



**UNIVERSIDAD
ANDRÉS BELLO**

**UNIVERSIDAD ANDRÉS BELLO
FACULTAD DE INGENIERÍA
ESCUELA DE INDUSTRIAS**

INGENIERÍA EN AUTOMATIZACIÓN Y ROBÓTICA

**“Simulación y Análisis de algoritmos de planificación y rutas para robot
móvil en Gazebo”**

AUTOR:

ÁLVARO ARANEDA MORALES.

Aaraneda.alvaro@gmail.com

989298999

PROFESOR GUÍA:

FELIPE CASTRO NIENY

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN AUTOMATIZACIÓN Y ROBÓTICA**

SANTIAGO – CHILE

ENERO, 2020

“Dedicado a mi familia, quienes me daban ánimos para seguir continuando, y a esas personas que conocí en el camino y formamos lindos lazos de amistad.”

Agradecimientos

Este trabajo está dedicado a todas las personas que me apoyaron a lo largo de este camino de forma emocional y que siempre estuvieron para mí, incluso en las últimas circunstancias. Siempre teniendo una mano amiga que me apoyaba y me daba energías para lograr conseguir terminar el proyecto, y que me preguntaban si tenía algún avance en mi tesis, algunas veces incluso se enojaban conmigo debido que no contaba cómo estaba preparado para el proyecto, pero estando enojado conmigo o no, jamás estuve solo porque me apoyaban de la forma que pudieran, a mi madre mi hermana mi pareja y padre que son los pilares fundamentales en mi vida. y por último quiero agradecer a las personas que me ayudan económicamente y que me permitieron seguir estudiando la carrera de ingeniera en automatización y robótica. Toda esta etapa me logro hacer crecer mucho como persona e intelectualmente, y a demostrarme a mí mismo que nada es imposible si uno lucha por ello cueste lo que cueste, ya si lo logras serás la persona más feliz del mundo en ese momento porque te atreviste y dejaste todas las críticas para seguir hasta el final.

INDICE DE CONTENIDOS

Agradecimientos.....	4
Introducción.....	6
Objetivo general	7
Objetivos específicos	8
Robótica móvil.....	9
Locomoción.....	9
Tipos de ruedas	13
Disposición de las ruedas	15
Tracción y dirección	21
Posicionamiento y navegación	23
Localización y mapeo simultáneo (SLAM)	24
Planificación de trayectorias.....	26
Vehículo terrestre no tripulado	32
Aplicaciones de la robótica móvil	44
Control de movimiento y evasión de obstáculos	50
Obstáculos estáticos.....	51
Bug algorithm.....	51
Algoritmo de búsqueda en profundidad (DFS).....	52
Algoritmo de búsqueda en anchura (BFS).....	55
Divide & conquer (divide y vencerás)	57
Bubble rebound algorithm	58
Robot operating system (ROS)	59
Desarrollo	62
Algoritmo bubble rebound	64
Divide and conquer	71
Pruebas y comparación de algoritmos	78
Tablas con los resultados para cada algoritmo y prueba	89
Conclusiones.....	92
Bibliografía	93

Introducción

La robótica móvil se considera actualmente un área de tecnología avanzada ya que son un gran apoyo hoy en día para el ser humano en distintas aplicaciones ya sea en rescate y emergencia como lo es el robot fabricado por milremrobotics con su robot Multiscope que está diseñado para ayudar a bomberos en la lucha contra los incendios, otras aplicaciones como es el caso del robot Themis diseñado para dar apoyo a la infantería militar, transporte, entretenimiento, entre otras. Esta tecnología ha ido creciendo de manera rápida, tal así que son indispensables en el uso industrial ayudando a mejorar los procesos productivos, y mejorar las condiciones de trabajo.

En el presente proyecto se abordará la investigación acerca de la robótica móvil terrestre, como lo es en su estructura física y mecánica, además se mostrara el tipo de algoritmo necesario para calcular la ruta del robot y evadir obstáculos no conocidos ya que la mayoría de los entornos en el mundo real son variables, es decir, son entornos dinámicos, más aún se dará a conocer algunos modelos matemáticos para la robótica móvil, con esto quiero decir cuáles son las partes principales y necesarias de un sistema robótico móvil y cuales son algunas aplicaciones de este sistema en el transcurso de la historia mostrando esta información en pequeños ficheros.

Objetivo general

- Elaborar una simulación del robot husky en gazebo para situaciones de búsqueda y rescate en donde sus entornos son variables.

Objetivos específicos

- Usar sensor Lidar, para la localización y mapeo simultánea.
- Emplear algoritmos de planificación local para evadir obstáculos imprevistos y con ellos conseguir la mejor trayectoria en el menor tiempo posible al punto de deseado.
- Comparar algoritmos en distintitos entornos de trabajo.

Robótica móvil

En el presente capítulo se abordará la investigación acerca de la robótica móvil terrestre, como lo es en su estructura física y mecánica, además se mostrará el tipo de posicionamiento y navegación de un robot, más aún se dará a conocer algunos modelos matemáticos para la robótica móvil, con esto quiero decir cuáles son las partes principales y necesarias de un sistema robótico móvil y cuáles son algunas aplicaciones de este sistema en el transcurso de la historia mostrando esta información en pequeños ficheros.

Locomoción


Según el área de trabajo el entorno del robot puede ser exterior o interior. Dicho de otra manera, el sistema de locomoción del robot se verá afectado según el terreno donde esté operando, en este proyecto investigativo se trabajará y simulará un terreno con distintos obstáculos los cuales el robot los tiene que superar sin complejidad. De modo que el sistema debe ser con ruedas, patas u orugas. En el presente artículo se muestra en forma breve el estado del arte de la robótica móvil, y algunas aplicaciones de los robots móviles en campos como la industria, seguridad, espacial, entre otros. Es necesario recalcar que se analizarán los sistemas de locomoción de la robótica móvil terrestre.

Robots móviles con locomoción a través de ruedas	Datos
	<p>“El mayor desarrollo está en los Robots Móviles con Ruedas (RMR), esto es debido a las ventajas que presentan las ruedas respecto a las patas y a las orugas. Dentro de los atributos más relevantes de los RMR, destacan su eficiencia en cuanto a energía en superficies lisas y firmes, a la vez que no causan desgaste en la superficie donde se mueven y requieren un número menor de partes, normalmente menos complejas en comparación con los robots de patas y de orugas, lo que permite que su construcción sea más sencilla.” (Barrientos Sotelo, V., & García Sánchez, J., & Silva Ortigoza, R., 2007)</p>
<p>https://www.redalyc.org/pdf/4026/402640448003.pdf</p>	
<p>C://Robots móviles/Locomoción/Ruedas</p>	

Las ruedas son unos de los componentes más importantes dentro de los robots, ya que estas proporcionan la tracción del robot la cual abordare en los siguientes capítulos.

Además, las ruedas en los robots permitieron de pasar de un robot manipulador fijo a un robot móvil que se pueda desplazar dentro de un entorno sin la ayuda de un operador. También cabe señalar los tipos de ruedas que emplean los RMR, Dentro de estas podemos encontrar dos clasificaciones: ruedas convencionales y ruedas omnidireccionales.

El siguiente aspecto trata de la locomoción de tipo de oruga, este sistema consta con una mayor área de contacto con la superficie, por lo tanto, supone una mejor maniobrabilidad y tracción que las ruedas y una movilidad superior a la locomoción por patas.

Robots móviles con locomoción tipo oruga o cadenas	Datos
	<ul style="list-style-type: none"> • "La pista en sí está hecha de material blando, por lo que puede deformarse de acuerdo con la forma de cualquier tipo de escaleras, eso ayuda a maximizar la superficie de contacto, haciéndola muy estable" (Coxworth, May 28th, 2019) • Capaz de levantar cargas de 6 kilogramos, su cuerpo esta formado de un material blando con el fin de resistir golpes y caídas. Es posible subir y bajar las escaleras (escalón: máximo 18 cm, inclinación: máximo 35 °)
<p>En el artículo aborda el prototipo actual de Amoeba, presenta pistas similares a las orugas hechas de un material esponjoso pero durable EPDM (etileno propileno dieno monómero). En el cual quieren solucionar las entregas de paquetes al hogar, remplazando con este robot a las operaciones que realizan dichas funciones.</p>	
<p>https://newatlas.com/amoeba-soft-delivery-robot/59888/</p>	
<p>C://Robots móviles/Locomoción/Orugas</p>	

En conclusión, los robots orugas son factibles para la entrega de productos en un posible futuro, dado sus prestaciones tales como la facilidad para subir escalera llevando una carga en sí mismo. Además, supone un diseño mecánico más simple que el de las patas o ruedas, esto debido a que no necesita una suspensión y trabaja con solo dos actuadores.

Robot móvil con locomoción por patas: Titan VIII	Datos
	<ul style="list-style-type: none"> • robot caminante cuadrúpedo desarrollado en el Tokyo Institute of Technology de Japón (Hirose y Kato, 1998). • “El robot TITAN VIII utilizaba detectores en sus pies; esto es una buena solución para cruzar un campo minado, pero resulta ineficiente en la exploración de todo un campo minado” (Roberto Ponticelli, 2011)
<p>En este trabajo de un robot cuadrúpedo se presenta la solución para minas terrestres, el robot era operado por un operario el cual manejaba el robot por un terreno con fin de encontrar minas y desactivarlas, fue uno de los pioneros en ser catalogado como desactivador de bombas ya que antes de él se creó el COMET-I, pero este era un hexápodo y muchísimo más grande llegando a pesar 120 kilogramos, dificultando la tarea de detección de minas.</p>	
<p>https://eprints.ucm.es/12318/1/T32658.pdf</p>	
<p>C://Robots móviles/Locomoción/Patas</p>	

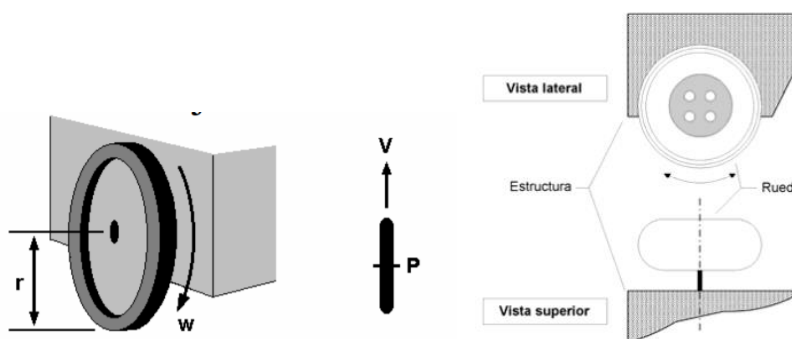
En lo que toca a locomoción por patas, esta tiene una ventaja sobre los robots móviles por rueda (RMR). Permitiendo aislar el cuerpo del robot del terreno usando solo algunos puntos de soporte. Teniendo en cuenta que el robot tiene otra configuración, esto le permite atravesar terrenos difíciles llenos de obstáculos, pero la locomoción es mucho menor en comparación al RMR. Algunas configuraciones de robots móviles por patas son: bípedos, cuadrúpedos, hexápodos.

Tipos de ruedas

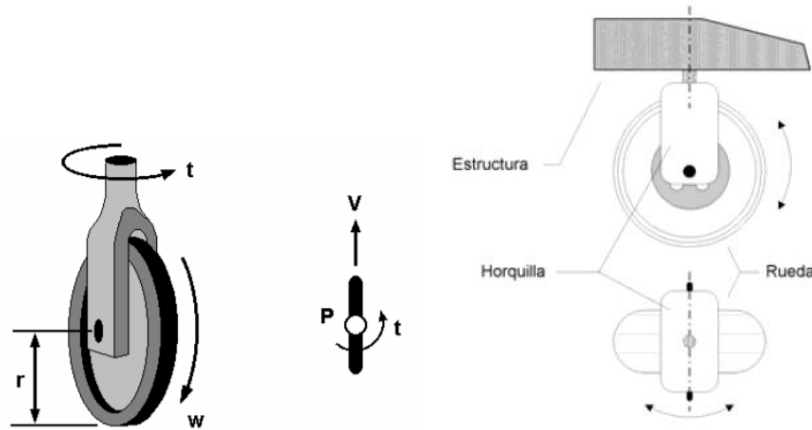
Entre los robots terrestre con ruedas su movilidad está caracterizada por dos factores: el tipo de rueda con el cual se moverá y la disposición de dichas ruedas en la estructura del robot.

Se asume que, durante el movimiento el plano de la rueda se mantiene vertical y que las ruedas rotan alrededor de su eje (horizontal), que tiene una orientación con respecto a la estructura que puede ser fija o variable. De las cuales podemos encontrar 2 clasificación: ruedas convencionales y ruedas no convencionales. En las ruedas de tipo convencional encontraremos 3 categorías de ruedas:

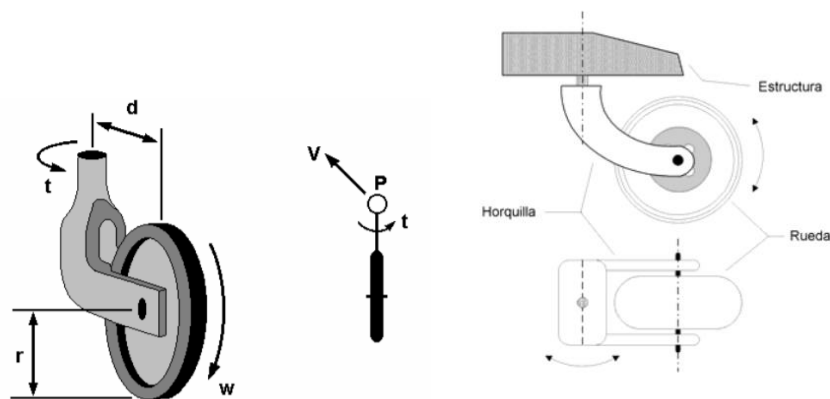
- Rueda fija: el eje de la rueda esta fijo a la estructura del robot, por lo general esta asociada al sistema de tracción del robot debido que en su mayor parte son utilizadas y/o conectadas a motor.



- Rueda orientable centrada: Es aquella rueda que tiene articulación de dirección, es decir es orientables respecto a la estructura del vehículo, pasando su eje de dirección por el centro de rotación de la rueda.



- Rueda orientable no-centrada (rueda loca): es una rueda orientable con respecto a la estructura, tal que la rotación del plano de la rueda es alrededor de un eje vertical el cual no pasa a través del centro de la rueda, su principal función es la de proporcionar dirección al robot.



Disposición de las ruedas

Según la disposición de las ruedas en el robot nos hace tener una gran variedad de robots móviles que se diferencian por su grado de maniobrabilidad las cuales pueden ser: omnidireccional, unicycle, triciclo, cuatriciclo. A continuación, se comentan brevemente las características más significativas y comunes de los robots móviles terrestres. Sin embargo, se ha limitado a ejemplos concretos.

Robot omnidireccional.

Este vehículo móvil es capaz de desplazarse en cualquier dirección, esto quiere decir, que el robot tiene máxima maniobrabilidad en el plano. A continuación, se muestra una breve ficha de un robot omnidireccional.

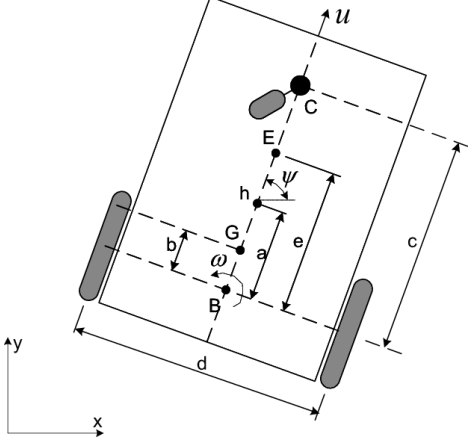
Robot omnidireccional	Datos
	<ul style="list-style-type: none">• “Robotino® ha sido diseñado por FestoDidactic, siendo una herramienta didáctica que permite manejar un robot a nivel académico con una diversa gama de sensores que facilitan la realización de diferentes tareas de control.” (Narváez V. Yandún F. Pozo D. Morales L. Rosero J. Rosales A. Auat F., DICIEMBRE 2014)• Cuenta con tres unidades de accionamiento omnidireccional permitiéndole movimientos en

	<p>todas las direcciones: adelante, atrás, y lateralmente, considerando que además puede girar sobre su propio eje. Cuenta con 3 grados de libertad.</p>
<p>En este artículo se describe desarrollo e implementación en matlab de un algoritmo de localización y mapeo simultáneo (slam) para un robot móvil comercial, para este caso se trabajó con el robot educativo robotino fabricado por Festo.</p>	
<p>https://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/178</p>	
<p>C://Robots móviles/Locomoción/Rueda/Disposición de las ruedas/Robot omnidireccional</p>	

Las ventajas de un robot omnidireccional se ven disminuidas por la complejidad mecánica y/o electrónicas necesarias para conservar una buena coordinación entre las ruedas y evitar derivas en la pose del robot.

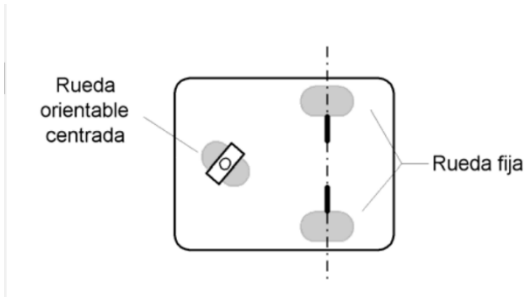
Robot Uniciclo.

El robot tipo uniciclo es, en general, es frecuentemente utilizado en tareas debido a su simplicidad y su buen movimiento dentro del plano son comúnmente vistos en áreas como la vigilancia, limpieza de pisos, transportes de cargas. Su estructura consta de dos ruedas fijas y una rueda loca que aporta estabilidad y libertad al propio robot.

Robot Unicycle	Datos
	<ul style="list-style-type: none"> robot móvil tipo unicycle, sus parámetros y variables de interés. En la figura, u y ω son, respectivamente, las velocidades lineal y angular, G es el centro de masa, C es la posición de la rueda castor, E es la localización de una herramienta a bordo del robot, h es el punto de interés (de coordenadas x y y en el plano XY), ψ es la orientación del robot, y a es la distancia entre el punto de interés y el punto central del eje virtual que conecta las ruedas (punto B). (Felipe Nascimento Martins, Ricardo Carelli, Mario Sarcinelli-Filho, Teodiano Freire Bastos, noviembre de 2008)
<p>Este artículo presenta el desarrollo de un controlador dinámico adaptable de seguimiento de trayectorias para robots móviles de tipo unicycle.</p>	
<p>https://www.researchgate.net/profile/Felipe_Martins3/publication/228415968_Un_Controlador_Dinamico_Adaptable_de_Seguimiento_para_Robots_Moviles_tipo_Unicycle/links/0a85e53a127df3fc1e000000.pdf</p>	
<p>C://Robots móviles/Locomoción/Rueda/Disposición de las ruedas/Robot Unicycle</p>	

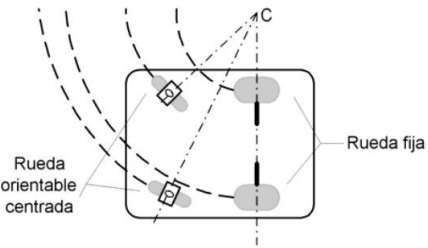
Robot triciclo.

La configuración de triciclo más común cuenta con una rueda delantera orientable centrada y dos ruedas fijas, la rueda delantera proporciona tracción como el direccionamiento. La importancia de este tipo de configuración es entregar una plataforma estable. Una base corta del robot genera un giro muy pequeño, hablase de base a la distancia de las ruedas fijas con la rueda de direccionamiento. Al igual que en caso anterior su estructura mecánica es sencilla.


Robot Triciclo	Datos
 <p>El diagrama muestra un robot triciclo con una rueda orientable centrada y dos ruedas fijas. La rueda orientable centrada está etiquetada como 'Rueda orientable centrada' y las dos ruedas fijas están etiquetadas como 'Rueda fija'. Una línea vertical punteada indica el eje de simetría del robot.</p>	<ul style="list-style-type: none">• La tracción y dirección son brindadas por un par de motores de corriente directa, los cuales se alimentan a 24 V. El primero puede desarrollar hasta 500 W de potencia, y el segundo hasta 100 W.• La velocidad desarrollada máxima es de 3.6 km/h a plena carga
<p>En este trabajo se presenta los principales cuidados para el transporte de materiales radiactivos el cual consta con una configuración tipo triciclo. De igual manera, se muestra que la maniobrabilidad del vehículo es adecuada para su teleoperación en los ambientes por donde la carga irradiada se debe hacer circular</p>	
<p>https://www.osti.gov/etdeweb/biblio/20662409</p>	
<p>C://Robots móviles/Locomoción/Rueda/Disposición de las ruedas/Robot Triciclo</p>	

Robot cuatriciclo.

La configuración cuatriciclo o también conocida como Ackerman es utilizada en vehículos de cuatro ruedas convencionales. Esta distribución de las ruedas soluciona los pequeños problemas que tiene el de tipo triciclo ya que en ciertas ocasiones su centro de gravedad se encuentra en los extremos lo cual puede ocasionar un volcamiento del robot.

Robot cuatriciclo o Ackerman	Datos
	<ul style="list-style-type: none">• Pruebas realizadas en robot lego NTX• Palabras clave: Robots móviles, localización, fusión de datos, filtros de Kalman, sistema de posicionamiento global, sistemas basados en eventos, recursos limitados, LEGO NXT.
En el presente trabajo se expone la implementación de un algoritmo de fusión de datos provenientes de un sistema de posicionamiento global con la estimación de la posición mediante sensores inerciales con el fin de mejorar la localización de robots móviles de recursos computacionales limitados y usando una configuración Arckerman	
https://www.osti.gov/etdeweb/biblio/20662409	
C://Robots móviles/Locomoción/Rueda/Disposición de las ruedas/Robot Cuatriciclo	

Morfología diseño de chasis

Diseño de chasis para vehículos	Datos
	<ul style="list-style-type: none">• En este proyecto se describen cuáles han sido los pasos dados para alcanzar el diseño final de un chasis.
<p>El chasis de un vehículo constituye su masa suspendida, es decir, la que no tiene contacto directo con la superficie de rodaje; en él se instalan los mecanismos (motores, ruedas, suspensión, sensores, circuitos, baterías, sistemas de visión). (Nasly Perez , Diego Salamanca, 2009)</p>	
<p>file:///C:/Users/aaran/Downloads/T44.09%20P415d.pdf</p>	
<p>C://Robots móviles/Chasis</p>	

Cabe señalar que mientras más pesado el chasis del robot móvil, a este le costara moverse mucho más por lo que su velocidad es menor y su costo de energía mucho mayor. En la construcción del chasis existen tres parámetros fundamentales los cuales son: ligereza, rigidez, economía. En primer lugar, la ligereza puede mejorar la potencia y el rendimiento de los actuadores. En segundo lugar, la rigidez es el parámetro fundamental de funcionamiento del chasis, es importante conseguir una estructura resistente a impactos para la protección del robot. Por último, la economía se refiere a los materiales que se usaran para la fabricación del chasis.

Tracción y dirección

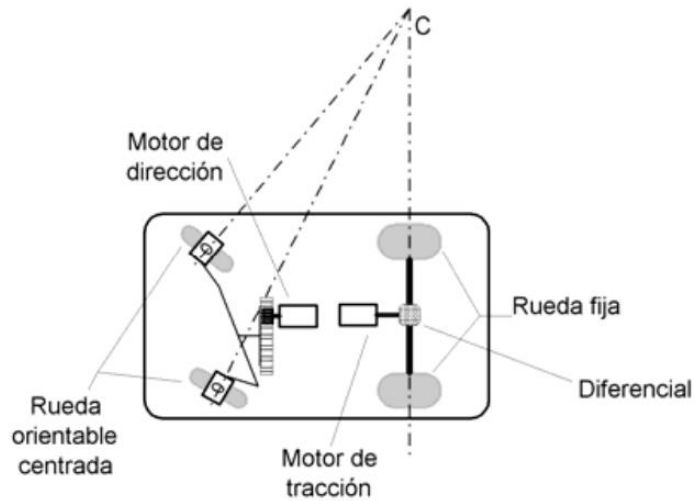
Uno del aspecto analizar en la construcción y/o simulación de un robot móvil es el sistema tracción y dirección de este mismo dado que estos términos van de la mano con el posicionamiento de las ruedas. Además, mientras más confiabilidad, es decir, alta maniobrabilidad, máxima tracción en sus ruedas, máxima adherencia de todas sus ruedas con el plano, entre otras. La mecánica, electrónica e informática es más compleja debido que si se quieren mejorar los aspectos antes mencionados se necesitan más recursos en la construcción del robot móvil.

Existen tres tipos básicos a parte de los cuales se pueden obtener diversas configuraciones:

- Tracción y dirección con ejes independientes
- Tracción y dirección en un mismo eje
- Tracción y dirección sobre todos los ejes

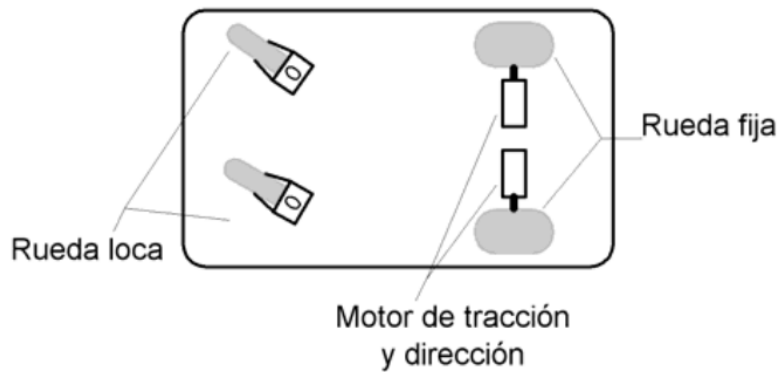
Tracción y dirección con ejes independientes

En el siguiente caso la tracción se efectuará en las ruedas traseras y el control de dirección en las ruedas delanteras, esto es muy común verlo en los vehículos de hoy en día y esto se logra conseguir gracias al diferencial que permite que las ruedas funcionen a distintos rpm. Además, posee un radio de giro bastante elevado en relación con otros sistemas, por lo que en este tipo de modelo no se pueden lograr cambios de dirección muy cerrados.



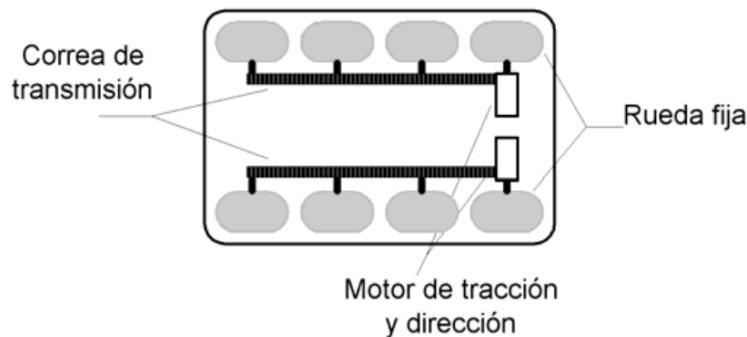
Tracción y dirección en un mismo eje (Tracción diferencial)

Esto se logra con motores independientes en las ruedas de un mismo eje y ruedas "locas" en el resto de los ejes. Este modelo es de construcción sencilla y permite radios de giro del orden del tamaño del vehículo. La única desventaja que posee es que los motores deben ser de características idénticas, para que el control de estos sea simple. Más adelante veremos que esta es una de las configuración de tracción que cuenta el robot móvil del simulador.



Tracción y direccionamiento sobre todo los ejes

Esta configuración necesita de un sistema odométrico complejo debido a la incertidumbre en los radios de giro asociada a este sistema de tracción y dirección, aunque existen entre los robots omnidireccionales estructuras que presentan menor complejidad para resolver los errores por odometría.



Posicionamiento y navegación

La navegación es la técnica que permite a un robot móvil desplazarse de un punto a otro en un entorno con obstáculos para poder llegar a su destino.

La complejidad de la navegación autónoma de un robot móvil depende de gran parte de la información obtenida de su entorno. Así, en un entorno que se tenga la información necesaria y que similar a la realidad, la creación de una trayectoria adecuada y su seguimiento son tareas que no presentan un alto grado de incertidumbre ya que en el mundo real la mayoría de los entornos son no estructurados. Por lo cual se necesita la capacidad del robot móvil para actuar ante las variaciones del medio que los rodea. De esta necesidad se pueden encontrar dos tipos de navegación en un entorno dinámico: navegación reactiva y navegación deliberativa.

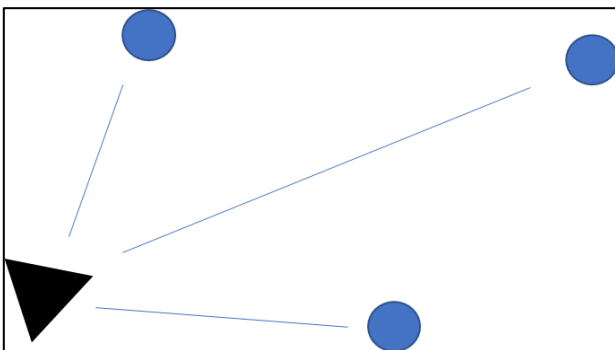
La navegación reactiva consiste en responder de manera inmediata a los estímulos del entorno obtenidos por los sensores del robot móvil, según dicha información obtenida se activan uno o más comportamientos simples. Por otro lado, la

navegación deliberativa consiste en la utilización de una representación o modelo espacial del entorno para basar todas las decisiones de navegación, que permite el mapeado del entorno mediante la información de sensores, la localización o estimación de posición del robot dentro del mapa y planificación de una trayectoria libre de obstáculos. La mayoría de los algoritmos de planificación de trayectorias no se basan solo en una de las metodologías antes mencionada, por lo cual hacen la utilización de las dos, dando origen a una navegación híbrida.

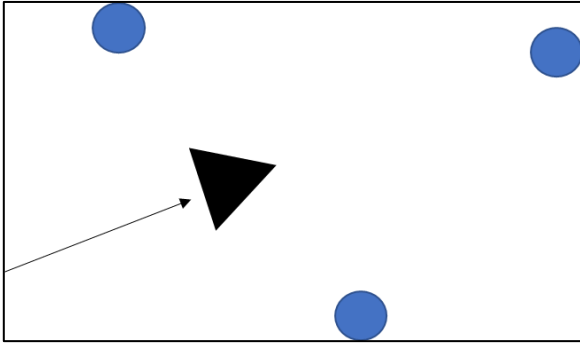
Localización y mapeo simultáneo (SLAM)

SLAM, o localización y mapeo simultáneo, es la técnica usada por robots autónomos para localizarse en un entorno, parcial o totalmente desconocido, a su vez va adquiriendo información de su entorno a través de los sensores que el robot posea y estimando su trayectoria al desplazarse dentro de su entorno. A continuación, se muestra una pequeña ilustración para una mejor comprensión, mostrando los posibles pasos en un proceso de SLAM.

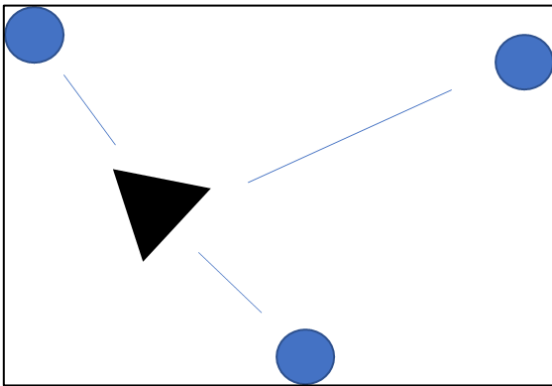
El robot comienza escaneando el entorno, recogiendo información sobre diferentes puntos de referencias y calculando su distancia relativa a ellos.



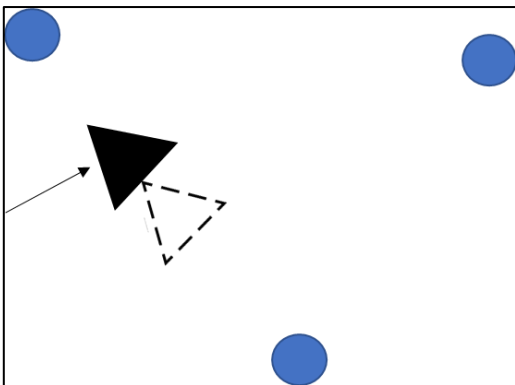
En el momento que el robot se mueve, hace uso de la odometría para estimar la nueva posición del robot dentro del entorno.



El robot vuelve a escanear con el láser el entorno, buscando los puntos de referencias ya encontrados anteriormente y calculando las nuevas distancias relativas.



Las cuales son utilizadas para calcular de manera más exacta su nueva localización dentro de su entorno donde esté operando.



Uno de los mayores retos de la localización y mapeo simultaneo son la acumulación de errores proveniente de la odometría, el problema de identificar correctamente la

nueva localización de los puntos de referencia y los cambios del entorno a medida que transcurre el tiempo.

En ROS existen actualmente varios algoritmos de Slam para sensores laser ya desarrollados. Entre los más destacados se encuentran HectorSLAM, Gmapping y Google Cartographer.

Planificación de trayectorias

Se define como planificación de trayectorias al resultado del problema de encontrar un camino libre de obstáculos por el cual el robot móvil se pueda mover y llegar a su destino final y con ello cumplir su objetivo a su vez poder minimizar cierto índice de coste, ya sea la distancia recorrida o el consumo energético del robot móvil.

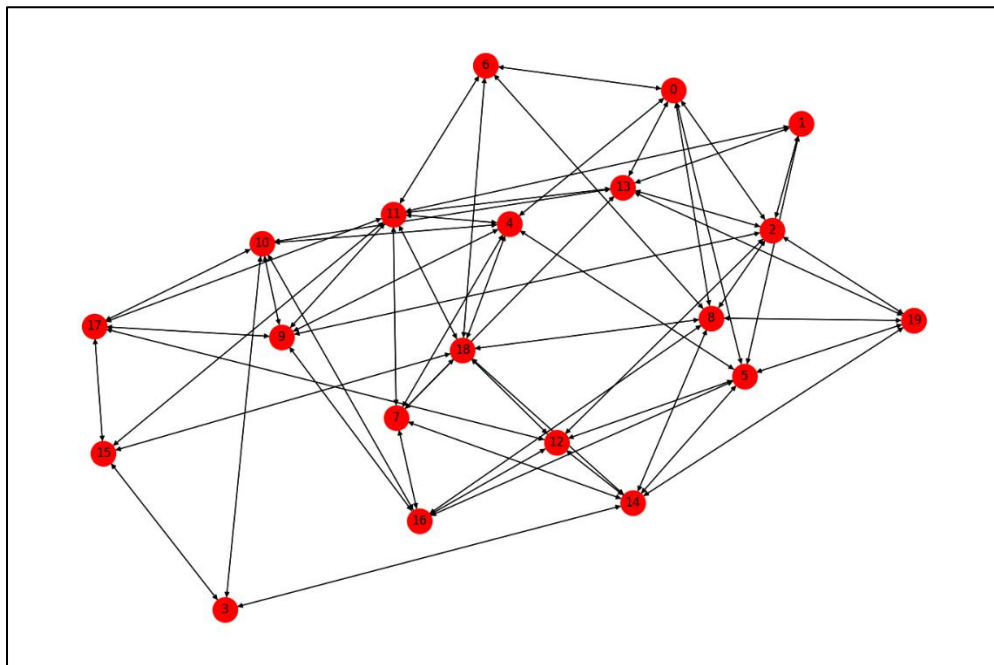
En un entorno conocido e invariable se puede guiar al robot y tener una única trayectoria, sin embargo, en un entorno no estructurado en donde los objetos que se encuentran en la escena pueden sufrir variaciones es necesario distinguir dos tipos de planificaciones de trayectorias: planificación global y planificación local.

La planificación global tiene en cuenta la información proporcionada en el instante inicial del movimiento, es decir, el mapa del entorno y lo que vean los sensores en dicho instante de tiempo, con el fin de desarrollar una trayectoria que conduzca desde la posición inicial hasta la posición final. Por otro lado, la planificación local realiza modificaciones a la planificación global para esquivar obstáculos no previstos en el camino. El conjunto de dichas planificaciones da lugar a la trayectoria final.

Los métodos de planificación de trayectorias contienen una etapa inicial de representación del entorno mediante algún tipo de grafo, tras la cual se emplea un algoritmo de búsqueda de grafos.


Los algoritmos de búsqueda son aquellos que están diseñado para la toma de decisiones, ya sea para localizar elementos en una base de datos o para seleccionar la mejor ruta. Estos algoritmos se componen de grafos los cuales se componen de


nodos y aristas que las unen si, lo cual representa cual es la relación de un nodo con otro nodo. Un grafo puede ser dirigido, es decir, sus aristas conectan puntos de origen con puntos de destino, o no dirigido, entendiéndose que no hay distinción entre los dos nodos que conecta cada una de ellas. Se da a presentar en la siguiente imagen como es un grafo.

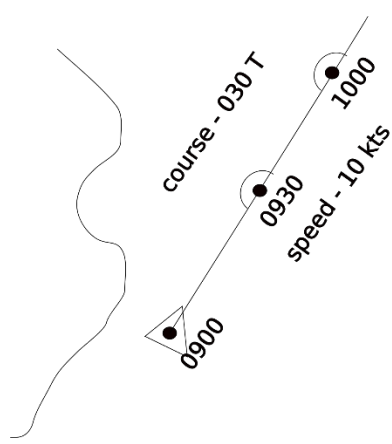


<p>Mapeo y localización simultanea de un robot para levantamiento de mapas en interiores.</p>	<p>Datos</p>
	<ul style="list-style-type: none"> • “Como la odometría del vehículo es frecuentemente errónea, se realiza paralelamente la localización del mismo, a este proceso se le denomina slam (Simultaneous Localization and Mapping). El SLAM se fundamenta en una sucesión de pasos, donde el objetivo para

	<p>este caso es utilizar las líneas rectas que describen el ambiente donde se mueve el robot para restablecer su posición” (Julie Stephany Berrío Perez , Eduardo Francisco Caicedo Bravo, Lina Maria Paz Perez, 2013)</p>
<p>El presente artículo muestra el desarrollo teórico y práctico de un algoritmo que da solución al problema del SLAM basado en la extracción de características del medio ambiente, las cuales son representadas por líneas rectas</p>	
<p>http://www.laccei.org/LACCEI2013-Cancun/RefereedPapers/RP039.pdf</p>	
<p>C://Robots móviles/Posicionamiento y navegación/Slam</p>	


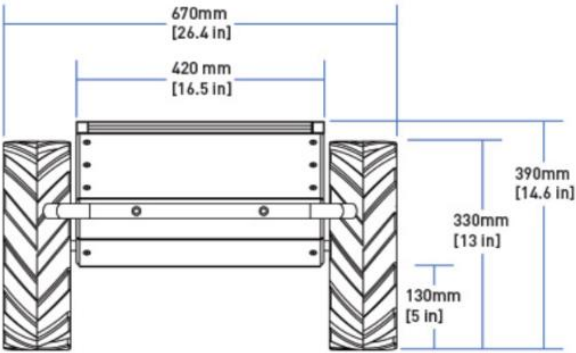
<p>Sensor laser sick lms5xx</p>	<p>Datos</p>
	<ul style="list-style-type: none"> • Fabricado por Sick • Protección IP67, IP65 • Tecnología de 5 ecos • Angulo de abertura 190°
<p>En el documento se presenta las ventajas de trabajar con un sensor sick, tales como: navegación y localización geográfica, seguridad, tráfico y cuáles son sus característica y dimensiones para cada sensor.</p>	
<p>https://www.sick.com/cl/es/soluciones-de-medicion-y-deteccion/sensores-2d-lidar/lms5xx/c/g179651</p>	
<p>C://Robots móviles/Posicionamiento y navegación/sensor lidar</p>	

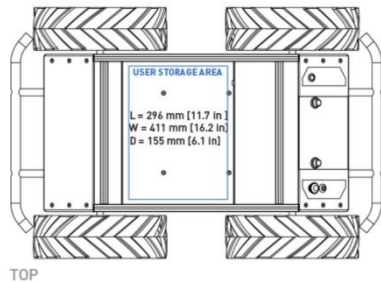
Mapeo autónomo en las minas peligrosas de Chile	Datos
	<ul style="list-style-type: none"> • “A medida que los robots se vuelvan más avanzados y, en última instancia, autónomos, los humanos podrán alejarse de trabajos peligrosos en la mina, como mapear un territorio desconocido.” (clearpathrobotics)
<p>En el presente artículo desarrollo como a través de radar, laser y visión sin ruido, permite que personas en movimiento estén lejos de tareas mineras peligrosas. Con lo cual los accidentes son mucho menores y solucionando problemas donde humanos no podían acceder debido al difícil acceso.</p>	
<p>https://clearpathrobotics.com/autonomous-mapping-chile-mines/</p>	
<p>C://Robots móviles/Posicionamiento y navegación</p>	

<p>Localización precisa usando radio UWB y Dead-reckoning para cooperación eficiente humano-robot</p>	<p>Datos</p>
 <p>The diagram illustrates a navigation path starting from a point labeled '0900' (marked with a triangle) and moving towards two subsequent points labeled '0930' and '1000' (marked with circles). The path is a straight line with a heading of 'course - 030 T' and a speed of 'speed - 10 kts'. A wavy line on the left side of the path represents a boundary or obstacle.</p>	<ul style="list-style-type: none"> • “El hecho de usar información inercial es muy importante ya que posibilitaría una fusión sensorial entre medidas de rango (relativamente ruidosas) y medidas dead-reckoning (precisas a corto plazo) para mejorar la continuidad y robustez en las estimaciones.” (Antonio R. Jimenez, Fernando Seco, septiembre de 2015)
<p>En este trabajo se presenta un método sensorial para medir la distancia y el posicionamiento relativo, e incluso absoluto, entre dos nodos electrónicos independientes (humano y robot) usando radio UWB y medidas dead-reckoning.</p>	
<p>https://www.ehu.eus/documents/3444171/4484749/23.pdf</p>	
<p>C://Robots móviles/Posicionamiento y navegación/Dead-reckoning</p>	

Estimación de la Posición de un Robot Móvil	Datos
	<ul style="list-style-type: none"> • “El sistema GPS adquiere datos en tiempo real de la posición, velocidad y altitud para navegación autónoma, usando diferentes métodos y algoritmos que ayudan al robot a reconocer el entorno que le rodea y, con ello, lograr definir la mejor trayectoria para alcanzar un objetivo”. (D. Reyes, G. Millán, R. Osorio, G. Lefranc, junio 2015)
<p>En este artículo se revisan los métodos existentes para la estimación de la posición de robots móviles y vehículos autónomos. Menciona que tan importante es el GPS en el robot para la planificación de rutas y a su vez la evasión de obstáculos en la trayectoria. Cabe mencionar que el GPS solo determina su posición, no así su orientación del robot.</p>	
<p>https://www.researchgate.net/publication/280404763_Mobile_robot_navigation_assisted_by_GPS</p>	
<p>C://Robots móviles/Posicionamiento y navegación/gps</p>	

Vehículo terrestre no tripulado

Husky ugv A200	Datos
  <p>SIDE</p>  <p>FRONT</p>	<ul style="list-style-type: none"> • Robot fabricado por Clearpath robotics con sede en Canadá. • Máxima velocidad 1.0 m / s. • Tiempo de ejecución (uso típico) 3 horas. • Controladores y api ROS, C ++, Mathworks • Peso 50 kg • Máxima carga útil 75kg • Carga útil todo terreno 20kg • Dimensiones externas 990 x 670 x 390 mm • Dimensiones internas 296 x 411 x 155 mm




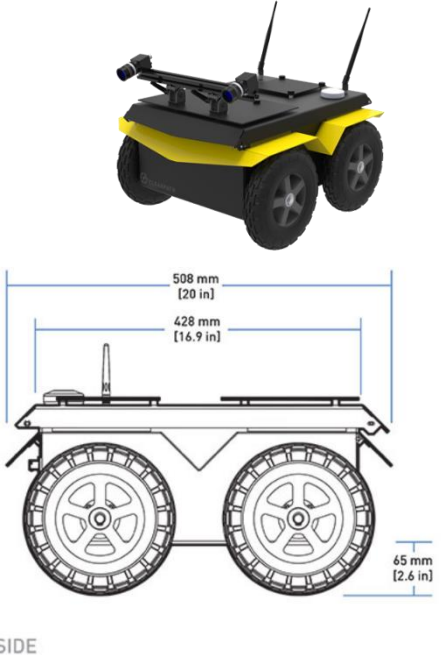
- Grado de ascenso 45°
- Tiempo de carga 10hr
- Comunicación RS-232
115200 Baud

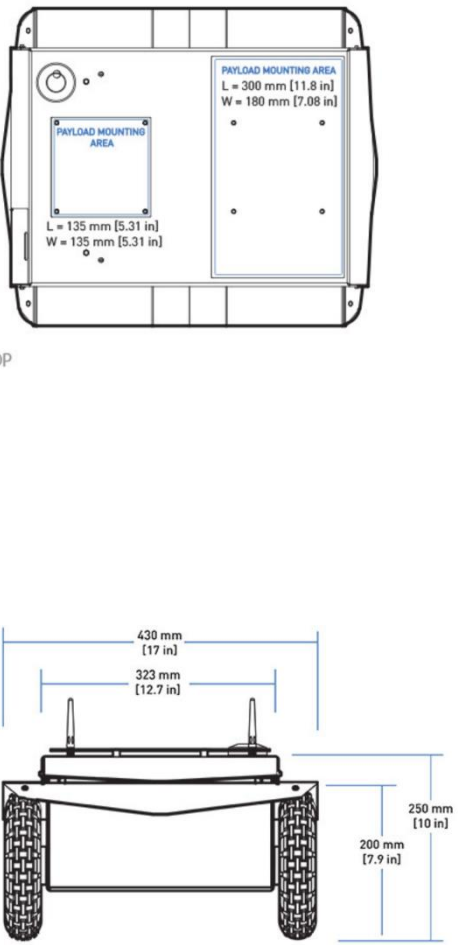
“Husky es un vehículo terrestre no tripulado (UGV) resistente y preparado para el exterior, adecuado para investigación y aplicaciones de creación rápida de prototipos. Husky utiliza un protocolo serie de código abierto y ofrece compatibilidad con API (interfaz de programación de aplicaciones) para ROS y opciones para C ++ y Python.” (Clearpath, 2015). En el artículo se muestra las capacidades y los distintos tipos de equipamiento que se le pueden añadir al robot, los cuales mientras más equipos tenga mejor se puede adaptar a los terrenos que este tiene que enfrentar. Cabe mencionar que el robot es resistente y está diseñado para exteriores, es fácil de manejar y es personalizable como antes lo mencionaba con sus equipos. La principal función del robot es de investigación, pero también puede realizar las siguientes tareas las cuales lo hacen referente en el campo: teleoperación, manipuladores autónomos o tele operados, navegación, sistemas multi robots.

<http://www.clearpathrobotics.com/assets/guides/husky/>

C://Robots móviles/UGV/Clearpath/Husky A200

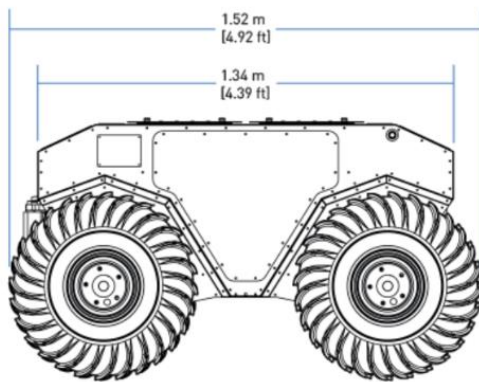
Aplicación de robot husky	Datos
	<ul style="list-style-type: none"> • Robot fabricado por Clearpath robotics con sede en Canadá. • Diseñado para gestionar los viñedos. • Cuenta con sensores como: cámaras visuales, multiespectrales, láseres de distancia, sensores inerciales y receptores GPS. Además, tiene un brazo robótico el cual es utilizado para evitar plagas y reproducción de insectos.
<p>En el artículo se señala como un robot móvil puede ayudar y ser un completo para realizar tareas vitícolas, consiguiendo un apoyo para el agricultor en diversos ámbitos de gestión de la viña.</p>	
<p>https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/</p>	
<p>C://Robots móviles/UGV/Clearpath/Husky A200</p>	

Jackal	Datos
 <p data-bbox="259 997 308 1018">SIDE</p>	<ul style="list-style-type: none"> <li data-bbox="893 378 1380 472">• Robot fabricado por Clearpath robotics con sede en Canadá. <li data-bbox="893 598 1331 640">• Máxima velocidad 2.0 m / s. <li data-bbox="893 766 1380 861">• Tiempo de ejecución (uso típico) 2 horas. <li data-bbox="893 987 1380 1081">• Controladores y api ROS, Mathworks <li data-bbox="893 1207 1104 1249">• Peso 17 kg <li data-bbox="893 1375 1266 1417">• Máxima carga útil 20kg <li data-bbox="893 1501 1331 1543">• Carga útil todo terreno 20kg <li data-bbox="893 1669 1380 1764">• Dimensiones externas 508 x 430 x 250 mm

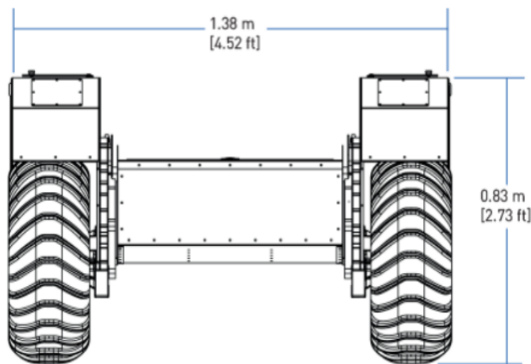
 <p>TOP</p> <p>FRONT</p>	<ul style="list-style-type: none"> • Dimensiones internas 250 x 100 x 85 mm • Poder de conducción 500W • Temperatura ambiente de funcionamiento -20 a 45°C • Tiempo de carga 4hr • Comunicación RS-232, Ethernet, Usb 3.0
<p>“Jackal es una plataforma de investigación de robótica de campo pequeña, rápida y de nivel de entrada. Cuenta con una computadora a bordo, GPS e IMU totalmente integrada con ROS para una capacidad autónoma fuera de la caja. Al igual que con todos los robots Clearpath, Jackal es compatible con plug-and-play con una enorme lista de accesorios de robot para expandir rápidamente su investigación y desarrollo.” (Clearpath, 2015). Este robot es ideal para estudiantes ya que es compacto y fácil de programar, además que es mucho mas ligero que el modelo husky.</p>	
<p>https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/</p>	
<p>C://Robots móviles/UGV/Clearpath/Jackal</p>	

warthog

Datos




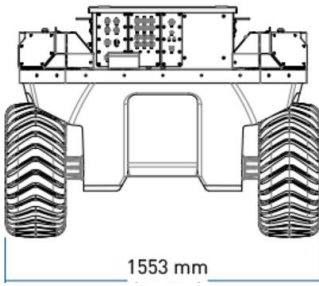
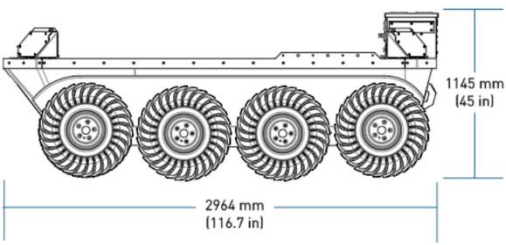
SIDE



FRONT

- Robot fabricado por Clearpath robotics con sede en Canadá.
- Máxima velocidad 18 km / h.
- Tiempo de ejecución (uso típico) 3 horas.
- Controladores y api ROS, Mathworks.
- Peso base (incluye batería) 280kg.
- Peso bruto del vehículo 590kg
- Máxima carga útil 272kg.
- Max inclinación 35 – 45°
- Dimensiones 1,52 x 1,38 x 0,83 m

	<ul style="list-style-type: none"> • Temperatura ambiente de funcionamiento -20 a 40°C • Tiempo de carga 4hr • Comunicación Ethernet, Usb, Control remoto, wi-fi. • Clasificación IP65 • Totalmente anfibio, 4km/h velocidad máxima en el agua.
<p>“Warthog es un gran vehículo terrestre no tripulado todo terreno capaz de viajar por tierra y por agua. Puede manejar ambientes difíciles con su construcción robusta, baja presión sobre el suelo y neumáticos de tracción, que permiten la movilidad sin esfuerzo a través de suelos blandos, vegetación, lodos gruesos y pendientes pronunciadas.” (Clearpath, 2015). Además, el vehículo solo está diseñado para ser capaz de flotar y no estar completamente sumergido, puede además superar terrenos blandos con gran facilidad.</p>	
<p>https://clearpathrobotics.com/warthog-unmanned-ground-vehicle-robot/</p>	
<p>C://Robots móviles/UGV/Clearpath/Jackal</p>	

Moose	Datos
  	<ul style="list-style-type: none"> • Robot fabricado por Clearpath robotics con sede en Canadá. • Máxima velocidad 30 km / h. • Tiempo de ejecución (uso típico) 6 horas. • Controladores y api ROS, Matlab api. • Peso base (incluye batería) 1077kg. • Peso bruto del vehículo 1590kg • Máxima carga útil 513kg. • Max inclinación 30– 35° • Dimensiones 2,96 x 1,5 x 1,14m • Temperatura ambiente de funcionamiento -10 a 50°C


	<ul style="list-style-type: none"> • Tiempo de carga 24hr • Comunicación Ethernet, Usb, Control remoto, wi-fi. • Clasificación IP67 debajo de lacubierta, IP65 sobre la cubierta. • Totalmente anfibio, 4km/h velocidad máxima en el agua.
<p>“Moose UGV es nuestro vehículo terrestre no tripulado todoterreno más grande hasta el momento. Puede manejar ambientes difíciles con su construcción robusta, baja presión sobre el suelo y neumáticos de tracción 8x8, que permiten una movilidad sin esfuerzo a través de suelos blandos, vegetación, lodos gruesos y pendientes pronunciadas.” (Clearpath, 2015), Una de las mayores particularidades de este robot es que posee la posibilidad de adaptar un gran número de accesorios en función de las labores que vaya a realizar dentro del ámbito militar o civil, ya sea de reconocimiento, exploración e investigación.</p>	
<p>https://clearpathrobotics.com/moose-ugv/</p>	
<p>C://Robots móviles/UGV/Clearpath/Moose</p>	

Iguana e	Datos
 	<ul style="list-style-type: none"> • Fabricado por la empresa Eca Group ubicada en Francia • Tiempo de ejecución 2h30 • Máxima velocidad 6km/h • Pendientes 30 – 45° • Temperatura ambiente de funcionamiento -20 a 50°C • Alcance de radio: hasta 800 m • Capacidad de elevación: 20 kg • Alcance vertical: hasta 2,8 m • Alcance horizontal: hasta 1,8 m • Carga útil: 40 kg • Protección IP65 • USB, ethernet • Cámara térmica
<p>Iguana E, vehículo terrestre no tripulado es el robot óptimo para respaldar las operaciones en entornos exigentes, espacios confinados y lugares difíciles de alcanzar. Con sus 40 kg de carga útil, incorpora una amplia gama de equipos para</p>	

proporcionar un robot único, fácil de implementar y eficiente. La unidad de control del operador fue diseñada especialmente para ser ultraligera y muy fácil de usar. (EcaGroup). El grupo eca ha estado diseñando, desarrollando, suministrando y apoyando sistemas robóticos en todo el mundo. Estos, así como simuladores de entrenamiento, sistemas operados a distancia y equipos especiales, apoyan los dominios de Seguridad Nacional, Fuerzas Especiales, Naval, Tierra y Fuerza Aérea. Este modelo de robot móvil es utilizado para el sector de defensa y seguridad.

<https://www.ecagroup.com/en/solutions/iguana-e-ugv-unmanned-ground-vehicle>

C://Robots móviles/UGV/EcaGroup/Iguana e

Aplicación de robot ugv iguana e	Datos
	<ul style="list-style-type: none"> • Fabricado por la empresa Eca Group ubicada en Francia • “En ECA GROUP, nos aseguramos de que nuestros clientes tengan el mejor equipo a su disposición y de que estén capacitados adecuadamente para alcanzar una alta competencia operativa.” (ecagroup, 2019)
<p>En la presente noticia se da a mostrar como este robot ayuda a las labores de los policías en Mónaco más específicamente para el escuadrón antibombas.</p>	
<p>https://www.ecagroup.com/en/solutions/cameleon-lg-e-ugv-unmanned-ground-vehicle</p>	
<p>C://Robots móviles/UGV/EcaGroup/Camaleon lg e</p>	

Camaleon Ige	Datos
	<ul style="list-style-type: none"> • Fabricado por la empresa Eca Group ubicada en Francia • Tiempo de ejecución entre 2h30 a 4h dependiendo de la misión • Máxima velocidad 5,1 km/h • Peso <20 kg • Pendiente 45° • Temperatura ambiente de funcionamiento -20 a 55°C • Alcance de radio: hasta 500 m • Alcance vertical de hasta 91 cm. • Alcance horizontal de hasta 91 cm. • Capacidad de remolque: 20 kg
<p>El camaleón Ige está diseñado para desplegarse rápidamente en el campo sin ralentizar ni obstaculizar los movimientos tácticos. Se puede llevar en una mochila además del equipo estándar de un soldado de infantería desplegado en las misiones OPEX (Operaciones en el extranjero). Está operativo en tan solo 3</p>	

minutos. El robot es de fácil entendimiento para el operador y aporta grandes ventajas en el campo.

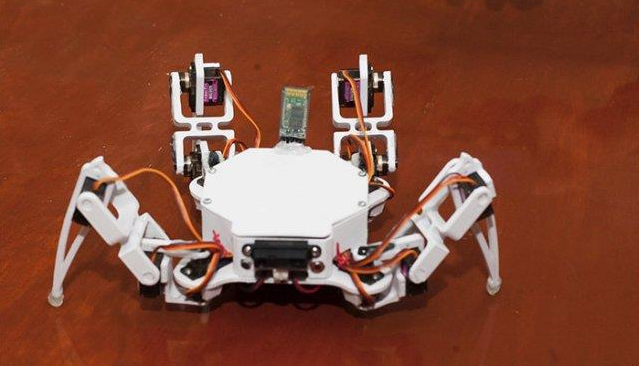
https://www.ecagroup.com/en/solutions/cameleon-lg-e-ugv-unmanned-ground-vehicle


C://Robots móviles/UGV/EcaGroup/Camaleon lg e

Aplicaciones de la robótica móvil

La utilización de robots industriales la encontramos en gran parte del planeta, otorgando beneficios tales como reducción de costes, aumento en la productividad, mejora en la calidad de producción y eliminación de condiciones peligrosas de trabajo o mejora de estas. Existen varias ramificaciones donde podemos encontrar un robot móvil terrestre ya sea en limpieza industrial, agricultura, transporte de mercancías, investigativos y educativos, domésticos y de oficina, entrenamiento, vigilancia y aplicaciones militares y búsqueda y rescate.

Para efectos de este proyecto nos enfocaremos en los robots exploradores, ya que son utilizados para conocer la situación del terreno. Las tareas básicas de este tipo de robots suelen ser el enviarlos a una determinada zona a comprobar si es seguro; y la comprobación puede ir desde mediante un sensor de calor o infrarrojos detectar que no exista ningún peligro.

Aplicación de búsqueda y rescate	Datos
	<ul style="list-style-type: none"> • Diseñado por estudiantes de México. • “Para auxiliar a las corporaciones de emergencia y auxilio durante la presencia de un desastre natural, estudiantes crearon el Robot Arácnido para Búsqueda y Rescate de Personas (RDB-10)” (La opinion, 2019)
<p>En la presente noticia se da a mostrar como este robot ayuda a las labores de búsqueda y rescate para localizar a personas atrapadas entre escombros.</p>	
<p>https://www.laopinion.com.co/tecnologia/crean-robot-para-busqueda-y-rescate-de-personas-171186#OP</p>	
<p>C://Robots móviles/UGV/RDB-10</p>	

Multiscope	Datos
	<ul style="list-style-type: none"> • Fabricado por la empresa Milrem Robotics con sede en Suecia y Etonia • Máxima velocidad 25 km / h • Longitud 240 cm • Anchura 200 cm • Altura 115 cm • Peso 1630 kg • Peso nominal de carga útil 750 kg • Peso de carga máxima 1200 kg • Pendiente lateral máxima 30% • Fuerza de tracción 21 000N • Tiempo de ejecución híbrido 12h a 15h • Tiempo de ejecución eléctrica 0,5h a 1,5h • Opciones de poder Motor Diesel y generador eléctrico • Rango de control de línea de vista hasta 1,5 km • Sensores LiDAR • Cámaras terminas y HDR

	<ul style="list-style-type: none"> • Este robot está equipado con mangueras de emergencias que ayudan a bomberos en distintos países europeos.
	<ul style="list-style-type: none"> • Este robot está equipado con un monitor de agua modular con un caudal de 3000 LPM. • Su diseño es especial para incendios forestales, bodegas y túneles.
<p>En estos artículos se muestran los diferentes robots de fuego y rescate que en su mayor parte ayudan al equipo de bomberos, sus diseños fueron hechos para ser duraderos y resistir altas temperaturas. La plataforma modular puede soportar condiciones difíciles y puede llegar a áreas que son de difícil acceso para vehículos más grandes. Las herramientas del multiscope pueden ser modificadas en el mismo sitio de trabajo.</p>	
<p>https://milremrobotics.com/product/pegasus/</p>	
<p>C://Robots móviles/UGV/Milrem Robotics/Pegasus: Multiscope</p>	

TheMIS UGV	Datos
	<ul style="list-style-type: none"> • Fabricado por la empresa Milrem Robotics con sede en Suecia y Etonia • Máxima velocidad 25 km / h • Longitud 240 cm • Anchura 200 cm • Altura 115 cm • Peso 1630 kg • Peso nominal de carga útil 750 kg • Peso de carga máxima 1200 kg • Pendiente lateral máxima 30% • Fuerza de tracción 21 000N • Tiempo de ejecución híbrido 12h a 15h • Tiempo de ejecución eléctrica 0,5h a 1,5h • Opciones de poder Motor Diesel y generador eléctrico • Rango de control de línea de vista hasta 1,5 km • Sensores LiDAR • Cámaras termicas y HDR

	<ul style="list-style-type: none"> • Peso máximo 350kg excluyendo arma y municiones • Altura máxima 650 mm excluyendo arma • Angulo de elevación -20 a 60° • Giro 360° • Localizador laser, cámara termínica
	<ul style="list-style-type: none"> • Peso máximo excluyendo el arma y municiones 172kg • Altura máxima 762mm • Temperatura de operación -46 a 65°C
	<ul style="list-style-type: none"> • Angulo de elevación -40 a 70° • Altura 630 mm • Con arma m3 y 300, municiones 12.7mm
	<ul style="list-style-type: none"> • Con equipamiento para detección y eliminación de explosivos



- Tamaño plataforma 1,18 x 1,18 x 0,59 m
- Peso 11kg
- Drone con cámara 720p zoom digital x8
- Transmisión de video en vivo, reporte de altura.

El Sistema de infantería modular híbrido rastreado, o THeMIS para abreviar, es el primer vehículo híbrido terrestre no modular completamente tripulado del mundo. El vehículo está destinado a brindar apoyo a las tropas desmontadas sirviendo como plataforma de transporte, estación de armas remota, unidad de detección y eliminación de IED y mucho más. Además de este equipo se encuentran distintas variaciones dependiendo la necesidad del usuario.

“El pelotón de infantería Estpla-32 que actualmente sirve en Malí desplegó por primera vez el vehículo terrestre no tripulado (UGV) Milrem Robotics THeMIS durante una operación militar, probando su implementación en apoyo de la infantería en el área de conflicto desde un punto de vista táctico y técnico. (milrem robotics, 2019)

<https://milremrobotics.com/themis/>

C://Robots móviles/UGV/Milrem Robotics/TheMis UGV

Control de movimiento y evasión de obstáculos

Como ya se ha explicado, la mayor desventaja de la planificación global es que asume un conocimiento perfecto del medio que lo rodea. Sin embargo, es muy común que un robot deba moverse por un entorno parcial o totalmente desconocido. Por ello, debe estar preparado para enfrentarse a obstáculos no tenidos en cuenta por la trayectoria global. De esta necesidad surge la creación de algoritmos de planificación de trayectoria local, que permita al robot móvil reaccionar ante obstáculos inesperados.

Obstáculos estáticos.

El problema más básico de la robótica móvil con respecto a la evasión de obstáculos es de capacitar al robot para esquivar objetos cuya posición y velocidad son constantes. Existen varios algoritmos que solucionan este problema, para efecto de este proyecto se analizaran y mostraran los algoritmos bubble rebound y divide and conquer.

Bug algorithm.

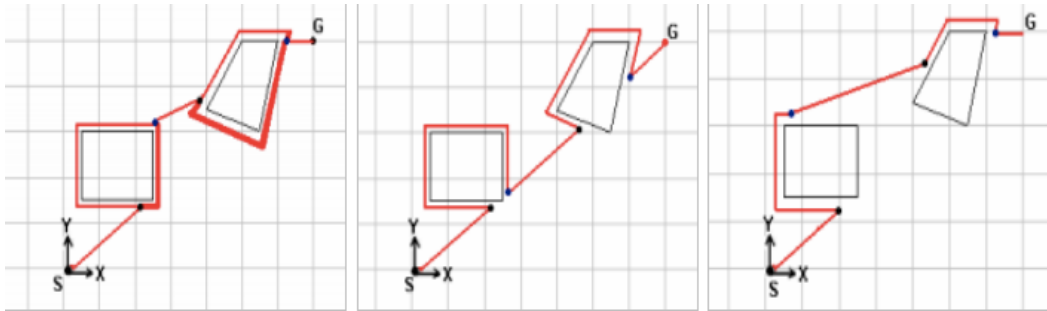
Este método es uno de los primeros en ser desarrollado para la evasión de obstáculos. Existen tres versiones diferentes según la manera de afrontar el obstáculo.

Método bug-1. Cuando el robot detecta que un objeto obstruye su camino, procede a rodearlo completamente buscando el punto de menor distancia al objetivo final. El robot calcula una nueva trayectoria a partir de dicho punto, al que llega rodeando una segunda vez al obstáculo.

Método bug-2. En lugar de rodear completamente al obstáculo, el robot lo hace hasta que encuentra un punto en el que la pendiente de la recta que lo une con su destino es igual a la que tenía en el momento en que detectó al obstáculo.

Método dist-bug. Al igual que en los casos anteriores, cuando el robot encuentra un obstáculo, procede a rodear el borde. Durante este movimiento, calcula continuamente su distancia con el punto final, eligiendo punto de desprendimiento aquel en el que la distancia sea menor que la que tenía cuando detectó el obstáculo.

A continuación, se muestra cómo se realiza la evasión de obstáculo para los tres métodos en orden de lectura.

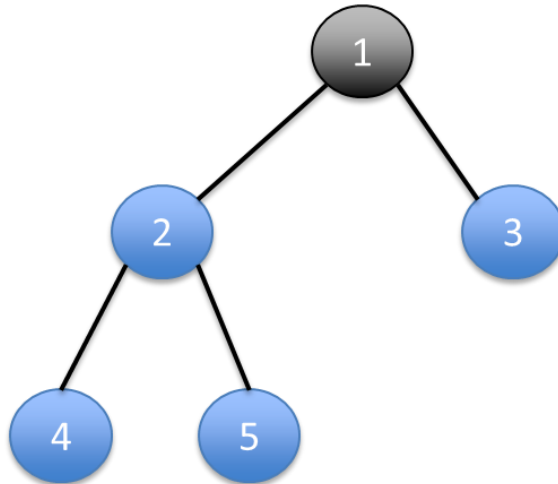


Algoritmo de búsqueda en profundidad (DFS)

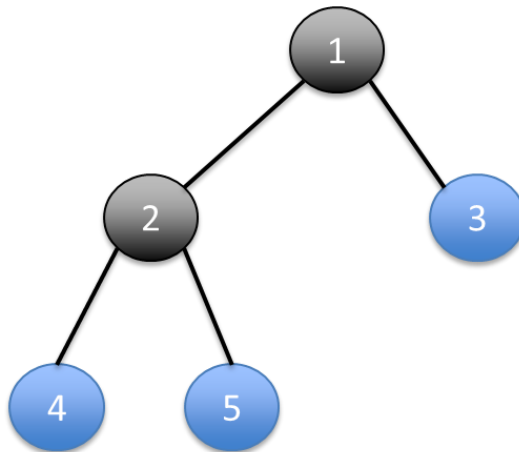
Este algoritmo de grafos BFS es una forma sistemática de encontrar todos los vértices alcanzables de un grafo desde un vértice de origen. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa, de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado. La forma de ejecución de este algoritmo es la siguiente: se selecciona cualquier nodo como punto de partida (por lo general el primer nodo del grafo) y se marcan todos los nodos del grafo como “no visitados”. El nodo inicial se marca como “visitado” y si hay un nodo adyacente a este que no haya sido “visitado”, se toma este nodo como nuevo punto de partida del recorrido. El recorrido culmina cuando todos los nodos hayan sido visitados. Se dice que el recorrido es en profundidad, porque para visitar otro nodo adyacente del nodo inicial, primero se deben visitar todos los nodos adyacentes al que se eligió antes.

A continuación se da un ejemplo expletivo de cómo funciona en un algoritmo de búsqueda en profundidad

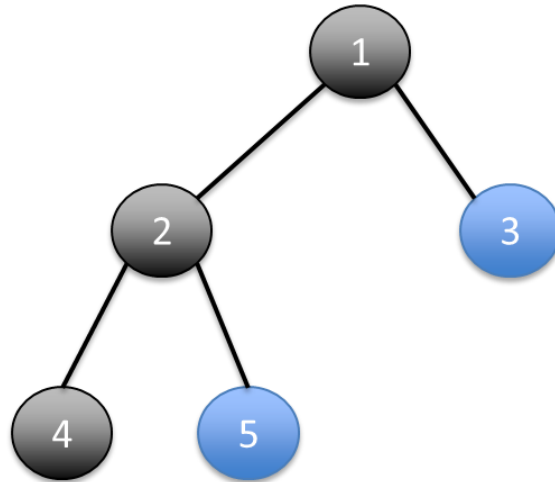
Como punto inicial tomamos el nodo número “1” y lo marcamos como visitado.



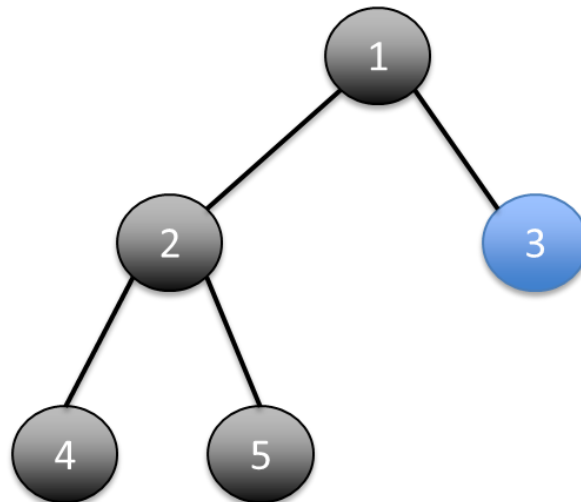
Su adyacente sería el nodo número “2”, el algoritmo se pregunta si este ya ha sido visitado. Para este caso el nodo numero 2 no ha sido visito por lo tanto es el siguiente en visitar y lo marcamos como visitado.



Luego el adyacente para el nodo número “2” es el nodo número “4”, se pregunta si ya fue visitado. Para este caso aún no ha sido visitado por lo cual se avanza al nodo número 4 y se marca como visitado

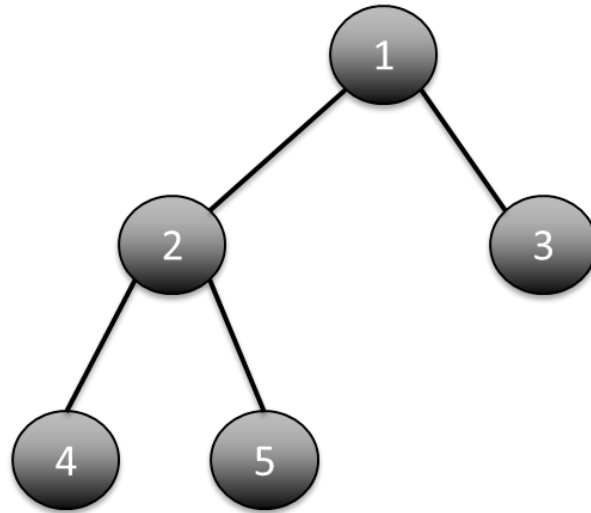


Ahora el nodo “4” es el inicial y se vuelve a preguntar, como su único adyacente en este caso es el nodo número “2”, por lo tanto tocaría el nodo número “2” pero como ya fue visitado la recursión termina por ese lado y el algoritmo retroceso y pregunta para el caso del nodo número “2”, para este caso el siguiente adyacente es el nodo número “5”, por lo tanto se marca como visitado y se continua con la búsqueda profunda.



Continuando con lo búsqueda, pasa lo mismo con el nodo número “4”, por lo tanto, el algoritmo se devuelve hasta el nodo número “1” ya que por nodo número “2” todos sus adyacentes han sido visitado. El nodo analizar ahora es el numero “1” y sus adyacentes son los nodos número “2” y “3”, como en nodo número “2” ya fue

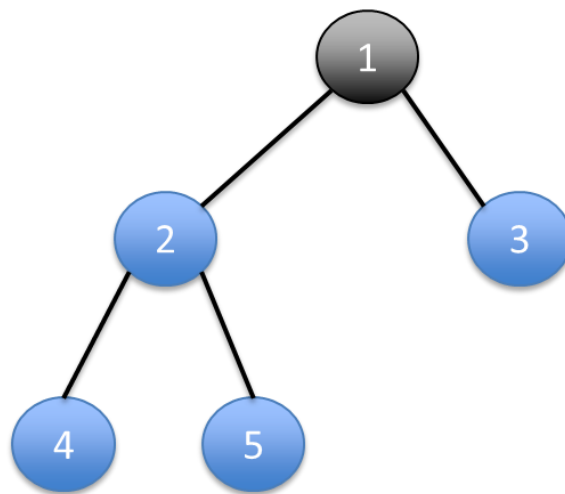
visitado, se da paso al nodo número “3” se lo marca como visitado y se da por finalizada la búsqueda en profundidad o también conocida como BFS en sus siglas en ingles



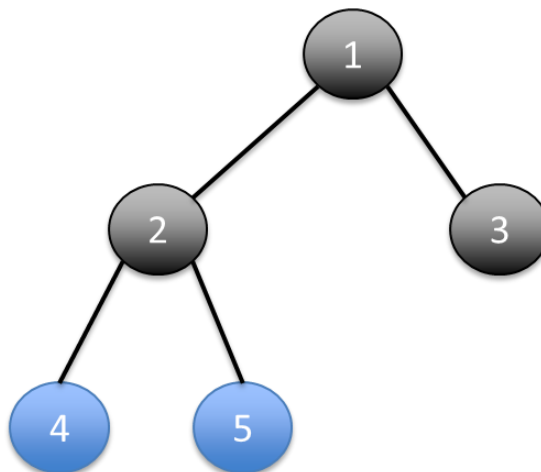
Algoritmo de búsqueda en anchura (BFS)

En este algoritmo también se utiliza la estrategia de marcas los nodos como “visitados” para detectar la culminación del recorrido, pero los nodos se recorren de una manera ligeramente distinta. De nuevo, se selecciona cualquier nodo como punto de partida (por lo general el primer nodo del grafo) y se marcan todos los nodos del grafo como “no visitados”. El nodo inicial se marca como “visitado” y luego se visitan TODOS los nodos adyacentes a este, al finalizar este proceso se busca visitar nodos más lejanos visitando los nodos adyacentes a los nodos adyacentes del nodo inicial. Este algoritmo de grafos es muy útil en diversos problemas de programación. Por ejemplo, halla la ruta más corta entre dos vértices cuando el peso entre todos los nodos es 1, cuando se requiere llegar con un movimiento de caballo de un punto a otro con el menor número de pasos, o para salir de un laberinto con el menor número de pasos, etc.

A continuación, se realiza el mismo ejemplo anterior pero ahora para el algoritmo de búsqueda en anchura a diferencia del algoritmo anterior este hace una búsqueda por niveles, de esta forma se garantiza que el algoritmo no pasara dos veces por el mismo nodo. Lo importante este tipo de búsqueda es que se hace de manera horizontal.

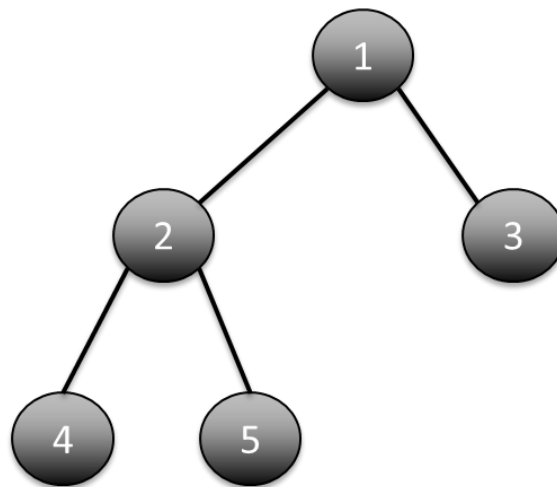


Marcamos nuestro nodo inicial, para este caso es el número "1", los adyacentes para el nodo número "1" son, son el nodo número "2" y "3", los cuales se marcan como visitados y se pasa al siguiente nivel de búsqueda.



En el siguiente nivel de búsqueda el nodo analizar es el "2", el cual sus nodos adyacentes son el 4 y "5", los marcamos como visitados y analizamos el nodo siguiente nodo que está al nivel del nodo número "2", que para este caso es el nodo

número “3”, como el nodo número “3” no tienen adyacentes se termina la búsqueda por esa parte y se da la búsqueda al siguiente nivel. Los nodos analizar son ahora el nodo número “4” primero y al no tener adyacentes se da la búsqueda finalizada para el nodo número “4” y pasamos al siguiente nodo que está a la misma escala del nodo número “4” que es el “5” para este caso y al no tener más nodos adyacentes se da por finalizada la búsqueda por anchura.



Las grandes diferencias que se pueden ver en ellos son los métodos de búsquedas que realizan uno realiza búsquedas de formas horizontal mientras que el otro hace búsquedas en forma vertical.

Divide & conquer (divide y vencerás)

Este es un algoritmo que prepara al robot para esquivar obstáculos en entornos densos o complejos. La idea base es la de analizar la configuración de los obstáculos, el robot y su destino, clasificándola en uno de sus seis casos que describen la situación de manera completa y exclusiva. Una vez clasificada la situación, el robot deberá realizar la acción correspondiente a su caso.

Bubble rebound algorithm

Este método establece una frontera de protección alrededor del robot mediante una burbuja, cuyas dimensiones dependen de la velocidad del robot y del intervalo de tiempo entre lecturas de láser. Cuando un obstáculo penetra en la frontera de la burbuja, el robot calcula el llamado ángulo de rebote, en la dirección en la que haya menos obstáculos. Toma dicha dirección hasta que encuentra un objetivo intermedio, a partir del cual puede continuar siguiendo la trayectoria global.

Algoritmo A*

Se encuentra entre los algoritmos de búsqueda en grafos. Su función es encontrar siempre y cuando se cumplan determinadas condiciones, el camino de menor costo entre un nodo origen y uno objetivo, es la forma más ampliamente conocida de la búsqueda primero el mejor, siendo la búsqueda A* tanto completa como óptima. A diferencia de los otros algoritmos como lo son DFS Y BFS, este algoritmo puede y permite al robot moverse en diagonal provocando con ello una mejor trayectoria.

Obstáculos dinámicos.

Actualmente, el mayor desafío de la robótica móvil es la navegación autónoma en entornos que contienen obstáculos en movimientos. Sobre todo, cuando su velocidad y dirección son impredecibles. Los métodos de evasión de obstáculos móviles deben dotar a los robots con la capacidad de predecir movimientos futuros de obstáculos.

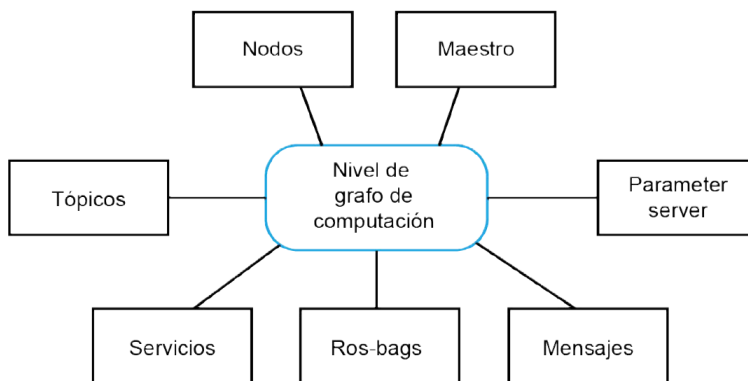
Aunque en este proyecto de título no se considere la evasión de obstáculos móviles, este es un tema de gran relevación y que hoy en la actualidad se centran muchos trabajos de investigación, un ejemplo de ello son los autos tesla.

Robot operating system (ROS)

Ros es actualmente una de las plataformas de desarrollo de aplicaciones robóticas más usadas en el mundo. Esto debido a que ofrece un software de código abierto hace posible que expertos en las diferentes áreas de la robótica móvil puedan colaborar en proyectos. Como su nombre lo indica proporciona servicios propios de un sistema operativo, como pueden ser la transmisión de mensajes entre procesos y la administración de paquetes, también proporciona herramientas y bibliotecas para obtener, construir, escribir y ejecutar código en C++ y Python.

Grafo de computación

El nivel conocido como grafo de computación contiene la red de procesos de ROS que gestiona información en conjunto. A continuación, se muestra la estructuración de nivel de grafo de computación y una pequeña descripción de sus componentes.



Maestro

El maestro se encarga de la gestión de mensajes y servicios establecidos entre nodos. Es el cerebro de la computación gráfica.

Nodos

Un nodo es un su base un archivo ejecutable dentro de un paquete de ros. La comunicación entre nodos se establece mediante la publicación o suscripción a tópicos, también es posible que un nodo publique y se suscriba a información al mismo tiempo.

Mensajes

Los nodos se comunican entre sí mediante mensajes. Para que un nodo publicador y otro suscriptor puedan compartir información entre sí, es necesario que lo hagan empleando un mismo tipo de mensaje. Además de datos tipo int, double, etc.

Tópicos

Si los mensajes son la información que se quiere comunicar, lo tópicos son el canal por el cual se establece dicha información, cada tópico es capaz de comunicar un tipo de dato específico, por lo que es necesario que los mensajes transmitidos o recibidos a través de él sean también de este tipo.

Servicios

A pesar de las ventajas del sistema de comunicación a través de tópicos, estos no son apropiados para interacción tipo pregunta-respuesta, la cual es necesaria para la comunicación entre sistemas. Esto se resuelve con el uso de servicios. A diferencia de los tópicos un servicio puede contener dos tipos de mensajes diferentes: uno para la pregunta y otro para la respuesta. En este caso un nodo tendrá el papel de proveedor, ofreciendo un servicio, y otro tendrá el papel de cliente, enviado una petición de servicio y atendiendo una respuesta.

Parameter server

Parameter server o servidor de parámetros en español, es una base de datos accesible a través de una red API. Proporciona a los nodos un lugar para almacenar o recuperar parámetros en su momento de ejecución.

Ros-bags

Son un formato para guardar y reproducir el contenido de los mensajes de ros. Resultan de gran utilidad a la hora de probar y depurar algoritmos con los mismos datos.

Sistemas de archivos.

El nivel del sistema de archivos engloba la estructuración interna de ROS y los ficheros mínimos necesarios para su funcionamiento. Como en otros sistemas operativos, un programa de ros está estructurada en carpetas, ordenados de forma jerárquica.

Comunidad

El nivel de la comunidad contiene los recursos ofrecidos por ros para la interacción entre desarrolladores de software robótico, facilitando la distribución de código y conocimiento. Esta facilidad es una de las ventajas más grandes de ros y siendo el principal motivo de su popularidad. Ros ha creado los llamados repositorios en los cuales cualquiera puede crear un repositorio, subir código y añadir documentación y licencias. Además, ros hace disponibles plataformas como ROS Wiki, que contiene toda la información incluyendo tutoriales sobre instalaciones, uso de programas. para finalizar cuenta con un apartado para preguntas en donde varios usuarios ayudan a resolver problemas para otros usuarios.

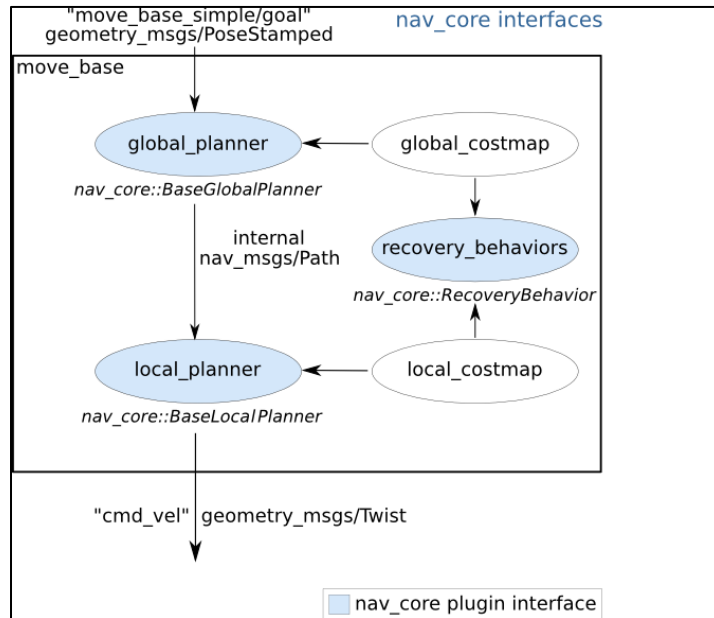
Desarrollo

Como se ha mencionado, el trabajo de este proyecto a continuación tiene como objetivo la creación de aplicaciones para robots móviles que permitan que estos sean capaces de moverse en su entorno evadiendo obstáculos imprevistos. Para poder trabajar con el simulador de gazebo es necesario tener un sistema operativo compatible con dicho simulador, por lo cual, es necesario crear una extensión en el disco duro del computador para poder ejecutar la simulación del proyecto. Para llevar a cabo con éxito el arranque de la simulación se requiere descargar el sistema operativo y bootear nuestra unidad usb.

Para la integración en el robot del código de desarrollo se ha hecho uso del paquete pluginlib de ROS, que proporciona herramientas para escribir y desarrollar plugins, o clases dinámicas cargables, que permiten extender o modificar el comportamiento de una aplicación sin la necesidad de tener el código fuente de la aplicación.

En concreto se desarrollaron plugins para el paquete de navegación move_base, con el objetivo de reemplazar el plugin de planificación de trayectoria local usado por defecto, el base_local_planner.

Cuando se crea un plugin de planificación local para el paquete de navegación, es necesario adherirse a la clase `nav::core::BaseLocalPlanner`, perteneciente al paquete de navegación `nav_core`. Este paquete proporciona interfaces comunes para acciones de robot específicas de la navegación, en este paquete encontraremos las interfaces `BaseGlobalPlanner`, `BaseLocalPlanner` y `RecoveryBehavior`, que permiten modificar fácilmente el planificador global, el planificador local o el comportamiento de recuperación. A continuación, se muestra una imagen de los paquetes antes mencionados.



Las funciones principales de la clase `nav_core::BaseLocalPlanner` que deben usarse son : `computeVelocityCommands()`, que calcula los comandos de velocidad para enviarlos a la base, `initialize()`, para la ejecución de variables, `isGoalReached()`, que devuelve “1” cuando se ha alcanzado el final de la trayectoria global, y `setPlan()`, que ejecuta el plan.

Para poder desarrollar algoritmos implementando técnicas de evasión de obstáculos, se ha decidido comenzar con un código de seguimiento de trayectoria. El plugin `simple_path_follower` tiene por objetivo conducir al robot por diferentes puntos de la trayectoria global, hasta llegar al destino.

El plugin comienza recibiendo la trayectoria global a través de la función `setPlan()`, en la que también se guarda la longitud del plan en la variable `length` y se inicia la variable `next`, que contendrá en todo momento la posición del siguiente punto de la trayectoria al que debe dirigirse el robot. Para simplificar el almacenaje de variables de posición y velocidad, se ha creado una estructura llamada `pos`, compuesta por tres variables tipo `double`: `x`, `y`, `z`, que representan la posición o velocidad lineal en

z e y, y para la orientación o velocidad angular en z, respectivamente. La variable next es de tipo pos.

Algoritmo bubble rebound

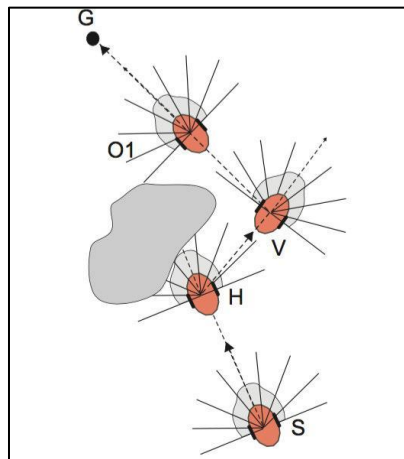
Este algoritmo consigue la evasión de obstáculos conduciendo al robot a una zona de menor densidad.

Los obstáculos se detectan mediante una frontera de protección alrededor del robot llamada burbuja, cuyas dimensiones dependen de la velocidad del robot y del intervalo de tiempo entre lecturas de láser. Cuando un obstáculo penetra en la frontera de la burbuja, el robot calcula el llamado ángulo de rebote, en la dirección en la que haya menor densidad de obstáculos. El robot toma dicha dirección hasta que encuentra un objetivo intermedio de la trayectoria global. A partir de este momento se dirige hasta dicho punto para continuar el seguimiento de la trayectoria original.

El método de cálculo del Angulo de rebote es mediante una media ponderada, en la que el ángulo es el dato y la distancia leída por el láser. A continuación, se muestra la ecuación de cálculo.

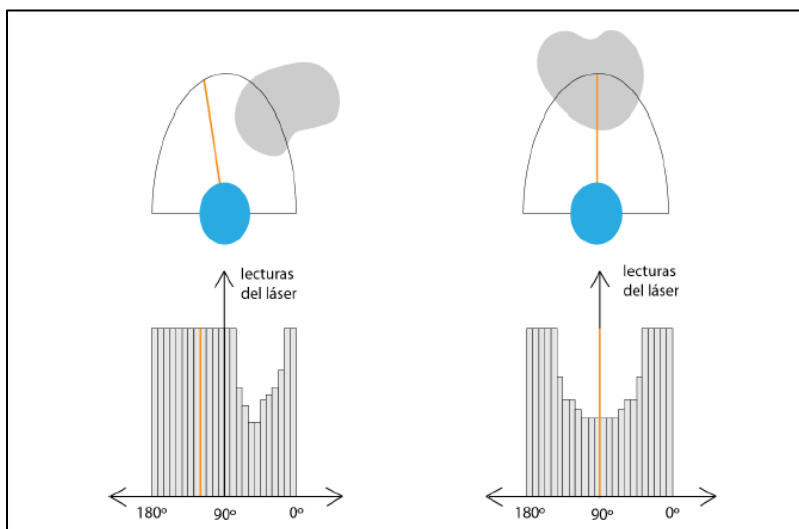
$$\alpha_r = \frac{\sum_0 180 \alpha_i * D_i}{\sum_0 180 D_i}$$

Don D_i es la lectura del láser correspondiente al ángulo α_i .



En la imagen se representa al algoritmo y su funcionamiento para la evasión de obstáculos.

Uno de los mayores inconvenientes del algoritmo resulta que cuando hay un obstáculo bloqueando el camino y este tiene una distribución bastante uniforme y centrada, en este caso la media ponderada dará como resultado un Angulo de rebote que conducirá al robot directo al obstáculo. A continuación, en la siguiente imagen se representa dicha problemática.

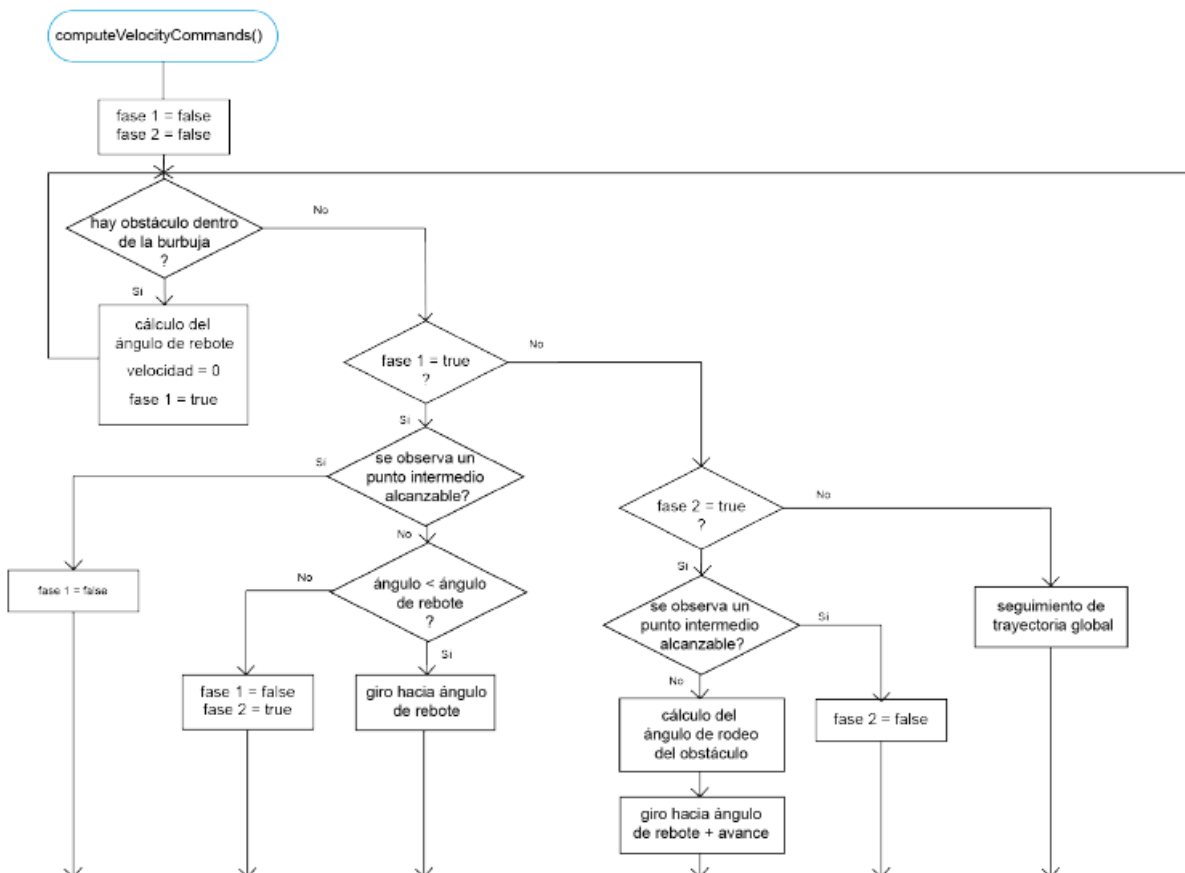


La otra problemática que resulta de este algoritmo es que no se encuentra preparado para superar obstáculo en forma de U. Esto ocurre porque el ángulo de rebote está pensado para calcularse solo una vez, lo cual, en el caso de un sensor de 180 grados, imposibilitaría la evasión de obstáculos en U ya que estos bloquean todas las direcciones posibles de rebote. Por lo cual, es importante que el robot entre en un bucle continuo de giro y cálculo de ángulo, hasta encontrar una salida del obstáculo. El Angulo de rebote definitivo conducirá al robot en dirección completamente contraria a la que lleva hasta el momento, ya que salir por donde ha entrado es el único camino aceptable. Una vez tomada esta nueva dirección de avance, es evidente que el robot jamás encontrara los puntos de trayectoria original, este es otro problema que se debe solucionar. Estos inconvenientes se solucionan modificando el ángulo de rebote que, aunque comienza buscando el camino de

menor densidad, continua con un ángulo que se modifica a medida que avanza el robot y que pretende seguir un camino rodeando y esquivando al obstáculo al que se encuentre en su trayectoria.

Al igual que en el `simple_path_follower`, el plugin comienza recibiendo la trayectoria global a través de la función `setPlan()`, almacenándose también la longitud del plan en la variable `length` e inicializándose la variable `next`.

La función `computeVelocityCommands()` contiene de nuevo el algoritmo principal. Éste incorpora el código de seguimiento de trayectoria ya desarrollado, que es el algoritmo que el robot sigue hasta que encuentra un obstáculo, además del código en el que se implementa el “bubble rebound algorithm”.

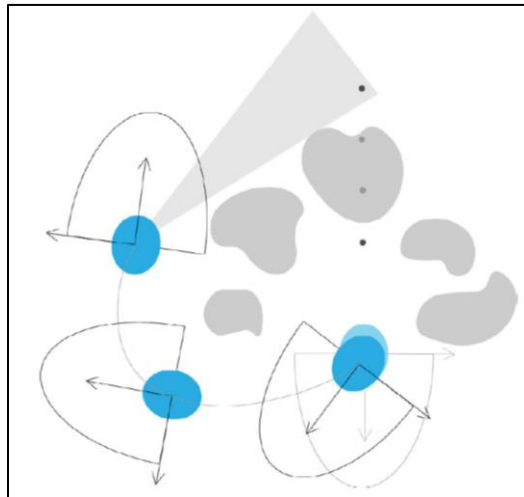


Como se ha mencionado previamente, se considera que el robot ha detectado un obstáculo en el momento en que éste último penetra en su burbuja de sensibilidad. En este instante, el robot pasará a seguir el algoritmo de evasión de obstáculos, constituida por 2 fases principales de movimiento: la de computación y el seguimiento del ángulo de rebote que busca la zona de menor densidad de obstáculos, y la de computación y el seguimiento del ángulo de rebote que busca rodear al obstáculo. Con esto, el algoritmo se convierte en un híbrido entre el algoritmo original y el algoritmo bug 2.

La manera en que el robot pasa por las sucesivas fases es la siguiente. Si el robot detecta un obstáculo, pasa a la fase 1, en la que se calcula una vez el ángulo de rebote que corresponde a la zona de menor densidad. Una vez el robot ha girado hacia la dirección calculada, el algoritmo pasa a la fase 2. En el caso de que el robot, al intentar avanzar de nuevo, se encontrara con otro obstáculo, volvería a la fase 1 para calcular un nuevo ángulo de rebote.

Una vez que el robot se encuentra en la fase 2, y que no hay obstáculos en el camino, el ángulo de interés pasa a ser aquel que permita al robot rodear al obstáculo con la mayor proximidad posible. Con esta fase se garantiza que el robot siempre encuentre un objetivo intermedio, resolviendo uno de los problemas anteriormente mencionados. Además, con ello se consigue solucionar también el otro problema ya que, usando el mismo ejemplo del obstáculo en U, y tras el giro de 180° debido al ángulo de rebote, el robot dejaría de buscar el camino de menor densidad, pasando a un movimiento alrededor del obstáculo.

Una vez que el robot ha rodeado casi completamente al obstáculo, es evidente que éste será capaz de observar con el láser alguno de los puntos de la trayectoria original que previamente se encontraban al otro lado del obstáculo. A continuación, se muestra una imagen de lo mencionado anteriormente.



Cálculo del ángulo de rebote

En el cálculo de los ángulos de rebote, es imprescindible tener en cuenta la conversión de ángulos de un sistema de referencia a otro. En el caso del código desarrollado, el ángulo de rebote se calcula en el sistema de referencia del láser, que no es el mismo que el del robot.

Para poder calcular la velocidad angular del robot, el ángulo de rebote siempre deberá ser pasado al sistema de referencias global. Esta conversión se hace del siguiente modo:

$$yaw_g = yaw_l - \beta + now.az$$

donde yaw_g es el ángulo en el sistema de coordenadas global, yaw_l es el ángulo en el sistema del láser, β son los grados para pasar del sistema de referencia del láser al del robot, y $now.az$ es la orientación actual del robot en el sistema de coordenadas global.

Como ya se ha explicado anteriormente, existen 2 fases por las que pasa el robot en la evasión de obstáculos, cada una con su propio método de cálculo del ángulo de rebote. Por lo tanto, se han desarrollado dos funciones diferentes,

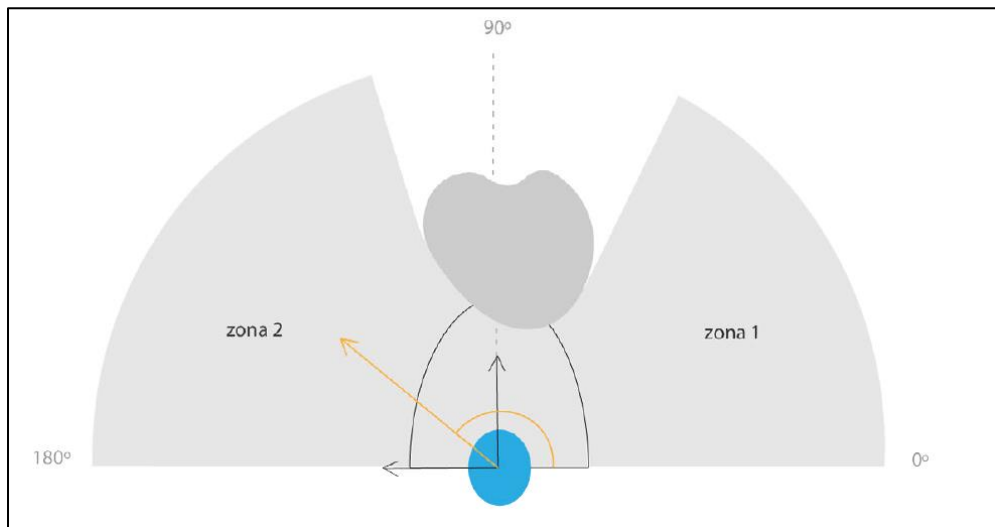
computeReboundAngle() y computeReboundAngle2(), una para cada una de las fases. A continuación, se explica en más detalle el funcionamiento de cada una:

computeReboundAngle()

La primera fase ha sido especialmente diseñada para la evasión de obstáculos en U o similares, en la cual el robot tendrá que calcular más de un ángulo de rebote antes de encontrar un camino seguro.

Como se abordó anteriormente el robot al encontrarse con un objeto uniforme y con un láser de 180° provocara que el robot colisione con el objeto. El problema anterior se resuelve si la visión del láser se divide en 2 zonas, cada una de 90 grados de amplitud. Las dos zonas se evalúan, comprobando en cuál de ellas la media de las lecturas de láser es mayor, es decir, cuál contiene, menor, densidad de obstáculos. Una vez elegida la zona más segura, se calcula el ángulo de rebote para esa zona, tal y como se hacía en el algoritmo original.

Si el robot intenta avanzar en la dirección escogida, pero se encuentra con un obstáculo, se volverá a calcular un ángulo de rebote. Si no bloquea el paso ningún obstáculo, el robot podrá pasar a la fase 2.

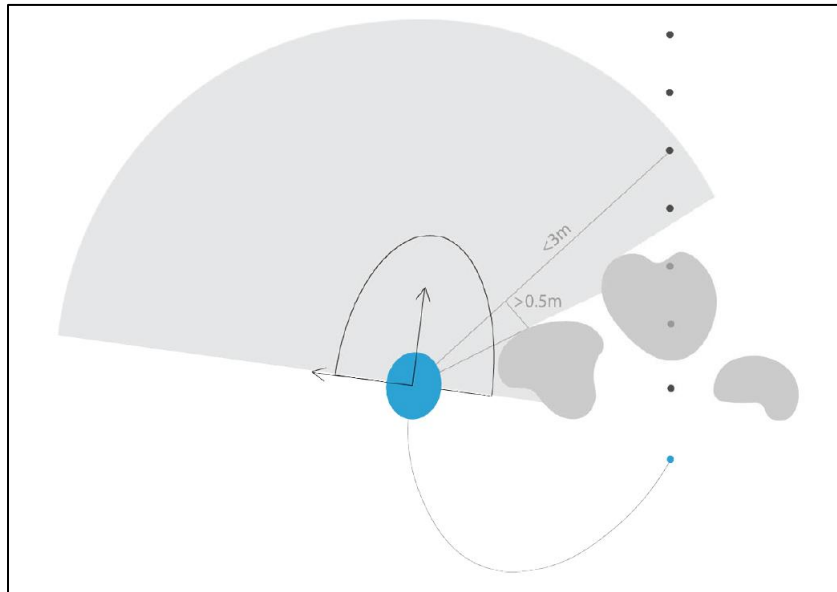


computeReboundAngle2()

La fase 2 tiene como objetivo conducir al robot alrededor del obstáculo, manteniendo una distancia de seguridad, hasta encontrar de nuevo la trayectoria global. Durante todo el movimiento, el robot puede encontrarse en dos situaciones: que esté demasiado cerca del obstáculo, o que se encuentre a una distancia segura.

Luego de haber rodeado al objeto se busca un punto intermedio para continuar con la trayectoria. Por ello es necesario que, a lo largo de la trayectoria local, sea el robot quien busque un punto de la trayectoria global a partir del cual continuar su movimiento normal. La función findIntermGoal() es la encargada de la búsqueda de dicho punto.

Cuando exista un punto de la trayectoria global a una distancia del robot menor de una predefinida, en cuya posición el láser no encuentre ningún obstáculo, y en cuya dirección el robot sea capaz de navegar sin chocarse con el obstáculo más cercano, se elegirá dicho punto como objetivo intermedio, y el robot se dirigirá a él. Una vez alcanzado el punto, el robot reanudará el seguimiento de la trayectoria global. A continuación, se muestra una imagen representativa de como el robot encuentra la trayectoria para llegar al destino final.

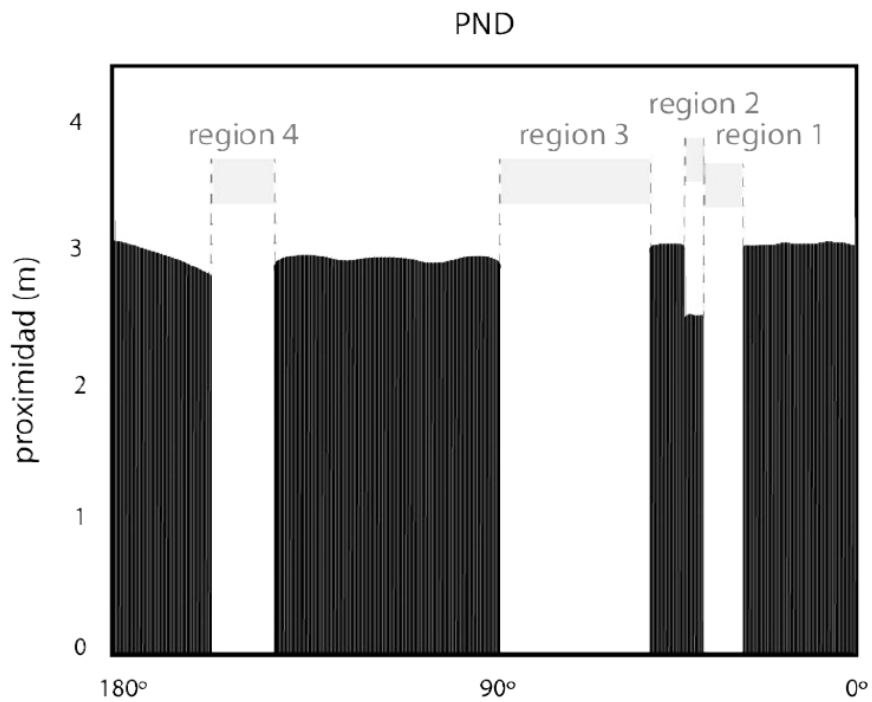
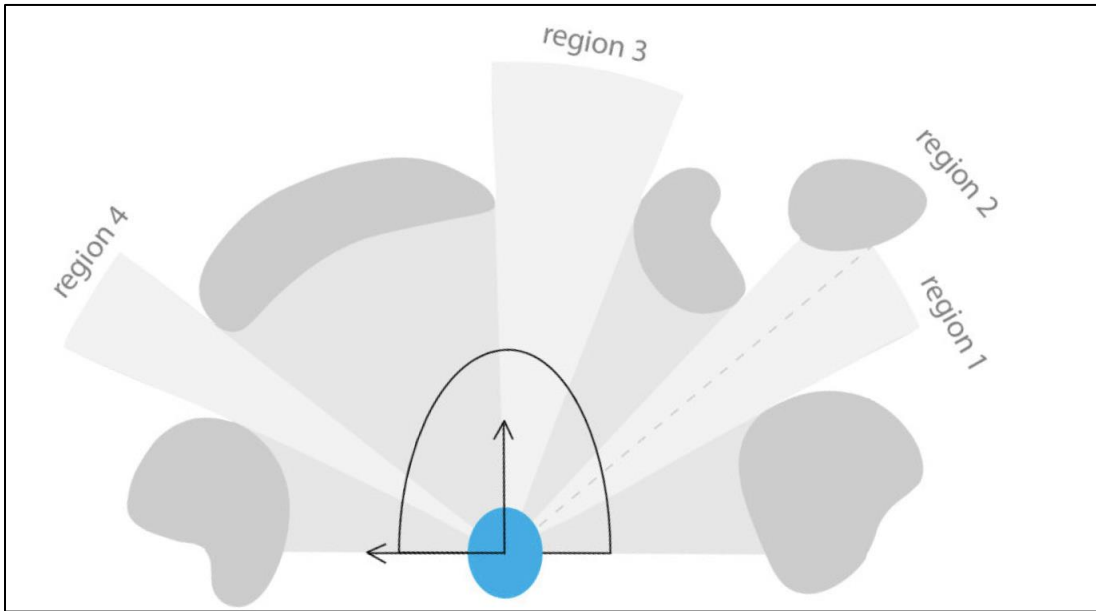


Divide and conquer

Para la comprensión de este algoritmo de evasión, se puede dividir su explicación en tres partes: herramientas, situaciones y acciones. La idea base es la de usar unas herramientas para analizar la configuración de los obstáculos, el robot y su destino, clasificándola en uno de seis casos que describen la situación de manera completa y exclusiva. Una vez clasificada la situación, el robot deberá realizar la acción asociada.

Herramientas.

Para analizar la relación entre el robot, los obstáculos y el objetivo intermedio, existe una herramienta llamada Nearness Diagram, o ND. En concreto, se utiliza el diagrama PND, que establece la cercanía de los obstáculos al punto central del robot. A continuación, se muestra una representación de un posible escenario para el robot y su respectivo diagrama PND.



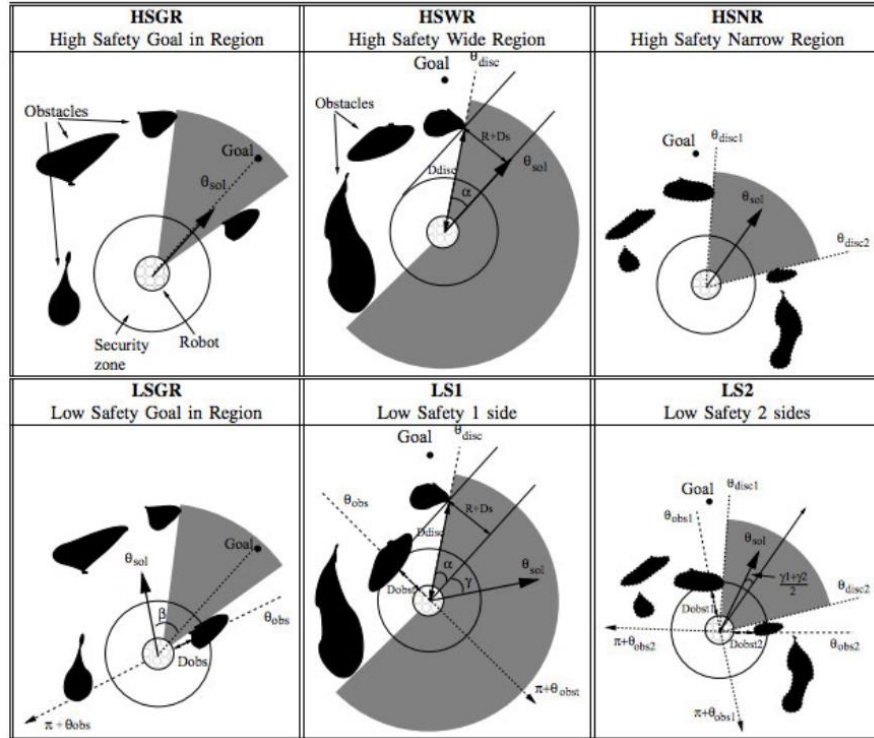
El PND funcionará como una representación del entorno, permitiendo su posterior análisis. Las zonas vacías del diagrama representan las llamadas regiones, delimitadas cada una por dos discontinuidades, o huecos. Una de las regiones será

elegida como la de libre circulación, y por ella navegará el robot hacia el objetivo. La región elegida será aquella más cercana al objetivo, que cumpla la definición de navegable, explicada más adelante.

Situaciones

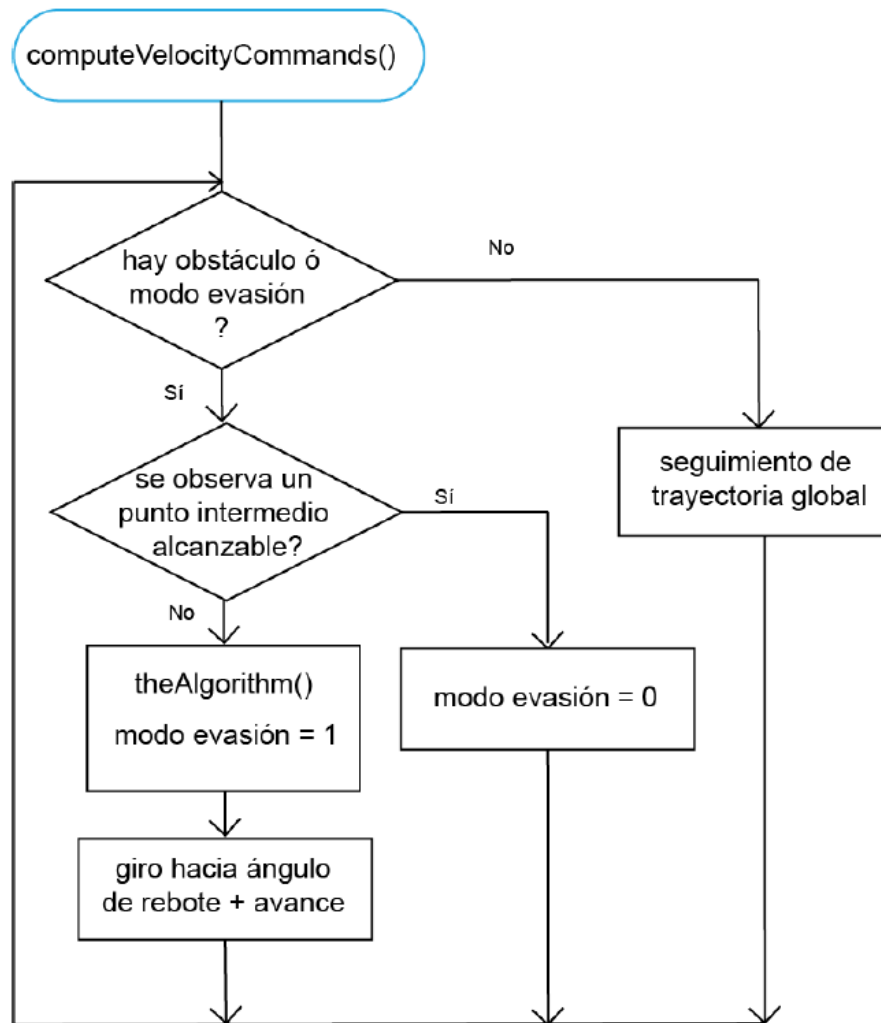
El primer criterio de clasificación es la seguridad, separando las situaciones en seguras y no seguras. Una situación se considera segura cuando no hay obstáculos dentro de la zona de seguridad y, del mismo modo, una situación no es segura si algún obstáculo penetra en la zona de seguridad. Dentro de una situación segura, existen dos criterios de clasificación, la primera siendo la anchura de la región. Si la anchura es superior a una predeterminada, la situación se llama High Safety Wide Region (HSWR). En el caso de ser inferior, recibe el nombre de High Safety Narrow Region (HSNR). Tras este análisis, si detecta que el objetivo se encuentra dentro del área de libre circulación, la situación recibe el nombre de High Safety Goal in Region (HSGR).

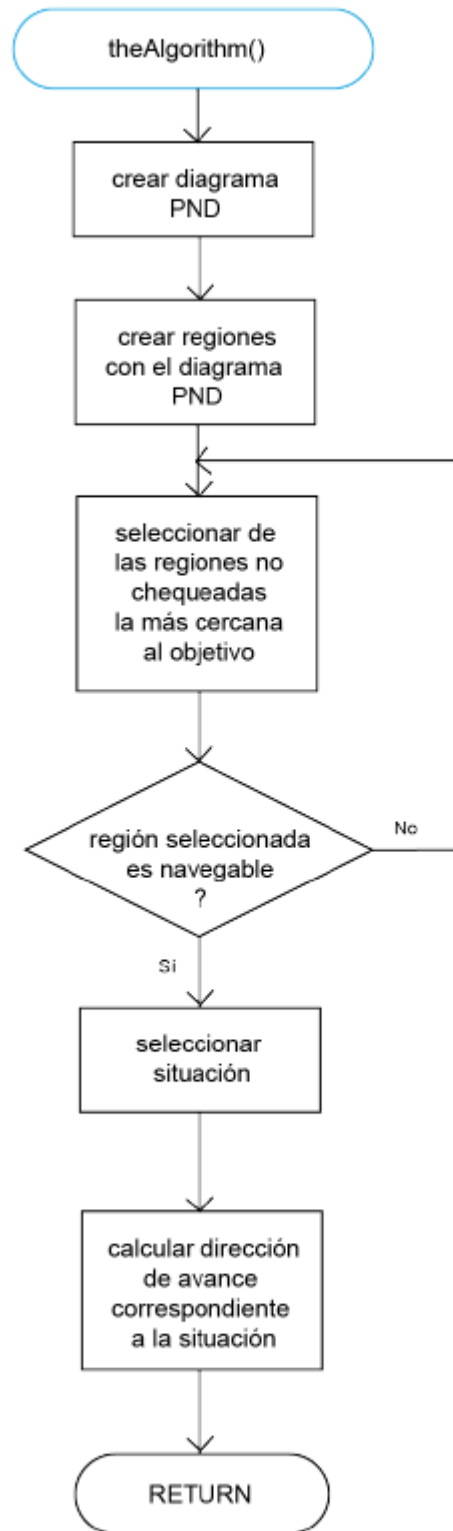
La situación de baja seguridad se llamará Low Safety 1 Side (LS1) si los obstáculos que penetran en la zona de seguridad se encuentran sólo a un lado de un cierto ángulo, el correspondiente a la discontinuidad más próxima al objetivo, perteneciente a la región de navegación. Se llamará Low Safety 2 Sides (LS2), si penetran por los dos lados. De nuevo, si el objetivo se encuentra dentro del área de libre circulación, la situación recibe el nombre de Low Safety Goal in Region (LSGR). A continuación, se muestra una imagen representativa de lo anteriormente mencionado.



Acciones

Asociada a cada una de las situaciones hay una acción, que determina la manera en que se debe calcular la dirección de avance del robot. En el caso de que el robot se encuentre a una distancia prudente de los obstáculos que lo rodean, se intentará mantener dicha distancia. En caso de que algún obstáculo se encuentre demasiado cerca, se querrá alejar al robot de éste, manteniendo el movimiento de avance por la región de navegación.





Búsqueda y selección de navegación.

Para encontrar la región navegable más cercana al objetivo que cumpla la definición de navegabilidad, el algoritmo hará un barrido de todas las regiones encontradas, comprobando su navegabilidad en orden de cercanía.

Una región se define como navegable si la distancia entre los dos puntos extremos de los obstáculos que forman la región es mayor que la dimensión máxima del robot.

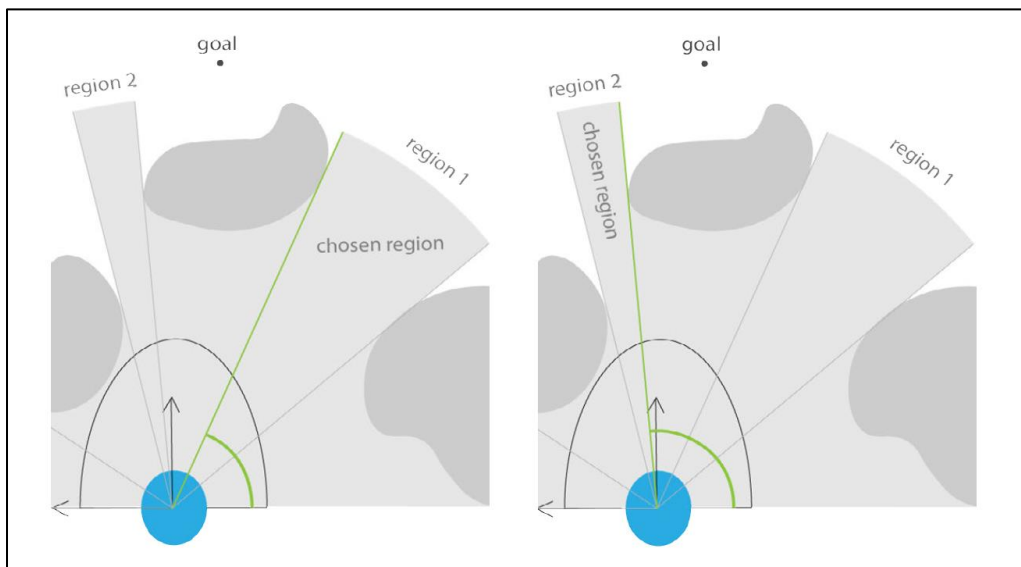
Para la elección de la región a analizar, se comienza marcando todas las regiones como no chequeadas. Se busca la región no chequeada cuyo ángulo medio sea el más cercano al objetivo, y se analiza su navegabilidad. Si no es navegable, se marca como chequeada y se analizan las siguientes regiones este se realiza hasta encontrar una región navegable o hasta que no queden regiones por chequear. En el caso de que ninguna región sea navegable, el robot deberá parar su movimiento.

Para poder clasificar la distribución del robot, obstáculos y objetivo intermedio en una de las 6 situaciones preestablecidas, es necesario comenzar analizando si existe algún obstáculo que penetre en la zona de seguridad.

En el caso de que esto ocurra, la función devolverá 1, indicando que la situación es de tipo Low Safety. En caso contrario, devolverá 0, y la situación será de tipo High Safety. Si estamos en el caso de High Safety, el entorno se clasifica en High Safety Goal in Region, High Safety Wide Region o High Safety Narrow Region, tal y como se hace en el algoritmo original.

Para el análisis dentro de la situación de tipo Low Safety es necesario conocer el ángulo de la región de navegación más cercano al obstáculo, que llamaremos α_{div} . Para ello, es necesario diferenciar dos casos: si la región está a la izquierda del objetivo, se elige como ángulo α_{div} el correspondiente al hueco que comienza la región. Si la región se encuentra a la derecha del objetivo, se elige como ángulo el del hueco que finaliza la región. El ángulo α_{div} se usará posteriormente en el cálculo de la dirección de avance, independientemente de si nos encontramos en una situación de tipo High Safety o Low Safety.

Una vez conocida α_{div} , podemos pasar a la siguiente etapa de la clasificación dentro de Low Safety. Si sólo penetran obstáculos a un lado de α_{div} , se trata de la situación Low Safety 1 Side. Si penetran obstáculos a los dos lados, se dirá que es tipo Low Safety 2. A continuación, se presenta una imagen de lo mencionado anteriormente con respecto al ángulo.



Pruebas y comparación de algoritmos

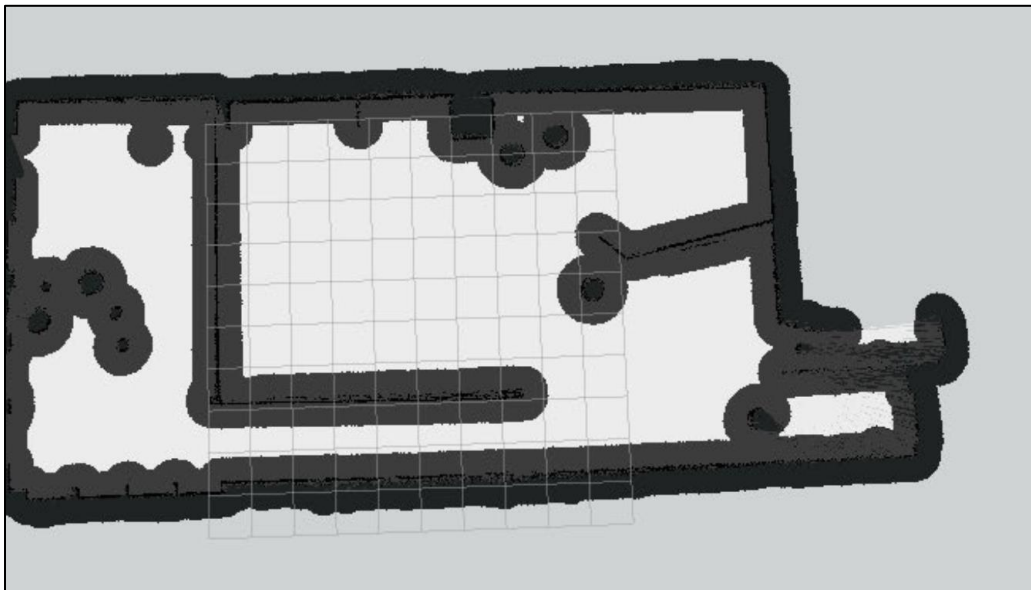
A continuación, se presenta las diferentes métricas empleadas en este trabajo:

- Numero de colisiones: cuenta el número de veces que el robot choca con un objeto dentro del entorno de trabajo.
- Duración de la trayectoria: mide el tiempo en segundo de lo que se demora el robot en conseguir la trayectoria total, es decir, desde el inicio hasta el final
- Longitud de trayectoria: mide en metro la longitud de la trayectoria total, es decir, cuantos metros recorrido desde el inicio hasta el final.

Se han generado una serie de escenarios con diferentes distribuciones de obstáculos en los cuales realizar las pruebas. Para cada algoritmo se ha realizado tres pruebas por cada mapa de trabajo.

Para la realización de las pruebas en simulación se ha construido previamente un mapa del entorno de pruebas vacío. De esta manera, el planificador global ha podido usar este mapa para el cálculo de una trayectoria global, mientras que el planificador local se ha encargado de la evasión de aquellos obstáculos que no aparecen en el ella.

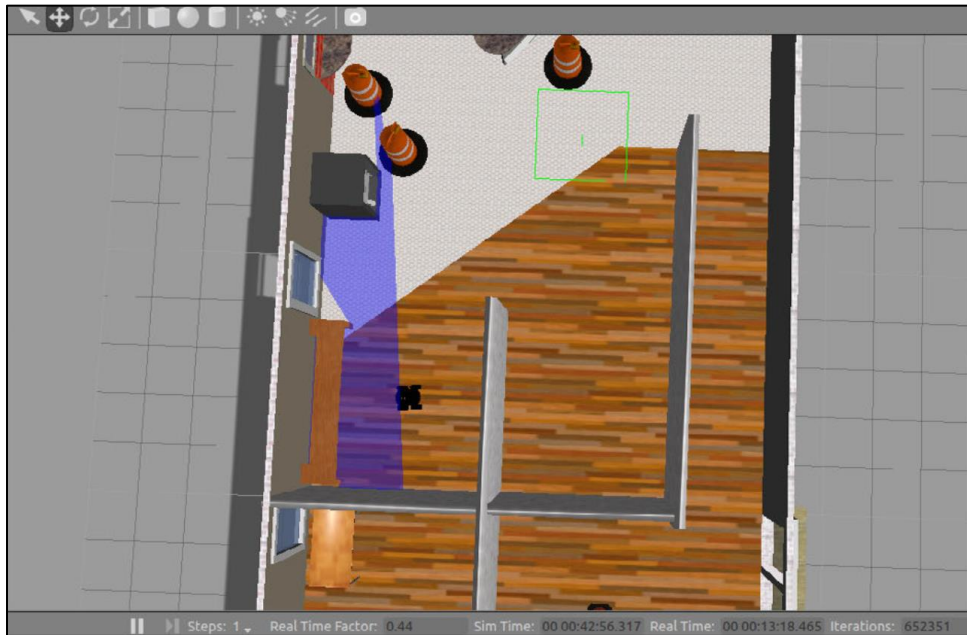
El mapa se ha creado mediante el uso del paquete de ROS de localización y mapeado simultáneo Gmapping. Con ello, se ha hecho al robot recorrer el entorno de pruebas a la vez que recogía datos con los que crear del mapa. El resultado obtenido se muestra en la siguiente imagen.



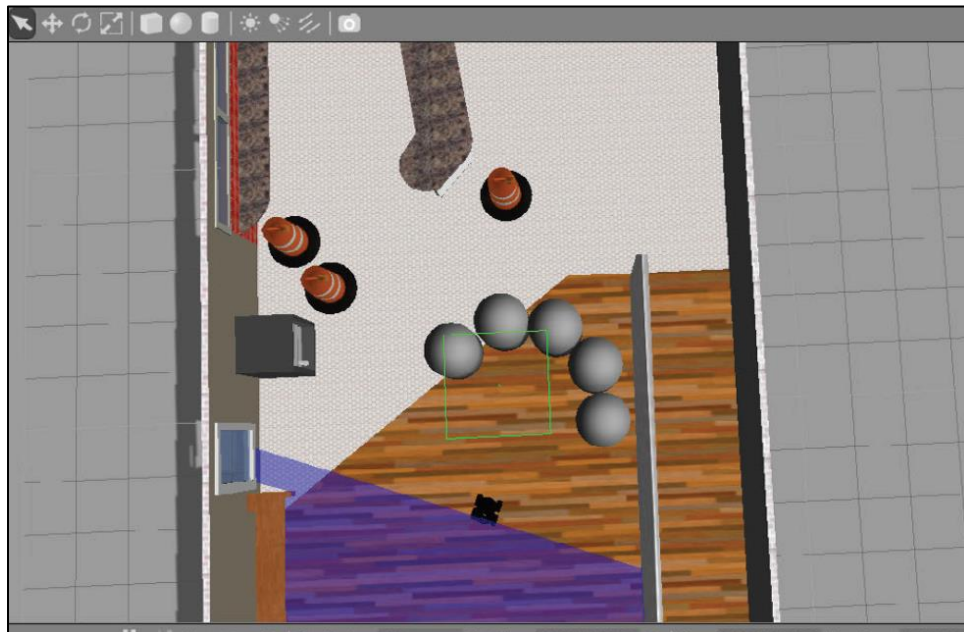
Para la comparación de los tres algoritmos se ha optado por la realización de las pruebas en tres tipos de escenarios diferentes: esquina ciega, obstáculo en U y espacio denso.

En el entorno con esquina ciega, todos los objetos en el espacio pertenecen al mapa creado previamente a excepción de una pared gris, colocada en el centro de la

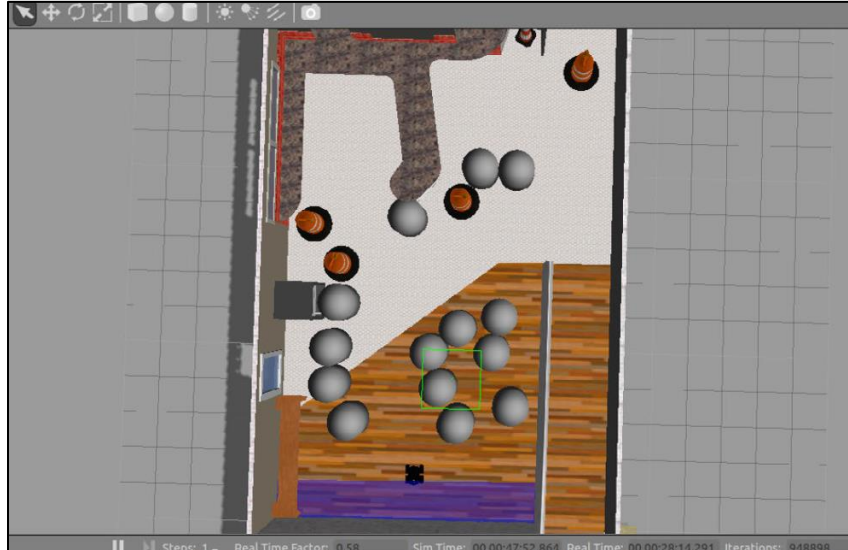
habitación. Se pretenderá que el robot, a un lado de la pared, viaje a un punto al otro lado. A continuación, se muestra el mapa de trabajo.



Para el obstáculo en U, se han añadido a la habitación una serie de cilindros grises distribuidos en forma de U. El robot se encontrará en el lado cóncavo de esta distribución, teniendo como destino un punto en el lado convexo. A continuación, se muestra una imagen de cómo queda el entorno con el obstáculo en U.



Por último, el espacio denso ha consistido en llenar el entorno de pruebas con muchos cilindros grises. El robot se encontrará a un lado de la habitación y el destino, en el lado opuesto.

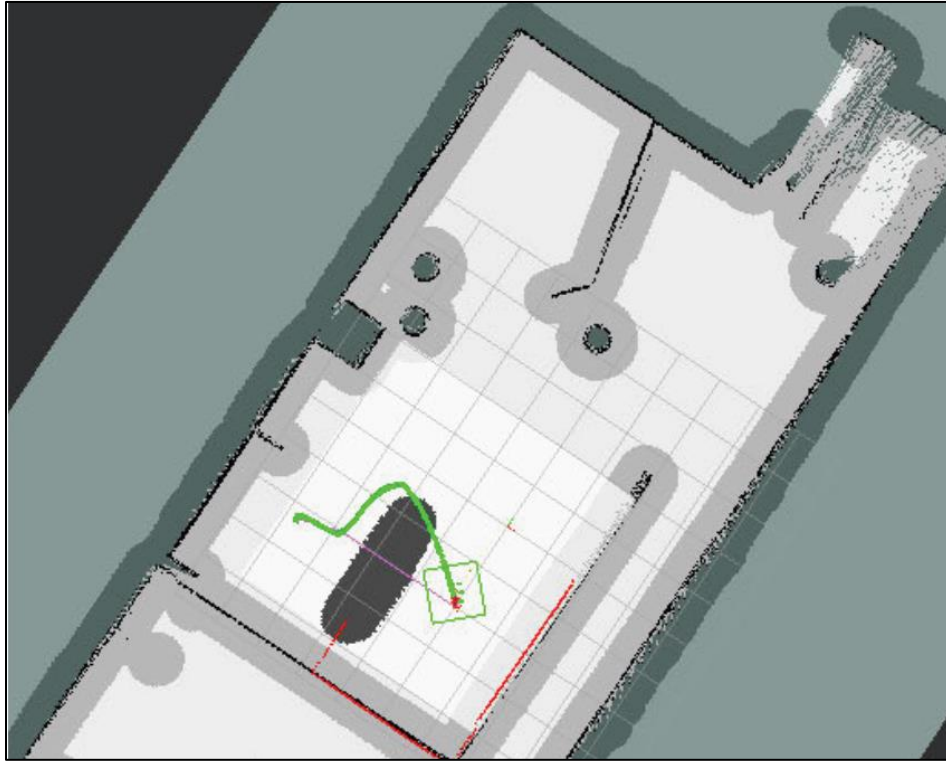


En los tres casos, el planificador global, al no tener en cuenta los nuevos obstáculos, calculará una trayectoria recta entre el robot y su destino. Durante su recorrido, el robot se encontrará con estos obstáculos, debiendo hacer uso del planificador local para esquivarlos.

Resultados del algoritmo divide and conquer

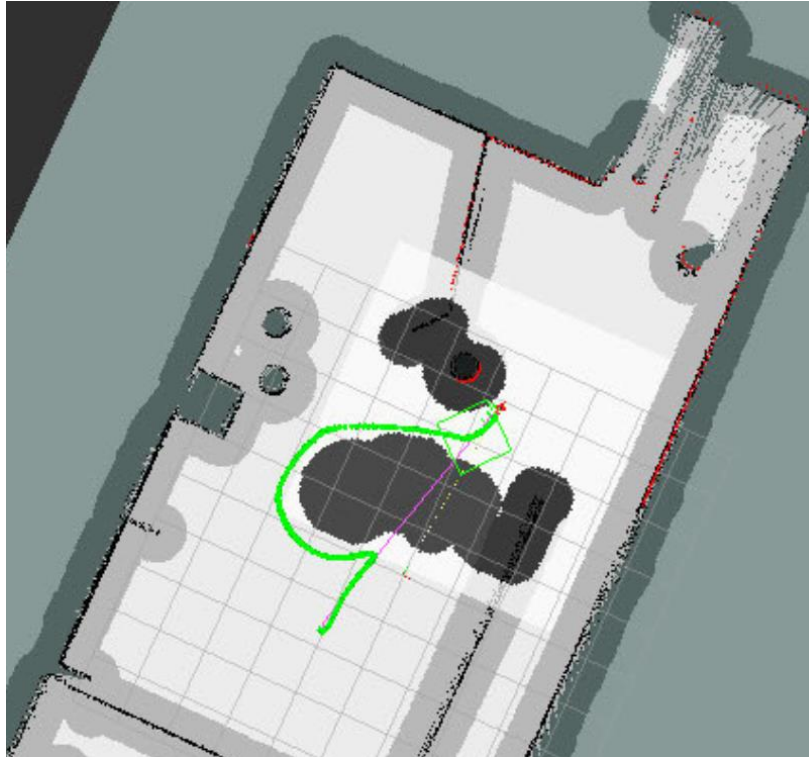
Prueba 1: Esquina ciega

En el caso de evasión de objetos de esquina a ciega con el algoritmo divide and conquer, los resultados han sido positivos. Ninguna de las pruebas realizadas ha resultado en colisión o fracaso. A continuación, se muestra la trayectoria realizada por el robot.



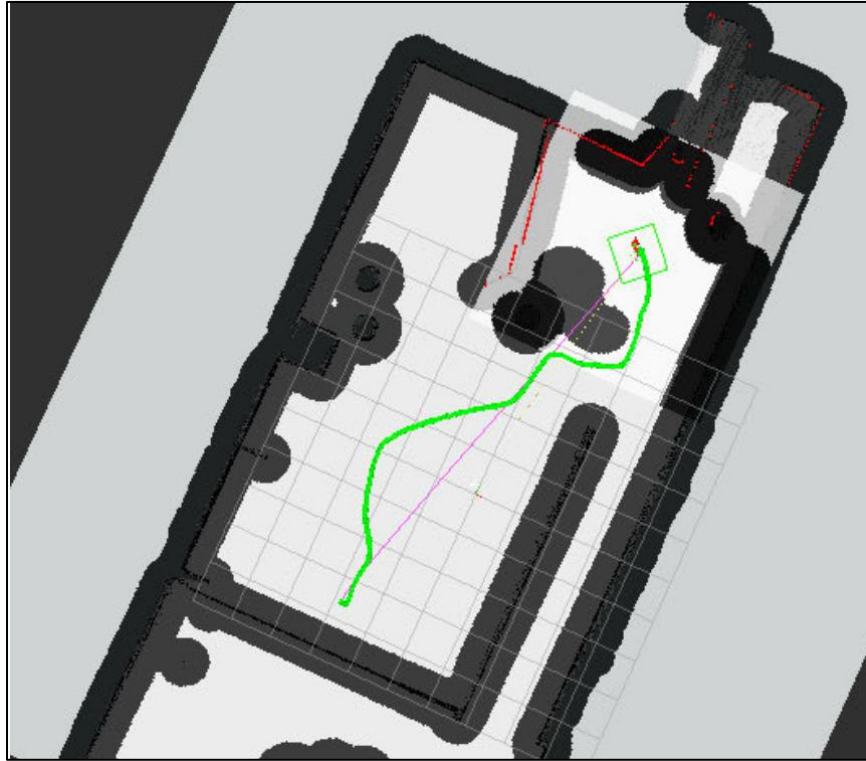
Prueba 2: obstáculo en U

Al igual que en el caso anterior la prueba ha resultado exitosa en todos los casos, ya que el robot no tuvo ninguna colisión ni trayectorias erróneas. El recorrido realizado por el robot para esta prueba se muestra a continuación.



Prueba 3: Espacio denso

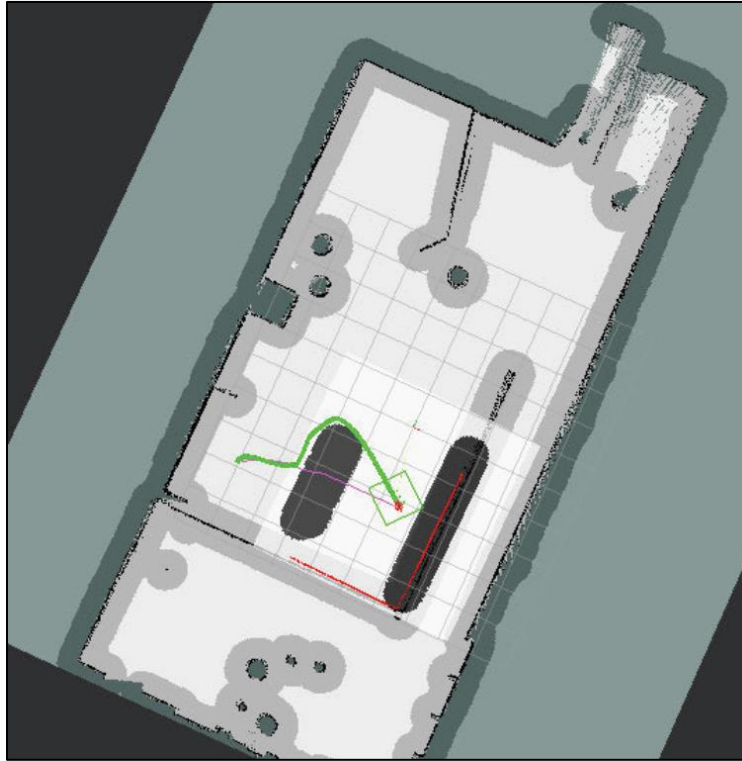
En el caso de evasión en un espacio denso con el algoritmo “Divide and Conquer”, existió un caso de colisión, pero este no es debido al algoritmo, sino a la visibilidad del sensor láser. Por esta razón, el fallo se comenta en este apartado, pero no se incluye más adelante en la tabla comparativa. Esto ocurre porque el cono con el cual colisiona el robot no es uniforme y el láser detecta la anchura del cono al nivel que se encuentra el láser, por ende, a ser más ancho en la parte inferior el robot colisiona con el objeto. A continuación, se muestra la trayectoria exitosa realizada por el algoritmo.



Resultados del algoritmo bubble rebound

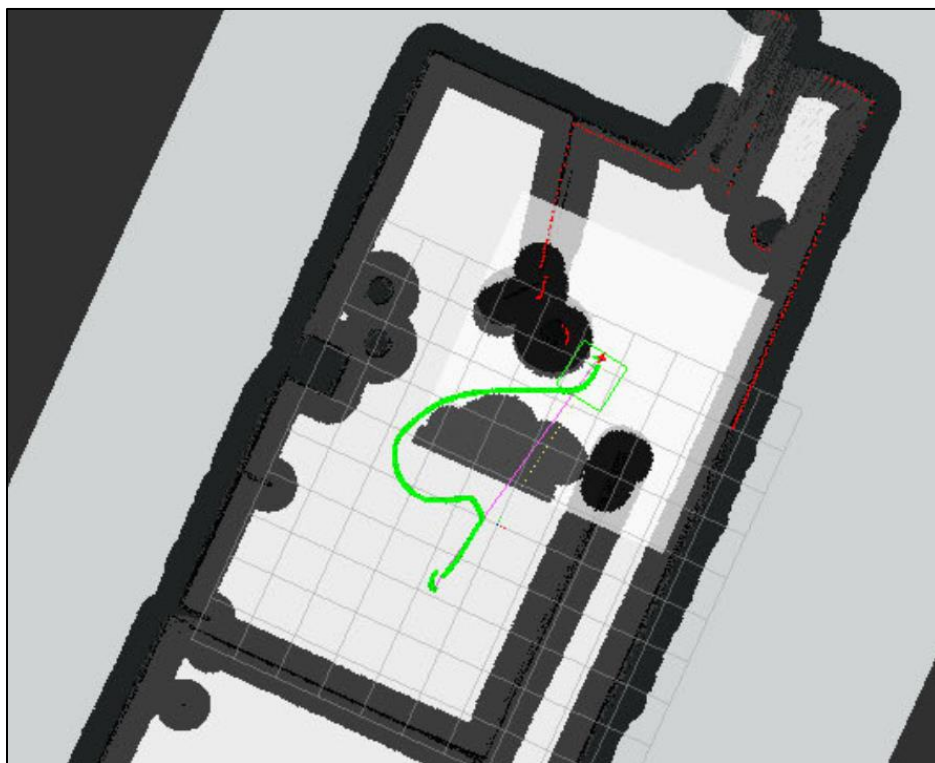
Prueba 1: Esquina ciega

La prueba en un entorno con esquina ciega ha tenido resultados bastante satisfactorios usando el planificador local mediante burbuja, ya que en las pruebas realizadas para este algoritmo y este entorno no han resultado con colisiones y trayectorias erróneas. A continuación, se muestra el trayecto realizada con éxito por el algoritmo bubble rebound.

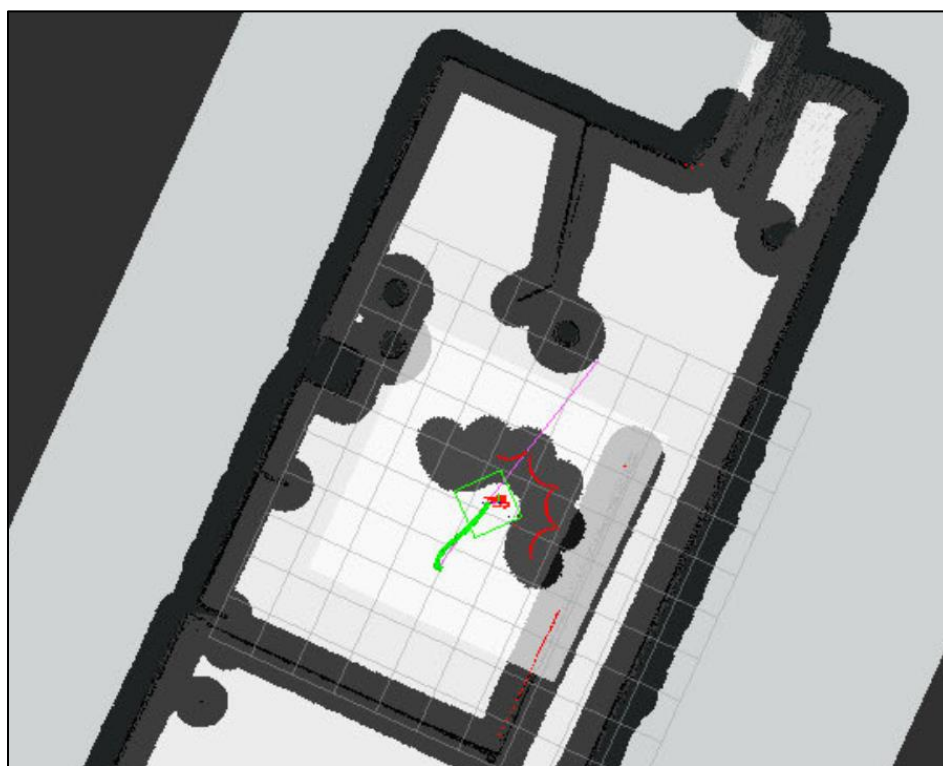


Prueba 2: Obstáculo en U

La prueba de Obstáculo en U ha sido una de las más complejas para el robot y en la cual fallo en una de sus pruebas. Sin embargo, el resto de las pruebas resultaron exitosas y logrando el objetivo de trazar una trayectoria correcta. A continuación, se muestra la trayectoria total realizada en los casos que si se pudo llegar a la meta.



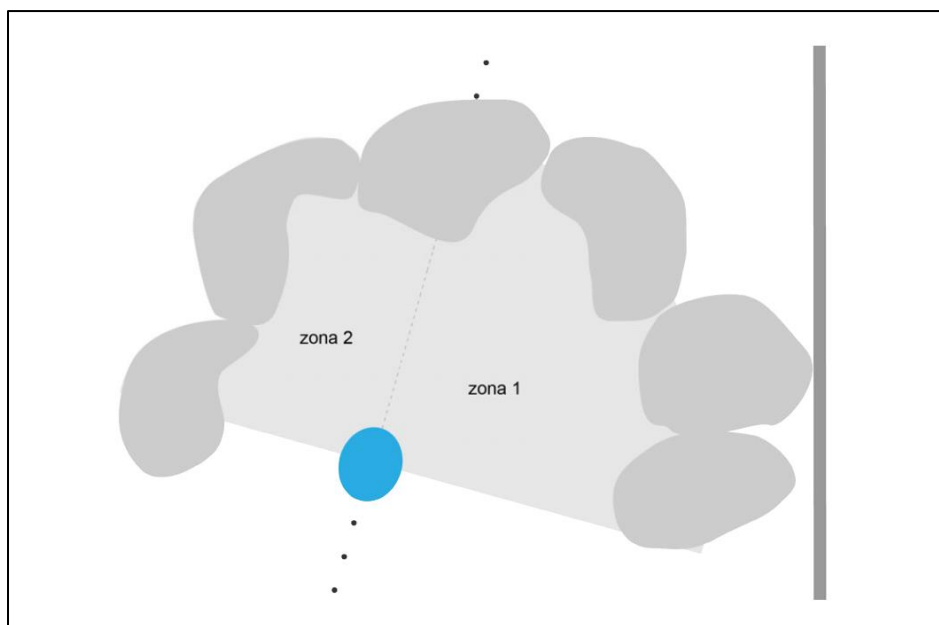
En el caso para la trayectoria que no resulto con éxito se refleja en la imagen a continuación.



Se puede observar que el robot toma como ángulo de rebote uno que lo hará rodear al obstáculo en U por la derecha. Puesto que a la derecha del obstáculo hay una pared, y que no hay espacio de paso entre el obstáculo y la pared, es evidente que no es posible rodear al obstáculo por este lado.

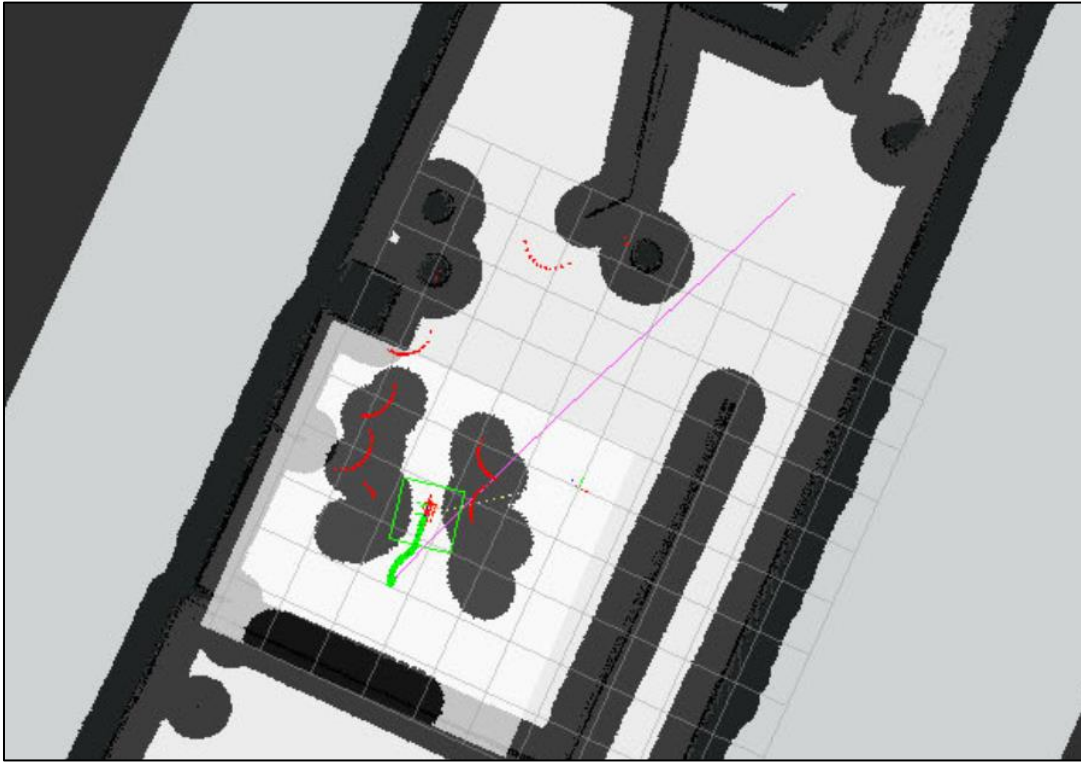
Este fallo surge por el hecho de que, cuando el robot penetra lo suficientemente en la zona del obstáculo en U, el láser sólo detecta los obstáculos que constituyen dicha zona. El robot es, por tanto, incapaz de saber que hay una pared a la derecha del obstáculo.

Esto ocurre cuando la trayectoria lleva al robot por un camino más pegado a la izquierda del obstáculo en U que de la derecha. La media de lecturas de láser será por tanto mayor en la zona 1, haciendo que el robot intente rodear al obstáculo por ese lado. Se puede observar más gráficamente este problema en la siguiente imagen.



Prueba 3: Espacio denso

La evasión de obstáculos en un espacio denso ha resultado con éxito. El robot ha realizado todas sus trayectorias sin ninguna colisión ni fallo en el ángulo de rebote. A continuación, se muestra la trayectoria realizada por el algoritmo para dicho entorno.





Tablas con los resultados para cada algoritmo y prueba

Para la realización del análisis y la comparativa de los dos algoritmos de evasión de obstáculos de planificador local, se han puesto a continuación las medidas obtenidas de dichas pruebas, ya sea para obstáculos en u, esquina ciega y entorno denso.

Entorno de pruebas		Obstaculo en U		Esquina ciega		Entorno denso	
		Bubble rebound	Divide& conquer	Bubble rebound	Divide& conquer	Bubble rebound	Divide& conquer
Numero de fracasos		1	0	0	0	0	0
Numero de colisiones	Media	0	0	0	0	0	0
Duracion (s)	Media	86,81	79,59	55,46	40,54	84,41	78,36
Longitud	Media	10,08	9,73	5,97	5,75	11,93	11,27

Como se puede observar el único fracaso realizado por las pruebas lo ha cometido el algoritmo bubble rebound y el algoritmo divide and conquer se ha comportado

mucho mejor en todas las situaciones, superando a por un leve rendimiento al algoritmo bubble rebound.

Si analizamos la longitud de los recorridos se puede observar que para el algoritmo divide and conquer que esta es mucho mejor ya que realiza su operación realizando los caminos más cortos.

Por último se añada una gráfica del movimiento lineal y movimiento angular del robot para cada prueba, obteniéndose la siguiente gráfica.

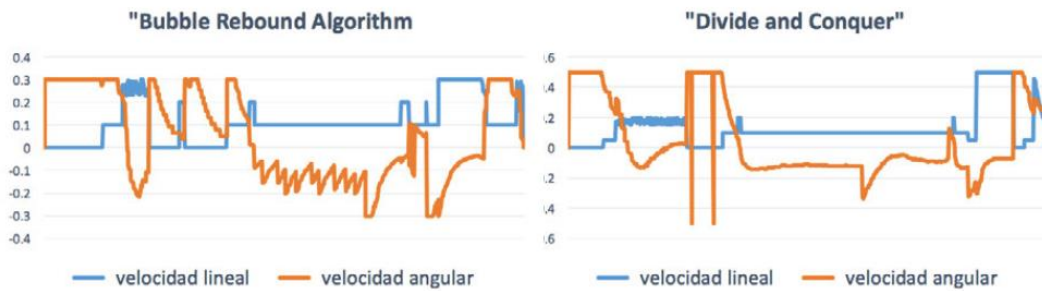
Prueba con esquinas ciegas



Prueba para entornos densos



Pruebas con obstáculo en forma de U



De las cuales se puede observar que en cuanto a la suavidad del movimiento a lo largo del trayecto no existe muchas diferencias entre los dos algoritmos, ya que en los dos casos las fluctuaciones que tiene la gráfica son varias y no tienen a mantenerse lineales.

Conclusiones

Terminado de realizar las pruebas para los dos algoritmos distintos y para cada entorno de trabajo alcanzando los objetivos de este proyecto, entre ellos el poder controlar el robot móvil a través de un entorno de simulación realizado en gazebo y el de dotar al robot móvil con la capacidad de navegar hasta su destino final sin que tengan ninguna colisión con posibles obstáculos inesperados en la ruta.

Por otro lado, durante la comparación de los algoritmos, se ha llegado al resultado que el método de evasión de obstáculos más fiable es el algoritmo divide and conquer, ya que, se encuentra más capacitado para las pruebas realizadas. Esto debido a que puede superar todo tipo de obstáculo y una mayor densidad de ellos, esto bajo los criterios analizados anteriormente mencionados como son; el número de colisiones, el número de fracaso, el tiempo de la trayectoria total y por último la longitud en metros realizadas por el robot. Además, se puede concluir que estos algoritmos trabajan con la información inmediata proporcionada por el láser para reaccionar ante entornos cambiantes.

Bibliografía

(12 de junio de 2019). Obtenido de ecagroup:

<https://www.ecagroup.com/en/business/delivery-of-a-new-iguana-ugv-to-the-monaco-police>

(1 de octubre de 2019). Obtenido de milrem robotics:

<https://milremrobotics.com/estonian-infantry-platoon-deploys-milrem-robotics-themis-ugv-on-patrol-for-the-first-time-in-mali/>

Antonio R. Jimenez, Fernando Seco. (septiembre de 2015). *Localización precisa usando radio UWB y Dead-reckoning*. bilbao.

clearpathrobotics. (s.f.). *clearpathrobotics*. Obtenido de

<https://clearpathrobotics.com/autonomous-mapping-chile-mines/>

Codelco. (s.f.). *Codelco*. Obtenido de

https://www.codelco.com/glosario/prontus_codelco/2016-06-22/175933.html

CoppeliSim. (s.f.). Obtenido de CoppeliSim :

<http://www.coppeliarobotics.com/helpFiles/index.html>

coto, e. (2003). *algoritmos basicos de grafos*.

D. Reyes, G. Millán, R. Osorio, G. Lefranc. (junio 2015). *Mobile Robot Navigation Assisted by GPS*. IEEE LATIN AMERICA TRANSACTIONS.

Eppstein, E. M. (s.f.). Obtenido de http://wiki.ros.org/nav_core

Eppstein, E. m. (s.f.). *wiki ros*. Obtenido de <http://wiki.ros.org/pluginlib>

EurecatEM. (18 de septiembre de 2017). *Un robot agrícola para gestionar los viñedos*. Obtenido de eurecat: <https://eurecat.org/es/un-robot-agricola-para-gestionar-los-vinedos/>

Felipe Nascimento Martins, Ricardo Carelli, Mario Sarcinelli-Filho, Teodiano Freire Bastos. (noviembre de 2008). *Un Controlador Dinámico Adaptable de*. San Juan, Argentina: Universidad Nacional del Sur, Bahía Blanca.

- Ioan Susnea, Viorel Minzu ,Grigore Vasiliu. (s.f.). *Simple, Real-Time Obstacle Avoidance Algorithm for Mobile*. romania.
- J. Minguez, J. O. (2004). *A "divide and conquer" strategy based on situations to achieve reactive collision avoidance in troublesome scenarios*. New Orleans.
- Julie Stephany Berrío Perez , Eduardo Francisco Caicedo Bravo, Lina Maria Paz Perez. (2013). *MAPEO Y LOCALIZACIÓN SIMULTÁNEA DE UN ROBOT*. Cancun, Mexico.
- La opinion*. (11 de febrero de 2019). Obtenido de <https://www.laopinion.com.co/tecnologia/crean-robot-para-busqueda-y-rescate-de-personas-171186#OP>
- Milremrobotics. (s.f.). *Milremrobotics*. Obtenido de <https://milremrobotics.com/product/pegasus/>
- Narváez V. Yandún F. Pozo D. Morales L. Rosero J. Rosales A. Auat F. (DICIEMBRE 2014). *Diseño e Implementación de un Sistema de Localización y Mapeo Simultáneos*. Valparaíso, Chile.
- Roberto Ponticelli. (2011). *Sistemas de exploracion con robots moviles*. Madrid.
- Barrientos Sotelo, V., & García Sánchez, J., & Silva Ortigoza, R. (2007). *Robots Móviles: Evolución y Estado del Arte*. Polibits.
- Clearpath*. (2015). Obtenido de <http://www.clearpathrobotics.com/assets/guides/husky/>
- Coxworth, B. (May 28th, 2019). Soft-bodied robot could soon be climbing up to your door. *New Atlas*.
- EcaGroup.(s.f.).*EcaGroup*.Obtenidode<https://www.ecagroup.com/en/solutions/iguan-e-ugv-unmanned-ground-vehicle>
- Nasly Perez , Diego Salamanca. (2009). *DISEÑO, MODELAMIENTO Y SIMULACIÓN 3D DE UN ROBOT MÓVIL*.

