un nodos fijo N. El ith nodo calcula el radio de la $i$th regla:

$$O_{3,1} = \varpi = \frac{\omega_i}{\sum_{j=1}^{18} \omega_j}, i = 1, \ldots, 18.$$

Los resultados de esta capa se llaman normalización de los puntos fuertes.

**Capa 4**: Cada nodo k en esta capa es un nodo de adaptación con una función:

$$O_{4,k} = \hat{\omega}_k(p_k x + q_k y + r_k), k = 1, \ldots, 4.$$

Donde $[p_k, q_k, r_k \ ; \ k = 1, \ldots, 4]$ es el conjunto de parámetros de este nodo.

**Capa 5**: La última capa esta capa nos regresar el valor de la velocidad que el robot debe de tener para navegar el terrenos con una menor probabilidad de que derrape. El único nodo de esta capa es uno nodo con etiqueta fija $\sum$, esta se encarga de computar los resultados globales de la suma de todas las señales de entrada (método de defusificación):

$$O =_{5,1}= \sum_{k=1}^{4} \hat{\omega}_k f_k$$

El procedimiento para desunificar asigna la salida difusa del mecanismo de inferencia a una señal nítida. Donde la fórmula para calcular la velocidad del robot es v= $V_max?V_FNN$ , donde $V_max$ es la velocidad máxima permitida del vehículo y $V_FNN$ es la salida de la red neuronal difusa, con un valor en los intervalos $[0, 1]$.

Con esto tenemos la explicación de la red neuronal difusa y como se realiza su entrenamiento y la obtención de la velocidad basada en la obtención de las rugosidades del terreno por el cual el robot se encuentra navegando.

En la siguiente sección se explicara la forma de evadir los obstáculos que encontremos en nuestras rutas de navegación para así poder generar nuevas rutas seguras

| Rule No. | Input | | Output |
|---|---|---|---|
| | Slope angle | Texture roughness | Velocity |
| 1 | PL | L | HV |
| 2 | PL | M | HV |
| 3 | PL | H | HV |
| 4 | SP | L | MV |
| 5 | SP | M | HV |
| 6 | SP | H | HV |
| 7 | SS | L | MV |
| 8 | SS | M | MV |
| 9 | SS | H | HV |
| 10 | MS | L | LV |
| 11 | MS | M | MV |
| 12 | MS | H | MV |
| 13 | HS | L | LV |
| 14 | HS | M | LV |
| 15 | HS | H | MV |
| 16 | VH | L | ST |
| 17 | VH | M | ST |
| 18 | VH | H | LV |

Tabla 3.1: Reglas IF-ELSE de inferencia del sistema neuronal difuso.

para navegar.

Para el entrenamiento fuera de línea de la red neuronal se utilizó la biblioteca de imágenes de terrenos exteriores creada por *the Columbia Utrecht Reflectance and Texture Database (CUReT)* y utilizada por Farid García en su tesis doctoral [29]. De esta biblioteca se tomaron 98 imágenes de diferente rugosidad de terrenos exteriores, algunas se muestran en la 3.21.

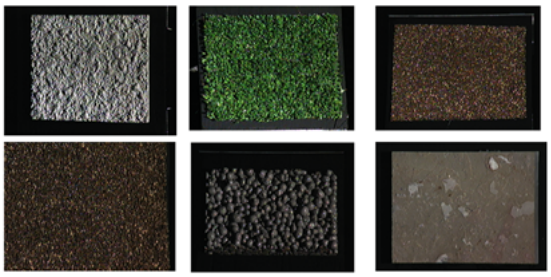

Figura 3.21: Imágenes de terrenos exteriores para entrenamiento para la red neuronal.

# Capítulo 4

# Pruebas y resultados

*A nivel simulación se muestra la creación de rutas del robot a partir de la información obtenida del mapa de navegación, y la evasión de obstáculos. Con el robot físico se prueban sus capacidades de navegación y evasión en terrenos exteriores, midiendo con odometría la precisión/error entre la posición final del robot y la meta, y el tiempo utilizado. Se prueba la evasión de obstáculos, móviles e inmóviles, que hace el robot sobre su ruta de navegación; también, algunas pruebas simples de evasión de obstáculos fijos haciendo el robot ajuste de velocidad.*

## 4.1. Plataforma Lego Mindstorm

Lego Mindstorm es un juego de robótica de la empresa lego, el cual contiene como elementos básicos de la teoría de robótica, como la unión de piezas y la programación de acciones, en forma interactiva, y tal que puede usarse para construir un robot industrial o un vehículo, cómo el que utilizamos en este tesis.

Características bloque NXT:

- Micro controlador ARM7 32-bits, a 48MHz de uso general, con 256Kb Flash, 64 Kb RAM.
- Micro controlador AVR 8-bit a 8MHz, control de los motores ( modulación PWM), 4kB Flash, 512 RAM.
- Bluetooth (SSP) + USB 2.0, 3 Salidas, 4 Entradas, LCD, 4 botones, Parlante, 6 pilas AA.

El nuevo sistema dispone de cuatro puertos con terminales de hilos cada uno. Los sensores pueden ser analógicos digitales y es importante destacar que se mantiene en retro compatibilidad con los sensores RCX. El robot móvil contara con dos ruedas fijas, además contara con dos sensores ultrasónicos, una cámara inalámbrica con una

resolución de 640*380 pixeles y la comunicación será realizada atreves de bluethooth, ver Figura 4.1.

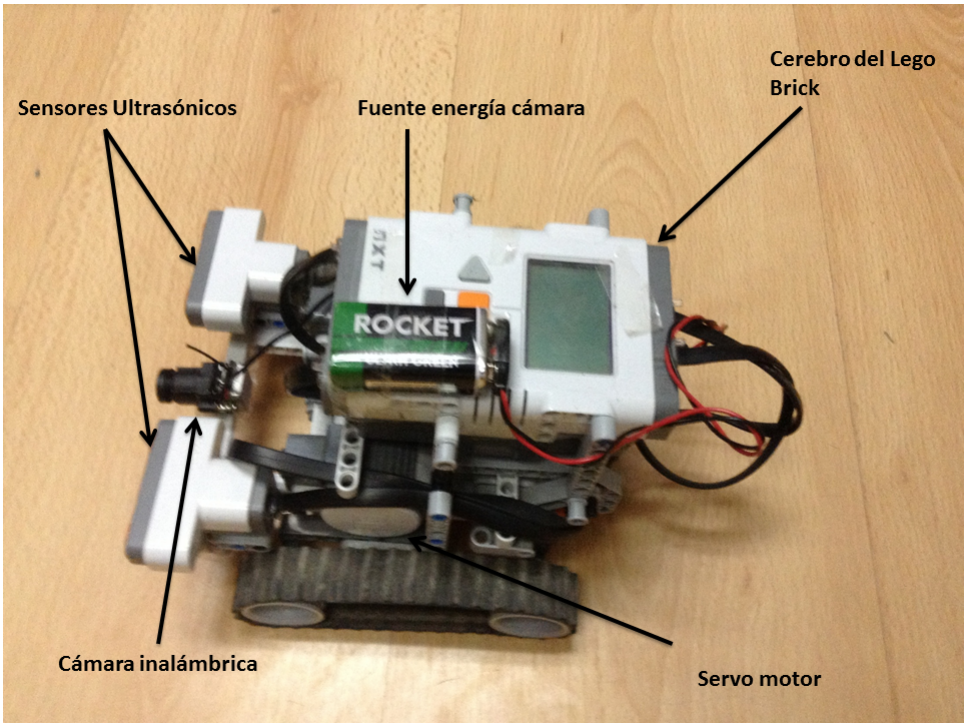


Figura 4.1: Vehículo de pruebas, Lego Mindstrom.

## 4.2. Simulación

El simulador es un software bajo el entorno JAVA, el cual proporciona la ruta optima que nuestro robot deberá de tomar para llegar a su punto meta. Está construida con los métodos mencionados en el capítulo anteriores para la obtención de valor así como para la evasión de los obstáculos, empezaremos por describir nuestro entorno mostrado en la Figura 4.2.

En la interfaz gráfica, en la parte central izquierda se encuentra el entorno de navegación; en esta área se agregan los obstáculos; en la parte central del lado derecho se muestran los datos del mapa arrojados por el campo potencial artificial, y en la parte inferior las posiciones origen y meta del robot. En la Figura 4.3, las coordenadas iniciales $(0, 0)$ y finales $(19, 19)$, los cuadros marcados con rojo es la ruta navegación del robot (el caso más sencillo sin obstáculos en la ruta). Los obstáculos son agregados marcados de color negro, como se muestra en la Figura 4.3, con el clic del ratón en cualquier recuadro dentro del área de navegación. Al agregar un obstáculo el área y la ruta se actualizarán generando otra ruta si es requerido. Se pueden agregar tantos obstáculos como el mapa de navegación lo permita.

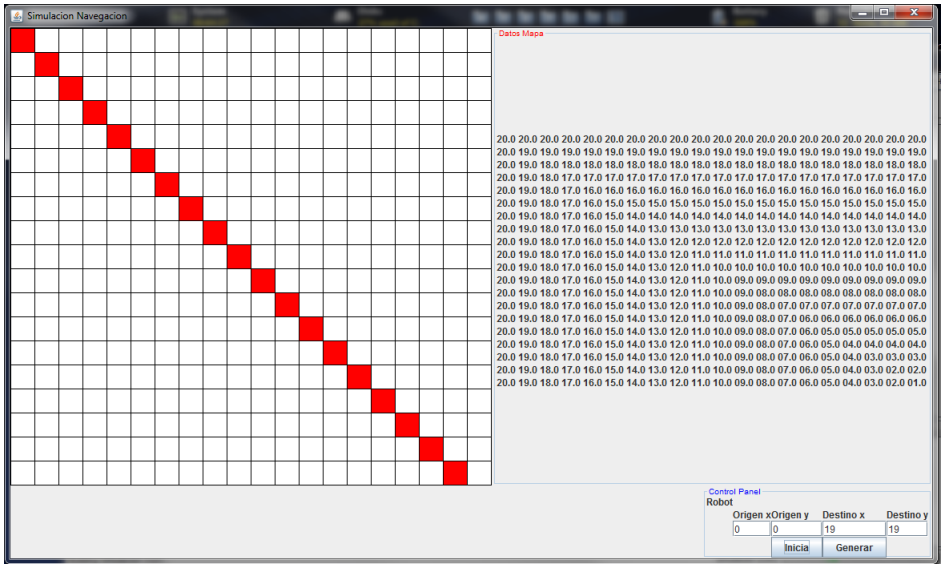


Figura 4.2: Interfaz gráfica para la simulación de ruta de navegación.

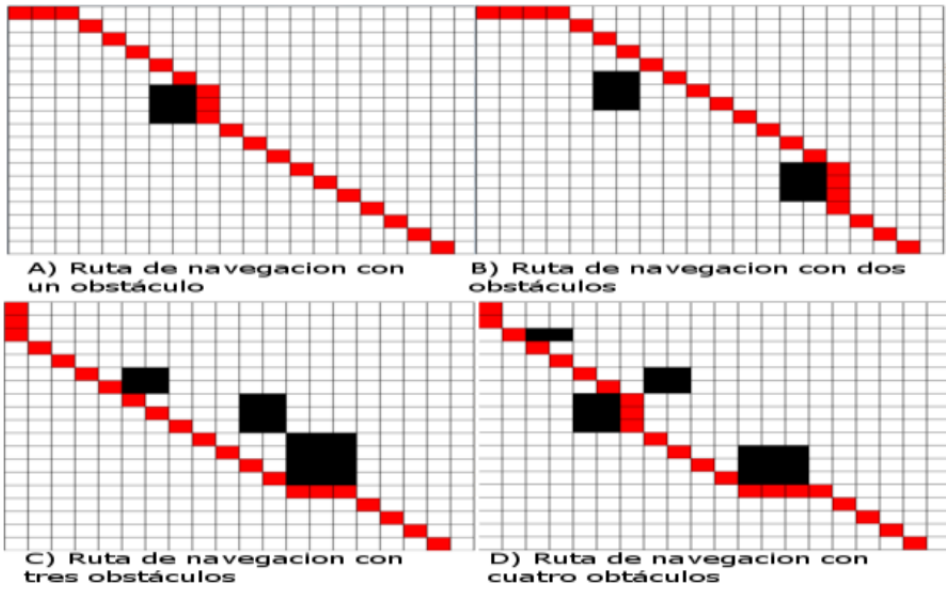


Figura 4.3: Simulación de generación de rutas de navegación con varios obstáculos.

Los resultados obtenidos con la simulación es la creación de una ruta navegable hasta para 6 obstáculos. Teniendo una implementación lo suficientemente solidad para crear rutas con un número considerable de obstáculos en nuestro entorno de navegación.

## 4.3. Pruebas físicas en trayectorias rectilíneas

La plataforma de trabajo física con el robot Lego Mindstorm es tal que la construcción está basada en un modelo diferencial teniendo dos ruedas fijas conectada a los motores, dos ruedas libres, y dos sensor ultrasonido para medir la distancias que existe entre él y un objeto. Las pruebas que se realizan a nivel físico son las siguientes:

- Pruebas de odometría: para obtener la medición de la distancia y obtener la posición por la cual el robot navega y registrar anomalías en la ruta.
- Pruebas de precisión: para conocer el error que tenemos para llegar al punto meta.
- Pruebas de evasión de obstáculos: el recorrido de la ruta con diferentes números de obstáculos y tipo de obstáculos.

En los diferentes terrenos las pruebas odometricas permiten medir la distancia recorrida por el robot, y de esa manera conocer la precisión y el error entre la posición final del robot y la meta indicada a lograr. Dependiendo del terreno, al navegar a una velocidad no adecuada le provocarían al robot derrapes o volcaduras, y por tanto mayor error respecto a la meta. Las pruebas a realizar son en terrenos de:

- Pasto o Césped.
- Tierra.
- Pavimento.
- Combinación de los terrenos.

Se eligieron estos tres terrenos debido a que son de los más común encontrar en terrenos exteriores, esta prueba se realizara con tres velocidad diferentes una baja, media y alta, además de una extra con un control de velocidad la cual se espera que obtenga un mejor resultado que las tres anteriores.

Los parámetros a considerar en las pruebas son:

1. Tiempo de recorrido.
2. Error con respecto a la meta.

A continuación, los primeros resultados obtenidos de la odometría [13] se reportan en las tablas 4.1, 4.2 y 4.3, cada una representando uno de los terrenos a navegar, a diferentes velocidades respectivamente. El error a la meta está representado en centímetros indicando la distancia entre la posición final del robot y la meta, navegando en línea recta a una distancias de 300 cm; la velocidad están descrita en centímetros sobre segundo (cm/s) siendo el avance del robot por cada segundo.

Figura 4.4: Terreno de navegación tipo pavimento.

Un terreno pavimentado, ver Figura 4.4, presenta pocas irregularidades y es navegable a velocidades media-altas con ruedas lisa u orugas, otorgando al robot una alta tracción al navegar. En el Tabla 4.1 se observan resultados con errores pequeños en todas las velocidades, dado que el pavimento es un terreno con pocas irregulares y ocurren pocos derrapes al robot durante la navegación. En este caso la navegación puede realizarse a cualquier velocidad.

Figura 4.5: Terreno de navegación tipo pasto.

Un terreno con pasto, ver Figura 4.5, presenta más dificultades al navegar debido a sus irregularidades y su propia textura. Las ruedas de tractor u orugas, otorgando

| Error a la meta (Centimetros) | | | | |
|---|---|---|---|---|
| Prueba / Velocidad | 18.82 cm/seg | **25.00 cm/seg** | 31.43 cm/seg | 37.5 cm/seg |
| 1 | 0 | **1** | 3 | 8 |
| 2 | 2 | **1** | 17 | 8 |
| 3 | 5 | **1** | 7 | 5 |
| 4 | 7 | **7** | 7 | 5 |
| 5 | 10 | **2** | 5 | 15 |
| 6 | 1 | **21** | 17 | 8 |
| 7 | 1 | **9** | 3 | 6 |
| 8 | 1 | **1** | 7 | 15 |
| 9 | 6 | **1** | 16 | 2 |
| 10 | 15 | **7** | 3 | 28 |
| **Promedio de Error** | 4.8 | **5.1** | 8.5 | 10 |

Tabla 4.1: Error en superficie de pavimento.

al robot una buena tracción. En la Tabla 4.2 los resultados de navegar en pasto muestran cómo se producen un mayor número de error, además de presentarse volcaduras durante la navegación (representadas por *). La velocidad más adecuada para navegar el terreno de paso en un línea recta es media-baja.

| Error a la meta (Centimetros) | | | | |
|---|---|---|---|---|
| Prueba / Velocidad | 18.82 cm/seg | **25.00 cm/seg** | 31.43 cm/seg | 37.5 cm/seg |
| 1 | 30 | **5** | 38* | 17 |
| 2 | 8 | **5** | 149* | 201* |
| 3 | 8 | **2** | 13 | 16 |
| 4 | 8 | **2** | 8 | 197* |
| 5 | 7 | **1** | 50* | 12 |
| 6 | 4 | **15** | 21 | 2* |
| 7 | 33 | **2** | 21 | 15 |
| 8 | 33 | **6** | 2 | 34 |
| 9 | 41 | **7** | 26 | 5 |
| 10 | 10 | **25** | 30* | 15 |
| **Promedio de Error** | 18.5 | **7** | 33.4* | 51.4* |

Tabla 4.2: Error en superficie de pasto.

Por último un terreno de tierra compacta y ciertas irregularidades, ver Figura 4.6, navegable a velocidades media-altas con ruedas de tractor u orugas, otorgando al robot una alta tracción al navegar. En la Tabla 4.3 se observa que, en este caso, la velocidad más baja no es la mejor dada la dificultad para girar las ruedas del robot que tenemos; es probable es que con ruedas de menor fricción sea más simple el giro

Figura 4.6: Terreno de navegación tipo tierra compacta.

y baste con poca potencia de motor para girar las ruedas. En contraste para navegar en línea recta en este terreno y con este robot, la velocidad media-alta es la más adecuada.

| Error a la meta (Centimetros) | | | | |
|---|---|---|---|---|
| Prueba / Velocidad | 18.82 cm/seg | 25.00 cm/seg | **31.43 cm/seg** | 37.5 cm/seg |
| 1 | 3 | 0 | **1** | 2 |
| 2 | 22 | 15 | **3** | 4 |
| 3 | 30 | 2 | **20** | 1 |
| 4 | 0 | 19 | **7** | 3 |
| 5 | 5 | 2 | **1** | 14 |
| 6 | 10 | 10 | **1** | 4 |
| 7 | 10 | 5 | **0** | 24 |
| 8 | 4 | 8 | **12** | 6 |
| 9 | 28 | 8 | **2** | 14 |
| 10 | 20 | 10 | **22** | 10 |
| **Promedio de Error** | 13.2 | 7.9 | **6.9** | 8.2 |

Tabla 4.3: Error en superficie tipo tierra

## 4.4. Evasión del obstáculo

Lo que se muestra con estas pruebas es la capacidad del robot de evadir los obstáculos fijos o móviles [13]. Las pruebas se diseñan en los siguientes escenarios:

Para esta prueba se diseñan diferentes escenario:

1. Un obstáculos sin movimiento.

2. Varios obstáculos sin movimiento.

3. Un obstáculos con movimiento sencillo.

### 4.4.1. Obstáculos Fijos

En las pruebas el robot navega a una velocidad media de 25 cm/seg y se encuentra a una distancias de 150 y 300 cm del punto meta. En las tablas siguientes se muestra los errores en precisión del robot con obstáculos en su ruta de navegación para los terrenos de pavimento, pasto y tierra compacta. Se muestra el error, entre la posición final del robot y la meta indicada, para cada una de un conjunto de pruebas y el promedio de error de todas ellas. El error varía dependiendo del terreno y el número de obstáculos que se evaden. En los resultados para este tipo de pruebas se pretende mostrar que el robot es capaz de resolver, satisfactoriamente, desde el caso más sencillo hasta uno de los más complejos. El error depende del número de obstáculos encontrados por el robot durante su navegación, a la distancia a la que se encuentran y el terreno por el cual navegó, como lo muestran los siguientes experimentos y resultados. En las Figuras 4.7 y 4.8 se muestra la trayectoria del robot en algunas de las pruebas realizadas para 1 y 2 obstáculos.

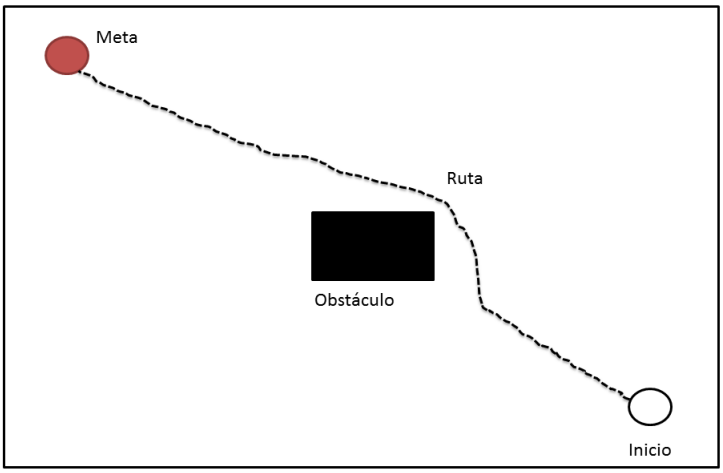


Figura 4.7: Ejemplo de ruta de navegación para la evasión de un obstáculo.

En el Tabla 4.4 podemos observar el error de llegar a la meta en un terreno pavimentado con obstáculos estáticos, sobre una ruta de 150 cm. La primera columna nos muestra el error que existe al evadir un obstáculo teniendo un error promedio del 4.92 %, mientras que la segunda columna nos muestra es para la navegación de dos obstáculos obteniendo un error de 8.84 %. El error en ambos de los escenarios de prueba, es muy bajo en porcentaje, aunque casi del doble si el robot evade dos obstáculos; este escalamiento de casi al doble, no siempre ocurre, como veremos en las siguientes pruebas.

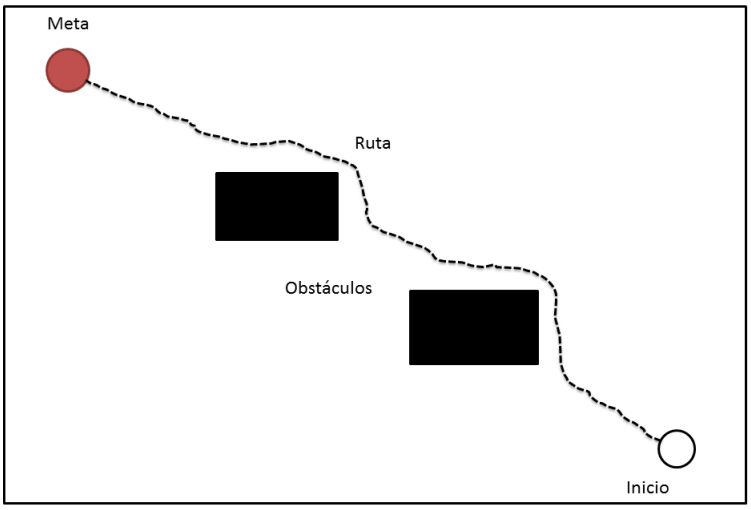


Figura 4.8: Ejemplo de ruta de navegación para la evasión de dos obstáculos.

| **Velocidad = 25 cm/seg** | **Error: distancia al objetivo /150 cm** | |
|---|---|---|
| Pavimento | 1 obstáculo | 2 obstáculos |
| 1 | 0 cm | 5 cm |
| 2 | 10.5 cm | 39 cm |
| 3 | 20 cm | 31 cm |
| 4 | 0 cm | 25.1 cm |
| 5 | 5 cm | 9 cm |
| 6 | 0 cm | 2 cm |
| 7 | 5 cm | 5 cm |
| 8 | 8 cm | 0 cm |
| 9 | 5 cm | 1 cm |
| 10 | 40.5 cm | 36 cm |
| 11 | 2 cm | 2 cm |
| 12 | 10 cm | 10 cm |
| 13 | 15 cm | 15 cm |
| 14 | 10.5 cm | 10.5 cm |
| **Promedio de error** | **7.39 cm (4.92 % error)** | **13.273 cm (8.84 % error)** |

Tabla 4.4: Errores durante la evasión en terreno pavimentado en 150 cm.

| **Velocidad = 25 cm/seg** | **Error: distancia al objetivo / 300 cm** | |
|---|---|---|
| Pavimento | 1 obstáculo | 2 obstáculos |
| 1 | 14.5 cm | 14 cm |
| 2 | 0 cm | 3 cm |
| 3 | 5 cm | 1.5 cm |
| 4 | 9.5 cm | 7.5 cm |
| 5 | 4.5 cm | 45 cm |
| 6 | 0 cm | 3 cm |
| 7 | 15.5 cm | 30 cm |
| 8 | 43.5 cm | 40 cm |
| 9 | 9.9 cm | 0 cm |
| 10 | 0 cm | 10 cm |
| 11 | 8.5 cm | 0 cm |
| 12 | 19 cm | 20 cm |
| 13 | 43.5 cm | 10 cm |
| 14 | 14.5 cm | 10 cm |
| **Promedio de error** | **14.42 cm (4.80 % error)** | **13.1 cm (4.36 % error)** |

Tabla 4.5: Errores durante la evasión en terreno pavimentado en 300 cm.

En el Tabla 4.5 podemos observar el error de llegar a la meta en un terreno pavimentado con obstáculos estáticos, en una ruta de 300 cm. La primera columna con la evasión de un obstáculo nos muestra un error del 4.80 %, mientras que el error que existe al evadir dos obstáculos es del 4.36 %. Es notable la disminución del error en una ruta del doble de longitud.

En el Tabla 4.6 podemos observar el error de llegar a la meta en un terreno de tierra compacta con obstáculos estáticos. La primera columna nos muestra un error al evadir un obstáculo del 9.52 %, mientras que para dos obstáculos tenemos un error del 8.84 %. Nótese que también dependiendo del terreno de navegación el error disminuye.

En el Tabla 4.7 podemos observar el error de llegar a la meta en un terreno de tierra compacta con obstáculos estáticos, en una ruta de 300 cm. La primera columna nos muestra un error al evadir un obstáculo del 4.31 %, mientras que para dos obstáculos tenemos un error del 4.91 %. En ese caso hay un incremento, no significativo, del porcentaje de error promedio en una ruta de doble longitud.

Como se muestra en las tablas entre mayor se la distancia de navegación del robot al punto meta, el error se disminuye: si los obstáculos tiene mayor distancia entre ellos es menos significativo el impacto al realizar la evasión, tanto en el tiempo total de recorrido como en el error entre la posición final del robot y la meta.

| **Velocidad = 25 cm/seg** | **Error: distancia al objetivo / 150 cm** | |
|---|---|---|
| Tierra compacta | 1 obstáculo | 2 obstáculos |
| 1 | 0 cm | 0 cm |
| 2 | 20 cm | 36 cm |
| 3 | 25 cm | 7 cm |
| 4 | 11 cm | 29 cm |
| 5 | 0 cm | 35 cm |
| 6 | 5 cm | 18 cm |
| 7 | 15 cm | 15 cm |
| 8 | 40 cm | 10 cm |
| 9 | 10 cm | 25 cm |
| 10 | 0 cm | 10 cm |
| 11 | 20 cm | 25 cm |
| 12 | 10 cm | 3 cm |
| 13 | 27 cm | 13 cm |
| 14 | 17 cm | 33 cm |
| **Promedio de error** | **14.28 cm (9.52 % error)** | **13.273 cm (8.84 % error)** |

Tabla 4.6: Errores durante la evasión en tierra compacta en 150 cm.

| **Velocidad = 25 cm/seg** | **Error: distancia al objetivo / 300 cm** | |
|---|---|---|
| Tierra compacta | 1 obstáculo | 2 obstáculos |
| 1 | 2 cm | 5 cm |
| 2 | 20 cm | 23 cm |
| 3 | 9 cm | 10 cm |
| 4 | 17 cm | 19 cm |
| 5 | 15 cm | 17 cm |
| 6 | 17 cm | 20 cm |
| 7 | 17 cm | 5 cm |
| 8 | 35 cm | 12 cm |
| 9 | 15 cm | 5 cm |
| 10 | 0 cm | 5 cm |
| 11 | 0 cm | 35 cm |
| 12 | 7 cm | 15 cm |
| 13 | 0 cm | 20 cm |
| 14 | 23 cm | 25 cm |
| **Promedio de error** | **12.93 cm (4.31 % error)** | **14.73 cm (4.91 % error)** |

Tabla 4.7: Errores para la evasión en tierra compacta en 300 cm.

### 4.4.2. Obstáculos en movimiento

En las pruebas de evasión en movimiento el robot navega a una velocidad media de 25 cm/seg y se encuentra a una distancia de 300 cm del punto meta. En las Tablas 4.8 y 4.9 se muestra los errores en precisión del robot con un obstáculo en movimiento (solo de avance) en su ruta de navegación para los terrenos de pavimento, pasto y tierra compacta. Se muestra el error, entre la posición final del robot y la meta indicada, para cada una de un conjunto de pruebas y el promedio de error de todas ellas. Hay que mencionar que el obstáculo se puede detener en cualquier momento.

En el Tabla 4.8 podemos observar el error de llegar a la meta en un terreno pavimentado con un obstáculo con movimiento sencillo, en una ruta de 300 cm. La primera columna nos muestra un error al evadir un obstáculo del 6.76 %.

| **Velocidad = 25 cm/seg** | **Error: distancia al objetivo / 300 cm** |
|---|---|
| **Pavimento** | 1 **obstáculo móvil** |
| 1 | 25 cm |
| 2 | 5 cm |
| 3 | 12 cm |
| 4 | 40 cm |
| 5 | 45 cm |
| 6 | 20 cm |
| 7 | 37 cm |
| 8 | 9 cm |
| 9 | 39 cm |
| 10 | 19 cm |
| 11 | 10 cm |
| 12 | 1 cm |
| 13 | 12 cm |
| 14 | 10 cm |
| **Promedio de error** | **20.28 cm (6.76 % error)** |

Tabla 4.8: Errores para la evasión de un obstáculo móvil en pavimento en 300 cm.

En el Tabla 4.9 podemos observar el error de llegar a la meta en un terreno de tierra compacta con un obstáculo en movimiento. La primera columna nos muestra un error al evadir un obstáculo del 7.59 %.

Nótemos que el error de llegar a la meta aumenta en comparación con la evasión de los obstáculos inmóviles. Esto es debido a la dificultad que presenta evadir obstáculos en movimiento por su comportamiento que puede ser de continuar su avance o detenerse bloqueando la ruta de navegación obligando a re-calcula una nueva ruta de navegación del robot.

El decidir cuál va ser la nueva ruta, presenta un reto para el robot ya que se de-

| **Velocidad = 25 cm/seg** | **Error: distancia al objetivo / 300 cm** |
|---|---|
| **Tierra Compacta** | 1 **obstáculo móvil** |
| 1 | 19 cm |
| 2 | 11 cm |
| 3 | 7 cm |
| 4 | 60 cm |
| 5 | 15 cm |
| 6 | 17 cm |
| 7 | 20 cm |
| 8 | 38 cm |
| 9 | 19 cm |
| 10 | 48 cm |
| 11 | 20 cm |
| 12 | 21 cm |
| 13 | 17 cm |
| 14 | 7 cm |
| **Promedio de error** | **22.78 cm (7.59 % error)** |

Tabla 4.9: Errores para la evasión de un obstáculo móvil en tierra compacta en 300 cm.

be de considerar un tiempo razonable para analizar el obstáculo y evadirlo: el robot debe determinar en ese pequeño lapso de tiempo si esperar a que el obstáculo deje de bloquear la ruta para continuar su camino o bien evadirlo por enfrente o por detrás.

Terrenos de pavimento o tierra compacta, por su poca rugosidad, es adecuado navegarlos a velocidad media-alta, facilitando el movimiento del robot y la evasión de los obstáculos. Mientras que en terrenos de pastos el robot tiene mayor complejidad en su navegación dadas las irregularidades, hojas de pasto y varas, entre otras, lo cual incremente la dificultad al realizar los giros para evadir obstáculos; todo esto provoca que el robot pudiera perderse, colisionar, y/o no llegar a la meta.

## 4.5. Ajuste de velocidad

Durante la navegación en exteriores las irregularidades del terreno puede causar que nuestro robot se desvié de su ruta optima u ocurra alguna volcadura; buscamos reducir que ocurran estos problemas mediante la realización, por parte del robot, de un ajuste de velocidad dependiendo de las características del terreno de navegación, de las variaciones del terreno: a distintas características distintas velocidades. La velocidad debe ser la máxima posible dadas las características específicas del terreno en que el robot avanza a la par de garantizar una navegación segura, sin derrapes ni colisiones.

Ajuste de velocidad de robots con ruedas sin obstáculos en terrenos de pavimento, pasto, tierra compacta y tierra con piedras sueltas, se ha hecho en [2] [29]. En esta tesis se logra hacerlo sobre terrenos de pasto, tierra compacta y pavimento, pero agregando obstáculos fijos a la ruta de navegación del robot. La navegación bajo estas condiciones, sobre superficies irregulares, conlleva mayores dificultades para el ajuste de velocidad de un robot con ruedas. Las pruebas a realizar son las siguientes:

1. Navegación en pavimento.
2. Navegación en pasto.
3. Navegación en tierra compacta.

En estas pruebas el robot navega ajustando su velocidad de navegación dependiendo de las características del terreno; se encuentra a una distancia 300 cm del punto meta. En las tablas siguientes se muestra los errores en precisión del robot con un obstáculo inmóvil en su ruta de navegación para los terrenos de pavimento, pasto y tierra compacta, combinados. Se muestra el error, entre la posición final del robot y la meta indicada, para cada uno de los conjuntos de pruebas y el promedio de error de todas ellas. El error varía dependiendo del terreno y el número de obstáculos que se evaden. En los resultados para este tipo de pruebas se pretende mostrar que el robot es capaz de resolver, satisfactoriamente, la evasión del obstáculo realizando un ajuste de velocidad dependiendo de las características del terreno de navegación. El error depende de obstáculo encontrado por el robot durante su navegación y el terreno por el cual navegó, como lo muestran los siguientes experimentos y resultados.

En el Tabla 4.10 podemos observar el error de llegar a la meta en un terreno pavimentado con un obstáculo estático, sobre una ruta de 300 cm, ajustando la velocidad del robot. La primera columna nos muestra el error que existe al evadir un obstáculo con un error promedio del 3.53 %. Comparando el resultado obtenido en esta prueba con la Tabla 4.5 que muestra un error de 4.80 %, logrando una disminución del error al momento de llegar a la meta.

| | Error: distancia al objetivo / 300 cm |
|---|---|
| **Pavimento** | 1 **obstáculo inmóvil** |
| 1 | 12 cm |
| 2 | 10 cm |
| 3 | 5 cm |
| 4 | 9 cm |
| 5 | 4 cm |
| 6 | 8 cm |
| 7 | 17.5 cm |
| 8 | 5.5 cm |
| 9 | 8 cm |
| 10 | 8.5 cm |
| 11 | 19 cm |
| 12 | 25 cm |
| 13 | 10 cm |
| 14 | 7 cm |
| **Promedio de error** | **10.60 cm (3.53 % error)** |

Tabla 4.10: Errores para la evasión en pavimento en 300 cm y ajustando la velocidad.

En el Tabla 4.11 podemos observar el error de llegar a la meta en un terreno pavimentado con tierra compacta en la mitad del camino y con un obstáculo estático, sobre una ruta de 300 cm, ajustando la velocidad del robot. La primera columna nos muestra el error que existe al evadir un obstáculo con un error promedio del 3.7 %. Comparando el resultado obtenido en esta prueba con la Tabla 4.5 y 4.7 nos muestra un error de 4.80 % en un terreno pavimentado y 4.31 % en un terreno de tierra compacta; con esto se logra una disminución al momento de llegar a la meta existiendo un cambio de terreno.

Por ultimo en el Tabla 4.12 podemos observar el error de llegar a la meta en un terreno mayormente de tierra compacta con pequeñas áreas de pasto y con un obstáculo estático, sobre una ruta de 300 cm, ajustando la velocidad del robot. La columna nos muestra el error que existe al evadir un obstáculo con un error promedio del 4.11 %. Comparando el resultado obtenido en esta prueba con la Tabla 4.7 nos muestra un error de 4.31 % en un terreno de tierra compacta; con esto se logra una disminución al momento de llegar a la meta existiendo un cambio de terrenos y además realizando la navegación en pequeñas áreas de pasto.

Como observamos, con este conjunto de pruebas de navegación tal que el robot realiza ajuste de velocidad dependiendo de las características del terreno, se minimiza el error de llegada a la meta, aproximadamente del 1 %, en consideración con los errores obtenido a velocidad constante y con un obstáculo fijo. Esto es debido a que el realizar un ajuste de velocidad permite al robot a minimizar los derrapes ocurrido en la ruta de navegación por los diferentes terrenos, mientras que mantener una ve-

| | Error: distancia al objetivo / 300 cm |
|---|---|
| **Pavimento + Tierra Compacta** | 1 **obstáculo inmóvil** |
| 1 | 20 cm |
| 2 | 10 cm |
| 3 | 10 cm |
| 4 | 9 cm |
| 5 | 5 cm |
| 6 | 8 cm |
| 7 | 25.5 cm |
| 8 | 10.5 cm |
| 9 | 0 cm |
| 10 | 8.5 cm |
| 11 | 15 cm |
| 12 | 8 cm |
| 13 | 19 cm |
| 14 | 7 cm |
| **Promedio de error** | **11.10 cm (3.7 % error)** |

Tabla 4.11: Errores para la evasión en Pavimento y tierra Compacta en 300 cm y ajustando la velocidad.

locidad constante el robot puede tener derrapes desviándolo de su ruta o volcaduras que podría dañar al robot.

A partir de los resultados obtenidos, se muestra que para realizar una navegación y una evasión apropiada, el robot debe considerar los obstáculos y el terreno por el cual navega, y para minimizar los derrapes es muy conveniente actualizar la velocidad dependiendo de las características de los terrenos. Todo lo anterior se sustenta en un método eficiente para construir y actualizar rutas en presencia de obstáculos e irregularidades evitando colisionar o derrapar. Específicamente, el ajuste de velocidad conlleva un menor margen de error con respecto a mantener una velocidad constante de navegación.

| | Error: distancia al objetivo / 300 cm |
|---|---|
| **Tierra Compacta + Pasto** | 1 **obstáculo inmóvil** |
| 1 | 17 cm |
| 2 | 15 cm |
| 3 | 19 cm |
| 4 | 9 cm |
| 5 | 17 cm |
| 6 | 10 cm |
| 7 | 9 cm |
| 8 | 10 cm |
| 9 | 15 cm |
| 10 | 10 cm |
| 11 | 10 cm |
| 12 | 9 cm |
| 13 | 10 cm |
| 14 | 13 cm |
| **Promedio de error** | **12.35 cm (4.11 % error)** |

Tabla 4.12: Errores para la evasión en tierra compacta y pasto en 300 cm y ajustando la velocidad.

# Capítulo 5

# Discusión y conclusiones

Actualmente se ha desarrollado mucho trabajo como podemos observar en la ROBOCUP: robots con ruedas con gran movilidad, rapidez en desplazamiento, toma de decisiones, cooperación entre robot, evasión de obstáculos [38] [19]. Sin embargo, limitando a solo terrenos de navegación de superficies lisas, sin irregularidades [21].

En este trabajo de tesis se abordó la navegación de robot móvil en terrenos exteriores y la evasión de obstáculos lográndose tener los resultados descritos, en particular se ha minimizando el error entre la posición final del robot y la meta. Los resultados del Capítulo 4 muestran a nivel físico y de simulación, la generación de rutas de navegación de un robot con ruedas sobre terrenos de tierra, pasto y pasto con tierra, y en presencia de obstáculos. En las pruebas físicas de navegación en exteriores se calcula el error con respecto a la meta sobre diversos terrenos: dependiendo de las características de la ruta, la velocidad elegida por el robot se minimiza el riesgo de derrapes o volcaduras, y en consecuencia, el error de distancia entre la posición final del robot y la meta, disminuye asimismo. Además, el ajuste de velocidad del robot en función del terreno, garantiza una navegación tal que se evitan derrapes que provoquen la pérdida de dirección o volcaduras. En [24] se restringe la navegación en terrenos que se consideran de derrape para el robot. En [35] el robot navega sobre terrenos exteriores y no considerar minimizar los derrapes. Por otro lado en [45] y [6], la navegación es sobre un terreno liso, y el robot navega sin la dificultad de las irregularidades en terrenos exteriores.

En un robot el mecanismo de evasión de los obstáculos es indispensable para adaptarse a los cambios durante la navegación, y ser capaz de re-planear su ruta y continuar con su navegación. El ajuste de velocidad es otro herramienta indispensable para mantener al robot libre de derrapes en terrenos exteriores, y mantener al robot fuera de terrenos que podría ponerlo en riesgo. Para hacer más robustos los resultados obtenidos, es necesario un conjunto de pruebas más amplio para validar que se mejora la precisión respecto a la meta mediante el ajuste de velocidad, sobre distintos tipos de terrenos y con obstáculos móviles en las rutas de navegación del robot.

## 5.1. Aportaciones

Las aportaciones realizadas en este trabajo fueron:

- Se construyeron algoritmos para la construcción de los mapas y las rutas de navegación.
- Se construyó un algoritmo para realizar la evasión de los obstáculos inmóviles y móviles.
- Se optimizo el programa de ajuste de velocidad reduciendo de 20 a 5 minutos los tiempos de cargado de los archivos de información, (3/4 del tiempo menos).
- Se incorporó construcción de rutas de navegación junto con el ajuste de velocidad realizado por Dr. Faríd García Lamont [29].

## 5.2. Conclusiones

La construcción de rutas de navegación en terrenos exteriores tales que se consideren las características del terreno y la evasión de obstáculos fijos o con movimiento, permiten hacer los ajustes de velocidad adecuados y evitar derrapes o colisiones, haciendo segura y eficiente la navegación del robot en tales terrenos. Los resultados de los experimentos muestran que los porcentajes de error dependen de la cantidad de obstáculos a evadir y de la dificultad del terreno de desplazamiento, siendo menos relevante la longitud de la ruta de navegación del robot. Asimismo, considerando las características del terreno para decidir la velocidad de navegación del robot, disminuye el margen de error a la meta, medido con odometría.

## 5.3. Publicación de la tesis

A. González-Sarabia and M. Alvarado, "Navegación de un Robot con Ruedas en Exteriores Evadiendo Obstáculos", A publicarse en memorias de XV Congreso Latinoamericano de Control Automático 2012 (CLCA), Lima, Peru, 23 - 26 de octubre de 2012.

## 5.4. Tema de investigación abiertos

El ajuste de velocidad en terrenos exteriores deberá dar respuesta a varios retos en la navegación de vehículos de exploración, asistencia a discapacitados y prevención medio ambiental, entre otros. El trabajo desarrollado en esta tesis sirve de base para dar mejores soluciones a diversos retos en la navegación en exteriores, como los siguientes:

- Integración más robusta del módulo de ajuste de velocidad con el modulo que realiza la navegación del robot minimizando los errores durante la ejecución del programa.
- Mejorar el módulo de evasión de obstáculos en movimiento: pasar de evasión de obstáculos en movimiento simples a otros de mayor complejidad
- Comparar con otros técnicas, como la de odometría visual [34] o GPS [36].
- Eliminar los sensores ultrasónicos y realizar la detección de los obstáculos basado en visión por computadora.

# Bibliografía


[1] E. Abbott and D. Powell. Land-vehicle navigation using gps. *Proceeding of the IEEE*, (1):145–162, 1999.

[2] Matías Alvarado and Farid García. Wheeled vehicles' velocity updating by navigating on outdoor terrains. *Neural Computing and Applications*, 20(7):1097–1109, 2011.

[3] Max Bajracharya, Mark W. Maimone, and Daniel M. Helmick. Autonomy for mars rovers: Past, present, and future. *IEEE Computer*, 41(12):44–50, 2008.

[4] Christopher A. Brooks and Karl Iagnemma. Visual detection of novel terrain via two-class classification. In *In Proceedings of the 2009 ACM Symposium on Applied Computing (SAC)*, Honolulu, Hawaii, USA, 2009.

[5] Rodney A. Brooks. *A robust layered control system for a mobile robot*. 1985.

[6] Widodo Budiharto, Achmad Jazidie, and Djoko Purwanto. Indoor navigation using adaptive neuro fuzzy controller for servant robot. *In Proceedings of the 2010 Second International Conference on Computer Engineering and Applications*, 1(14-15), july 2010.

[7] L. Ciobanu and N. Thirer. Modeling vechicles and mobile robots. In *In IEEE 25th Convention of Electrical and Electronics Engineers in Israe*, Honolulu, Hawaii, USA, 2008.

[8] Koditschek D. Exact robot navigation by means of potential functions: Some topological considerations. In *In Robotics and Automation, Proceedings of the 1987 IEEE International Conference on Robotics and Autonomation*, Raleigh, North Caroline, USA, 1987.

[9] D. de Falco, G. Di Massa, and S. Pagano. On the castor dynamic behavior. *Journal of the Franklin Institute*, 347:116–129, 2010.

[10] Universidad de Sevilla Departamento de Ingenieria de Sistemas y Automática. http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf.

[11] M. Deng, A. Inoue, K. Sekiguchi, and L. Jiang. Two-wheeled mobile robot motion control in dynamic environments. *Robotics and Computer-Integrated Manufacturing*, (3):268–272, 2010.

[12] Guilherme N. DeSouza and Avinash C. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 24(2):237–267, 2002.

[13] A. González-Sarabia and M. Alvarado. Navegación de un robot con ruedas en exteriores evadiendo obstÃ¡culos. In *A Publicarse en memorias de XV Congreso Latinoamericano de Control Automático 2012 (CLCA)*, Lima, Peru, 2012.

[14] Luis Gracia and Josep Tornero. Kinematic modeling and singularity of wheeled mobile robots. *Advanced Robotics*, 21(7):793–816, 2007.

[15] Luis Gracia and Josep Tornero. A new geometric approach to characterize the singularity of wheeled mobile robots. *Robotica*, 25(5):627–638, 2007.

[16] Luis Gracia and Josep Tornero. Kinematic control of wheeled mobile robots. *Latin American Applie Research*, 38(1):7–16, 2008.

[17] Luis Gracia and Josep Tornero. Optimal trajectory planning for wheeled mobile robots based on kinematics singularity. *Journal of Intelligent and Robotic Systems*, 53(2):145–168, 2008.

[18] Wang Hai-hua. Optimal obstacle avoidance path planning in robot soccer. In *2010 International Conference on Environmental Science and Information Application Technology (ESIAT)*, Jiaozuo Univ., Jiaozuo, China, 2010.

[19] Woong-Gie Han, Sung Kyun Kwan, Seung-Min Baek, and Tae-Yong Kuc . Ga based online path planning of mobile robots playing soccer games. In *In Proceedings of the 40th Midwest Symposium onCircuits and Systems, 1997.*, Sacramento, CA, USA, 1997.

[20] Ayanna M. Howard, Homayoun Seraji, and Barry Brian Werger. Global and regional path planners for integrated planning and navigation. *Journal of Robotic System*, 22(12):767–778, 2005.

[21] J. Hrabec and B. Honzik. Mobile robots playing soccer. In *In 7th International Workshop on Advanced Motion Control, 2002.*, Maribor, Slovenia, 2002.

[22] Yuan-Pao Hsu, Ching-Chih Tsai, Zeng-Chung Wang, Yi-Jiang Feng, and Hung-Hsing Lin. Hybrid navigation of a four-wheeled tour-guide robot. In *In Proceedings of the 2009 ACM Symposium on Applied Computing (SAC)*, Fukuoka, Japan, 2009.

[23] L. Huang. Velocity planning for a mobile robot to track a moving target - a potential field approach. *Robotics and Autonomous Systems*, 57(1):55–63, 2009.

[24] Karl Iagnemma and Chris C. Ward. Classification-based wheel slip detection and detector fusion for mobile robots on outdoor terrain. *Autonomous Robots*, 26(1):33–46, 2009.

[25] J.L. Jones, B.A. Seiger, and A.M. Flynn. *Mobile Robots: Inspiration to Implementation*. Ak Peters Series. A.K. Peters, 1999.

[26] Rahul Kala, Anupam Shukla, and Ritu Tiwari. Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness. *Neurocomputing*, 74:2314–2335, 2011.

[27] Daniel E. Koditschek. The robotics review 1. chapter Robot planning and control via potential functions, pages 349–367. MIT Press, Cambridge, MA, USA, 1989.

[28] Kurt Konolige. A gradient method of realtime robot control. In *In Proceding of the 2000 IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, Takamatsu, Japan, 2000.

[29] F. G. Lamont. *Wheeled-Robot's velocity updating and odometrybased localization by navigating on outdoor terrains*. PhD thesis, Phd, Computation, Centro de Investigación y de Estudios Avanzados del Instituto Politecnico Nacional, Distrito Federal, México, 2010.

[30] Jean-Claude Latombe. *Robot Motion Planning:*. Kluwer international series in engineering and computer science: Robotics. Kluwer Academic Publishers, 1990.

[31] Ales Leonardis and Horst Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99–118, 2000.

[32] Vladimir J. Lumelsky and Alexander A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *in Autonomous robot vehicles: Springer-Verlag New York, Inc.*, pages 363–390, 1990.

[33] D. Nair and J.K. Aggarwal. Moving obstacle detection from a navigating robot. *IEEE Transactions on Robotics and Automation*, (3):404–416, 1998.

[34] David Nistér, Oleg Naroditsky, and James R. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[35] T. Ohki, K. Nagatani, and K. Yoshida. Safety path planning for mobile robot on rough terrain considering instability of attitude maneuver. In *In 2010 IEEE/SICE International Symposium on System Integration (SII)*, December 21-22 2010.

[36] Kazunori Ohno, Takashi Tsubouchi, Bunji Shigematsu, and Shin'ichi Yuta. Differential gps and odometry-based outdoor navigation of a mobile robot. *Advanced Robotics*, 18(6):611–635, 2004.

[37] Steven Ratering and Maria L. Gini. Robot navigation in a known environment with unknown moving obstacles. *Autonomous Robots*, 1(2):149–165, 1995.

[38] Peter Robmeyer, Birgit Koch, and Dietmar P. F. Möller. Robocup 2004. chapter Mobile autonomous robots play soccer; an intercultural comparison of different approaches due to different prerequisites, pages 661–668. Springer-Verlag, Berlin, Heidelberg, 2005.

[39] K. Sekiguchi, Mingcong Deng, and A. Inoue. Obstacle avoidance and two wheeled mobile robot control using potential function. In *In IEEE International Conference onIndustrial Technology, 2006. (ICIT)*, Mumbai, India, 2006.

[40] S. Sheel and G. Verma. Velocity form pid controller for control and modeling of nonlinear processes using nn based approach. In *In 2011 Annuial IEEE India Conference (INDICON)*, Hyderebad, India, December 16-18 2011.

[41] R. Siegwart and I.R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Intelligent Robotics and Autonomous Agents. Mit Press, 2004.

[42] Martin Udengaard and Karl Iagnemma. Analysis, design, and control of an omnidirectional mobile robot in rough terrain. *Journal of Mechanical Design*, (12), 2009.

[43] Weidong Wang, Lei Zhou, Zhijiang Du, and Lining Sun. Track-terrain interaction analysis for tracked mobile robot. In *In Proceedings of the AIM 2008. IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Xian,Chinese, July 2-5 2008.

[44] C.W Warren. Global path planning using artificial potential fields. In *In Robotics and Automation, Proceedings of the 1989 IEEE Internacional Conference on Robotics and Automation*, Scottsdale, Arizona, USA, 1989.

[45] Su Weijun, Meng Rui, and Yu Chongchong. A study on soccer robot path planning with fuzzy artificial potential field. In *Proceedings of the 2010 International Conference on Computing, Control and Industrial Engineering (CCIE)*, Washington, DC, USA, 2010.

# Apéndice A

# Programas principales del robot Lego

```
1  //Giro  PID
2  #define K1 20//2.5//0.5555555      //3    10       6
3  #define K2 7//0.5//0.0333333      //1    2        2
4  #define K3 2//0
5  //Avanze PID
6  #define K4 20      //3    10       6
7  #define K5 7      //1    2        2
8  #define K6 2
9  #define DT 10
10 #define CMXG 0.0453419226957383548     //0.03222222
11 #define D    13.5
12
13 //Buzones
14 #define INBOXX 0 //buzon coordenada x
15 #define INBOXY 1 //buzon coordenada y
16 #define INBOXC 5 //buzon de confirmacion iniciada
17
18 #define OUTBOXDi 3  //buzon regreso sensor izquierdo
19 #define OUTBOXDd  4 //buzon regreso sensor derecho
20 #define OUTBOXc   2 //buzon regreso confirmacion para
21 //cambiar coordenadas
22 #define OUTBOXruta 6//buzon para actualizacion de ruta
23
24
25 //variables de mensajes
26 string respuestax;
27 string conf="z";
28 string Y1,X1;
```

```
29 string conf1;
30 string ruta="0";
31 string velocidad;
32 //Variblas de posicion
33 //float x1=0;
34 //float y1=0;
35 float bandera = 0;
36 float bandera1 =0;
37 float xa=0,ya=0;
38 float sensord=255,sensori=255; //sensores del robot
39 int vel;
40 int xx,yy;
41
42 //variables odometria
43 int speed=25;
44 byte handle;
45 int   bandesc=1;
46 float theta=0,
47       thetag=0,
48       c=0,
49       grad=0,
50       x1=-1,
51       y1=-1,
52       dr=0;
53
54
55 //funcion de odometria
56 task odometria()
57 {
58  int tikda=0,
59      tikia=0,
60      tikdp=0,
61      tikip=0,
62      deltagd=0,
63      deltagi=0;
64
65 float   deltad=0,
66         deltai=0,
67         px=0,
68         py=0,
69         deltas=0,
70         deltat=0,
71         cuad=0;
```

```
72
73 int       x=0,
74          y=0;
75
76  string s,sx,sy,sd;
77  int bytesWritten;
78
79  DeleteFile("Odo.txt");
80  CreateFile("Odo.txt",50000,handle);
81
82       while(1){
83
84            tikda=MotorRotationCount(OUT_C);
85            tikia=MotorRotationCount(OUT_B);
86            deltagd=tikda-tikdp;
87            deltagi=tikia-tikip;
88            deltad=deltagd*CMXG;
89            deltai=deltagi*CMXG;
90            deltas=(deltai+deltad)/2;
91            deltat=(deltad-deltai)/D;
92
93            x=x+deltas*cos(theta+(deltat/2));
94            y=y+deltas*sin(theta+(deltat/2));
95
96            xx=x;
97            yy=y;
98
99            cuad=(x*x)+(y*y);
100           dr=sqrt(cuad);
101           theta=theta+deltat;
102           thetag=(theta*360)/(2*PI);
103           tikdp=tikda;
104           tikip=tikia;
105
106
107           TextOut(0,LCD_LINE1, "distactual:");
108           NumOut(65,LCD_LINE1,dr);;
109
110           NumOut(0,LCD_LINE8,x);
111           NumOut(25,LCD_LINE8,y);
112
113           sx = NumToStr(x);
114           sy = NumToStr(y);
```

```
115             sd = NumToStr(theta);
116             s= StrCat(sx,"      ",sy);
117             WriteLnString(handle, s,  bytesWritten);
118             Wait(150);
119
120       }
121      /* Off(OUT_BC);  */
122       CloseFile(handle);
123 }
124
125 //funcion de avance del robot
126 void avancepid()
127 {
128 int i,
129     pwr,
130     e1=0,
131     e2=0,
132     itg,
133     der,
134     band=1;
135
136    while(band == 1)
137    {
138        i=dr;
139        e1=c-i;
140        der=e1-e2;
141        itg=e2+e1;
142        pwr=K4*e1+K5*der+K6*itg;
143        if (pwr>30)
144            OnFwdSync(OUT_BC, vel,0); //cambio de la velocidad del robot
145        else
146            OnFwdSync(OUT_BC, vel,0); //cambio de la velocidad del robot
147        e2=e1;
148        Wait(DT);
149        if (pwr < 1)
150            band=0;
151    }
152    bandesc=0;
153    Off(OUT_BC);
154 }
155
156 //funcion del giro del robot
157 void giropid()
```

```
158 {
159     int  i=0,
160          pwr=5,
161          e1=0,
162          e2=0,
163          itg ,
164          der ,
165          pb,
166          pc ,
167          band=1;
168     ClearScreen ();
169     while(band == 1)
170     {
171          i=thetag;
172         e1=(i-grad);
173         der=e1-e2;
174         itg=itg+e1;
175         pwr=K1*e1+K2*der+K3*itg;
176         TextOut(0,LCD_LINE3, "PWR:");
177         NumOut(50,LCD_LINE3, pwr);
178         if(pwr>60 | pwr<40)
179             pwr=35;
180         if(grad>=0)
181         {
182           if(thetag>grad){
183                pb=pwr;
184                OnFwd(OUT_B, pb);
185                pc=-pwr;
186                OnFwd(OUT_C, pc);
187           }
188           else{
189                pb=-pwr;
190                OnFwd(OUT_B, pb);
191                pc=pwr;
192                OnFwd(OUT_C, pc);
193           }
194         }
195         if(grad<0)
196         {
197           if(thetag<grad){
198                pb=pwr;
199                OnFwd(OUT_B, pb);
200                pc=-pwr;
```

```
201                 OnFwd(OUT_C, pc);
202             }
203             else{
204                 pb=-pwr;
205                 OnFwd(OUT_B, pb);
206                 pc=pwr;
207                 OnFwd(OUT_C, pc);
208             }
209         }
210
211         e2=e1;
212         Wait(DT);
213         TextOut(0,LCD_LINE4, "Theta:");
214         NumOut(50,LCD_LINE4, thetag);
215         TextOut(0,LCD_LINE6, "Theta:");
216         NumOut(50,LCD_LINE6, grad);
217         NumOut(0,LCD_LINE7,x1);
218         NumOut(15,LCD_LINE7,y1);
219
220         if((thetag>(grad-1)) & (thetag<(grad+1))){
221             band=0;
222             Off(OUT_BC);
223           }
224     }
225     //TextOut(0,LCD_LINE8,"Correcto Giro");
226       Off(OUT_BC);
227 }
228
229
230 //funcion de direccionamiento
231 //mediante los grados
232 void gradg()
233 {
234   float cuad=0,
235         //c=0,
236         //grad=0,
237         stet=0,
238         thet=0;
239
240   int ante=0;
241   string sx;
242
243   if (x1>=0)
```

```
244         if (y1>=0)
245             ante=1;
246         else
247             ante=4;
248     else
249         if (y1>=0)
250             ante=2;
251         else
252             ante=3;
253
254
255     cuad=(x1*x1)+(y1*y1);
256     c=sqrt(cuad);
257     stet=y1/c;
258     thet=asin(stet);
259     grad=(thet*360)/(2*PI);
260
261     switch (ante)
262     {
263       case 2 :
264                 grad=180-grad;
265                 break;
266       case 3 :
267                 grad=grad-90;
268                 break;
269                 default :
270                 break;
271     }
272      sx = NumToStr(grad);
273      TextOut(0,LCD_LINE6,"Grad:");
274      NumOut(25,LCD_LINE6,grad);
275      TextOut(0,LCD_LINE1, "Distar:");
276      NumOut(25,LCD_LINE1,c);
277 }
278
279 task distancia(){//Sensa la distancia que existen entre el robot y un obj
280     SetSensorLowspeed(IN_1);
281     SetSensorLowspeed(IN_4);
282     int d,i;
283     string disd,disi;
284     while(1){
285          i=SensorUS(IN_1);
286          d=SensorUS(IN_4);
```

```
287         sensord= d;
288         sensori= i;
289         disi=NumToStr(i);
290         disd=NumToStr(d);
291 //se recive la velocidad
292         NumOut(50,LCD_LINE2, vel);
293         NumOut(0,LCD_LINE2,velocidad);
294     }
295 }
296
297 //funcion recive las coordenadas a donde
298 //dirigir el robot
299 void buzon(){
300  int xante=-1;
301  int yante=-1;
302           ReceiveRemoteString(INBOXX,true,X1);
303           ReceiveRemoteString(INBOXY,true,Y1);
304           ReceiveRemoteString(INBOXC,true,velocidad);
305           vel = atoi(velocidad);
306           //vel = vel*5;
307           x1 = atof(X1);
308           y1 = atof(Y1);
309           if(x1!= xante && y1 != yante){
310             bandera = 1;
311             xante=x1;
312             yante=y1;
313           }
314 }
315
316
317 //funcion pricipal del robot
318 task main(){
319   int delay=0;
320   conf1 = NumToStr(1);
321   StartTask(distancia);
322   StartTask(odometria);
323
324    while(1){
325      buzon();
326      //banderas de confirmacion con la PC
327      SendResponseString(OUTBOXc,conf1);
328      SendResponseString(OUTBOXruta,ruta);
329      //deteccion de obstaculos
```

```
330         if((sensord > 25 || sensori > 25 || delay >1000) && bandera==1){
331             gradg();
332             giropid();
333             avancepid();
334             bandera=0;
335             conf1=NumToStr(2);
336             ruta=NumToStr(0);
337             delay=0;
338         }//fin if sensores
339         if(sensord < 25 || sensori<25 ){
340             delay=delay+1;
341         }
342         ruta=NumToStr(1);
343         conf1=NumToStr(1);
344         Off(OUT_BC);
345         Wait(250);
346     }
347     StopAllTasks();
348 }
```

# Apéndice B

# Programa navegación y creación de rutas

```
1  #include <cstdlib>
2  #include <iostream>
3  #include <string>
4  #include <conio.h>
5  #include <windows.h>
6  #include <sstream>
7
8  //libreria NXT C++
9  #include "Librerias Lego\nxt.h"
10
11 //librerias openCv
12 #include <cv.h>
13 #include <cxcore.h>
14 #include <highgui.h>
15
16 //Librerias ajuste de velocidad
17 #include "velocity.h"
18
19 using namespace std;
20 #define btcomm 3
21
22 //Definicio de variables de mapa
23 #define wm 10
24 #define hm 10
25
26 //Definicion de variales de la imagen
27 #define N 480
28 #define M 640
```

```
29
30
31 //llamada a funciones del mapa de navegacion
32 void creaMapa(void);
33 void imprimir(void);
34 void expaOnda(void);
35 void navega(void);
36 void obstaculo(int,int);
37 void coorenadas(void);
38 void SensarDis(void);
39
40 //funciones ajuste de velocidad
41 void RGBtoGray(double* pGray,uchar* pRGB,int width,int height);
42 void empilar(double* out,double* inp);
43 double norma(double* v);
44 void Leido();
45 void matlab_init();
46 int compute_velocity(double* data_out,double* pp,double* sl);
47 void matlab_terminate();
48
49 //variables globales del mapa de navegacion
50 int mapa[wm][hm];
51 int rutax[13]={1,1,2,2,2,2,3,4,5,6,7};
52 int rutay[13]={1,1,2,3,4,5,6,7,7,7,7};
53 int mini[wm*2];
54 int obstaculos = 900;
55 int multiplo = 10;
56
57
58 //Variables Globales ajuste de velocidad
59 //Crea una matriz en momoria donde se guardan los vectores propios
60 double VV [98][N*M];
61 //Crea un vector en memoria donde se guarda el vector promedio
62 double C[N*M];
63 //variables de dimenciones del frame
64 int height,width;
65 //representación de la velocidad estimada
66 double* v=new double[1];
67 double* sl=new double[1];
68 double velocidad;
69
70
71 //variables globales robot -> Pc
```

```
72 string x;
73 string y;
74  string vel;
75 int temp =0;
76 int control=0;
77
78 //Variables de cambio de funcion
79 int derecho,izquierdo;
80 int cambio;
81 int fin=1;
82 int actualizaruta;
83
84 //variables de proceso de navegacion
85 int bandera=1;
86 int b2 = 1;
87
88
89
90 /***********************************
91 El programa debe de estar en el NXT
92 para poder iniciarse de manera remota
93 ***********************************/
94
95 //set up the NXT
96 Connection *connection = new Bluetooth();
97 Brick *nxt = new Brick(connection);
98
99 int main()
100 {
101   // Nombre del programa a ejecutar en el NXT
102   string program_name = "buzon3.rxe";
103   int delay=0;
104
105   //iniciamos Vision y ajuste de velocidad
106   *v=0.5;
107    *sl=5;
108    uchar* data; //Informacion del frame capturado
109    // Frame de la camara RGB
110    IplImage* frame = cvCreateImage(cvSize(M,N),IPL_DEPTH_8U,1);
111    CvCapture* capture = cvCaptureFromCAM(1);
112    if(!capture) {
113      fprintf( stderr, "ERROR: capture is NULL \n" );
114      getchar();
```

```
115       return -1;
116     }
117     printf("Leyendo Vectores\n");
118     Leido();
119     //cvNamedWindow("RGB");
120     //iniciamos Matlab
121     matlab_init();
122    //Enlace con el robot NXT
123    try{
124       cout << "Tratando de enlazar al NXT" << endl;
125       //Conexión por bluetooth a través del puerto btcomm
126       connection->connect(3);
127       cout << "Connectado" << endl;
128       cout << "Comenzar programa: " << program_name << endl;
129       //Inicia de manera remota el programa program_name en el NXT
130       nxt->start_program(program_name,true);
131
132     }
133
134     catch (Nxt_exception& e){
135        //some error occurred - print it out
136        cout << e.what() << endl;
137        cout << "error code: " << e.error_code() << endl;
138        cout << "error type: " << e.error_type() << endl;
139        cout << e.who() << endl;
140        connection->disconnect();
141     }
142
143
144
145         creaMapa();
146     //obstaculo(3,3);
147         expaOnda();
148         navega();
149         //Sleep(8000);
150         coorenadas();
151         temp++;
152         Sleep(1000);
153         cout<<"terminos las funciones"<<endl;
154
155         while(temp != 13){
156
157                         //obtienes el frame de la camara
```

```
158                     frame = cvQueryFrame(capture);
159                     data = (uchar *)frame->imageData;
160
161                     //escala de Grises
162                     double* Gray= new double[N*M];
163                     RGBtoGray(Gray,data,N,M);
164
165                     //Endereza la imagen y la empila
166                     double* I=new double[N*M];
167                     empilar(I,Gray);
168                     delete [] Gray;
169
170                     //Normalización de la imagen
171                     double norm=norma(I);
172                     for(int j=0;j<(N*M);j++){
173                             I[j]=I[j]/norm;
174                     }
175
176                     //Resta entre la imagen y el vector promedio
177                     for(int j=0;j<(N*M);j++){
178                             I[j]=I[j]-C[j];
179                     }
180
181                     //Multiplicación de la imagen
182                     //con los vectores propios
183                     double* p = new double[98];
184                     for(int i=0;i<98;i++){
185                             p[i]=0;
186                             for(int j=0;j<(N*M);j++){
187                                     p[i]+=(VV[i][j]*I[j]);
188                             }
189                     }
190
191                     delete[]I;
192                     compute_velocity(v,p,sl);
193                     delete[] p;
194
195                     if(*v>1) *v=1;
196                     if(*v<=0) *v=0.01;
197
198                     velocidad = *v*13.88;
199                 cout<<velocidad<<endl;
200                     velocidad = rand();
```

```
201                         if (velocidad < 0.5 )
202                                 velocidad =7;
203                          if (velocidad >= 0.5)
204                                 velocidad =9;
205                             Sleep(1000);
206                             coorenadas();
207                             temp++;
208                             cout<<"termine"<<endl;
209                             delay++;
210                             Sleep(500);
211         }
212
213   //termina matlab
214   matlab_terminate();
215   //libera captura
216   cvReleaseCapture(&capture);
217   cout << "Deteniendo programa en el NXT" << endl;
218   //Cierra la comunicacion por bluetooth con el NXT
219   connection->disconnect();
220   //Finaliza el programa en el NXT
221   nxt->stop_programs(true);
222   return 0;
223 }
224
225 //Envio de las coordenadas al robot
226 //se hace la tranforamacion de un int a un string
227 void coorenadas(void){
228         stringstream s,s1,velo;
229         //envia velocidad
230         velo <<velocidad*8;
231         vel = velo.str();
232         cout<<"Valor:"<<vel<<endl;
233         nxt->write_msg(vel,5,false);
234         //envia coordenada x
235     s << rutax[temp] * multiplo;
236     x = s.str();
237     cout<<"Valor:"<<x<<endl;
238     nxt->write_msg(x,0,false);
239     //envia coordenada y
240     s1<<rutay[temp] * multiplo;
241     y=s1.str();
242     cout<<"Valor: "<< y<<endl;
243     nxt->write_msg(y,1,false);
```

```
244
245     fin = mini[temp];
246
247 }
248
249 //Expacion de la onda para la creacion de nuestro mapa de navegacion
250 void expaOnda(){
251         int posx = wm-1;//eje y
252         int posy = hm-1;//eje x
253         int wave = 0;
254         int w;
255         //Expacion de la onda
256         mapa[posx][posy]=wave;
257         while(posy!=0){
258                 // si existe un obstaculo
259                 if(mapa[posx][posy] != obstaculos){
260                         wave = mapa[posx][posy];
261                 }
262                 else{
263                         if(mapa[posx][posy-1]!=0 &&
264                         mapa[posx][posy-1]!= obstaculos){
265                                 wave = mapa[posx][posy-1];
266                         }
267                         else if(mapa[posx-1][posy]!=0 &&
268                          mapa[posx-1][posy]!= obstaculos){
269                                 wave = mapa[posx-1][posy];
270                         }
271                 }
272         //expacion del mapa
273                 if(mapa[posx-1][posy-1]==0 &&
274                 mapa[posx-1][posy-1]!=hm+wm*2){
275                         mapa[posx-1][posy-1]=wave+multiplo;
276                 }
277                 if(mapa[posx-1][posy]==0 &&
278                  mapa[posx-1][posy]!=hm+wm*2){
279                         mapa[posx-1][posy]=wave+multiplo;
280                 }
281                 if(mapa[posx][posy-1]==0 &&
282                 mapa[posx][posy-1]!=hm+wm*2){
283                         mapa[posx][posy-1]=wave+multiplo;
284                 }
285                 posx--;
286                 if(posx==0){
```

```
287                             posx=wm-1;
288                             posy--;
289                     }
290             }
291 }
292
293 //inicializacion del mapa y ruta de navegacion
294 void creaMapa(){
295             int i,j;
296
297             for(i=0;i<wm-1;i++){
298                     for(j=0;j<hm-1;j++){
299                             mapa[i][j]=0;
300                     }
301             }
302             /*for(i=0;i<=wm*2-1;i++){
303                     rutax[i]=wm*2;
304                     rutay[i]=wm*2;
305             }*/
306 }
307
308
309 //Creacion de la ruta de navegacion en el mapa
310 void navega(){
311             int posx=0;
312             int posy=0;
313             int i=1;
314             int min=mapa[posx][posy];
315
316             rutax[0]=posx+1;
317             rutay[0]=posy+1;
318             mini[0]=min;
319             int px=posx,py=posy;
320
321             while(posx!=wm && posy!=hm){ //hm-1
322                     //negativos
323                     //if(posx-1 <= wm-1 && posy <=hm-1){
324                 if((posx-1>=0 && posy>=0)){
325                     if(min > mapa[posx-1][posy]){
326                         px=posx-1;
327                         py=posy;
328                         min=mapa[posx-1][posy];
329                     }
```

```
330                 }
331                     //}
332                     //if(posx-1 <= wm-1 && posy-1 <=hm-1){
333                 if(posx-1>=0 && posy-1>=0){
334                     if(min > mapa[posx-1][posy-1]){
335                         px=posx-1;
336                         py=posy-1;
337                         min=mapa[posx-1][posy-1];
338                     }
339                 }
340             // }
341              //if(posx <= wm-1 && posy-1 <=hm-1){
342                 if(posx>=0 && posy-1>=0){
343                     if(min > mapa[posx][posy-1]){
344                         px=posx;
345                         py=posy-1;
346                         min=mapa[posx][posy-1];
347                     }
348                 }
349             // }
350                     //-------------------------
351
352                     //positivos
353                     //if(posx+1<=wm-1 && posy<=hm-1){
354                 if((posx+1>=0 && posy>=0)){
355                     if(min > mapa[posx+1][posy]){
356                         px=posx+1;
357                         py=posy;
358                         min=mapa[posx+1][posy];
359                     }
360                 }
361                     //}
362                     //if(posx<=wm-1 && posy+1<=hm-1){
363                 if((posx>=0 && posy+1>=0)){
364                     if(min > mapa[posx][posy+1]){
365                         px=posx;
366                         py=posy+1;
367                         min=mapa[posx][posy+1];
368                     }
369                 }
370                     //}
371                     //if(posx+1<=wm && posy+1<=hm){
372                 if(posx+1>=0 && posy+1>=0){
```

```
373                     if(min > mapa[posx+1][posy+1]){
374                         px=posx+1;
375                         py=posy+1;
376                         min=mapa[posx+1][posy+1];
377                     }
378                 }
379                     //}
380
381                     //------------------------------
382                     //mas menos
383                     //if(posx+1<=wm-1 && posy-1<=hm-1){
384                 if(posx+1>=0 && posy-1>=0){
385                     if(min > mapa[posx+1][posy-1]){
386                         px=posx+1;
387                         py=posy-1;
388                         min=mapa[posx+1][posy-1];
389                     }
390                 }
391                     //}
392                     //if(posx-1<=wm-1 && posy+1<=hm-1){
393                     if(posx-1>=0 && posy+1>=0){
394                     if(min > mapa[posx-1][posy+1]){
395                         px=posx-1;
396                         py=posy+1;
397                         min=mapa[posx-1][posy+1];
398                     }
399                 }
400             // }
401                     posx=px;
402                     posy=py;
403                     rutax[i]=posx;
404                     rutay[i]=posy;
405                     mini[i]=min;
406                     i++;
407             }
408 }
409
410 void SensarDis(){
411             istringstream(nxt->read_msg(2,true))>>cambio;
412             istringstream(nxt->read_msg(6,true))>>actualizaruta;
413                     Sleep(500);
414             if(actualizaruta == 1){
415                 creaMapa();
```

```
416                 obstaculo(x,y);
417                         expaOnda();
418                         navega();
419                         temp++;
420                         coorenadas();
421         }
422 }
423
424 void RGBtoGray(double* pGray,uchar* pRGB,int width,int height)
425 {
426         int index;
427         unsigned char r,g,b;
428
429         for(int j=0; j<height; j++){
430                 for(int i=0; i<width; i++)
431                 {
432                         index=j*width+i;
433                         b=pRGB[3*index];
434                         g=pRGB[3*index+1];
435                         r=pRGB[3*index+2];
436                         pGray[index]=((double)r+(double)g+(double)b)/3;
437                 }
438         }
439 }
440
441 void empilar(double* out,double* inp){
442         int k,j;
443
444         k=0;
445         for(int i=N; i>=1; i--){
446                 j=0;
447                 for(int c=0; c<M; c++){
448                         out[i+j-1]=inp[k];
449                         j+=N;
450                         k++;
451                 }
452         }
453 }
454
455 double norma(double* v){
456         double suma=0;
457
458         for(int k=0; k<(N*M); k++)
```

```
459                     suma+=pow(v[k],2);
460
461             return sqrt(suma);
462     }
463
464     void Leido(){
465             char datoVV[100];
466             char datoC[100];
467             int m=0;
468
469             FILE *inputC,*inputVV;
470             inputC = fopen("C.txt","r");
471             inputVV = fopen("VV.txt","r");
472
473             //Lectura de los vectores propios y
474             //del vector promedio
475             for(int i=0;i<98;i++){
476                     if(i==10*m){
477                             printf("=");
478                             m++;
479                     }
480                     for(int j=0;j<N*M;j++){
481                             fgets(datoVV,100,inputVV);
482                             VV[i][j]=atof(datoVV)/100000;
483                             if(i==0){
484                                     fgets(datoC,100,inputC);
485                                     C[j]=atof(datoC)/100000;
486                             }
487                     }
488             }
489         printf("Completo\n");
490             fclose(inputVV);
491             fclose(inputC);
492
493     }
```