SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Evaluación del algoritmo Theta* para planeación de trayectorias

presentada por

Ing. Ana Monserrat Rojas Fernández

como requisito para la obtención del grado de

Maestra en Ciencias de la Computación

Director de tesis

Dr. Dante Mújica Vargas

Cuernavaca, Morelos, México. Mayo de 2019.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

"2019, Año del Caudillo del Sur, Emiliano Zapata"

Cuernavaca, Morelos a 05 de marzo del 2019
OFICIO No. DCC/009/2019

Asunto: **Aceptación de documento de tesis**

DR. GERARDO V. GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **Ing. Ana Monserrat Rojas Fernández**, con número de control M15CE092, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis profesional titulado "**Evaluación del algoritmo Theta* para planeación de trayectorias**" y hemos encontrado que se han realizado todas las correcciones y observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DIRECTOR DE TESIS

Dr. Dante Mújica Vargas
Doctor en Comunicaciones y Electrónica
09131756

REVISOR 1

Dr. Manuel Mejía Lavalle
Doctor en Ciencias Computacionales
8342472

REVISOR 2

M.C. Gerardo Reyes Salgado
Maestro en Ciencias de la Computación
2493370

C.p. M.T.I. María Elena Gómez Torres - Jefa del Departamento de Servicios Escolares.
Estudiante
Expediente

NACS/lmz

"2019, Año del Caudillo del Sur, Emiliano Zapata"

Cuernavaca, Mor., 6 de marzo de 2019
OFICIO No. SAC/116/2019

Asunto: Autorización de impresión de tesis

ING. ANA MONSERRAT ROJAS FERNÁNDEZ
CANDIDATA AL GRADO DE MAESTRA EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Evaluación de algoritmo Theta* para planeación de trayectorias"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y tecnología al servicio de México"

SEP TecNM
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

C.p. M.T.I. María Elena Gómez Torres .- Jefa del Departamento de Servicios Escolares.
Expediente

GVGR/mcr

Dedicatorias

Dedico está tesis a mis papás Olivia y Mauricio por siempre impulsarme a seguir adelante, a mi hermano Sebastian por apoyarme en mis decisiones y a mi hermana Marisol que desde el cielo me cuida y me dice a donde ir.

A mi esposo Josue por su amor y su apoyo incondicional.

Agradecimientos

Agradezco a Dios por darme la vida y permitirme vivir esta experiencia.

A mis papás Mauricio y Olivia por hacer de mi la mujer que soy, por ser un ejemplo para mi hermano y para mí, pero sobre todo por su cariño. A mi hermano Sebastian por querer construir metas conmigo, a mi hermana Marisol que desde el cielo me cuida y me guía.

A mi esposo y mejor amigo Josue, por crecer a mi lado y estar presente apoyándome y alentándome en cada uno de mis sueños.

A mi tío Mere (Q.E.P.D.) por regalarme momentos de alegría y aprendizaje que nunca olvidare.

Al Dr. Marco Antonio Oliver (Q.E.P.D.) por introducirme en el camino de la investigación y por estar pendiente de mí.

Al Dr. Dante Mújica Vargas, por su paciencia y conocimientos que me guiaron en este camino de formación como estudiante e investigador, pero sobre todo por su amistad, su confianza y sus palabras de aliento.

A los revisores: Dr. Manuel Mejía Lavalle y Dr. Gerardo Reyes Salgado por su tiempo, observaciones y sugerencias para la mejora del presente trabajo.

A Joshua por ser un sticker en mi vida, por tu amistad, por los secretos y momentos de alegría compartidos.

A mis personas favoritas del clan: Diana por su amistad, su confidencialidad, sus enseñanzas y los momentos de diversión que compartimos este tiempo. A Mildred y Luis por abrirme las puertas de su casa y hacerme reír con sus ocurrencias.

Al Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) por permitirme realizar los estudios de posgrado.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo proporcionado para la realización de los estudios de maestría.

Resumen

La planeación de trayectorias y el seguimiento de rutas, son actividades de la robótica que permiten generar y seguir de manera autónoma el mejor camino de un punto inicio a un punto meta. La implementación de estos algoritmos puede llevarse a cabo a través de herramientas como el Sistema Operativo Robótico (*Robot Operating System*, ROS), ya que proporciona elementos necesarios para el desarrollo e implementación de aplicaciones robóticas.

En este documento de tesis, se presenta la implementación de un algoritmo de planeación de trayectorias en ROS y su comparativa con los algoritmos Dijkstra y A*. En el estado del arte se han propuesto diferentes métodos para la planeación y generación de trayectorias para los robots móviles, los cuales se basan en grafos, probabilidad, basados en optimización entre otros. Sin embargo, algunos algoritmos carecen de fácil aplicación o entendimiento, generan trayectorias rectilíneas, es decir, que carecen de realismo provocando que el robot realice un esfuerzo extra al seguir una trayectoria, todo esto con una ineficiencia con respecto al tiempo de ejecución o con la longitud de la trayectoria obtenida.

Para esta investigación, se contempló el uso de un algoritmo de planificación de trayectorias más sofisticado llamado Theta*, con el objeto de lograr trayectorias cortas con un tiempo de ejecución aceptable. Las pruebas realizadas a nivel simulación se implementaron con un robot de configuración diferencial, en cinco mapas diferentes, y se evaluaron veintisiete trayectorias con métricas de tiempo y distancia, donde el algoritmo Theta* generó y llegó a la meta en el 100 % de los casos, a diferencia de los algoritmos Dijkstra que generó el 84 % de las trayectorias y llegó a la meta el 58 % y de A* que generó el 97 % de las trayectorias y llegó a la meta en el 80 % de los casos.

Abstract

The path planning and route tracking, they are activities of robotics that allow to generate and follow autonomously the best way from a starting point to a goal point. The implementation of these algorithms can be carried out through tools such as the Robotic Operating System (ROS), since it provides necessary elements for the development and implementation of robotic applications.

In this thesis document, we present the implementation of an algorithm for path planning in ROS and its comparison with the Dijkstra and A* algorithms. In the state of the art different methods for planning and generating trajectories for mobile robots, which are based on graphs, probability, based on optimization among others. However, some algorithms lack easy application or understanding, generate rectilinear trajectories, that is, they lack realism causing the robot to make an extra effort when following a trajectory, all this with an inefficiency with respect to the execution time or with the length of the trajectory obtained.

For this investigation, was contemplated the use of a more sophisticated trajectory planning algorithm called Theta*, in order to achieve short trajectories with an acceptable execution time. The tests carried out at the simulation level were implemented with a differential configuration robot, in five different maps, and twenty-seven trajectories were evaluated with time and distance metrics, where the Theta* algorithm generated and reached the goal in 100 % of the cases, unlike the Dijkstra algorithms that generated 84 % of the trajectories and reached the goal 58 % and A * that generated 97 % of trajectories and reached the goal in 80 % of cases.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

La robótica móvil constituye una valiosa herramienta para el desarrollo y creación de robots de navegación autónoma, permitiendo de esta manera explorar entornos inaccesibles a los seres humanos ya sea por su lejanía, costo, peligro o sencillamente por tratarse de tareas repetitivas o laboriosas (Martínez, 1997).

Esta tesis está enfocada en dos de las tareas más importantes requeridas en la navegación autónoma de un robot. La primera es la planificación de rutas, que tiene como objetivo calcular el mejor camino entre un punto de inicio y un punto meta. La segunda es el seguimiento de rutas, donde el robot debe ser capaz de seguir y recuperar la ruta planificada evitando obstáculos (Azkune, 2011). Para llevar a cabo la planificación de trayectorias en la literatura se presentó A* como un algoritmo de búsqueda de grafos teniendo su mayor utilización en la robótica y los videojuegos. Según (Nash, 2012) tiene las siguientes propiedades:

1. Simplicidad: Son fáciles de aplicar y entender.
2. Eficiencia: Proporcionan una buena solución de compromiso con respecto al tiempo de ejecución de la búsqueda y de la longitud de la trayectoria resultante.
3. Generalidad: Se puede utilizar para buscar cualquier grafo de tipo euclidiano.

Sin embargo, las trayectorias que genera el algoritmo A* se condicionan a los bordes de una malla, generando rutas rectilíneas, provocando que el robot realice un esfuerzo extra al seguir una trayectoria.

El objetivo principal de esta tesis, fue evaluar un algoritmo más sofisticado que no se limite a los bordes del grafo, sin perder las propiedades de simplicidad, generalidad y eficiencia.

1.1. Descripción del problema

La tarea de navegación para un robot móvil significa recorrer una trayectoria que lo conduzca desde una posición inicial hasta otra final, pasando por ciertos puntos intermedios que conforman la trayectoria. En la literatura para tratar la planeación de trayectorias en la robótica, se recurre con frecuencia al algoritmo A* debido a su simplicidad y a que encuentra la trayectoria de menor costo siempre y cuando esta exista. Sin embargo, las propiedades de este algoritmo se condicionan a los vértices del grafo a partir del entorno

del robot, lo cual hace que las trayectorias sean rectilíneas y tengan un valor computacional alto.

1.1.1. Delimitación del problema específico

La planeación de trayectorias en el campo de la robótica se ha tratado con algoritmos como Dijkstra, A* y versiones mejoradas de A*. Por ejemplo: A* en Tiempo Real (*Real-Time A*, RTA**) enfocado a la planeación de trayectorias en tiempo real, A* Simplificado de Memoria Acotada (*Simplified Memory Bounded A*, SMA**), algoritmo que minimiza el tiempo de memoria para la búsqueda de trayectorias y A* post-suavizado, el cual abarca el suavizado de trayectorias con un tiempo de ejecución alto.

El proyecto de tesis, está enfocado en la planeación de trayectorias utilizando ocho vértices de búsqueda, mediante la implementación del algoritmo Theta* en ROS. De acuerdo a la literatura, este algoritmo genera trayectorias cortas y en un tiempo de ejecución aceptable, de modo que el robot móvil tenga un mejor rendimiento a la hora de la navegación en el entorno.

1.1.2. Complejidad del problema

La complejidad del problema radica en los siguientes puntos:

- Comprender el algoritmo Theta*.
- Comprender la versión Indigo de ROS para ejecutar los algoritmos Dijkstra y A*.
- Implementar el algoritmo Theta* en la versión Kinect de ROS.

1.1.3. Hipótesis

Incrementar el número de vértices de búsqueda a ocho en el algoritmo de planeación de trayectorias, puede encontrar la ruta más corta y directa al generar búsquedas en diagonal, mejorando el rendimiento del robot al navegar en un entorno conocido estático.

1.2. Objetivos

1.2.1. Objetivo general

Evaluar el algoritmo Theta* en la tarea de planeación de trayectorias utilizando ocho vértices de búsqueda.

1.2.2. Objetivos específicos

- Conocer y comprender los conceptos básicos de la robótica móvil.
- Estudiar los algoritmos recientes de planeación de trayectorias.
- Comprender el funcionamiento del algoritmo de planeación de trayectorias Theta*.
- Comprender el funcionamiento de ROS (Sistema Operativo Robótico).
- Implementar el algoritmo Theta* en ROS.
- Validar la eficacia y generalidad del algoritmo Theta* implementado en ROS.

1.3. Alcances y limitaciones

1.3.1. Alcances

- Analizar cinco de los algoritmos tradicionales para la planificación de trayectorias.
- Analizar principalmente el algoritmo de planificación de trayectorias A* disponible en ROS.
- Las trayectorias serán cortas y en tiempo de ejecución aceptable.
- Utilizar mapas 2D.
- Las pruebas de realizarán a nivel simulación, con un robot de configuración diferencial.

1.3.2. Limitaciones

- El entorno del robot no será dinámico.
- Solo se trabajarán con cinco mapas con extensión *.yaml*, los cuales son ejecutados por ROS.

1.4. Justificación

El Sistema Operativo Robótico (ROS), cuenta con el algoritmo Dijkstra y A* para la generación de trayectorias, implementar el algoritmo Theta* en esta plataforma y evaluar su comportamiento a nivel simulación con respecto a los algoritmos que proporciona ROS, sería un punto de partida para que el algoritmo Theta* se implemente en una plataforma robótica de configuración diferencial.

1.5. Organización del documento

Este documento, se estructura de la siguiente manera:

Capítulo 2. Describe los fundamentos necesarios para comprender el tema de tesis, considerando el Sistema Operativo Robótico (ROS), la planeación de trayectorias y los algoritmos utilizados para realizar esta tarea como parte de la navegación robótica.

Capítulo 3. Menciona algunos trabajos relacionados con la planeación de trayectorias que fueron realizados en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET). Dentro de este mismo Capítulo, se detallan algunos trabajos tomados como base para la investigación y realización del trabajo de tesis, los cuales se involucran en la planeación de trayectorias.

Capítulo 4. Aborda la implementación y ejecución en simulación cada uno de los tres algoritmos de planificación de trayectorias en el sistema ROS, así como los nodos utilizados para ejecutar algunos procesos de la navegación.

Capítulo 5. Detalla las métricas, describe los mapas utilizados y experimentos realizados para evaluar la eficacia y generalidad del algoritmo Theta*.

Capítulo 6. Muestra las conclusiones del trabajo de investigación, las aportaciones y recomendaciones para trabajos futuros.

Capítulo 2. Marco Conceptual

En este capítulo, se detallan a fondo los conceptos y herramientas necesarias para el entendimiento y realización del tema de tesis, profundizando en la planeación de trayectorias y las generalidades del Sistema Operativo Robótico (ROS).

2.1. Planeación de trayectorias

El problema de la generación de un camino a seguir por un robot móvil evadiendo los obstáculos del entorno se ha tratado ampliamente en la literatura (Latombe, 1991). En la actualidad existe un gran número de métodos para la planeación y generación de trayectorias, desde algoritmos simples hasta lógicas de programación más complejas, por ende es importante seleccionar un método adecuado acorde a la aplicación requerida (Yandún y Sotomayor, 2011).

Convencionalmente, el problema de generación de trayectorias para un robot móvil indica que dado un robot y su entorno de navegación, se puede determinar una trayectoria entre los puntos específicos de inicio y meta (Espitia, 2011).

Considerando las técnicas desarrolladas para la planeación de trayectorias se puede proponer la siguiente clasificación:

a) Determinísticos:

- Basados en grafos:
  - *Algoritmo A*.*

    Propuesto en 1968, es un algoritmo de búsqueda que busca el camino más corto desde un punto inicial al punto meta usando una heurística óptima, garantizando encontrar la ruta óptima entre el inicio y la meta, sí es que ella existe.

  - *Grafos de visibilidad.*

    Propuesto por Nils J. Nilsson, el grafo se determina aplicando el concepto de visibilidad el cual consiste en determinar sí dos puntos sobre el espacio se pueden unir mediante un segmento de recta que no presente interferencia con ningún obstáculo. Se genera un grafo de conectividad entre vértices visibles y se escoge la mejor ruta (Nilsson, 1969).

- *Modelado del espacio libre.*

  Método desarrollado por Rodney A. Rrooks, generalmente se aplica a espacios de trabajo con obstáculos poligonales. La construcción del grafo se realiza mediante Cilindros Rectilíneos Generalizados (CRG), al emplearlos se busca que el móvil se desplace de la forma más lejana de los obstáculos. La ruta desde la configuración inicial hasta la final está compuesta por una serie de CRG interconectados (Brooks, 1983).

  - *Descomposición en celdas.*

  Estos métodos consiste en descomponer el espacio libre en un conjunto de celdas desde el punto inicial hasta el final, por lo tanto, el problema consiste en encontrar una sucesión de celdas que no presente discontinuidades. Con la descomposición exacta de celdas, el planificador de rutas del robot tiene como objetivo encontrar un grafo de adyacencia que indique cuáles celdas libres de obstáculos comparten una frontera común, con el objetivo de identificar el espacio libre por donde se puede desplazar el robot. Los nodos del grafo corresponden a las celdas y las aristas conectan nodos de celdas adyacentes.

  - *Campos potenciales artificiales.*

  Para generar la trayectoria con estos potenciales solo se requiere el cálculo de gradientes, sin embargo, los campos potenciales generados pueden ser no convexos presentando mínimos locales lo cual dejaría caer al robot en una trampa. Otra desventaja consiste en la estimación de los parámetros del campo potencial para lograr una adecuada evasión de obstáculos.

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

- Planeador aleatorio de trayectorias.

  El algoritmo de planeación aleatoria de trayectorias fue desarrollado por Jean-Claude Latombe y Jérome Barraquand en 1991. Este algoritmo emplea movimientos Brownianos para poder salir de mínimos locales (Barraquand *et al.,* 1992).

- Mapas probabilísticos.

  Los mapas buscan crear un mapa generado de forma aleatoria libre de colisiones el cual conecta de forma rápida el móvil con el objetivo.

- Árboles de exploración rápida.

  Es una alternativa en la generación aleatoria de trayectorias, este método fue propuesto por Steven M. LaValle en 1998. En este método se crean árboles que se expanden de forma aleatoria para tener una conexión desde el punto de partida hasta el objetivo (Lindemann, 2006).

- Basados en optimización:
  - *Algoritmos genéticos.*

  Los algoritmos genéticos son estrategias de búsqueda basados en el mecanismo de selección natural y en algunos casos se involucran aspectos de genética natural, imitando a la evolución biológica como estrategia para resolver problemas. Los algoritmos genéticos tienen su campo de aplicación importante en problemas de optimización complejos, donde se tiene diferentes parámetros o conjuntos de variables que deben ser combinadas para su solución (Aguado y Jiménez, 2013).

  - *Colonia de hormigas.*

  Son algoritmos aproximados utilizados para obtener soluciones lo suficientemente buenas a problemas complejos en una cantidad razonable de tiempo de cómputo. Estos algoritmos, que están basados en una colonia de hormigas artificial, son agentes computacionales que trabajan de manera conjunta para poder comunicarse a través de rastros de feromonas artificiales (Colorni, 1992).

  - *Enjambre de partículas.*

  El comportamiento de enjambres ha sido empleado con éxito en problemas de optimización, es un algoritmo inspirado en el comportamiento social de individuos dentro de enjambres en la naturaleza. Este método emplea una búsqueda basada en poblaciones; en la cual los individuos, para desplazarse en el espacio de búsqueda, usan su propia experiencia y el conocimiento de sus vecinos con distintos grados de confianza (Franco, 2011).

- Redes Neuronales Artificiales.

  Las Redes Neuronales Artificiales, se basan en la concepción sobre el funcionamiento del sistema nervioso humano y en la teoría general de redes a la solución de problemas (Hurtado and Alvarez, 2002).

2.1.1. Algoritmo Dijkstra

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, fue creado por Edsger Dijkstra en 1959. El algoritmo consiste en encontrar los caminos más cortos desde el nodo origen a todos los demás nodos de la red, siempre y cuando no haya enlaces de longitud negativa, (Dijkstra *et al.*, 1980) cuando se obtiene el camino más corto desde el vértice origen, el algoritmo se detiene. Debido a que en este algoritmo se visitan todos los nodos, el cálculo de una solución para grafos extensos se hace lento, lo cual aumenta el valor computacional.

El pseudocódigo del algoritmo Dijkstra se observa a continuación (Felner, 2011).

```
Algoritmo: Dijkstra
Input: Graph G = (V, E)
1  (∀x ≠ s) dist [x] = +∞  //Initialize dist []
2  dist[s] = 0;
3  S = ∅;
4  Q = V   //Keyed by dist[]
5          while Q ≠ ∅ do
6                  u = extract_min(Q);
7                  S = S ∪ {u};
8                  foreach vertex v ∈ Adj(u) do;
9                          dist[v] = min(dist[v], dist[u] + w(u, v));
10                         //"Relax" operation
```

Dado un vértice (s) en un gráfico dirigido ponderado $G = (V, E)$ donde todos los bordes deben ser positivos, Dijkstra encuentra el camino con el menor costo (camino más corto) entre (s) y otro vértice en G.

Se inicializan las distancias en cero, ya que son valores desconocidos al principio, $dist[s] = 0$ y $dist[x]$, es la longitud de la ruta conocida más corta desde (s) hasta $x \in Q$. En cada ciclo, Dijkstra extrae el vértice $u \in Q$ con el mínimo $dist[]$ en Q. Luego, para cada vecino v de u establece la operación de relajación (2.1.1):

$$dist[v] = min(dist[v], dist[u] + w(u, v)) \tag{2.1.1}$$

La Figura 2.1 muestra una ruta generada por el algoritmo Dijkstra.

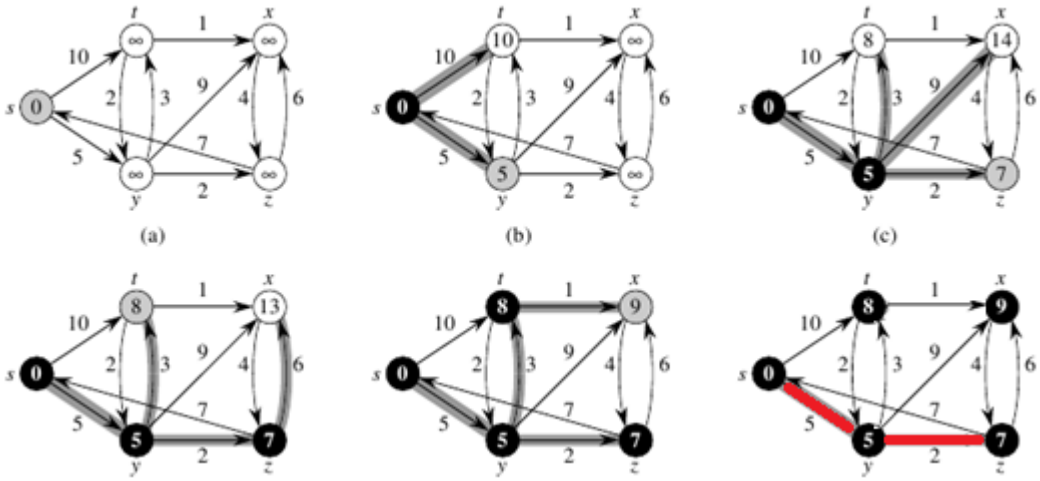


Figura 2.1: Ejemplo de la ruta con el algoritmo Dijkstra (IAGraph, 2017).

2.1.2. Algoritmo A*

El algoritmo A* propuesto por Peter E. Hart, Nils J. Nilsson y Bertram Raphael en 1968, es un algoritmo de búsqueda basado en grafos que busca la ruta más corta (de menor costo) entre dos puntos. Es uno de los algoritmos de planificación más usados (Fletchet *et al.*, 2008), debido a que de haber una solución, la encuentra y garantiza que es óptima, tomando en cuenta que el valor computacional no es tan alto como el del algoritmo Dijkstra.

Según (Nash, 2012) tiene las siguientes propiedades:

1. Simplicidad: Son fáciles de aplicar y entender.
2. Eficiencia: Proporcionan una buena solución de compromiso con respecto al tiempo de ejecución de la búsqueda y de la longitud de la trayectoria resultante.
3. Generalidad: Se puede utilizar para buscar cualquier grafo de tipo euclidiano.

El pseudocódigo del algoritmo A* se observa a continuación (Nash *et al.*, 2007).

Algoritmo: A*

```
1  Main()
2      g(s_star)=0;
3      parent(s_star)= s_star;
4      open=∅;
5      open.Insert(s_star,g(s_star)+h(s_star));
6      closed=∅;
7      while open≠ ∅ do
8          s =open.Pop();
9          if s = s_goal then
10                 return "path found";
11         closed=closed ∪{s};
12         [UpdateBounds(s)];
13         foreach s' ∈ succ(s) do
14             if s' ∉ closed then
15                 if s' ∉ open then
16                     g(s') = ∞;
17                     parent(s') = NULL;
18                 UpdateVertex(s,s');
19     return "no path found";
20 end
21 UpdateVertex(s,s')
22     if g(s) + c(s,s') < g(s') then
23         g(s') = g(s) + c(s,s');
24         parent(s') = s;
25         if s' ∉ open then
26             open.Remove(s');
27         open.Insert(s', g(s') + h(s'));
28 end
```

Para enfocar su búsqueda, el algoritmo A* utiliza los siguientes valores:

$h(s) \rightarrow$ Consistentes en la distancia estimada desde la posición inicial a la posición de destino de final, en este caso se usa una función heurística para calcular el valor estimado.

$g(s) \rightarrow$ Es la longitud del camino desde la posición inicial a la posición actual.

El padre $parent(s)$, se utiliza para extraer la ruta después de la búsqueda.

A * mantiene dos estructuras de datos globales:

1. La lista abierta, una cola de prioridad que contiene vértices a considerar para la expansión.
2. La lista cerrada, que contiene vértices que ya se han expandido y asegura que cada vértice se expande sólo una vez.

Mediante la ecuación (2.1.2) se obtiene el valor de $f(s)$ que es el valor calculado, que indica el camino más corto.

$$f(s) = g(s) + h(s) \tag{2.1.2}$$

La Figura 2.2 muestra una ruta generada por el algoritmo A*.

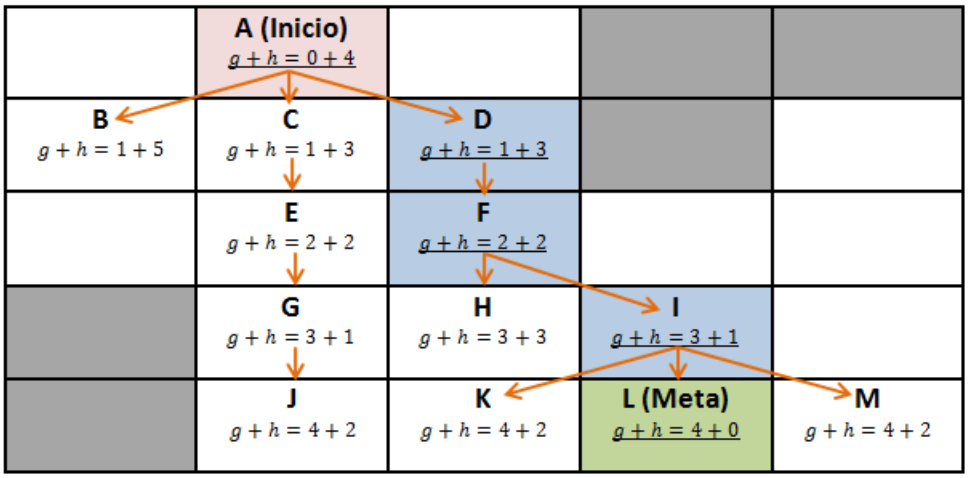


Figura 2.2: Ejemplo de la ruta con el algoritmo A*.

2.1.3. Algoritmo Theta*

El algoritmo Theta*, tiene sus bases de búsqueda en el algoritmo A* que planifica trayectorias propagando la información a lo largo de los bordes de la cuadrícula sin restringir las trayectorias a los nodos de la cuadrícula (Nash *et al.*, 2007).

Este algoritmo se basa en A* combinado con las propiedades deseables de dos técnicas de planificación de ruta existentes:

- Grafos de visibilidad: Un vértice está conectado a través de una línea recta a otro vértice sí y sólo sí tiene una línea de visión directa hacia el otro vértice sin pasar a través de una celda bloqueada. Los cambios de rumbo ocurren sólo en las esquinas de las celdas bloqueadas, como se muestra en la Figura 2.3 (derecha).
- Uso de rejillas: Es el uso de mallas en el grafo, como se muestra en la Figura 2.3 (izquierda).
- Líneas de visibilidad: Dos vértices tienen línea de visión directa sí y sólo sí ninguno de los puntos representados corresponden a una celda bloqueada.

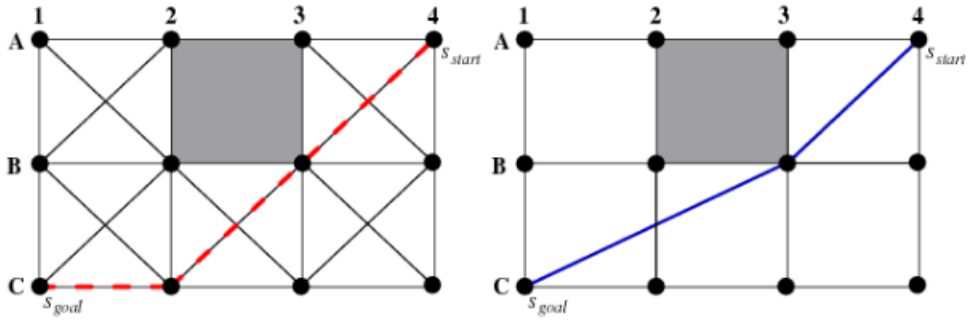


Figura 2.3: Propiedades de grafos de visibilidad (derecha) y rejillas (izquierda) (Nash *et al.*, 2007).

El pseudocódigo del algoritmo Theta* se observa a continuación (Nash *et al.*, 2007).

```
Algoritmo: Theta*
1  Main()
2      g(s_star)=0;
3      parent(s_star)= s_star;
4      open=∅;
5      open.Insert(s_star,g(s_star)+h(s_star));
6      closed=∅;
7      while open≠ ∅ do
8          s =open.Pop();
9          if s = s_goal then
10                 return "path found";
11         closed=closed ∪{s};
12         [UpdateBounds(s)];
13         foreach s' ∈ succ(s) do
14             if s' ∉ closed then
15                 if s' ∉ open then
16                     g(s') = ∞;
17                     parent(s') = NULL;
18                 UpdateVertex(s, s');
19     return "no path found";
20 end
21 ComputeCost(s,s')
22     if LineofSight(parent(s), s') then
23         // Path 2 //
24         if g(parent(s)) + c(parent(s), s') < g(s') then
25             parent(s') = parent(s);
26             g(s') = g(parent(s)) + c(parent(s), s');
27     else
28         // Path 1 //
29         if g(s) + c(s), s' < g(s') then
30             parent(s') = s;
31             g(s') = g(s) + c(s, s');
32 end
```

La diferencia clave entre Theta* y A*, es que Theta* permite que el padre de un vértice sea cualquier vértice, a diferencia de A* donde el padre debe ser un vecino visible, actualizando la $g - value$ y el padre de un vecino visible no expandido s' de vértice s considerando los dos caminos siguientes (línea 21) del pseudocódigo de Theta*.

Para permitir rutas en más ángulos (línea 24), Theta* también considera la ruta desde el vértice de inicio al $parent(s)[= g(parent(s))]$ y desde el $parent(s)$ a s' en línea recta$[= c(parent(s), s')]$, lo que da como resultado una longitud de $g(parent(s)) + c(parent(s), s')$ sí s' tiene línea de visión con el $parent(s)$.

Theta* considera la ruta desde el vértice de inicio $s[= g(s)]$ y de s a s' en línea recta$[= c(s, s')]$, lo que da como resultado una longitud de$g(s) + c(s, s')$ (línea 29).

La Figura 2.4 muestra una ruta generada por el algoritmo Theta*.

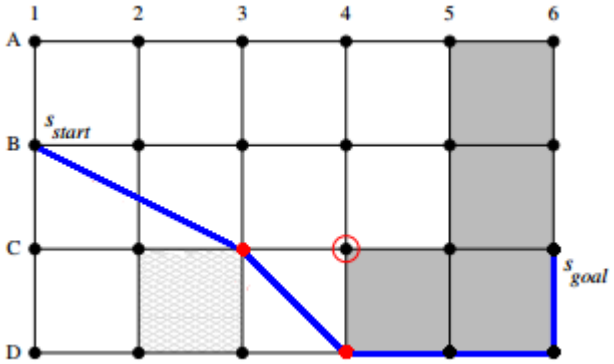


Figura 2.4: Ejemplo de la ruta con el algoritmo Theta* (Nash *et al.*, 2007).

2.1.4. Complejidad algorítmica

La complejidad algorítmica "orden de complejidad", representa la eficiencia de un algoritmo, de acuerdo a la cantidad de recursos (temporales) que necesita un algoritmo para resolver un problema y por tanto permite determinar la eficiencia de dicho algoritmo.

La Tabla 2.1 muestra la complejidad algorítmica de cada uno de los algoritmos utilizados para la planificación de trayectorias.

Tabla 2.1: Complejidad algorítmica.

Algoritmo	**Orden de complejidad**		**Observación**
Dijkstra	Cuadrática	$O(n^2)$	Se da en caso de que cada (u) fuera adyacente con todos los vértices (v).
A*	Lineal	$O(n)$	La complejidad está relacionada con la heurística utilizada, pero en el mejor de los casos trabaja de forma lineal.
Theta*	Lineal	$O(2n)$	Se da debido a que realiza la busqueda la realiza doble, es decir, $O(n+n)$ (Path 1 y Path 2).

2.2. Sistema Operativo Robótico (ROS)

ROS (*Robot Operating System*) es un sistema operativo para ayudar a los desarrolladores de software a crear aplicaciones para robots (ROS, 2016a). ROS provee controladores de hardware y diferentes dispositivos embebidos, librerías, administración de paquetes y más (ROS, 2016b). El sistema operativo está orientado principalmente para el sistema Linux (Ubuntu), las versiones de ROS están referidas por un sobrenombre en lugar de por una versión numérica, las más actuales son:

1. **Kinetic Kame** liberada el 23 de mayo de 2016.
2. **Jade Turtle** liberada el 23 de mayo de 2015.
3. **Indigo Igloo** liberada el 22 de julio de 2014.

ROS está representado por una arquitectura de grafos con una topología centralizada, donde el procesamiento toman lugar en nodos que pueden recibir y enviar información de sensores, actuadores, control y trayectorias (Martínez y Fernández, 2013). El sistema operativo proporciona simuladores virtuales algunos en 3D como Gazebo, Stage, V-REP; un visualizador 3D de sensores llamado Rviz, también incluye paquetes que ayudan en tareas de localización, mapeo y navegación, los cuales se detallan en la Tabla 2.2.

Tabla 2.2: Paquetes disponibles en ROS (ROS, 2016b).

Localización	**Mapeo**	**Navegación**
AMCL, *Adaptive Monte Carlo Localization:* Localización Adaptativa de Monte Carlo es un sistema de localización probabilístico para un robot que se mueve en un entorno conocido 2D.	Gmapping: Provee SLAM basado en laser, generando mapas en 2D.	*base_local_planner:* Proporciona implementaciones de la trayectoria de implantación y ventana dinámica para la navegación del robot.
fake_localization: Proporcione información de la odometría y transforma los datos del formulario de AMCL.	*crsm_slam:* Localización y asignación simultánea de localización de exploración de rayos críticos, realiza un escaneo de emparejamiento utilizando el algoritmo *Random Restart Hill Climbing.*	*clear_costmap_recovery:* Proporciona un comportamiento de recuperación para la navegación que intenta limpiar el espacio para revertir el costo del mapa utilizado para el mapa estático fuera de un área determinada.
robot_pose_ekf: Estima la posición 3D de un robot. Se basa en el Filtro de Kalman Extendido.	*hector_slam:* Aprende del entorno y puede estimar simultáneamente sin odometría la posición 2D.	*costmap_2d:* Proporciona una implementación del costo del mapa en 2D, que toma los datos generados por los sensores para construir mapas de rejillas en 2D o 3D.
		dwa_local_planner: Proporciona una implementación de un procedimiento de ventana dinámica para la navegación local de robots.
		global_planner: Este paquete en especial proporciona algoritmos para la planificación de rutas como Dijkstra y A*.
		move_base: Proporciona una implementación de una acción que dada una posición objetivo, intentará llegar con una base móvil. Trabaja la unión de un planificador global y local para llevar a cabo su tarea de navegación.
		nav_core: Proporciona interfaces comunes para las acciones específicas de navegación del robot.
		navfn: Proporciona una función de navegación rápida interpolada que se puede utilizar para crear planos de una base móvil.

2.2.1. Niveles conceptuales de ROS

Nivel 1: Sistema de archivos

Los conceptos de nivel de sistema de archivos, se refieren principalmente a los recursos de ROS que se encuentran en el disco, como por ejemplo (GitBook, 2017):

- Espacio de trabajo: Contiene las carpetas que incluyen los paquetes de ROS.
- Paquete: Un paquete tiene la estructura mínima y contenido para crear un programa dentro de ROS. Puede tener procesos de tiempo de ejecución ROS (nodos), bibliotecas, archivos de configuración entre otros.
- Manifestó del paquete: Definido por el archivo *package.xml* en cada paquete. Proporcionan metadatos acerca de cada paquete, incluyendo su nombre, versión, descripción, información sobre la licencia y las dependencias, entre otras.
- Repositorio: Es una colección de paquetes que comparten la misma versión.
- Servicio: Es una estructura de tipo *request-response*, es decir, mediante un mensaje se realiza una solicitud y se espera una respuesta. Definido por el archivo con extensión *.srv*.

Nivel 2: Gráfico de procesos

Es la red de procesos *peer-to-peer* de ROS. Los componentes básicos para administrar, aprovechar y optimizar los procesos son (GitBook, 2017):

- Nodos: Son ejecutables que ocupa ROS para comunicarse con otros nodos de procesos. El uso de nodos separa el código y funcionalidades que simplifican el sistema.
- Servidor de parámetros: Es un diccionario compartido. Todos los nodos pueden leer y escribir parámetros en tiempo de ejecución.
- Mensaje: Es simplemente una estructura de datos para el paso de información entre los nodos de ROS. Se define por el archivo con extensión *.msg*.
- Temas: El tema es un nombre que se utiliza para identificar el contenido del mensaje.
- Servicios: Otra forma de comunicar nodos pero con un modelo de publicación y suscripción (muchos a muchos).
- Tópicos: Son canales de comunicación utilizados por los nodos para transmitir datos y publicar mensajes. Cuyo contenido puede ser compartido entre todos los nodos mediante un patrón de publicación y suscripción.
- Bolsas: Las bolsas son un formato para guardar y reproducir datos. Se define por el archivo con extensión *.bag*.

2.3. Conclusiones

La tarea de planeación de trayectorias dentro de la navegación autónoma, ha sido un tema de vital importancia para la robótica, motivo por el cual se han desarrollado bastantes algoritmos que realizan esta tarea, Dijkstra y A* se consideran algoritmo pioneros según la literatura, motivo por el cual el Sistema Operativo Robótico (ROS) los incluye en el paquete de navegación, para poder visualizar como los algoritmos realizan el seguimiento de la trayectoria que generan. En el siguiente capítulo se abordarán algunos de los trabajos que se han realizado con respecto a la planeación de trayectorias en esta institución y en otros lugares.

Capítulo 3. Estado del Arte

En este capítulo, se mencionan los trabajos que tratan la planeación de trayectorias y los algoritmos implementados para realizar esta tarea, desarrollados en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), posteriormente se detallan trabajos y artículos que fueron parte fundamental para el desarrollo del proyecto de tesis.

3.1. Antecedentes

Los trabajos mostrados a continuación fueron desarrollados en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) y abordan el tema de las trayectorias, utilizando la mayoría de ellos la odometría visual.

Análisis cinemático, dinámico y control en tiempo real de un vehículo guiado automáticamente (Camarena, 2009)

En este trabajo se abordó el problema asociado al seguimiento de trayectorias en un robot móvil con ruedas, de tipo TWMR (*Two Wheel Mobile Robot*). Se utilizó un prototipo de robot móvil mecánicamente simple en su construcción el cual se observa en la Figura 3.1, que logró disminuir costos y hacer más sencilla su configuración. El control de seguimiento de una trayectoria fue en tiempo real con un vehículo de tres grados de libertad guiado automáticamente, implementando modelos cinemático y dinámico, además de una ley de control no lineal. El objetivo fue contar con una plataforma experimental para evaluar diferentes esquemas de control avanzado, relacionados con el seguimiento de trayectorias, evasión de obstáculos y planeación de trayectorias.

Figura 3.1: Vista lateral derecha del prototipo construido (Camarena, 2009).

Control embebido de un vehículo guiado automáticamente mediante Redes Neuronales Artificiales (Martínez, 2012)

El objetivo principal de esta tesis fue desarrollar un esquema de control basado en Redes Neuronales Artificiales, aplicado a un vehículo guiado automáticamente para el seguimiento de trayectorias predefinidas. Migrando los sistemas de control actuales a un esquema embebido, el robot fue realmente autónomo y no tuvo necesidad de una computadora para su control. Se presentaron resultados de simulación del control, en donde se utilizó el identificador neuronal y se diseñó una ley de control basado en este. Se aplicó el algoritmo de la ley de control propuesto al vehículo guiado automáticamente y se probó con diversas trayectorias. El desempeño fue mejor comparado a los índices de desempeño de (Camarena, 2009).

Planeación de trayectorias y control por platitud diferencial de múltiples robots móviles (Escamilla, 2013)

En esta tesis, para la planeación de trayectorias el autor se basó en la platitud diferencia, que es una propiedad que posee cierta clase de sistemas dinámicos, la cual permite la transformación de ecuaciones diferenciales no lineales a un sistema algebraico de ecuaciones el cual puede ser relativamente fácil de resolver. Aunado a lo anterior, como los estados y las entradas se pueden expresar en términos de las salidas planas, entonces fue de gran utilidad, ya que la trayectoria pudo ser planeada en el espacio de las salidas planas del sistema usando funciones de interpolación o polinomios de orden apropiado. Los controladores para las trayectorias se implementaron en un dispositivo comercial de la marca LEGO®, que cuenta con las características adecuadas, en software y hardware, para la optimización de las trayectorias, en la Figura 3.2 se puede ver el ejemplo de una trayectoria.

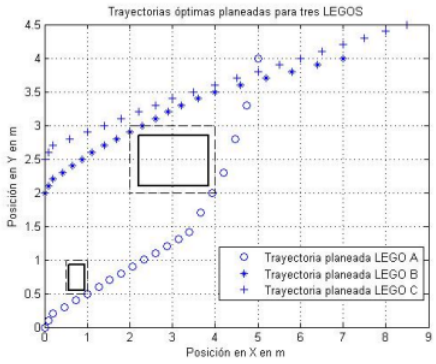


Figura 3.2: Trayectorias óptimas para tres LEGOS caso 1(Escamilla, 2013).

Navegación visual en trayectorias cerradas (Peñaloza, 2014)

En este documento de tesis, se presentó una solución al problema del seguimiento de trayectorias en robots móviles, partiendo de un sistema de navegación visual al cual se entrenó para seguir una trayectoria, dicho entrenamiento consistió en la adquisición de

una memoria visual de la trayectoria a recorrer. Finalmente se logró desarrollar un sistema de visión binocular capas de guiar a un robot móvil a través de una ruta cerrada y sin obstáculos. La ruta estaba especificada por imágenes obtenidas mediante un algoritmo de correspondencia basado en el seguimiento de puntos destacados. La base del sistema se centró en la detección, descripción y correspondencia de un punto destacado en las imágenes adquiridas.

Navegación, localización y mapeo de robots móviles para trayectorias pre-especificadas por imágenes (Vergara, 2015)

En esta tesis, se presentó un sistema de visión artificial que permitió a un robot móvil realizar un recorrido autónomo sobre una trayectoria, la cual se definió por una secuencia de imágenes extraídas del video capturado durante un recorrido de entrenamiento. Finalmente, las trayectorias fueron generadas a través de imágenes, los experimentos se realizaron en entornos de interiores, con condiciones controladas y sin obstáculos, mostrando que el sistema fue capaz de determinar en qué parte de la trayectoria se encontraba el robot y finalmente permitió la trayectoria de entrenamiento de principio a fin de forma autónoma con un error de seguimiento aceptable (menor a 10 cm).

Evaluación de las técnicas SLAM disponibles en ROS (López, 2017)

En este documento de tesis, se centra en la metodología de Localización y el Mapeo Simultaneo (*Simultaneous Localization And Mapping*, SLAM), en el campo de la robótica que permite la construcción de mapas en 2D o en 3D utilizando un robot móvil. Y la implementación en una plataforma robótica TurtleBot, conformada por un robot Roomba 645 y un sensor RGB-D Kinect, mediante el Sistema Operativo Robótico. Los algoritmos que permitieron realizar la localización del robot y el mapeo del entorno, fueron, Gmapping, HectorSLAM y CRSM SLAM; mientras que para la navegación, se utilizó el paquete *Frontier Exploration* y la teleoperación.

Los resultados representados en la Figura 3.3, mostraron que Gmapping tuvo el mejor desempeño para la localización y el mapeo, ya que generó mapas aproximados de todos los entornos del Cenidet (planta alta, planta baja, aula 3103 y aula 2110) que se utilizaron para las pruebas y tuvo la capacidad de recuperar la pose del robot en la mayoría de los casos. En cuanto a la navegación y construcción de mapas de manera semi-autónoma, el uso de los algoritmos Frontier Exploration y Gmapping, permitió la exploración y la generación de mapas aproximados de los entornos de interiores de manera semi- autónoma.

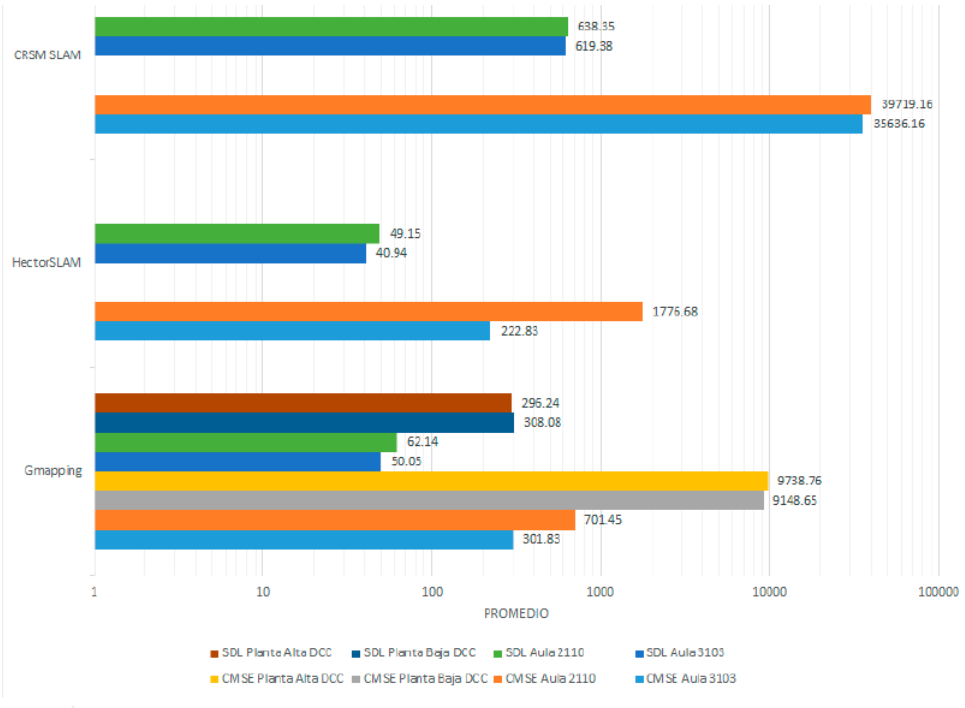


Figura 3.3: Comparativa general del desempeño de los algoritmos en los distintos entornos (López, 2017).

3.2. Trabajos relacionados

Dentro de la comunidad científica se han realizado investigaciones y trabajos en los últimos cinco años referentes al tema de la planeación de trayectorias. A continuación se resumen algunos de ellos.

Planificación de trayectos en tiempo real para el robot en entornos conocidos (Duchoň *et al.*, 2014)

Los autores realizaron una comparación de algunos algoritmos de planificación y su análisis para las aplicaciones en la robótica móvil. Para ello describieron, verificaron y compararon los algoritmos de planificación en entornos conocidos. Los resultados mostraron que los algoritmos como Theta* simple e Incremental Phi*, fueron capaces de encontrar la ruta más corta, aunque necesitaron mayor tiempo de cálculo. El algoritmo más rápido en mapas de gran simetría fue JPS (*Jump Point Searc*h), el cual no encontró la ruta más corta posible cuando se comparó con otros algoritmos como Theta*simple, en la Figura 3.4, se muestra la trayectoria que generó JSP. El algoritmo D* no está adaptado para mapas con una mayor simetría, ya que buscó casi el doble del número de células que otros algoritmos. De acuerdo a las recomendaciones del artículo se menciona que el algoritmo JPS funciona rápido en trayectorias para entornos estáticos, Theta* simple consigue una longitud corta en mapas grandes siempre y cuando no exista el requisito de trabajar en tiempo real.

En cuanto a Incremental Phi* puede ser utilizado en entornos dinámicos; sin embargo, los resultados mostraron que sólo cuando se produce una pequeña cantidad de obstáculos dinámicos colocados no inmediatamente al inicio del robot.

Figura 3.4: Trayectoria JSP (Duchoň *et al.,* 2014).

Un nuevo enfoque de planificación de rutas para robots móviles (Goyal and Nagla, 2014)

Este artículo presentó un método de planificación de trayectoria basado en el algoritmo A*, el cual se modificó para encontrar un camino más seguro para robots móviles de dimensión específica, el cual supone que el tamaño virtual del obstáculo presente en el entorno se incrementa aproximadamente $(2n + 1)$ veces del tamaño de la celda, en la Figura 3.5 se muestra la trayectoria generada, al incrementar tres veces los obstáculos. El análisis experimental del método propuesto mostró una mejora en la planificación de trayectorias al reducir la posibilidad de colisiones. Los resultados mostraron que el algoritmo A* pudo determinar una trayectoria más corta, pero el robot chocaba con algunos obstáculos en un entorno real, mientras que el método propuesto determinó la ruta más segura para el robot.

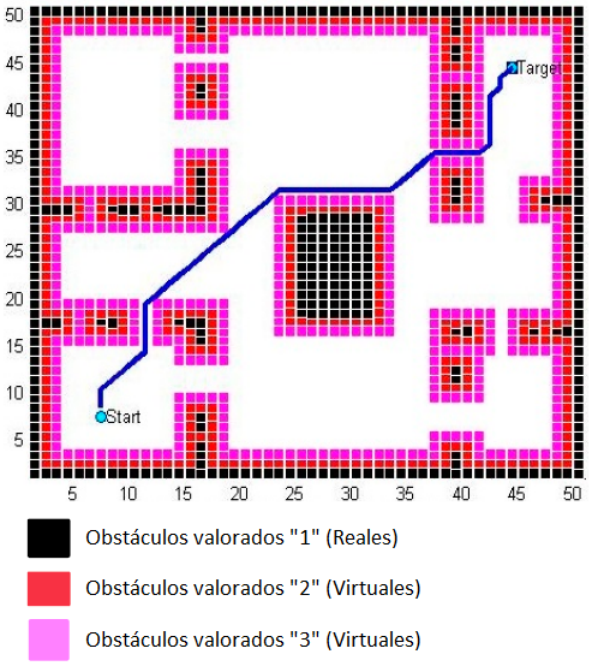


Figura 3.5: Planificación de ruta propuesta tres veces más grande (Goyal and Nagla, 2014).

Seguimiento de trayectorias con un robot móvil de configuración diferencial (Guzmán y Vásquez, 2015)

En este trabajo se desarrolló un sistema de control, aplicado en un robot móvil de configuración diferencial, para seguir una trayectoria determinada. Se implementó la cinemática directa del robot para simular el comportamiento del mismo. Para cumplir el objetivo, dos sistemas de control (holonómico y no-holonómico) fueron desarrollados e implementados a partir de la cinemática inversa. Posteriormente, se desarrollaron pruebas que permitieron comparar el rendimiento de los controladores, determinando cuál proporciona la mejor solución. Para los resultados se realizaron pruebas de rendimiento luego de implementar los dos sistemas donde los criterios a comparar fueron el tiempo de respuesta y distancia recorrida. El control holonómico presentó mejores resultados, porque en las pruebas el tiempo empleado y la distancia recorrida son menores. Esto se debe a que el movimiento del robot presenta menos oscilaciones y su error es más cercano a cero.

Algoritmo de planificación de trayectorias para móviles autónomos robot en entorno dinámico (Ganeshmurthy and Suresh, 2015)

En este artículo se propuso un método basado en la heurística para buscar eficientemente el camino inicial factible. Se utilizaron los vértices de los obstáculos dinámicos como espacio de búsqueda para obtener el camino óptimo para los robots, el enfoque buscó el camino factible para el robot en un entorno dinámico que contenía obstáculos estáticos y dinámicos, y luego se utilizó el recorrido simulado para obtener la ruta óptima en el mapa dinámico. Uno de los algoritmos de planificación de trayectorias A* se simuló en

MATLAB y la planificación de trayectoria dinámica se visualizó en el Entorno Virtual de Robots 3D (VREP). Como novedad presentaron un robot móvil que pudo ser controlado manualmente y también operado de forma autónoma desde cualquier lugar de la red.

C-Theta*: Planificación de rutas basadas en clústeres en las cuadrículas (Mendonca and Goodwin, 2015)

En este trabajo se mostró una optimización del algoritmo Theta* simple, este algoritmo funcionaba mediante la utilización de líneas de control de vista durante la búsqueda y llegaba a su destino a medida que aumenta el tamaño del mapa de cuadrícula. C-Theta* es un algoritmo que tuvo un menor tiempo para llegar a su destino, para ello se dividió un mapa en regiones de alta y baja densidad utilizando un algoritmo de agrupación sin supervisión basado en el número de nodos bloqueados en un mapa de cuadrícula. Los resultados mostraron que el tiempo que tardó en encontrar el camino más corto fue reducido significativamente (en algunos casos) en comparación con Theta* simple, mientras que la longitud de la trayectoria permanecía tan corta como Theta* simple. El uso de mapas más complejos, detallados y el aumento en el tamaño del mapa de juego trajo consigo nuevos desafíos para optimizar el algoritmo de planificación de rutas. C-Theta* en algunos casos obtuvo una longitud de trayectoria similar o igual a la de Theta* simple.

Planificación de trayectorias en cuadrículas: El efecto de la colocación de vértices en la longitud de trayectorias estrella (Tovey *et al.,* 2015)

En este artículo los autores estudiaron el efecto de dos diseños para los entornos de videojuegos, los cuales son la colocación de vértices y la conectividad de la red. Demostraron que la colocación de vértices en las esquinas de las celdas en lugar de en los centros tiende a dar lugar a trayectos más cortos. También, cuantificaron la ventaja de las redes de 8 vecinos sobre las cuadrículas de 4 vecinos en una métrica particular de interés que permitió determinar cómo varían las longitudes de una trayectoria verdaderamente más corta, una trayectoria de vértice más corta y una trayectoria de rejilla más corta. Los resultados teóricos cuantificaron la ventaja de colocar vértices en las esquinas, en lugar de los centros cuando se encuentran los trayectos de rejilla más cortos en cuadrículas cuadradas de 4 y 8 vecinos.

Una comparación empírica algoritmos de planificación de la trayectoria en cualquier ángulo (Uras and Koenig, 2015a)

Los autores compararon cinco algoritmos de planificación de la trayectoria en cualquier-ángulo en términos de calidad de la solución y el tiempo de ejecución en mapas de videojuegos. Los algoritmos de interés son Theta*, A*, D*, ya que, la planificación que incluían a cualquier ángulo de dirección es una área nueva de investigación, y no existía una comparación directa entre los algoritmos de cualquier ángulo y cualquier ángulo de submeta gráfica. Theta* es un algoritmo simple de implementar y se puede aplicar en gráficos de entornos 2D o 3D. El algoritmo D* se puede utilizar fácilmente para incrementar la ruta planeada. Los resultados fueron comparados con otros artículos utilizando por

ejemplo, el mapa de Dragon Age donde los algoritmos A* y Theta* fueron entre 1.5 y 2.5 veces más rápido, aunque A* presentó mejores resultados en juego y mapas aleatorios.

Aceleración de la ruta de cualquier ángulo en las cuadrículas (Uras and Koenig, 2015b)

En este artículo, se analizaron y aprovecharon las similitudes entre Gráficos de Submeta (estos gráficos se construyeron a partir de cuadrículas colocando sub-objetivos en las esquinas de los obstáculos y conectándolos) y los Gráficos de Visibilidad ayudaron a los Gráficos de Submeta a encontrar rápidamente caminos "de cualquier ángulo", extendiendo así su aplicabilidad. Las trayectorias de cualquier ángulo fueron generalmente más cortas y más realistas que las trayectorias de la rejilla, ya que el movimiento a lo largo de estas trayectorias no tuvo restricción a los bordes de la rejilla. Los resultados mostraron que los Gráficos de Submeta se pueden utilizar para encontrar trayectorias de cualquier ángulo con algunas modificaciones simples, además en comparación con los algoritmos Theta* y A* resultó ser más eficiente utilizando entornos de (5×5 bloques).

Planificación del movimiento no-holonómico basada en RRT usando polarización de trayecto de cualquier ángulo (Palmieri *et al.,* 2016)

En este artículo, se explicó que la los arboles de exploración rápida o RRT (*Rapidly Exploring RandomTree*) es una técnica ocupada para la planificación de trayectorias, particularmente en robots de ruedas con limitaciones no holonómicas complejas. Debido a que existe un error al escalar estos robots se propuso Theta*-RRT combinando jerárquicamente la búsqueda de cualquier ángulo con la planificación del movimiento RRT para robots de ruedas no holonómicos. Theta*-RRT es una variante de RRT que genera una trayectoria expandiendo un árbol de longitud hacia estados muestreados cuya distribución resume la información geométrica de la trayectoria de cualquier ángulo. Se demostró mediante algunos experimentos tanto para un sistema de accionamiento diferencial como para un sistema de camión y remolque de alta dimensión, que Theta*-RRT encuentra trayectorias más cortas significativamente más rápido que otros planificadores de línea de base (RRT, RRTA*, A*) sin pérdida de suavidad.

Navegación de robots móviles en mapas parcialmente conocidos usando una versión rápida del algoritmo A* (Muntean, 2016)

En este artículo se evalúo la eficiencia de dos versiones del algoritmo A* para la navegación de robots móviles en ambientes interiores con la ayuda de las aplicaciones de software y el robot Pioneer 2DX. La versión rápida del algoritmo A* utilizó un enfoque avanzado de fusión de sensores (es decir, combinar las entradas de múltiples sensores de robot) para tratar mejor con entornos parcialmente conocidos. Los resultados mostraron que esta versión se puede utilizar con éxito para la navegación de robots móviles en interiores. En la Figura 3.6 se observa el mapa 2D utilizado para probar los algoritmos, así como una trayectoria generada por la versión rápida del algoritmo A*, donde el punto de inicio fue la esquina superior izquierda.

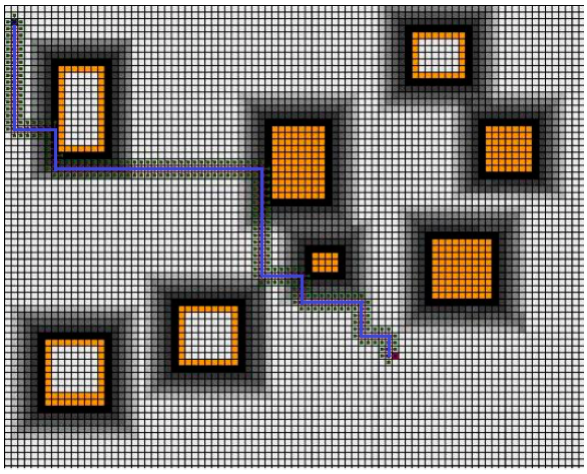


Figura 3.6: El mapa 2D utilizado para probar los algoritmos A* y A* rápida. (Muntean, 2016).

3.3. Discusión del Estado del Arte

En los trabajos se ha tratado la planeación de trayectorias con diferentes algoritmos o técnicas principalmente A* y sus versiones mejoradas como A* modificado lógicamente para lograr una trayectoria más segura para un robot, A* y un método basado en la heurística con el objetivo de planificar trayectorias para robot en entornos dinámicos.

Algunos de los trabajos se enfocan en presentar nuevos algoritmos y compararlos con otros algoritmos de planeación para medir la eficiencia de cada uno de ellos, por ejemplo en (Mendonca and Goodwin, 2015) donde el algoritmo C-Theta* se compara con una versión anterior llamado Theta* simple, en (Uras and Koenig, 2015a) se realiza la comparación de cinco algoritmos que realizan trayectorias en algo llamado "cualquier ángulo" y (Muntean, 2016) donde se evalúan la eficiencia de dos versiones del algoritmo A*.

A continuación, se muestra una Tabla 3.1 con la información más relevante de cada artículo.

Tabla 3.1: Resumen de artículos de planeación de trayectorias.

Trabajo	**Algoritmo (s)**	**Objetivo**	**Utilidad**
Planificación de trayectos en tiempo real para el robot en entornos conocidos.(Duchoň *et al.,* 2014)	Theta*simple, Incremental Phi*, D* y JSP.	Comparar algunos algoritmos de planificación y analizar sus aplicaciones en la robótica móvil.	Información sobre los recientes algoritmos y su aplicación en la robótica.
Un nuevo enfoque de planificación de rutas para robots móviles. (Goyal and Nagla, 2014)	A* y A * modificado lógicamente.	Generar la ruta más segura para el robot móvil.	Información del algoritmo A* y conocimiento de la seguridad de las rutas para un robot.
Seguimiento de trayectorias con un robot móvil de configuración diferencial. (Guzmán y Vásquez, 2015)	Sistemas de control (holonómico y no-holonómico).	Desarrollar un sistema de control, aplicado en un robot móvil de configuración diferencial, para seguir una trayectoria determinada.	Información sobre los robots de configuración diferencial y métodos para el seguimiento de trayectorias.
Algoritmo de planificación de trayectorias para móviles autónomos robot en entorno dinámico. (Ganeshmurthy and Suresh, 2015)	A* y un método basado en la heurística.	Implementar un enfoque de recolección simulado para abordar el problema de planificación de trayectorias para robots en entornos dinámicos.	Información sobre el algoritmo A* y su simulación en un entorno dinámico.
C-Theta*: Planificación de rutas basadas en clústeres en las cuadrículas. (Mendonca and Goodwin, 2015)	C-Theta*	Presentar el algoritmo optimizado de Theta*, llamado C-Theta* y compararlos.	Información sobre el algoritmo C-Theta* y posible comparación de resultados.
Planificación de trayectorias en cuadrículas: El efecto de la colocación de vértices en la longitud de trayectorias estrella. (Tovey *et al.,* 2015)	Manipulación de rejillas.	Estudiar el efecto de dos decisiones de diseño y la determinación de las longitudes de las trayectorias resultantes.	Información sobre la manipulación de rejillas en los mapas, para la obtener una trayectoria de menor longitud.

Tabla 3.1: Resumen de artículos de planeación de trayectorias (Continuación).

Trabajo	**Algoritmo (s)**	**Objetivo**	**Utilidad**
Una comparación empírica algoritmos de planificación de la trayectoria en cualquier-ángulo.(Uras and Koenig, 2015a)	Theta*, A*, D*, cualquier ángulo y cualquier ángulo de submeta gráfica.	Comparar cinco algoritmos de planificación de la trayectoria en cualquier-ángulo en términos de calidad y ejecución.	Comparación de resultados con el algoritmo Theta* en entornos 2D.
Aceleración de la ruta de cualquier ángulo en las cuadrículas. (Uras and Koenig, 2015b)	Gráficos de Submeta.	Buscar generar trayectorias de cualquier ángulo.	Información sobre la generación de trayectorias en cualquier ángulo y posible comparación de resultados en cuanto a tiempo de ejecución de una trayectoria.
Planificación del movimiento no-holonómico basada en RRT usando polarización de trayecto de cualquier ángulo. (Palmieri *et al.,* 2016).	Theta*-RRT.	Eliminar el error de escalamiento que se causa en el robot de ruedas con limitaciones no holonómicas causadas por el algoritmo RRT.	Información sobre el algoritmo Theta*-RRT y búsqueda de suavidad en las trayectorias.
Navegación de robots móviles en mapas parcialmente conocidos usando una versión rápida del algoritmo A*. (Muntean, 2016)	A* y A* rápido	Evaluar la eficiencia de dos versiones del algoritmo A*, A* clásico y A* rápido.	Información del comportamiento del algoritmo A* y A* rápido en entornos interiores.

3.4. Conclusiones

En este capítulo se mostraron algunos trabajos realizados en este plantel y en otras instituciones relacionadas con la tarea de planificación y seguimiento de trayectorias en el campo de la robótica. Algunas de las investigaciones implementan nuevos algoritmos o técnicas para mejorar la trayectoria y/o el valor computacional y se comparan con otros algoritmos.

Las tesis publicadas hasta ahora en el Cenidet, han trabajado en la tarea de navegación utilizando diferentes robots, algunos con dos ruedas como el caso de (Camarena, 2009) y (Escamilla, 2013) el cual utilizo un LEGO, generando buenas trayectorias, sin embargo, las trayectorias se pegan a los obstáculos, aunque esto se compensa con los sensores de proximidad con los que se equipe el robot. Después de haber leído algunos trabajos se llegó a la conclusión de que implementar un algoritmo en un robot puede ser una tarea complicada o laboriosa, lo que pretende este trabajo de tesis es compactar todo el trabajo en un paquete de ROS creado por nosotros mismos, donde se encuentre el algoritmo de planeación de trayectorias con el que se desea trabajar y así mismo tener la facilidad de llevarlo al mundo real implementándolo en un robot omnidireccional o diferencial (dos ruedas). Finalmente la tesis de (López, 2017) fue de suma importancia debido a que proporciono conocimientos del sistema ROS en cuando a su conexión con el robot Roomba y el Kinect para generar mapas en buenas condicione, los cuales me permitieron realizar la planeación y seguimientos de trayectorias en entornos conocidos.

Con respecto a los trabajos relacionados en (Duchoň *et al.,* 2014), (Mendonca and Goodwin, 2015) y (Uras and Koenig, 2015a) hablan sobre el desempeño del algoritmo Theta* en comparación con otros algoritmos y poniéndolo a prueba en diferentes entornos, esto fue el motivo de que se implementara Theta* en ROS, para ponerlo a prueba a nivel simulación con los algoritmos Dijkstra y A*. El artículo de (Goyal and Nagla, 2014) fue tomado en cuenta para estudiar como incrementar el tamaño de los obstáculos, ya que esto es una buena alternativa para robot que no cuenten con sensores de proximidad. A partir de esta información, en el siguiente capítulo se procede a mostrar la implementación del algoritmo Theta* en ROS para ser evaluado con respecto a los algoritmos Dijkstra y A*.

Capítulo 4. Implementación

En este capítulo, se mencionan los tres algoritmos de planificación de trayectorias la forma en la que cada algoritmo es ejecutado en ROS.

Los algoritmos de planificación de trayectorias Dijkstra y A* proporcionados por ROS se ejecutan de manera diferente al algoritmo Theta* debido a que este algoritmo es externo al sistema robótico y su proceso es un poco más extenso. La Figura 4.1 muestra el proceso de forma general para la ejecución de un algoritmo de planificación.

Primero en ROS se crea el espacio de trabajo, el cual será el encargado de contener todo el proyecto, posteriormente se van creando carpetas dentó del espacio de trabajo las cuales contendrán principalmente los archivos lanzadores (*.launch*), archivos con el código del algoritmo (*.cpp*) y la carpeta de mapas con las extensiones (*.yaml* y *.png*).

Para ejecutarlo se abren terminales de ROS; primero se compila el espacio de trabajo para que las modificaciones, como el cambio de mapa fueron realizados, una vez que la compilación llegó al 100 %, en la segunda terminal se carga el algoritmo, finalmente se carga Rviz y se agregan los tópicos necesarios para visualizar el mapa y la trayectoria, posteriormente se le da un punto meta para que el algoritmo genere y haga el seguimiento de la trayectoria.

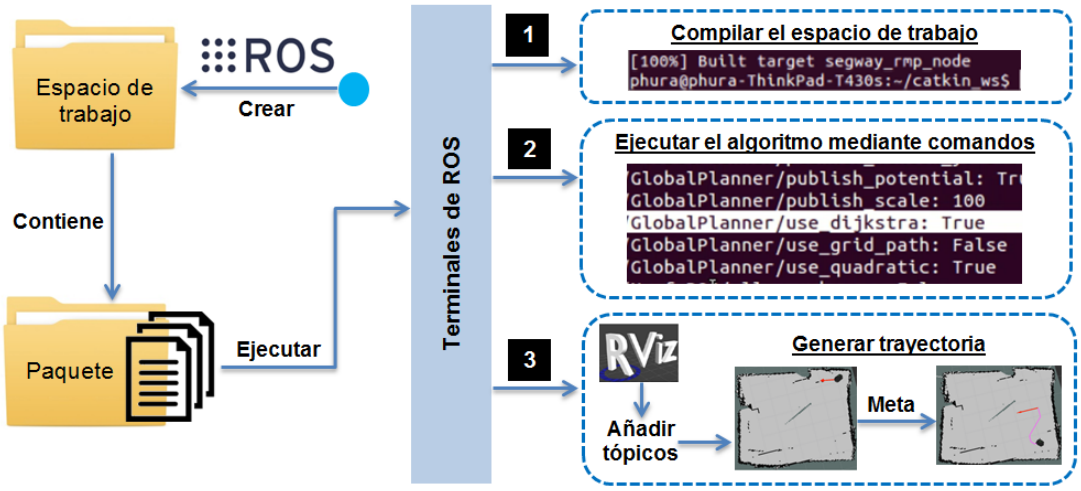


Figura 4.1: Metodología de solución.

4.1. Instalación y ejecución de los algoritmos de planeación de trayectorias

4.1.1. Algoritmos Dijkstra y A*

Se requiere tener instalado Ubuntu en la versión 14.04 y ROS Indigo, para poder ejecutar estos algoritmos con ayuda de los siguientes pasos:

Paso 1: Crear un espacio de trabajo utilizando una sola terminal (ROS, 2016b).

Inicializar variables de estado colocando después el nombre del distrito, en este caso indigo.

```
$ source /opt/ros/indigo/setup.bash
```

Crear el espacio de trabajo con el nombre deseado, en este caso "*algoritmos_gp*":

```
$ mkdir -p ~/algoritmos_gp/src
$ cd ~/algoritmos_gp/src
$ catkin_init_workspace
```

Una vez creado es necesario tener archivos en la carpeta "*src*", se procede a llenarla (se generarán dos carpetas "*build*" y "*devel*")

```
$ cd ~/algoritmos_gp/
$ catkin_make
```

Para continuar agrega en la fuente de instalación el archivo (.sh)

```
$ source devel/setup.bash
```

Paso 2: Completar con paquetes del sistema.

En este caso fue necesario clonar de internet al espacio de trabajo, tres paquetes básicos de ROS para realizar simulaciones y no dañar el sistema al editar el código fuente de cada algoritmo.

El paquete *navegation*, contiene paquetes asociados a la localización y navegación entre ellos *global_planner* con los códigos de los algoritmos.

```
$ git clone https://github.com/ros-planning/navigation.git
```

El paquete *turtlebot_simulator*, para ejecutar el simulador Gazebo.

```
$ git clone https://github.com/turtlebot/turtlebot_simulator.git
```

El paquete *turtlebot_apps* que permite hace la configuración de los parámetros y los lanzadores para los algoritmos de navegación.

```
$ git clone https://github.com/turtlebot/turtlebot_apps.git
```

Paso 3: Añadir el repositorio de mapas y archivos modificados.

El repositorio de mapas de añadió en la siguiente dirección: /home/indigo/ algoritmos_gp/src/turtlebot_apps/turtlebot_navigation/maps

Los archivos modificados en carpetas de la siguiente dirección: /home/indigo/ algoritmos_gp/src/turtlebot_apps/turtlebot_navigation

En la carpeta "*launch*", se almacenan el archivo encargado de la ejecución de los nodos del sistema, a continuación se describen los archivos que se encuentran en esta carpeta:

- move_base.launch: Este archivo vincula un planificador global y local para llevar a cabo la navegación. Se debe colocar las rutas completas en la que se encuentran los archivos dentro del espacio de trabajo, tal como se muestra en el siguiente cuadro.

```
<arg name="map_file" default="/home/indigo/algoritmos_gp/src/turtlebot_apps/
turtlebot_navigation/maps/MapaA3.yaml"/>
# Se coloca el nombre del mapa para la simulación de la trayectoria.
<include file="/home/indigo/algoritmos_gp/src/turtlebot_apps/turtlebot_navigation/
launch/includes/amcl/amcl.launch.xml">
<rosparam file="/home/indigo/algoritmos_gp/src/turtlebot_apps/
turtlebot_navigation/param/costmap_common_params.yaml"
<rosparam file="/home/indigo/algoritmos_gp/src/turtlebot_apps/
turtlebot_navigation/param/local_costmap_params.yaml"
<rosparam file="/home/indigo/algoritmos_gp/src/turtlebot_apps/
turtlebot_navigation/param/global_costmap_params.yaml"
<rosparam file="/home/indigo/algoritmos_gp/src/turtlebot_apps/
turtlebot_navigation/param/dwa_local_planner_params.yaml"
<rosparam file="/home/indigo/algoritmos_gp/src/turtlebot_apps/
turtlebot_navigation/param/move_base_params.yaml"
<rosparam file="/home/indigo/algoritmos_gp/src/turtlebot_apps/
turtlebot_navigation/param/global_planner_params.yaml"
<rosparam file="/home/indigo/algoritmos_gp/src/turtlebot_apps/
turtlebot_navigation/param/navfn_global_planner_params.yaml"
```

En la carpeta "*param*", se encuentras los archivos de configuración de parámetros para habilitar y ejecutar los algoritmos de navegación contenidos dentro del paquete *global_planner*, los cuales se describen a continuación:

- global_planner_params.yaml: Este archivo contiene los parámetros de configuración básica del algoritmo de navegación. El parámetro más importante a modificar es el que se muestra a continuación en color rojo, donde se cambia el valor para aplicar el algoritmo Dijkstra o A*.

```
GlobalPlanner:
old_navfn_behavior: false #Refleja el comportamiento de Navfn, por defecto es falso.
use_quadratic: true #Utilizar la aproximación cuadrática potencial.
use_dijkstra: false #Utiliza el algoritmo A*, de lo contrario usa Dijkstra.
use_grid_path: false #Crear una ruta que siga los límites de la cuadrícula, por defecto es falso.
```

- move_base_params.yaml, en este archivo se modifica el parámetro global (GlobalPlanner).

```
base_global_planner: "global_planner/GlobalPlanner" #alternatives: navfn/NavfnROS, carrot_planner/CarrotPlanner
```

Paso 4: Ejecutarlo en simulación.

Terminal 1: Inicializar las variables de entorno e iniciar la comunicación con ROS.

```
$ source /opt/ros/indigo/setup.bash
$ roscore
```

Terminal 2: Ejecutar el simulador Gazebo.

```
$ cd algoritmos_gp
$ source devel/setup.bash
$ cd src/turtlebot_simulator/turtlebot_gazebo/launch
$ roslaunch turtlebot_world.launch
```

Terminal 3: Ejecutar el archivo move_base.launch.

```
$ cd algoritmos_gp
$ source devel/setup.bash
$ cd src/turtlebot_apps/turtlebot_navigation/launch
$ roslaunch move_base.launch
```

Terminal 4: Ejecutar Rviz.

```
$ rosrun rviz rviz
```

Paso 5: Agregar tópicos.

- Pestaña By display type
  - RobotModel
- Pestaña By topic
  - Path

- Topic: /move_base/GlobalPlanner/plan

- Map
  - Topic: /map
- Pose
  - Topic: /move_base/current_goal

4.1.2. Algoritmo Theta*

Se requiere tener instalado Ubuntu en la versión 16.04 y ROS Kinect, para poder ejecutar estos algoritmos con ayuda de los siguientes pasos:

Paso 1: Crear un espacio de trabajo como el caso anterior, con el nombre "*catkin_*ws".

Paso 2: Crear un paquete con el nombre "*primer_intento*" (ROS, 2016b).

Primero se debe estar dentro del espacio de trabajo.

```
$ cd ~/catkin_ws/src
```

Crear el paquete "*primer_intento*", con las siguientes dependencias.

```
$ catkin_create_pkg primer_intento std_msgs roscpp rospy navig_msgs nav_msgs geometry_msgs
```

Los servicios que atienten las dependencias son las siguientes:

- *std_msgs*, *roscpp* y *rospy*, son dependencias por defecto que sirven para enviar y recibir mensajes con lenguajes de programación C++ y Python.
- *navig_msgs*: Atiende al servicio *nav_msgs/OccupancyGrid*, el cual representa al mapa de cuadrícula 2D y marca las celdas ocupadas y libres.
- *nav_msgs*: Define los mensajes para interactuar con la navegación, incluye el servicio de *path* que coloca las poses para que el robot siga la ruta.
- *geometry_msgs*: Atiende a los servicios *geometry_msgs/Posestart_pose* (punto inicio) y *geometry_msgs/Pose goal_pose* (punto meta).

Paso 3: Añadir el código del algoritmo y la carpeta de mapas.

En la carpeta "*src*" de "*primer_intento*" se añade el archivo principal «theta_path_finder.cpp», el cual contiene el código del algoritmo de planificación Theta*.

La carpeta de mapas se añade en la siguiente dirección /home/user/catkin_ws/src/ planning/knowledge/navigation/occupancy_grids

Paso 4: Ejecutarlo en simulación.

Terminal 1: Este comando ejecutar el espacio de trabajo y el paquete que contiene el algoritmo Theta*.

```
$ jcm
```

Terminal 2: Este comando abre Rviz para poder realizar las trayectorias.

```
$ roslaunch surge_et_ambula theta_test.launch
```

En este archivo *theta_test.launch,* se modificara el mapa en el que se desea ejecutar el algoritmo la parte que se modifica es *map_server*, a continuación se muestra la línea de código que se modifica:

```
<node name="map_server" pkg="map_server" type="map_server" output="screen" args="$(find knowledge)/navigation/occupancy_grids/MapaPB.yaml"/> #Se coloca el nombre del mapa para la simulación de la trayectoria.
```

Paso 5: Agregar tópicos.

- Pestaña By display type
  - RobotModel
- Pestaña By topic
  - Path
    - Topic: /move_base/GlobalPlanner/plan
  - Map
    - Topic: /map
  - Pose
    - Topic: /move_base/current_goal

4.2. Nodos utilizados

Para poder ejecutar los algoritmos es necesario el uso de nodos, debido a que son ejecutables que permiten la comunicación a servicios y parámetros, incluso suscribir a tópicos necesarios para las trayectorias.

4.2.1. *Rviz*

Rviz (*Ros Visualizator)*, es un visualizador de datos de ROS. El programa permite guardar configuraciones personalizadas con los tópicos que se necesitan como usuario para la trayectoria, tal como se observa en la Figura 4.2, la Tabla 4.1 muestra a detalle para que sirve cada tópico (UNAM, 2017).

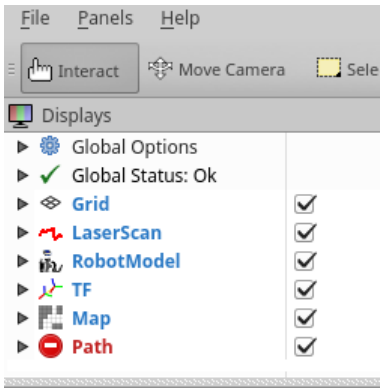


Figura 4.2: Tópicos suscritos en Rviz para la planeación de trayectorias.

Tabla 4.1: Nodo *Rviz*.

Tópicos suscritos	/navigation/localization/map [nav_msgs/OccupancyGrid]	Representa un mapa de cuadrícula bidimensional.
	/hardware/scan_[sensor_msgs/ LaserScan]	Escanear solo desde un rango de alcance.
	/tf [tf/tfMessage]	Publica la transformación de odometría.
	/navigation/mvn_pln/ last_calc_path [nav msgs/Path]	Representa el algoritmo encargado de la planificación.

Sintaxis en un archivo launch

Para ejecutar la configuración especial de Rviz es necesario especificar el nombre del nodo, el paquete y la dirección donde se guarda la configuración.

```
<node name="rviz" pkg="rviz" type="rviz" args="-d
$(find knowledge)/hri/rviz_config.rviz/>
```

4.2.2. *map_server*

Los mapas basados en celdas de ocupación dividen el entorno en una serie de celdas uniformes, pudiendo ser representado en dos dimensiones o tres dimensiones. Cada celda

contiene un valor de probabilidad, que nos indica sí está ocupada por un obstáculo o está libre por el momento (Morave and Elfes, 1985) ejemplo en la Figura 4.3, se observa un ejemplo donde las celdas verdes son consideradas como libres cuando no existe un obstáculo que abarque más de la mitad de la celda y las de color rojo son consideradas ocupadas cuando aproximadamente más de la mitad de la celda está ocupada por un obstáculo, un borde o la zona fuera del mapa.

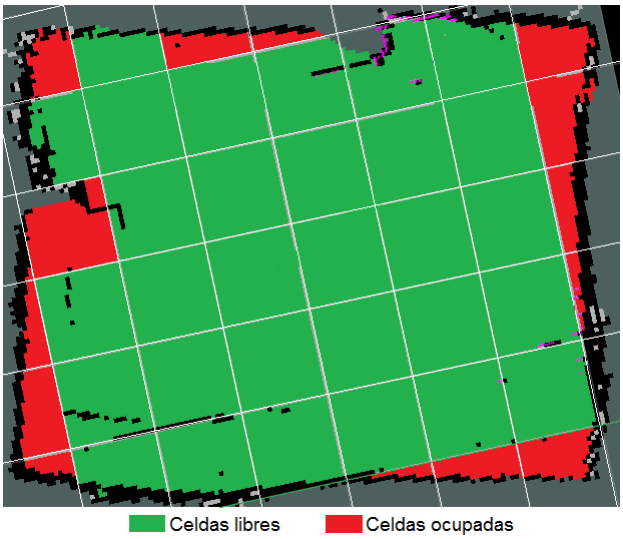


Figura 4.3: Ejemplo de clasificación de celdas.

ROS ocupa este tipo de mapas, debido a que son de fácil construcción incluso en entornos grandes, es fácil para un robot determinar su posición dentro del mapa tan sólo conociendo su posición y orientación, para ello el nodo ocupa información detallada en la Tabla 4.2.

Tabla 4.2: Nodo *map_server*.

Tópicos publicados	/navigation/localization/ map_metadata[nav_msgs/ MapMetaData]	Metadatos del mapa
	/navigation/localization/map [nav msgs/OccupancyGrid]	Mapa celdas ocupación
Servicios	navigation/localization/static map [nav_msgs/GetMap]	Obtención del mapa
Parámetros	frame_id [string, default: \map"]	Marco de referencia establecido en el encabezado (*header*) del mapa publicado.
occupied_thresh:	Grado de probabilidad para considerar que celda está totalmente ocupada.	
free_thresh:	Grado de probabilidad para considerar que la celda está complemente libre.	

ROS guarda los mapas 2D como una imagen por eso la resolución la proporciona en m/pix, comúnmente la resolución del mapa $0.05\,m/pix$, y el tamaño de las celdas es de $(5cm\,x\,5cm)$ (UNAM, 2017).

Sintaxis en un archivo launch

Para ejecutar este nodo se requiere especificar como argumento el archivo *.yaml* que contiene los metadatos del mapa que se quiere proveer (UNAM, 2017).

```
<node name="map_server" pkg="map_server" type="map_server" output="screen"
args="$(find knowledge)/navigation/occupancy_grids/MapaPA.yaml"/>$
 # Dirección completa del repositorio del mapa y nombre del mapa.
```

4.2.3. *amcl*

AMCL (*Monte Carlo Localization Approach)*, es un sistema de enfoque adaptativo de localización de Monte Carlo para un robot que se mueve en 2D.

Este nodo implementa AMCL, que utiliza un filtro de partículas para rastrear la pose de un robot en un mapa conocido, transforma los escaneos láser entrantes al sistema de referencia *odometry*. Por lo tanto, debe existir un camino a través de ramificaciones desde el sistema de referencia en el que los escaneos láser se publican hacia el sistema de referencia de odometría. Durante la operación AMCL estima y pública la transformación del marco de referencia de la base con respecto al marco de referencia global (*map*) y odometría (*odometry*) (ROS, 2016b), la información se detalla a continuación en la Tabla 4.3 (Flores, 2015).

Tabla 4.3: Nodo *amcl*.

Tópicos publicados	/hardware/scan_[sensor_msgs/ LaserScan]	Escaneos láser
	/navigation/localization/ particlecloud [geometry_msgs/ PoseArray]	Mapa celdas ocupación
Servicios	/navigation/localization/ global localization [std_srvs/Empty]	Inicio de la localización global, donde las partículas se dispersan al azar.
Parámetros	odom model type [string, default: \diff"]	Configuración del robot, ya sea diferencial (diff), omnidireccional (omni), etc.

4.2.4. *path_calculator*

Este nodo se encarga de calcular una ruta desde un punto inicial hasta una punto meta utilizando el algoritmo Theta*.

Sintaxis en un archivo launch

Para correr este nodo solo se requiere especificar el nombre que se le desea dar al nodo, el paquete (carpeta) en el que se encuentra y el nombre del ejecutable (UNAM, 2017).

```
<node name="path_calculator" pkg="primer_intento" type="theta_path_finder"
 output="screen"/>
```

4.3. Conclusiones

El presente capítulo, detallo el proceso para implementar cada algoritmo en ROS, junto con los nodos o bloques ejecutables utilizados para realizar una tarea en especial o ejecutar un proceso, pero que al establecer una comunicación entre ellos y los tópicos construyen el todo para realizar la simulación de la planificación de trayectorias. El próximo capítulo, presentara la experimentación, es decir, las trayectorias generadas en los cinco mapas por los algoritmos Dijkstra, A* y Theta*.

Capítulo 5. Experimentación y resultados

En este capítulo, se presentan las trayectorias generadas en los mapas por los algoritmos Dijkstra, A* y Theta*. Se detallan las métricas, parámetros y los mapas empleados en las experimentaciones.

5.1. Entorno de desarrollo

Para la implementación de los algoritmos de planificación de trayectorias, fue necesario utilizar el Sistema Operativo Robótico (ROS), el cual provee controladores de hardware y diferentes dispositivos embebidos, librerías, administración de paquetes y más (ROS, 2016b). Se utilizaron dos versiones de este sistema para poder ejecutar los algoritmos.

- **Arquitectura de software:**
  - Xubuntu 16.04
  - Ubuntu 14.04
  - ROS Versión Indigo Igloo
    - Liberada el 22 de julio de 2014
    - Paquete de navegación *global_planner*
    - Ejecutar el algoritmos Dijkstra y A*
  - Kinetic Kame
    - Liberada el 23 de mayo de 2016
    - Ejecutar el algoritmo Theta*
- **Arquitectura de hardware:**
  - Laptop LENOVO
  - Procesador Intel Core i5-4210U
  - Memoria RAM 8GB
  - Unidad de procesamiento gráfico Nvidia GeForce 820M

5.2. Repositorio de mapas

Para la validación del algoritmo implementado, se utilizaron cinco mapas: tres mapas son entornos reales los cuales se muestran en la Figura 5.1 del Departamento de Ciencias Computacionales del CENIDET, el mapa arena_a es un mapa obtenido de internet y el mapa willow_garage que es el mapa que trae por defecto ROS.

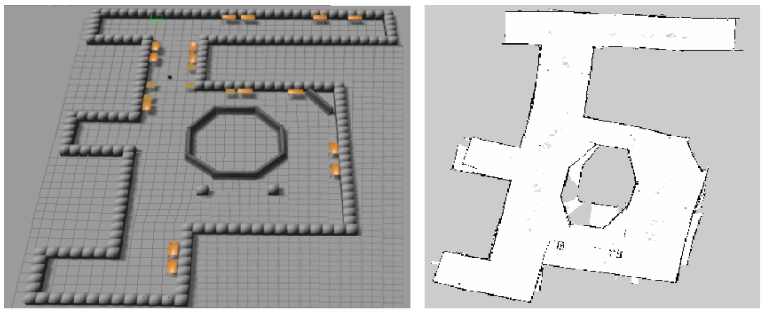
(a) Planta alta.

(b) Planta baja.

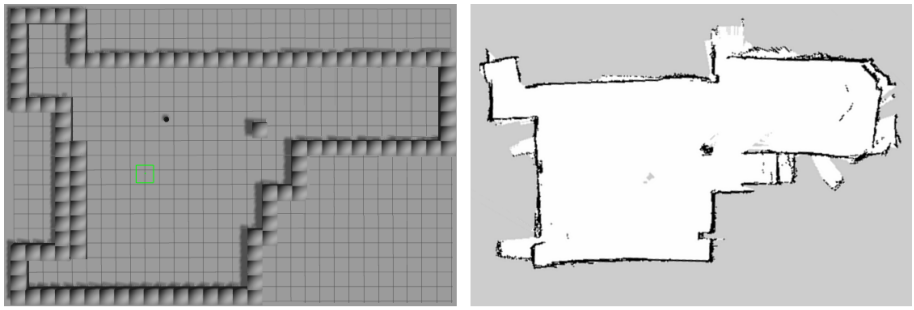
(c) Aula 3103.

Figura 5.1: Entornos reales (López, 2017).

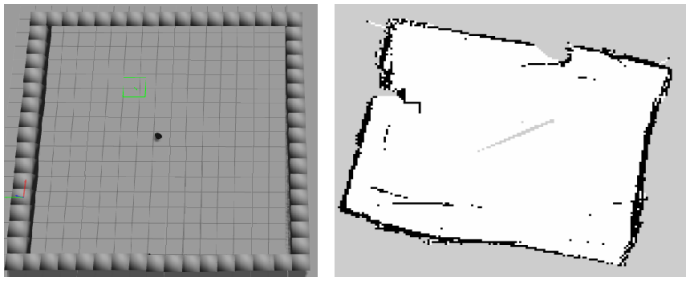
Una vez que se simulo el entorno en Gazebo como se muestra en la Figura 5.2 (lado izquierdo), se prosigue a ejecutar el algoritmo de Gmapping para el mapa en 2D tal como se observa en la Figura 5.2 (lado derecho), dichos mapas se obtuvieron de la tesis de maestría en Ciencias de la computación por CENIDET, de Diana Elizabeth López Borreguero, la cual lleva por nombre "Evaluación de técnicas SLAM disponibles en ROS". La finalidad de ocupar estos tres mapas, fue para ver la eficiencia de los mapa generados en la tesis de (López, 2017) y el desempeño del algoritmo en estos entornos, para que en un trabajo futuro se ejecuten los algoritmos en un robot móvil real.



(a) Mapa planta alta.



(b) Mapa planta baja.



(c) Mapa aula 3103.

(d) Mapa arena_a.

(e) Mapa willow_garage.

Figura 5.2: Entornos para simulación.

- Descripción de los mapas:
  - Mapa planta alta, resolución $992\,x\,960\,m/pix$, a nivel simulación es un mapa de dificultad media, puesto que solo tiene dos obstáculos cuadrados y el hexágono del centro.
  - Mapa planta baja, resolución $704\,x\,704\,m/pix$, es un mapa sin complicación alguna.
  - Mapa aula 3103, resolución $352\,x\,512\,m/pix$ y es el mapa más fácil debido a que es un entorno pequeño y libre de cualquier obstáculo.
  - Mapa arena_a, resolución $1984\,x\,1024\,m/pix$, es el mapa más grande del repositorio utilizado, con dificultad media alta, debido a los espacios angostos y líneas consideradas muros, lo cual incrementa giros en las trayectorias.
  - Mapa willow_garage, resolución $584\,x\,526\,m/pix$, es el mapa más difícil, principalmente porque no está cerrado al 100 %, tiene muchos cuartos y pasillos angostos que no están cerrados lo cual ocasiona que el robot se pierda fácilmente y no recupere la trayectoria generada.

5.3. Métricas

Los parámetros para la evaluación del desempeño de cada algoritmo, se eligieron de la literatura, con la finalidad de obtener una comparación entre el algoritmo implementado y los algoritmos que están dentro del paquete de navegación del sistema ROS. Los parámetros seleccionados se enlistan a continuación:

- El robot en la simulación ¿llego a la meta?
- Tiempo ejecución: Es el tiempo que tardó el algoritmo en generar la trayectoria del punto inicio al punto meta. ROS tiene la capacidad de configurar un reloj simulado para los nodos (ROS, 2016b).

$$ros :: Timebegin = ros :: Time :: now();$$

- Distancia Euclidiana (D_E): Es la distancia en metros entre dos puntos $P1\,y\,P2$, de coordenadas cartesianas $(x1, y1)$ y $(x2, y2)$ respectivamente, considerando en este caso la resolución de la celda que tiene el mapa, la cual equivale a $0.05\,m/pix$:

$$D_E(P_1P_2), = \sqrt{(0.05(x_2 - x_i)^2) + (0.05(y_2 - y_1)^2)} \tag{5.3.1}$$

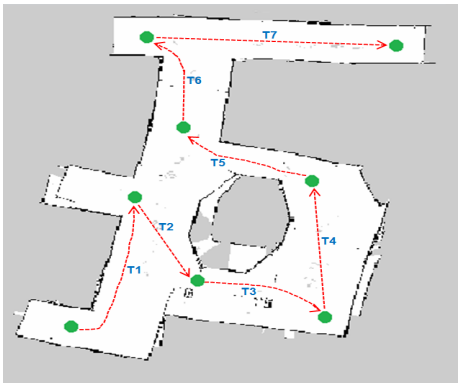
- Distancia de la trayectoria recorrida (P_L): Es la distancia total recorrida por el robot desde el punto inicio al punto meta. Para una trayectoria en el plano $x-y$ compuesta por n puntos, y asumiendo el punto de inicio como $(x1, f(x1))$ y la meta como el punto$(xn, f(xn))$, puede ser calculado por:

$$P_L = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (f(x_{i+1}) - f(x_i))^2} \tag{5.3.2}$$

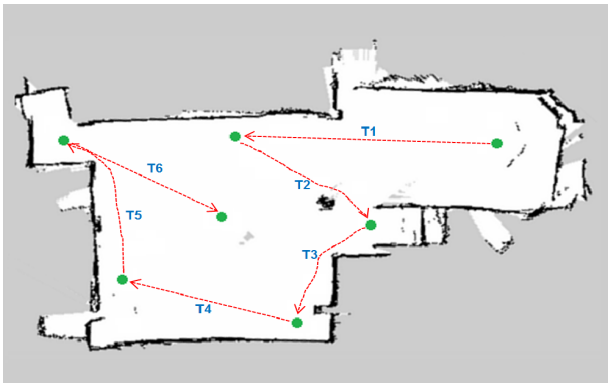
Donde $(xi, f(xi))$, $i = 1, 2, ..., n$ son los n puntos de la trayectoria en coordenadas cartesianas (Muñoz *et al.*, 2011).

5.4. Pruebas

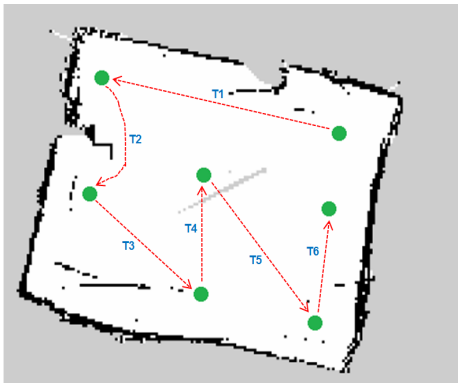
En la fase de realizaron pruebas, a cada mapa se le diseño una serie de trayectorias para que el algoritmo y el sistema tuvieran la capacidad de generar la trayectorias, aunque se hayan recorrido varios metros o se haya tenido un error anteriormente, las trayectorias se aprecian en la Figura 5.3, y fueron dibujadas por un humano al azar con el objetivo de que se recorriera todo el mapa, motivo por el cual no todo los mapas tienen el mismo número de trayectorias, excepto el mapa willow_garage, este mapa cuenta con tres trayectorias debido a las condiciones del mapa.



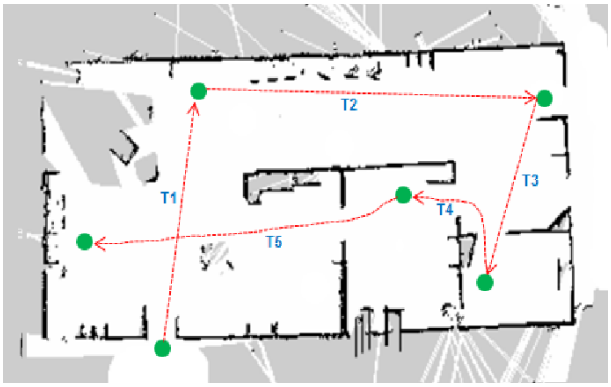
(a) Diseño de trayectorias planta alta



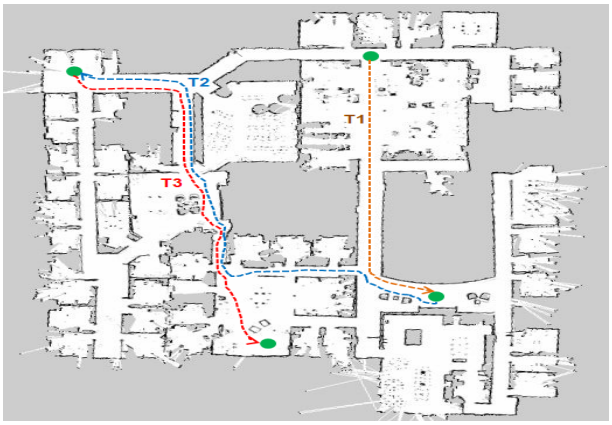
(b) Diseño de trayectorias planta baja



(c) Diseño de trayectorias aula 3103



(d) Diseño de trayectorias arena_a



(e) Diseño de trayectorias willow_garage

Figura 5.3: Mapas utilizados para generar trayectorias.

Los resultados se resumieron en tablas y gráficas, para poder observar el desempeño de cada algoritmo en cada mapa.

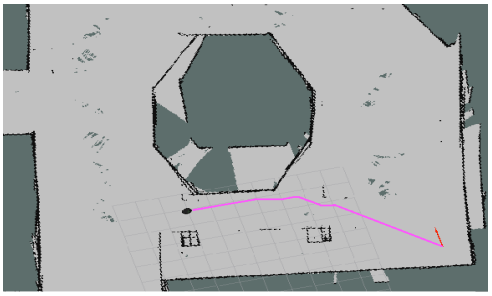
5.4.1. Trayectorias en el mapa planta alta

- **Algoritmo Dijkstra**

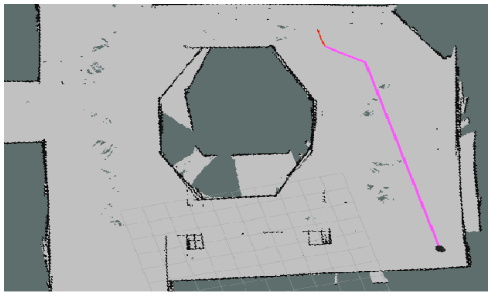
Las trayectorias generadas por el algoritmo Dijkstra se muestran en la Figura 5.4 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.

(a) Trayectoria T1 generada por Dijkstra

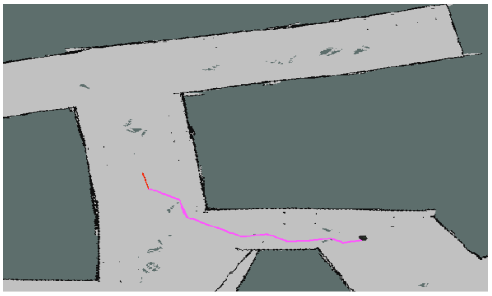
(b) Trayectoria T2 generada por Dijkstra



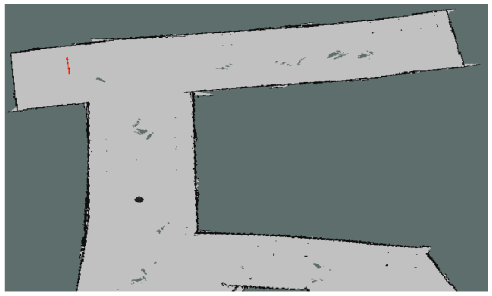
(c) Trayectoria T3 generada por Dijkstra



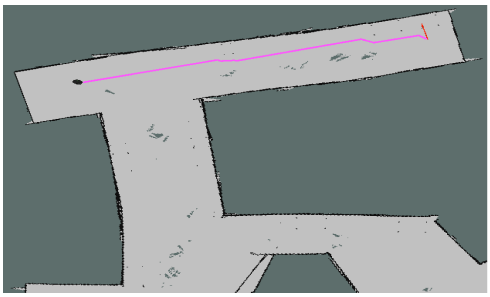
(d) Trayectoria T4 generada por Dijkstra



(e) Trayectoria T5 generada por Dijkstra



(f) Trayectoria T6 generada por Dijkstra

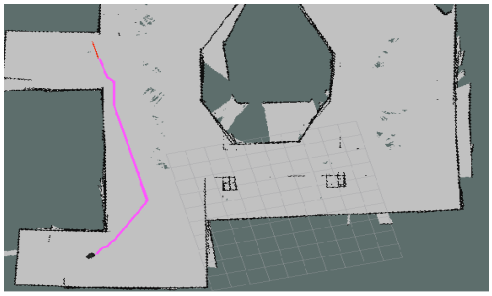


(g) Trayectoria T7 generada por Dijkstra

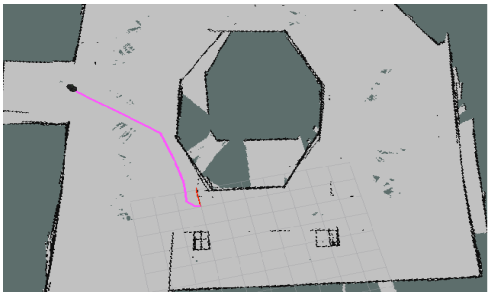
Figura 5.4: Trayectorias con Dijkstra en el mapa planta alta.

- **Algoritmo A***

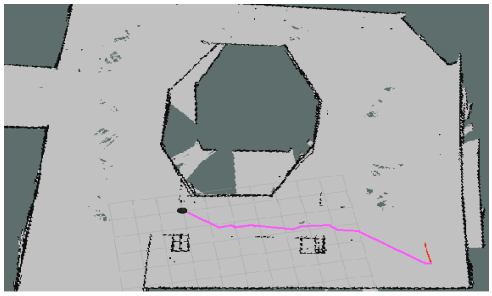
Las trayectorias generadas por el algoritmo A* se muestran en la Figura 5.5 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.



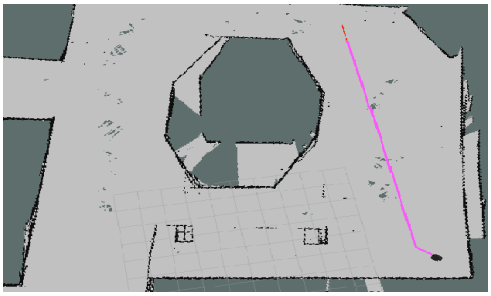
(a) Trayectoria T1 generada por A*



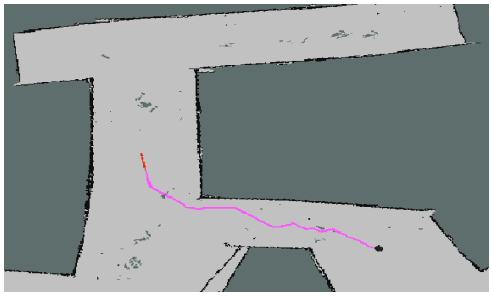
(b) Trayectoria T2 generada por A*



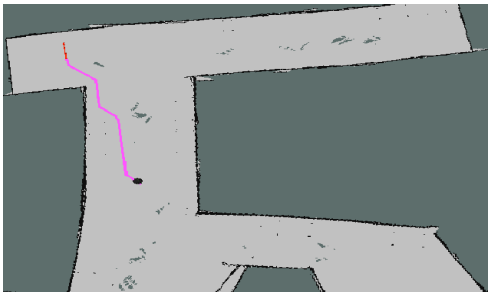
(c) Trayectoria T3 generada por A*



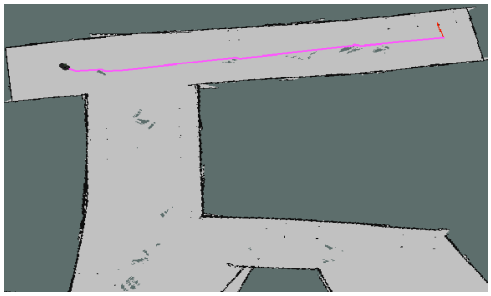
(d) Trayectoria T4 generada por A*



(e) Trayectoria T5 generada por A*



(f) Trayectoria T6 generada por A*

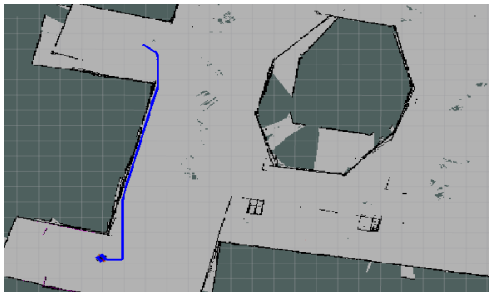


(g) Trayectoria T7 generada por A*

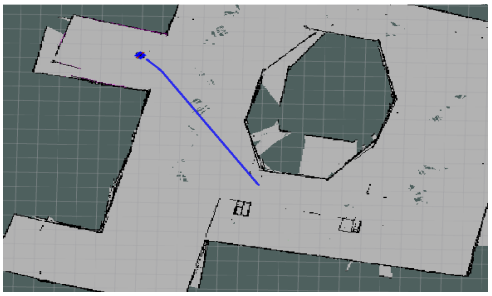
Figura 5.5: Trayectorias con A* en el mapa planta alta.

- **Algoritmo Theta***

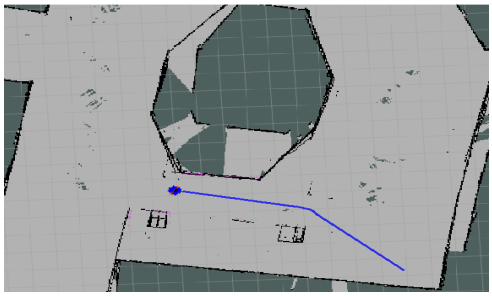
Las trayectorias generadas por el algoritmo Theta* se muestran en la Figura 5.6 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.



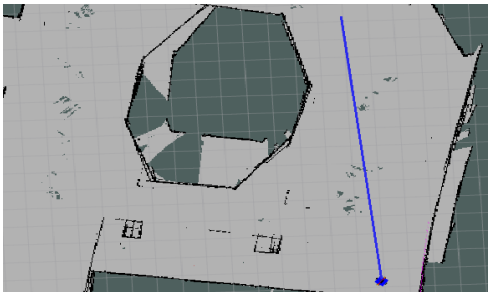
(a) Trayectoria T1 generada por Theta*



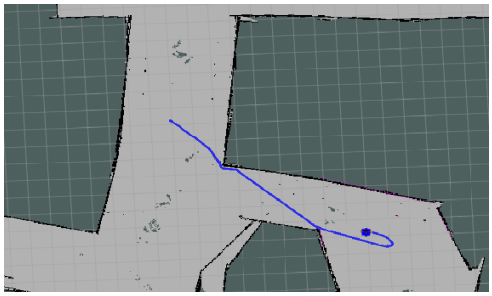
(b) Trayectoria T2 generada por Theta*



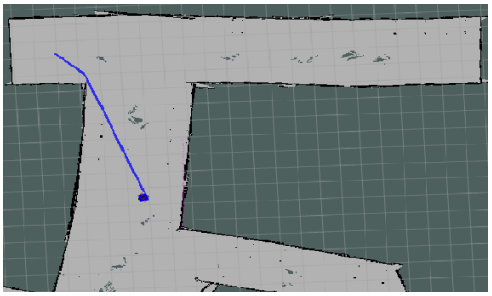
(c) Trayectoria T3 generada por Theta*



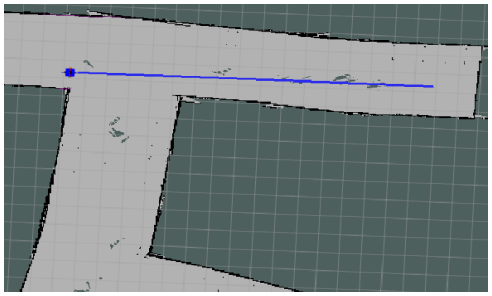
(d) Trayectoria T4 generada por Theta*



(e) Trayectoria T5 generada por Theta*



(f) Trayectoria T6 generada por Theta*



(g) Trayectoria T7 generada por Theta*

Figura 5.6: Trayectorias con Theta* en el mapa planta alta.

En la Tabla 5.1, se muestran los resultados obtenidos del desempeño de cada algoritmo en el mapa de la planta alta del Departamento de Ciencias Computacionales del CENIDET.

Tabla 5.1: Tabla de resultados de trayectorias simuladas por los tres algoritmos en el mapa planta alta.

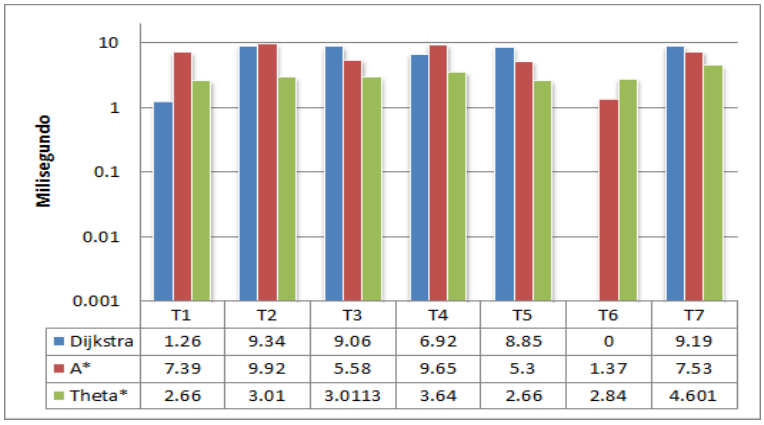
Trayectoria	**Algoritmo**	**¿Llego a la meta?**	**Tiempo en que genera la trayectoria (milisegundos)**	**Distancia de trayectoria generada (metros)**	**Distancia recorrida (metros)**
T1	Dijkstra	No	1.25	14.7109	NA
	A*	Sí	7.39	14.7109	28.0393
	Theta*	**Sí**	**2.66**	**14.4641**	
T2	Dijkstra	No	9.34	10.9263	NA
	A*	Sí	9.92	10.0667	22.9087
	Theta*	**Sí**	**3.01**	**9.43398**	
T3	Dijkstra	Sí	9.06	11.6545	15.0208
	A*	No	5.58	11.3269	NA
	Theta*	**Sí**	**3.01**	**10.4551**	**10.703**
T4	Dijkstra	No	6.92	14.8701	NA
	A*	Sí	9.65	14.8238	27.4737
	Theta*	Sí	3.64	15.0333	15.5633
T5	Dijkstra	Sí	8.85	11.5761	12.8405
	A*	Sí	5.30	12.3791	14.8335
	Theta*	Sí	2.66	13.218	
T6	Dijkstra	No generó trayectoria			
	A*	Sí	1.37	9.40602	10.0778
	Theta*	Sí	2.84	6.48683	10.0174
T7	Dijkstra	Sí	9.19	18.7943	19.7336
	A*	Sí	7.53	18.4083	22.7784
	Theta*	**Sí**	**4.60**	**17.00**	**17.0292**

A continuación se muestran tres gráficas donde se puede ver a detalle el comportamiento que tuvo cada algoritmo.

La Figura 5.7 a), muestra el tiempo que tarda cada algoritmo en generar la trayectoria y se puede observar que el algoritmo Theta* generó las siete trayectorias en un menor tiempo.

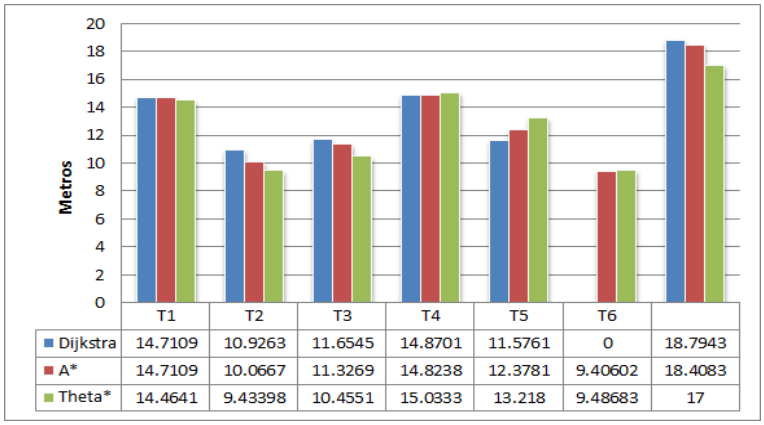
La Figura 5.7 b), muestra la distancia en metros de la trayectoria generada por cada algoritmo, en esta gráfica se observa que el algoritmo Theta* generó en la mayoría de los casos la trayectoria más corta, en el caso particular de la trayectoria (T6) el algoritmo Dijkstra no generó la trayectoria y el algoritmo A* generó la trayectoria más corta.

La Figura 5.7 c), muestra la distancia en metros que recorre cada algoritmo cuando hace el seguimiento de la trayectoria generada, se aprecia que en cuatro trayectorias (T1, T2, T4 y T6) el algoritmo Dijkstra no llegó a la ruta o no generó la trayectoria por lo cual no tiene valor, al igual que el algoritmo A* que no llegó a la meta en la trayectoria (T3).



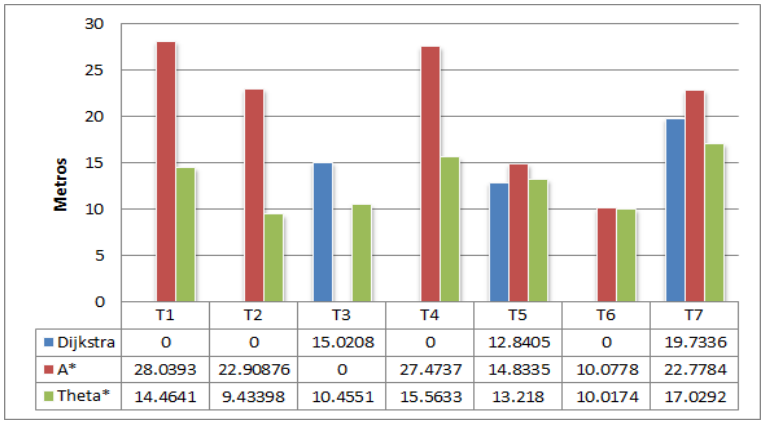
	T1	T2	T3	T4	T5	T6	T7
Dijkstra	1.26	9.34	9.06	6.92	8.85	0	9.19
A*	7.39	9.92	5.58	9.65	5.3	1.37	7.53
Theta*	2.66	3.01	3.0113	3.64	2.66	2.84	4.601

(a) Tiempo en que el algoritmo genera la trayectoria



	T1	T2	T3	T4	T5	T6	
Dijkstra	14.7109	10.9263	11.6545	14.8701	11.5761	0	18.7943
A*	14.7109	10.0667	11.3269	14.8238	12.3781	9.40602	18.4083
Theta*	14.4641	9.43398	10.4551	15.0333	13.218	9.48683	17

(b) Distancia de la trayectoria generada



	T1	T2	T3	T4	T5	T6	T7
Dijkstra	0	0	15.0208	0	12.8405	0	19.7336
A*	28.0393	22.90876	0	27.4737	14.8335	10.0778	22.7784
Theta*	14.4641	9.43398	10.4551	15.5633	13.218	10.0174	17.0292

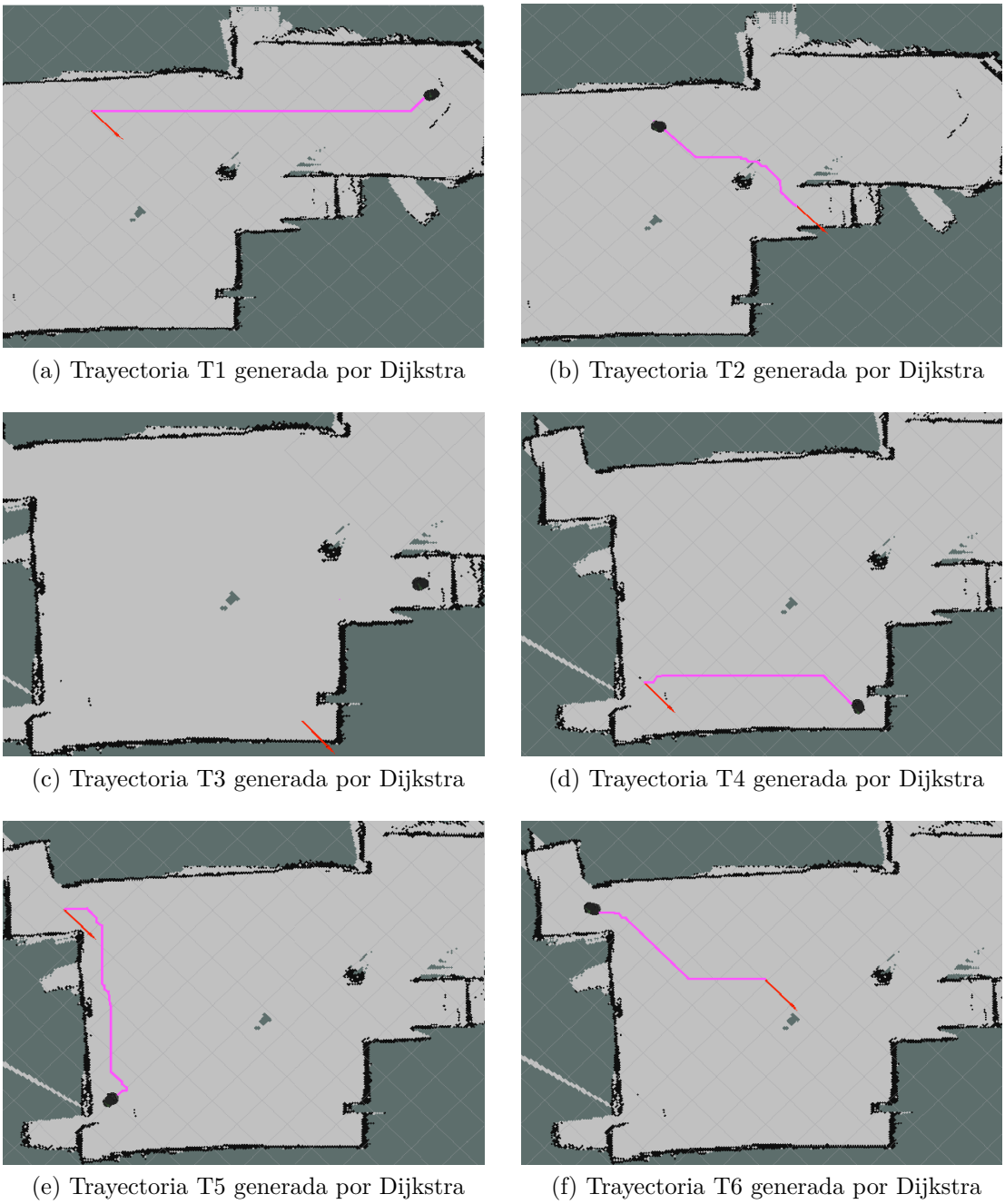
(c) Distancia de la trayectoria recorrida

Figura 5.7: Gráficas de resultados de los algoritmos en el mapa planta alta.

5.4.2. Trayectorias en el mapa planta baja

- **Algoritmo Dijkstra**

Las trayectorias generadas por el algoritmo Dijkstra se muestran en la Figura 5.8 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.



(a) Trayectoria T1 generada por Dijkstra

(b) Trayectoria T2 generada por Dijkstra

(c) Trayectoria T3 generada por Dijkstra

(d) Trayectoria T4 generada por Dijkstra

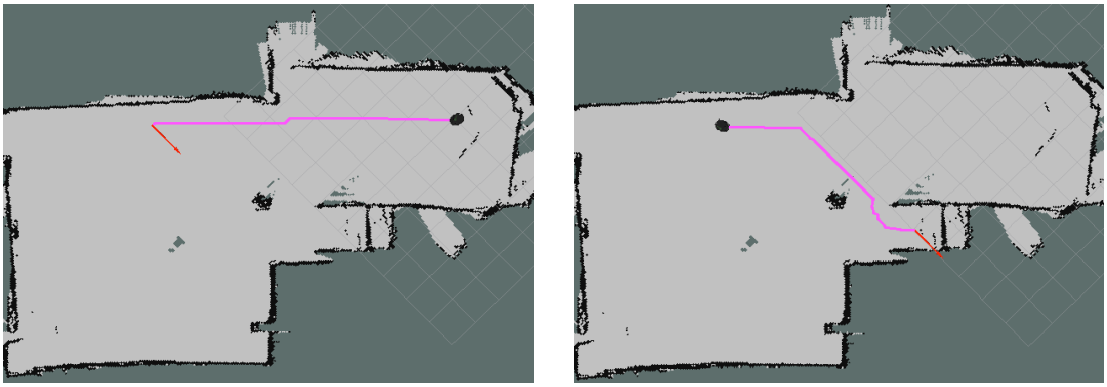
(e) Trayectoria T5 generada por Dijkstra

(f) Trayectoria T6 generada por Dijkstra

Figura 5.8: Trayectorias con Dijkstra en el mapa planta baja.

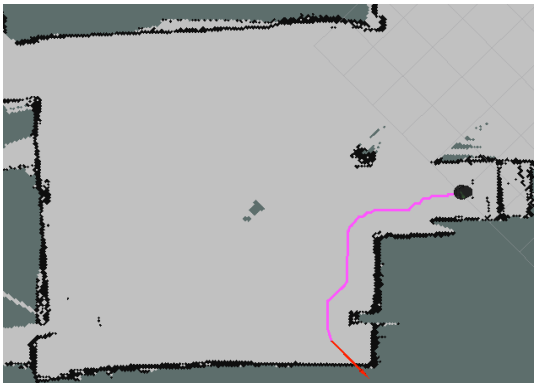
- **Algoritmo A***

Las trayectorias generadas por el algoritmo A* se muestran en la Figura 5.9 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.

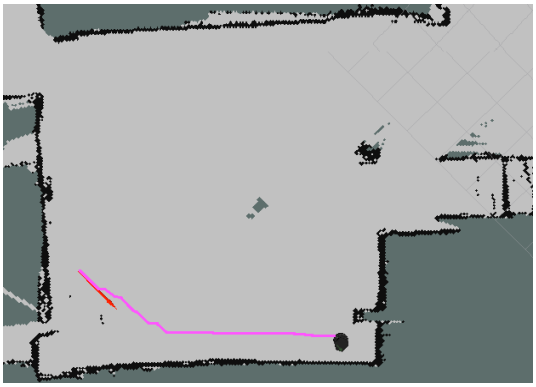


(a) Trayectoria T1 generada por A*

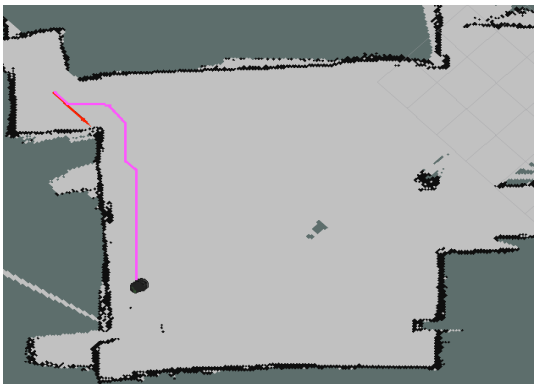
(b) Trayectoria T2 generada por A*



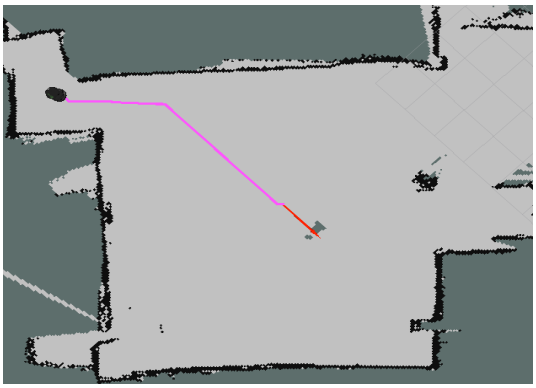
(c) Trayectoria T3 generada por A*



(d) Trayectoria T4 generada por A*



(e) Trayectoria T5 generada por A*

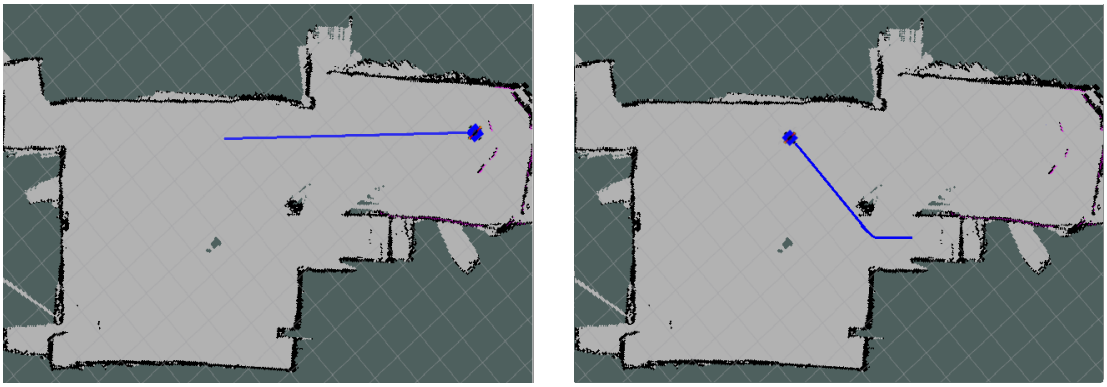


(f) Trayectoria T6 generada por A*

Figura 5.9: Trayectorias con A* en el mapa planta baja.

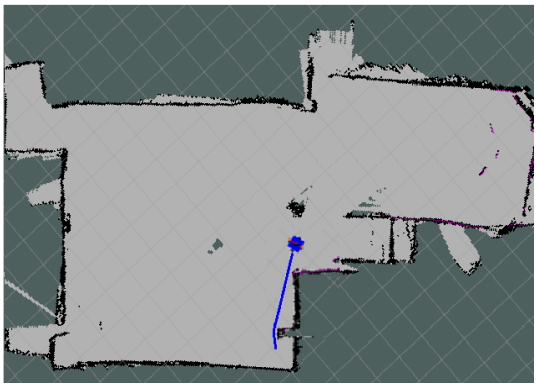
- **Algoritmo Theta***

Las trayectorias generadas por el algoritmo Theta* se muestran en la Figura 5.10 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.

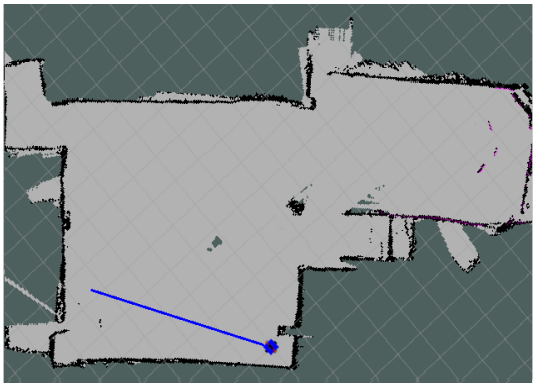


(a) Trayectoria T1 generada por Theta*

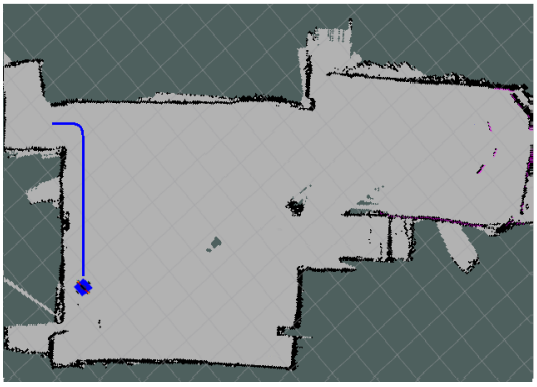
(b) Trayectoria T2 generada por Theta*



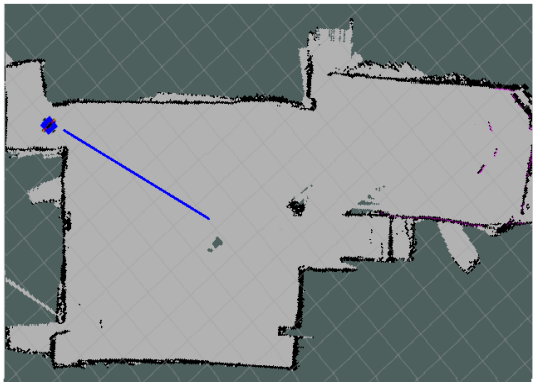
(c) Trayectoria T3 generada por Theta*



(d) Trayectoria T4 generada por Theta*



(e) Trayectoria T5 generada por Theta*



(f) Trayectoria T6 generada por Theta*

Figura 5.10: Trayectorias con Theta* en el mapa planta baja.

En la Tabla 5.2, se muestran los resultados obtenidos del desempeño de cada algoritmo en el mapa de la planta baja del Departamento de Ciencias Computacionales del CENIDET.

Tabla 5.2: Tabla de resultados de trayectorias simuladas por los tres algoritmos en el mapa planta baja.

Trayectoria	**Algoritmo**	**¿Llego a la meta?**	**Tiempo en que genera la trayectoria (milisegundos)**	**Distancia de trayectoria generada (metros)**	**Distancia recorrida (metros)**
T1	Dijkstra	Sí	6.74	10.4802	11.5277
	A*	Sí	1.872	9.95898	12.3546
	Theta*	**Sí**	**2.372**	**7.81025**	
T2	Dijkstra	Sí	6.55	5.23168	5.62904
	A*	Sí	7.27	7.08729	9.3772
	Theta*	Sí	1.661	5.42931	
T3	Dijkstra	No generó trayectoria			
	A*	No	5.49	5.05479	NA
	Theta*	**Sí**	**1.277**	**3.62996**	
T4	Dijkstra	Sí	8.04	6.59517	
	A*	Sí	1.092	6.66495	7.07034
	Theta*	**Sí**	**1.38515**	**5.65685**	
T5	Dijkstra	Sí	2.9	6.00989	12.0538
	A*	Sí	6.13	5.81086	8.18086
	Theta*	Sí	1.09411	5.99425	
T6	Dijkstra	Sí	6.97	5.60371	6.18745
	A*	Sí	1.015	6.46811	6.8111
	Theta*	**Sí**	**8.5185**	**4.12311**	

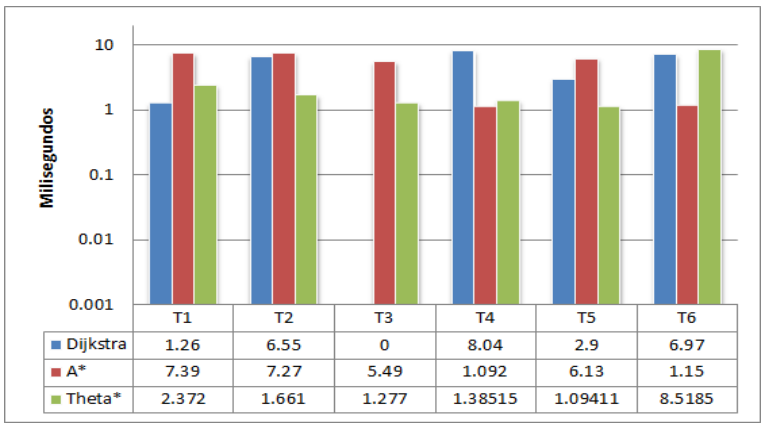
A continuación se muestran tres gráficas donde se puede ver a detalle el comportamiento que tuvo cada algoritmo.

La Figura 5.11 a), muestra el tiempo que tarda cada algoritmo en generar la trayectoria y se puede observar que el algoritmo Theta* generó las mayoría de las trayectorias en un menor tiempo.

La Figura 5.11 b), muestra la distancia en metros de la trayectoria generada por cada algoritmo, en esta gráfica se observa que el algoritmo Theta* generó en la mayoría de los casos la trayectoria más corta, excepto en la trayectoria (T5) donde el algoritmo A* fue mejor.

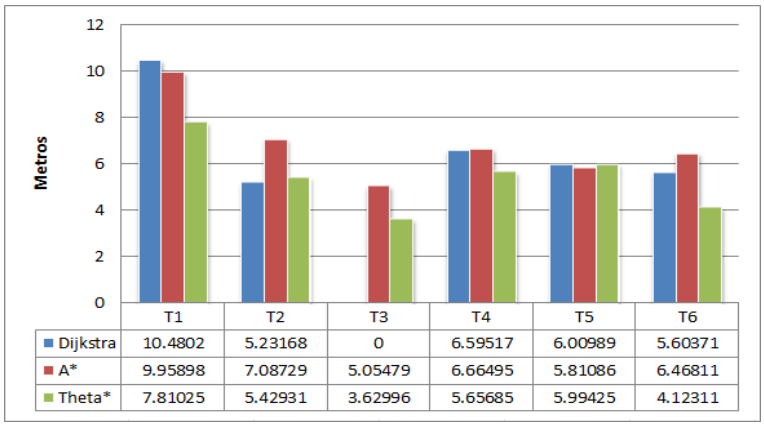
La Figura 5.11 c), muestra la distancia en metros que recorre cada algoritmo cuando hace el seguimiento de la trayectoria generada, se aprecia el algoritmo Theta* recorre menos

metros en todas las trayectorias, en el caso particular de la trayectoria (T3) A* no llegó a la meta y Dijkstra no generó trayectoria.



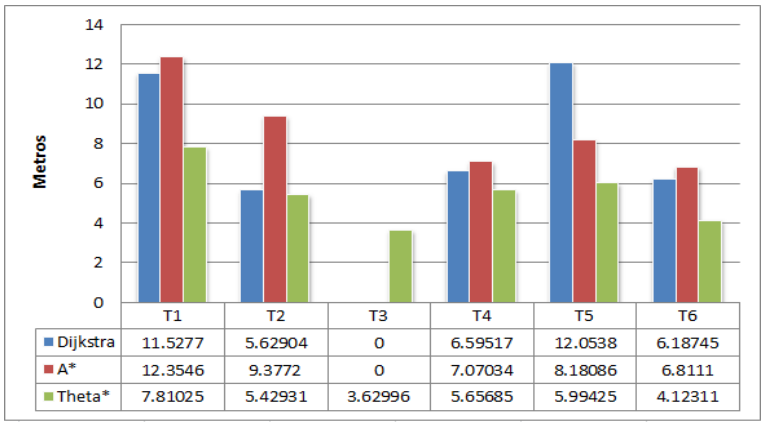
	T1	T2	T3	T4	T5	T6
Dijkstra	1.26	6.55	0	8.04	2.9	6.97
A*	7.39	7.27	5.49	1.092	6.13	1.15
Theta*	2.372	1.661	1.277	1.38515	1.09411	8.5185

(a) Tiempo en que el algoritmo genera la trayectoria



	T1	T2	T3	T4	T5	T6
Dijkstra	10.4802	5.23168	0	6.59517	6.00989	5.60371
A*	9.95898	7.08729	5.05479	6.66495	5.81086	6.46811
Theta*	7.81025	5.42931	3.62996	5.65685	5.99425	4.12311

(b) Distancia de la trayectoria generada



	T1	T2	T3	T4	T5	T6
Dijkstra	11.5277	5.62904	0	6.59517	12.0538	6.18745
A*	12.3546	9.3772	0	7.07034	8.18086	6.8111
Theta*	7.81025	5.42931	3.62996	5.65685	5.99425	4.12311

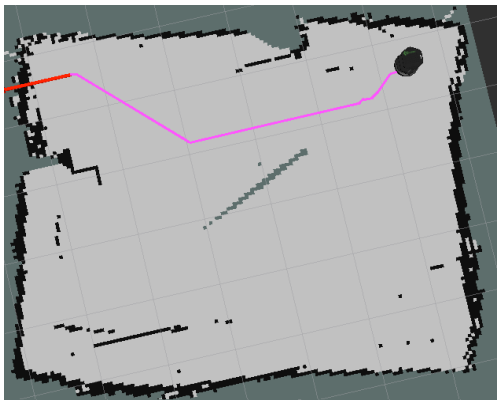
(c) Distancia de la trayectoria recorrida

Figura 5.11: Gráficas de resultados de los algoritmos en el mapa planta baja.

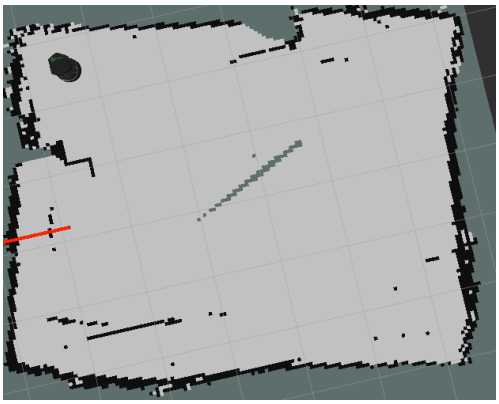
5.4.3. Trayectorias en el mapa aula 3103

- **Algoritmo Dijkstra**

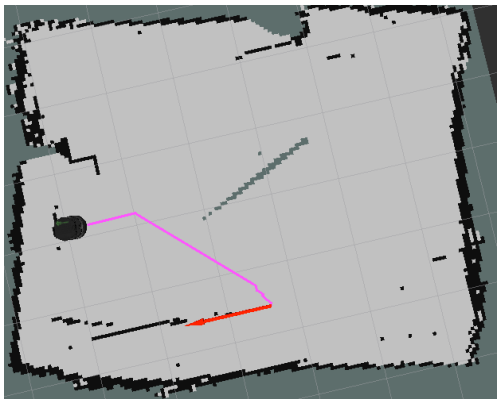
Las trayectorias generadas por el algoritmo Dijkstra se muestran en la Figura 5.12 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.



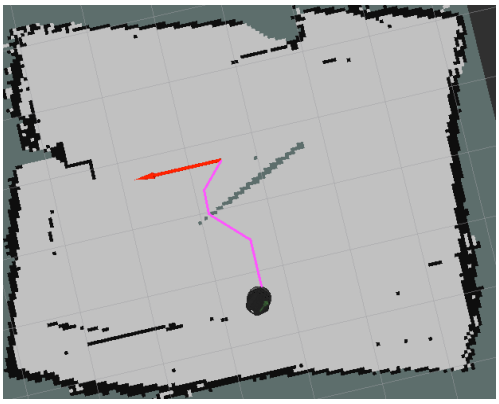
(a) Trayectoria T1 generada por Dijkstra



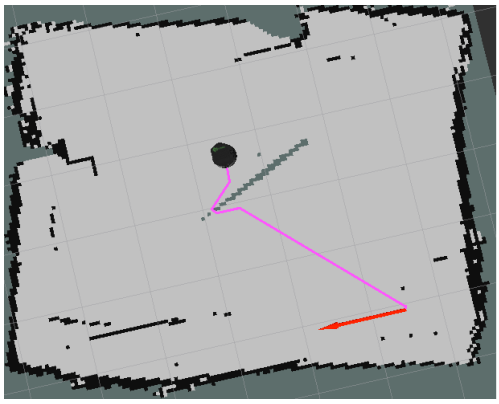
(b) Trayectoria T2 generada por Dijkstra



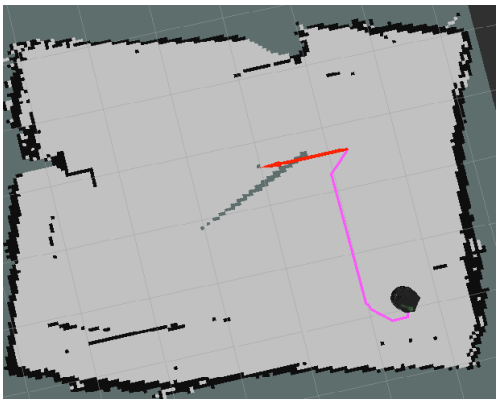
(c) Trayectoria T3 generada por Dijkstra



(d) Trayectoria T4 generada por Dijkstra



(e) Trayectoria T5 generada por Dijkstra

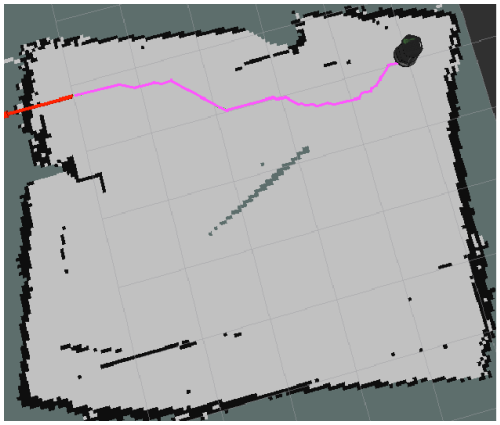


(f) Trayectoria T6 generada por Dijkstra

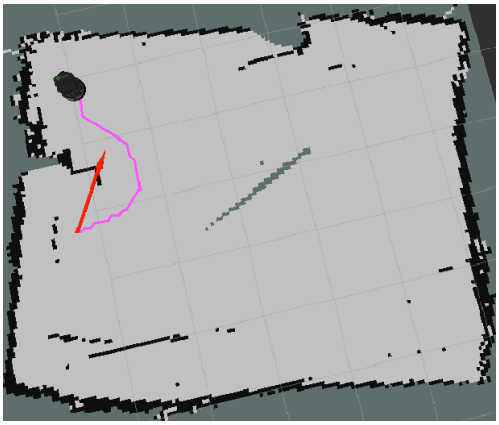
Figura 5.12: Trayectorias con Dijkstra en el mapa aula 3103.

- **Algoritmo A***

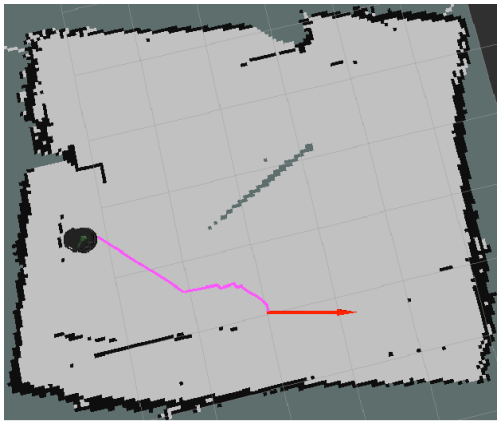
Las trayectorias generadas por el algoritmo A* se muestran en la Figura 5.13 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.



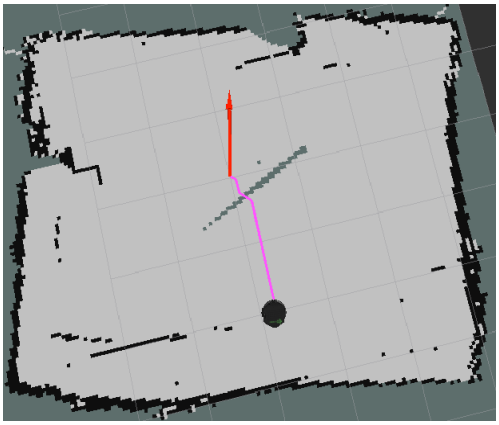
(a) Trayectoria T1 generada por A*



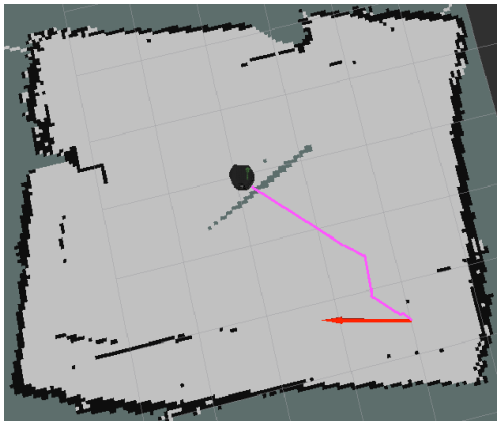
(b) Trayectoria T2 generada por A*



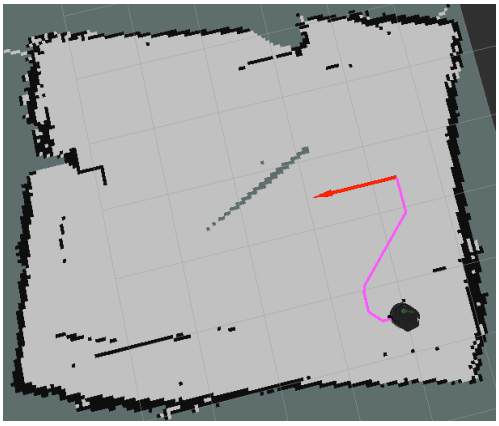
(c) Trayectoria T3 generada por A*



(d) Trayectoria T4 generada por A*



(e) Trayectoria T5 generada por A*

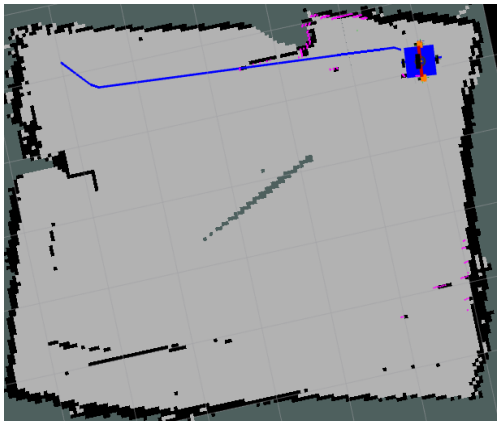


(f) Trayectoria T6 generada por A*

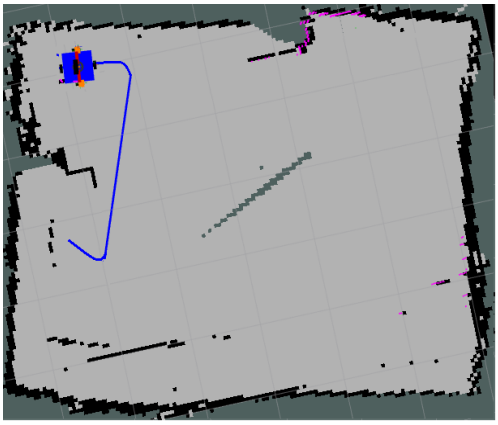
Figura 5.13: Trayectorias con A* en el mapa aula 3103.

- **Algoritmo Theta***

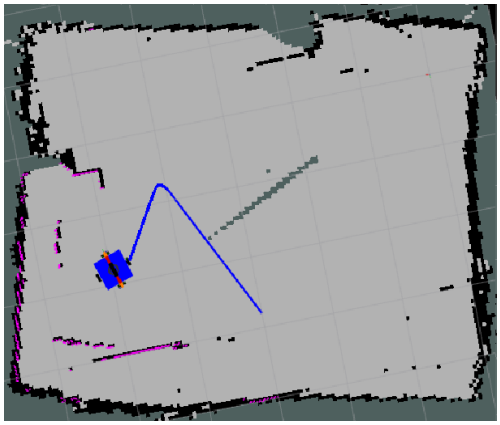
Las trayectorias generadas por el algoritmo Theta* se muestran en la Figura 5.14 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.



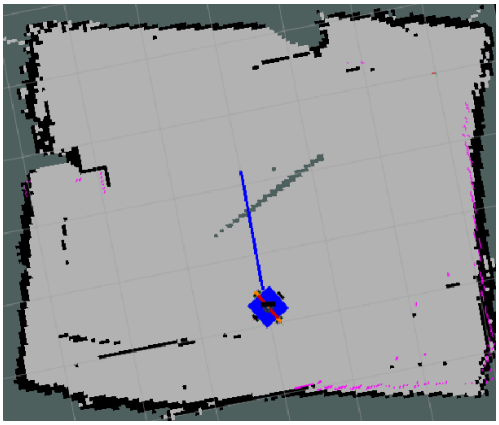
(a) Trayectoria T1 generada por Theta*



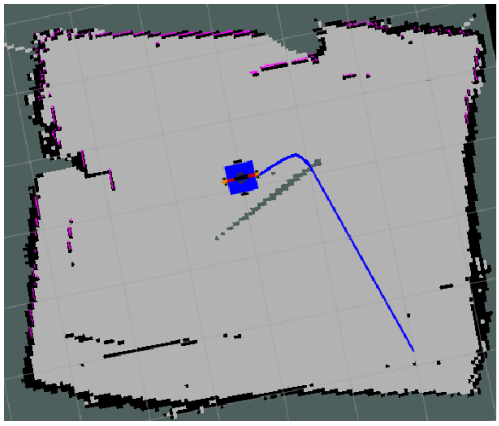
(b) Trayectoria T2 generada por Theta*



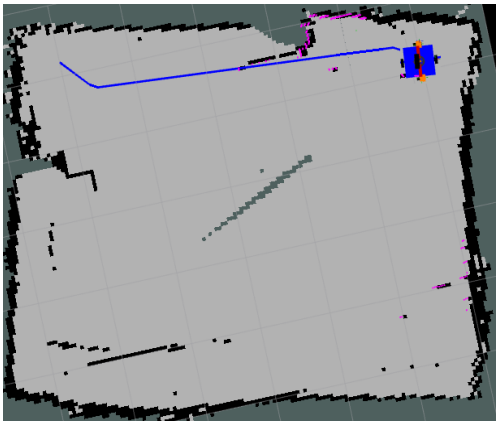
(c) Trayectoria T3 generada por Theta*



(d) Trayectoria T4 generada por Theta*



(e) Trayectoria T5 generada por Theta*



(f) Trayectoria T6 generada por Theta*

Figura 5.14: Trayectorias con Theta* en el mapa aula 3103.

En la Tabla 5.3, se muestran los resultados obtenidos del desempeño de cada algoritmo en el mapa del aula 3103 del Departamento de Ciencias Computacionales del CENIDET.

Tabla 5.3: Tabla de resultados de trayectorias simuladas por los tres algoritmos en el mapa aula 3103.

Trayectoria	**Algoritmo**	**¿Llego a la meta?**	**Tiempo en que genera la trayectoria (milisegundos)**	**Distancia de trayectoria generada (metros)**	**Distancia recorrida (metros)**
T1	Dijkstra	Sí	9.67	5.16511	5.46391
	A*	Sí	6.315	5.18738	5.28849
	Theta*	Sí	9.2777	5.26958	5.33318
T2	Dijkstra	No generó trayectoria			
	A*	Sí	1.019	2.06897	3.45146
	Theta*	Sí	7.7216	3.16228	
T3	Dijkstra	Sí	7.19	3.18792	4.31111
	A*	Sí	7.21	3.18792	3.58249
	Theta*	**Sí**	**6.176**	**2.23607**	
T4	Dijkstra	Sí	9.82	3.18792	10.5334
	A*	Sí	7.41	2.16351	3.0407
	Theta*	**Sí**	**4.787**	**1.00**	
T5	Dijkstra	Sí	8.2	3.62468	8.28849
	A*	Sí	9.12	3.30884	3.6625
	Theta*	Sí	1.56	4.47214	4.64116
T6	Dijkstra	Sí	9.4	2.87957	6.06248
	A*	Sí	1.181	2.09389	2.6604
	Theta*	**Sí**	**5.8131**	**2.00**	

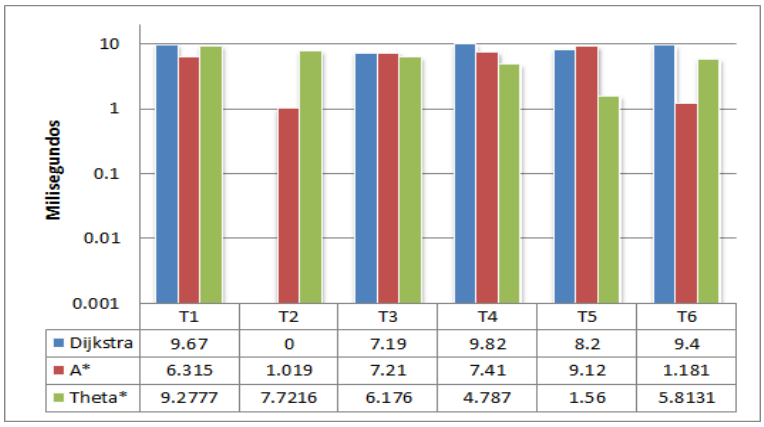
A continuación se muestran tres gráficas donde se puede ver a detalle el comportamiento que tuvo cada algoritmo.

La Figura 5.15 a), muestra el tiempo que tarda cada algoritmo en generar la trayectoria y se puede observar que el algoritmo Theta* generó las trayectorias en un menor tiempo a comparación de los demás algoritmos.

La Figura 5.15 b), muestra la distancia en metros de la trayectoria generada por cada algoritmo, en esta gráfica se observa que Theta* generó en tres trayectorias menos metros, excepto en la trayectoria (T1) donde Dijkstra generó la menor distancia, aunque no generó la trayectoria (T2), en el caso de la trayectoria (T3) Dijkstra y A* tuvieron la misma distancia generada y A* tuvo el mismo comportamiento en la trayectoria (T2) y (T5).

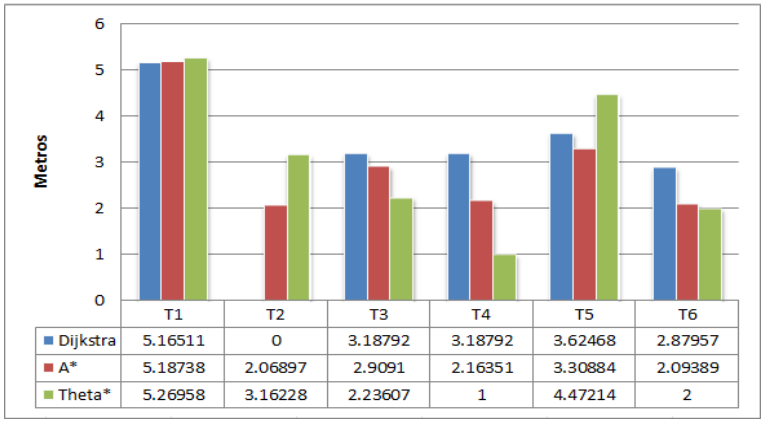
La Figura 5.15 c), muestra la distancia en metros que recorre cada algoritmo cuando hace el seguimiento de la trayectoria generada, el algoritmo Dijkstra recorrió más metros en

las trayectorias (T4, T5 y T6), el algoritmo Theta* fue el algoritmo que en la mayoría de los casos recorrió los mismos metros que marca la trayectoria generada, incluso en la trayectoria (T2) donde A* había destacado por generar la trayectoria más corta.



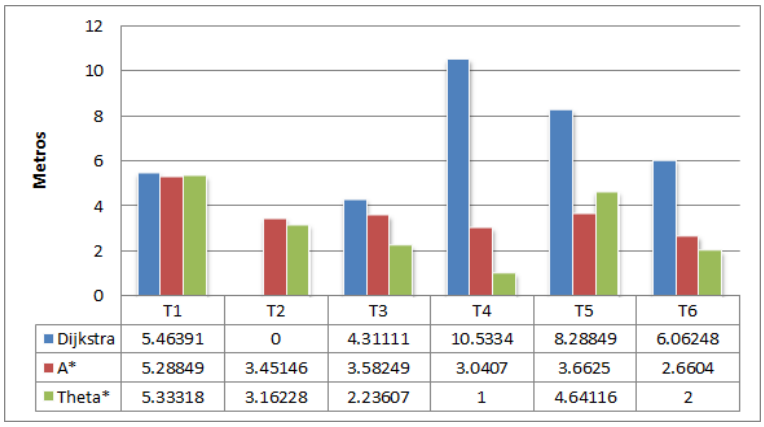
	T1	T2	T3	T4	T5	T6
Dijkstra	9.67	0	7.19	9.82	8.2	9.4
A*	6.315	1.019	7.21	7.41	9.12	1.181
Theta*	9.2777	7.7216	6.176	4.787	1.56	5.8131

(a) Tiempo en que el algoritmo genera la trayectoria



	T1	T2	T3	T4	T5	T6
Dijkstra	5.16511	0	3.18792	3.18792	3.62468	2.87957
A*	5.18738	2.06897	2.9091	2.16351	3.30884	2.09389
Theta*	5.26958	3.16228	2.23607	1	4.47214	2

(b) Distancia de la trayectoria generada



	T1	T2	T3	T4	T5	T6
Dijkstra	5.46391	0	4.31111	10.5334	8.28849	6.06248
A*	5.28849	3.45146	3.58249	3.0407	3.6625	2.6604
Theta*	5.33318	3.16228	2.23607	1	4.64116	2

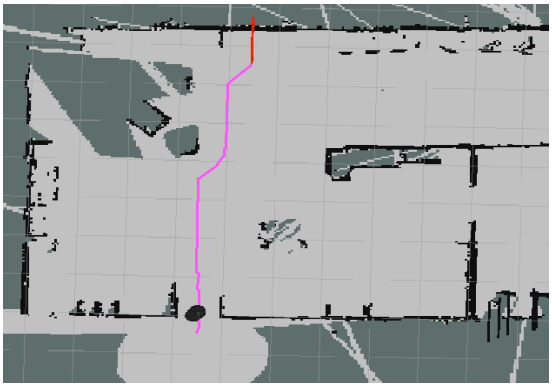
(c) Distancia de la trayectoria recorrida

Figura 5.15: Gráficas de resultados de los algoritmos en el mapa aula 3103.

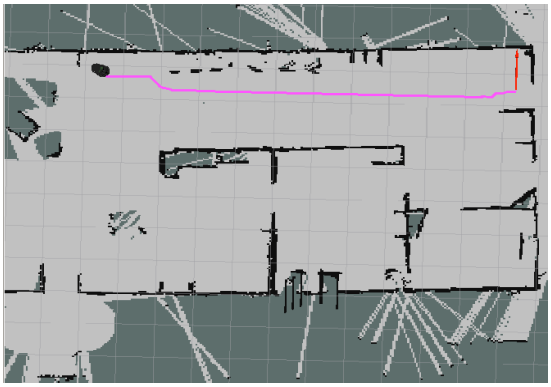
5.4.4. Trayectorias en el mapa arena_a

- **Algoritmo Dijkstra**

Las trayectorias generadas por el algoritmo Dijkstra se muestran en la Figura 5.16 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.

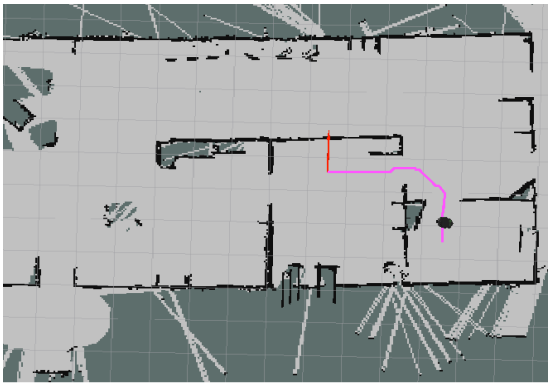


(a) Trayectoria T1 generada por Dijkstra

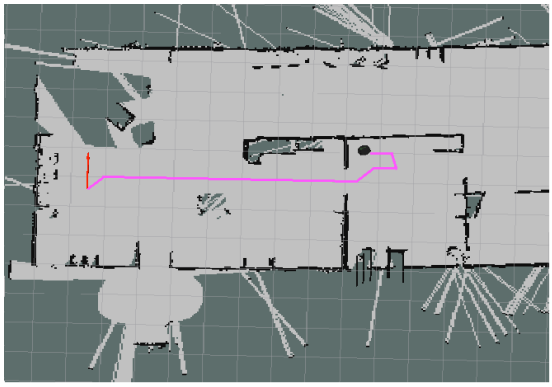


(b) Trayectoria T2 generada por Dijkstra

(c) Trayectoria T3 generada por Dijkstra



(d) Trayectoria T4 generada por Dijkstra

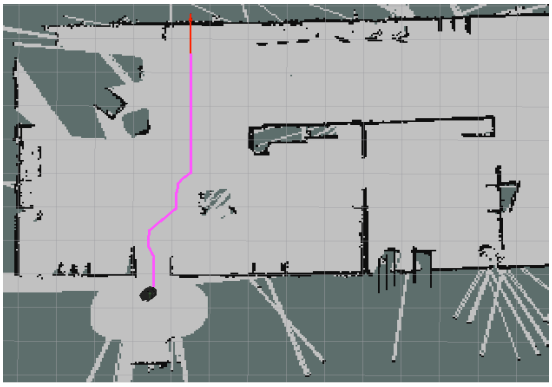


(e) Trayectoria T5 generada por Dijkstra

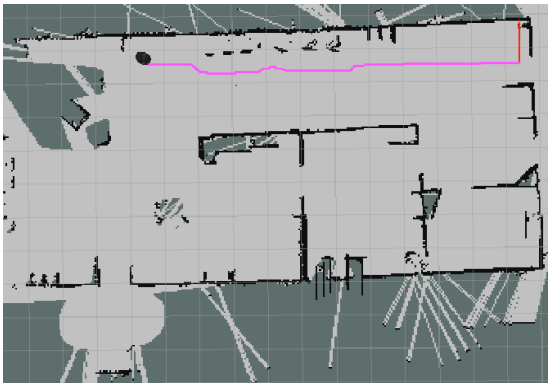
Figura 5.16: Trayectorias con Dijkstra en el mapa arena_a.

- **Algoritmo A***

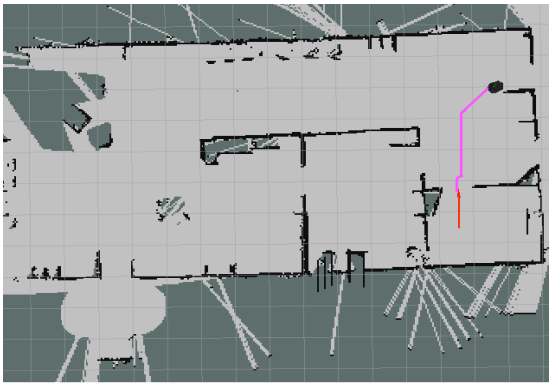
Las trayectorias generadas por el algoritmo A* se muestran en la Figura 5.17 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.



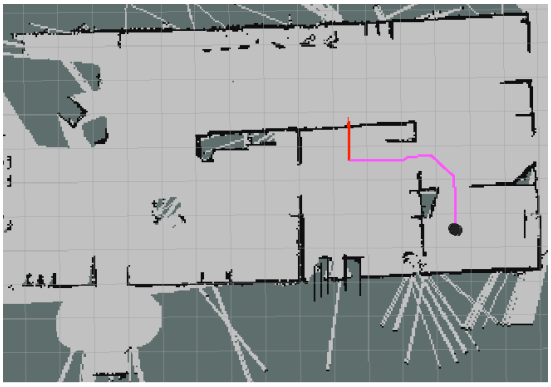
(a) Trayectoria T1 generada por A*



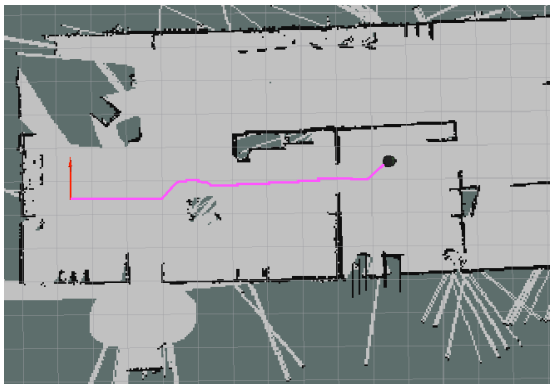
(b) Trayectoria T2 generada por A*



(c) Trayectoria T3 generada por A*



(d) Trayectoria T4 generada por A*

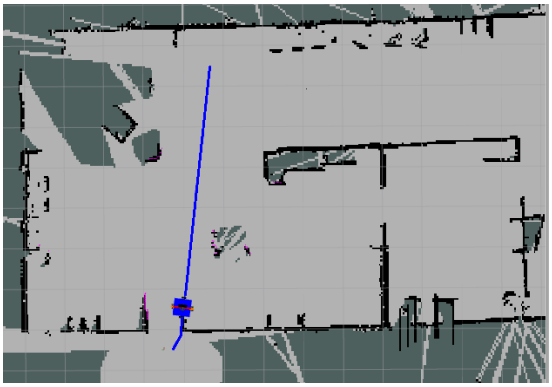


(e) Trayectoria T5 generada por A*

Figura 5.17: Trayectorias con A* en el mapa arena_a.

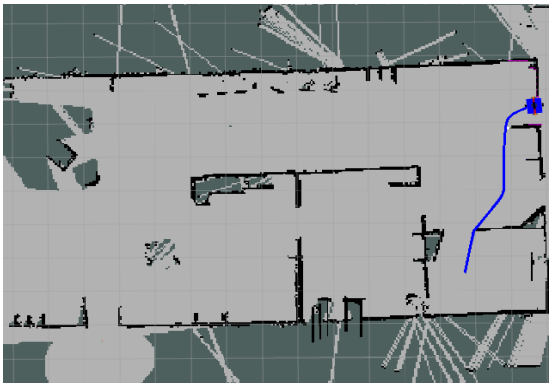
- **Algoritmo Theta***

Las trayectorias generadas por el algoritmo Theta* se muestran en la Figura 5.18 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.

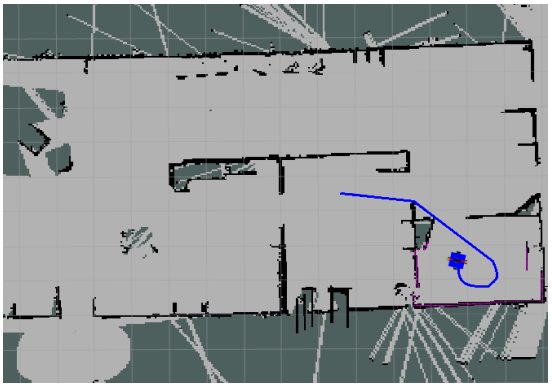


(a) Trayectoria T1 generada por Theta*

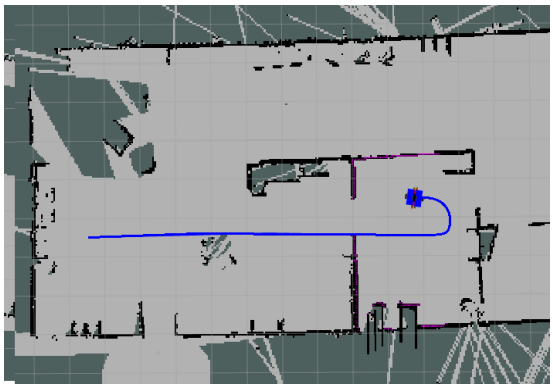
(b) Trayectoria T2 generada por Theta*



(c) Trayectoria T3 generada por Theta*



(d) Trayectoria T4 generada por Theta*



(e) Trayectoria T5 generada por Theta*

Figura 5.18: Trayectorias con Theta* en el mapa arena_a.

En la Tabla 5.4, se muestran los resultados obtenidos del desempeño de cada algoritmo en el mapa de la planta alta del Departamento de Ciencias Computacionales del CENIDET.

Tabla 5.4: Tabla de resultados de trayectorias simuladas por los tres algoritmos en el mapa arena_a.

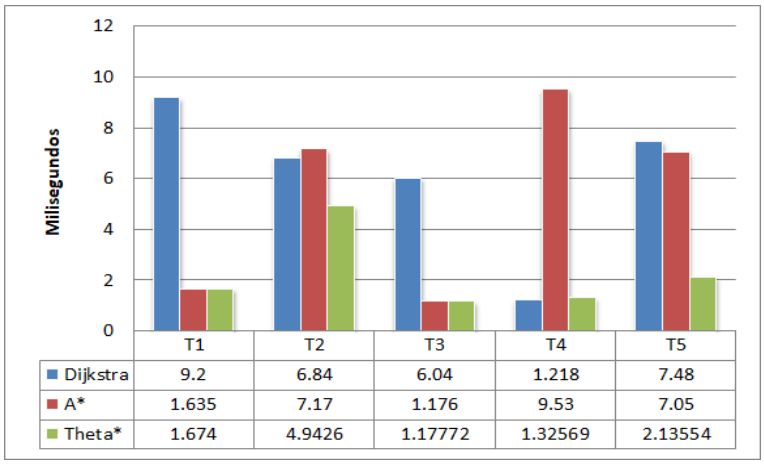
Trayectoria	**Algoritmo**	**¿Llegó a la meta?**	**Tiempo en que genera la trayectoria (milisegundos)**	**Distancia de trayectoria generada (metros)**	**Distancia recorrida (metros)**
T1	Dijkstra	Sí	9.2	7.43722	7.82629
	A*	Sí	1.635	7.42353	36.0812
	Theta*	**Sí**	**1.674**	**7.07107**	**7.10106**
T2	Dijkstra	Sí	6.84	11.6155	25.0807
	A*	Sí	7.17	11.6580	23.2019
	Theta*	**Sí**	**4.9426**	**11.0454**	
T3	Dijkstra	Sí	6.04	5.53045	6.56055
	A*	Sí	1.176	5.53545	6.29352
	Theta*	**Sí**	**1.17772**	**4.53585**	
T4	Dijkstra	No	1.218	3.83625	NA
	A*	Sí	9.53	3.87071	5.23365
	Theta*	Sí	1.32569	4.64061	
T5	Dijkstra	Sí	7.48	9.59676	11.0094
	A*	Sí	7.05	9.73323	11.8854
	Theta*	Sí	2.13554	10.002	

A continuación se muestran tres gráficas donde se puede ver a detalle el comportamiento que tuvo cada algoritmo.

La Figura 5.19 a), muestra el tiempo que tarda cada algoritmo en generar la trayectoria y se puede observar que el algoritmo Theta* generó las cinco trayectorias en un menor tiempo.

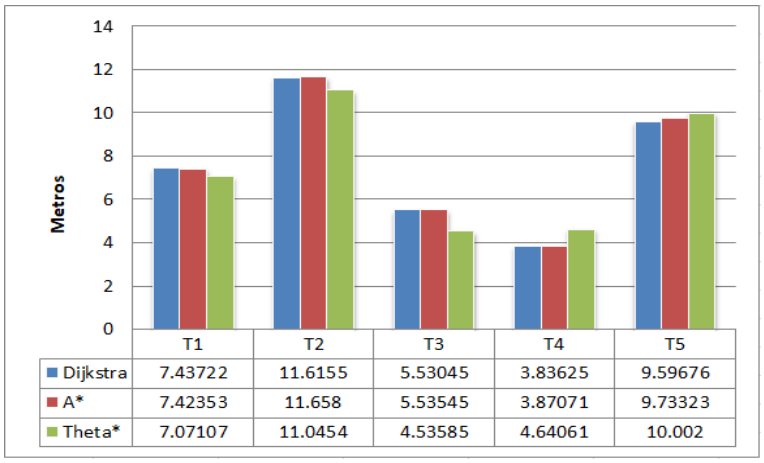
La Figura 5.19 b), muestra la distancia en metros de la trayectoria generada por cada algoritmo, en esta gráfica se observa que el algoritmo Theta* generó las trayectorias (T1, T2 y T3) más cortas y Dijkstra fue mejor que Theta* en las trayectorias restantes.

La Figura 5.19 c), muestra la distancia en metros que recorre cada algoritmo cuando hace el seguimiento de la trayectoria generada, se aprecia que en algoritmo Theta* recorrió menos metros en todas las trayectorias, también se puede ver que el algoritmo Dijkstra no llegó a la meta en la trayectoria (T4).



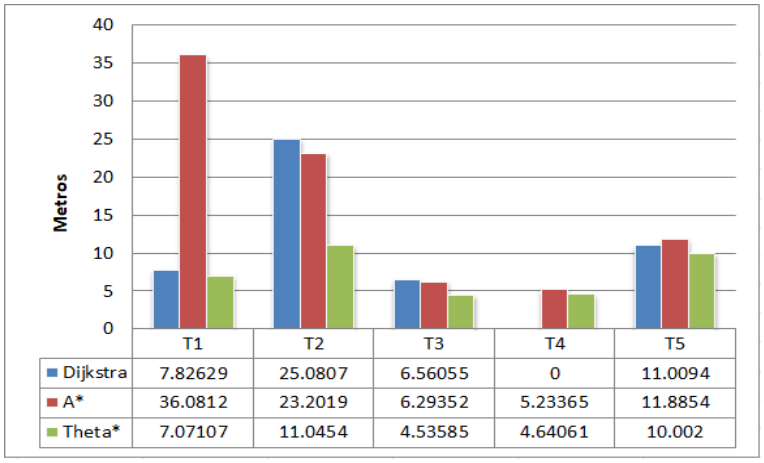
	T1	T2	T3	T4	T5
Dijkstra	9.2	6.84	6.04	1.218	7.48
A*	1.635	7.17	1.176	9.53	7.05
Theta*	1.674	4.9426	1.17772	1.32569	2.13554

(a) Tiempo en que el algoritmo genera la trayectoria



	T1	T2	T3	T4	T5
Dijkstra	7.43722	11.6155	5.53045	3.83625	9.59676
A*	7.42353	11.658	5.53545	3.87071	9.73323
Theta*	7.07107	11.0454	4.53585	4.64061	10.002

(b) Distancia de la trayectoria generada



	T1	T2	T3	T4	T5
Dijkstra	7.82629	25.0807	6.56055	0	11.0094
A*	36.0812	23.2019	6.29352	5.23365	11.8854
Theta*	7.07107	11.0454	4.53585	4.64061	10.002

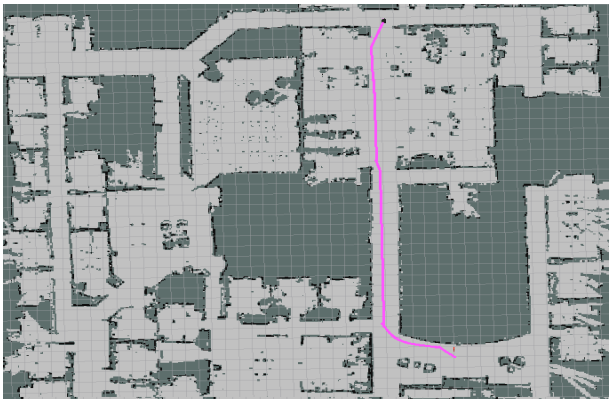
(c) Distancia de la trayectoria recorrida

Figura 5.19: Gráficas de resultados de los algoritmos en el mapa arena_a.

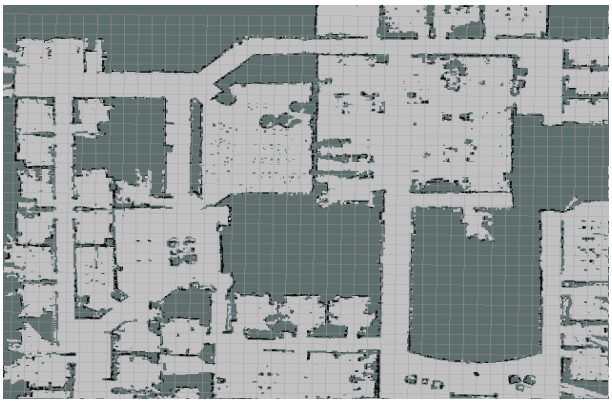
5.4.5. Trayectorias en el mapa willow_garage_map

- **Algoritmo Dijkstra**

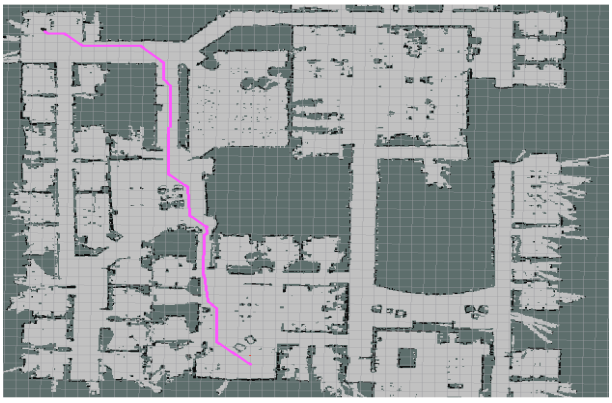
Las trayectorias generadas por el algoritmo Dijkstra se muestran en la Figura 5.20 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.



(a) Trayectoria T1 generada por Dijkstra



(b) Trayectoria T2 generada por Dijkstra



(c) Trayectoria T3 generada por Dijkstra

Figura 5.20: Trayectorias con Dijkstra en el mapa willow_garage_map.

- **Algoritmo A***

Las trayectorias generadas por el algoritmo A* se muestran en la Figura 5.21 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.

(a) Trayectoria T1 generada por A*

(b) Trayectoria T2 generada por A*

(c) Trayectoria T3 generada por A*

Figura 5.21: Trayectorias con A* en el mapa willow_garage_map.

- **Algoritmo Theta***

Las trayectorias generadas por el algoritmo Theta* se muestran en la Figura 5.22 donde el robot marca el punto inicial y el inicio de la flecha el punto meta.

(a) Trayectoria T1 generada por Theta*

(b) Trayectoria T2 generada por Theta*

(c) Trayectoria T3 generada por Theta*

Figura 5.22: Trayectorias con Theta* en el mapa willow_garage_map.

En la Tabla 5.5, se muestran los resultados obtenidos del desempeño de cada algoritmo en el mapa de la planta alta del Departamento de Ciencias Computacionales del CENIDET.

Tabla 5.5: Tabla de resultados de trayectorias simuladas por los tres algoritmos en el mapa willow_garage_map.

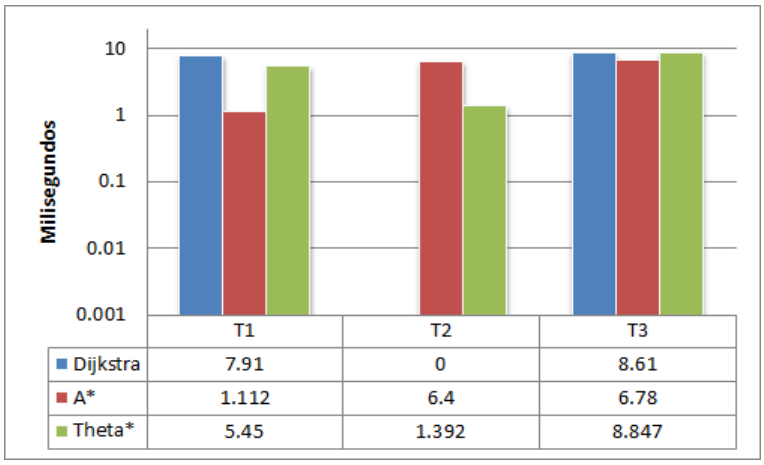
Trayectoria	**Algoritmo**	**¿Llego a la meta?**	**Tiempo en que genera la trayectoria (milisegundos)**	**Distancia de trayectoria generada (metros)**	**Distancia recorrida (metros)**
T1	Dijkstra	No	7.91	16.9491	NA
	A*	Sí	1.11	17.1586	23.4382
	Theta*	Sí	5.45	42.9546	
T2	Dijkstra	No generó trayectoria			
	A*	No	6.40	23.3219	NA
	Theta*	Sí	1.392	59.26	61.6098
T3	Dijkstra	No	8.61	21.5581	NA
	A*	No	6.78	21.4901	NA
	Theta*	Sí	8.85	49.096	

A continuación se muestran tres gráficas donde se puede ver a detalle el comportamiento que tuvo cada algoritmo.

La Figura 5.23 a), muestra el tiempo que tarda cada algoritmo en generar la trayectoria y se puede observar que el algoritmo Dijkstra generó las trayectorias en un menor tiempo.

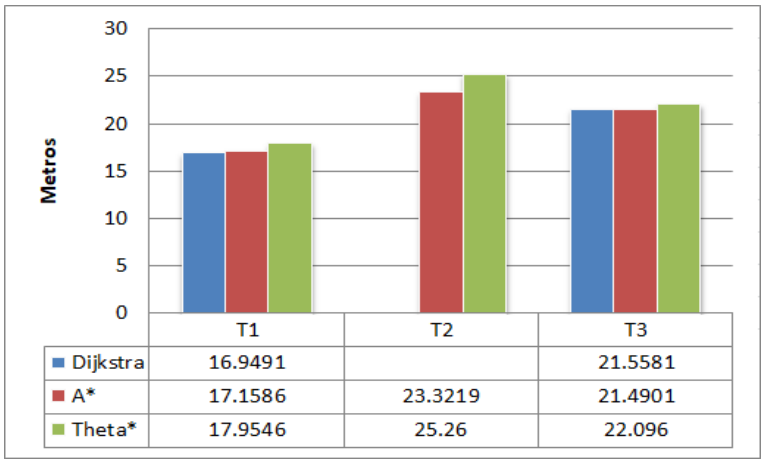
La Figura 5.23 b), muestra la distancia en metros de la trayectoria generada por cada algoritmo, en esta gráfica se observa que la trayectoria (T1) el algoritmo Dijkstra generó la trayectoria más corta, sin embargo, no generó la trayectoria (T2), y que Theta* fue el algoritmo que generó las trayectorias más largas en las tres ocasiones.

La Figura 5.23 c), muestra la distancia en metros que recorre cada algoritmo cuando hace el seguimiento de la trayectoria generada, se puede apreciar que el algoritmo Dijkstra no generó la trayectoria (T2) y no llegó a la meta en las trayectorias (T1) y (T3), el algoritmo A* solo llegó a la meta en la trayectoria (T1) y la diferencia de los metros generados a los metros recorridos fue de 6.2796, finalmente Theta* el cual llegó a la meta en todas las ocasiones.



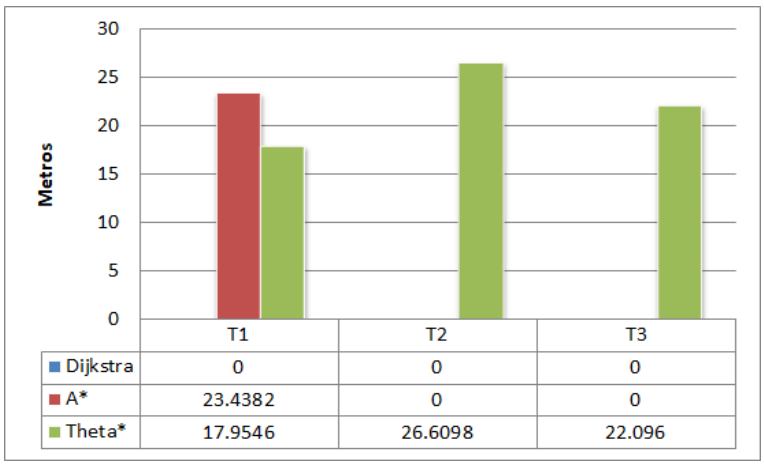
	T1	T2	T3
Dijkstra	7.91	0	8.61
A*	1.112	6.4	6.78
Theta*	5.45	1.392	8.847

(a) Tiempo en que el algoritmo genera la trayectoria



	T1	T2	T3
Dijkstra	16.9491		21.5581
A*	17.1586	23.3219	21.4901
Theta*	17.9546	25.26	22.096

(b) Distancia de la trayectoria generada



	T1	T2	T3
Dijkstra	0	0	0
A*	23.4382	0	0
Theta*	17.9546	26.6098	22.096

(c) Distancia de la trayectoria recorrida

Figura 5.23: Gráficas de resultados de los algoritmos en el mapa willow_garage.

La Tabla 5.6, muestra el desempeño general de los tres algoritmos, donde podemos concluir de forma general que a pesar de que el algoritmo Theta* generó trayectorias más largas o en mayor tiempo comparado con los resultados de los otros dos algoritmos, fue el algoritmo que en todas las ocasiones llegó a la meta.

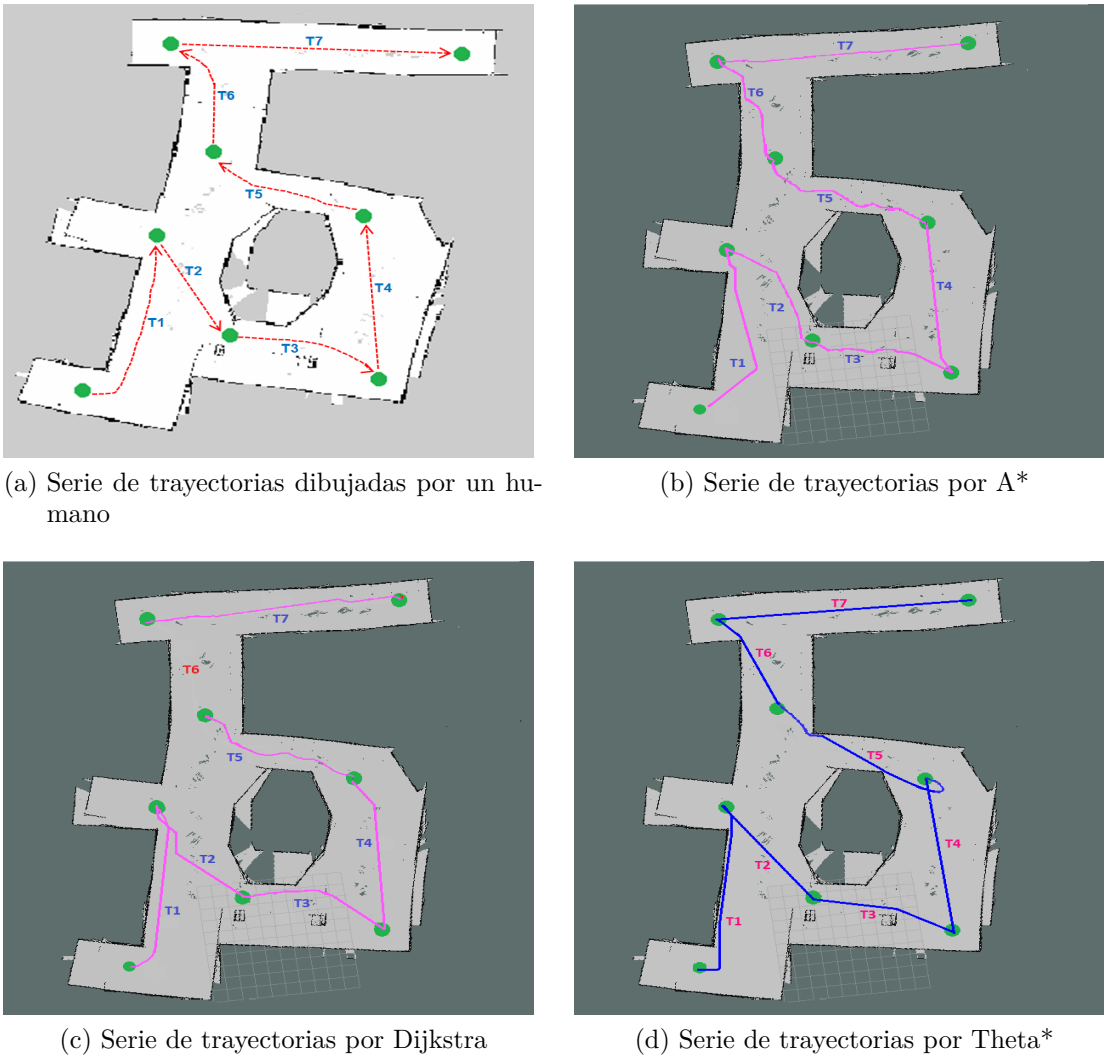
Tabla 5.6: Tabla general de resultados.

Mapa	**Algoritmo**	**Veces que generó trayectoria**	**Veces que llegó a la meta**	**Suma de tiempos de secuencias generadas (milisegundos)**	**Total de metros generados**	**Total de metros recorridos**
Planta Alta	Dijkstra	6/7= 85 %	3/7=43 %	44.62	82.5322	47.5949
	A*	6/7=85 %	6/7=85 %	46.74	91.1272	126.1114
	Theta*	**7/7=100 %**	**7/7=100 %**	**22.4223**	**89.09131**	**90.42898**
Planta Baja	Dijkstra	5/6=83 %	5/6=83 %	25.72	33.92065	41.99316
	A*	6/6=100 %	5/6=83 %	28.522	41.03598	43.7941
	Theta*	**6/6=100 %**	**6/6=100 %**	**16.30776**	**32.64373**	**32.64373**
Aula 3103	Dijkstra	5/6=83 %	5/6=83 %	44.28	17.37307	34.75939
	A*	**6/6=100 %**	**6/6=100 %**	**32.255**	**17.73169**	**21.68604**
	Theta*	**6/6=100 %**	**6/6=100 %**	**35.3354**	**18.20367**	**18.30909**
Arena_a	Dijkstra	5/5=100 %	4/5=80 %	30.778	38.01618	50.47694
	A*	5/5=100 %	5/5=100 %	26.561	38.26092	82.69567
	Theta*	**5/5=100 %**	**5/5=100 %**	**11.25555**	**37.30493**	**37.33492**
Willow_garage	Dijkstra	2/3=67 %	0/3=0 %	16.52	38.5072	0
	A*	3/3=100 %	1/3=33 %	14.292	61.97.06	23.4382
	Theta*	**3/3=100 %**	**3/3=100 %**	**15.689**	**65.3106**	**66.6604**

5.5. Comparativa de trayectorias

Cada algoritmo generó diferentes trayectorias, sin embargo, cuando se diseñó la serie de trayectorias se dibujaron tomando en cuenta la pregunta ¿qué camino tomaría un humano? A continuación se muestra una comparativa de las trayectorias que dibujo el humano vs. como las generó el algoritmo.

El conjunto de trayectorias generadas por los algoritmos Dijkstra, A* y Theta* en el mapa de la planta alta del DCC, se muestran en la Figura 5.24.



(a) Serie de trayectorias dibujadas por un humano

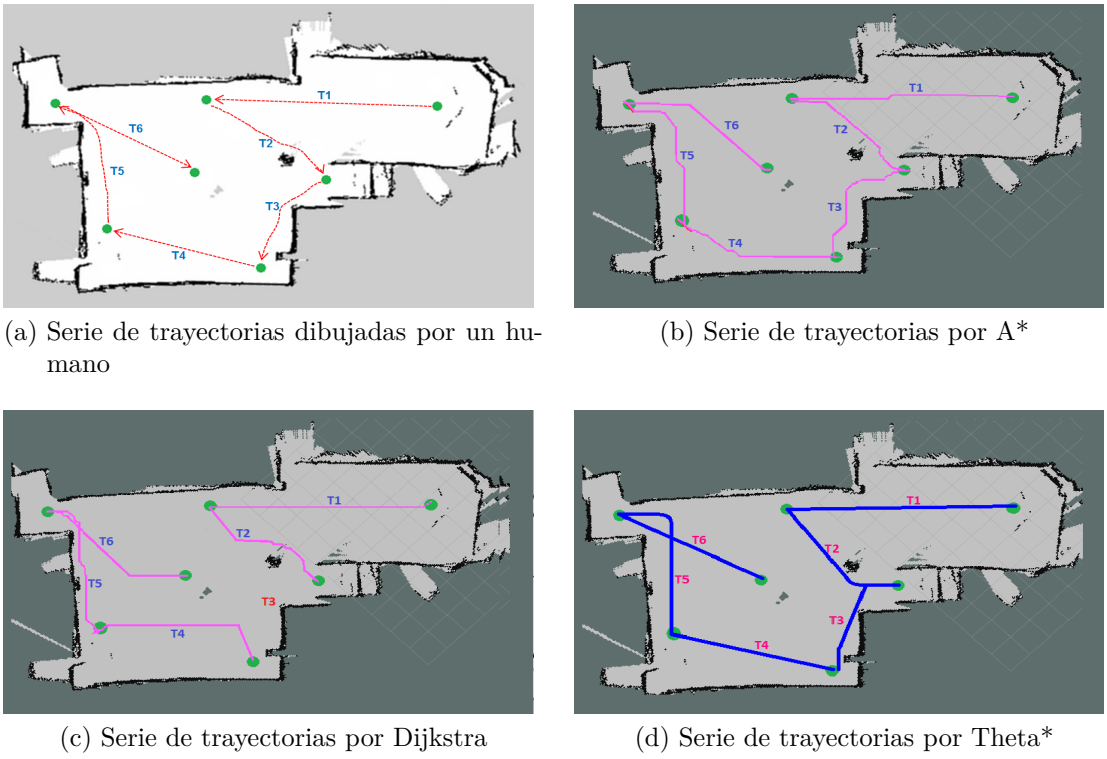
(b) Serie de trayectorias por A*

(c) Serie de trayectorias por Dijkstra

(d) Serie de trayectorias por Theta*

Figura 5.24: Comparación de trayectorias en el mapa planta alta, humavo vs. algoritmos.

El conjunto de trayectorias generadas por los algoritmos Dijkstra, A* y Theta* en el mapa de la planta baja del DCC, se muestran en la Figura 5.25.



(a) Serie de trayectorias dibujadas por un humano

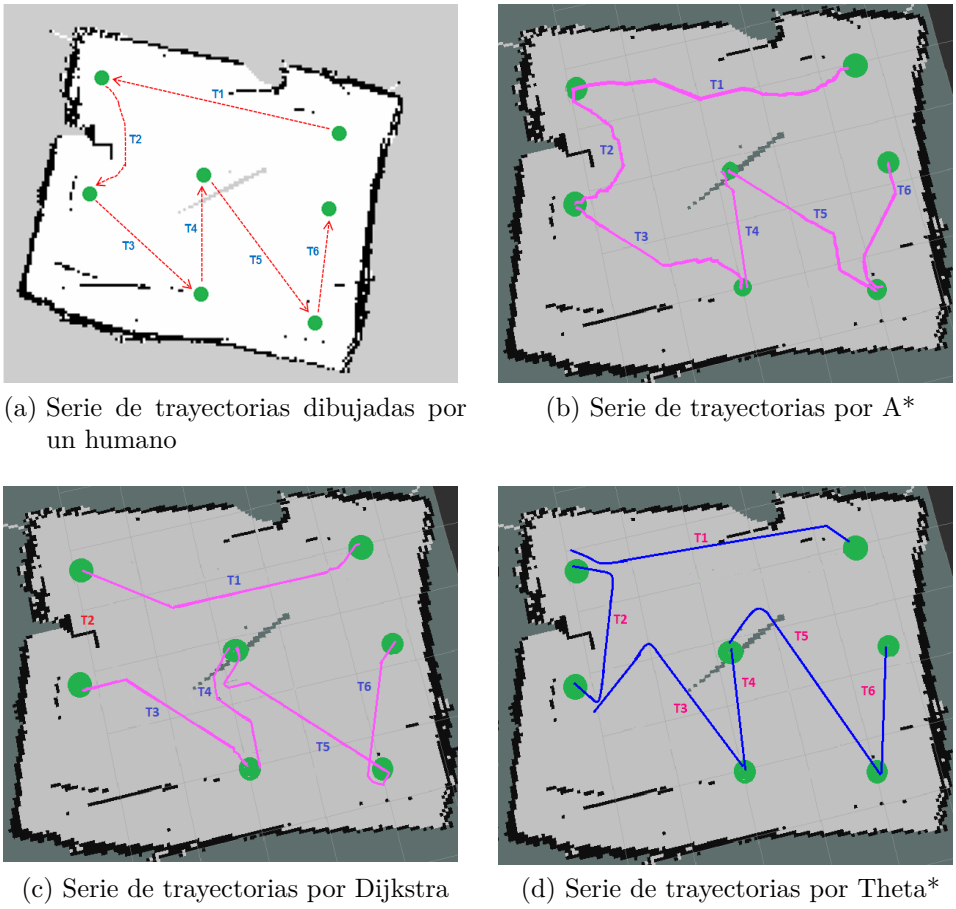
(b) Serie de trayectorias por A*

(c) Serie de trayectorias por Dijkstra

(d) Serie de trayectorias por Theta*

Figura 5.25: Comparación de trayectorias en el mapa planta baja, humavo vs. algoritmos.

El conjunto de trayectorias generadas por los algoritmos Dijkstra, A* y Theta* en el mapa aula 3103 del DCC, se muestran en la Figura 5.26.



(a) Serie de trayectorias dibujadas por un humano

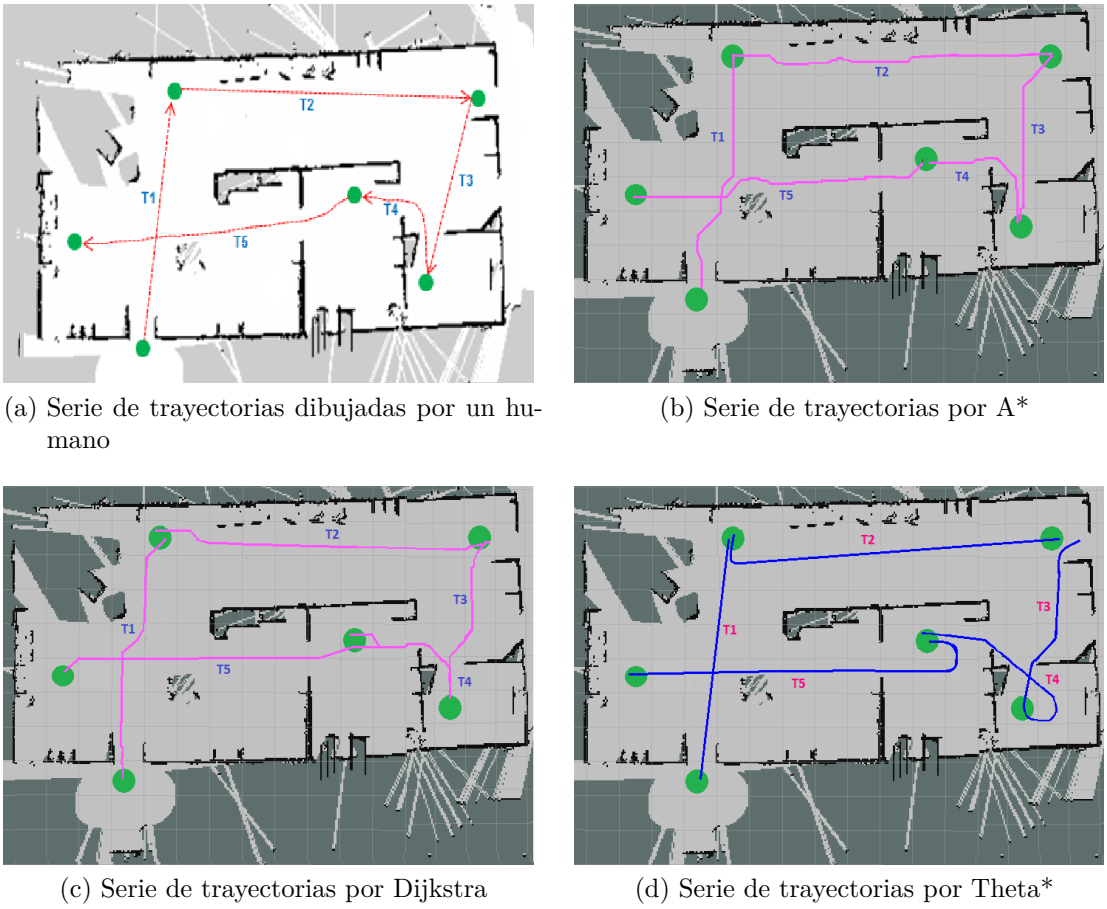
(b) Serie de trayectorias por A*

(c) Serie de trayectorias por Dijkstra

(d) Serie de trayectorias por Theta*

Figura 5.26: Comparación de trayectorias en el mapa aula 3103, humavo vs. algoritmos.

El conjunto de trayectorias generadas por los algoritmos Dijkstra, A* y Theta* en el mapa arena_a, se muestran en la Figura 5.27.



(a) Serie de trayectorias dibujadas por un humano

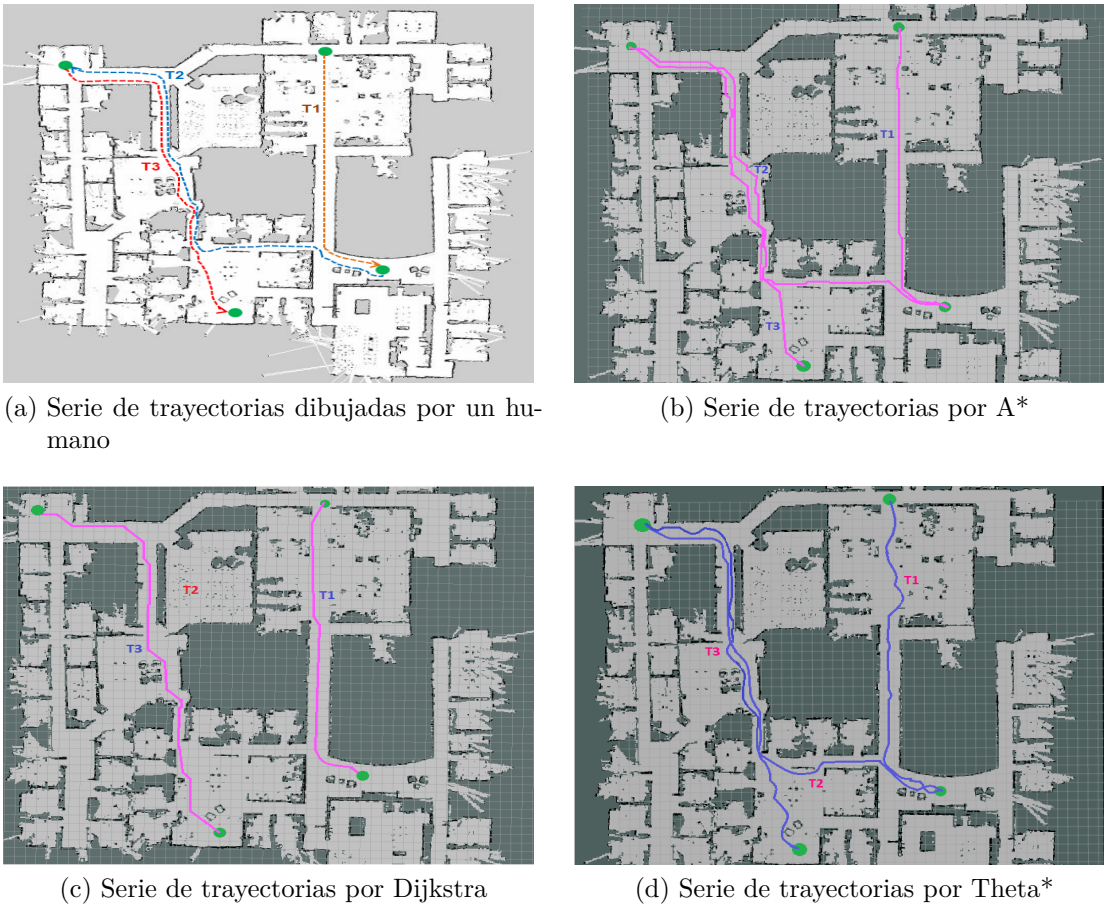
(b) Serie de trayectorias por A*

(c) Serie de trayectorias por Dijkstra

(d) Serie de trayectorias por Theta*

Figura 5.27: Comparación de trayectorias en el mapa arena_a, humavo vs. algoritmos.

El conjunto de trayectorias generadas por los algoritmos Dijkstra, A* y Theta* en el mapa willow_garage, se muestran en la Figura 5.28.



(a) Serie de trayectorias dibujadas por un humano

(b) Serie de trayectorias por A*

(c) Serie de trayectorias por Dijkstra

(d) Serie de trayectorias por Theta*

Figura 5.28: Comparación de trayectorias en el mapa willow_garage, humavo vs. algoritmos.

5.6. Conclusiones

En este capítulo, se mostraron las pruebas que se realizaron utilizando los algoritmos Dijkstra, A* y Theta* por medio de la simulación en cinco diferentes mapas. Para ello, se diseñaron trayectorias que los algoritmos debían generar para poder ser evaluados, con respecto al tiempo y distancia.

Con respecto a los resultados obtenidos se realizó un análisis en tabas y gráficas, dando como resultado que el algoritmo Theta* fue el algoritmo que generó la trayectoria y llegó a la meta en el 100 % de los casos en un tiempo de ejecución aceptable y los metros recorridos y generados fueron menos a los otros dos algoritmos. En cuanto a los mapas, recomiendo todos los algoritmos excepto willow_garage para poner a prueba cualquier

algoritmo a nivel simulación, debido a que este mapa no cuenta con todos los bordes cerrados y pasillos angostos que dificultan la tarea de planeación de trayectorias.

En el próximo capítulo, se presentarán las conclusiones del presente trabajo de investigación y los trabajos futuros.

Capítulo 6. Conclusiones

El algoritmo Theta* al tener sus bases de búsqueda en el algoritmo A* no perdió las propiedades de simplicidad, al ser fácil de entender y programar para implementarlo en ROS, la eficiencia se demostró cuando el algoritmo generaba la trayectoria y las recorría en la mayoría de los casos en menos metros en comparación con los otros algoritmos y la generalidad al encontrar una trayectoria en el 100 % de los casos.

De acuerdo a las pruebas y resultados presentados en el capítulo anterior, se comprobó que la hipótesis planteada para este trabajo es verdadera:

- Incrementar el número de vértices de búsqueda a ocho en el algoritmo de planeación de trayectorias, puede encontrar la ruta más corta y directa al generar búsquedas en diagonal, mejorando el rendimiento del robot al navegar en un entorno conocido estático.

En la evaluación del desempeño de los algoritmos utilizando la simulación de las trayectorias con un robot con configuración diferencial, con dos versiones de ROS diferentes, ROS Indigo en Ubuntu 14.04 para ejecutar los algoritmos Dijkstra y A* y ROS Kinect en Ubuntu 16.04 para implementar y ejecutar el algoritmo Theta*. Cada uno de los 5 mapas contaba con un número de trayectorias diferentes, las cuales se diseñaron en base al tamaño y complejidad del mapa, danto un total de 27 trayectorias, con la finalidad de que los algoritmos realizarán la tareas de planificación y seguimiento de trayectorias. A continuación a manera de resumen se mostrara de forma general el desempeño de cada algoritmo.

En la evaluación del desempeño de los tres algoritmos utilizados para realizar la tarea de planificación y seguimiento de trayectorias a nivel simulación, con un robot de configuración diferencial, en mapas de entornos estáticos, se observó que el algoritmo Theta* fue el algoritmo con el mejor desempeño, ya que generó las 27 trayectorias y llegó a la meta en todas las ocasiones, además, en casi todas las trayectorias los metros que generaba eran los mismos metros que recorría, a pesar de que en algunas trayectorias realizaba algunos giros innecesarios los metros de diferencia entre la distancia generada y la distancia recorrida no eran mayores a los 5 metros. A*, por su parte, tuvo un desempeño medio generando 26 trayectorias y llegando a la meta en 23 ocasiones. Dijkstra, fue el algoritmo con el peor desempeño, teniendo algunos problemas para generar las trayectorias en mapas con obstáculos como el de mapa alta y willow_garage, generando 23 trayectorias y llegando a la meta en 17 ocasiones.

Los algoritmos Dijkstra y A* presentaron problemas para llegar a la meta a pesar de que generaban la trayectoria, esto se puede atribuir a los pequeños cambios de dirección que se observaban en las trayectorias, ya que estos giros provocaban que el robot perdiera su localización haciendo que se saliera de los límites del mapa, donde los algoritmos no pueden generar trayectorias, sí el mapa no volvía a los límites del mapa para recuperar la trayectoria ROS detenía el proceso.

6.1. Objetivos y alcances logrados

En la Tabla 6.1, se muestran las actividades realizadas en relación a los objetivos y alcances planteados para el desarrollo de la tesis.

Tabla 6.1: Tabla de objetivos y alcances.

Objetivos general	**Actividades**
Evaluar el algoritmo Theta* en la tarea de planeación de trayectorias utilizando ocho vértices de búsqueda.	Se estudió el algoritmo y se implementó en ROS, para finalmente compararlo y evaluarlo con los algoritmos Dijkstra y A* proporcionados por el paquete *global_planner* de ROS.
Objetivos específicos	**Actividades**
Conocer y comprender los conceptos básicos de la robótica móvil.	De los documentos estudiados en el estado del arte, se analizaron los relacionados a la robótica móvil y las tareas involucradas dentro de la navegación autónoma.
Estudiar los algoritmos recientes de planeación de trayectorias.	Se realizó una búsqueda de más de 20 artículos y documentos para encontrar los algoritmos más recientes para la planeación de trayectorias dentro de la robótica.
Comprender el funcionamiento del algoritmo de planeación de trayectorias Theta*.	Se estudió el algoritmo a fondo con ayuda de artículos que explican sus ventajas y su función.
Comprender el funcionamiento de ROS (Sistema Operativo Robótico).	Se revisó el foro de ROS, se realizaron tutoriales y se tomaron clases extras para poder entender el manejo del sistema operativo y sus versiones.
Implementar el algoritmo Theta* en ROS.	Se desarrolló el algoritmo en el lenguaje C++ y se implementó el algoritmo en una versión de ROS que otorgaba más permisos que la versión que se estaba utilizando.
Validar la eficacia y generalidad del algoritmo Theta* implementado en ROS.	Para las pruebas, se trabajó con cinco entornos descritos en la sección 5.2 y se realizaron simulaciones como se puede ver a partir de la subsección 5.4.1, para validar los algoritmos se utilizaron las métricas descritas en la sección 5.3.

Tabla 6.1: (Continuación) Tabla de objetivos y alcances.

Alcances	Actividades
Analizar cinco de los algoritmos tradicionales para la planificación de trayectorias.	Se analizaron algoritmos tradicionales (Dijkstra y A*) y algoritmos que hicieran suavizado en las trayectorias (A* post-suavizado), así como algoritmos derivados.
Analizar principalmente el algoritmo de planificación de trayectorias A* disponible en ROS.	Como se comentó en el capítulo 4, ROS trae en su paquete de navegación el algoritmo Dijkstra y A*, la finalidad de analizar A* principalmente fue con el propósito de saber sí se podría ocupar el código para implementar e algoritmo Theta*.
Las trayectorias serán cortas y en tiempo de ejecución aceptable.	Se demostró en las pruebas de la subsección 5.4.1, que el algoritmo Theta* generaba trayectorias en un tiempo y distancia superiores a las de A* y Dijkstra, incluso en la distancia recorrida era ligeramente mayor o igual a la distancia que generaba.
Utilizar mapas 2D para la implementación del algoritmo.	Se utilizaron mapas desarrollados por las herramientas de ROS, con la finalidad de realizar las secuencias de trayectorias, tal como se describe en la sección 5.4 del documento.
Las pruebas de realizarán a nivel simulación, con un robot de configuración diferencial.	La generación de trayectorias se realizó en el visualizador (Rviz) de ROS, utilizando un mapa de configuración diferencial y los mapas elegidos.

6.2. Productos entregables

A continuación, se mencionan en forma de lista los entregables que se entregaron durante el desarrollo del proyecto de investigación.

- Reporte del Estado del Arte.

  Como parte de los productos entregables, se incluyó un informe del estado del arte, el cual se incluirá en el Disco Compacto anexo a la tesis, en dicho documento se describen los artículos y trabajos que se utilizaron para el estudio e implementación del algoritmo Theta*.

- Reporte del estudio de el algoritmo Theta*.

  Este reporte incluye la información del algoritmo y el código en C++, incluido en el Disco Compacto anexo a la tesis.

- Reporte de la implementación del algoritmo Theta* en ROS.

  Tal como se describe en la subsección 4.1.2, la cual detalla el proceso a seguir para implementar el algoritmo Theta* en el ROS Kinect y Ubuntu versión 16.04.

- Simulaciones del algoritmo.

  Se incluirán imágenes presentadas en la sección 5.4 y videos de la ejecución de cada algoritmo en la versión del sistema ROS correspondiente en un Disco Compacto anexo a la tesis.

- Artículo de los resultados obtenidos.
- Presentación de resultados.
- Documento de tesis.

6.3. Aportaciones

1. Se implementó un nuevo algoritmo de planificación de trayectorias en el sistema ROS Kinect.
2. Se desarrolló una guía para implementar y ejecutar los algoritmos Dijkstra y A*, para Ubuntu 14.04 y ROS Indigo.
3. Se probó la eficiencia de los mapas elaborados por (Lopéz, 2017).
4. Se tiene el conocimiento y la serie de pasos para implementar estos tres algoritmos en un robot de configuración diferencial real.

6.4. Trabajos futuros

El trabajo futuro de esta investigación, después de la información y conocimientos obtenidos se pueden realizar las siguientes actividades:

1. Implementar en un robot móvil con configuración diferencial o bien omnidireccional, ya que se cuenta con ellas en la línea de investigación y no se implementó debido a situaciones ajenas a la institución.
2. Hacer las adecuaciones requeridas para que el algoritmo pueda trabajar en 3D, y evaluarlo en la realidad con un vehículo aéreo no tripulado.
3. Incrementar el número de vértices de búsqueda significativamente, en el algoritmo Theta* de tal forma que se pueda generar trayectorias suavizadas.

Bibliografía

Aguado Aranda, A. y Jiménez de Vega, J. (2013) Optimización de rutas de transporte. Facultad de Informática U.C.M, Madrid.

Azkune Gorka (2011). Navegación autónoma. Recuperado de https://cuentos-cuanticos.com/2011/11/12/navegacion-autonoma/

Barraquand, J., Langlois, B., and Latombe, J. C. (1992). *Numerical potential field techniques for robot path planning.* IEEE Transactions on Systems, Man, and Cybernetics, 22(2), 224-241.

Brooks, R. A. (1983). *Solving the find-path problem by good representation of free space.* IEEE Transactions on Systems, Man, and Cybernetics, (2), 190-197.

Camarena García, J. F. (2009). Análisis cinemático, dinámico y control en tiempo real de un vehículo guiado automáticamente. (Tesis de Maestría, Centro Nacional de Investigación y Desarrollo Tecnológico, Departamento de Mecatrónica).

Colorni, A. (1992). *Distributed optimization by ant colonies.* Proceedings of the rst European Conference on Artificial Life (ECAL 91).

Dijkstra, E. W., and Scholten, C. S. (1980). *Termination detection for diffusing computations.* Information Processing Letters, 11(1), 1-4.

Duchoň, F., Hubinský, P., Babinec, A., Fico, T., and Huňady, D. (2014). *Real-time path planning for the robot in known environment.* In Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on (pp. 1-8). IEEE.

Escamilla, A. C. (2013). Planeación de Trayectorias y Control por Platitud Diferencial de Múltiples Robots Móviles. (Tesis de Maestría, Centro Nacional de Investigación y Desarrollo Tecnológico, Departamento de Ingeniería Electrónica).

Espitia Cuchango H. E., (2011). Propuesta de un algoritmo para la planeación de trayectorias de robots móviles empleando campos potenciales y enjambres de partículas activas brownianas. (Tesis de Maestría, Universidad Nacional de Colombia).

Felner, A. (2011, July). *Position paper: Dijkstra's algorithm versus uniform cost search or a case against Dijkstra's algorithm.* In Fourth Annual Symposium on Combinatorial Search.

Fletcher, L., Teller, S., Olson, E., Moore, D., Kuwata, Y., How, J., and Nathan, A. (2008). *The MIT–Cornell collision and why it happened.* Journal of Field Robotics, 25(10), 775-807.

Flores Méndez, J. (2015). Robot móvil omnidireccional para SLAM y navegación en ROS. (Tesis de Maestría, Universidad Nacional Autónoma de México, Facultad de Ingeniría Mécanica-Mecatrónica).

Franco Gómez, J. A. (2011). Un algoritmo basado en la optimización por enjambre de partículas para el problema de asignación axial 3-dimensional. (Tesis de Maestría, Instituto Tecnológico De La Paz).

Ganeshmurthy, M. S., and Suresh, G. R. (2015, March). *Path planning algorithm for autonomous mobile robot in dynamic environment.* In Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on(pp. 1-6). IEEE.

GitBook, E. R. (2017). Erle Robotics GitBook. San Francisco, California, Estados Unidos.

Goyal, J. K., and Nagla, K. S. (2014). *A new approach of path planning for mobile robots.* In Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on (pp. 863-867). IEEE.

Hurtado, J. E., and Alvarez, D. (2002). Optimización basada en confiabilidad p or medio de redes neuronales y algoritmos evolutivos. Revista internacional de métodos numéricos.

Latombe, J. C. (1991). Robot Motion Planning. Kluwer Academic Publishers.

Lindemann, S. R., and LaValle, S. M. (2006). *Steps toward derandomizing RRTs.* In Robot Motion and Control (pp. 287-300). Springer, London.

Lopéz Borreguero, D. (2017). Evaluación de técnicas SLAM disponibles en ROS. (Tesis de Maestría, Centro Nacional de Investigación y Desarrollo Tecnológico, Departamento de Computación)

Martínez, A., y Fernández, E. (2013). *Learning ROS for robotics programming.* Packt Publishing Ltd.

Martínez Díaz, M. (1997). Algoritmos de construcción de mapas para la navegación de robots, basada en información procedente de sensores de ultrasonido. (Tesis de Doctorado), Universitat Politècnica de València.

Martínez Ramírez, F. (2012). Control embebido de un vehículo guiado automáticamente mediante Redes Neuronales Artificiales. (Tesis de Maestría, Centro Nacional de Investigación y Desarrollo Tecnológico, Departamento de Electrónica).

Méndez Navarro, A. (2015). Red científica. Recuperado el Junio de 2017, de Red científica (Ciencia, tecnología y pensamiento): http://www.redcientifica.org/primeros_pasos_hacia_la_navegacion_autonoma_en_robots_moviles.php

Mendonca, P., and Goodwin, S. (2015). *C-Theta*: Cluster based Path-Planning on Grids.* In 2015 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 605-608). IEEE.

Moravec, H., and Elfes, A. (1985). High resolution maps from wide angle sonar. In Proceedings. 1985 IEEE international conference on robotics and automation (Vol. 2, pp. 116-121). IEEE.

Muntean, P. (2016) *Mobile Robot Navigation on Partially Known Maps using a Fast A* Algorithm Version.*

Nash, A. (2012). Any-Angle Path Planning (Doctoral dissertation, University of Southern California).

Nash, A., Daniel, K., Koenig, S., and Felner, A. (2007, July). *Theta*: Any-Angle Path Planning on Grids.* In AAAI (pp. 1177-1183).

Nilsson, N. J. (1969). *A mobile automaton: An application of artificial intelligence techniques.* SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER.

Palmieri, L., Koenig, S., and Arras, K. O. (2016). *RRT-Based Nonholonomic Motion Planning Using Any-Angle Path Biasing.* In 2016 International Conference on Robotics and Automation (ICRA) (pp. 2775-2781). IEEE.

Peñaloza Membrila, K. M. (2014). Navegación visual en trayectorias cerradas. (Tesis de Maestría, Centro Nacional de Investigación y Desarrollo Tecnológico, Departamento de Computación).

ROS (2016a). Recuperado de http://www.ros.org

ROS (2016b). Recuperado de http://wiki.ros.org

Tovey, J. B. C., Koenig, T. U. S., and Nash, A. (2015). *Path Planning on Grids: The Effect of Vertex Placement on Path Length*.*

UNAM. (2017). Manual de usuario del robot Justina. (Posgrado en ingeniería eléctrica, Laboratorio «Biorobotics»).

Uras, T., and Koenig, S. (2015a). *An empirical comparison of any-angle path-planning algorithms.* In Eighth Annual Symposium on Combinatorial Search.

Uras, T., and Koenig, S. (2015b). *Speeding-Up Any-Angle Path-Planning on Grids.* In ICAPS (pp. 234-238).

Vergara Bahena, A. (2012). Navegación, localización y mapeo de robots móviles para trayectorias pre-especificadas por imágenes. (Tesis de Maestría, Centro Nacional de Investigación y Desarrollo Tecnológico, Departamento de Computación).