ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

TEMA: "PLANIFICACIÓN DE CAMINOS PARA ROBOTS MÓVILES EN AMBIENTES PARCIALMENTE ESTRUCTURADOS EN REALIDAD VIRTUAL"

AUTORES:

ORFAIT PATRICIO ORTIZ CARVAJAL

ALEX MAURICIO SANTANA GALLO

DIRECTOR: ING. JULIO ACOSTA N. MsC.

LATACUNGA

2017

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Certifico que el trabajo de titulación, "***PLANIFICACIÓN DE CAMINOS PARA ROBOTS MÓVILES EN AMBIENTES PARCIALMENTE ESTRUCTURADOS EN REALIDAD VIRTUAL***" realizado por los señores ***ALEX MAURICIO SANTANA GALLO, ORFAIT PATRICIO ORTIZ CARVAJAL***, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a los señores ***ALEX MAURICIO SANTANA GALLO, ORFAIT PATRICIO ORTIZ CARVAJAL*** para que lo sustenten públicamente.

Latacunga, 19 de julio del 2017

JULIO FRANCISCO ACOSTA NUÑEZ
DIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

AUTORÍA DE RESPONSABILIDAD

Nosotros, ***ALEX MAURICIO SANTANA GALLO, ORFAIT PATRICIO ORTIZ CARVAJAL***, con cédulas de ciudadanía N°:0503454027 y 0503138950, declaramos que este trabajo de titulación "***PLANIFICACIÓN DE CAMINOS PARA ROBOTS MÓVILES EN AMBIENTES PARCIALMENTE ESTRUCTURADOS EN REALIDAD VIRTUAL"*** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 19 de julio del 2017

---------- ----------

ALEX MAURICIO SANTANA GALLO
C.C.:0503454027

ORFAIT PATRICIO ORTIZ CARVAJAL
C.C.:0503138950

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Nosotros, ***ALEX MAURICIO SANTANA GALLO, ORFAIT PATRICIO ORTIZ CARVAJAL***, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en el repositorio institucional el trabajo de titulación **"*PLANIFICACIÓN DE CAMINOS PARA ROBOTS MÓVILES EN AMBIENTES PARCIALMENTE ESTRUCTURADOS EN REALIDAD VIRTUAL*"** cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 19 de julio del 2017

----------	----------
ALEX MAURICIO SANTANA GALLO	ORFAIT PATRICIO ORTIZ CARVAJAL
C.C.:0503454027	C.C.:0503138950

DEDICATORIA

Dedico este trabajo a mis padres ***LUIS FERNANDO y ROSA ADELA*** *por todo el apoyo que me brindan a diario, así como a mis hermanos* ***LUISA FERNANDA*** *y* ***RUBEN DARIO*** *por ser la luz de mi vida. Incluyo en este homenaje a mis abuelos* ***LUIS ANIBAL, LUIS ENRIQUE, MARIA LUISA, ROSA ELVIRA****, por dar la vida a mis padres y el resto de mi familia junto con mis mentores por hacerme feliz, además de recordándome que el tiempo corre la mejor manera de viajar en él es junto a nuestros seres amados.*

ALEX SANTANA

A Dios, por permitirme llegar a este momento tan especial en mi vida. Por los triunfos y los momentos más difíciles que me han enseñado a valorarlo cada día más, a mis padres ***Patricia Carvajal y Heriberto Ortiz*** *quienes han velado por mi durante este arduo camino para convertirme en profesional y que me acompañaron durante todo mi trayecto estudiantil y de vida, a mis hermanas* ***Adriana Ortiz y Estefany Ortiz*** *con sus consejos su amor, por compartir momentos de alegría, tristeza y demostrarme que siempre podré contar con ustedes hermanas queridas.*

ORFAIT ORTIZ

AGRADECIMIENTO

Este trabajo agradase de forma especial a cada integrante de sus respectivas familias y seres amados, así como al grupo de mentores ya sea como director del proyecto al Ing. Julio Acosta M.Sc. además, dentro del entorno familiar del Laboratorio de Investigación: Automatización, Robótica y Sistemas Inteligentes (LARSI) de la Universidad de las Fuerzas Armadas ESPE-Latacunga con una mención especial al director del laboratorio Phd. Víctor Andaluz.

Orfait Ortiz C.

Alex Santana

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Una de las tendencias de la robótica es la creación de robots móviles que posean la capacidad de determinar dónde y cómo se desplazan de un punto a otro, lográndose esto de manera artificial; al desarrollar tecnología, el progreso reciente en el campo de autómatas móviles es de gran interés práctico, ya que existe una amplia variedad de aplicaciones en distintas áreas como: la gestión de residuos, exploración espacial, trabajo submarino, asistencia para los discapacitados, la cirugía médica y especialmente para la informática. De esta manera se plantea desafíos entorno a complejos problemas computacionales en los cuales surgen nuevos conceptos de amplia utilidad donde se presentan el hecho de dar solución a uno de los principales hitos como es la planificación de caminos en el cual se centra el desarrollo de este trabajo. El objetivo del presente tema de investigación consiste en exponer un procedimiento factible para desplazarse en su espacio de trabajo y ejecutar diferentes tareas mediante un ambiente virtual en el que se simula el robot móvil con el entorno del que hace parte. Este trabajo además presenta la implementación de un sistema de navegación autónomo para el control cooperativo de robots móviles en que se considera un algoritmo de planificación de caminos que usan funciones que se prioriza el mínimo tiempo que cumple el algoritmo, el uso mínimo de energía que gasta para desplazarse y la jerarquía para seleccionar cada uno de los caminos.

PALABRAS CLAVE:

- **ROBÓTICA**
- **ROBOTS MOVILES**
- **ROBÓTICA – PLANIFICACIÓN DE TRAYECTORIAS**

ABSTRACT

One of the robotic trends is the creations of the mobile robots that have the capacity to determine how and where they move from one point to another achieving this artificially. The development of technology and the recent progress in the field of mobile automatons have an important practical interest because it has a broad variety of applications on different areas such as: Waste management, space exploration, underwater work, assistant for disabled people, medical surgery, and specially for computing. In this way it is consider challenges of complex computational problems. Consequently, it is emerging new concepts of wide utility, where is represent the fact of give a solution to the principal milestones of how is the path planning in which the development of this investigative work is focused. The main objective of this research work is exposing a feasible process to move in a work space and perform different tasks using a virtual environment. That is to say the mobile robot will simulate with the environment of which is part. Moreover, this work presents the algorithm of an autonomous navigation system for the cooperative control of mobile robots. In order to consider path planning algorithm that use functions to minimize the application time the algorithm reached. Also, the hierarchy to select each of the paths by optimization process algorithm.

PALABRAS CLAVE:

- **ROBÓTICS**
- **ROBOTS MOVILES**
- **ROBOTIC – PATH PLANNING**

CAPÍTULO I

1 PROBLEMA

1.1 Planteamiento del problema

Para (Barraquand, 1991) la planificación de caminos tiene como fin principal crear algoritmos que permitan establecer caminos considerando restricciones en los movimientos de robots móviles, como a su vez la dinámica del entorno trabajado. Las aplicaciones del Path Planing (planificación del trayecto) están orientadas a diferentes labores que interactúan con el ser humano en diferentes ámbitos: i) salud: porque permite el apoyo de robots en tareas para personas de la tercera edad y personas con paraplejia, ii) militares: debido a que está enfocado en la supervisión de robots teledirigidos, autónomos y armamento inteligente, iii) industriales: monitoreo de robots con inteligencia artificial mediante el uso de robots móviles.

(Nageswara, 2000) expone en su teoría que la autonomía de un robot móvil de llegar desde un punto inicial a un punto final está delimitada por los problemas de choque con objetos fijos y móviles tanto en ambientes estructurados y no estructurados, sobre el cual debe determinar una ruta para llegar a la dirección definida, es decir encontrar el camino más corto que dicho robot pueda seguir.

Además se puede decir que la presencia de sistemas de robots múltiples en dominio de servicios es todavía limitada, acerca de extender sus horizontes a hogares para contribuir con las personas, debido al funcionamiento autónomo de múltiples robots móviles en escenarios reales, lo cual es difícil por la limitación sensorial y altos grados de complejidad computacional (Chinnaiah, 2016).

Los sistemas de robots múltiples son propensos a las restricciones en la planificación de trayectorias (Hachour, 2008). A diferencia de los

algoritmos disponibles para un solo robot en ambientes de servicio basados en los sistemas de robots múltiples, se necesita además que sean sencillos pero efectivos; y a su vez que logren lo óptimo dentro de los límites dados.

Los algoritmos tradicionales utilizados como Prim´s, Dijkstra y A* pueden no ser una solución idónea para los sistemas de robots múltiples porque se basan en gráficos pesados y árboles de extensión mínima para cálculos de planificación de trayectos que conduce a complejidad y aumento del costo de producción.

1.2 Justificación

Con el objetivo de que de que el robot pueda arribar a la meta deseada se han propuesto métodos aleatorios como RPP (“Random Path Planning)” (Lozano, 1983), y PPP “Probabilistic Path Planning” (Kavraki, 1994). Estos métodos permiten incluir en las restricciones de movimiento de los sistemas y en el espacio de configuraciones, por ejemplo, un camino encontrado, pues no es siempre válido o posible de ejecutar en referencia a las restricciones en los desplazamientos del robot “no holonomía”.

Para (Lozano, 1983) la planificación de caminos está encaminada a soluciones geométricas sin prestar atención a las restricciones impuestas por la “no holonomía” de los sistemas; Lozano además trabajó en la búsqueda de caminos en términos geométricos e introdujo conceptos de espacio de configuraciones que definen el espacio físico de trabajo del sistema y dentro del cual está localizado por su posición y orientación.

Lo que trae en los sistemas de robots múltiples más cercano al "Mundo de los robots de servicio", son los desafíos clave para muchos investigadores en todo el mundo ya que se ocupan ahora de abordar a los sistemas para que estos sean adaptables y amigables en sus aplicaciones (Latombe, 1991).

1.3 Objetivos

1.3.1 Objetivo General

- Proponer un algoritmo de planificación de caminos para la locomoción de robots móviles en ambientes parcialmente estructurados de realidad virtual.

1.3.2 Objetivos Específicos

- Investigar los diferentes algoritmos para la planificación de caminos para robots móviles, en ambientes de trabajo parcialmente estructurados.
- Proponer un algoritmo para la planificación de caminos off-line de robots móviles tipo uniciclo, para tareas de locomoción en ambientes de trabajo parcialmente estructurados.
- Desarrollar un ambiente en realidad virtual que permita simular el comportamiento de la interacción robot-entorno de robots móviles para tareas de locomoción.
- Evaluar el desempeño del algoritmo de planificación de caminos propuesto para robots móviles que interactúan en ambientes de realidad virtual parcialmente estructurados.

1.4 Artículos Realizados

Los resultados del presente proyecto de investigación constan en dos artículos presentados en diferentes revistas y el documento presentado en la estancia de investigación: 1) 18th Towards Autonomous Robotic Systems (TAROS) Conference que será próximo a ser publicado en Springer Lecture Notes in Artificial Intelligence series 2017, 2) Malaysia Technical Scientist

Association (MALTESAS), y la estancia de investigación en las instalaciones de la Escuela Politécnica Superior Universidad Autónoma de Madrid en el grupo: 3) HCTLab, se presentan en el Anexo A. es así que los artículos realizados son los siguientes.

- Victor H. Andaluz, Christian P. Carvajal and **Alex Santa G**. **"Navigation and Dynamic Control of Omnidirectional Platforms",** TAROS, Springer Lecture Notes in Artificial Intelligence series 2017.

- **Alex Santana G., Orfait Ortiz C**., Julio F. Acosta, and Victor H. Andaluz, **"Autonomous Assistance System for People with Amyotrophic Lateral Sclerosis"**, Malaysia Technical Scientist Association (MALTESAS).

- **Alex Santana G., Orfait Ortiz C**., Julio F. Acosta, and Victor H. Andaluz, **"PLANIFICACIÓN DE TRAYECTORIAS PARA ROBOTS MOVILES EN AMBIENTES SEMIESTRUCTURADOS DE REALIDAD VIRTUAL 3D VISUALIZADOS EN OCULUS RIFT.",** Escuela Politécnica Superior Universidad Autónoma de Madrid en el grupo: HCTLab.

CAPÍTULO II

2 MARCO TEÓRICO

2.1 Robótica

Para (Nourbakhsh S. R., 2004) la robótica es el conjunto de conocimientos teóricos y prácticos que permiten concebir, realizar y automatizar sistemas basados en estructuras metálicas poliarticuladas, dotadas de un determinado grado de “inteligencia” y designados principalmente a la sustitución del ser humano en diversas labores, o a la producción industrial. Es así que un sistema robótico se define como: "aquel que es capaz de recibir información, comprender su entorno a través del empleo de modelos, además de formular, ejecutar planes, y controlar o supervisar su operación".

Los grandes avances en la robótica hacen que la misma se presente como algo cotidiano para las presentes generaciones, de esta manera se puede decir que es común ver robots industriales que realizan soldaduras, pintan y mueven grandes piezas, además robots que colaboran en los laboratorios farmacéuticos, quirófanos o las actividades diarias de personas minusválidas; hasta autómatas que examinan y desactivan minas personales, buscan sobrevivientes en zonas de desastres naturales, así como también que realizan tareas agrícolas como fumigación y cosecha, sin mencionar los robots de grandes asistencias para empleados en accidentes nucleares, naufragios o exploración extraterrestre. En definitiva, la robótica ayuda en el entorno brindando seguridad y calidad.

2.1.1 Robots Móvil y Robótica

Los robots móviles (Figura 1) y los vehículos auto-guiados son dispositivos de desplazamiento autónomo, es decir, una plataforma mecánica dotada de un sistema de locomoción capaz de navegar por medio de un ambiente de trabajo definido, dotado de un cierto grado de libertad para su movimiento portando cargas o cumplimiento con una tarea (Nourbakhsh, 2004). Es así que al inicio varias investigaciones sobre el control de movimiento han sido basadas en modelos cinemáticos de robots móviles, demostrando que los controles basados en modelo cinemáticos de los robots móviles son sólo válidos para bajas velocidades y baja carga. Luego, tomando en cuenta aplicaciones de alta velocidad y de alta carga en los robots, se han desarrollado estudios sobre control de movimiento asentado en el modelo dinámico de los robots móviles.

Figura 1: Pioneer-AT-large

Los robots móviles operando en grandes entornos no estructurados deben enfrentarse con incertidumbres significativas en la posición y reconocimiento de objetos. En efecto, (Figura 2) la incertidumbre es tal que desplazase desde un punto A hasta un punto B es una labor arriesgada para

un robot móvil, relativamente trivial para un manipulador industrial. En retribución por enfrentarse con más incertidumbres del ambiente, no se espera que el mismo siga trayectorias o logre su objetivo final con el mismo nivel de precisión que se espera de un manipulador industrial (en el orden de las centésimas de milímetro).

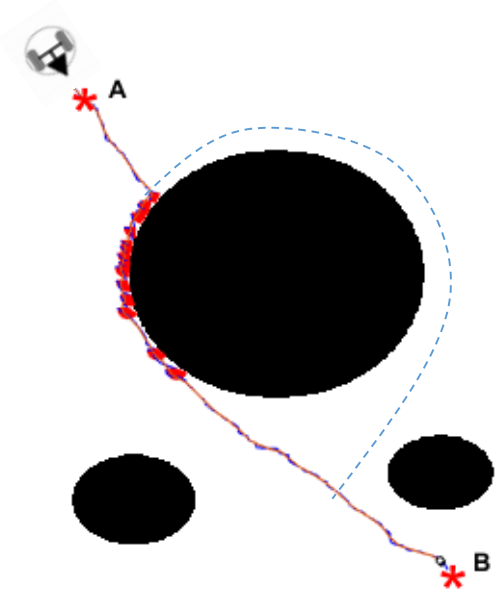


Figura 2: Algunas de las posibles trayectorias que podría seguir el robot móvil

Los robots móviles se caracterizan por su especialidad de hacerlos idóneos para un determinado contexto que lo rodea; es la tarea misma, la que define en una primera etapa las singularidades estructurales del robot que van desde el modelo de rueda, el sistema de tracción y dirección, así como la forma física del robot; en una segunda etapa la tarea establecida, las propiedades sensoriales del robot, es decir entonces de autómatas aéreos, acuáticos, terrestres, etc. (Jones, 1993).

Los robots móviles en general distribuyen sus sistemas de tracción y dirección sobre los ejes de sus ruedas de acuerdo a las exigencias de velocidad, maniobrabilidad y características del terreno. La precisión y velocidad con que el robot móvil debe obtener su destino, requiere contar con un sistema de tracción confiable y un sistema de dirección que dé maniobrabilidad al mismo. Esta confiabilidad y maniobrabilidad que debe

tener el robot móvil, determinan las características del sistema de tracción y dirección, no sólo en lo que respecta a la técnica, sino también al número de ruedas necesarias y al tipo y disposición de éstas para conseguir una estructura mecánica estable (Campion, 1996)

2.1.2 Tipos de Robots Móviles

La clasificación de los robots móviles por su tipo de locomoción, son una de las primeras características que está condicionada por su entorno. En general, los tres medios terrestres de movimiento son:

a) **b)** **c)**

Figura 3: Tipos de robots móviles según la locomoción: a) por patas, b) por ruedas, c) por orugas

Investigaciones realizadas indican que la locomoción por patas y orugas ha sido ampliamente experimentada, pero 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, se puntualiza 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. (Ortigoza, 2007)

a. Robots Móviles con Ruedas

El robot terrestre con ruedas, distinguido por su movilidad consta de dos factores: el tipo de ruedas que tienen y su disposición sobre una estructura mecánica; se toma en cuenta además que mientras se da el movimiento, el plano de la rueda se mantiene vertical y que las ruedas giran alrededor de su eje horizontal, que tiene una orientación con respecto a la estructura que puede ser estable o cambiante.

La mayoría de los robots móviles distribuyen sus sistemas de tracción y dirección sobre los ejes de las ruedas de acuerdo a las exigencias de rapidez, maniobrabilidad y características del terreno. La precisión y rapidez con que el robot móvil debe alcanzar su destino, incluye contar con un sistema de tracción confiable y un sistema de dirección que permita la maniobrabilidad al robot. Esta confiabilidad y maniobrabilidad que debe tener el robot móvil, definen las características del sistema de tracción y dirección, no sólo en lo que se relaciona a la técnica, sino también al número de ruedas necesarias, al tipo y disposición de éstas, para de esta forma, lograr una estructura fija.

Para la colocación de las ruedas sobre la estructura metálica y los diversos tipos de configuraciones, llega a tener una gran variedad de robots móviles que se distingue por su grado de maniobrabilidad. A continuación, se presentan de manera breve las principales particularidades de diseño en diversos tipos de robots móviles. (Bambino, 2008)

- **Configuración Diferencial**

Para los robots móviles es la más simpe de todas, (Figura 4) posee dos ruedas colocadas en el eje perpendicular a la dirección del robot, cada una de ellas es controlada por un motor tal, que la manera de giro del robot se origina en la diferencia de las velocidades de las ruedas.

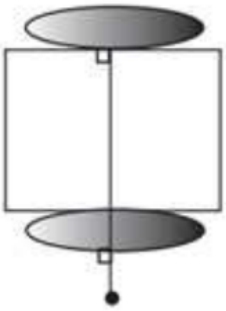


Figura 4: Robot móvil en configuración de direccionamiento diferencial
FUENTE: (Aldás, 2014)

Una de las dificultades que tiene este prototipo de robot es conservar el equilibrio, debido a que solo tiene dos ruedas, por el cual se agregan ruedas de libre giro, pero colocándolas se presenta otro problema, que es la pérdida de tracción en terrenos irregulares por lo que su uso se restringe a superficiales planas.

- **Uniciclo**

Comúnmente es el seleccionado por los investigadores a la hora de experimentar nuevas estrategias de control por tener una cinemática sencilla. También es una estructura que tiene dos ruedas fijas convencionales sobre el mismo eje, controladas de manera independiente y una rueda loca que le confiere consistencia, (Figura 5). El sistema de tracción-dirección asociado al robot le admite independizar las consignas de velocidad lineal y angular respectivamente. Las ventajas que se derivan de la estructura mecánica y de la electrónica de control hacen de esta configuración la favorita para robots de laboratorio.

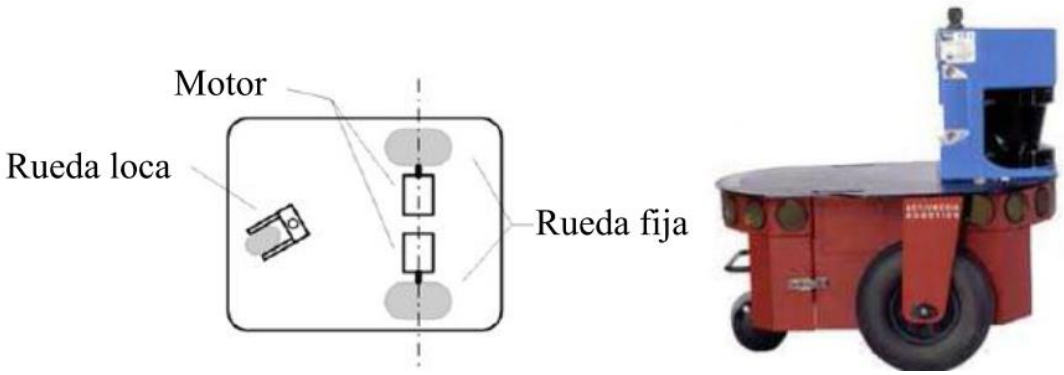


Figura 5: Robot móvil uniciclo
Fuente: (Aldás, 2014)

- **Triciclo**

El robot tipo triciclo está construido por dos ruedas convencionales fijas sobre un mismo eje y una rueda convencional centrada orientable que concentra las funciones de tracción-dirección, (Figura 6).

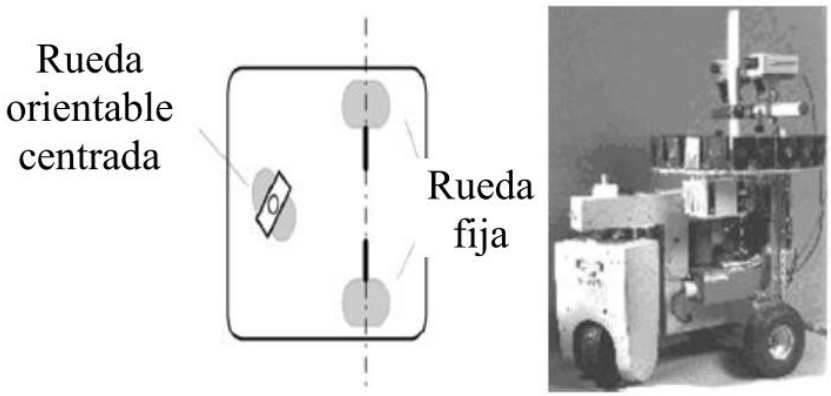


Figura 6: Robot móvil triciclo
Fuente: (Aldás, 2014)

Al igual que el caso anterior, la estructura mecánica y la electrónica de control son sencillas, su tratamiento cinemático resulta de interés en áreas específicas de control de robots móviles. En cuanto a aplicaciones industriales esta configuración es apta para el transporte de cargas pesadas a baja velocidad.

- **Cuatriciclo**

Un problema relacionado con la configuración tipo triciclo es que el centro de gravedad del vehículo, se posiciona en algunas oportunidades, en los límites de la superficie de equilibrio, establecida por las tres ruedas, cuando el vehículo está en movimiento. Esto produce una pérdida de tracción en el mismo y es fuente de error a la hora de evaluar la posición del robot. Una solución a este problema lo presenta el sistema de dirección Ackerman.

Como se observa en la Figura 7 los ejes de las dos ruedas frontales se interpretan en un punto C que pertenece al eje común de las ruedas traseras. El lugar de los puntos en el plano trazados por cada rueda, alrededor de este punto C, es un conjunto de arcos concéntricos donde todos los vectores velocidad instantánea son tangentes a estos arcos. Esta estructura, además de brindar mayor estabilidad, evita el deslizamiento en las ruedas y por lo tanto reduce los errores de odometría.

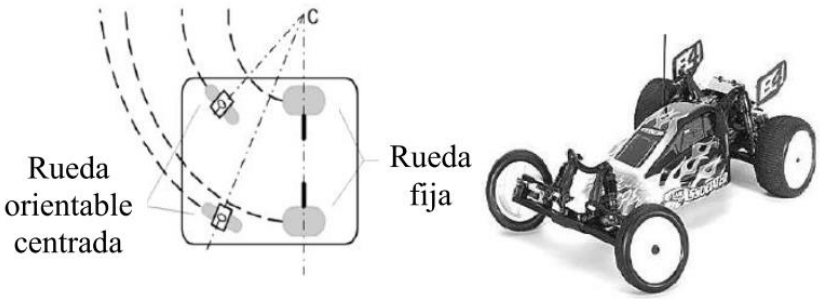


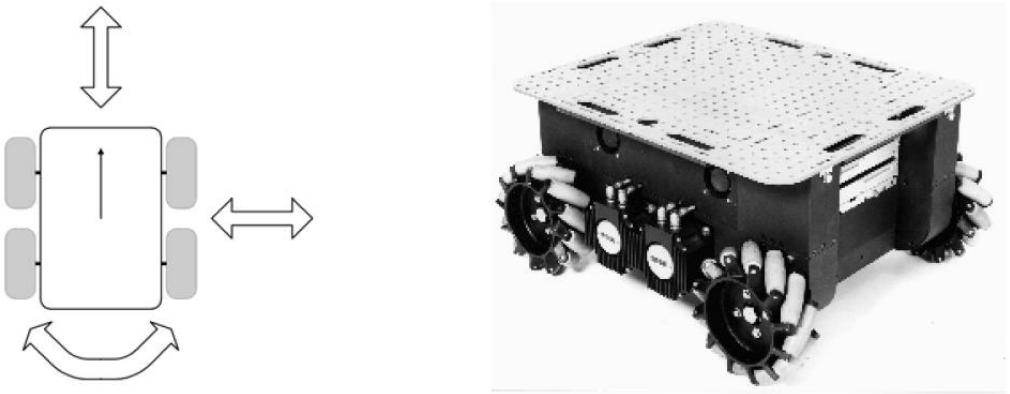
Figura 7: Robot móvil cuatriciclo
Fuente: (Aldás, 2014)

Si bien su cinemática, estructura mecánica y electrónica de control no son tan sencillas esta configuración presenta un gran interés para los amantes de los robots todo terreno en donde los principales desarrollos se producen en la recolección de información sensorial y su posterior tratamiento para lograr reconstrucción de entornos o para el control en tiempo real.

- **Robot Omnidireccional**

Estos robots tienen máxima maniobrabilidad en el plano: esto quiere decir que los mismos pueden desplazarse en cualquier dirección sin necesidad de reorientarse (Figura 8). A diferencia de los otros tipos de robots tienen una maniobrabilidad condicionada.

La Figura 8 muestra el robot omnidireccional Uranus desarrollado en la universidad de Michigan y el conjunto de movimientos posibles que el mismo puede desarrollar, de acuerdo a la rotación de cada una de las ruedas el robot puede avanzar, girar o desplazarse literalmente sin necesidad de reorientarse.



a) **b)**

Figura 8: Robot con ruedas especiales para locomoción omnidireccional: a) Maniobrabilidad, b) Robot Uranos (Carnegie Mellon University)
Fuente: (Baturone A. O., 2007)

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

2.2 Robótica Móvil Cooperativa

Actualmente, en el campo de la robótica, se está generando un gran desarrollo en el campo de los robots cooperativos. Algunos problemas son complejos de resolver para un único robot; empujar una caja, explorar un campo; es así que busca diseñar sistemas compuestos de varios robots capaces de resolver problemas de manera conjunta. Los robots que forman parte de un sistema multi-robot son simples en términos de diseño y control, además de ser menos costosos que los sistemas de un sólo robot especializado. Estos tipos de sistemas son más orientados para resolver problemas de los cuales con un solo robot le es imposible o resulta ser muy costoso, en términos de diseño y tiempo. (Rodrigo L. D., 2006)

En la industria hay varias tareas cuya ejecución resulta demasiado compleja o imposible de realizar para un solo robot, por lo cual es necesario emplear dos o más robots que trabajen de manera cooperativa para resolver la tarea que se ha consignado, además de realizar las tareas de manera más eficaz, el trabajo cooperativo muestra otros beneficios como compartir recursos e información para asistirse entre sí, y contar con mayor compresión a errores.

La robótica cooperativa en varios casos, enfrentan tareas a través del uso de enjambres de robots que contribuyen con mejores características de tolerancia, eficiencia, robustez y simplicidad a la solución. Un ejemplo de ello es el empleo de un enjambre de robots para transportar cargas, en este caso, el sistema puede utilizar más o menos robot, pero dependerá del tamaño y el peso de la carga, lo que aporta flexibilidad y eficiencia energética a la solución. Otro ejemplo, es el uso de enjambres de robots para posicionar sensores, en este caso el área de cobertura se puede ajustar modificando la formación del enjambre de robots.

En este mismo ejemplo, cuando un robot presenta errores, el enjambre puede reconfigurarse para minimizar el efecto sobre el área de cobertura, dando al sistema un importante aporte en firmeza. Finalmente, se emplea un refrán popular que se aplica dentro del contexto de robótica cooperativa “divide y vencerás”, el cual propone abordar los problemas

difíciles como conjuntos de problemas sencillos. En este caso, robots con características físicas limitadas pueden hacer frente a problemas complejos si trabajan de forma cooperativa (González, 2015).

En el presente los robots se utilizan para la exploración espacial, pues ellos tienen características que les permite mejorar movilidad en las condiciones extremas del espacio, donde su trabajo conjunto es fundamental para el mapeo y localización de puntos.

(Grace, 2014) expone que el trabajo cooperativo es aplicado incluso en la medicina, pero con la intervención de un ser humano, debido a que este campo aún requiere del razonamiento de una persona con una formación adecuada, por la cual se han diseñado sistemas en lo que uno o más robots trabajan cooperativamente con las personas.

Un factor a resaltar del uso de las tareas cooperativas, es que los integrantes compartan el espacio físico y trabajan sobre el mismo objeto, por lo cual es importante planear las tareas para especificar y determinar a qué robot le corresponde realizar cada acción y así evitar choques entre ellos (Figura 9).

Figura 9: Robots cooperativos fabricados a partir de piezas de LEGO

2.2.1 Tipos de Control en Sistemas Cooperativos

Para sistemas cooperativos se tienen dos tipos de control en función del nivel de comunicación e interacción entre los distintos robots que forman el sistema.

a. Control Centralizado

En este tipo de control se registran dos casos; un ordenador que es el encargado de controlar, supervisar y establecer las acciones de los robots o un robot central que se en carga de la toma de decisiones a través de información enviada continuamente por cada uno de los demás robots, la elaboración de los algoritmos resulta simple debido a que se la hace en un solo punto aunque por otro lado, el tiempo de toma de decisiones se puede alargar debido a que todos los otros robots constantemente están enviando información a la unidad central.

El usar un ordenador como control centralizado da el beneficio de que todos los movimientos y conflictos entre los distintos robots móviles son resueltos de forma fácil. La función del mismo es la distribución de tareas del sistema y la transmisión del flujo de información a los robots, tomando en cuenta que en la unidad central almacena información al que es más eficiente a la hora de obtener soluciones. De la misma forma esto supone también un problema, si hay algún fallo o el exceso de transmisión de datos a los otros robots, se puede provocar una congestión al recibir una gran cantidad de datos, y ralentizarse la toma de decisiones en el centro de control, el sistema entero se detiene y queda inutilizado.

b. Control Distribuido

Para el control distribuido, cada robot decide que hacer por su cuenta sin recibir órdenes de ninguna otra entidad. La elaboración de los algoritmos de programación es mucho más difícil diseñarlos, aunque son muy simples,

deben de trabajar apropiadamente para todos los robots. Con este tipo de control el sistema es mucho más eficiente y rápido, no es necesario tanto el envió como el almacenamiento de información; además, si deja de funcionar un robot, todos los demás pueden seguir funcionando independientemente.

Es difícil hallar un algoritmo sencillo que funcione, pero una vez encontrado es muy fácil de emplear e implementar. En este caso, dependiendo del nivel de comunicación entre los robots, los sistemas pueden ser:

- **Muy acoplados o colectivos**

Cada persona tiene su propia inteligencia, pero también poseen capacidad de comunicación, así como de colaboración y coordinación; de esta manera se concibe que la inteligencia colectiva es la suma de la inteligencia de cada individuo. Cada robot debe tener la capacidad de transportar información a otro colaborador y contar la suficiente inteligencia para pedir ayuda y coordinar los movimientos con los demás individuos. Por ejemplo, cuando dos robots intentan mover una caja empujándola de uno de sus lados y que se mantenga recta. Los dos robots deben de coincidir sus velocidades para evitar que la caja se ladee hacia un lado u otro; son sistemas poco robustos, pues si falla un robot, falla todo el sistema.

- **Poco acoplados o manadas**

(Rodrigo, 2006) expresa que son sistemas donde los robots funcionan independientemente del resto, sin tener en cuenta los movimientos o decisiones de los otros. No hay una inteligencia central, sino que todo el control es distribuido, cada elemento simplemente sigue unas reglas. Este tipo de comportamiento trata de imitar sistemas biológicos como el de los insectos sociales. Por ejemplo, se quiere tener un grupo de robots, que recojan las pelotas de un campo de tenis una vez acabados los

entrenamientos, y las lleven a una esquina del campo. Cada robot buscará una pelota, la detectará, recogerá, y trasladará al lugar adecuado. Para este trabajo no es necesario que los robots se comuniquen entre ellos.

2.2.2 Ventajas y Desventajas del uso de Robots Cooperativos

Tabla 1: Ventajas y desventajas de los sistemas cooperativos

VENTAJAS	DESVENTAJAS
Acciones mucho más rápidas: Al dividir las tareas entre varios equipos, el trabajo se realiza de forma más rápida y eficiente.	**Interferencia de robots:** Debido a los sensores o el tipo de control que se tiene, los robots pueden confundirse entre sí y no realizar la tarea de manera adecuada.
Tolerancia a fallos: Si un robot falla, este error puede ser compensado por los otros robots que forman parte del equipo.	**Costo de comunicación:** Los robots deben tener una comunicación fluida entre ellos o con la unidad central, por lo cual los equipos para este fin son muy importantes.
Programación distribuida: Cada robot tiene un algoritmo específico.	**Incertidumbre entre robots:** El sistema pierde eficiencia cuando un robot realiza una acción no planificada que fue realizada por otro miembro del equipo.
Bajos costos: Resulta más económico tener varios robots no especializados que uno solo muy costoso.	

Fuente: (Grace P. Cano, 2014)

2.2.3 Trabajos Realizados Alrededor del Transporte Cooperativo

El transporte de objetos es una de las aplicaciones del trabajo cooperativo que suelen ser un problema para un solo robot, mediante la

utilización de múltiples mecanismos como: sujeción, truncamiento o simplemente arrastre son los usados en el transporte cooperativo; a continuación se muestran algunos trabajos:

- (Rodrigo, 2006) diseño una plataforma cooperativa meramente reactiva aplicada al transporte de una caja larga, su enfoque utilizo la cooperación en tres niveles: censado, acción y control, para la distribución de tareas entre dos robots hexápodos equipados con sensores de luz y de contacto que les permitía generar comportamientos de sujeción y seguimiento de luz dependiendo del estado de la tarea. (M. Mataric, 1995)
- Desarrollo de un sistema multirobot descentralizado cooperativo enfocado al movimiento de un grupo de cajas desordenadas, su diseño consistía en el desarrollo de tres habilidades. La primera consistía en lanzar una cuerda alrededor de los objetos con la intención de sujetarlos. La segunda se enfoca en la utilización de una cuerda floja para acomodar las cajas internamente. Por último el sistema tenía la capacidad de transportar las cajas a acomodadas a otro sitio asegurando su orden con la implementación de trinquetes sobre las cajas. (B. Donald, 2000)
- (Bonabeaub, 2000) presento el primer modelo formal para transporte cooperativo cuyo enfoque fue la resolución de problemas descentralizada mente. En este trabajo el autor evidencia que si el tamaño del objeto es más pequeño con respecto al sistema de sujeción de los robots el desempeño de la plataforma decrece drásticamente ya que la superficie de contacto es limitada.
- Desarrollo un sistema multirobot enfocado en el transporte de objetos cooperativamente para lo cual utilizo una estrategia que le permitía atrapar o encerrar el objeto con los robots a través del diseño de algoritmos descentralizados basados en campos de potencial que le permitieron a la plataforma mantener el objeto atrapado simultáneamente de irlo arrastrando hasta una meta deseada. (Kumar, 2002)

2.3 Path Planning

En el campo de la robótica móvil se conoce como planificación de trayectorias o (path planning) que consiste en encontrar una secuencia de acciones que transforma desde algún estado inicial, hacia un estado de llegada deseado. La planificación de trayectorias tiene por objetivo determinar un camino específico que lleve a la meta o destino. Usualmente descompone el camino en sub-puntos o sub-metas, los cuales constituyen los puntos del camino.

Éstos tienen una posición fija en el entorno o espacio de búsqueda del robot, posición representada por las coordenadas (x, y). Una técnica del planificador de trayectorias depende principalmente del tipo de sensores que posea el robot (a bordo o fuera del cuerpo del mismo), ya que éstos son los encargados de recoger información del ambiente que lo rodea, y a partir de estos datos el robot es capaz de construir un modelo abstracto del espacio en el que se desenvuelve.

No toda la información disponible es relevante para la planificación de trayectorias; mientras más datos se introduzcan al sistema más variables tendrán que controlar, siendo más difícil dirigir un robot al objetivo. Los sensores pueden ser desde detectores de tacto hasta avanzados sistemas de visión, GPS y radares que permiten tener un rango de alcance mucho mayor, un algoritmo de planificación de trayectorias es óptimo si siempre encontrara un camino recomendable. Existen varios enfoques para calcular trayectorias dado alguna representación del medio ambiente. En general, las dos técnicas populares son los algoritmos determinísticos basados en heurísticas (Hart, Nilsson & amp; Rafael 1968, Nilsson 1980) y Algoritmos aleatorios (Kavraki et al., 1996; LaValle 1998; LaValle y Kuffner 1999; 2001). (Ferguson D., 2005)

2.3.1 Métodos para planeación de trayectorias

Convencionalmente la dificultad de la generación de trayectorias para un robot móvil indica que, dado un robot y un ambiente, se puede determinar una trayectoria entre dos puntos específicos si el robot está libre de colisiones y si satisface cierto criterio de desempeño. (Eduardo H., 2011)

Existen multitudes de algoritmos de planificación en robótica, y la mayor parte de ellos se pueden agrupar o considerar ciertas técnicas desarrolladas para la planeación de trayectorias, se pueden proponer la siguiente clasificación:

a. Determinísticos:

- Algoritmos Bug (pulga).
- Métodos de Wavefront.
- Basados en grafos:
  - Grafos de visibilidad.
  - Diagramas de Voronoi.
  - Modelado del espacio libre.
  - Descomposición en celdas.
- Campos potenciales artificiales.

b. Probabilísticos y aleatorios (algoritmos de búsqueda aleatoria):

- Planeador aleatorio de trayectorias (Randomized Path Planner RPP).
- Mapas probabilísticos (Probabilistic Road Maps PRM).
- Arboles de exploración rápida (Rapidly Exploring Random Tree RRT).
- Basados en optimización:

- Algoritmos genéticos (Genetic Algorithms GA).
- Colonia de hormigas (Ant Colony Optimization ACO).
- Enjambre de partículas (Particle Swarm Optimization PSO).
- Quimiotaxis bacteriana (Bacterial Chemotaxis BC).

2.3.2 Planificación Bug (pulga)

Los Bug son un conjunto de algoritmos que pueden ser implementados como planificadores simples, para la construcción de trayectorias de navegación, en ambientes reales con presencia de obstáculos. Cuando uno de estos algoritmos, se enfrenta a una dificultad desconocida, es capaz de generar fácilmente el contorno del objeto obstáculo con el que se encuentra, este contorno es representado como un conjunto de coordenadas de una superficie 2D, solo si un camino a la meta existe.

Entonces el objetivo es crear un camino o trayectoria libre de choques mediante la circunnavegación y las restricciones dadas por el obstáculo u objeto. Los algoritmos Bugs han sido implementados en el dominio de la robótica móvil con el objetivo de planificar los movimientos a través de la edificación de una trayectoria solución libre de colisiones, que parte desde un punto inicial hacia un punto final, los algoritmos actúan bajo tres hipótesis acerca del robot móvil: El robot es un punto, maneja localización perfecta (no incertidumbre), sensores precisos. El conjunto de los algoritmos bugs se descompone de los siguientes: Bug 1, Bug 2 y DisBug, entre otros. (Márquez, 2011).

El algoritmo Bug 1, tan pronto como sea detectado el obstáculo i, el robot hace un contorno total alrededor, comenzando con el punto de llegada H_i. El contorno completo apunta a evaluar el punto de distancia mínima del objeto L_i. El robot continuo con el movimiento alrededor del contorno hasta encontrar el punto que estaba buscando nuevamente, donde se va a lo largo de un camino recto hacia el objeto. L_i es nombrado como se va al punto. Esta técnica es muy ineficiente, pero da garantía de que el robot llegara a

cualquier punto deseado. (Figura 10) representa una situación con dos obstáculos donde $H1$ y $H2$ son los puntos de intersección y $L1$ y $L2$ los puntos de llegada de cada obstáculo. (Ribeiro, 2005)

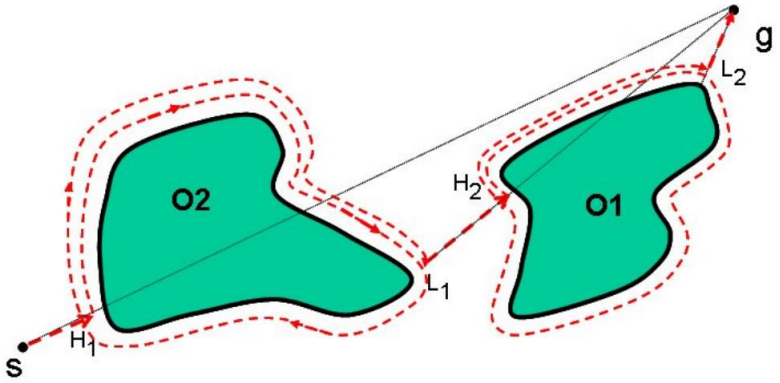


Figura 10: Evitación de obstáculos con el algoritmo Bug 1
Fuente: (Ribeiro, 2005)

El algoritmo Bug 2, empieza apegarse al contorno del obstáculo en el punto H_i y termina cuando el robot cruza la línea del objetivo. Esto permite el punto de autorización L_i del comportamiento de los límites del obstáculo; de L_i el robot se mueve directamente al objetivo, el procedimiento se repite si más obstáculos son detectados. (Figura 11) representa el camino generado por el algoritmo Bug 2 por dos obstáculos.

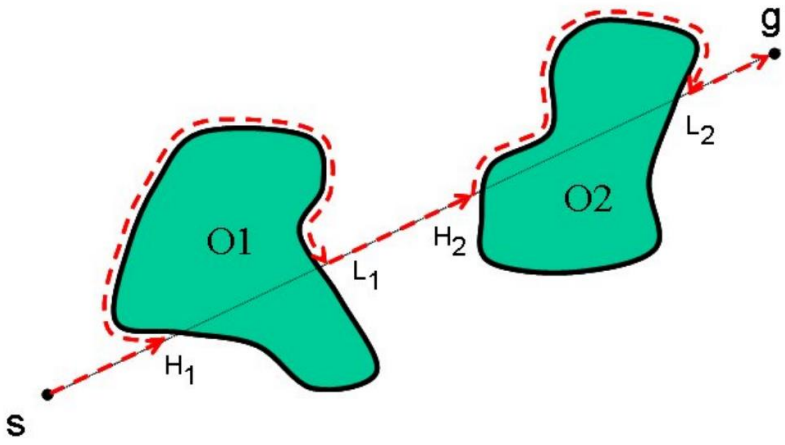


Figura 11: Evitación de obstáculos con el algoritmo Bug 2
Fuente: (Ribeiro, 2005)

En general al algoritmo Bug 2 tiene un tiempo de recorrido más corto que el algoritmo Bug 1 y especialmente más eficiente en espacios abiertos. Sin embargo, hay situaciones donde puede convertirse no óptimo. En particular el robot puede estar atrapado en estructuras de laberinto. (Ribeiro, 2005)

(Ribeiro, 2005), manifiesta que el algoritmo DistBug trata de asegurar que el destino sea alcanzado, si es posible, o indicar si el objetivo es inalcanzable. El algoritmo es reactivo en el sentido de que depende de los datos recolectados de su entorno para tomar decisiones locales, y no utiliza ninguna forma de representación interna del medio ambiente. El algoritmo se compone de dos modos de locomoción: se mueve directamente hacia la meta entre los obstáculos siguiendo los bordes de los obstáculos.

De esta manera los obstáculos son circunnavegados hasta que cierta condición indica que se debe volver hacia el punto final. La dirección que indica en qué modo se debe seguir los bordes, depende de la posición del punto final e influirá en la longitud de la ruta que será cubierta. El algoritmo decide en qué dirección se moverá sobre la base de la información adquirida.

El algoritmo DistBug, fue inventado por Kamon y Rivlin en 1997. Es muy similar al algoritmo bug2 porque tiene una condición idéntica acerca de irse, con apenas una sutil diferencia; la única divergencia real es que DistBug no mantiene una lista de los puntos anteriores, mientras que Bug2 sí. La siguiente descripción explica cómo funciona el algoritmo DistBug en forma global: Traza una ruta directa desde el punto inicial al punto final, si el objetivo es alcanzado realiza una parada, y en caso de hacer contacto con un obstáculo sigue las fronteras (hacia la izquierda) hasta que llegar a la meta sea posible, repetir.

Esta técnica es simple, pero en un ambiente controlado es capaz de evitar obstáculos y a su vez una trampa como los mínimos locales. Sin embargo, se limita a un medio ambiente estático, sin obstáculos en movimiento. En este algoritmo, el robot tiene un máximo de detección sensorial representado en un rango *R* (*distancia detectada*) y dos comportamientos básicos que son límite de seguimiento, y de movimiento a las acciones de meta.

Como se observa en la Figura 12, el modelo de planificación de movimientos es representado en plano cartesiano de dos dimensiones. Y opera de la siguiente manera: en primer lugar, el robot se mueve hacia *G* (*punto final*) hasta que se enfrenta a un obstáculo. Entonces su límite de acción de seguimiento se activa en sentido horario (por defecto). Durante los limites siguientes el robot realiza el registro de las distancias mínimas, denominadas *dmin*(*G*), que significa la distancia obtenida de la trayectoria a *G* (*punto final*) lograda desde el punto de impacto pasado. El robot también detecta la distancia en el espacio libre *F* (*distancia libre*), que es una distancia de un obstáculo del robot de localización, *X* (punto de localización), en la dirección de G (*punto final*).

Si ningún obstáculo se detecta, *F* (*distancia libre*) se encuentra en *R* (*distancia detectada*). El robot deja obstáculo límite como se indica en la (Figura 12), solo cuando la ruta de acceso a *G* (*punto final*) es clara, es decir cuándo: $d(X,G) - dmin \leq F(G) - Paso$ donde, $d(X,G)$ es la distancia localizada en un punto X al punto G (punto final) - *dmin* que corresponde a las distancias obtenidas de la trayectoria, son menores o iguales a la distancia libre a G – *Paso,* que es la constante de desplazamiento predefinida. (Márquez, 2011)

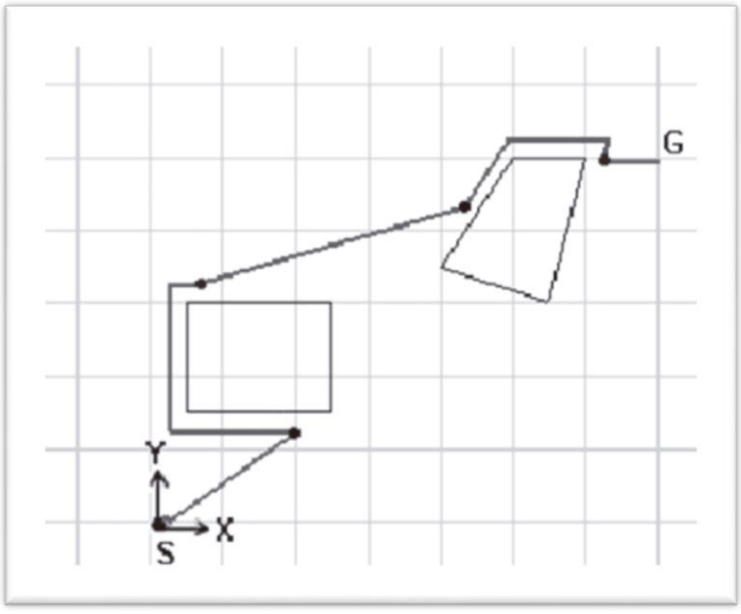


Figura 12: Camino generado por el algoritmo DistBug
Fuente: (Márquez, 2011)

La simplicidad de estos algoritmos conduce a una serie de deficiencias. Ninguno de ellos toma en cuenta la cinemática del robot que es una severa limitación especialmente en el caso de los robots holonómicos, Además, solo se tiene en cuenta los datos sensoriales más recientes como el sensor ultrasónico que tiene un gran impacto en el rendimiento del robot. (Ribeiro, 2005)

2.3.3 Planificación Basada En Grafos

Estos métodos consisten en construir un grafo que interconecte el origen con un destino deseado, posteriormente es necesario desarrollar caminos óptimos con factores de decisión como la distancia más corta y latencia.

a. Grafos De Visibilidad.

Los grafos de visibilidad proporcionan un enfoque geométrico útil para resolver el inconveniente de la planificación; supone un entorno bidimensional en el cual los obstáculos están modelados mediante polígonos, para la generación del grafo este método introduce el concepto de *visibilidad*, según el cual precisa dos puntos del entorno como *visibles* si y solo si se pueden unir mediante un segmento rectilíneo que no intersecte ningún obstáculo (si dicho segmento resulta tangencial a algún obstáculo se consideran los puntos afectados como visibles). Es decir, el segmento definido debe encontrarse en el espacio libre del ambiente (Nilsson, 1.969).

Así, si se considera como nodos del grafo de visibilidad la posición inicial, la final y todos los vértices de los obstáculos del entorno, el grafo resulta de la unión mediante arcos de todos aquellos nodos que sean visibles.

La Figura 13 se muestra el grafo de visibilidad construido junto a los obstáculos poligonales existentes en el entorno y las configuraciones inicial *qa* y final *qf*. (Baturone, 1995)

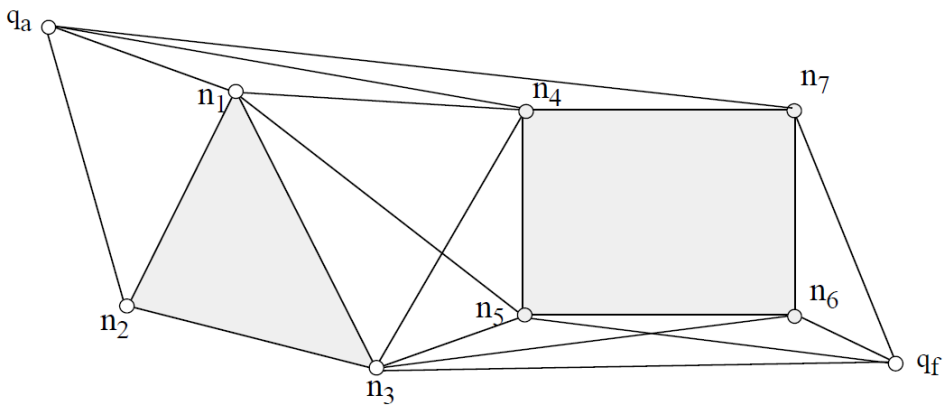


Figura 13: Grafo de visibilidad en un entorno de dos obstáculos
Fuente: (Baturone A. O., 2007)

En el grafo mostrado de la Figura 13 se puede observar cómo sólo están unidos los nodos directamente visibles, de tal forma que el conjunto de arcos estará formado por las aristas de los obstáculos, más el resto de líneas que relacionan los vértices de los diferentes polígonos.

Mediante un algoritmo de búsqueda en grafos se elige la trayectoria que una la configuración inicial con la final minimizando alguna función de coste; la ruta que cumple el objetivo de la navegación queda definida como una sucesión de segmentos que siguen los requisitos especificados.

Aunque en principio el método está desarrollado para entornos totalmente conocidos, existe una versión denominada LNAV (Rao *et al.* 1.988) capaz de efectuar una planificación local a medida que se realiza la labor de navegación. Este algoritmo, que parte de una determinada posición, determina los nodos visibles desde el punto actual, elige el más cercano de los nodos visibles, según distancia euclídea a la posición final para desplazarse posteriormente al nodo seleccionado y marcarlo como visitado. Desde esta nueva posición se vuelve a iterar el proceso hasta llegar a la posición final (éxito), o bien no existen más nodos sin visitar (fracaso).

Dentro de los métodos basados en grafos de visibilidad, se encuentran algoritmos especializados en la búsqueda del camino óptima que

lleve al vehículo desde la posición inicial A hasta la final B a través de un entorno en el cual el espacio libre entre obstáculos está modelado mediante el uso de dos cadenas de segmentos (Lodares, Abellana y Muñoz, 1.993).

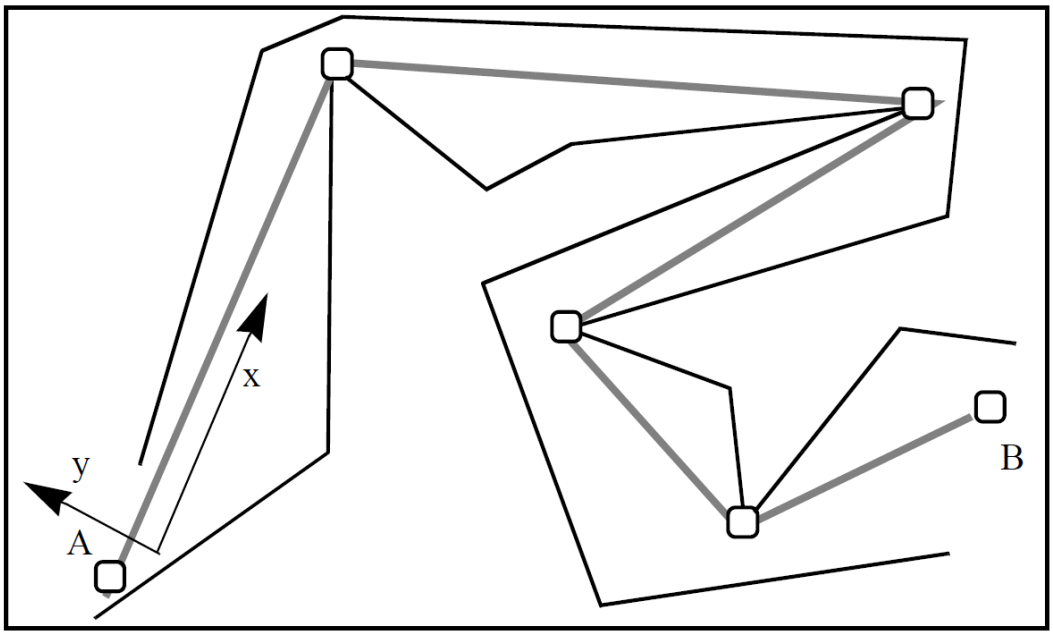


Figura 14: Planificación con el espacio libre de obstáculos modelado mediante cadenas
Fuente: (Baturone D. D., 1995)

Aunque están restringidos a esquemas de entornos muy concretos, el uso queda justificado debido a su bajo coste computacional; como se puede observar en la Figura 14, los algoritmos desarrollados para encontrar la ruta óptima bajo las condiciones descritas, se basan en enlazar los nodos situados en las zonas convexas del entorno tal que dos nodos consecutivos son visibles.

El uso de métodos de planificación basados en grafos de visibilidad está muy extendido, debido a que se pueden construir algoritmos a bajo coste computacional que resuelvan el referido problema, sin embargo, utilizar como nodos los vértices de los obstáculos implica que no son inmediatamente aplicables en la práctica, ya que un robot móvil real no consiste en un punto. Por ello, algunos autores como (Latombe, 1.991) denominan al camino planificado semi-libre de obstáculos.

b. Diagramas de Voronoi.

Al contrario que los métodos basados en grafos de visibilidad, la planificación basada en diagramas de Voronoi sitúa la ruta lo más alejada posible de los obstáculos; con ello elimina el problema presentado por los grafos de visibilidad de construir rutas parcialmente libres de obstáculos. (Baturone, 1995)

Los diagramas de Voronoi se definen como una proyección del espacio libre del entorno en una red de curvas unidimensionales yacientes en dicho espacio libre. Formalmente se definen como una retracción (Janich, 1.984) con preservación de la continuidad. Si el conjunto C_l define las posiciones libres de obstáculos de un entorno, la función retracción *RT* construye un subconjunto C_v continuo de C_l.

$$\boldsymbol{RT(q): C_l \rightarrow C_v / C_v \subset C_l} \qquad \textbf{(1)}$$

De esta forma, se dice que existe un camino desde una configuración inicial qa hasta otra final qf, supuestas ambas libres de obstáculos, si y solo si existe una curva continua desde *RT(qa)* hasta *RT(qf).*

La definición de la función retracción RT implica la construcción del diagrama de (Voronoi y Rombaut, 1.991). La idea fundamental como se ha expuesto es ampliar al máximo la distancia entre el camino del robot y los obstáculos, por ello el diagrama de Voronoi resulta el lugar geométrico de las configuraciones que se encuentran a igual distancia de los dos obstáculos más próximos del entorno; es así que el mismo estará formado por dos tipos de segmentos: rectilíneos y parabólicos.

La elección de la modalidad de segmento corresponde con la clase de elementos de los obstáculos más cercanos que se encuentren enfrentados entre sí; de esta forma, el lugar geométrico de las configuraciones que se hallan a igual distancia de dos aristas de dos obstáculos diferentes es una línea recta, mientras que en el caso de tratarse de un vértice y una arista resulta una parábola.

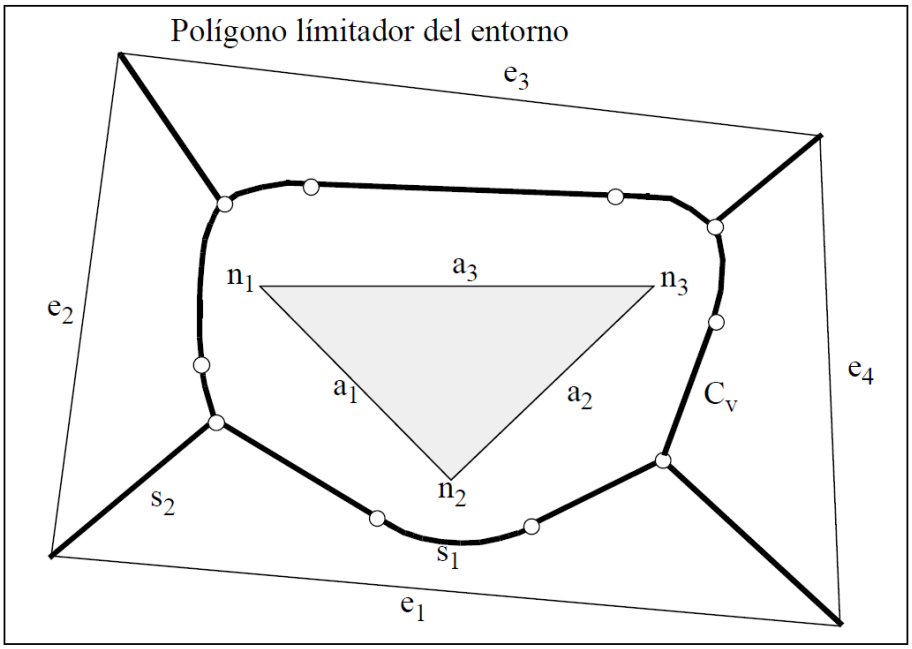


Figura 15: Retracción del espacio libre en un diagrama de Voronoi
Fuente: (Baturone D. D., 1995)

En la Figura 15 se muestra un entorno delimitado por un polígono de aristas $\{e_1, e_2, e_3, e_4\}$ y un obstáculo triangular de vértices $\{n_1, n_2, n_3, \}$ y aristas $\{a_1, a_2, a_3, \}$. La retracción del espacio libre en una red continua de curvas es el diagrama de Voronoi C_v, representado mediante las líneas de trazo grueso. Los dos tipos de segmento utilizados en la construcción del diagrama pueden distinguirse en la mencionada figura, así, el segmento s_1 es el lugar geométrico de los puntos equidistantes entre la arista e_1, y el vértice n_2. Por otra parte, puede observarse como el segmento rectilíneo s_2 cumple la misma condición, pero con respecto a las aristas e_1 y e_2.

Dado una configuración q, no perteneciente a C_v, existe un único punto *p* más cercano perteneciente a un vértice o arista de un obstáculo. La función *RT(q)* se define como el primer corte con C_v, de la línea que une *p* con *q* (Figura 16).

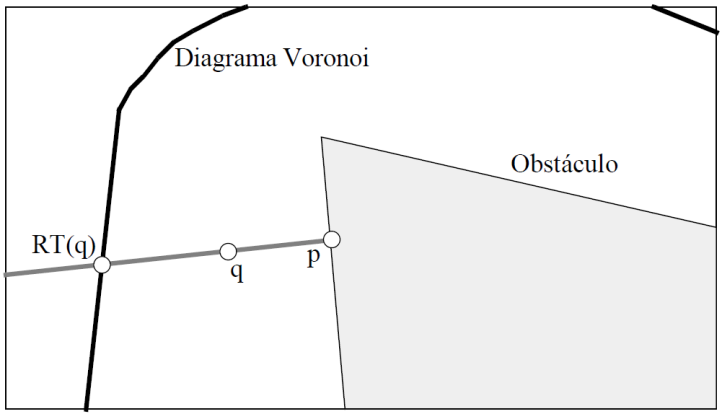


Figura 16: Una configuración q en el diagrama de Voronoi
Fuente: (Baturone A. O., 2007)

El algoritmo de planificación en esencia, consiste en encontrar la secuencia de segmentos s_i del diagrama de Voronoi tal que conecten *RT(qa)* con *RT(qf),* Dicha secuencia conforma el camino buscado. A continuación se describe el algoritmo:

i) Calcular el diagrama de Voronoi.
ii) Calcular *RT(qa)* y *RT(qf).*
iii) Encontrar la secuencia de segmentos {s_i,..., s_p } tal que *RT(qa)* pertenece a s_i y *RT(qf)* pertenece a s_p.
iv) Si se encuentra dicha secuencia, devolver la ruta. Si no indicar condición de error.

Al igual que los grafos de visibilidad, este método también trabaja en entornos totalmente conocidos y con obstáculos modelados mediante polígonos. Sin embargo, también existen versiones para la utilización del mismo con obstáculos inesperados (Meng, 1.988).

c. Modelado del espacio libre

Según (Brooks, 1.983) se aplica patrones de entornos con obstáculos poligonales, y la planificación en este caso se realiza mediante el modelado

del espacio libre Esta acción se lleva a cabo por los denominados cilindros rectilíneos generalizados (CRG). Al igual que los diagramas de Voronoi, con el uso de los CRG se pretende que el vehículo navegue lo más alejado de los obstáculos. De forma que la ruta que lleve al robot desde una configuración inicial hasta otra final estará compuesta por una serie de CRG interconectados, de tal modo que la configuración de partida se encuentre en el primer cilindro de la sucesión y la final en el último. (Baturone, 1995)

La construcción de un CRG se realiza a partir de las aristas de los distintos obstáculos que se encuentran en el entorno. Para que un par de aristas ${}^1a_i$ y ${}^2a_j$ pertenecientes a los obstáculos *b1* y *b2* respectivamente puedan formar un cilindro generalizado, deben cumplir las siguientes condiciones:

i) La arista ${}^1a_i$ está contenida en una recta que divide al plano en dos regiones. La arista ${}^2a_j$ debe yacer por completo en la región opuesta en la que se encuentra situada *b1*. Este criterio es simétrico.

ii) El producto escalar de los vectores normales con dirección hacia el exterior del obstáculo que contiene cada arista debe resultar negativo.

Si se cumplen estas condiciones significa que ambas aristas se encuentran enfrentadas, y por tanto se puede construir un CRG con ellas (Figura 17).

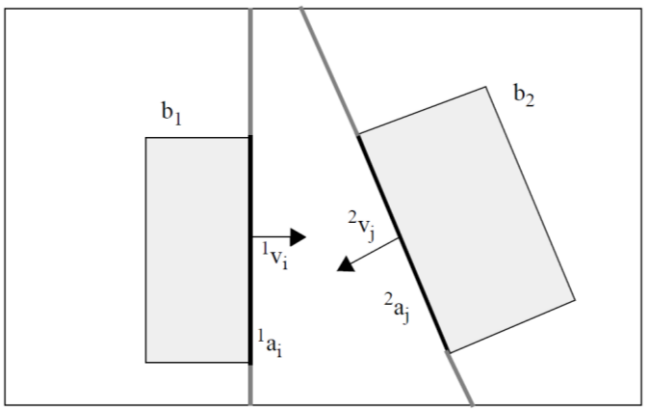


Figura 17: Condiciones que deben cumplir dos aristas para construir un CRG

Fuente: (Baturone A. O., 2007)

Una vez detectadas dos aristas que pueden formar un CRG, el siguiente paso será construirlo. El proceso para alcanzar este cometido se encuentra descrito en la Figura 18.

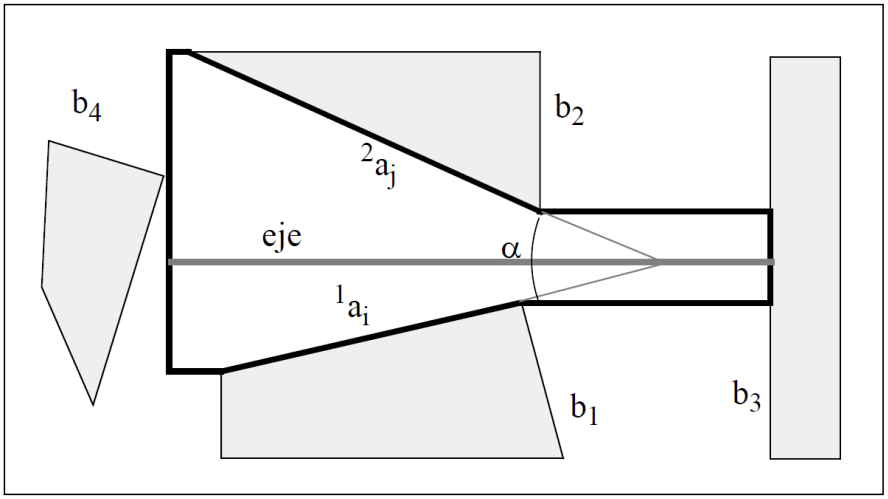


Figura 18: Construcción de un CRG
Fuente: (Baturone D. D., 1995)

El primer paso es el cálculo del eje del CRG, el cual se define como la bisectriz del ángulo formado por el corte de las rectas que contienen las aristas ${}^{1}a_i$ y ${}^{2}a_j$ que cumplen las condiciones i) y ii) expuestas más arriba. Por ambos lados de dichas aristas se construyen segmentos rectilíneos paralelos al eje, con origen en los vértices de las aristas implicadas y con extremo señalado por la proyección del primer obstáculo que corta el eje.

Repitiendo este proceso, se construye una red CRG en el entorno del robot que modela el espacio libre del mismo. El robot navegará por el eje del cilindro, en el cual se encuentran anotadas para cada punto el rango de orientaciones admisibles. El paso de un CRG a otro se produce siempre y cuando sus ejes intercepten y la intersección del rango de orientaciones admisibles en el punto de corte de ambos ejes no sea nulo.

d. Descomposición en celdas

Para (Thorpe, 1,984) este tipo de métodos se fundamenta en una descomposición en celdas del espacio libre Así, la búsqueda de una ruta desde una postura inicial qa hasta otra final *qf*, consiste en encontrar una sucesión de celdas que no presente discontinuidades, tal que la primera de ellas contenga a *qa* y la última a *qf*.

Al contrario que los métodos expuestos a lo largo de este apartado, no encuentra una serie de segmentos que modele la ruta, sino una sucesión de celdas; por ello se hace necesario un segundo paso de construcción de un grafo de conectividad, encargado de definir el camino. (Baturone, 1995)

Para la planificación según el método de descomposición en celdas, se precisa la resolución de dos problemas: la descomposición del espacio libre en celdas y la construcción de un grafo de conectividad. El primero de ellos implica construir unas celdas con determinada forma geométrica tal que resulte fácil de calcular un camino entre dos configuraciones distintas pertenecientes a la celda, y la comprobación para averiguar si dos celdas son adyacentes debe disfrutar de la mayor simpleza posible. Aparte de estas características, la descomposición global del espacio libre implica que no deben existir solapamientos entre celdas y que la unión de todas ellas corresponde exactamente al espacio libre.

El grafo de conectividad es un grafo no dirigido, y su construcción está asociada a la descomposición en celdas efectuada en el paso anterior, de tal forma que los nodos van a ser cada una de las celdas, existiendo un arco entre dos celdas si y solo si son adyacentes (Figura 19).

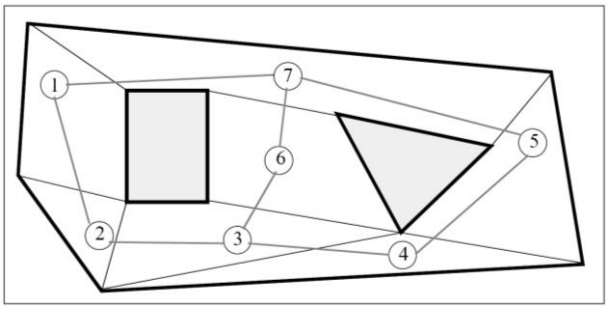


Figura 19: Descomposición en celdas y grafo de conectividad
Fuente: (Baturone D. D., 1995)

Una vez especificado el grafo de conectividad, sólo queda emplear un algoritmo de búsqueda en grafos para la detección de la celda que contiene la postura a la cual se desea llegar, tomando como partida la que contiene la postura inicial.

Los distintos métodos basados en este principio, se distinguen por la forma en la cual realizan la descomposición en celdas y como se construye el grafo de conectividad. El método más sencillo de descomposición del espacio libre del entorno en celdas resulta el denominado descomposición trapezoidal (Latombe, 1,991). Este método se basa en la construcción de segmentos rectilíneos paralelos al eje Y del sistema global a partir de los vértices de cada uno de los elementos del entorno. El final del segmento queda delimitado por el primer corte de la línea con un elemento del entorno. Esta descomposición es la mostrada en (Figura 20).

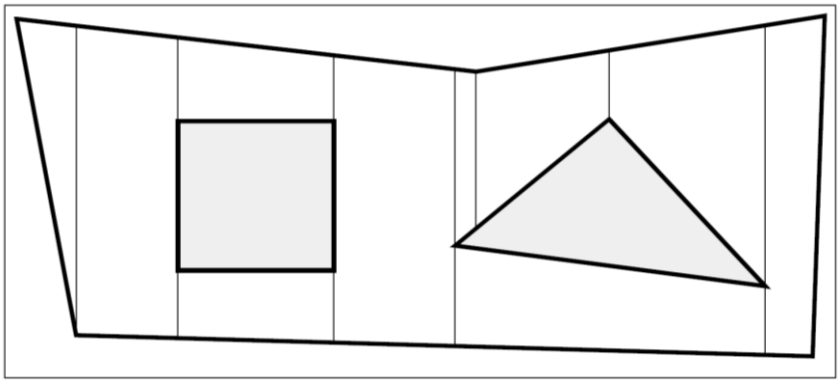


Figura 20: Descomposición trapezoidal del espacio libre
Fuente: (Baturone D. D., 1995)

El grafo de conectividad se construye por medio de la unión de los puntos medios de los segmentos verticales definidos (Figura 21).

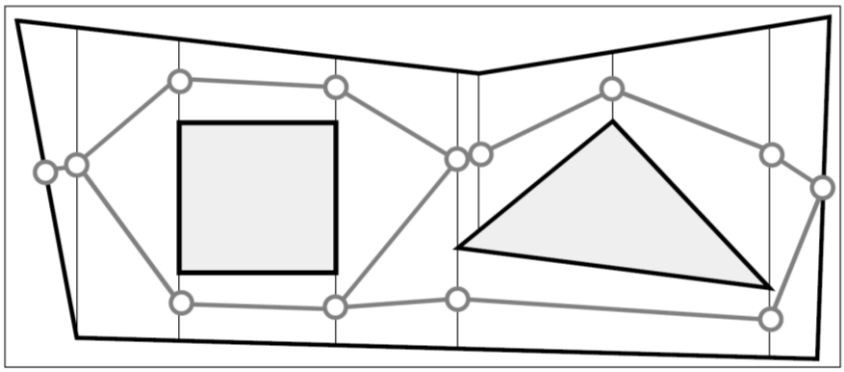


Figura 21: Grafo de conectividad de una descomposición trapezoidal
Fuente: (Baturone D. D., 1995)

Este tipo de enfoque se presta a muchas variantes, por ejemplo la utilización de varios niveles de resolución para una búsqueda jerárquica (Kambhampati y Davis, 1.986), o bien el uso de celdas en tres dimensiones para la planificación de caminos en espacios tridimensionales (Stentz, 1.990).

2.3.4 Planeación de trayectorias empleando campos potenciales

Los métodos basados en campos potenciales poseen una concepción totalmente distinta a los expuestos más arriba al estar basados en técnicas reactivas de navegación. El ámbito de uso de esta técnica se centra en la planificación local en entornos desconocidos, como puede ser el sorteo en tiempo real de obstáculos o de los que no se tiene constancia (Borenstein y Koren, 1.991)

La teoría de campos potenciales considera al robot como una partícula bajo la influencia de un campo potencial artificial, cuyas variaciones modelan el espacio libre. La función potencial *U* en un punto *p* del espacio euclídeo, se define sobre el espacio libre y consiste en la composición de un potencial atractivo *Ua(p)*, que atrae al robot hacia la posición destino, y otro repulsivo *Ur(p)* que lo hace alejarse de los obstáculos, es decir:

$$U(p) = U_a(\mathrm{p}) + U_r(\mathrm{p}) \qquad \textbf{(2)}$$

La fuerza artificial *F(p)* a la que afecta el vehículo en la posición *p*, por el potencial artificial *U(p)* resulta:

$$F(p) = -\nabla U(p) \qquad \textbf{(3)}$$

Al igual que la función potencial, la fuerza artificial es el resultado de la suma de una fuerza de atracción $F_a(p)$ proveniente de la posición destino, y otra fuerza de repulsión $F_r(p)$ debidas a los obstáculos del entorno de trabajo:

$$F(p) = F_a(p) + F_r(p) \quad \textbf{(4)}$$

Así la navegación basada en campos potenciales se basa en llevar a cabo la siguiente secuencia de acciones:

i) Calcular el potencial *U(p)* que actúa sobre el vehículo en la posición actual p según la información recabada de los sensores.
ii) Determinar el vector fuerza artificial *F(p)* según la expresión (3).
iii) En virtud del vector calculado construir las consignas adecuadas para los actuadores del vehículo que hagan que éste se mueva según el sentido, dirección y aceleración especificadas por *F(p)*.

La iteración continua del ciclo expuesto proporciona una navegación reactiva basada en campos potenciales. El comportamiento del vehículo está muy ligado a la definición que se efectúe de los potenciales de atracción y repulsión. El potencial de atracción debe ir en función de la distancia euclídea a la posición destino, de forma que a medida que el robot móvil se acerca para que este disminuya su influencia. Por otra parte, el potencial repulsivo conviene que sólo influya en el movimiento del vehículo cuando éste se encuentre demasiado próximo a un obstáculo, de forma que la fuerza debida a este hecho tenga una dirección tal que lo aleje del mismo. En la posición destino es necesario que la suma de ambos potenciales resulte nula.

En el caso de conocer todo el entorno de trabajo y realizando una simulación del movimiento del robot a través del mismo, resulta posible construir una ruta que lleve al vehículo desde la posición inicial hasta la final. Dada la posición actual p_i , la próxima posición que debe alcanzar en un ciclo de simulación p_{i+1} resulta:

$$p_{i+1} = p_i + \delta_i \, \mathrm{J}(U(p)) \qquad \textbf{(5)}$$

Donde δ_i es un factor de escalado y $J(U(p))$ representa al jacobiano de la función potencial en el punto *p*. El factor de escalado define la longitud del segmento con origen en *pi* y final en p_{i+1}, y debe ser tal que dicho segmento esté libre de obstáculos.

El problema en este tipo de métodos deviene en la aparición de mínimos locales, es decir lugares que no son la posición destino en los cuales el potencial resulta nulo. Una situación de este tipo puede hacer que el robot quede atrapado en una posición que no sea el destino, o bien debido a la naturaleza discreta del método girar alrededor de ella. Solucionar este conflicto implica definir ciertas funciones potenciales que eviten la aparición de mínimos locales lo cual resulta arduo, si bien existen soluciones que lo aseguran en entornos donde los obstáculos están modelados mediante círculos (Rimon y Koditschek,1.988).

Otra solución para evitar caer en un mínimo local se encuentra en el uso de un algoritmo de búsqueda en grafos. Para ello se divide el entorno mediante el uso de una rejilla. Cada celda tiene almacenado un valor que indica su potencial. Un algoritmo de búsqueda utilizable es A* usándose como función de coste la función potencial. La expansión de la celda elegida se realiza mediante el recurso a las celdas vecinas. (Baturone, 1995)

2.3.5 Algoritmos de búsqueda aleatoria

Para este tipo de métodos se pueden considerar como locales puesto que solo exploran una parte del espacio de configuraciones y están basados

en movimientos aleatorios para determinar la respectiva secuencia de configuraciones para que el robot pase de la configuración inicial a la final. (Eduardo H., 2011)

a. Planeación aleatoria de trayectorias

El algoritmo de planeación aleatoria de trayectorias (Randomized Path Planner RPP) fue desarrollado por (Claude, Latombe, Jérome yBarraquand en 1991). Este algoritmo emplea movimientos Brownianos para poder salir de mínimos locales.

b. Mapas probabilísticos

Otro de los algoritmos de planificación de actualidad se conoce como “Probabilistic roadmap”, mapas probabilísticos o PRMs. Una de sus principales virtudes consiste en su eficacia en el cálculo de trayectorias de robots con muchos grados de libertad. Existen dos tipos fundamentales de mapas probabilísticos: Los PRM de consulta única y los de múltiple consulta.

En los PRM de múltiple consulta pueden coexistir varios grafos, estos métodos inician el proceso de planificación generando múltiples puntos aleatorios dentro de C_{free}. Después, a partir de éstos, se ejecuta un algoritmo de planificación local entre pares de puntos próximos entre sí. Si la maniobra devuelta presenta alguna colisión se desecha, si no se incorpora a un grafo que representa todos los puntos entre los que se ha logrado conectividad. Por último, cuando el grafo contiene un camino entre la configuración inicial y la final se obtiene la trayectoria (Figura 22).

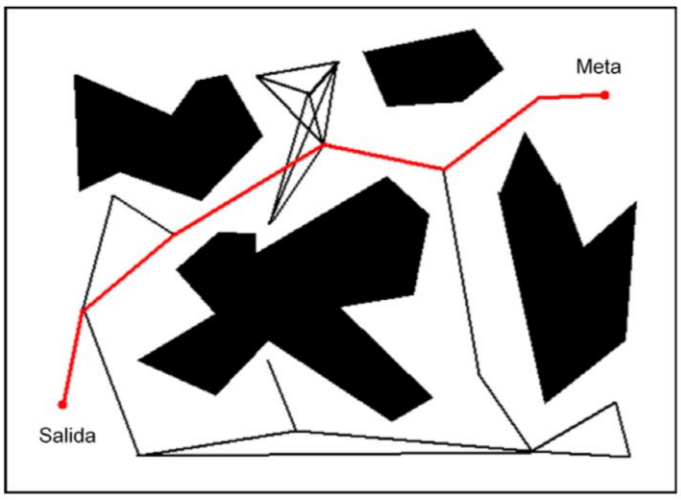


Figura 22: Grafo generado por un PRM de múltiple consulta
Fuente: (García, 2011)

La principal ventaja de los PRM reside en que no es necesario calcular los límites de los *C*-obstáculos, y por ende de C_{free}, tarea nada fácil cuando el número de dimensiones del problema es elevado, sino que basta con utilizar un algoritmo que compruebe si existe o no colisión. En la mayoría de los casos esto resulta bastante más práctico. Otra de las propiedades que hacen atractivo este método se denomina "completitud probabilística" y consiste en que la probabilidad de que un PRM encuentre un camino si es que existe, tiende a uno exponencialmente con el número de nodos agregados (Kavraki y otros, 1996).

Un planificador de consulta única mantiene un único grafo (o dos como mucho), y restringe la búsqueda a aquellas configuraciones que sean accesibles desde los nodos de este grafo (o par de grafos). Esto puede hacerse construyendo un árbol de configuraciones (grafo en el que el planificador local va agregando nuevos nodos y arcos computando trayectorias en direcciones aleatorias) desde la configuración inicial o la final hasta que se puede establecer una conexión con la otra configuración.

También pueden construirse dos árboles simultáneos, cada uno de los cuales tendrá como raíz a una de las configuraciones que se desean unir con una trayectoria, y cuyo crecimiento se detendrá en el momento en que sea posible establecer una conexión entre los dos árboles.

En el primer caso (árbol único) se dice que se hace una búsqueda unidireccional, mientras que en el segundo caso (dos árboles) se trata de una búsqueda bidireccional.

Los PRM tienen ciertas ventajas e inconvenientes con respecto a los campos de potencial. Una virtud evidente es que no hay que resolver problemas de mínimos locales, de hecho ésta es una de las causas por las que a veces este método es preferido con respecto al anterior. La desventaja más ostensible se centra en el proceso del planificador local. Éste ha de ejecutarse con elevada frecuencia y su coste computacional no es leve (mucho menos si ha de atenderse algún tipo de restricción cinemática o dinámica). Además muchos de los arcos del grafo no aportan nada a la solución, con lo que su resolución agrega tiempo superfluo al proceso.

Como resultado los PRM son planificadores relativamente lentos, pero con ellos se obtiene un grafo que proporciona más información sobre el entorno (varias trayectorias entre los puntos origen y destino además de acceso a otras regiones del plano) que otros métodos. (García, 2011)

c. Árboles de exploración rápida

Steven M. LaValle en 1998 publicó el primer artículo en el que se describe la forma básica de un algoritmo de planificación bastante prometedor bautizado con el nombre de “Rapidly Exploring Random Tree”, abreviadamente RRT Inicialmente el RRT se ideó tanto como planificador independiente, como estructura algorítmica a utilizar en otros métodos que precisan algún tipo de proceso de exploración aleatoria (mínimos locales en campos de potencial, y PRMs). Inmediatamente surgieron nuevos artículos poniendo de manifiesto las virtudes del RRT como método de planificación autónomo. En 1999 el autor del método y J. Kuffner publican una adaptación del método para sistemas con restricciones cinemáticas y dinámicas (LaValle y Kuffner, 1999). Al año siguiente, los mismos autores presentaron una mejora que consigue hacer más eficaz al método (Kuffner y LaValle, 2000).

Otros autores también han publicado diferentes adaptaciones para los distintos escenarios en los que se aplicaba (LaValle y Kuffner, 2000), e implementaciones prácticas (Xiaoshan, 2005), sus resultados competitivos (Bruce y Veloso, 2002), y evaluaciones sobre su eficacia (Ranganathan, 2003). El RRT tiene por tanto diferentes formas (LaValle, 2006), entre las que se puede distinguir la básica (la original descrita en LaValle, 1998) y sus adaptaciones (LaValle y Kuffner, 1999). En las siguientes secciones se describen las adaptaciones más relevantes.

Algoritmo RRT básico

El algoritmo RRT originario se basa en la construcción de un árbol de configuraciones que crece explorando a partir de un punto origen. Para entender el algoritmo se usarán los siguientes conceptos:

ρ = métrica definida dentro de *C*. Puede ser distancia euclídea u otra ponderación de proximidad que pueda interesar.

q_{ini} = Es la configuración inicial (en el caso de un robot que se mueve en un plano, las coordenadas *x* e *y* de un punto de referencia y la orientación del vehículo respecto a uno de los ejes del sistema de referencia).

q_{fin} = Es la configuración final que se desea alcanzar.

q_{rand} = Es una configuración aleatoria que genera el algoritmo dentro del espacio de configuraciones ($q_{rand} \in C$).

q_{near} = Es la configuración más próxima a q_{rand}, en el sentido definido por ρ de entre las existentes en un árbol.

q_{new} = Es la configuración que se va a añadir al árbol.

ε = Longitud del segmento de crecimiento. En realidad, es la distancia entre un punto del árbol y el siguiente con el que está conectado.

El objetivo original del método RRT consiste en construir un árbol de exploración que cubra uniformemente todo el espacio de configuraciones

libres de colisión (C_{free}), se desarrolla el algoritmo que se muestra (Figura 23) (LaValle, 1998); el algoritmo tiene como misión seleccionar un punto (q_{rand}) de forma aleatoria y extender hacia él el árbol de configuraciones.

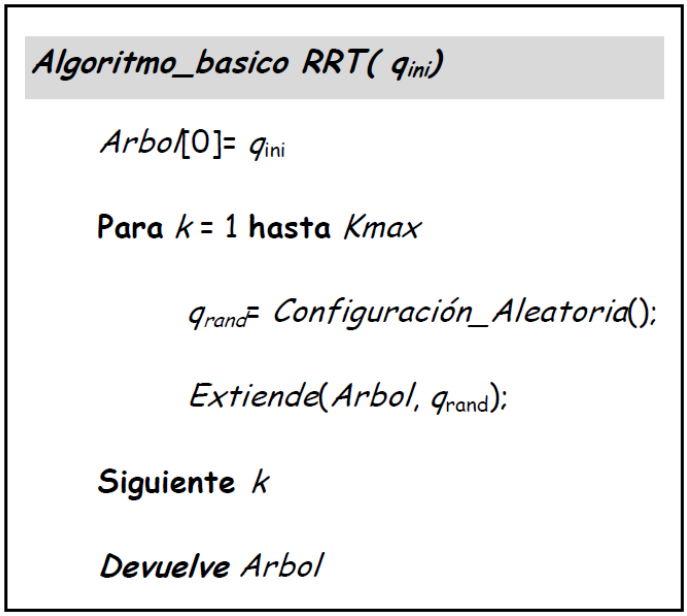


Figura 23: Algoritmo RRT básico
Fuente: (García, 2011)

Obsérvese cómo se hace uso de la función "Extiende". Dicha función tiene el cometido de ampliar el árbol en el sentido que marca q_{rand}. El esquema de dicha función se presenta (Figura 24).

El algoritmo comienza inicializando la tabla asociada al árbol con la configuración origen. Seguidamente entra en un bucle limitado por un valor *Kmax*, cuya función es finalizar el algoritmo una vez se ha realizado un número prefijado de iteraciones. Este valor se utilizará posteriormente para detener el algoritmo en el caso en que no se alcance la configuración final. Es importante resaltar que la determinación de dicho valor dependerá de las características del problema (número de obstáculos, tiempo límite del algoritmo, etc.).

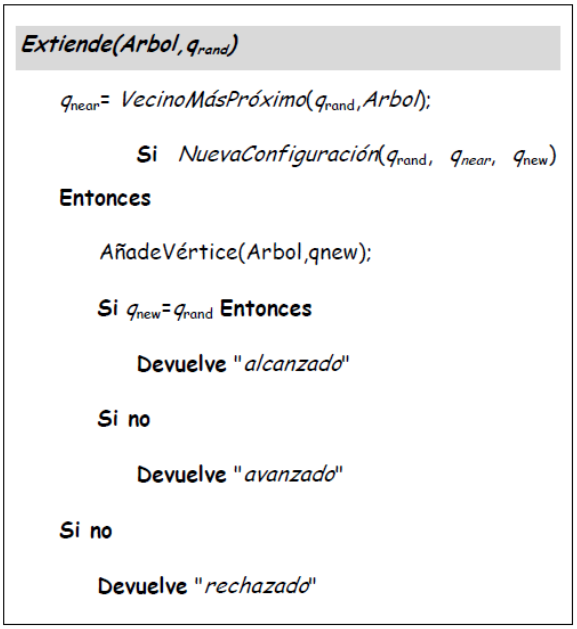


Figura 24: Algoritmo RRT función extendida
Fuente: (García, 2011)

Dentro del bucle del algoritmo RRT hay dos instrucciones. Con la primera se obtiene un punto al azar dentro del espacio de configuraciones (*C*); la segunda hace crecer el árbol en dirección a la configuración aleatoria anteriormente obtenida.

El crecimiento del árbol se consigue con la función extiende la estructura de dicha función comienza con el cálculo de q_{near}. Esto se realiza gracias a la función *VecinoMásPróximo* que aplica la métrica ρ definida anteriormente a todos los vértices del árbol, obteniendo el punto más cercano a q_{rand}. Seguidamente la función *NuevaConfiguración* calcula q_{new}, el nuevo punto a agregar mediante un salto de tamaño ε partiendo de q_{near} en dirección hacia q_{rand} (Figura 25). Para la obtención de q_{new} se tiene en cuenta si hay alguna colisión en dicho desplazamiento, devolviendo "verdadero" o "falso" según sea un movimiento posible o por el contrario, colisione con algún obstáculo.

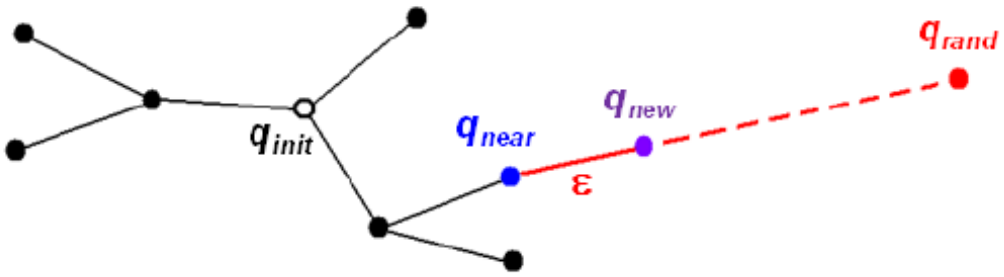


Figura 25: Esquema de expansión del RRT
Fuente: (García, 2011)

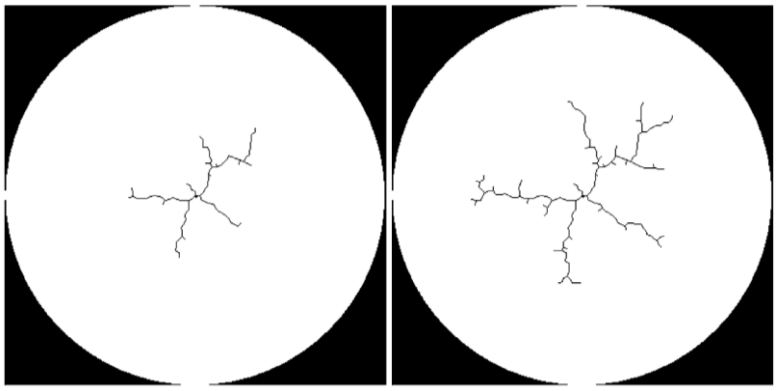
Si no se ha detectado colisión se agrega el nuevo punto al árbol distinguiendo entre dos casos. Si el punto aleatorio pertenecía a un círculo de centro q_{near} y radio ε, entonces q_{new} coincidirá con q_{rand} y por tanto, dicho punto ha sido "*alcanzado*". El algoritmo notificará entonces tal circunstancia. Si por el contrario (caso más general), no se ha producido el alcance, entonces devolverá el valor "*avanzado*". Por último en caso de que la función *NuevaConfiguración* haya advertido de la existencia de algún obstáculo en el camino que une q_{near} con q_{new}, se informa de que no ha habido nuevas ramas y el punto no es agregado al árbol.

El comportamiento de este algoritmo con respecto a otros es mejor en cuanto a la homogeneidad del espacio explorado. La naturaleza del RRT le exige al principio avanzar con más avidez hacia zonas inexploradas, pues es allí donde hay más posibilidad de que se defina q_{rand}. Esto puede comprenderse observando el crecimiento del árbol en distintos entornos. Obsérvese el ejemplo de la progresión del algoritmo representado a través de la Figura 26.

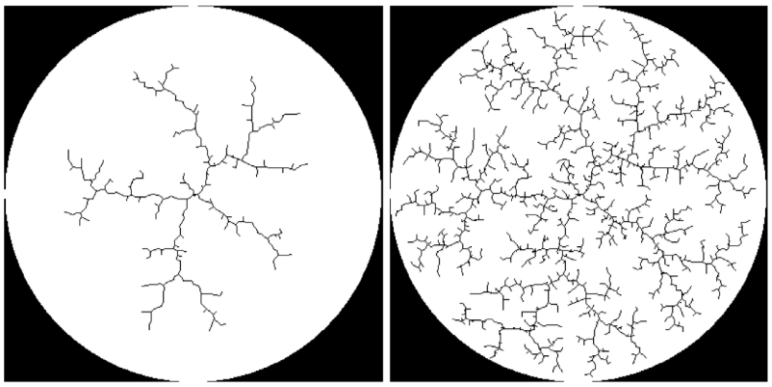
En concreto para que un punto determinado del árbol agregue una rama, es necesario que q_{rand} quede más próximo a éste que a ningún otro. Esto equivale a decir que q_{rand} pertenezca a la región de Voronoi asociada a dicho punto. Si q_{rand} se genera de forma equiprobable en todo el espacio de configuraciones y éste se divide en regiones de Voronoi, la probabilidad de

que q_{rand} pertenezca a una de estas regiones será igual al cociente entre el tamaño de la región y el del espacio de configuraciones.

En el caso de la Figura 26, el espacio libre consiste en un círculo. Esto significa que, desde el punto central, no hay ninguna preferencia en cuanto a la dirección de crecimiento. Efectivamente puede verse que las ramas iniciales del árbol surgen en direcciones aleatorias y crecen sin predominio entre ellas.



a) 100 iteraciones **b) 200 iteraciones**



c) 400 iteraciones **d) 800 iteraciones**

Figura 26: Evolución del Algoritmo RRT en un entorno circular
Fuente: (García, 2011)

Otro hecho a observar es la inexistencia de una alta densidad de ramas en el origen. La naturaleza del RRT evita precisamente esto (que es lo que ocurría en el método de escape de mínimos locales en los campos de potencial). Es más, al principio las regiones de Voronoi de los puntos periféricos del árbol son las más extensas, por lo que el crecimiento se

invierte en la prolongación de unas pocas ramas. Conforme el árbol va llenando el espacio libre, este proceso decae y las ramas aumentan su número emergiendo por doquier. Se puede contemplar que el árbol tiende a lograr una densidad uniforme de ramas sobre el espacio libre.

Obsérvese ahora las Figuras 27 a, b, y c. El escenario en este caso es cuadrado y se muestra tanto el árbol RRT como las regiones de Voronoi asociadas a sus puntos. En este caso las ramas tienden a crecer hacia las esquinas, dado que en estas direcciones hay más configuraciones que en otras y ello se traduce en una mayor probabilidad de establecer q_{rand}

En las Figuras 27 d, e, f y g se representa la progresión del algoritmo RRT sobre un espacio donde el desequilibrio en regiones de Voronoi es más acusado. Para ello se sitúa el origen del árbol RRT a un lado del escenario. En la evolución se aprecia que desde el punto origen, situado dentro del rectángulo pequeño, parte una rama que crece con acusada preponderancia con respecto a las demás. La causa es la alta frecuencia con que aparecen puntos aleatorios a la derecha del origen del árbol. Estos puntos dirigen el crecimiento en detrimento de otras ramas anexas a dicho punto origen. En este caso se ha utilizado una variante en la que $q_{rand} \in C_{free}$, con lo que este comportamiento resulta más acusado.

Al igual que en el caso de la Figura 26, tras las suficientes iteraciones, puede comprobarse que la densidad de ramas resulta razonablemente homogénea a pesar del desequilibrio inicial existente entre los dos rectángulos.

A continuación se expone una lista de las características de los algoritmos basados en RRT (LaValle, 1998), cuyos dos primeros elementos se acaban de explicar:

- La expansión de los algoritmos basados en RRT se inclina decididamente hacia los espacios inexplorados.
- La distribución de sus vértices aproxima la de la función de probabilidad utilizada.
- Son simples facilitando el análisis de comportamiento y la implementación en cualquier escenario.

- Siempre permanecen conexos, aún con pocos vértices.
- Un RRT es un módulo que puede ser implementado en otros planificadores.
- No requieren la existencia del par origen-destino, con lo que su ámbito de aplicación se diversifica.
- No requieren una definición explícita de C_{free}, sólo utiliza una función que comprueba la existencia de colisión.
- Es probabilísticamente completo (la probabilidad de encontrar un camino si existe tiende a 1 exponencialmente con el número de nodos).

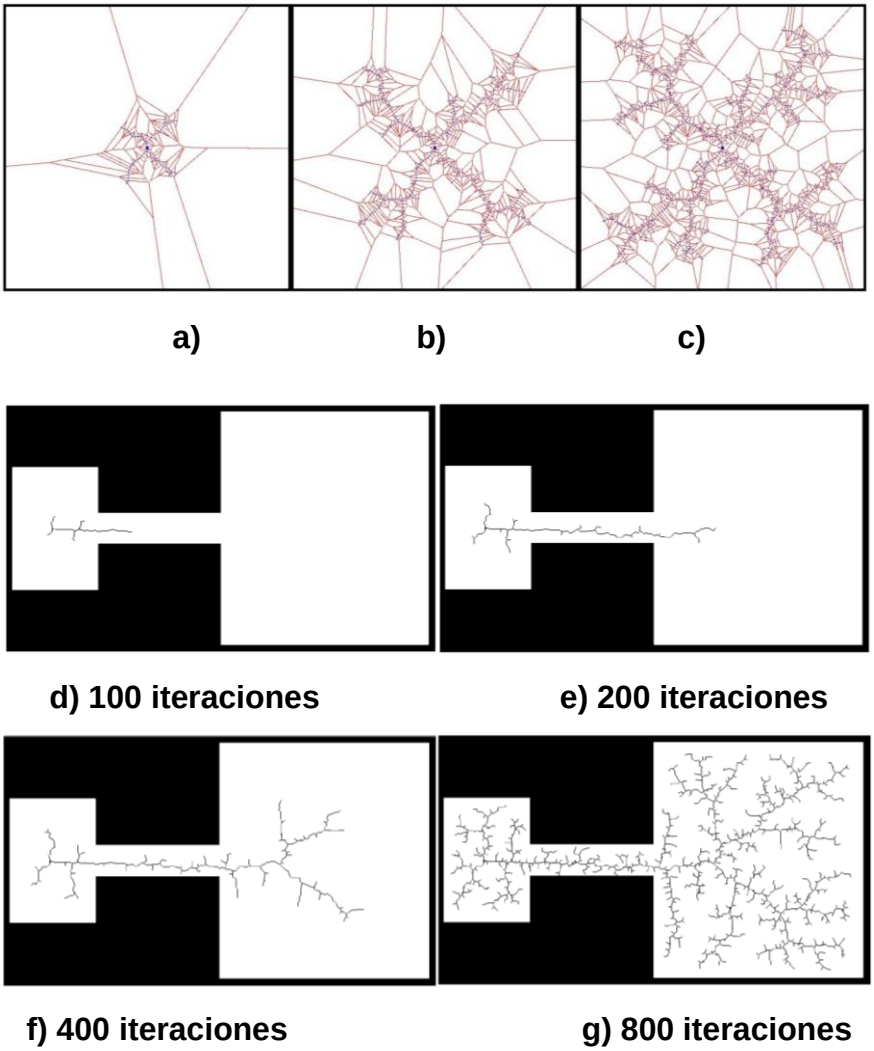


Figura 27: Evolución del Algoritmo RRT en un entorno asimétrico
Fuente: (García, 2011)

Por último, se advierte que el resultado indicado de la función *extiende*, no se utilizan en este algoritmo que sólo se ocupa para generar un árbol capaz de explorar de modo equiprobable al espacio libre. Esta peculiaridad atiende al diseño inicial del RRT como estructura de datos a implementar sobre otros algoritmos de planificación (campos de potencial y PRMs). (García, 2011)

Existen varias extensiones, así como otras mejoras del RRT que permite establecer un camino entre configuración origen y destino, una de ellas es la adaptación del algoritmo RRT – Bidireccional la que fue dominado por (LaValle y Kuffner, 2000), que se basan en la generación simultanea de dos árboles cuyas raíces son las configuraciones inicial y final. Dichos árboles crecen explorando el espacio vacío y buscándose entre sí hasta conectar, obteniendo entonces una trayectoria como resultado. El método utilizado para ello varía, dando lugar a distintos algoritmos.

2.4 Realidad virtual

La realidad virtual se podría definir como un sistema informático que genera en tiempo real representaciones de la realidad, que de hecho no son más que ilusiones ya que se trata de una realidad perceptiva sin ningún soporte físico y que únicamente se da en el interior de los ordenadores.

La simulación que hace la realidad virtual se puede referir a escenas virtuales, creando un mundo virtual que sólo existe en el ordenador de lugares u objetos que existen en la realidad. También permite capturar la voluntad implícita del usuario en sus movimientos naturales proyectándolos en el mundo virtual que estamos generando, proyectando en el mundo virtual movimientos reales. Además, permite experimentar completamente en un mundo virtual, desconectando los sentidos completamente de la realidad teniendo la sensación la persona que está dentro de que la realidad corresponde en el mundo virtual.

La realidad virtual fue ideada y desarrollada fallidamente en los años 90 para su uso específicamente en videojuegos como en la consola Virtual

Boy de Nintendo; lamentablemente el concepto de inmersión no fue logrado por las propuestas debido a la poca capacidad de procesamiento de los dispositivos de aquella época y al alto costo al usuario final de los prototipos. Recientemente en 2012 mediante la empresa Oculus (Oculus, 2015), ha traído de vuelta al escenario tecnológico la realidad virtual con su HMD Oculus Rift, en gran medida debido al desarrollo tecnológico en pantallas portátiles de alta definición, procesadores y sensores de bajo coste de producción. La versión usada en este desarrollo corresponde al Oculus Rift DK2 y se la puede (Figura 28).

Figura 28: Oculus Rift Development Kit Versión 2

Las aplicaciones que, en la actualidad la realidad virtual relaciona a actividades de la vida cotidiana son diversas. Hay que destacar: la reconstrucción de patrimonios culturales, la medicina, la simulación de cualquier entorno y la sensación de presencia. La reconstrucción del patrimonio cultural consiste en la recuperación a través de la simulación de piezas únicas de la antigüedad que han sido destruidas o se encuentran degradadas.

En algunas, a partir de unos pocos restos se pueden simular piezas enteras. Además, la realidad virtual permite mostrar la pieza en perfecto estado en diversos lugares del mundo a la vez, e incluso permite crear museos enteros con piezas virtuales. La aplicación en la medicina la encontramos en la simulación virtual del cuerpo humano. A partir de

imágenes del cuerpo humano, se puede hacer la recreación en 3D del paciente, lo que facilita la elaboración de un diagnóstico, o la simulación de operaciones en caso que sea necesario.

2.5 Motor Gráfico Unity

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies, está disponible como plataforma de desarrollo para Windows y OS X, y permite crear juegos para las plataformas: Windows, OS X, Linux, Xbox 360, PlayStation 3, PlayStation Vita, Wii, Wii U, iPad, iPhone, Android y Windows Phone y WebGL. El motor gráfico utiliza Direct3D (en Windows), OpenGL (en Mac y Linux), OpenGL ES (en Android y iOS), e interfaces propietarias (en el caso de Wii). Tiene soporte para mapeado de relieve, reflexión de mapeado, mapeado por paralaje, pantalla de espacio oclusión ambiental (SSAO), sombras dinámicas utilizando mapas de sombras, render a textura y efectos de post-procesamiento de pantalla completa en el entorno de desarrollo como se puede ver en Figura 29.

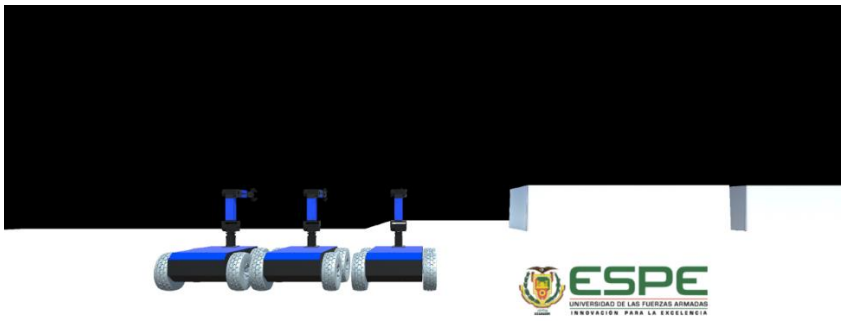


Figura 29: Entorno de desarrollo de Unity

Unity es la herramienta de creación orientada a videojuegos más popular del momento, debido a que ofrece integración nativa o por plugins con gran cantidad de dispositivos de entrada. Con el auge de la realidad

virtual, realidad aumentada y tecnologías de control inalámbricas, y una gran comunidad de desarrolladores, Unity se ha ido convirtiendo en la herramienta de desarrollo por excelencia al momento de crear experiencias visuales y videojuegos.

Entre las ventajas que ofrece Unity, también incluye el Unity Asset Server, mostrado en Figura 30, que ofrece a los desarrolladores recursos como modelos 3D, scripting, proyectos, audio, creados por la comunidad de desarrolladores de Unity para su uso gratuito y también de pago. Con esta funcionalidad el desarrollador se concentra en crear su proyecto y deja a un lado la problemática del diseño, funciones especiales o incluso inteligencia artificial aplicable en su proyecto.

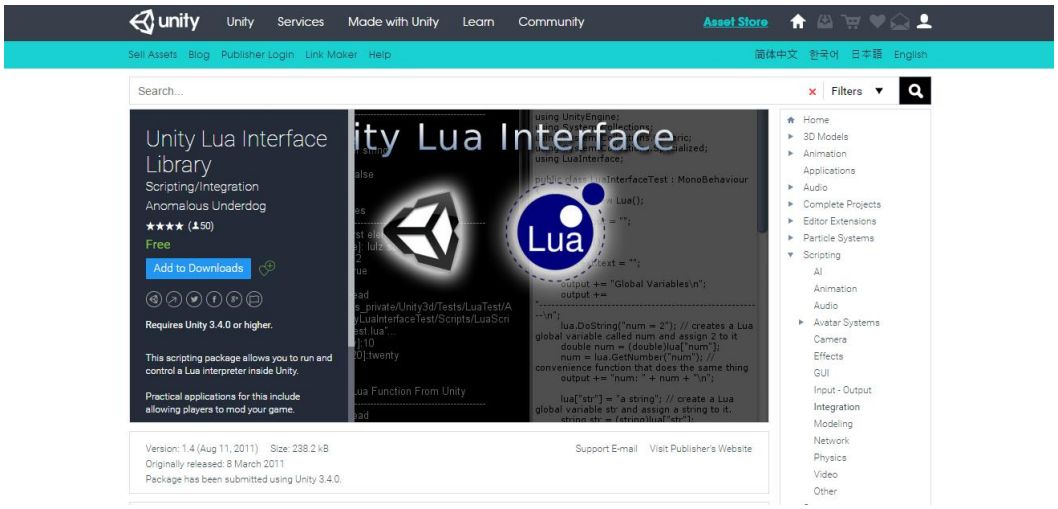


Figura 30: Interfaz de Unity Assets Store

CAPÍTULO III

3 DESARROLLO DEL ALGORITMO

3.1 Planificación de trayectorias

Como se pudo ver en el capítulo anterior, la finalidad de planificación de trayectorias es que el robot cumpla con el objetivo de búsqueda y obtención de trayectorias adecuadas, seguras y que poseen la mayor eficiencia en su desplazamiento. Es importante tener en cuenta el modelo cinemático y dinámico del robot que se va a controlar para poder llevar a cabo una buena planificación de trayectorias. Donde el usuario podrá definir las características de operación y tiempos de ejecución en los movimientos del robot dentro de la aplicación, basándose en la: precisión, restricciones acordes a las condiciones del ambiente ya sean estructurados o no.

En la Figura 31 se muestra en forma general la integración de algoritmos Path Planning y control en el desplazamiento de un robot a través de una trayectoria.

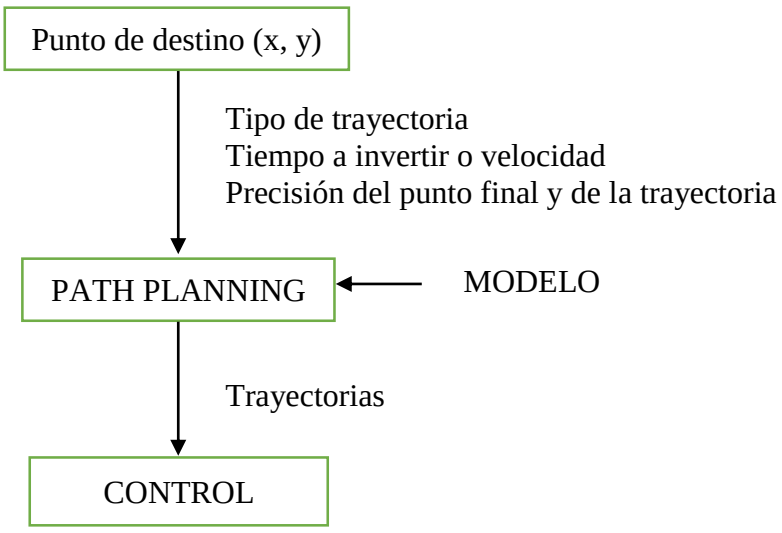


Figura 31: Diagrama de integración de algoritmos Path Planning

3.2 Navegación DistBug

La navegación Distbug asegura el destino de llegada, o indica si la trayectoria del objetivo es compleja, además es reactivo en el sentido de que depende de los datos tomado de su entorno para tomar decisiones locales, y no utiliza ninguna forma de representación interna del medio ambiente. (Márkez, Durley y Guzmán, 2011)

3.1.1 Desarrollo del algoritmo DistBug

Se presenta el Path Planning basado en el algoritmo DistBug para la construcción de trayectoria de navegación de un robot móvil, en el cual para llegar a un objetivo se usa un láser destinado a la detección de la presencia o ausencia de obstáculos en el entorno, donde los ramales de ubicación son distancias euclidianas que parten desde el punto de ubicación xi, yi posicionados en la región de origen y se trasladan hasta encontrar un obstáculo mediante el láser que registra un barrido de 180º dando paso a una nueva ubicación fuera de este; como se explica en la Figura 32, siendo la etapa final una restricción de camino junto con la medición de un intervalo de distancia como indicador de llegada a la posición final xf, yf.

```
LÁSER (Xstart,Ystart)
1       para (i=1to ANGLE do)
2          Xnew ← d*cos(ANGLE).add(Xstart)
3          Ynew ← d*sin(ANGLE).add(Ystart)
4               a ← map(Ynew,Xnew)
5        si (a==1)
6               da ← squart((Xnew)^2.add(Ynew)^2)
7        line ([Xstart Xnew],[Ystart Ynew])
8         ANGLE ← ANGLE.add(π/180)
9    RETURN HAZ_LÁSER (Xnew,Ynew)
```

Figura 32: Código de trabajo del láser

```
    RRT(qstart,qgoal)
1           ANGLE ← atan2(Ygoal,Xgoal)
2               di ← DISTANCE(qgoal,qnew)
3  PATH   mientras  (di>=cte)
4             Xnew ← d*cos(ANGLE).add(Xstart)
5               Ynew ←  d*sin(ANGLE).add(Ystart)
6               Xi  ←  Xgoal.substract(Xnew)
7               Yi  ←  Ygoal.substract (Ynew)
8            DISTANCE_EUCLIDIAN (qi)
9         RESTRICTION TRAYECTORY(Xnew,Ynew)
10        si   (a!=NULL)
11           qnew.setLine(qnear)
12        entonces
13         LÁSER(Xstart,Ystart)
14            qnew  ←  HAZ_LASER(Xnew,Ynew)
15            qnew.setLine(qnear)
16      NEWANGLE ←  atan2(qnew,qgoal)
17              di  ←  DISTANCE(qnew,qgoal)
18   returnNEW PATH
```

Figura 33: Algoritmo Path Planning basado en DistBug

A continuación, la simulación del algoritmo sin derivaciones de suavización de trayectoria, mostrado en la Figura 34:

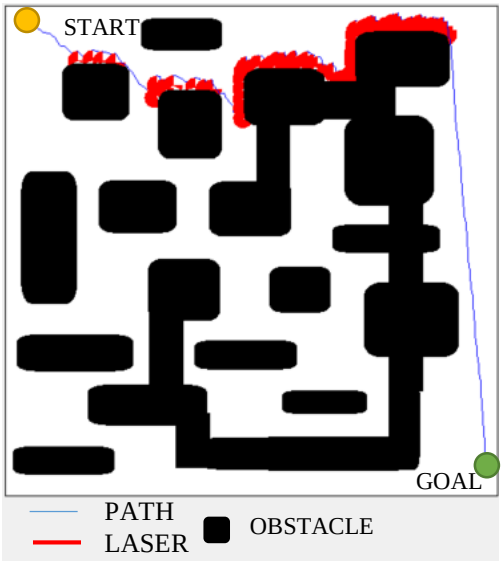


Figura 34: Representación de Path Planning

3.2 Planificación de Trayectorias por Campos Potenciales Virtuales

La planificación de trayectorias por el método del campo potencial se inspira en la propiedad física de las partículas para atraer y repeler, donde el robot es considerado como un punto de carga eléctrica dentro de un campo potencial situado en el espacio. En robótica es posible simular la navegación de cargas debido a la existencia de mínimos locales y globales dentro de una función, en este contexto el algoritmo emula un potencial, así como el destino deseado.

De este modo el robot se considera una partícula como carga eléctrica en el espacio libre que se define como un campo potencial, donde los obstáculos tienen carga eléctrica del mismo signo del robot, para obtener una fuerza de repulsión, mientras que el punto de llegada está asociado con una carga de signo opuesto el robot que resulta una fuerza de atracción, Finalmente, por medio de una gradiente de movimiento asociado al campo; la ruta se calcula con criterios de la mayor línea de potenciales en problemas de mínimos locales.

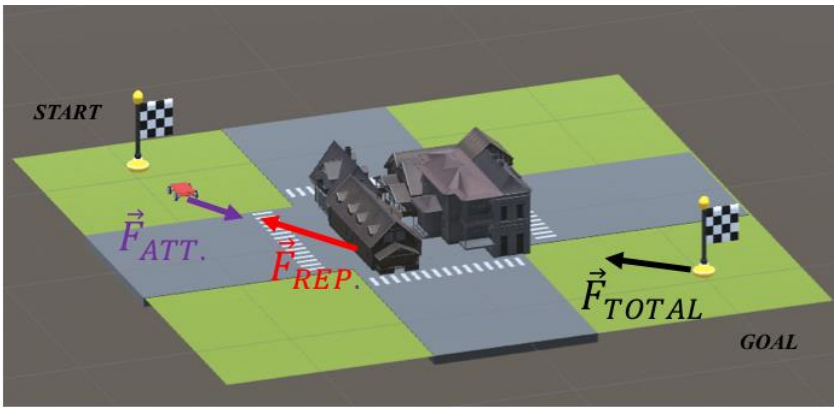


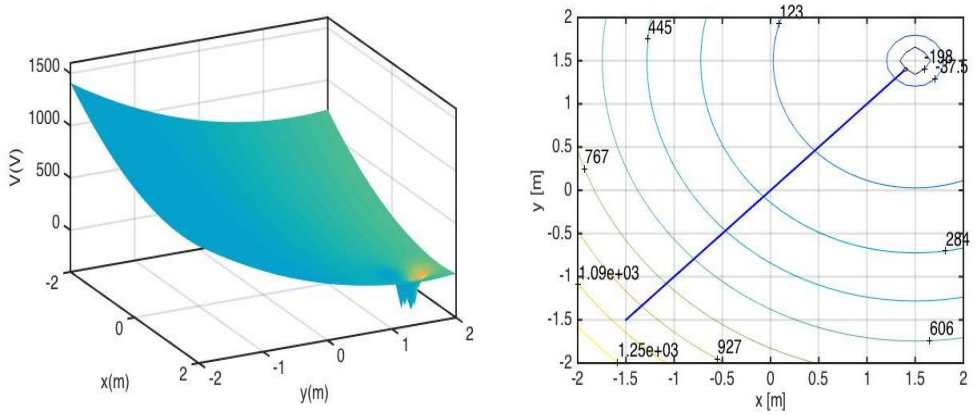
Figura 35: Medio ambiente semi-estructuradas representados en 3D

$$\vec{F}_{total} = \vec{F}_{att} + \vec{F}_{rep} \quad \textbf{(6)}$$

Además, la solución de la disminución del gradiente conduce a tres casos específicos, mismos que se muestran en las Figuras 36,37,38; donde la respuesta tiende asintóticamente a infinito debido a que cae en un pozo

de potencial, generando así una variación en el algoritmo de campos virtuales potenciales para encontrar una salida de estas condiciones críticas, por lo tanto, la pendiente del gradiente siempre busca mínimos locales de la función.

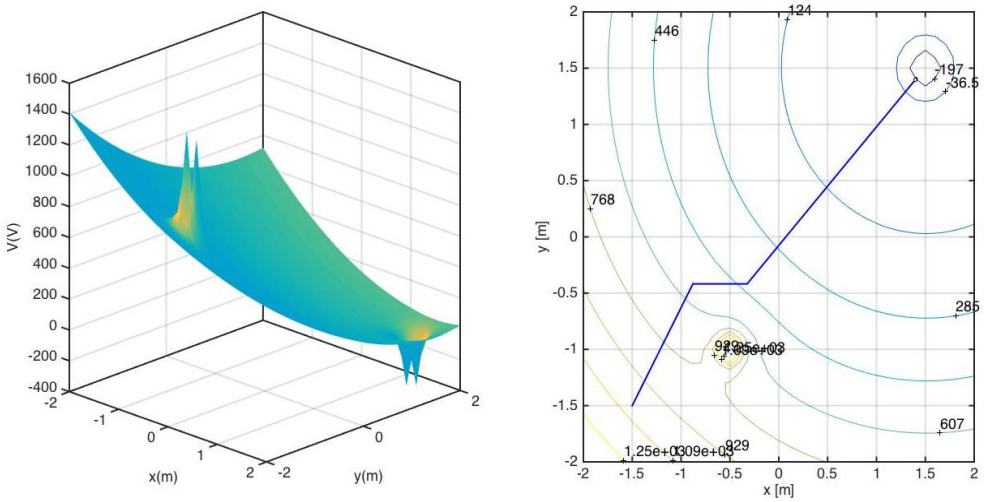
Caso I



a) **b)**

Figura 36: a) La superficie sin obstrucciones en la ruta b) líneas de potencial con ningún obstáculo en el camino

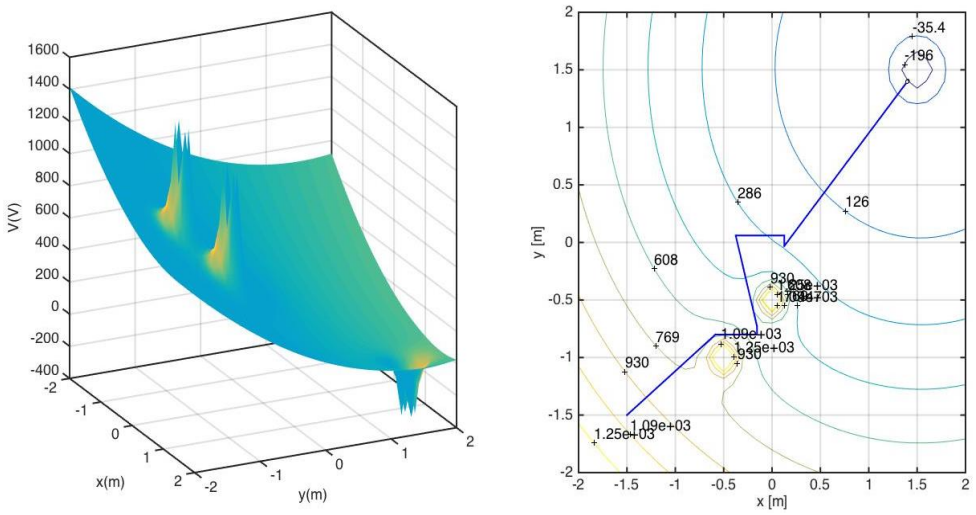
Caso II



b) **b)**

Figura 37: a) Superficie con un obstáculo en el camino, b) líneas de potencial con un obstáculo en el camino

Caso III



a) **b)**

Figura 38: a) Superficie con dos obstáculos en medio de la ruta b) posibles líneas con dos obstáculos en medio del camino

3.3 Árboles De Exploración Rápida (RRT)

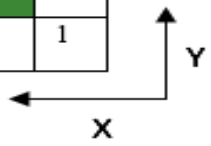
Es un algoritmo diseñado para buscar eficientemente en espacios de dimensiones grandes en forma aleatoriamente para la construcción de un árbol, este se construye incrementalmente a partir de muestras tomadas al azar de la búsqueda del espacio y es intrínsecamente sesgada a crecer hacia la gran área del problema. Son fácilmente adaptables para manejar problemas con obstáculos y limitaciones del diferencial (no holonómicos, cinéticos - dinámicos) y han sido ampliamente utilizados en la planificación de movimiento robóticos autónomos (LaValle, 1988).

La capacidad de moverse en entornos de sistemas autónomos, ya sea estructurada o no, se logra a través de una adecuada planificación técnica centrada en la robótica de ayuda específicamente como la silla de ruedas, pues asegura una navegación que evita riesgos para el usuario.

Ambiente de Mapeado: El entorno se representa mediante imágenes en dos dimensiones, que son interpretadas por una matriz binaria la cual brinda información como la posición y dimensión de obstáculos presentes, formando así un mapa con todos los elementos que componen el ambiente, esto se indica en la Figura 39. El siguiente paso es aplicar un algoritmo de detección de esquinas de Harris, donde los bordes de obstáculos presentes en el entorno se guardan en una pila, así se generan datos para que el algoritmo los procese en la planificación.

a)

	1	2	3	4	5	6	7
1	1	1	1	1	1	0	0
2	1	0	0	1	1	1	1
3	1	0	0	1	1	1	1
4	1	1	1	1	0	0	1
5	1	1	1	1	0	0	1
6	0	0	1	1	0	0	1
7	0	0	1	1	1	1	1



b)

Figura 39: Representación de un entorno a través de un mapa binario 2D con rejilla donde el valor 0 representa los obstáculos y el valor 1 representa las posiciones que el robot puede tener en los espacios abiertos de las coordenadas (x, y) del mapa

Para el posicionamiento del robot, se usa una composición de cuatro cuadrantes que simulan las dimensiones del robot, a pesar de ser tomado como un punto de superficie que se muestra en la Figura 40, este grupo se utiliza para determinar las posibles direcciones a las que pueden girar el robot.

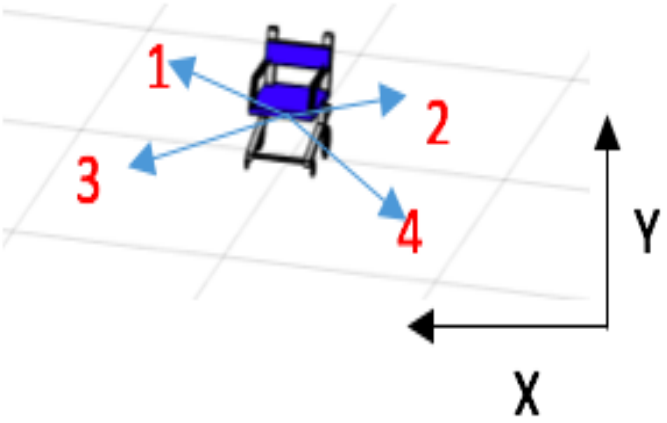


Figura 40: Representación de la superficie puntual que cubre el robot además de las rejillas que conforman sus dimensiones en los espacios libres en las coordenadas (x, y) del mapa

Algoritmo RRT, este algoritmo es una variante del RRT inspirado en la planificación de la ruta de exploración biológica de la conducta colectiva, donde el robot es considerado como una superficie puntual ubicada en las condiciones iniciales del mapa; a partir de esto se crea una línea segmentada hacia la meta cuyos datos de distancia euclidiana y el ángulo de rotación del robot colocada delante de los obstáculos se presentan en el camino hacia la meta. Una vez que los obstáculos de la zona de colisión son conocidos, se llama pila resultante de la detección de esquinas en la asignación de la etapa anterior, junto a este proceso muestra una rutina de sub-robots representado por x, que identifica y redimensiona los bordes, el robot principal ocupa una dimensión de cuatro cuadrantes evitando así cualquier futura colisión, todo esto se muestra en la Figura 41.

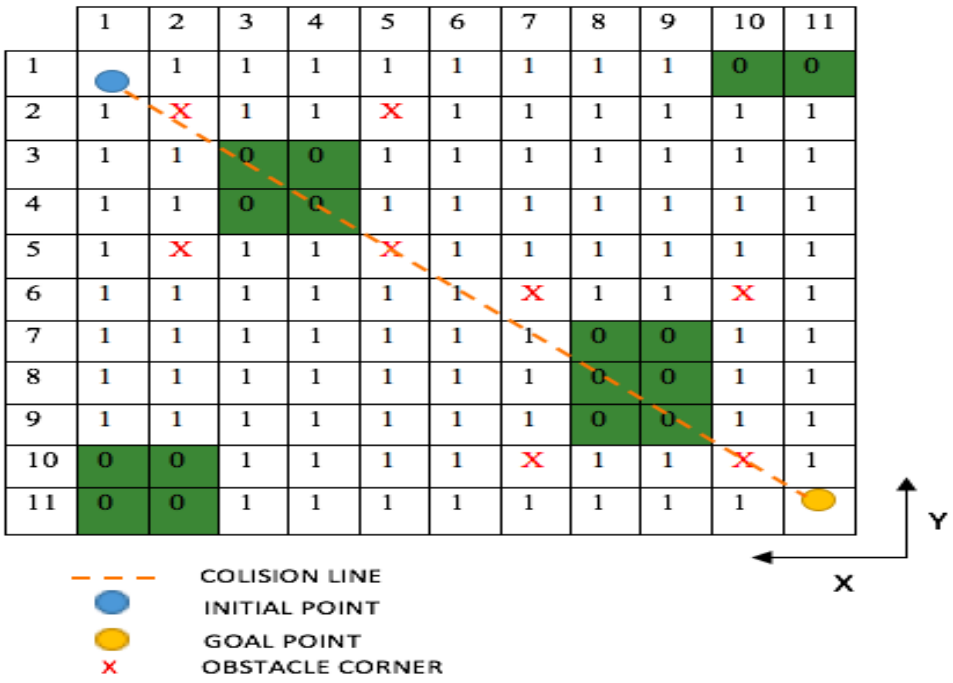


Figura 41: Representación del algoritmo de planificación RRT

El siguiente paso se crea una estructura de árbol, brindando así todas las respuestas posibles que permitirán llegar al destino; por lo cual, la pila de esquinas detectadas es para utilizar el sub-robot que llama a la rutina del principal robot para recoger cada rincón Figura 42. Este almacén se da por una estructura de repetición, que en cada paso adquiere información de dos sub-robots situada en el vértice de la figura que constituye el obstáculo, para continuar con la recopilación de la información del sub-robot situado en el borde correspondiente al vértice anterior formando así la estructura del árbol Figura 43.

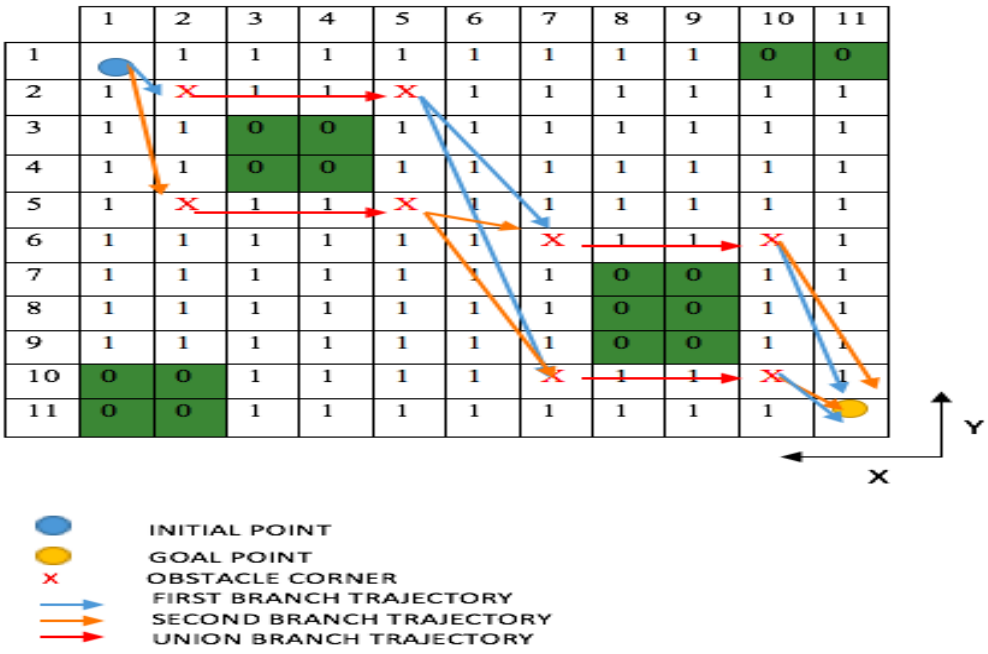


Figura 42: Posibles trayectorias en el mapa

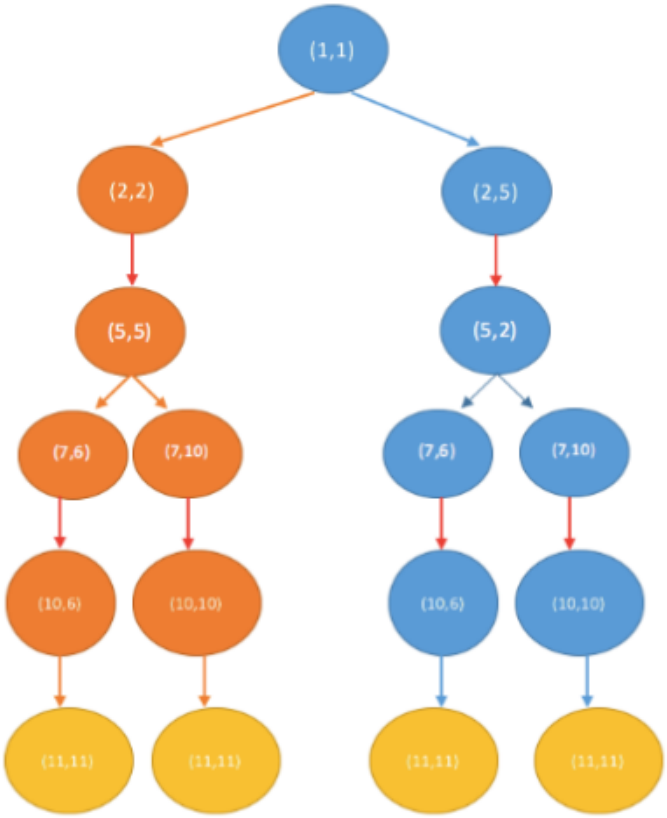


Figura 43: Creación de la estructura de árbol mediante la recopilación de datos del robot principal y sub-robot

A la hora de determinar la posición principal de la estructura de árbol, esta se puede ampliar mediante la otra posición posible respecto a la meta. En la planificación de la ruta, el algoritmo (Figura 44) determina la distancia más corta de las trayectorias respecto al objetivo deseado.

```
Function RRT( INT: inpos, INT:outpos )
1     BEGIN
2       map();
3       Obstacle_discriminator(map);
4       Corner_detection(obstacle_map);
5       Subrobotcornerredimension(corner_map);
6       Tree(inpos);
7       Euclidian_distance(inpos,outpos);
8     While(Euclidian_distance >0)
9     BEGIN
10      If obstacle==0 Then
11        addTree(actualpos);
12        Return closestcornerposition;
13        Euclidian_distance(treepos,goal);
14      Else
15        Euclidian_distance(treepos,goal);
16        Find_goal_position();
17        Eliminate_null_branch_tree();
18        addTree(goalpos);
19        Stack_pair_path(pair_tree_pos);
20        Stack_odd_path(odd_tree_pos);
21        Optimizationpath(shortestsdiststack);
22      END
23        Return Shortest_stack;
24    END
```

Figura 44: Algoritmo para determinar las distancias más cortas

El proceso de optimización se da sólo cuando la estructura de árbol está creando y todas las ramas no terminan en el destino deseado, de esta manera la información de la pila es adquirir la respuesta de todas las sucursales situadas en posiciones impares y pares, mostrada en la Figura 45; una vez que la pila se obtiene, calcula la distancia entre las posiciones y el resultado de estas sumas se compara para encontrar una solución optimizada (Figura 46).

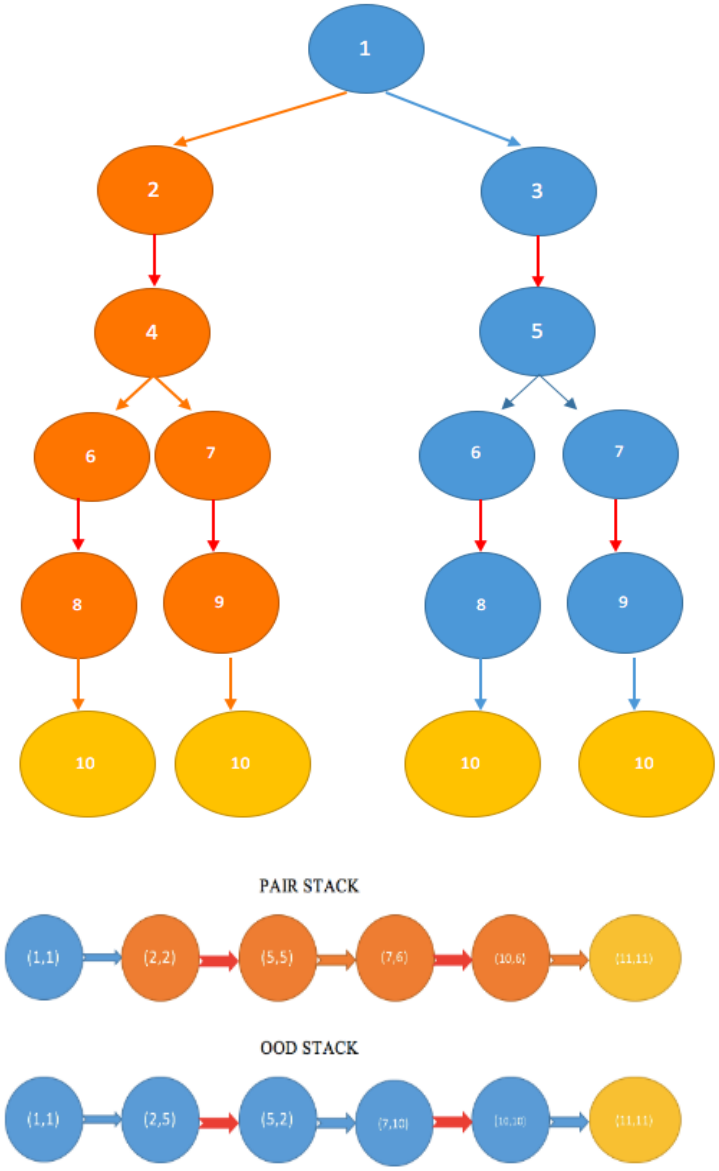
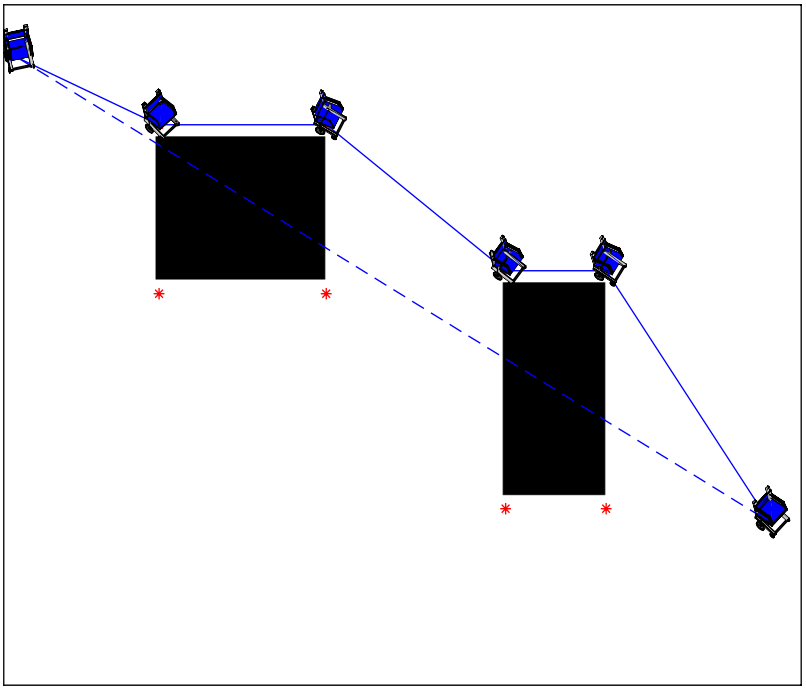


Figura 45: Asignación de valores de jerarquía para la posición de la estructura de árbol, pila de organización de las principales trayectorias al objetivo deseado. Casi la optimización de la estructura de los árboles recopilados por el algoritmo RRT

a)



b)

Figura 46: a) Medio ambiente semi-estructuradas representado en 3D b) La estructura de árbol generado por RRT

3.4 Planificación Cooperativa

El algoritmo de planificación cooperativa es el resultado de generar una trayectoria libre de colisiones para un caso especial de entorno en donde los obstáculos se ubican geométricamente de tal forma que generan un paso con mayor depresión como se muestra en la Figura 47. Para resolver este tipo de entorno o ambientes similares se presenta el algoritmo de la Figura 50, en donde el ambiente se identifica mediante imágenes, éstas son obtenidas por una cámara y procesadas para generar una matriz que contiene los pixeles que representan los obstáculos en dimensión y posición que se identifica a través de las regiones que forman el espacio de trabajo con todos los elementos que lo componen como se muestra en la Figura 48.a.

Además, el algoritmo amplió las dimensiones del obstáculo detectado para proteger la superficie de los obstáculos. La posición de los robots, se tiene en cuenta una región compuesta de cuatro direcciones donde el robot se utiliza para determinar posibles direcciones a las que puede girar.

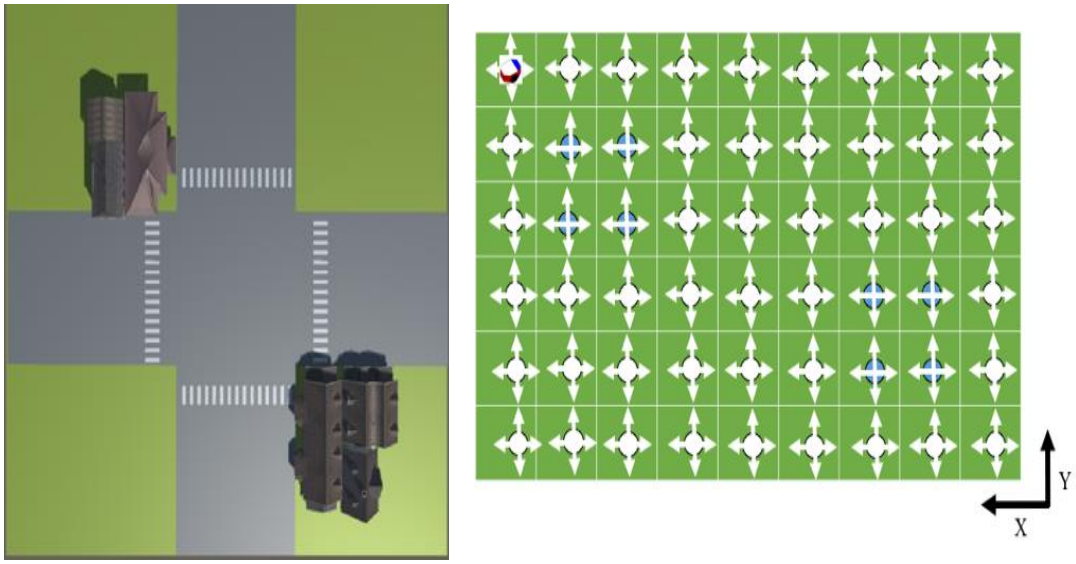
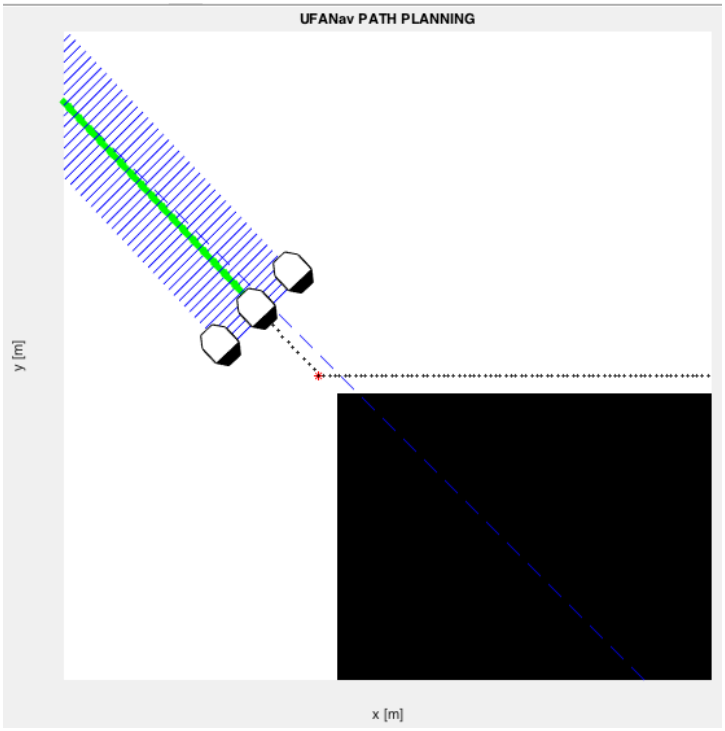
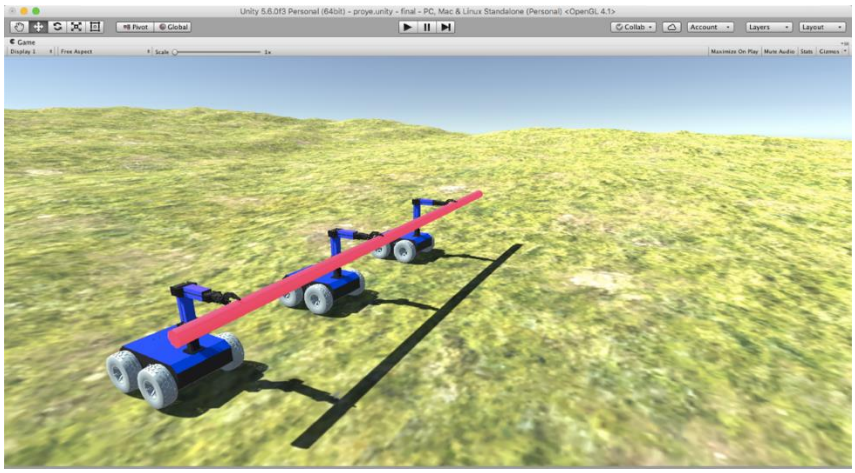


Figura 47: Representación de un entorno a través de un mapa binario con regiones donde las regiones azules representan obstáculos y las regiones blancas representan las posiciones que el robot puede tomar en regiones abiertas en el espacio de referencia del mapa

Formación cooperativa se da a partir de la necesidad de solucionar un problema para este caso general es llevar un objeto con similar distancia desde los extremos en la formación y al centro de la misma como se ve en la Figura 48.a, b.



a) ***Formación cooperativa de robots en el algoritmo de planificación***



b) Formación de tres robots en un ambiente virtual

Figura 48: Pruebas de la formación de robots para manipular un objeto

3.4.1 Algoritmo De Planificación Cooperativo

Se ha inspirado en el comportamiento de la navegación en la naturaleza, donde la formación de robots cooperativa se ubica en un punto dentro del ambiente previamente mapeado y con todos los procesos de planificación previamente presentados en la sección anterior del algoritmo RRT. Así se sitúa al robot en las coordenadas de salida, a partir de este punto un camino diagonal que representa una distancia euclidiana hacia la posición deseada es mostrado, además de los obstáculos en la trayectoria junto con los datos de distancia y posición angular los cuales sitúan a la formación frente a obstáculos posibles.

El siguiente paso es identificar y expandir el rango para evitar colisión, estas posiciones se guardan en el procedimiento de expansión de obstáculos en la capa de procesamiento de imagen, donde el robot principal de la formación ocupa una dimensión de cuatro direcciones evitando así cualquier colisión en el momento de navegación, todo esto se muestra en la Figura 49.

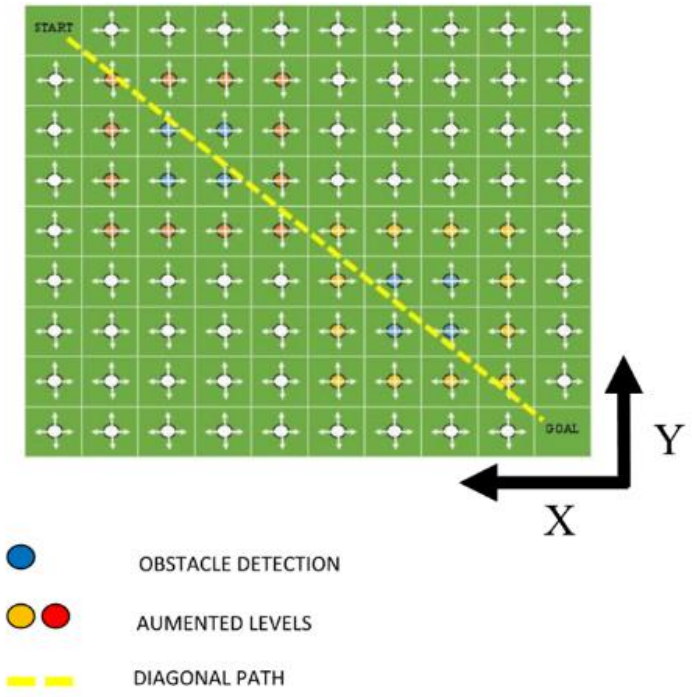


Figura 49: Representación de las capas utilizadas para la planificación en el ambiente semiestructurado

Se propone una función de detección de esquinas de la expansión representadas en el mapa como las regiones con capas amarillas y rojas alrededor de los obstáculos mostrados en una capa azul, todo esto usado por la formación cooperativa para evitar colisiones en navegación dentro del medio ambiente indicado en la Figura 49.

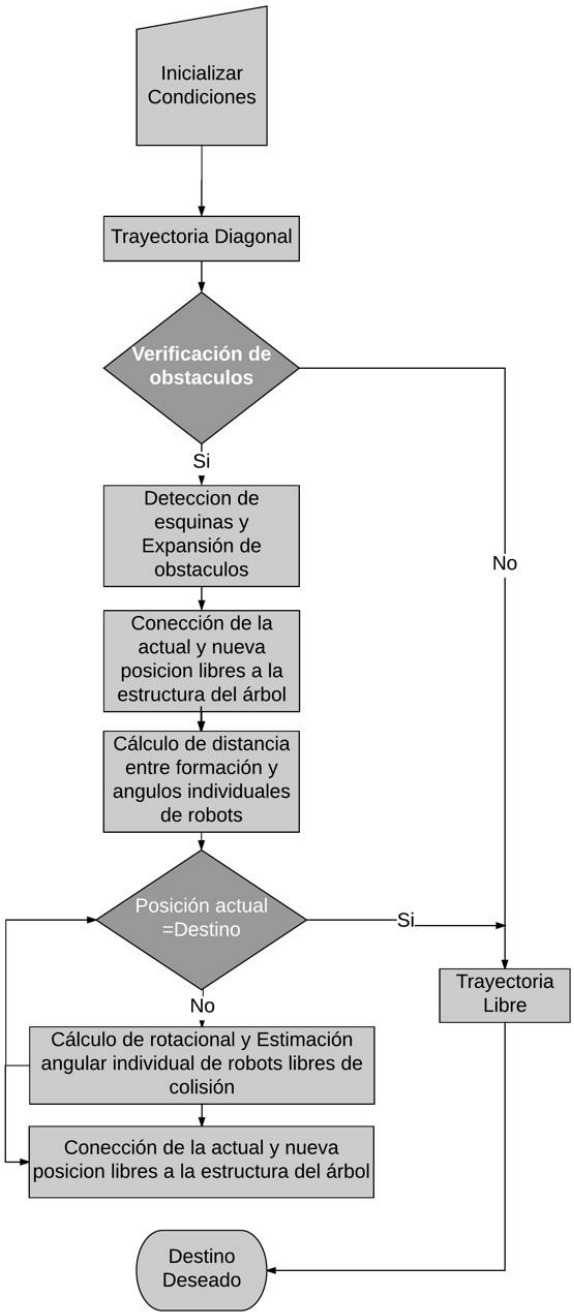


Figura 50: Diagrama de flujo del algoritmo de planificación para la formación de robots cooperativos

La ilustración de la Figura 50 representa el proceso de adquisición, localización y planificación de trayectorias a través de etapas, en las que se encargan de cargar todas las condiciones iniciales para resolver el algoritmo, así como de trayectoria diagonal calcula la dirección y la distancia hacia el objetivo en línea de vista para verificar la existencia de obstáculos.

(7)

(8)

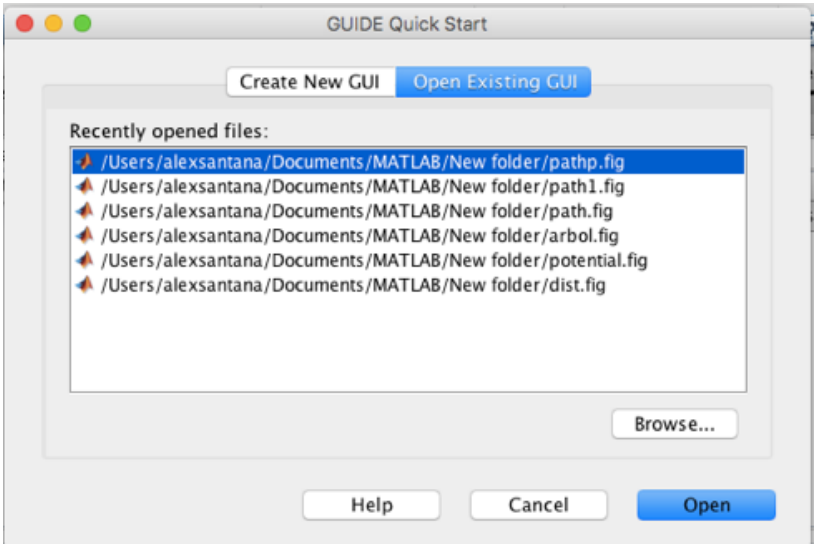
(9)

Donde puede elegir entre un camino libre de colisiones y un camino con posibles colisiones. Si este es el caso, el algoritmo se encarga de procesar la imagen que representa el obstáculo de tal manera que la estructura del árbol se carga con las posiciones actual y nueva junto con los valores de los ángulos individuales de los robots en formación, por lo que se comprueba nuevamente si se alcanzó el objetivo del sistema de referencia $R(X,Y)$, caso de que la condición no sea correcta y exista un obstáculo se recalcula mediante una matriz de rotación (7,8) un ángulo calculado en (9) adecuado para que la formación pueda pasar por la dificultad del ambiente, el proceso se repite sucesivamente hasta encontrar la solución.

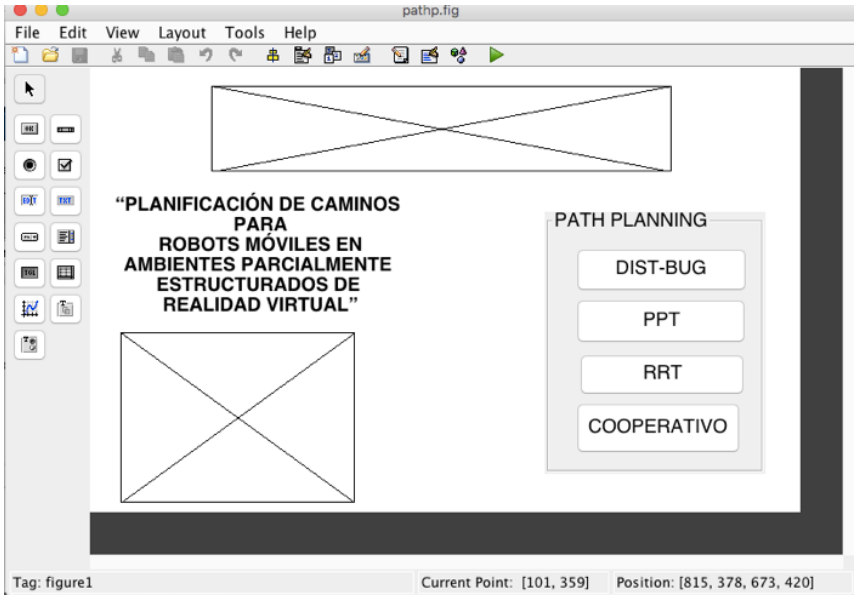
3.5 Desarrolló Interfaz Gráfica

Una vez desarrollados los algoritmos se creó una interfaz gráfica que permite la visualización de los distintas planificaciones de trayectorias propuestas en esta investigación, mediante Matlab se crea una figura que es

un editor gráfico de capas aquí se diseña un panel de la aplicación el cual se crea a partir del llamado a la función GUIDE en el Comand WIndow -> MATLAB, aquí se crea las ventanas de la aplicación posteriormente se diseña el entorno de la interfaz para luego compilarla como una ventana panel de la aplicación todo esto mostrado en las Figuras 51.a,b,c.



a) Crear ventana GUIDE de Matlab

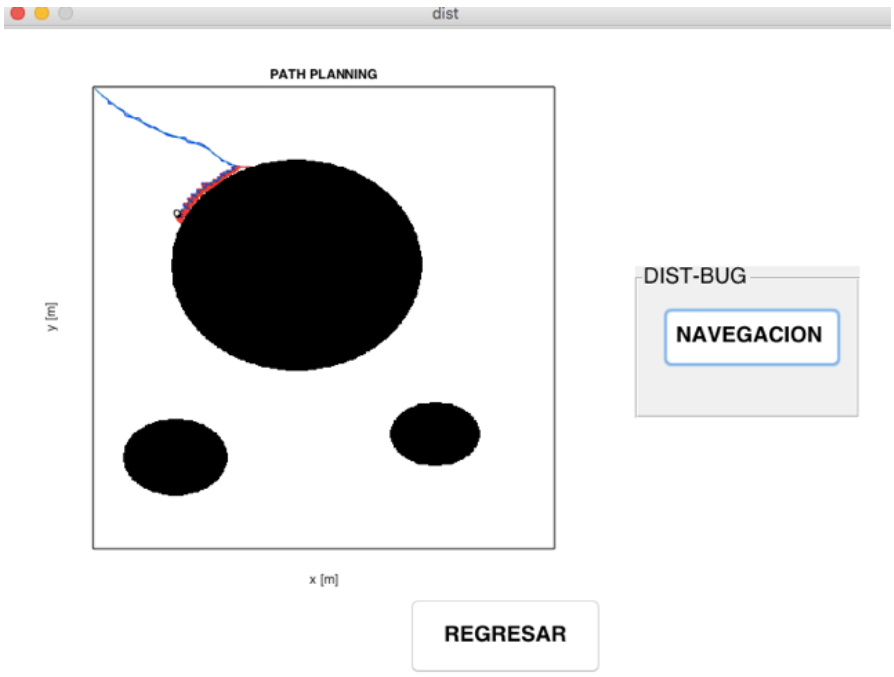


b) Diseño de la interfaz gráfica en modo GUIDE

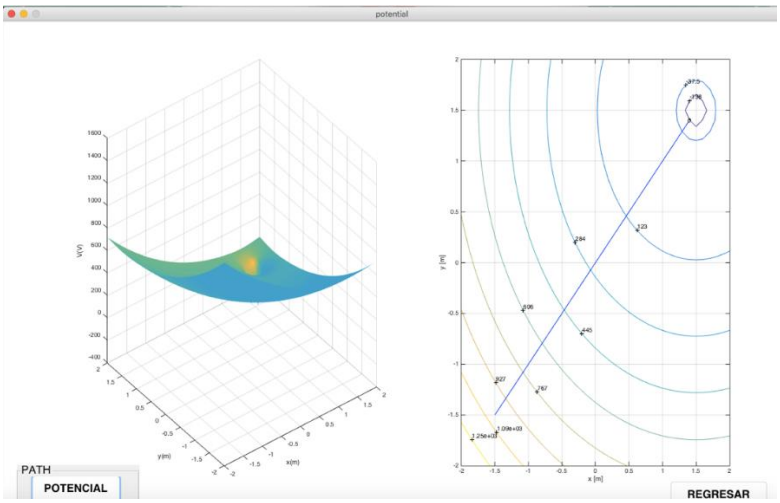
c)Panel de la aplicación diseñada en Matlab

Figura 51: Panel de la interfaz gráfica para los distintos algoritmos de planificación

Para mostrar los algoritmos se diseñaron ventanas individuales donde los scripts de cada proceso se encargan de mostrar el funcionamiento de la planificación como el caso de DISTBUG, Campos potenciales virtuales indicados en la Figura 52.a, b.



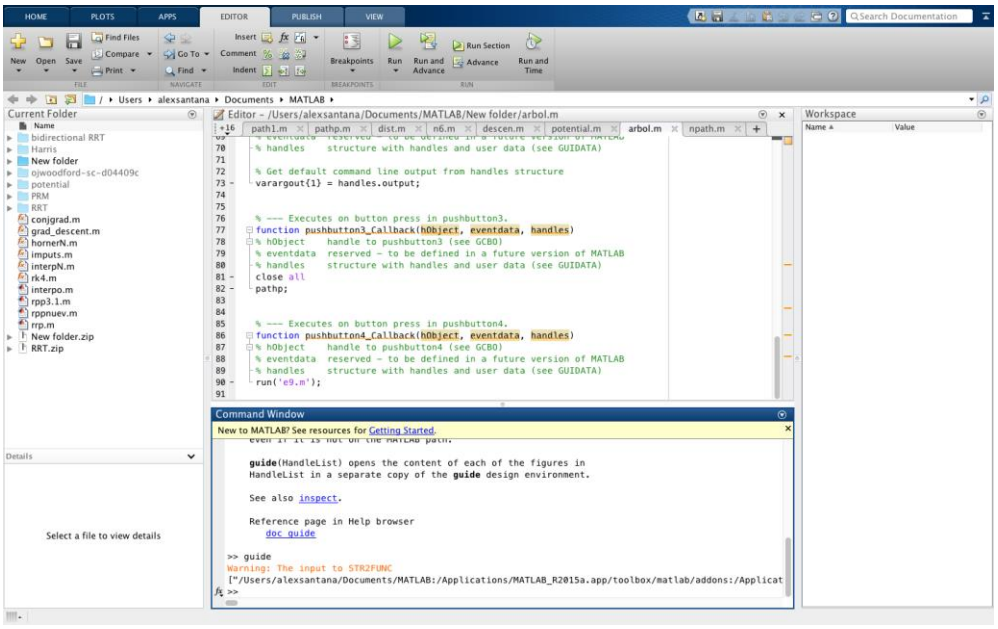
a) Algoritmo de planificación DISTBUG



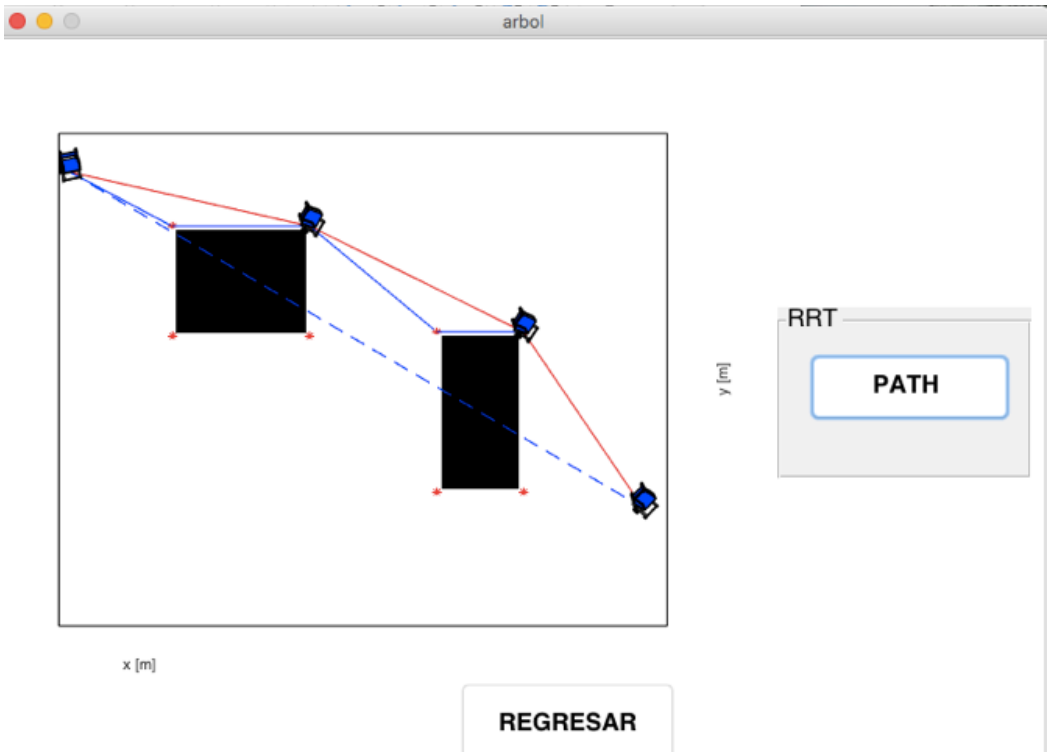
b) Algoritmo de planificación por Campos Potenciales Virtuales

Figura 52: Interfaz gráfica de algoritmo de planificación para robots

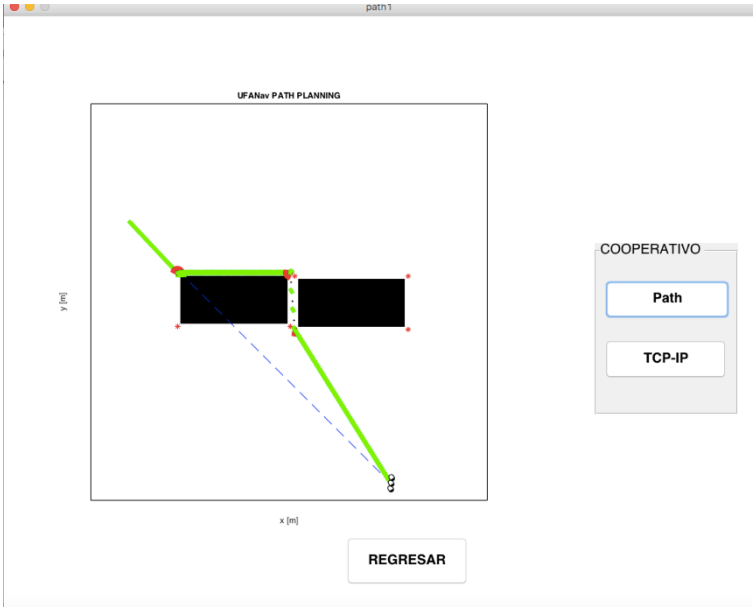
A cada ventana de la aplicación se le asigna un editor de diseño gráfico además de un script propio de ventana encargado de manipular todas las acciones de botones, capas, entre otros elementos gráficos; para mostrar los algoritmos se usa el comando run dentro de las ventanas de esta manera un script llama a otro para ser usado en la interfaz como se muestra en la Figura 53.a, b, c.



a) Script de manipulación de ventanas



a) Algoritmo de planificación RRT



b) Algoritmo de planificación Cooperativo

Figura 53: Interfaz gráfica de algoritmo de planificación para la formación de robots cooperativos

3.6 Creación De Entorno de realidad virtual

De las utilidades que tiene Unity para el usuario existen herramientas para crear entornos en campo abierto, se puede agregar un objeto de terreno a la escena seleccionada, el GameObject de tipo Terrain cumple esta función. A este se puede encontrar en el menú GameObject > 3DObject > Terrain. Mostrado en la Figura 54.

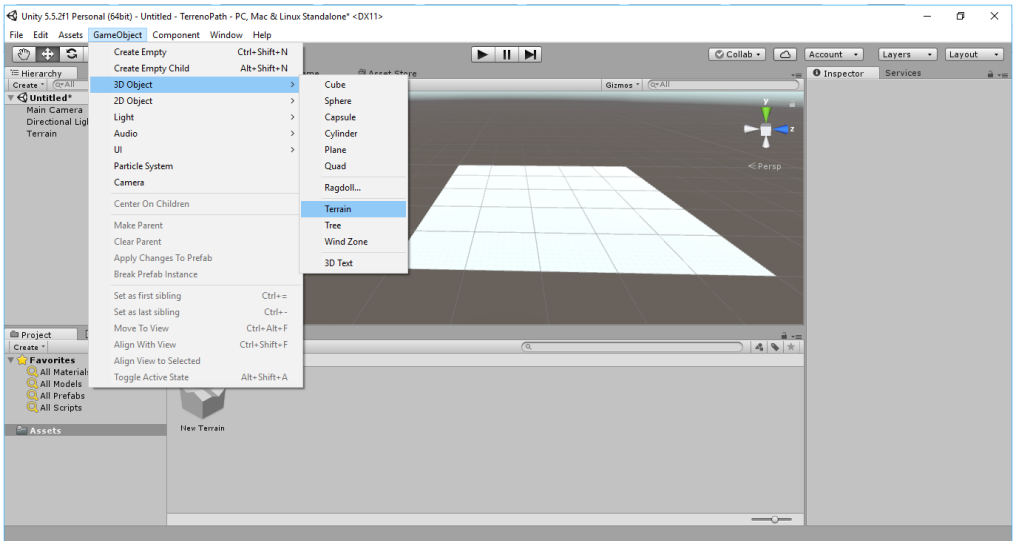


Figura 54: Espacio de terreno en Unity

De los packages de Unity usaremos el que contiene materiales, así como texturas de realismo al terreno que se creará, por lo cual para estos recursos es importante importar el Package Environment que se encuentra al ingresar en Assets > Import Package > Environment, Así como se indica en la (Figura 55).

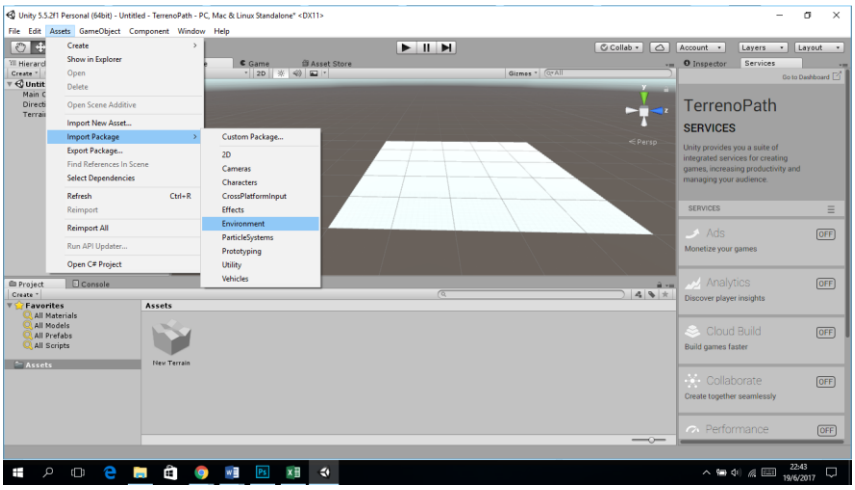


Figura 55: Importación del paquete del entorno

3.7 Protocolo De Comunicación Matlab-Unity

La integración de aplicaciones que permitan visualizar procesos de navegación con características físicas apegadas a la realidad se hace posible la unión de las principales capacidades para simulación empleadas en un computador esto gracias a lenguajes de programación matricial que facilitan el cálculo computacional de algoritmos y entornos de programación visual encargados de interpretan las condiciones de fenómenos físicos facilitando la interacción de los posibles casos, mejorando la predicción de sistemas simulados enfocados en la robótica.

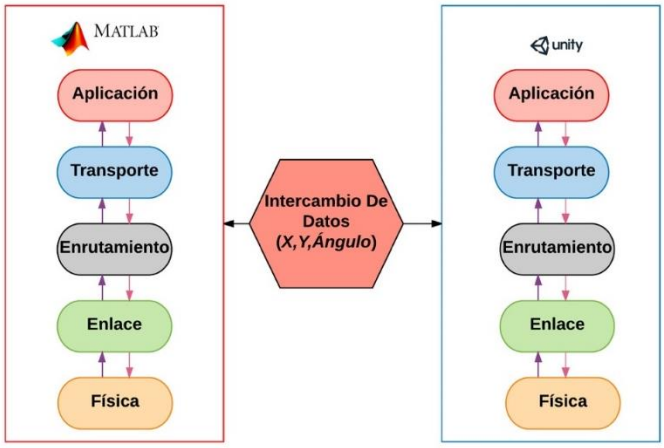


Figura 56: Importación del paquete del entorno

Para garantizar el adecuado intercambio de datos entre aplicaciones se usa un modelo jerarquizado de capas como se muestra en la Figura 56, los cuales contienen funciones encargadas de los niveles inmediatamente superiores e inferiores al relacionar resultados.

```
tcpip( INT: X, INT:Y, REAL:ÁNGULO )
1    BEGIN
2    Carga_trayectoria(x,y,angulo);
3    tcpip_Server(ip,port,'NetwrorkrRole');
4    While(1)
5      BEGIN
6      tcpip_Server(dimension_buffer);
7      fopen(tcpip_Server);
8      For  (i=0,i=dimension_tray,i++);
9          a=X[i];
10         b=Y[i];
11         e=angulo[i];
12         fwrite(a,b,e);
13       END
14         fclose(tcpip_Server);
15   END
```

Figura 57: Pseudo-Código del Script Servidor

En la aplicación se usó una arquitectura servidor – cliente de Socket mostrado en la Figura 57, donde Matlab en un script servidor a partir de la función ***tcpip*** a la cual se le asigna diferentes parámetros como : una dirección ip de servidor: '0.0.0.0' encargada de direccionar todo el flujo de información a los clientes, el puerto: 55000 debido a su disponibilidad ya que los puertos con numeraciones menores son usados para las actividades propias del sistema, además de la característica propia del Socket que se configura con 'NetworkRole' asignándole un parámetro 'Server'.

Todo esto dentro de una estructura repetitiva que permita el constante flujo de información una vez dentro de este proceso se abre el Socket (***fopen***), escritura de información (***fwrite***) y posterior cerrado de Socket (***fclose***); todo esto genera una trama de datos enviada por el protocolo hacia las aplicaciones en escucha.

```
Cliente( INT: X, INT:Y, REAL:ÁNGULO )
1     BEGIN
2       Inicializar_comunicación(.Net);
3       socket_Ready ();
4       TcpClient ();
5       Network_Stream ();
6       Stream_Reader ();
7       setup_Socket ();
8     For(i=0,i=splitArray.Length-2 >0,i++)
9     BEGIN
10      If i%3==0 Then
11        TryParse (splitArray [i], out x);
12        Return position_x;
13      Else
14        TryParse (splitArray [i], out y);
15        Return position_y;
16        i++;
17        TryParse (splitArray [i], out angulo);
18        Return position_angulo;
19        i++;
20      END
21    END
```

Figura 588: Pseudo-Código del Script Cliente

Por otro lado Unity en un script cliente interpreta los datos enviados por el servidor indicado en la Figura 58, para esto se inicializa las distintas

librerías de comunicación (.Net) así como los parámetros: dirección del equipo en este caso se usó “localhost”, puerto:55000 debe asignarse el mismo del servidor esto agiliza el intercambio de datos, cliente para el Socket (“TcpClient mySocket”), la función de lectura (“StreamReader theReader”),entre otras variables usadas para el proceso; una estructura de repetición presenta la lectura de datos a partir de la instanciación de los parámetros creados anteriormente y la función ***readSocket ()***, todo esto se almacena en un vector que se descifra a partir de operaciones matemáticas que manipulan las posiciones de los datos para diferenciar las coordenadas y ángulos enviados por el servidor.

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS

4.1 Prueba y evaluación del algoritmo DistBug

Con el fin de probar el funcionamiento del algoritmo, se realizó un ambiente de simulación propio, garantizando que este ambiente de simulación cumpla con todos los aspectos de planificación de caminos necesarios para el correcto funcionamiento y así pueda implementarse fácilmente a un robot físico con el propósito de que se evada obstáculos en tiempo real.

Las simulaciones en su totalidad se realizaron en un sistema computacional doble de 4 núcleos con 8 GB RAM usando el ambiente de simulación auto generada en un software de matemáticas. La misma fue construida teniendo en cuenta la funcionalidad del algoritmo en la generación del gráfico, el tiempo en gasto computacional, entre otros diversos factores generados durante la prueba al crear y solucionar determinada trayectoria. Como gráfica inicial se toma una entrada en forma de matriz que representa la imagen con obstáculos, representadas por regiones negras, por otro lado, el camino se personaliza con una región blanca.

Se ofrece una serie de ambientes con determinada complejidad para un montaje experimental, esto con el fin de indicar el buen funcionamiento del algoritmo, como por los conceptos expuestos en el apartado del desarrollo del algoritmo.

Se tomó un mapa arbitrario semiestructurado como se muestra en la Figura 59. En cada encuentro con obstáculos se indica un láser que permanece activo en toda la gráfica utilizado por el algoritmo de planificación de camino; el tiempo gastado en cada superficie es calculado por el

algoritmo del que se planifica y visualiza la solución. Se llevó a cabo además 4 pruebas para comprobar la funcionalidad del algoritmo.

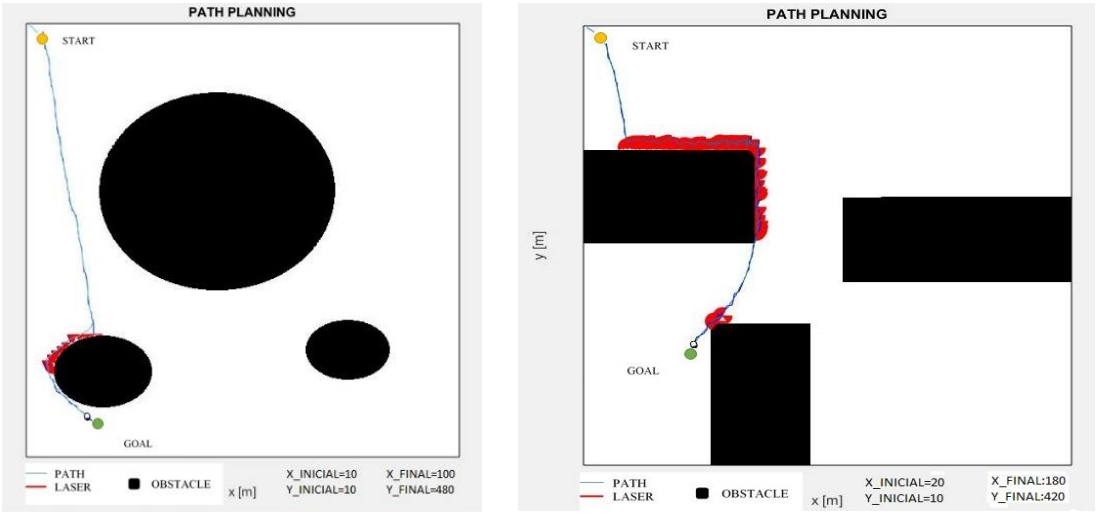
Figura 59.a la prueba se realizó en un ambiente con obstáculos de superficies circulares para garantizar que el robot se desempeñe en lugares donde no existen vértices.

Figura 59.b La segunda prueba se hizo usando tres obstáculos, esto probando la capacidad del autómata para evadir obstáculos de superficies con vértices.

Figura 59.c La tercera prueba fue de mayor dificultad en escalabilidad, pues al robot se le aumento el número de obstáculo; de esta forma se demostró la capacidad del algoritmo para resolver ambientes más complejos.

Figura 59.d La última prueba fue de obstáculos en forma de laberinto. En este caso la solución del algoritmo no es completamente eficiente por la complejidad del mismo, debido a que si el robot móvil ingresa a un espacio cerrado el bucle del algoritmo entra en un ciclo infinito a causa del rango del láser que por sus características no es eficiente.

En dicha prueba se comprobó la capacidad del algoritmo para desplazarse tanto por obstáculos pequeños, así como irregulares. Después de estas pruebas se estudió la capacidad de adaptarse del algoritmo tanto al cambio de ambientes como al de posiciones de salida y llegada al objetivo. Los distintos resultados a estas pruebas se muestran en la Figura 59.



a) **b)**

b) **d)**

Figura 59: Pruebas de simulación indicadas a), b), c), d) en ambientes semiestructurados elegidos de acuerdo a determinada complejidad para la evaluación del algoritmo

4.2 Prueba Y Resultado Del Algoritmo De Campos Potenciales Virtuales

Es así que se realizaron varios experimentos de seguimiento de trayectorias tras el control de una plataforma móvil omnidireccional. Los resultados más representativos son presentados en esta sección. La prueba experimental fue implementada para los resultados del artículo cientifico **"Navigation and Dynamic Control of Omnidirectional Platforms",** descrito en la Sección 1.4 de este documento, en un robot OMNIBEE, que apoya las respectivas velocidades angulares a cada prueba como señales de referencia de entrada (Figura 60).

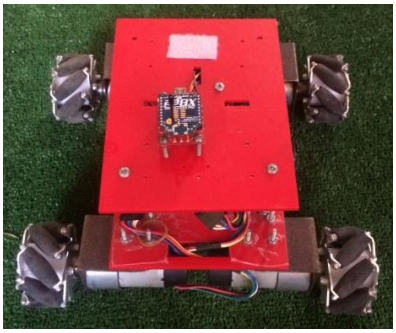


Figura 60: OMNIBEE robot móvil omnidireccional

La prueba corresponde al caso II y III (Figura 37 y Figura 38) de planificación de ruta donde se tiene un obstáculo, entonces el método proporciona la ruta deseada para el robot al moverse. La rapidez esperada del robot móvil depende del error, entonces la velocidad de referencia para la ruta en este caso se expresa como: $\left|\mathbf{v}_{\chi d}\right| = \upsilon_P / \left(1 + k\rho\right)$ [1], donde υ_P es de velocidad constante de definir por usuario, k es una constante positiva que pesa en el módulo de control y ρ el error; asimismo, la ubicación requerida se define como el punto más cercano de la ruta de acceso al robot móvil.

En el caso II de planificación de ruta de las siguientes figuras se evidencian los resultados del experimento; por ello la Figura 61 muestra el movimiento estroboscópico en el espacio; puede observarse además que el controlador propuesto funciona correctamente en la Figura 62, el error permanece cerca de cero; y en la Figura 63 se ilustra las velocidades de control.

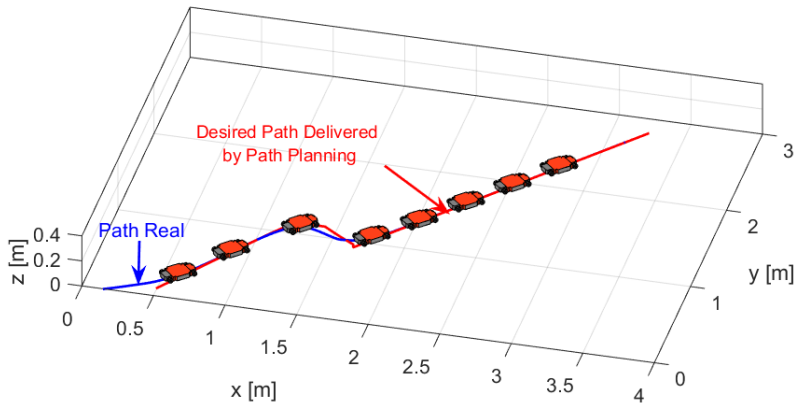


Figura 61: Movimiento estroboscópico de la plataforma móvil Omnidireccional en la ruta

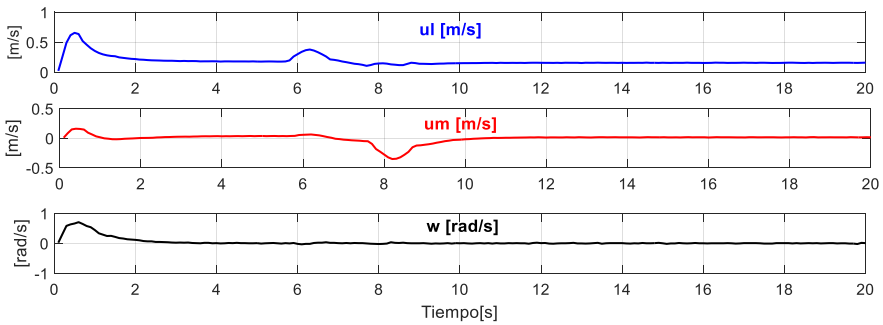


Figura 62: Control de las velocidades de la plataforma móvil

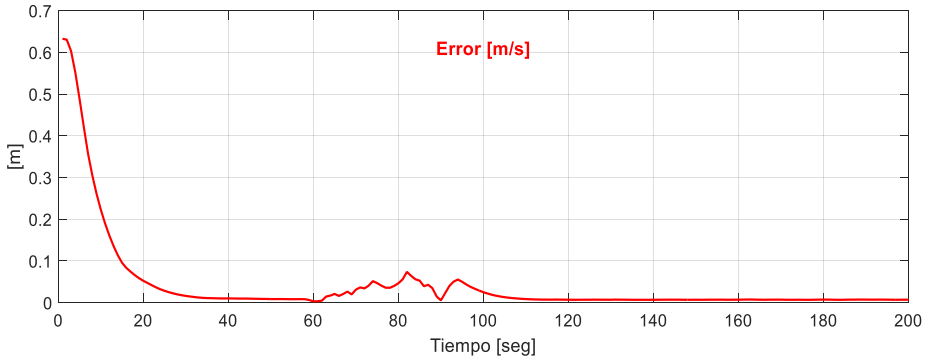


Figura 63: Distancia entre la posición de la plataforma móvil y el punto más cercano de la ruta

En el caso III de planificación de ruta de las siguientes Figuras 64,65,66. Donde las cifras muestran los resultados del experimento; La Figura 64 indica el movimiento estroboscópico en el <x, y, z> espacio; puede observarse también que el controlador propuesto funciona correctamente en la Figura 65, el error ρ(t) se mantiene cercano a cero; y la Figura 66 ilustra las velocidades de control.

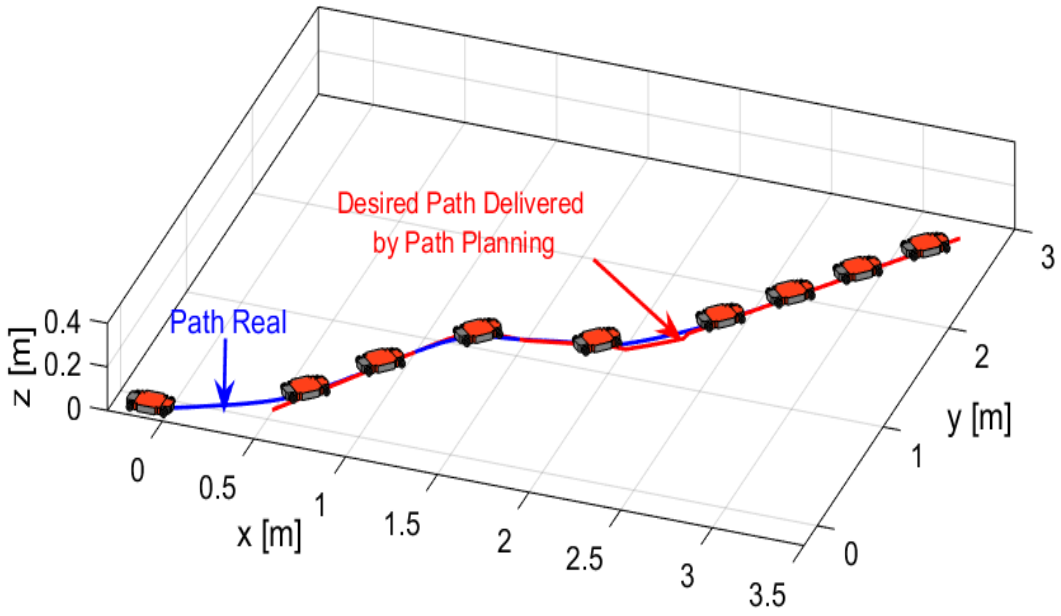


Figura 64: Movimiento estroboscópico de la plataforma móvil omnidireccional en la ruta siguiente

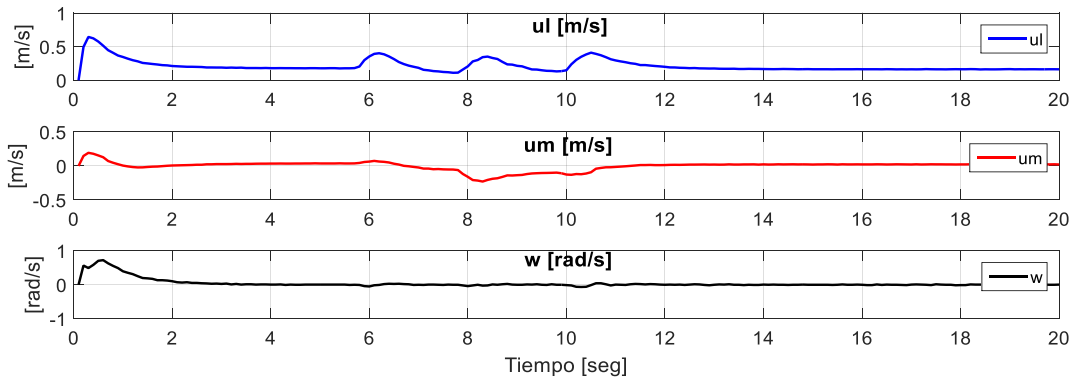


Figura 65: Control de las velocidades de la plataforma móvil

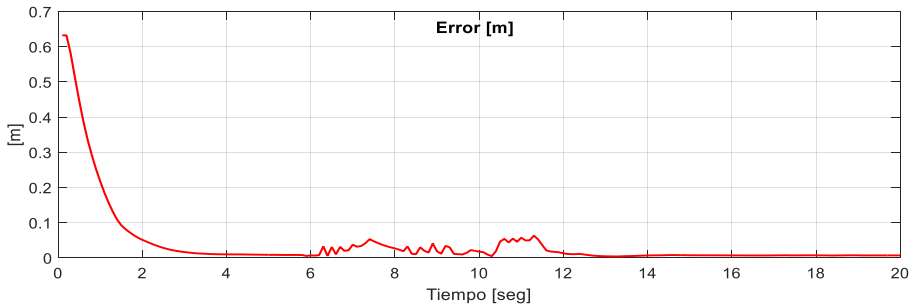


Figura 66: Distancia entre la posición de la plataforma móvil y el punto más cercano de la ruta

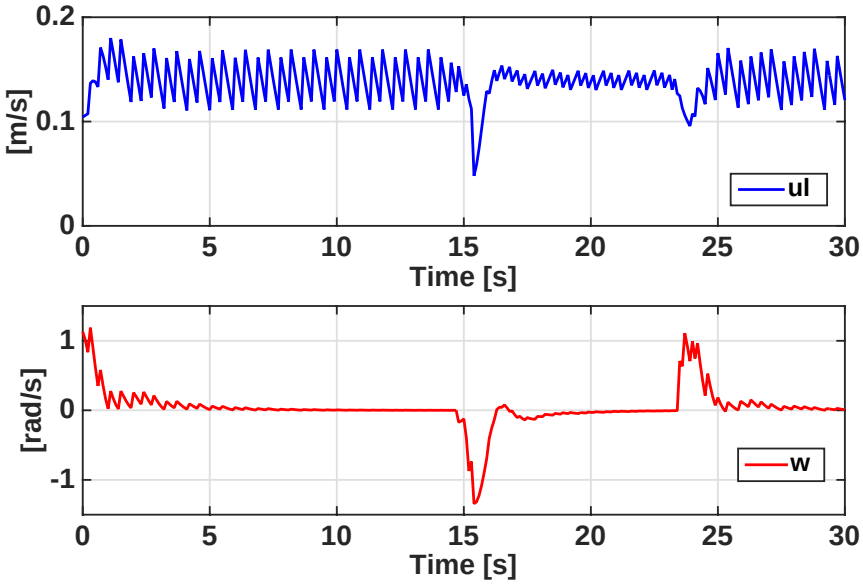
4.3 Resultados Experimentales Del Algoritmo RRT

Los resultados experimentales obtenidos, corresponden a la aplicación del algoritmo en un sistema autónomo de asistencia para personas con esclerosis lateral amiotrófica, donde una interfaz de usuario detecta la posición de mirada en un entorno, para así identificar un objetivo deseado y junto a la posición actual del usuario son datos necesarios para planificar una ruta libre de colisión mediante el algoritmo RRT (Figura 67).

Figura 67: Representación de la interacción entre la interfaz hombre-máquina y el usuario

Varios lugares deseados fueron detectados resultando una exactitud en la planificación de trayectorias con RRT del 85% de las pruebas, debido a la gran cantidad de obstáculos en formaciones que no tienen salida, así como también aumentando el tiempo y la cantidad de memoria utilizada en cada iteración; siendo un ejemplo de planificación en un ambiente interior. Una vez optimizado el RRT, la arquitectura de control vinculado a la silla de ruedas es probada para determinar la adecuada velocidad lineal y angular, de modo que la rapidez del sistema auto-contenida evita cambios bruscos cuando sigue la trayectoria, esto con el fin de eliminar algún riesgo en el usuario (Figura 68).

a) Prueba del controlador en silla de ruedas aplicado a la trayectoria dada por RRT

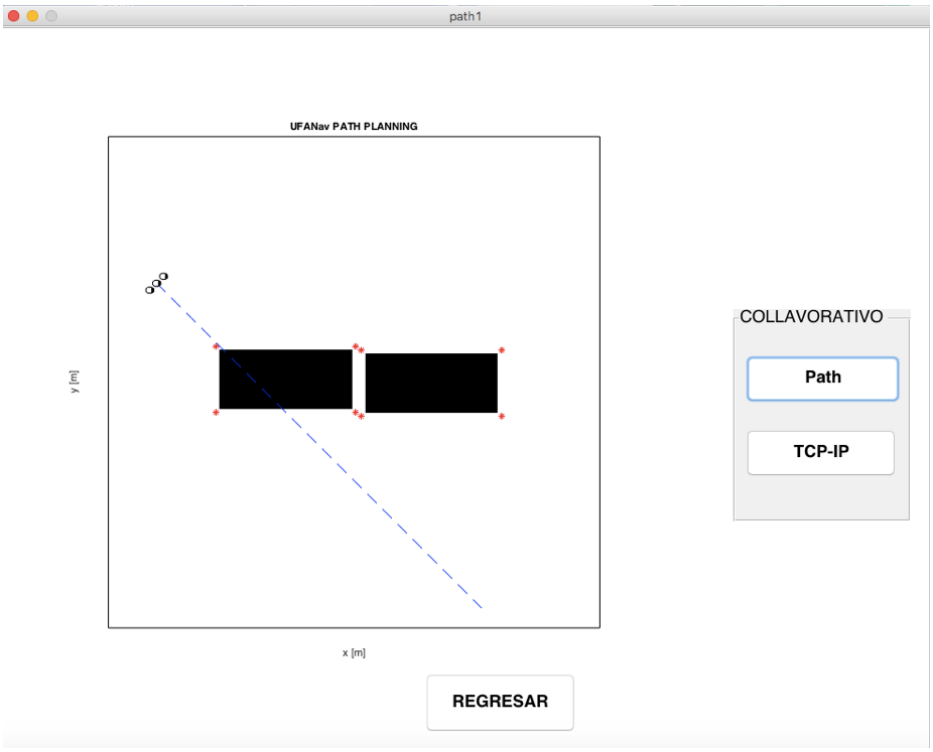


b) Velocidad angular y lineal de la trayectoria de la silla de ruedas
Figura 68: Test experimental del sistema autónomo de asistencia

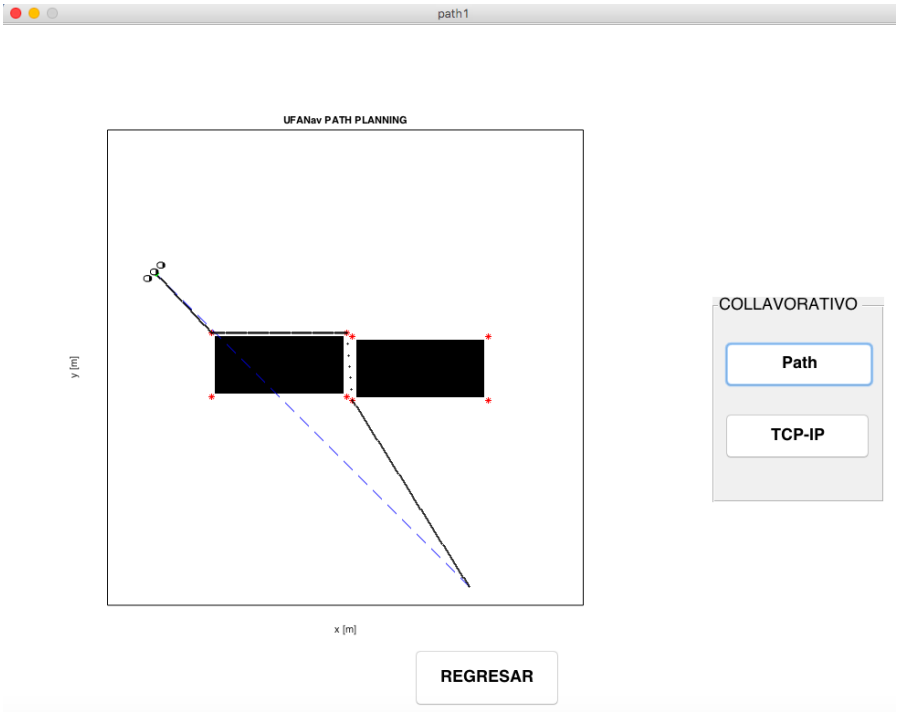
4.4 Resultado Algoritmo Cooperativo

Todo este proceso se optimiza para alcanzar la localidad de llegada deseado, en una secuencia de simulación junto con el ejemplo del movimiento estroboscópico que se muestran en las Figura 69 a, b y Figura 70 a, b.

La trayectoria de color azul entrecortada representa una distancia euclidiana al objetivo donde reconoce una amenaza en el camino, después los asteriscos de color rojo discriminan las esquinas de las amenazas presentes en el mapa de la Figura 69.a, una posible solución de puntos negros da una posible solución de una trayectoria optimizada mostrada en la Figura 69.b.



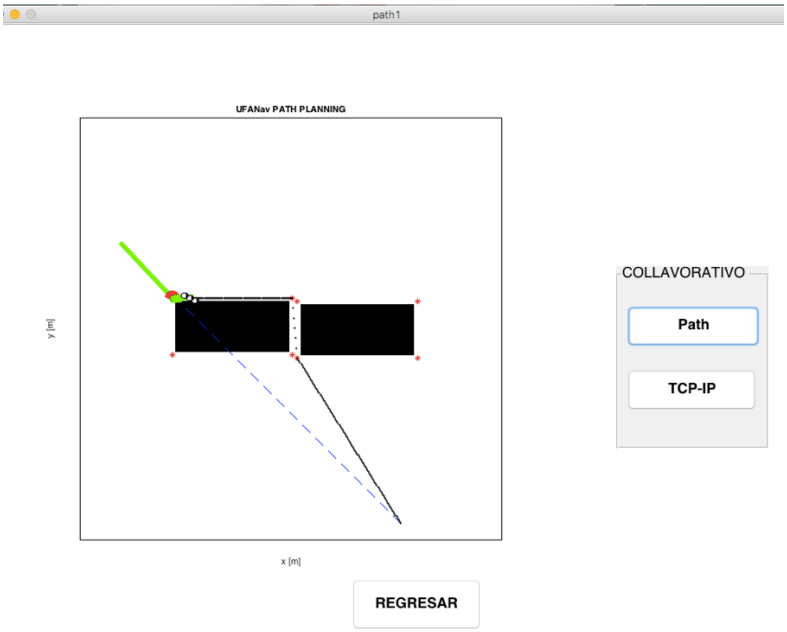
a) Formación cooperativa de robots en el entorno de simulación



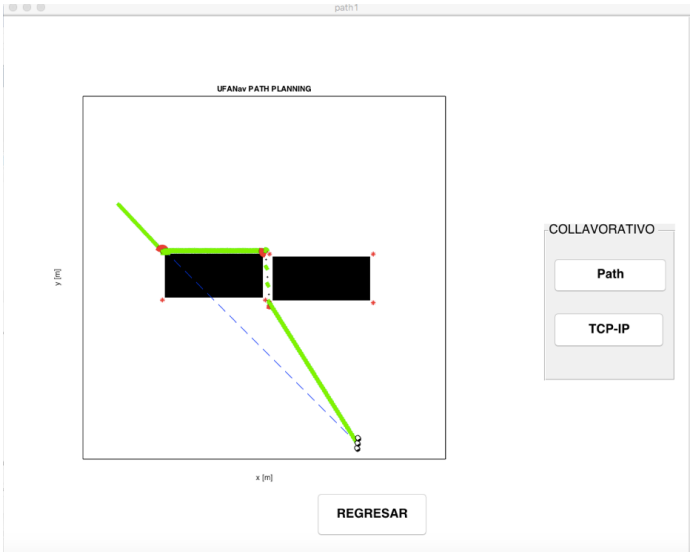
b) Construcción de las posibles soluciones hacia el objetivo deseado

Figura 69: Algoritmo de planificación de robots cooperativos

En la navegación unas saetas de color verde se despliegan con el valor del ángulo que se predijo en el algoritmo mostrado en la Figura 70.a, al comunicarse los resultados entre los robots permite a la planificación decidir un cambio de ángulo para que la formación pueda salir de la amenaza sin comprometer la geometría de los robots.



a) Prueba de la matriz de rotación para evadir obstáculos.



b) Prueba de planificación de la formación de robots en un entorno semiestructurado.

Figura 70: Test del algoritmo de planificación para la formación de robots cooperativos

Sin embargo, la alineación de robots podría variar si la amenaza no respeta la distancia al objetivo esto se puede dar por dos razones:

1) Problemas en identificar la ausencia de amenazas debido a la baja detección.
2) Las dimensiones a la que se restringen los movimientos se reducen causando una colisión, aunque la identificación de la amenaza fuese la correcta.

CAPÍTULO 5

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Concusiones

Los algoritmos de Path Planning son una herramienta funcional que permiten planificar trayectorias, tomando en cuenta las dimensiones para robots tipo uniciclo así como el menor gasto de energía al buscar el camino más corto al destino.

La comunicación entre interfaces gráficas implementadas en Matlab y Unity usando librerías de C-Sharp, reducen de forma significativa el procesamiento en algoritmos de planificación lo cual es relevante en sistemas simulados.
Para cada desempeño en los algoritmos de Path Planing propuestos, se obtuvo diferentes resultados para simulaciones y pruebas dentro de ambientes parcialmente estructurados como son:

- El algoritmo DistBug permite que un robot móvil siga el camino de un ambiente con objetos estáticos con un tamaño al menos comparable con el tamaño del robot. El algoritmo se basa en llegar al punto indicado (GOAL) con especificaciones previas de camino y posición en los objetos. El funcionamiento del algoritmo fue destacando las similitudes fundamentales entre cada uno de los algoritmos previstos, junto con sus fortalezas individuales, debilidades y dominios de problemas aplicables, mostrando mejoras utilizando un filtrado de media móvil (Sgolay Filter) de Matlab. El resultado simulado en Matlab muestra la ayuda de un sensor laser que se incluye para indicar un buen desempeño de la estrategia del planificador de caminos en los diferentes entornos.

- En el algoritmo de Campos potenciales virtuales, la programación de trayectoria virtual por campos se gestiona para generar trayectorias con evasión de obstáculos en el camino, donde el análisis dinámico y cinemático acepta como referencia las velocidades correspondientes a las velocidades angulares de cada rueda, se propuso un controlador no lineal en álgebra lineal. Es así que el rendimiento del controlador propuesto fue presentado por medio de pruebas experimentales para cada caso de programación de ruta, se observó que las trayectorias dadas para el robot alcanzan el fin planeado.

- Para el algoritmo RRT, una silla de ruedas fue equipada con una cámara de detección de posición de la mirada la cual localiza un destino deseado por el usuario. Para la planificación, se utilizó los datos de la posición deseada para generar una trayectoria y a través del controlador propuesto llego a destino con gran precisión. El concepto se caracteriza con un innovador sistema autónomo de asistencia humanitaria multi-capa que se puede escalar para su uso en el sistema de asistencia hospitalaria.

5.2 Recomendaciones

Se recomienda tener en cuenta los requisitos para sistemas de simulación, como: desarrollos en tiempo real y los recursos de unidades autónomas, dependiendo del tipo de robot que se use la velocidad a la que pueda transportarse. El cálculo empleado para realizar estos movimientos respecto a la necesidad de almacenamiento y estructura del algoritmo son parámetros esenciales para verificar el consumo dinámico de memoria respecto al tiempo en que se desarrolla la planificación. Así como en los ambientes de simulación se pueden asignar dichas memorias para comunicar programas como en este caso Unity y Matlab en sistemas operativos como Windows, pero al no poder acceder por este método en un sistema operativo OSX se

debe considerar comunicar las aplicaciones por un protocolo dentro de las capas de transporte OSI como lo es TCP-IP logrando mejorar considerablemente la escalabilidad de aplicación de los algoritmos no solo en la cantidad de trayectorias planificadas sino en la autonomía de diferentes sistemas operativos.

REFERENCIAS BIBLIOGRÁFICAS

B. Donald, L. G. (2000). Distributed manipulation of multiple. *Proceedings of IEEE International Conference in Robotics and Automation, vol. 1*, 450-457.

Bambino, I. (Recuperado el 20 de Mayo del 2017). *Una Introducción a los Robots Móviles.* Obtenido de AADECA: http://www.aadeca.org/pdf/CP_monografias/monografia_robot_movil.pdf

Barraquand, J. K.-Y. (1997). A Random Sampling Scheme for Path Planning. *Intl.\ J.\ of Robotics Research*, 759-774.

Baturone, D. D. (Recuperado el 5 de Julio del 2017). Planificación de Trayectorias para Robots Móviles. 21-52. Obtenido de http://webpersonal.uma.es/~VFMM/tesis.html#CONTENIDO

Bonabeaub, C. K. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems, vol. 30*, 85-101.

Campion, G. B.-N. (1996). Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 47-62.

Chinnaiah, M. C. (2016). An unveiling path planning algorithm with minimal sensing using embedded based robots. *2016 International Conference on Advances in Human Machine Interaction, HMI 2016*, 48-52.

Eduardo H., C. E. (2011). PROPUESTA DE UN ALGORITMO PARA LA PLANEAC ON DE TRAYECTORIAS DE ROBOTS OVILES EMPLEANDO CAMPOS POTENCIALES Y ENJAMBRES DE PAR ICULAS ACTIVAS BROWNIANAS. 137. Bogotá, Colombia. Recuperado el 20 de Abril de 2017, Obtenido de www.bdigital.unal.edu.co/4230/1/02299874.2011.pdf

Ferguson D., L. M. (2005). A Guide to Heuristic-based Path Planning. *Proceedings of International Conferences*, 10.

García, D. L. (21 de Julio de 2011). Tesis Doctoral Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos. Huelva.

González, F. (2015). PLATAFORMA PARA ROBÓTICA COOPERATIVA SOPORTADA POR TELÉFONOS INTELIGENTES ANDROID. Bogotá, Colombia.

Grace P. Cano, D. J. (Febrero de 2014). EL CONTROL DE ROBOTS COOPERATIVO UTILIZANDO FPGA´s. Quito, Ecuador.

Hachour, O. (2008). Path planning of Autonomous Mobile robot. *International Journal of Systems Applications, Engineering & Development.*, 178-190.

Jones, J. a. (1993). *Mobile Robots. Inspiration to Implementation.* USA: A.K. Peters, Ltd.

Kavraki, L. E.-C. (1994). Randomized preprocessing of configuration for fast path planning. *IEEE International Conference on Robotics and Automation (ICRA)*, 2138-2145.

Kumar, Z. D. (2002). Object closure and manipulation by multiple cooperating mobile robots. 394-399.

Latombe, J.-C. (1991). Robot motion planning. *Springer Science*, 667.

LaValle, S. M. (1998). Rapidly-Exploring Random Trees: Anew Tool for Path Planning. *TR 98-11, Computer Science Dept. Iowa State University*, 1-4.

Leena, N. S. (2014). A survey on path planning techniques for autonomous mobilerobots. *IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)*, 76-79.

Lozano, P. T. (1983). Spatial Planning: A Configuration Space Approach. *Computers, IEEE Transactions on*, 108-120.

M. Mataric, M. N. (1995). Cooperative multi-robot box-pushing. *In Proc. of the 1995 IEEE/RSJ Int. Conf. Vol. 3. IEEE Computer Society Press*, 556-561.

Macchiavello, T. (Recuperado el 22 de Abril del 2017). Obtenido de http://www.monografias.com/trabajos31/robotica/robotica.shtml

Márquez, I. D. (2011). Planificación de movimiento de un robot lego basado en la aplicación del algoritmo Dist-Bug Motion´s planning of a lego robot based on the implementation of the algorithm Dist-Bug. *Revista Avances en Sistemas e Informática*, 205-210.

Nageswara, R. (2000). Terrain Model Acquisition By Mobile Robot Teams and n-Connectivity. *Distributed Autonomous Robotic Systems 4*, 231-240.

Nourbakhsh. (2004). Introduction to Autonomous Mobile Robots. *Cambridge, MA: The MIT Press.*

Oculus. (Recuperado el 22 de Agosto del 2016). *Developer Center - Documentation and SDK*. Obtenido de https://developer.oculus.com/documentation/pcsdk/latest/

Ortigoza, R. (2007). Robots Móviles: Evolución y Estado del Arte. *Polibits.Gelbukh.Com*, 12-17.

Rekleitis Ioannis, D. G. (2000). Graph-Based Exploration using Multiple Robots. *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS) 2000*, 241-250.

Ribeiro, M. I. (2005). Obstacle Avoidance. *Institute For Systems and Robotics*, 14.

Rodrigo, L. D. (2006). Diseño y construcción de microrobots. Madrid.

Sukhatme., G. D. (2000). Landmark-based matching algorithm for cooperative mapping by autonomous robots. *Proc. 5th Int'l Symp. Distributed Autonomous Robotic Systems*, 251-260.

Zhang, C. K. (1997). Task modelling in collective robotics. *Autonomous robots Vol. 4*, 53-72.

ANEXOS

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por los señores:

ALEX MAURICIO SANTANA GALLO, ORFAIT PATRICIO ORTIZ CARVAJAL

En la ciudad de Latacunga, a los 19 días del mes de julio del 2017.

Ing. Julio Acosta
DIRECTOR DEL PROYECTO

Aprobado por:

Ing. Franklin Silva
DIRECTOR DE CARRERA

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO